60485200

CONTROL DATA

DMS-170

CYBER DATABASE CONTROL SYSTEM VERSION 2 DATA ADMINISTRATION REFERENCE MANUAL

CDC® OPERATING SYSTEMS:
NOS 2
NOS/BE 1

REVISION RECORD

Revision	Description
A (05/14/82)	Initial release under NOS 2 and NOS/BE 1; PSR level 564. This manual represents a complete reorganization according to audience of documentation for CDCS, DDL, and FDBF (with FDBF separated into components of DDL and DML). This manual contains documentation intended for a data administrator, and supersedes the following manuals for users of the NOS 2 and NOS/BE operating systems: the CDCS 2 reference manual (Pub. No. 60491800); the DDL Version 3 reference manuals, volumes 1 and 2 (Pub. No. 60481900 and 60482000); and the FDBF 1 reference manual (Pub. No. 60482200). New features of CDCS 2.3 are also documented in this revision; the features include automatic recovery, data base transaction processing, data base versions, the basic recovery utility (DBREC), and extensions to the master directory and to the operator interface.
B (08/26/83)	Released at PSR level 587. This revision documents improved duration loading capabilities of CDCS and miscellaneous technical corrections.
C (09/22/83)	Released at PSR level 596. This revision documents a change to the procedure that initializes CDCS under NOS 2.2.
D (02/20/84)	Released at PSR level 599. This revision documents support of concatenated keys for the FORTRAN interface and includes miscellaneous technical corrections.
E (05/13/86)	Released at PSR level 647. This revision removes references to FORTRAN 4 and incorporates miscellaneous technical and editorial changes.

REVISION LETTERS I, O, Q, AND X ARE NOT USED

Address comments concerning this manual to:

CONTROL DATA CORPORATION
Publications and Graphics Division
P.O. Box 3492
SUNNYVALE, CALIFORNIA 94088-3492

or use Comment Sheet in the back of this manual

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

Page	Revision
Front Cover	-
Title Page	-
ii	E
iii/iv	E
v	D
vi	A
vii	B
viii thru xii	E
xiii	A
1-1 thru 1-3	A
1-4	E
1-5 thru 1-12	A
2-1 thru 2-21	A
2-22	B
2-23	D
2-24 thru 2-30	A
2-31	B
2-32	B
2-33	A
2-34	A
2-35	B
2-36	B
2-37	A
3-1 thru 3-8	A
3-9 thru 3-11	B
3-12 thru 3-33	A
3-34	E
3-35 thru 3-37	A
3-38	B
3-39	A
4-1 thru 4-20	E
5-1	A
5-2 thru 5-4	E
6-1	A
6-2	A
6-3	D
6-4	D
6-4.1/6-4.2	D
6-5 thru 6-18	A
7-1 thru 7-9	A
8-1	A
8-2	D
8-3	E
8-4	E
8-4.1/8-4.2	E
8-5	E
8-6	D
8-7 thru 8-21	A
9-1 thru 9-3	A
9-4	E
9-4.1/9-4.2	E
9-6	A
9-7	A
9-8	E
9-8.1/9-8.2	E
9-9	A
9-10	A
9-11	E
9-12	A
9-13	A
9-14	E
9-15	A
9-16	A
9-17	E
9-18	B
9-19 thru 9-27	A
10-1 thru 10-3	A
10-4 thru 10-8	E
10-8.1	E
10-8.2	E
10-9	C
10-10 thru 10-17	A
11-1	E
11-2 thru 11-5	A
11-6 thru 11-11	E
A-1 thru A-4	A
B-1	A
B-2	A
B-3	E
B-4	E
B-4.1/B-4.2	E
B-5	A
B-6	B
B-7	A
B-8	E
B-9	B
B-10 thru B-16	E
B-16.1	E
B-16.2	E
B-17 thru B-22	A
B-23	B
B-24	C
B-24.1/B-24.2	C
B-25 thru B-54	A
B-55	C
B-56 thru B-60	A
B-61	C
B-62 thru B-97	A
B-98	D
B-99	A
B-100	A
C-1	A
C-2	A
C-3 thru C-5	B
C-6 thru C-9	A
D-1 thru D-3	A
E-1	A
E-2	A
E-3	B
E-4	B
E-5	A
E-6	A
E-7 thru E-9	E
E-10 thru E-13	A
F-1	A
F-2	A
G-1	A
G-2	E
G-3	E
H-1 thru H-4	A
H-5	E
H-6	A
I-1	A
I-2	A
J-1	A
J-2	E
J-3	E
Index-1 thru Index-10	E
Comment Sheet/Mailer	E
Back Cover	-

PREFACE

This manual describes both the definition phase of a DMS-170 data base and the execution phase of CONTROL DATA® CYBER Database Control System (CDCS) Version 2. A data base is defined by the use of the Data Description Language (DDL) and is controlled by CDCS, which monitors all access by application programs. Several products are involved in the definition and execution phases; these products are CDCS Version 2.3, DDL Version 3.2, and the DDL portion of the FORTRAN Data Base Facility (FDBF) Version 1.3. As described in this publication, these products operate under control of the following operating systems:

NOS/2 for the CONTROL DATA CYBER 180 Computer Systems; CYBER 170 Computer Systems; CYBER 70 Computer System Models 71, 72, 73, and 74; and 6000 Computer Systems

NOS/BE 1 for the CDC® CYBER 180 Computer Systems; CYBER 170 Computer Systems; CYBER 70 Computer System Models 71, 72, 73, and 74; and 6000 Computer Systems

This manual is designed for the data administrator. The data administrator is the programming or managerial person or group responsible for defining, creating, controlling, and monitoring data bases. It is assumed that the data administrator is knowledgeable in both systems and application programming, has some familiarity with data management concepts and terminology, and has used previously Control Data computers and software.

Detailed information for application programmers using CDCS is contained in the CDCS 2 Application Programming reference manual. Related material is contained in the publications listed below.

The NOS Manual Abstracts and the NOS/BE Manual Abstracts are instant-sized manuals containing brief descriptions of the contents and intended audience of all NOS and NOS product set manuals, and NOS/BE and NOS/BE product set manuals, respectively. The abstracts manuals can be useful in determining which manuals are of greatest interest to a particular user. The Software Publications Release History serves as a guide in determining which revision level of software documentation corresponds to the Programming System Report (PSR) level of installed site software. The abstracts manuals are included in the list of publications of secondary interest.

Manuals are listed alphabetically within groupings that indicate relative importance to readers of this manual.

The following manuals are of primary interest:

Publication	Publication Number
CYBER Record Manager Advanced Access Methods Version 2 Reference Manual	60499300
DMS-170 CYBER Database Control System Version 2 Application Programming Reference Manual	60485300

The following manuals are of secondary interest:

Publication	Publication Number
CYBER Loader Version 1 Reference Manual	60429800
CYBER Record Manager Basic Access Methods Version 1.5 Reference Manual	60495700
FORM Version 1 Reference Manual	60496200
Networks Products Transaction Facility Version 1 Reference Manual	60459500
NOS Version 2 Manual Abstracts	60485500

NOS Version 2 Reference Set, Volume 3 System Commands	60459680
NOS/BE Version 1 Manual Abstracts	84000470
NOS/BE Version 1 Reference Manual	60493800
Software Publications Release History	60481000

CDC manuals can be ordered from Control Data Corporation, Literature and Distribution Services, 308 North Dale Street, St. Paul, Minnesota 55103.

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or parameters.

CONTENTS

APPENDIXES

INDEX

FIGURES

TABLES

NOTATIONS

Reference formats are presented throughout the manual to illustrate essential elements of syntax. The notations used in the reference formats follow two conventions: the COBOL convention and the FORTRAN convention. The COBOL convention is used in formats that describe syntax for the schema, COBOL and Query Update subschemas, master directory, and data base recovery utilities. The FORTRAN convention is used in formats that describe syntax for the FORTRAN subschema and for all control statements. The differences in the conventions are in the interpretation of uppercase words, the omission of underlined uppercase words from the FORTRAN convention, and the use of punctuation.

NOTATION USED IN REFERENCE FORMATS

UPPERCASE — COBOL convention. Uppercase words are reserved words and must appear exactly as shown. Reserved words can be used only as specified in the reference formats. If not underlined, they are optional.

FORTRAN convention. Uppercase words are keywords and must appear exactly as shown. Keywords can be used only as specified in the reference formats.

UNDERLINED UPPERCASE — COBOL convention. Underlined uppercase words are required when the format in which they appear is used.

FORTRAN convention. Underlined uppercase words are not used.

Lowercase — Lowercase words are generic terms that represent the words or symbols supplied by the user. When generic terms are repeated in a format, a number is appended to the term for identification.

[ ] — Brackets enclose optional portions of a reference format. All of the format within the brackets can be omitted or included at the user's option. If items are stacked vertically within brackets, only one of the stacked items can be used.

{ } — Braces enclose one item or several vertically stacked items in a reference format. When one item is enclosed in braces and followed by ellipsis, the item can be repeated at the user's option. When several items are enclosed in braces, one of the enclosed items must be used.

‖ ‖ — Vertical bars enclose two or more vertically stacked items in a reference format when at least one of the enclosed items must be used. Each of the vertically stacked items can be used once.

... — Ellipses immediately follow a pair of brackets or braces to indicate that the enclosed material can be repeated at the user's option.

Punctuation use differs for the conventions as follows:

COBOL convention

Punctuation symbols shown within the formats are required unless enclosed in brackets and specifically noted as optional. In general, commas and semicolons are optional. One or more spaces separate the elements in a reference format.

FORTRAN convention

Punctuation symbols shown within formats are required unless enclosed in brackets and specifically noted as optional.

Numbers shown within formats are decimal unless otherwise specified.

NOTATION USED IN EXAMPLES

↑ indicates the position of an assumed decimal point in an item.

A plus or minus sign above a numeric character ($\overset{+}{n}$) indicates an operational sign is stored in combination with the numeric character.

Character positions in storage are shown by boxes.

A	B	C	D

Δ indicates a space (blank).

INTRODUCTION TO DATA BASE PROCESSING WITH DMS-170 1

The DMS-170 software package functions as a data management system for Control Data computer systems. Through this data management system, a data base can be defined, maintained, and controlled in an environment independent of the applications that are accessing it. Conventional files otherwise owned and processed by a number of distinct applications can be described through the data description language facilities of DMS-170. Consequently, the responsibility for tasks such as data description, data conversion, and validity checking is transferred from the application programmer to the data administrator.

The DMS-170 data management system is composed of the following elements:

Data Description Language (DDL), which creates the schema definition, as well as the COBOL, FORTRAN and Query Update subschema definitions.

CDC CYBER Database Control System (CDCS), which controls, monitors, and interprets data base requests from COBOL, FORTRAN, and Query Update application programs.

CDC CYBER Record Manager (CRM), which handles all input/output processing requests on a data base from an application program.

Data Manipulation Languages (DML), which provide for data base access through the COBOL and FORTRAN programming languages. The COBOL DML consists of features within the COBOL language. The FORTRAN DML consists of DML statements that are used within a FORTRAN-coded program and processed by a DML preprocessor before FORTRAN compilation.

Query Update language, which provides for data base access in both interactive and batch modes. Query Update is a language that enables individuals with varying levels of technical expertise to access and manipulate the data base and to produce special-purpose reports.

Each element of the DMS-170 system is used either in the definition or in the processing of a data base. The definition of the data base is accomplished through the capabilities of DDL and the master directory utility. Processing of the data base involves retrieval and updating of the data by application programs through the facilities of CDCS.

The DMS-170 environment defines two roles: the data administrator and the application programmer. The data administrator is responsible for the definition of a data base. The data administrator is a person or group of persons who develop and define the data base as well as monitor and control the day-to-day processing of that data base. The application programmer uses interface capabilities of COBOL, FORTRAN, and Query Update in developing applications for data base processing.

CDC offers guidelines for the use of the software described in this manual. These guidelines appear in appendix F. Before using the software described in this manual, the reader is strongly urged to review this appendix. The guidelines recommend use of this software in a manner that reduces the effort required to migrate application programs to future hardware or software systems.

DATA BASE DEFINITION

To define a data base, the data administrator uses the Data Description Language (DDL). Through this language, four types of data descriptions can be created: the schema, the COBOL subschema, the FORTRAN subschema, and the Query Update subschema. Each of these data descriptions follows specific structuring conventions, includes unique clauses and statements, and conforms to an individual set of rules. Once the schema and COBOL, FORTRAN, and Query Update subschema descriptions have been created and compiled, the data administrator creates the master directory through one of the data base utility routines provided as a part of CDCS.

The relationship of the elements involved in defining a data base is shown in figure 1-1. The figure indicates that the schema directory must be available to the DDL compiler to generate subschema directories. Both the schema directory and subschema directories must be available to generate the master directory.

The data descriptions in the schema and subschema are organized into file-like structures. By convention, the following terms are used when referring to these structures:

Area in the schema

Realm in the subschema

The schema description of an area applies to all data within the structure. The subschema definition of a realm usually applies to a portion of data within the structure but can apply to all the data. The area provides the actual description of the data. The realm provides the description of data from the viewpoint of the application programmer.

The term file is used in this manual to refer to an area or realm.

SCHEMA DEFINITION

The schema is a detailed English-like description of the data in a data base. A user site can have many data bases, but only one schema is allowed for each data base.

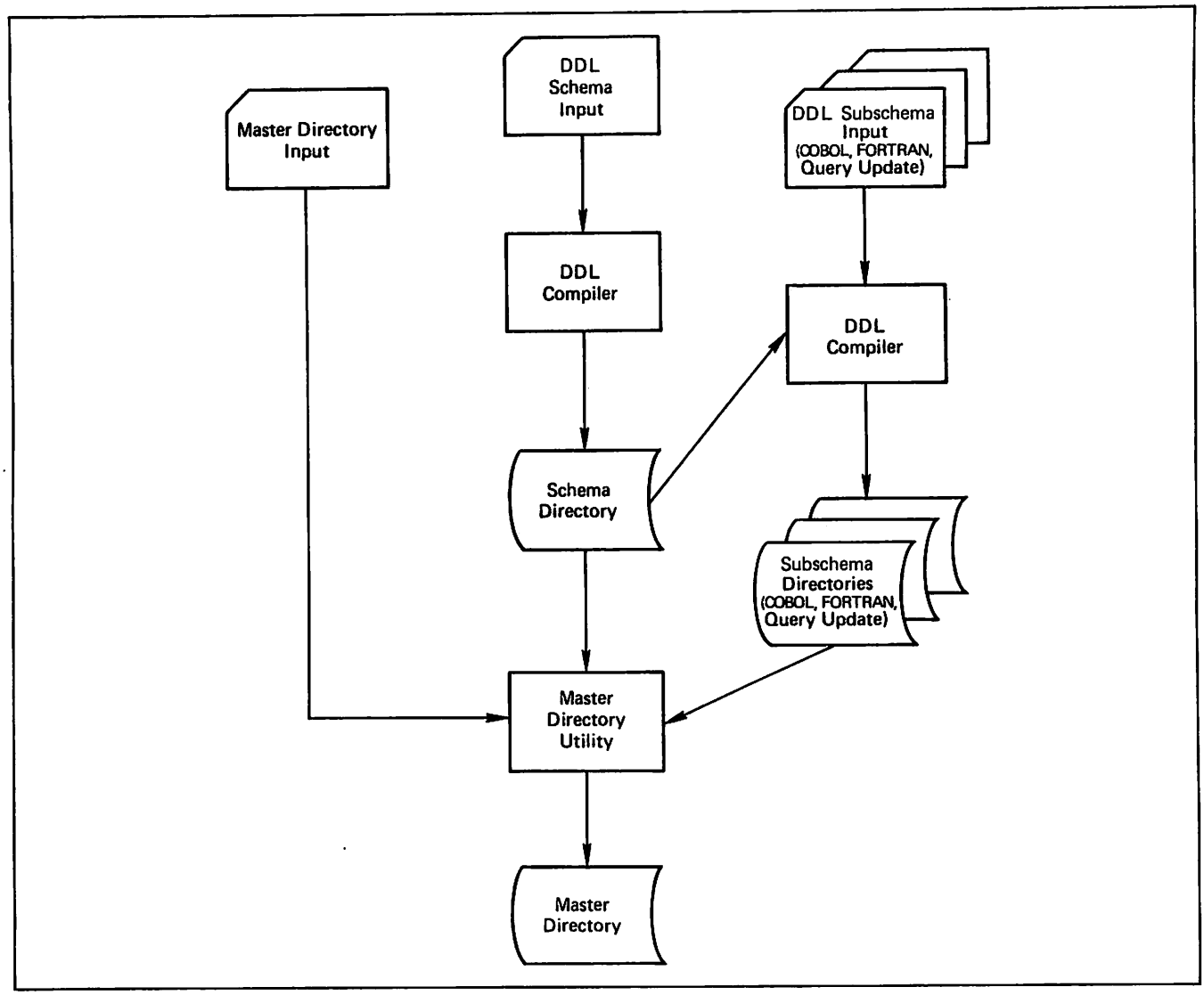


Figure 1-1. Data Base Definition

The schema provides the logical description of the data base. A schema can be associated and used with several physical data bases. A physical data base is a set of files associated with the areas of the schema. This association is described later in the Master Directory Definition subsection and in the Data Base Versions subsection.

The schema description is created by DDL statements that name the schema, organize the schema into areas, describe each record type with characteristics of the data, and describe relationships and constraints among areas. The schema also includes access control locks that provide area privacy. The DDL source statements describing the data are used as input to the DDL compiler and are compiled into an object schema, or schema directory. The data administrator then uses the schema to define any number of subschemas.

SUBSCHEMA DEFINITIONS

A subschema is a detailed description of selected portions of data described by a schema. The subschema defines the portion of the data base available to the application program; the application program uses the subschema descriptions to access the data base. The subschema is based on the schema.

The data descriptions in the subschema are organized into realms that correspond to areas in the schema. The realms included in a subschema can be a subset of the areas in the schema. The data items within the realm in a subschema can be a subset of the data items described for the corresponding area in the schema. Figure 1-2 illustrates the situation in which a subschema describes a portion of a data base.

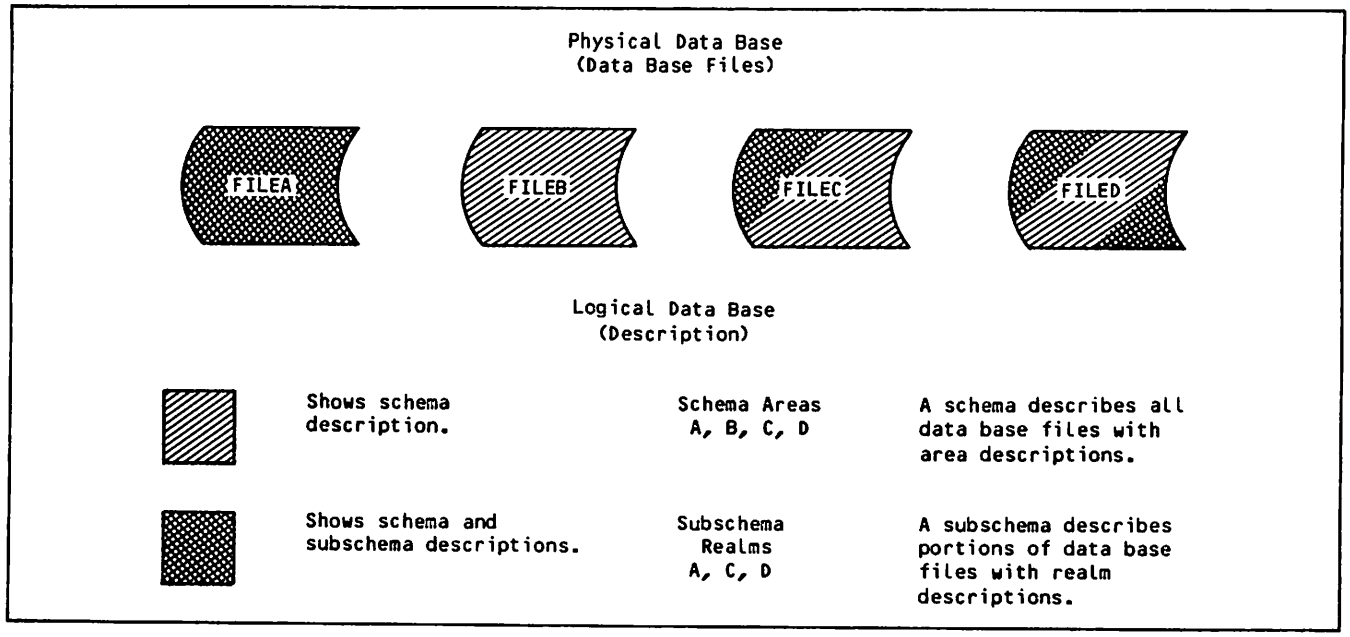


Figure 1-2. Subschema Describing a Portion of the Data Base

Although only one schema is allowed for each data base, any number of subschemas can be defined to meet the needs of different types of applications. Subschemas are defined by the data administrator for use by application programs written in the COBOL, FORTRAN, and Query Update languages.

COBOL Subschemas

A COBOL subschema is defined through the capabilities of the DDL language. COBOL subschemas describe in COBOL-like syntax the parts of a data base that can be accessed by a COBOL program. Data descriptions in COBOL subschema source statements are written to correspond to data descriptions in the schema. Certain differences are allowed to exist; these differences are resolved by DDL and CDCS. The COBOL subschema is generated by DDL source statements that identify the schema and subschema, specify realms and the content and structure of records, identify relations among realms to be used, specify record qualification for relation processing, and indicate any changes in data format required by the application program.

The DDL source statements describing the subschema are compiled by the DDL compiler into an object subschema, or COBOL subschema directory. The schema must be compiled, however, before any subschemas using it can be compiled. A COBOL programmer then uses a listing of the subschema to learn the names and descriptions of the data to be referenced in the COBOL program. Data descriptions from the subschema are automatically included in the COBOL program when it is compiled.

FORTRAN Subschemas

A FORTRAN subschema is defined through the facilities of the DDL language. FORTRAN subschemas use statements similar to FORTRAN specification statements to describe the parts of a data base that can be accessed by a FORTRAN program. Data descriptions in FORTRAN subschema source statements are written to correspond to data descriptions in the schema. Certain differences are allowed to exist; these differences are resolved by DDL and CDCS. The FORTRAN subschema is generated by DDL source statements that identify the schema and subschema, specify realms and the content and structure of records, indicate changes in data format required by the application program, identify relations among realms to be used, and specify record qualification for relation processing.

FORTRAN subschemas, like COBOL subschemas, cannot be compiled until the schema being used has been compiled. Once the schema has been compiled, the DDL source statements describing each subschema are compiled by the DDL compiler into an object subschema, or FORTRAN subschema directory. A listing of the subschema is used by the FORTRAN programmer to obtain the names and descriptions of the data to be referenced in the FORTRAN program.

Query Update Subschemas

Query Update subschemas are defined through the capabilities of the DDL language. Query Update subschemas describe in COBOL-like syntax the parts of a data base that can be accessed through Query Update directives. The data descriptions in Query Update subschemas are written to correspond to data descriptions in the schema. Certain differences between the subschema and schema data descriptions are allowed to exist; these differences are resolved by DDL and CDCS. For each subschema, the DDL source statements used as input to the DDL compiler name the schema and subschema, specify needed realms and the content and structure of records, identify relations among realms to be used, specify record qualification for relation processing, and indicate any changes in data format required by the Query Update program.

After the schema has been compiled, the DDL source statements describing the subschema are compiled by the DDL compiler into an object subschema, or subschema directory. The names and descriptions of data to be referenced in a Query Update program are obtained from a listing of the subschema.

Through facilities of DMS-170, a CDCS-controlled data base can be accessed through Query Update directly by CRM. Refer to appendix J for information about this interface. This manual discusses only Query Update data base access through CDCS.

MASTER DIRECTORY DEFINITION

The master directory must be created by the data administrator before any application programs accessing data base files can be executed. The master directory contains all information about the data base known to CDCS. This includes information about schemas and subschemas as previously described in this section. This also includes information about the following elements, which are described later in this section: data base procedure libraries, logging specifications for data base files, log files, and data base versions. In addition, the master directory functions as the source of all data base and media descriptions for CDCS.

If a data base has versions, these data base versions are defined in the master directory. A data base version is a set of permanent files for areas described by a schema. Through definition of versions in the master directory a single schema can be associated with more than one set of permanent files.

To create or update the master directory, the data administrator uses the DBMSTRD utility. Input for the utility contains information that associates subschemas, a data base procedure library, log and recovery files, and data base versions with a schema. Input can also associate areas of particular data base versions with permanent files and logging specifications. The information about permanent files that is specified in the master directory provides CDCS the information required to attach the permanent files, including the actual permanent file associated with a particular area and version, index files, log and recovery files, and procedure library files. After the master directory has been generated, it must be stored as a permanent file.

In the process of maintaining a data base environment, the data administrator might want to add information for one or more new schema definitions, delete or modify existing schema information, or modify permanent file information for data base files and procedure library files. Under any of these circumstances, appropriate changes must be made to the master directory through a modification run. A new data base cannot be accessed by application programs until the appropriate information from the corresponding schema and subschema is added to the master directory. Similarly, when information pertaining to a schema is deleted from the master directory, subschemas associated with the schema can no longer be used by application programs. One other form of modification allows the addition or deletion of information pertaining to subschemas. No subschema can be referenced by a user during execution unless information about that subschema exists in the master directory.

DATA BASE PROCESSING

Once a data base has been defined by the data administrator, it can be accessed and modified by users of the COBOL, FORTRAN, and Query Update application programming languages. The relationship of the elements involved in processing a data base is shown in figure 1-3.

The following subsections introduce the processing facilities of CDCS.

APPLICATION LANGUAGES

The data in a data base can be accessed by the following application languages: COBOL 5, FORTRAN 5, and Query Update. Processing of the data base by COBOL, FORTRAN, and Query Update programs is controlled and monitored by CDCS. These application languages can be used in either batch or interactive mode.

COBOL Processing

A COBOL program accesses data base files through conventional input/output statements. The files are opened and closed and records are read, written, deleted, and updated using the same means as for files that are not part of a data base. Relation processing is also accomplished by conventional COBOL statements. Data retrieved by the program is accessed in accordance with the way it is described in the COBOL subschema.

When a COBOL program using CDCS is to be compiled, the file containing the subschema directory must first be attached. Once the program is compiled using the subschema, it can be executed later without reattaching the subschema directory.

Execution of an input/output statement for a data base file in a COBOL program causes the COBOL object-time routines to route input/output calls to CDCS. CDCS controls all processing of data base files.

FORTRAN Processing

A FORTRAN program accesses data base files through DML statements coded within the FORTRAN program. The DML consists of FORTRAN-like statements. These statements allow the FORTRAN user to access and modify data base files.

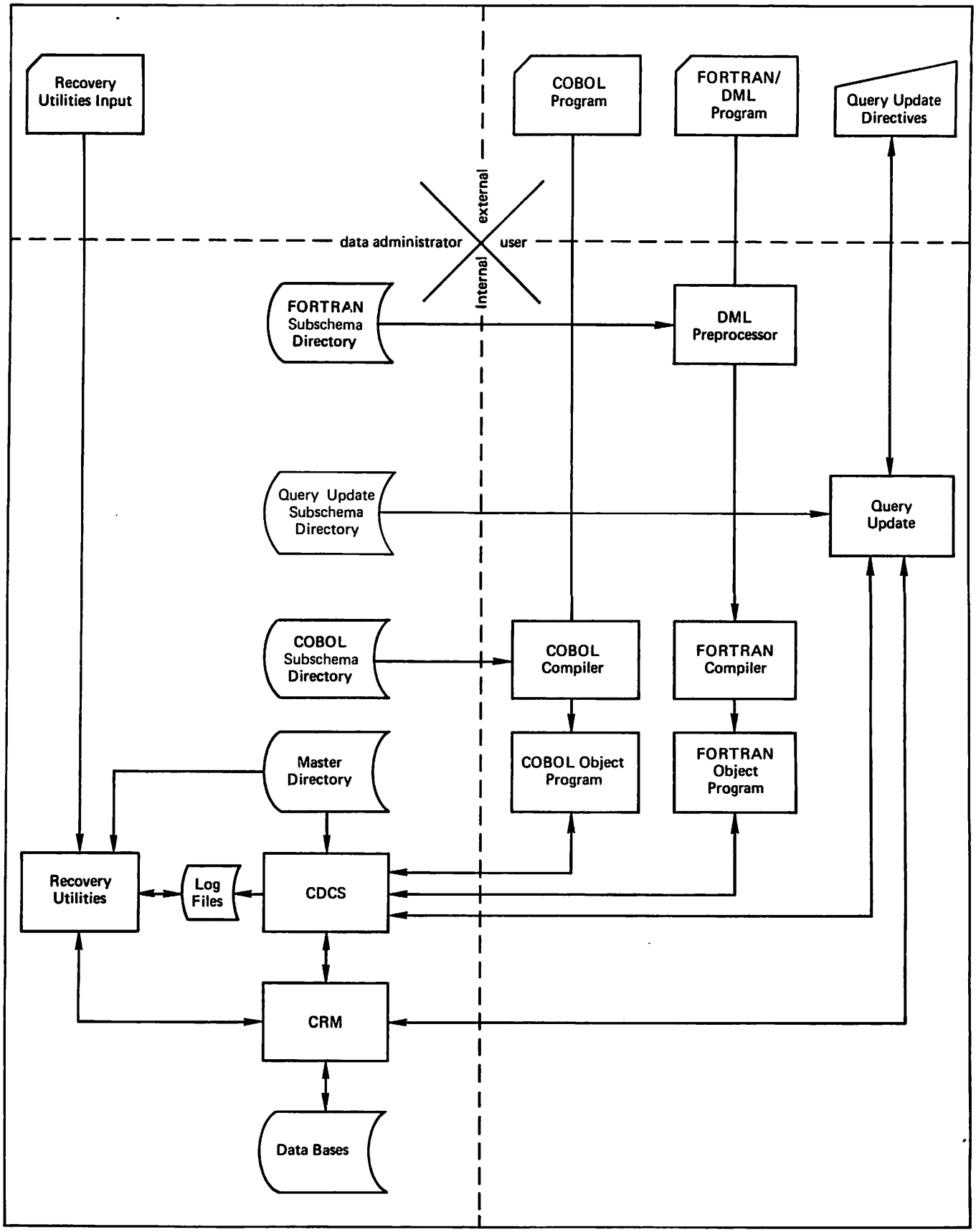


Figure 1-3. Data Base Processing With CDCS

Before a program containing FORTRAN DML statements is compiled, the DML preprocessor is called via a control statement to translate the DML statements into FORTRAN specification statements and CALL statements. Data descriptions are obtained from the FORTRAN subschema directory, which must be attached during the preprocessing phase. Following the preprocessing, compilation of the FORTRAN program proceeds as for a conventional FORTRAN program; the translated DML statements are compiled like other FORTRAN statements. Once the program is preprocessed using the subschema, it can be compiled and executed later without reattaching the subschema directory.

When a FORTRAN program using the FORTRAN DML is executed, CDCS controls all processing of data base files.

Query Update Processing

Query Update functions within the data base environment whenever a Query Update subschema is specified by a Query Update user. The Query Update language, which is a special nonprocedural, interactive language, can be used by both programmers and nonprogramming personnel to perform several functions. Through simple directives, search, retrieval, update, and display operations can be performed on data base files as well as on conventional files. In addition, a single Query Update directive can be used in relation processing to display to the user data from more than one file. A comprehensive report writing capability is an integral part of Query Update.

When a Query Update program accesses data base files, CDCS controls all processing of data base files. The concurrency, access control, logging, and recovery features of CDCS are used by Query Update.

CONCURRENCY

An important feature provided by CDCS is the concurrency feature. Concurrency means that two or more application programs can access the same data base files at the same time. Programs can access a file concurrently for retrieval or update purposes. During concurrent update operations, CDCS provides a locking mechanism by which resources (files or records) can be locked and unlocked at appropriate times. Automatic locking and unlocking are performed by CDCS when certain input/output operations are specified. In addition, explicit lock and unlock requests can be issued from an application program.

The CDCS locking mechanism provides two kinds of locks: protected and exclusive. A protected lock allows the user holding the lock to update the resource and other users to read it. An exclusive lock allows the user holding the lock to update the resource and allows no other user to access it.

A deadlock situation can occur when two programs attempt to access resources that have been locked by CDCS or by other programs. When this situation occurs, CDCS selects one of the contending programs and releases all locked resources held by that program. Appropriate code to handle recovery from a deadlock should be included in application programs.

The immediate return feature of CDCS provides COBOL and FORTRAN application programs with the ability to receive an immediate response from CDCS when either a resource conflict or a fatal error occurs. When this feature is used, CDCS returns control to the application. Normally, the application program waits for CDCS to gain access to these resources. However, when the immediate return feature is enabled, the application program can contain logic to determine the action taken in this situation.

For further information about CDCS locking, deadlock, and immediate return, refer to the CDCS 2 Application Programming reference manual.

DATA VALIDATION

Data validation based on criteria specified in the schema is performed by CDCS. Through value range checking, CDCS verifies that the value of a particular data item falls within a specified range. CDCS can also call data base procedures to perform data validation.

RECORD MAPPING

Record mapping between the schema and subschema is performed by CDCS. CDCS performs data conversions and structural reformatting so that requested data in the data base is delivered to the application program according to the format specified in the subschema.

FILE PRIVACY

Another valuable function provided by CDCS is the access control (privacy checking) mechanism. Through this mechanism, access to data base files can be controlled on the basis of criteria specified in a data base procedure or on the basis of access control locks declared in the schema.

When a data base procedure is used for privacy checking, the procedure decides whether to allow the use of a data base area. The decision is based on the access control key supplied by the application program and on the job name of the program.

When access control locks are used for privacy checking, a clause in a schema specifies the locks that apply to the use of an area. The application programs must specify the appropriate access control key (privacy key) to gain access at execution time to the data base file controlled by the lock.

RELATIONS

The relation facility of CDCS allows an application program to access data from files related by the relation definition with a single read request (a display request for Query Update). In the schema, the data administrator links areas together into a logical, meaningful relationship, called a relation, by specifying a relation entry. The relation entry assigns a name to the relation and specifies the data items to be used to link the files.

The COBOL, FORTRAN, and Query Update users access relations based upon the particular relations included in the respective subschemas. The relations in the COBOL and FORTRAN subschemas are based on the relations defined in the schema. In the subschema, relations can be qualified (called a restriction). If a restriction is defined for a relation, CDCS retrieves only records from the files joined in the relation that meet the qualification criteria.

During relation processing, data can be retrieved from each file joined in the relation. A COBOL or FORTRAN application program accesses a relation by specifying a single read request with the name of the relation that is to be read. CDCS processes the requests and returns a record occurrence from each file in the relation to the user's work area for the file. Query Update programs access relations by specifying in a DISPLAY or EXTRACT directive the data names of data defined within the areas that are joined in a relation.

CONSTRAINTS

The constraint facility of CDCS is an independent feature that provides a means of protecting the integrity of data in a data base. Use of the facility prevents the possible introduction of inconsistent data into a data base as a result of update operations by application programs on records in logically related files, or on items within a single file.

In the schema, the data administrator establishes a dependency condition between two areas or between items within an area by specifying a constraint entry. The constraint entry assigns a name to the constraint and specifies the data items involved in the dependent relationship. The areas involved in a two-area constraint must each contain a description of a common data item, which is used to define the constraint. During data base processing, a dominant record occurrence corresponds to a dependent record occurrence if both records contain the same value for the common item.

When a COBOL, FORTRAN, or Query Update application program updating a data base is executed, CDCS enforces the constraints established in the schema. A write (store), delete (remove), or rewrite (modify) request is permitted or rejected by CDCS on the basis of the effect of the proposed operation on the dependency condition in the applicable constraint.

DATA BASE VERSIONS

The data base version facility of CDCS allows an application program to use the same schema and subschema to access a specific group of files when a number of groups are defined as data base versions. Data base versions are defined by the data administrator in the master directory. If no data base versions are defined, a version named MASTER is assumed by default. If alternate data base versions are defined, one version named MASTER exists along with the other data base versions.

Data base versions are referenced in application programs by specifying the version name. By specifying different version names, an application program can perform operations on different sets of files.

Figure 1-4 illustrates the use of data base versions. When the application program specifies a version name, CDCS makes available to the program all the permanent files of that data base version that are associated with the realms included in the subschema being used. The illustration shows file M1 being shared by the versions MASTER and TEST.

Permanent files can be shared by versions in only one way: any particular permanent file defined for version MASTER can also be defined for the same area in any alternate version.

DATA BASE PROCEDURES

Data base procedures are special subprograms written by the data administrator to perform a variety of supplemental operations not otherwise performed by CDCS. The procedures are called at execution time when specific situations occur during CDCS processing. The conditions under which data base procedures are to be executed are specified in the schema. The order of execution of the procedures and the names of the data base procedures are also indicated in the schema. When the schema is compiled, an alphabetic list of the data base procedures is printed at the end of the source program listing.

Some of the functions that can be performed by data base procedures are: data validation; data conversion not supported by CDCS; calculation of values for actual or virtual data items; hashing to determine the primary key value for direct access files; compression and decompression of data; additional processing on creation, retrieval, or update of data base records; privacy checking; and special handling of error conditions detected within CDCS. The use of data base procedures to perform these functions provides a well-defined method of tailoring the CDCS system to meet the needs of a particular installation.

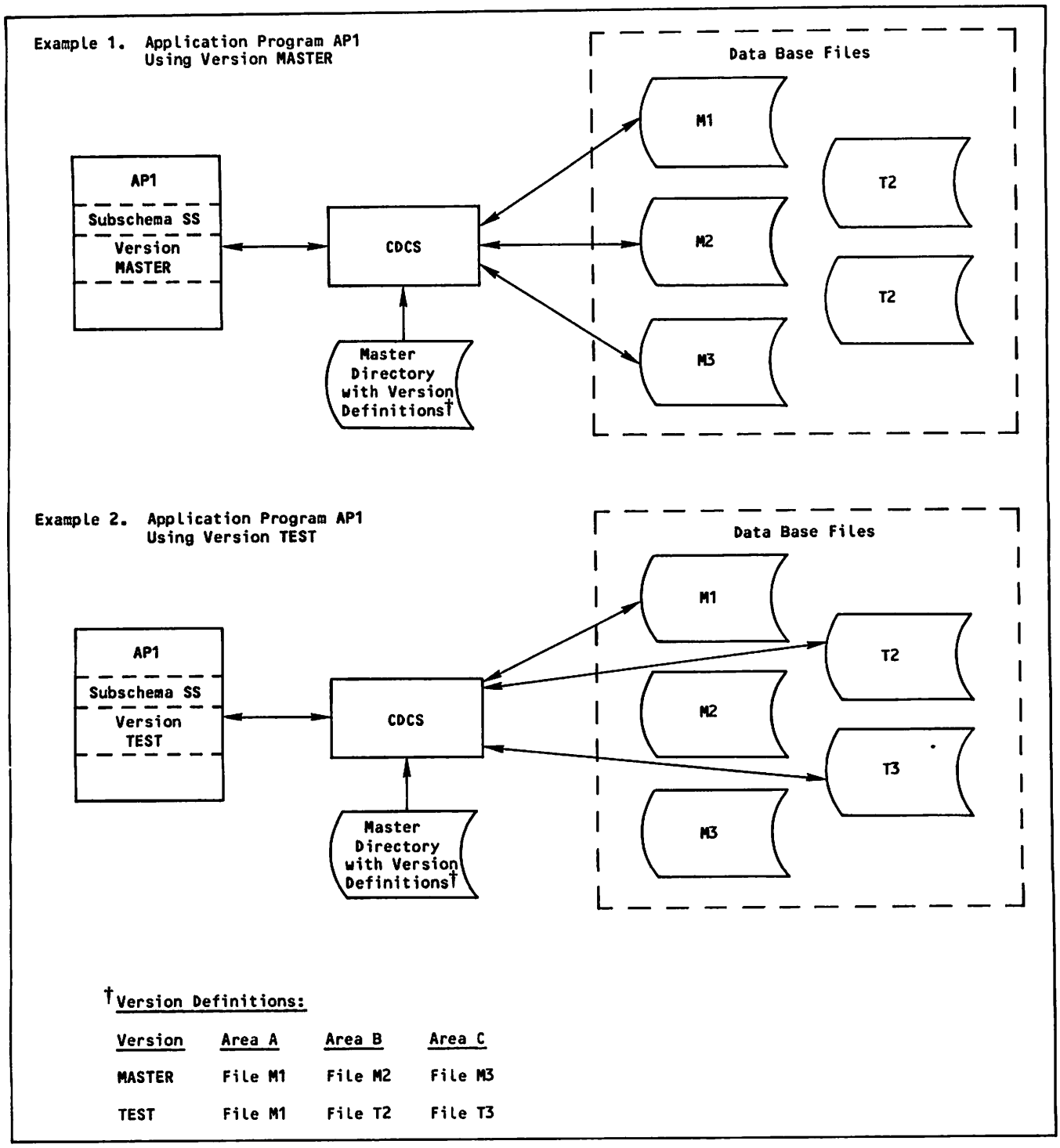
INPUT/OUTPUT PROCESSING

The input/output capabilities of CRM handle all operations concerning the physical storage and access of data in a data base. When an application program requests execution-time processing of input/output statements for data base files, CDCS directs the request to CYBER Record Manager Advanced Access Methods (AAM). AAM processes the requests according to the requirements and restrictions for conventional files.

All data base files supported by CDCS are conventional extended AAM files.

File Organization

File organization of data base files is specified in the operating system FILE control statement when the schema is compiled. The file organization information is stored in the schema directory. The only file organizations allowed for data base files that are to be accessed through CDCS are indexed sequential, direct access, and actual key.



†Version Definitions:

Version	Area A	Area B	Area C
MASTER	File M1	File M2	File M3
TEST	File M1	File T2	File T3

Figure 1-4. Processing Using Data Base Versions

In files of these organizations, records are stored by value of a data item within the record that is defined in the schema as the primary key. During input/output operations on files, AAM enforces the following rule: the value of the primary key cannot be duplicated within a file.

Records in indexed sequential files are stored in ascending order by value of the primary key. The records can be accessed either randomly by key or sequentially by position. This file organization should be used for files that are to be accessed both randomly and sequentially.

Records for direct access files are stored randomly in fixed length blocks. The number of the block to receive a record is determined by a calculation performed by the system on the record key. Records can be accessed randomly by key or serially. Direct access file organization is used most effectively when rapid random access is required.

Actual key files contain records whose primary key values are assigned by the system. The primary key value (also called the actual key) is a number that identifies the block and the position within the block in which the record is stored. Records can be accessed randomly by actual key; records also can be accessed serially. Actual key file organization is used most effectively when alternate keys are needed and when performance and file growth characteristics are of primary concern. It is also used when no unique key exists so that a system-defined key must be designated.

NOTE

Refer to appendix F for recommendations on access methods.

Multiple-Index Processing

Multiple-index processing is performed when alternate keys are defined for indexed sequential, direct access, and actual key files. An index is created for each alternate key in a data file when the file is created. The indexes are updated automatically whenever the data file is updated. Records can then be retrieved by the primary key or by an alternate key. For detailed information refer to the CYBER Record Manager Advanced Access Methods reference manual.

DATA BASE RECOVERY

The recovery facilities of CDCS provide ways to deal with the following situations:

Recovery of a data base from a system failure

Recovery of a data base from a program failure

Restarting a program after a system failure

Recovery of a lost, partially destroyed, or invalid data base

The facilities of automatic recovery, transaction processing, manual recovery, and logging deal with recovery situations.

Automatic Recovery

Automatic recovery provides for data base recovery from a system or program failure. After a system failure occurs, CDCS automatically performs recovery operations at CDCS-initialization time before CDCS becomes available to application programs. After a program failure, CDCS automatically performs the recovery operations with no effect on other applications.

The use of automatic recovery features for data base files is optional. The data administrator selects the features by specifying clauses in the master directory.

Data Base Transaction

The data base transaction feature is the application programming interface to automatic recovery. A data base transaction (called, simply, a transaction) is a sequence of one or more updates that usually involve related files. An application program must use data base transactions for CDCS to perform automatic recovery.

The application program specifies the beginning of the transaction, performs the update operations, and specifies the end of the transaction, which can be either a commit or a drop. When a commit transaction is specified, CDCS makes the updates permanent. When a drop transaction is specified, all the updates performed within the transaction are reversed by CDCS; therefore the data base is restored to its state before the beginning of the transaction. If the application program fails to commit a transaction because of a system or program failure, automatic recovery is performed and the data base is restored to its state before the beginning of the transaction.

The data base transaction feature is available to COBOL and FORTRAN application programs. It is not available to Query Update users.

The data base transaction feature also provides a restart capability for application programs after a system failure. Using this feature, the application program can determine the last transaction that was committed by the program before a failure occurred.

Manual Recovery Utilities

Manual recovery utilities supplement the capabilities of automatic recovery. The manual recovery utilities allow the data administrator to reconstruct a destroyed data base or to restore an invalid data base.

Logging

CDCS performs logging at execution time. Log files provide information necessary for automatic recovery, for manual recovery, and for restarting an application program.

Four types of log files can be used in data base recovery operations. The first, the journal log file, contains a record of each occurrence of an update or write operation on a data base. In addition, a record is maintained on the journal log file of certain user requests and privacy breach attempts. The second, the quick recovery log file, is used internally by CDCS to write blocks of records before the data base is modified by AAM. The third, the transaction recovery file, contains records used by CDCS to support automatic recovery. The fourth, the restart identifier file, contains information used by CDCS to provide for application program restart if program or system failure occurs.

All logging is optional and is selected by the data administrator in clauses of the master directory. All log files are assigned on a per-schema basis and selected for use for particular areas and versions.

The log files can provide information about data base usage. The journal log file, in addition to being input to a recovery run, can be processed by a program for statistical analysis at a later date. The restart identifier file can be processed by a program to determine which application programs failed and were never restarted.

A journal log file support feature provided by CDCS is journal log file maintenance. CDCS automatically detects a full journal log file, switches logging to an alternate journal log file, and initiates the transfer of the contents of the full file to a tape file. This feature provides for continuous logging during execution of CDCS.

OTHER DATA BASE UTILITIES

Other utilities applicable for use with a CDCS-controlled data base include FORM, a file management utility used to manipulate records and reorganize files, and two AAM utilities used with extended indexed sequential files: FLSTAT for obtaining statistical information, and FLBLOK for estimating the optimal block and buffer sizes for files. Two additional AAM utilities can be used with any of the extended AAM file organizations: MIPGEN for adding alternate key access to an existing extended AAM file, and MIPDIS for disassociating and reassociating data and alternate key index files. The FORM reference manual and the CYBER Record Manager Advanced Access Methods reference manual should be consulted for further details on these utilities.

PROCESSING THROUGH TAF

CDCS supports the Transaction Facility (TAF), which allows processing through TAF under NOS. Processing through TAF provides for high speed handling of repetitive executions of a relatively small number of jobs called tasks. The tasks can be executed by many different people from many locations. A task usually performs one of the following manipulations on a data base:

Stores a new record

Alters or deletes an existing record

Produces formatted output

An online banking system is an example of processing frequently performed through TAF: tellers in many locations use terminals connected online to a central processor to make deposits or withdrawals for an account and to print confirmations. A task (a deposit or withdrawal) is initiated by a teller through the terminal; once initiated, the task is executed through TAF and CDCS. The task can communicate with the terminal through TAF and the Network Access Method (NAM) and can initiate subsequent tasks.

Figure 1-5 shows the CDCS/TAF interface. Access to the data base through TAF is concurrent with access in both batch and interactive modes. All access to the data base is monitored by CDCS.

TAF tasks must be coded with special TAF requests. Refer to the TAF reference manual for information about the requests.

Since both TAF and CDCS provide for transaction processing, the term transaction processing used in this manual refers to data base (or CDCS) transaction processing. The term TAF transaction processing refers to processing through the TAF interface.

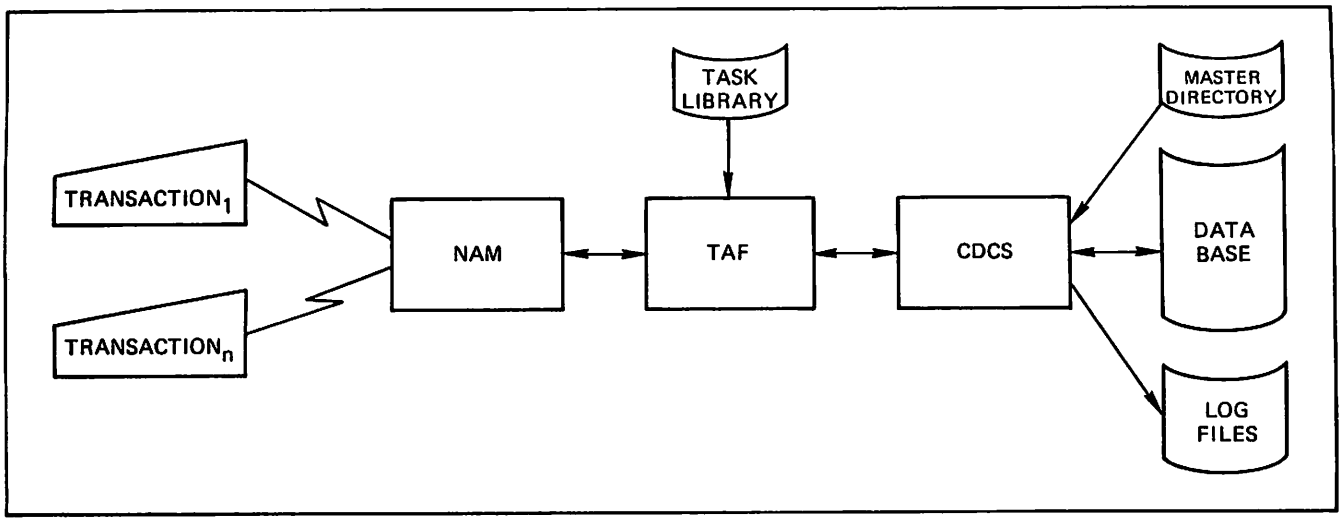


Figure 1-5. CDCS/TAF Interface

CDCS PROCESSING FLOW

Execution of an application program in the CDCS data base environment requires that CDCS be at a system control point and that the master directory be available. The COBOL compiler and the FORTRAN DML preprocessor generate calls in COBOL and FORTRAN programs, respectively, to object-time routines that format input/output requests to CDCS during execution. Query Update interprets directives and generates the appropriate calls to CDCS for execution of the directives involved in data base access. The first executable call is an invoke call to CDCS. During invocation, CDCS registers the user program, reads the master directory, builds internal tables and structures, and activates log files for the schema being used. CDCS also attaches all data and index files that are associated with realms referenced in the subschema used by the program. In addition, any file that is associated with a realm that is involved in a constraint with a realm that is referenced in the subschema is also attached by CDCS. The system flow for basic retrieval and basic update requests is as follows:

CDCS receives an input/output request from the application program via a call from the appropriate object-time routine.

CDCS analyzes the call and supplements information provided in the call with schema and subschema information from the master directory.

Depending on the call, CDCS can perform the following operations:

- Reads additional parameters from the user control point.
- Performs any required record mapping.
- Performs logging operations.
- Checks for violations of constraints.
- Determines whether update operations are being performed within a data base transaction.
- Performs automatic recovery if the application program issues a request to drop a transaction.
- Issues AAM requests as required to execute the call. AAM transfers data between the data base files and the input/output buffers in the CDCS field length.
- Loads and executes data base procedures.

CDCS transfers the status of the input/output request and any data to the user control point and allows control to be returned to the user program.

CDCS LOADING AND EXECUTION

CDCS resides on the system library as an absolute program and normally operates through the system control point facility of the NOS and NOS/BE operating systems. An application program using CDCS resides at a user control point. CDCS and one or more application programs can reside at the same control point when the CDCS Batch Test Facility is used. Refer to the following subsection for more information on this facility.

Figure 1-6 illustrates the data base environment at the system and user control points. This diagram also includes mass storage requirements for data base files, log files, the data base procedure library, and the master directory.

After CDCS is initiated by the operator at a system control point, the remainder of the field length represents managed memory under the control of the Common Memory Manager (CMM). Managed memory contains CDCS internal tables, CYBER Record Manager access methods, CYBER Record Manager buffers, mapping capsules, and data base procedure code capsules loaded via Fast Dynamic Loader (FDL). Establishment of CDCS as an active subsystem at a system control point is described in section 10. For further details on basic field length requirements for CDCS refer to appendix G.

The user control point includes the user program, compiled object-time routines, error status registers, and the user work area. User programs can be relocatable, absolute, or overlay programs. Refer to the CDCS 2 Application Programming reference manual for information on application programming.

CDCS BATCH TESTING

The CDCS Batch Test Facility can be used for program development when data and file definitions are changing frequently. The CDCS Batch Test Facility can be used with COBOL and FORTRAN application programs. In batch test mode, CDCS and one or more user programs run at one control point as a normal batch job. When the CDCS Batch Test Facility is used, new versions of the master directory file can be attached each time the job is run. In normal CDCS mode, the system control point must be dropped and reinitiated to attach a new master directory file.

The CDCS Batch Test Facility resides on the system library as an absolute program and is called into execution by the CDCSBTF control statement. Multiple copies of the program can be run concurrently with each other and with a system control point version of CDCS. As many as 16 user jobs can be run with one copy of the CDCSBTF program. User programs must be in relocatable binary format. Programs in absolute binary format, as well as segmented programs and overlays, cannot be run in batch test mode. For further details on the use of the CDCS Batch Test Facility refer to the CDCS 2 Application Programming reference manual.

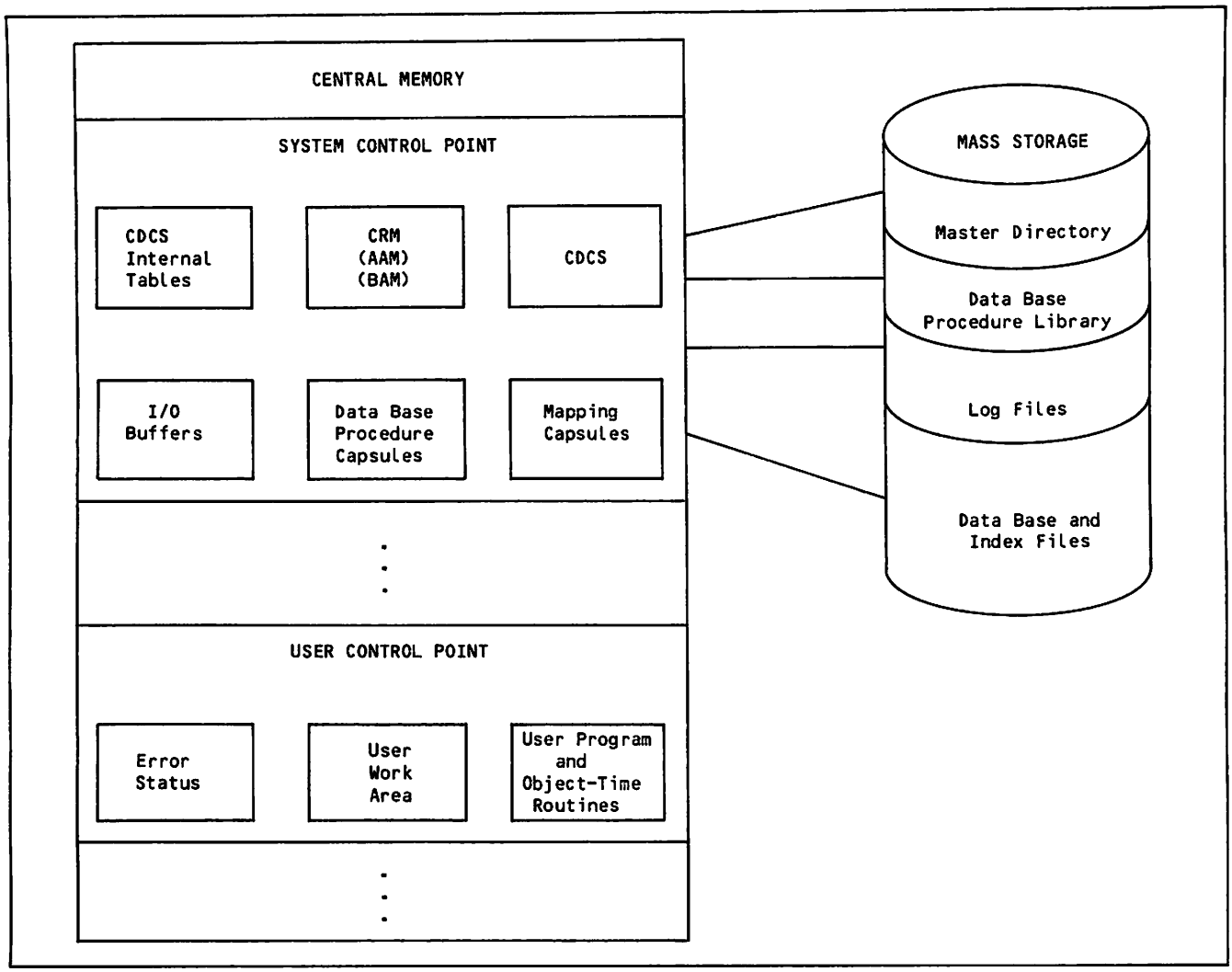


Figure 1-6. Data Base Environment Under Operating System

The schema is a directory that describes the characteristics of data items within the data base and specifies the organization and storage of the data. The internal storage format is specified for individual data items. The data items are organized into records within addressable storage units called areas.

The schema directory, or object schema, is generated when the schema source program is compiled by the DDL3 compiler. The schema directory is often called simply the schema.

SCHEMA STRUCTURING CONVENTIONS

The schema source program consists of six major types of entries: four required, two optional.

Schema Identification Entry

Assigns a name to the schema; this entry is required.

Area Description Entry

Assigns a name to a schema area and can specify a data base procedure to be performed when the area is opened or closed; at least one area description entry is required.

Record Description Entry

Describes the structure and characteristics of a DDL record type and can specify data base procedures to be executed whenever specific user functions manipulate a record; at least one record description entry is required.

Data Control Entry

Supplies special information related to the schema areas; this entry is required.

Constraint Entry

Assigns a name to a constraint and specifies data items used to associate areas in a constraint; this entry is optional.

Relation Entry

Assigns a name to the relation and specifies the data items to be used as join terms to link areas in a relation; this entry is optional.

DATA DESCRIPTION

Each DDL record type in the data base is described in a record description entry. The first statement in the entry assigns a name to the record type, specifies the area in which it resides, and optionally designates a data base procedure to be executed when the record is involved in a specific user function. The remainder of the record description entry consists of a series of data description entries that describe the individual data items within the record.

A data description entry contains a data name, one or more clauses describing the data item, and a terminating period. The entry can also have a level number that designates the position of the data item in the hierarchical structure of the record. If the level number is omitted, level number 01, the highest level in the hierarchy, is assumed by default.

The data name in a data description entry is a user-defined name that identifies the data item. It must conform to the rules governing user-defined names as described in the User-Defined Names subsection, which appears later in this section. The data name cannot be qualified; it must be unique within the record description entry.

At least one clause must be included in a data description entry; additional clauses can be specified as needed. The required clause must be the PICTURE clause, the TYPE clause, or the OCCURS clause. Certain restrictions are placed on the combination of clauses that can be included in a data description entry. Table 2-1 lists the clauses that can be specified and indicates the valid combinations.

The only punctuation required in a data description entry is the terminating period. Semicolons or commas can be used to separate clauses. Commas can also be used to separate repeated options.

DATA ORGANIZATION

Data items in the data base are organized first into areas and then into records. Within a record, two types of data items can be specified: elementary items and group items. Repeating data items, which can be elementary or group items, are fixed or variable in length.

Areas

The data items defined in the schema must be organized into addressable storage units called areas. At least one area must be described in the schema; if only one area is described, it encompasses the entire data base. The maximum number of areas allowed is 4095.

An area is a portion of mass storage that can be accessed in the same manner as a file. An area is opened and closed by the user OPEN and CLOSE functions.

TABLE 2-1. VALID CLAUSE COMBINATIONS

	PICTURE	TYPE	OCCURS	VIRTUAL RESULT	ACTUAL RESULT	CHECK	ENCODING	DECODING	CALL
PICTURE			X	X	X	X	X	X	X
TYPE			X	X	X	X	X	X	X
OCCURS	X	X				X	X	X	X†
VIRTUAL RESULT	X	X				X			
ACTUAL RESULT	X	X				X		X	X
CHECK	X	X	X	X	X		X	X	X
ENCODING	X	X	X			X		X	X
DECODING	X	X	X		X	X	X		X
CALL	X	X	X†		X	X	X	X	

†This combination of clauses can only be used in the data description entry for an elementary item (vector).

An area can contain more than one DDL record type. When multiple record types exist within an area, the schema must designate the means for determining the specific record type to be used for the occurrence of an actual record. A record code identifies each record type in the area. The record code is determined in one of two ways:

A data item in the actual record contains a unique value for each record type.

Execution of a data base procedure returns a unique value for each record type.

If the record code is contained in a data item, the data item must be in the same position in each record type within the area. The data items must also be the same size and data class. The data names, however, need not be the same.

NOTE

Refer to appendix F for recommendations on the use of multiple record descriptions.

Records

Within each area in the schema, data items are organized into records. Each DDL record type is defined in a record description entry that assigns a name to the record, designates the area in which the record resides, optionally specifies data base procedures to be executed when the record is manipulated, and describes the data items included in the record. The maximum number of record types allowed is 4095. A maximum record size of 81870 characters is allowed.

The hierarchical structure of the record is indicated by level numbers. Only repeating group items can designate subordinate data items and therefore must specify level numbers. A nonrepeating data item that is not subordinate to a repeating group item is the highest level (lowest number) data item in the record.

Elementary Items

An elementary item is the smallest unit of named data; it cannot be subdivided into other data items. If it is part of a repeating group item, the elementary item has the highest level number of the group to which it belongs. Elementary items that do not belong to a repeating group have the lowest level number in the record; if a level number is not specified, level 01 is assumed by default. The maximum number of items allowed is 81870; the maximum item size is 32767 characters. The maximum number of items per record is 4095.

Group Items

The only group items that can be specified in the schema are repeating groups. Nonrepeating group items are not supported in the schema. A repeating group is a collection of related data items organized in a hierarchical structure; the entire collection is repeated a number of times. The collection can include elementary items and other repeating group items. Group items can be nested to three levels.

The group name data description entry has the lowest level number; it must also include the OCCURS clause. A data description entry is written for each item within the group item; these entries must have level numbers that are higher than the group name level number.

Repeating Data Items

Two types of repeating data items can be designated in the schema: vectors and repeating groups. A vector is an elementary data item that is repeated a number of times in each record. A repeating group is a collection of data items that is repeated; the entire collection, not individual data items, is repeated a number of times in each record.

Repeating data items are specified by including the OCCURS clause in the data description entry. A repeating data item can occur a fixed number of times in each record or a variable number of times depending on the value of another data item in the record.

Vectors

A vector is an elementary data item that contains a series of values. It is described with the OCCURS clause and either the PICTURE or TYPE clause. No data item can be subordinate to a vector; it must be an elementary data item. A vector can be a data item within a repeating group; however, it must be a fixed occurrence vector.

Repeating Groups

A repeating group is described with two or more data description entries. The first entry consists of the group data name and the OCCURS clause. Each additional entry is subordinate to the first entry and describes a repeating group, a vector, or an elementary data item. The vector or elementary item contains the PICTURE or TYPE clause. Up to three levels of nested groups can be specified. A repeating group that is subordinate to another repeating group must be a fixed occurrence repeating group.

Level numbers must be specified in a repeating group. The lower the position of the data item in the hierarchy of the repeating group, the higher the level number must be. Data items in the same hierarchical position must have the same level number.

Fixed Occurrence Data Items

A data item that is repeated a fixed number of times is described with the OCCURS integer TIMES clause. The integer specifies the exact number of occurrences of the data item for each record occurrence; it must be a positive number greater than zero. A fixed occurrence data item can be a subordinate entry in a repeating group.

NOTE

Refer to appendix F for recommendations on the use of repeating groups.

Variable Occurrence Data Items

A data item that is repeated a variable number of times is described with the OCCURS data name TIMES clause.

The exact number of times the data item is repeated in a record occurrence depends on the value of the data item referenced by data name in the OCCURS clause.

The data item that controls the size of the variable occurrence data item must be described as an integer; the CHECK IS VALUE clause must be included to designate the minimum and maximum number of occurrences that are allowed. The controlling data item cannot be within a variable occurrence data item. If it is within a fixed occurrence data item, the first occurrence of the controlling data item determines the size of the variable occurrence data item.

The following rules apply to variable occurrence data items:

A variable occurrence data item must be the last item in the record. Only subordinate data description entries can follow the entry containing the OCCURS data name TIMES clause.

A variable occurrence data item cannot be a subordinate entry in a repeating group item.

Only one variable occurrence data item can be specified for a record.

DATA SIZE AND CLASS

The size and class of a data item are designated by either the PICTURE clause or the TYPE clause. The data description entry for an elementary data item must include one of these two clauses.

Data size is specified in the PICTURE clause by the number of character position designators (A, X, 9, . (decimal point), and T) in the picture specification. In the TYPE clause, the number of positions for the data item is designated by specifying an integer. For a numeric data item, the integer is the maximum number of significant digits; for an alphanumeric data item, the integer is the number of character positions.

Each data item falls into a data class category. The data class is determined by the description of the data item. Refer to the descriptions of the PICTURE clause and the TYPE clause for information on the notations used in describing the data classes.

Display Alphanumeric (Data Class 0)

TYPE CHARACTER clause with integer-3

PICTURE clause with alphanumeric picture specification (A, X, and 9 characters)

Display Alphabetic (Data Class 1)

PICTURE clause with alphabetic picture specification (character A only)

Display Integer (Data Class 3)

PICTURE clause with numeric picture specification (9 and T characters)

Display Fixed Point (Data Class 4)

PICTURE clause with fixed point numeric picture specification (9, V, T, P, and . characters)

Coded Binary Integer (Data Class 10)

TYPE FIXED clause with integer-1 in the range 1 through 18

Coded Floating Point Normalized (Data Class 13)

TYPE FLOAT clause with integer-1 in the range 1 through 14

Coded Double Precision (Data Class 14)

TYPE FLOAT clause with integer-1 in the range 15 through 29

Coded Complex (Data Class 15)

TYPE COMPLEX clause

Refer to section 5 for a summary of data definition for a CDCS controlled data base; refer to appendix J for a summary of data definition in DMS-170.

SCHEMA/SUBSCHEMA COMPATIBILITY

The schema describes the characteristics of all data items in the data base. The subschema describes only those data items to be accessed by one or more application programs. Within certain limitations, the characteristics of the data items can be changed in the subschema to meet the requirements of the application programs. The process of changing data characteristics between the schema and the subschema is called conversion.

Conversion occurs during record mapping, which is the CDCS operation for generating a record image. Record mapping produces a record image conforming to the subschema description when a user GET function (READ statement) requests the record. A record image conforming to the schema description is produced as a result of a user STORE or MODIFY function (WRITE or REWRITE statement).

Refer to sections 3 and 4 for detailed descriptions of the conversion limitations imposed on the subschema by the schema description. Conversion and record mapping are discussed in detail in section 5 and appendix H.

DATA CONVERSION

The description of a data item determines the data class of the item. Conversion from a schema data class to a different data class in the subschema is restricted to specific data class conversions. Table 2-2 indicates the valid subschema data classes for each schema data class. When the schema is created, the general usage of data items and the valid data class conversions should be considered. Conversion is inhibited when a data item is described with the CHECK IS PICTURE clause.

TABLE 2-2. VALID SCHEMA/SUBSCHEMA CLASS CONVERSIONS

Schema Data Class†	Subschema Data Class†							
	0	1	3	4	10	13	14	15
0	X	X	X					
1	X	X						
3	X		X	X	X	X	X	
4			X	X	X	X	X	
10			X	X	X	X	X	X
13			X	X	X	X	X	X
14			X	X	X	X	X	X
15					X	X	X	X

†The schema and subschema data classes are identified by the following codes:

0 Display Alphanumeric
1 Display Alphabetic
3 Display Integer
4 Display Fixed Point
10 Coded Binary Integer
13 Coded Floating Point Normalized
14 Coded Double Precision
15 Coded Complex

Data conversion operates in a source-to-target mode. The source and target data items are defined as follows:

GET Function (READ Statement)

Source=schema data item
Target=subschema data item

STORE or MODIFY Function (WRITE or REWRITE Statement)

Source=subschema data item
Target=schema data item

When conversion is required and the source data item and target data item are unequal in length, the following rules apply to a character data item:

If the source data item is larger than the target data item, the source data item is truncated on the right to the size of the target data item. Truncation of nonblank characters results in an error.

If the source data item is shorter than the target data item, the source data item is blank filled on the right to the size of the target data item.

If an error occurs during data conversion, an error code is returned to the user program, and the conversion operation is terminated. If the schema specifies a data base procedure in a CALL ON ERROR clause, the procedure is executed. The following paragraphs discuss conversion of numeric data items.

Coded Arithmetic to Numeric Picture

If the target data item has a numeric picture specification, conversion occurs only if the source data item is coded arithmetic or can be converted to coded arithmetic. The value of the source data item is converted to character representation. Insignificant zeros are added to or removed from either end of the target data item to conform to the precision and scaling factor specified for the target data item. If the scaling factor of the target data item is less than that of the source data item, rounding takes place in the least significant digit. If the precision of the target data item is not sufficient to contain all significant digits of the whole part of the source data item, an error occurs and conversion does not take place.

Numeric Picture to Coded Arithmetic

When the source data item has a numeric picture specification and the target data item is a coded arithmetic data item, the value of the source data item is converted to the internal representation. Insignificant zero digits are added or removed and rounding occurs as needed in the least significant digit. If the precision of the target data item is not sufficient to contain all significant digits in the whole part of the source data item, an error results and conversion does not occur.

Coded Arithmetic to Coded Arithmetic

The value of the source data item is converted, if necessary, to the base, scale, mode, and precision of the target item. Insignificant zeros are added or removed and rounding occurs as needed in the least significant digit. If the precision of the target data item is not sufficient to contain all significant digits of the whole part of the source data item, an error results and conversion does not occur.

Numeric Picture to Numeric Picture

When both the target and source data items are fixed point and differ only in the number of digits or the decimal point position, insignificant zeros are added or removed to conform to the target data item picture specification and rounding occurs as needed. If the target data item precision is not sufficient to contain all significant digits, an error condition results and conversion does not occur.

RECORD MAPPING

Record mapping is performed by CDCS whenever a user function reads or writes a data base record. The schema and subschema record descriptions are used to generate a record image. Each data item is transferred from the source record to the target record. Conversion is performed before the data item is transferred. When no source data item corresponds to a target data item, the target data item is given a null value (blanks or zeros depending on the data class of the data item).

In addition to resolving differences between the schema and subschema descriptions, any of the following clauses included in the schema description of a data item are handled during record mapping:

RESULT Clause

The data base procedure is executed to determine the value of the data item.

ENCODING or DECODING Clause

The data base procedure is executed to perform a nonstandard conversion for the data item.

CHECK Clause

The data item is checked for the restrictions imposed by this clause.

CALL Clause

The data base procedure is executed when the specified function is performed on the data item.

SCHEMA PROGRAMMING CONVENTIONS

The DDL schema source program consists of a series of statements that describe the data base. The rules, conventions, and hierarchical structures of DDL are similar to those of the COBOL programming language.

LANGUAGE ELEMENTS

DDL source statements are composed of clauses that contain reserved words, user-defined names, and literals. The use of these elements is described in the following paragraphs. The specific formats of the clauses are defined in the Schema Syntax subsection, which appears later in this section.

Reserved Words

Reserved words are English words and abbreviations that have special meanings to the DDL compiler. These words can be used only as shown in the format specifications. A reserved word must be spelled correctly; it cannot be replaced by another word. Appendix D contains a complete list of DDL reserved words.

Two types of reserved words are recognized by the DDL compiler: keywords and optional words. A keyword is a reserved word that must be used in a specific clause. Keywords are essential to convey the meaning of a clause to the compiler. An optional word is a reserved word that can be included in a clause to improve readability. Optional words are recognized by the compiler but are not needed to compile the object coding. In the format specifications, keywords are shown as upper case words that are underlined; optional words are shown as upper case words that are not underlined.

User-Defined Names

Many of the format specifications include names that are supplied by the user. User-defined names identify the schema, areas, relations, constraints, records, data items, and data base procedures. The type of name to be supplied is indicated in the format specification by a lower case word.

User-defined names have two formats: normal and escape. The formation of normal names is governed by the following rules:

The name can contain up to 30 characters.

Letters (A-Z), digits (0-9), and the hyphen (-) can be used in a name.

The first character must be a letter.

The hyphen cannot be used to begin or end a name.

Adjacent hyphens are not allowed.

Spaces (blanks) cannot be used in a name.

A name cannot be spelled exactly the same as a reserved word.

Two types of user-defined names must conform to additional rules:

Area Names

Only letters (A-Z) and digits (0-9) can be used.

The first character must be a letter.

The first seven characters must be unique.

Data Base Procedure Names

Only letters (A-Z) and digits (0-9) can be used in the name.

The first character must be a letter.

A maximum of seven characters can be specified.

The escape format is used when the name does not conform to the rules for normal names. An escape name is most commonly used when the data base procedure already exists and the name cannot be used as a normal name. The formation of escape names is governed by the following rules:

The name can contain up to 30 characters.

Any character in the DDL character set can be used in the name.

The name is delimited by the national currency character ($).

The delimiter ($) can be included in the name by specifying it two consecutive times for each occurrence.

Literals

In some formats, the user must supply a literal as part of the clause. A literal is a string of characters that represents a specific value. Literals are either numeric or nonnumeric.

Numeric Literals

A numeric literal can contain the numbers 0 through 9, the decimal point, and the plus or minus sign. Numeric literals are expressed as either fixed point or integer. The rules for formation of numeric literals are as follows:

Fixed Point

One sign character can be specified as the leftmost character; if a sign is not specified, the literal is positive.

One decimal point can be specified in any position except the rightmost position; if a decimal point is not specified, the literal is an integer.

Integer

A decimal point cannot be specified.

One sign character can be specified as the leftmost character; if a sign is not specified, the literal is positive.

The maximum size for a numeric literal is 30 digits; however, only 18 significant digits can be specified. Up to 12 leading or trailing zeros can be specified for decimal point alignment.

Nonnumeric Literals

A nonnumeric literal is a string of up to 255 characters. The string must be enclosed in quotation marks. Any character in the DDL character set, including the space, can be used in a nonnumeric literal. If a quotation mark is to be included in the literal, the quotation mark, must be specified twice for each occurrence. For example, "A""B" would yield the literal A"B.

Data Reference

Each user-defined name in the schema must be capable of being uniquely referenced. Unless the name itself is unique because no other name has the identical spelling, a method for obtaining unique identification is necessary. Unique reference is recognized through the qualification, subscripting, and identifier concepts.

Qualification

Qualification is permitted in any clause that references a data name. When a name exists within a hierarchy of one or more names, the higher level names can be used to make the name unique. The data name is written followed by the word OF or IN and the qualifier. The choice of OF or IN is based on readability; the two words are logically equivalent. Qualification must be made to the level necessary to make the name unique; however, qualification can be used even when the name does not need to be qualified.

Subscripting

Subscripting within the DDL syntax is permitted only in the JOIN clause. Subscripts are used to indicate which occurrence of a repeating group or elementary item is to be referenced. The data name is written, followed by a positive integer constant enclosed within parentheses. The following paragraphs describe specific rules for subscript use.

Identifier

The term identifier is used in the schema JOIN clause to reflect a unique reference to a data name. The data name is referenced uniquely through a combination of subscripts and qualifiers. Qualification and subscripting for a data name follow the formats shown in figure 2-1.

Format 1 is used for subscripting and qualification in either a source identifier or a target identifier. Format 2 is used to describe a target identifier that is composed of all the occurrences of a repeating item and is an alternate key or the major part of an alternate key. Refer to the ANY option discussed in the JOIN clause subsection.

Subscripting in the schema is permitted only in the JOIN clause. The only qualifier allowed for an identifier is record name, as specified in the RECORD NAME clause in the schema.

DDL Character Set

The set of characters recognized by the DDL compiler can be combined according to the specified rules for forming names and values in the source program. The DDL character set consists of the letters A through Z, the numbers 0 through 9, and the following special characters:

	Blank or Space
+	Plus Sign
-	Minus Sign or Hyphen
,	Comma
;	Semicolon
*	Asterisk
.	Period or Decimal Point
"	Quotation Mark
(	Left Parenthesis
)	Right Parenthesis
$	Dollar Sign (national currency character)
/	Slash

The following reserved symbols are also included in the DDL character set:

=	Equals
>	Greater than
<	Less than

Punctuation

Most punctuation marks in a DDL source program are optional. When punctuation marks are used, the following rules apply:

A period must terminate each complete statement as indicated by statement formats.

A period must be followed by at least one space.

A left parenthesis must not be followed by a space, and a right parethensis must not be preceded by a space.

At least one delimiter (a space, comma, or semicolon) must separate successive words in a statement.

Commas and semicolons can separate clauses in a statement.

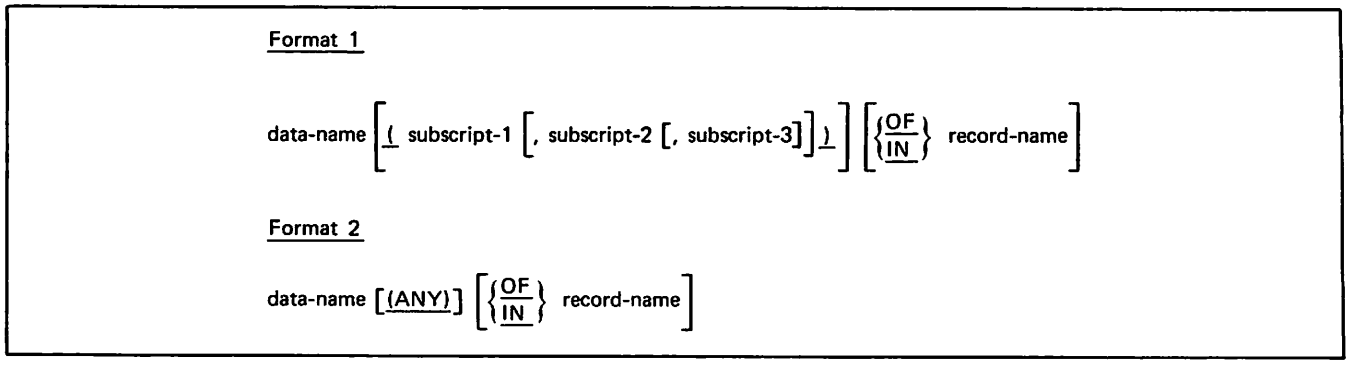


Figure 2-1. Identifier Format

DDL CODING

DDL source programs can be written on standard coding sheets. Coding a schema is similar to coding the Data Division of a COBOL applications program. Columns 1 through 72 are used to write DDL statements; columns 73 through 80 are reserved for sequence numbers. A DDL statement is written in free-format within the chosen columns. A DDL statement terminates with a period.

DDL Statements

The DDL source program consists of a series of entries. Each entry contains at least one statement; each statement contains at least one clause. The entries must appear in the source program as described in the Schema Syntax section.

Sequence Numbers

A sequence number consisting of digits only can be entered in columns 73 through 80. The sequence number is optional and has no effect on the source program.

Comment Lines

Comments can be included in the source listing for documentation purposes. A comment must be preceded and followed by a delimiter, which consists of two special characters. The preceding delimiter is a slash and an asterisk (/*); the following delimiter is an asterisk and a slash (*/).

A comment can begin wherever a space is entered. All characters between the delimiters are considered a comment and are not processed by the DDL compiler.

SCHEMA SYNTAX

The source program for a DDL schema consists of six major types of entries. Each entry contains at least one statement terminated by a period. The statement contains one or more clauses. The general format of the DDL schema definition is shown in figure 2-2.

```
schema identification entry

{area description entry} ...

{record description entry} ...

{data control entry}

[constraint entry] ...

[relation entry] ...
```

Figure 2-2. General Format, Schema

The schema identification entry must be the first entry in the source program. An area description entry must precede the record description entries for all records in the area; all area description entries can precede the first record description entry. The data control entry must follow the record description entry. The constraint entry, if included, must follow the data control entry and precede any relation entry. The relation entry, if included, must be the last entry in the source program.

The function and placement of the major entries in a schema source program are as follows:

Schema Identification Entry

Assigns a name to the schema.

Area Description Entry

Assigns a name to an area and can also specify data base procedures to be executed whenever the area is opened or closed. One entry must be included for each area in the data base.

Record Description Entry

Describes the attributes of a record. The first statement assigns a name to the record and specifies the area that contains the record; it can also specify data base procedures to be executed whenever specific user functions manipulate the record. Additional statements (data description entries) define the characteristics of data items within the record.

Data Control Entry

Provides information related to the data base areas described in the schema. The area control entries supply compression and decompression procedures, record keys, record codes, and the collating sequence; each statement pertains to a specific area.

Constraint Entry

Assigns a name to the constraint and specifies data items used to associate areas in a constraint.

Relation Entry

Assigns a name to the relation and specifies the data items to be used as join terms.

These entries and the clauses that can be included in each entry are described in detail in this section of the manual. Refer to the Schema Structuring Conventions subsection, which appears earlier in this section, for the rules governing the structure of the schema. Appendix E contains the the complete summary of all DDL clauses used to create a schema source program.

SCHEMA IDENTIFICATION ENTRY

The first entry in a schema source program is the schema identification entry. This entry and following entries up to the end-of-information indicator constitute the schema description. The format of the schema identification entry is as follows:

SCHEMA NAME clause.

The schema identification entry is required and consists of one statement that identifies the schema.

SCHEMA NAME Clause

The SCHEMA NAME clause assigns a name to the schema. The format of the SCHEMA NAME clause is shown in figure 2-3.

```
SCHEMA NAME IS schema-name.
```

Figure 2-3. SCHEMA NAME Clause Format

The schema name must be unique among all schema names known to CDCS. The name assigned in this clause is the name specified in all subschemas that reference the data base described in the entries following the SCHEMA NAME clause.

Other Clauses

Only one clause is acceptable in the schema identification entry. Future software releases will provide additional clauses that can be included in this entry.

AREA DESCRIPTION ENTRY

An area description entry is included in the schema source program for each area (file) in the data base. This entry assigns a name to the area and can specify a data base procedure to be executed when the area is opened or closed by an applications program. The format of the area description entry is shown in figure 2-4.

```
AREA NAME clause

    [CALL clause]

    [ACCESS-CONTROL clause] ... .
```

Figure 2-4. Area Description Entry Format

At least one area description entry must be included in the schema. If only one entry is specified, it encompasses the entire data base. Each entry consists of one statement.

The area description entry must precede the record description entries for all records in the area described by the entry. All area description entries for the schema can be specified before the record description entries.

AREA NAME Clause

The AREA NAME clause assigns a name to an area in the schema. This clause is required and must be the first clause in an area description entry. The format of the AREA NAME clause is shown in figure 2-5.

```
AREA NAME IS area-name
```

Figure 2-5. AREA NAME Clause Format

The area name must be unique among all area names in the schema. It can contain up to 30 characters; however, operating system limitations impose the following restrictions on the area name:

- The first seven characters of the name must be unique.
- Only letters (A-Z) and digits (0-9) can be used.
- The first character of the name must be an alphabetic character.

For the area name to be valid in the master directory, it must not duplicate a reserved word for the DBMSTRD utility (refer to appendix D).

The area is associated with a particular file organization through a FILE control statement included in the set of control statements preceding the schema source program. The information from the FILE control statement is placed in the schema directory along with the information in the area description entry of the source program. Refer to the FILE Control Statement subsection, which appears later in this section, for more information.

CALL Clause (Area Description Entry)

The CALL clause in an area description entry causes a data base procedure to be executed when the area is opened or closed. The procedure can be invoked before or after the function is performed or when an error is detected while opening or closing the area. The FOR phrase gives the option of specifying either the UPDATE or RETRIEVAL usage mode. The format of the CALL clause is shown in figure 2-6.

The procedure named by data-base-procedure is executed when any specified function is executed on the area specified in the AREA clause. The data base procedure name must not exceed seven characters.

The BEFORE, ERROR, and AFTER options determine when the data base procedure is invoked.

BEFORE

The procedure is executed immediately before the area is opened or closed.

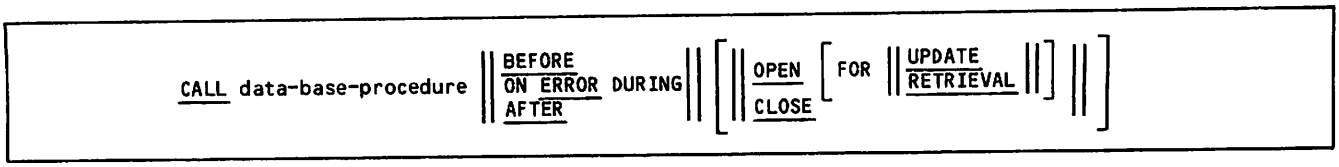


Figure 2-6. CALL Clause Format (Area Description Entry)

ERROR

The procedure is executed each time CDCS detects and reports an error during the opening or closing of the area.

AFTER

The procedure is executed immediately after the area is opened or closed.

At least one of these options must be specified; each option can be specified once in the CALL clause. If two or three options are included in the clause, the data base procedure is executed at each designated time.

In the following example, the data base procedure OPENSLS is executed both before and after the area is opened for updating purposes.

```
CALL OPENSLS BEFORE AFTER OPEN FOR UPDATE
```

The OPEN or CLOSE option specifies the user function that invokes the data base procedure at the designated time. Each option can be specified once in a CALL clause. Separate CALL clauses for each type of the OPEN function and the CLOSE function can invoke the same or different data base procedures. If no user function is specified in the CALL clause, the data base procedure is executed when any OPEN or CLOSE function is executed for the area. If OPEN is specified and the FOR option is not stated, the procedure is invoked when any type of OPEN function is executed for the area.

If the FOR option is stated, the data base procedure is invoked whenever the specified type of OPEN function is executed. In the FOR phrase, UPDATE and RETRIEVAL can appear only once.

Multiple data base procedures can be executed at a designated time. When more than one CALL clause for either OPEN or CLOSE specifies the same time to invoke a procedure (before, during, or after the user function is performed), the procedures are executed in the order the CALL clauses are stated in the area description entry. In the following example, the data base procedure EMPOPEN is executed, and then the procedure EMPCHK is executed before the area is opened.

```
CALL EMPOPEN BEFORE OPEN
CALL EMPCHK BEFORE OPEN
```

ACCESS-CONTROL Clause

The ACCESS-CONTROL clause in any area description entry specifies the privacy locks that apply to the use of an area. The format of the ACCESS-CONTROL clause is shown in figure 2-7.

The optional FOR phrase allows the access control locks to apply specifically to UPDATE or RETRIEVAL. If the FOR phrase is omitted, all literals or procedures apply to any use of the area.

The literals are privacy locks to be matched with the pertinent privacy key. All literals must be alphanumeric, and cannot exceed a maximum of 30 characters. The procedures named are privacy lock procedures, which, when given access to a privacy key, either return a yes or no result, or do not return at all.

Multiple privacy locks connected by OR phrases are considered satisfied if any one is satisfied. The privacy locks are processed in the order listed until the outcome of the ACCESS-CONTROL clause is known. The same literal or data base procedure can be specified for one or more options included in this clause.

A separate ACCESS-CONTROL clause can be stated for each usage mode. However, the same usage mode cannot be specified in more than one ACCESS-CONTROL clause. If the ACCESS-CONTROL clause is omitted, the use of the area being described is unrestricted.

RECORD DESCRIPTION ENTRY

Each DDL record type in the data base must be described in a record description entry. This entry assigns a name to the record, designates the area in which it resides, specifies data base procedures to be used in conjunction with record manipulation, and describes the data items within the record. The format of the record description entry is shown in figure 2-8.

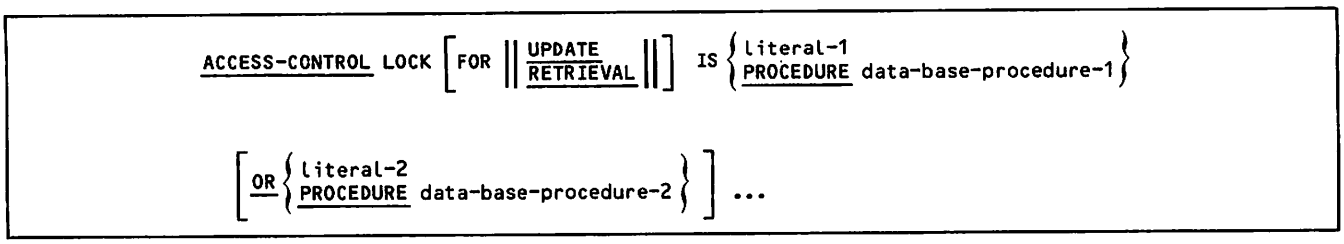


Figure 2-7. ACCESS-CONTROL Clause Format

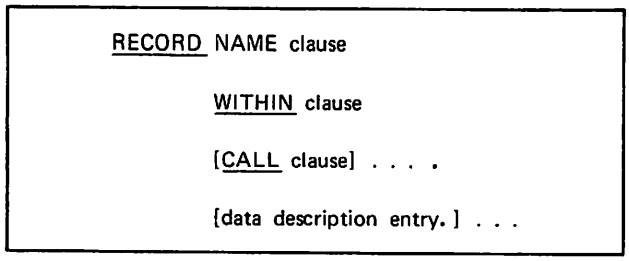


Figure 2-8. Record Description Entry Format

At least one record description entry must be included in the schema. The clauses in the record description entry must be specified in the order shown in the format. Multiple DDL record types can be specified for an area.

Each record description entry begins with a record name statement. This can be followed by a series of statements (data description entries) describing the individual data items within the record. If the record description entry defines a record type that contains only control information, no data description entries follow the record name statement.

The record described in the schema can be considered a contiguous collection of data in the data base; however, it is not necessarily equivalent to a physical record. The record description is machine independent and its actual placement in the data base is not defined in the record description entry.

RECORD NAME Clause

The RECORD NAME clause assigns a name to the record. It must be the first clause specified in the record description entry. The format of the RECORD NAME clause is shown in figure 2-9.

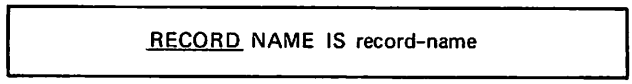


Figure 2-9. RECORD NAME Clause Format

The record name must be unique among all record names in the schema. It can be up to 30 characters in length. The name specified in this clause is associated with all the data items described in the same record description entry.

WITHIN Clause

The WITHIN clause specifies the area that contains the DDL record type. This clause is required in each record description entry. The format of the WITHIN clause is shown in figure 2-10.

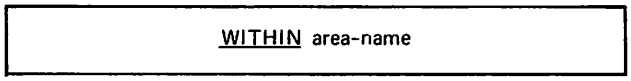


Figure 2-10. WITHIN Clause Format

The area name must be defined in an area description entry that precedes the record description entry. The same area name can be specified in more than one record description entry.

CALL Clause (Record Description Entry)

The CALL clause in a record description entry causes a data base procedure to be executed when a specified user function is performed on the record. A user function is a request from an applications program to perform an operation involving the record. The procedure can be invoked before or after the function is performed, or when an error is detected during performance of the function. The same data base procedure can be specified in separate CALL clauses. The format of the CALL clause is shown in figure 2-11.

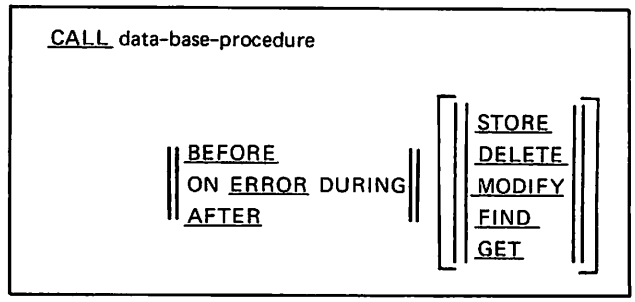


Figure 2-11. CALL Clause Format (Record Description Entry)

The BEFORE, ERROR, and AFTER options determine when the data base procedure is invoked.

BEFORE

For a STORE, MODIFY, or GET function, the procedure is executed immediately before record mapping occurs. For a DELETE or FIND function, the procedure is executed immediately before CYBER Record Manager is called to process the record.

ERROR

The procedure is executed each time CDCS detects and reports an error during the performance of the specified function.

AFTER

For a STORE, MODIFY, or GET function, the procedure is executed immediately after record mapping occurs. For a DELETE or FIND function, the procedure is executed immediately after CYBER Record Manager is called to process the record.

At least one of these options must be specified; each option can be specified once in the CALL clause. If two or three options are included in the clause, the data base procedure is executed at each designated time. The only exception occurs when the ERROR and AFTER options are specified and an error causes execution of the data base proce-

dure. In this situation, the data base procedure specified in the AFTER option is not executed. In the following example, the data base procedure VERIFY is executed both before and after the MODIFY function is performed.

```
CALL VERIFY BEFORE AFTER MODIFY
```

The user function specified in the CALL clause determines the type of record manipulation that causes the data base procedure to be executed at the designated time. Five functions can be specified:

STORE

The procedure is executed when a user function (WRITE statement) stores a record in the data base.

DELETE

The procedure is executed when a user function (DELETE statement) deletes a record from the data base.

MODIFY

The procedure is executed when a user function (REWRITE statement) modifies a record in the data base.

FIND

The procedure is executed when a user function (START statement) locates a specific record for subsequent processing of data base records.

GET

The procedure is executed when a user function (READ statement) reads a record in the data base.

Each option can be specified once in a CALL clause. Separate CALL clauses for individual user functions can invoke the same or different data base procedures. If no user function is specified in the CALL clause, the data base procedure is executed at the designated time for all five user functions.

Multiple data base procedures can be executed at a designated time. When more than one CALL clause for the same user function specifies the same time to invoke a procedure (before, during, or after the user function is performed), the procedures are executed in the order the CALL clauses are stated in the record description entry. In the following example, the data base procedure DELCHK is executed, and then the procedure DELVER is executed before the DELETE function is performed.

```
CALL DELCHK BEFORE DELETE
CALL DELVER BEFORE DELETE
```

DATA DESCRIPTION ENTRY (RECORD DESCRIPTION ENTRY)

A data description entry defines the characteristics of a data item within the record. An entry is required for each data item in the record. One or more clauses are included in each data description entry. The record description entry includes zero, one, or more data description entries; unless the record type contains only control information, at least one data description entry must be included. The format of the data description entry is shown in figure 2-12.

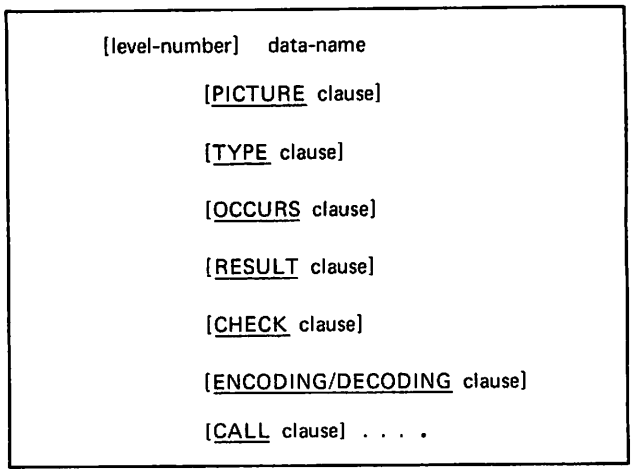


Figure 2-12. Data Description Entry Format

Each data description entry must begin with a data name that identifies the data item. The data name must be unique within the record. The level number is optional; it is used to indicate the structural level of the entry within the record description entry. A level number must always be the following:

An unsigned decimal integer

In the range 01 through 99

Greater than or equal to the level number of the first data description entry in the record description entry

If the level number is omitted, level 01 is assumed by default.

One or more clauses follow the data name. At least one of the clauses must be the PICTURE, TYPE, or OCCURS clause.

PICTURE Clause

The PICTURE clause describes the display coded storage characteristics of a data item. An elementary data item must be described with either the PICTURE clause or the TYPE clause. If both PICTURE and TYPE clauses are specified for a data item, the PICTURE clause is ignored, and a warning diagnostic is issued. The format of the PICTURE clause is shown in figure 2-13.

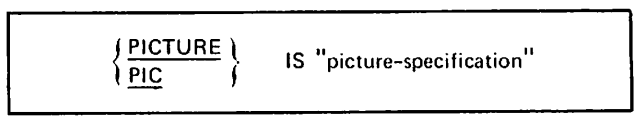


Figure 2-13. PICTURE Clause Format

The picture specification determines the size and class of a data item. The class can be described as either character or numeric. A character data item is described with the characters A, X, and 9 in the picture specification. The characters 9, V, . (decimal point), P, and T are used to describe a numeric data item. The picture specification can contain up to 30 characters. The characters comprising the picture specification must be enclosed in quotation marks. PIC is the legal abbreviation for PICTURE.

The size of the data item is indicated by the number of A, X, 9, . (decimal point), and T characters in the picture specification. The characters V and P are not counted in determining the size of the data item. Consecutive identical characters in the picture specification can be specified by following the character with an unsigned integer enclosed in parentheses. Each of the following picture specifications describes a data item with eight character positions:

```
99999999 or 9(8)

XXXXXXXX or X(8)

9999AAAA or 9(4)A(4)
```

A picture specification can contain a maximum of 30 characters including parentheses, but the actual data item can be larger than 30 characters. For example, a picture specification containing the character A repeated 75 times is too long, but A(75) is a valid description for a data item with 75 alphabetic characters. Parentheses can be used to indicate a repetition factor only with the characters A, X, P, and 9. Numeric data items can contain a maximum of 18 significant digits; additional leading or trailing zeros indicated by the character P can be specified for up to 30 characters.

Character Data Item

The picture specification to describe a character data item can contain the characters A, X, and 9. The size of the data item cannot exceed 32767 characters.

The function of each character in the picture specification for a character data item is as follows:

- A Each A in the picture specification represents a character position that can contain either a letter of the alphabet or a space (blank).
- X Each X in the picture specification represents a character position that can contain any character in the DDL display code character set. (Refer to appendix A.)
- 9 Each 9 in the picture specification represents a character position that can contain any decimal digit.

Each A, X, or 9 in the picture specification is counted in determining the size of the data item. Figure 2-14 illustrates some character data item picture specifications and the representation of actual data values.

Picture-Specification	Data Value	Display Code Stored
"AAAAA" or "A(5)"	COSTS	C O S T S
"AAAA" or "A(4)"	WXYZ	W X Y Z
"XXXXXXXX" or "X(8)"	ABCD-***	A B C D - * * *
"XXXXXXXX" or "X(8)"	123.4567	1 2 3 . 4 5 6 7
"AAAA999" or "A(4)9(3)"	ABCD123	A B C D 1 2 3

Figure 2-14. PICTURE Clause Character Data Items

Numeric Data Item

The picture specification for a numeric data item describes an arithmetic data item in display code format. A combination of the characters 9, V, . (decimal point), P, and T describes a numeric data item. The picture specification consists of one or two parts: a whole part and or a fractional part. Digit positions in each part must be represented by the characters 9 and T. A numeric picture specification is in fixed-point decimal format. A maximum of 18 significant digits can be described.

The function of each character in the picture specification for a numeric data item is as follows:

- 9 Each 9 in the picture specification represents a digit position that can contain a number. The 9 is counted in determining the size of the data item.
- V The character V is used in the picture specification to indicate the position of an assumed decimal point. A V as the rightmost character in the picture specification is redundant. The character V cannot appear more than once in the picture specification. Since the assumed decimal point does not occupy a character position, the V is not counted in the size of the data item.
- . A decimal point (.) in the picture specification indicates the position of an actual decimal point. The decimal point is counted in the size of the data item.
- P The character P in the picture specification indicates an assumed decimal point that is outside the number in the data item. One P is specified for each implied position between the rightmost character and the assumed decimal point or between the leftmost character and the assumed decimal point. Since P indicates an assumed decimal point, a V in the picture specification is redundant. The character

P is not counted in the number of significant digits (maximum of 18); however, it is counted in determining the total number of characters in the data item (maximum of 30).

T The character T in the picture specification indicates a digit position that contains a plus sign or minus sign combined with the digit. The T can be only the rightmost character in the picture specification. The plus sign (12-row punch) or minus sign (11-row punch) is combined with the digit punch. The representation of the sign in the rightmost digit is shown in figure 2-15.

Digit	0	1	2	3	4	5	6	7	8	9
Plus Representation†	<	A	B	C	D	E	F	G	H	I
Minus Representation†	V	J	K	L	M	N	O	P	Q	R

†The NOS system card reader does not support the character translation shown for +0 or -0.

Figure 2-15. Sign Representation in Rightmost Digit

If the picture specification does not contain a V, a decimal point, or a P, the decimal point is assumed at the immediate right and the data item is an integer. The data item is assumed to be a positive number when the picture specification does not contain the character T. Figure 2-16 illustrates some numeric picture specifications and the representation of actual data values.

Picture-Specification	Data Value	Display Code Stored
"999"	123	1 2 3
"99V999"	12345	1 2 3 4 5 (↑ after 2)
"99V99T"	+12345	1 2 3 4 E (↑ after 2)
"PPP99999"	.00012345	↑ 0 0 0 1 2 3 4 5
"PPP9(4)T"	-.00012345	↑ 0 0 0 1 2 3 4 N

Figure 2-16. PICTURE Clause Numeric Data Items

TYPE Clause

The TYPE clause describes the characteristics of a data item that is stored in the computer's internal binary format. An elementary item must be described with either the TYPE clause or the PICTURE clause. If both TYPE and PICTURE clauses are specified for a data item, the TYPE clause takes precedence and a warning diagnostic is issued. The format of the TYPE clause is shown in figure 2-17.

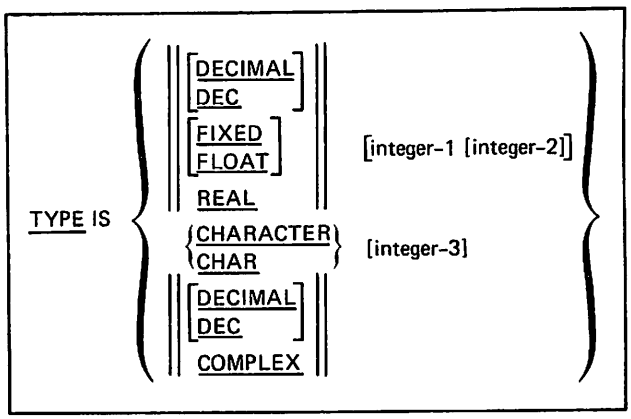


Figure 2-17. TYPE Clause Format

A data item described with the TYPE clause and a data item described with the PICTURE clause differ in internal representation of numeric data items. A data item described with the TYPE clause is stored in binary format; a data item described with the PICTURE format is stored in display code format. The TYPE clause should be used for data items that are primarily used for computation; the PICTURE clause should be used for data items that are primarily used for display. Character data items can also be described with the TYPE clause.

The DECIMAL, FIXED or FLOAT, REAL, and COMPLEX options can be used to describe a numeric data item. A character data item is described with the CHARACTER option. DEC is the legal abbreviation for DECIMAL; CHAR is the legal abbreviation for CHARACTER.

At least one of the reserved words DECIMAL (or DEC), FIXED, FLOAT, REAL, CHARACTER (or CHAR), or COMPLEX must be specified as a type option.

Numeric Data Item

The TYPE clause describes a numeric data item in terms of base, scale, mode, and precision

Base

The base of a data item is decimal; if the DECIMAL option is not specified, it is assumed by default.

Scale

Scale is described by either the FIXED or FLOAT option; if neither is specified, FIXED is assumed by default.

Mode

The mode of a data item is real; if the REAL option is not specified, it is assumed by default.

Precision

Precision is described by specifying one or two numbers; the first number (integer-1) indicates the number of significant digits and the second number (integer-2) indicates the position of an assumed decimal point.

A floating point data item has a mantissa and an exponent. The precision of the data item in the mantissa is specified by integer-1 (the number of decimal digits). Integer-1 must be in the range 1 through 29. Integer-2 must not be included when the FLOAT option is specified.

A fixed point data item is always a binary integer; integer-1 is in the range 1 through 18. A single precision data item is stored in one computer word; double precision data items require two words. Integer-1 specifies the maximum number of significant digits for the data item; it must be in the range 1 through 18.

A complex data item has a real part and an imaginary part. It always occupies two computer words. Precision cannot be specified for a complex data item.

The position of an assumed decimal point in a fixed point data item is specified by integer-2. The position is specified as follows:

A negative integer-2 locates the assumed decimal point the specified number of positions to the right of the rightmost actual digit.

A positive integer-2 that is less than or equal to integer-1 locates the assumed decimal point the specified number of positions to the left of the rightmost actual digit.

A positive integer-2 that is greater than integer-1 indicates that the data item is a fraction, and the assumed decimal point is located outside the actual number. The decimal point is located the specified number of positions to the left of the rightmost actual digit.

If integer-2 is not specified, it is assumed to be zero and the fixed point data item is an integer.

Integer-1 and integer-2 are optional. If no integer is specified, integer-1 is assumed to be 14 and integer-2 (for a fixed point data item only) is assumed to be zero.

A schema description that includes the TYPE clause with the FIXED option specified and with integer-1 designating the desired precision is equivalent to a subschema description that specifies COMPUTATIONAL-1. The TYPE clause with the FLOAT option specified and with integer-1 in the range 1 through 14 is equivalent to a subschema COMPUTATIONAL-2 description.

Figure 2-18 illustrates some TYPE clauses for numeric data items and the representation of actual data values.

Character Data Item

The TYPE clause can be used to describe a character data item. The CHARACTER option is specified with integer-3 designating the number of characters in the data item. If integer-3 is not specified, it is assumed to be 1. A maximum of 32767 characters can be specified for a character data item.

Figure 2-19 illustrates some TYPE clauses for character data items and the representation of actual data values.

TYPE Clause	External Representation	Decimal Value	Internal Representation
DECIMAL 3	123	123	0000 0000000000000173
FLOAT 14 FLOAT REAL	4.096E3	4096	1734 4000000000000000
FIXED 5,2	123.4	123.4	2000 0000000000030064
FLOAT 15	1	1	1720 4000000000000000 1654 0000000000000000
COMPLEX	(1.1,3.2)	(1.1,3.2)	1720 4314631463146315 1721 6314631463146315

Figure 2-18. TYPE Clause Numeric Data Items

TYPE Clause	External Representation	Internal Representation
CHARACTER 2	AA	AA
CHARACTER 10	ABCDEFGHIJ	ABCDEFGHIJ
CHARACTER 5	COSTS	COSTS

Figure 2-19. TYPE Clause Character Data Items

OCCURS Clause

The OCCURS clause is used to describe a data item that is repeated a number of times within a record. A repeating elementary item is a vector; all occurrences of the elementary item are identical in every respect except actual value. A collection of data items can be specified as a repeating group; the entire collection is repeated a number of times. The format of the OCCURS clause is shown in figure 2-20.

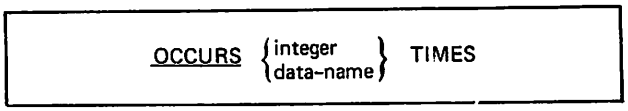


Figure 2-20. OCCURS Clause Format

The data item described with the OCCURS clause is either a fixed occurrence data item or a variable occurrence data item. The integer option specifies the exact number of times the data item is repeated in each record occurrence (a fixed occurrence data item). The specified integer must be greater than zero. The data name option references a data item whose current value represents the number of occurrences (a variable occurrence data item). The data item referenced by the data name must conform to the following rules:

- It must be an elementary item within the same record as the variable occurrence data item.
- It must be described as an integer.
- The CHECK IS VALUE clause must be specified to denote the range of occurrences; the lower limit of the range must be greater than or equal to zero.
- It cannot be within a variable occurrence data item.
- It cannot be described with a VIRTUAL RESULT clause.

The data name specified in the OCCURS clause cannot be subscripted. A variable occurrence data item should be the last item in the record. If it is a repeating group item, only subordinate entries can follow the entry containing the OCCURS data name TIMES clause.

If a record description entry contains a variable occurrence data item, certain restrictions are placed on the area under the following conditions:

- The area has more than one record description entry.
- The FILE control statement for the area specifies RT=T (CYBER Record Manager record type is trailer count).

When these two conditions exist, all record description entries for the area must contain a variable occurrence data item at the same location in the record and must specify the same maximum size. The data item that specifies the number of occurrences must be in the same location in each record description entry; the data description entry for these data items must also specify the same size and description.

The OCCURS clause cannot be included in a data description entry that also includes the RESULT clause. Figure 2-21 illustrates some examples of the OCCURS clause.

Two types of repeating data items can be specified: vectors and repeating groups. A repeating data item can be subordinate to another repeating data item with the following restrictions:

- Repeating data items can be nested up to three levels.
- A variable occurrence data item cannot be subordinate to another repeating data item.

Vectors

A vector is an elementary data item that is repeated a number of times. It can be a fixed or variable occurrence data item. A data item that is a vector must be described with the OCCURS clause and either the PICTURE or TYPE clause. A vector can be subordinate to a repeating group.

Repeating Groups

A repeating group is a collection of data items that is repeated a number of times. The collection can consist of elementary items, vectors, and other repeating groups. Subordinate entries describe the data items in the repeating group. These entries must have a level number that is greater than the repeating group entry level number. When a repeating group contains other repeating groups, entries at the same level must have the same level number.

NOTE

Refer to appendix F for recommendations on the use of repeating groups.

RESULT Clause

The RESULT clause specifies that the value of the data item is established by the execution of a data base procedure. The value of the data item can be changed only by execution of the data base procedure. The time the procedure is invoked depends upon whether the ACTUAL or VIRTUAL option is specified. The format of the RESULT clause is shown in figure 2-22.

Example 1

```
01  PAYMENTS          TYPE DECIMAL FIXED 6,2
                         OCCURS 12 TIMES.
```

The data item PAYMENTS is a vector; it is an elementary data item that is repeated 12 times in each record.

Example 2

```
01  MONTHLY-ORDERS          OCCURS 12 TIMES.
    03  NUM-ORDERS          TYPE DECIMAL FIXED 2.
    03  TOTAL-AMT           TYPE DECIMAL FIXED 6,2.
```

The data item MONTHLY-ORDERS is a repeating group that occurs 12 times in each record. It consists of two subordinate data items that represent one occurrence of the repeating group.

Example 3

```
01  NUM-ITEMS               PICTURE "99"
                                CHECK VALUE 1 THRU 15.
01  ITEM                    OCCURS NUM-ITEMS TIMES.
    03  QUANTITY            PICTURE "9(4)".
    03  DESC-A              PICTURE "X(16)".
    03  UNIT-PRICE          TYPE DECIMAL FIXED 5,2.
    03  EXT-PRICE           TYPE DECIMAL FIXED 6,2.
```

The data item ITEM is a repeating group that occurs a variable number of times in each record. The data item NUM-ITEMS contains the number of occurrences for the record; the number must be in the range 1 through 15.

Example 4

```
01  QUARTER-TOTS            OCCURS 4 TIMES.
    03  MNTH                TYPE DECIMAL FIXED 5,2
                               OCCURS 3 TIMES.
    03  CUM-TOT             TYPE DECIMAL FIXED 6,2.
```

The repeating group QUARTER-TOTS contains a vector (MNTH) and a nonrepeating elementary data item (CUM-TOT). Both subordinate data items are at the same level and must have the same level number.

Figure 2-21. OCCURS Clause Examples

IS { ACTUAL / VIRTUAL } RESULT OF data-base-procedure

Figure 2-22. RESULT Clause Format

A data item can be described with either the ACTUAL RESULT or the VIRTUAL RESULT clause. The data item described with the RESULT clause cannot be a repeating data item (vector or repeating group). It cannot be subordinate to a repeating group data item.

The ACTUAL option specifies that the data item is physically included in the record, and that a value is always stored in the record. When a new record is stored in the data base, the data base procedure is executed to establish a value for the data item. Subsequently, the specified data base procedure is executed to update the data item each time the record is modified. If the ACTUAL option is specified in the RESULT clause, the ENCODING clause cannot be included in the same data description entry.

The VIRTUAL option specifies that the value of the data item is established when the data item is requested by an applications program. The data item is not physically stored in the record; therefore, the size of the data item should not be included when the maximum record length (MRL) is calculated for the area. The data item is stored in the CDCS buffer when a user requests the record, and the buffer size must be large enough to include the data item. The specified data base procedure is invoked when a user GET function (READ statement) includes the data item. When the VIRTUAL option is used to describe a data item, the following restrictions apply:

- Neither the ENCODING clause nor the DECODING clause can be used to describe the data item.
- A CALL clause cannot be included in the data description entry.
- The data item cannot be referenced in an OCCURS data name TIMES clause.

Figure 2-23 illustrates the processing of a data item described with the VIRTUAL RESULT clause.

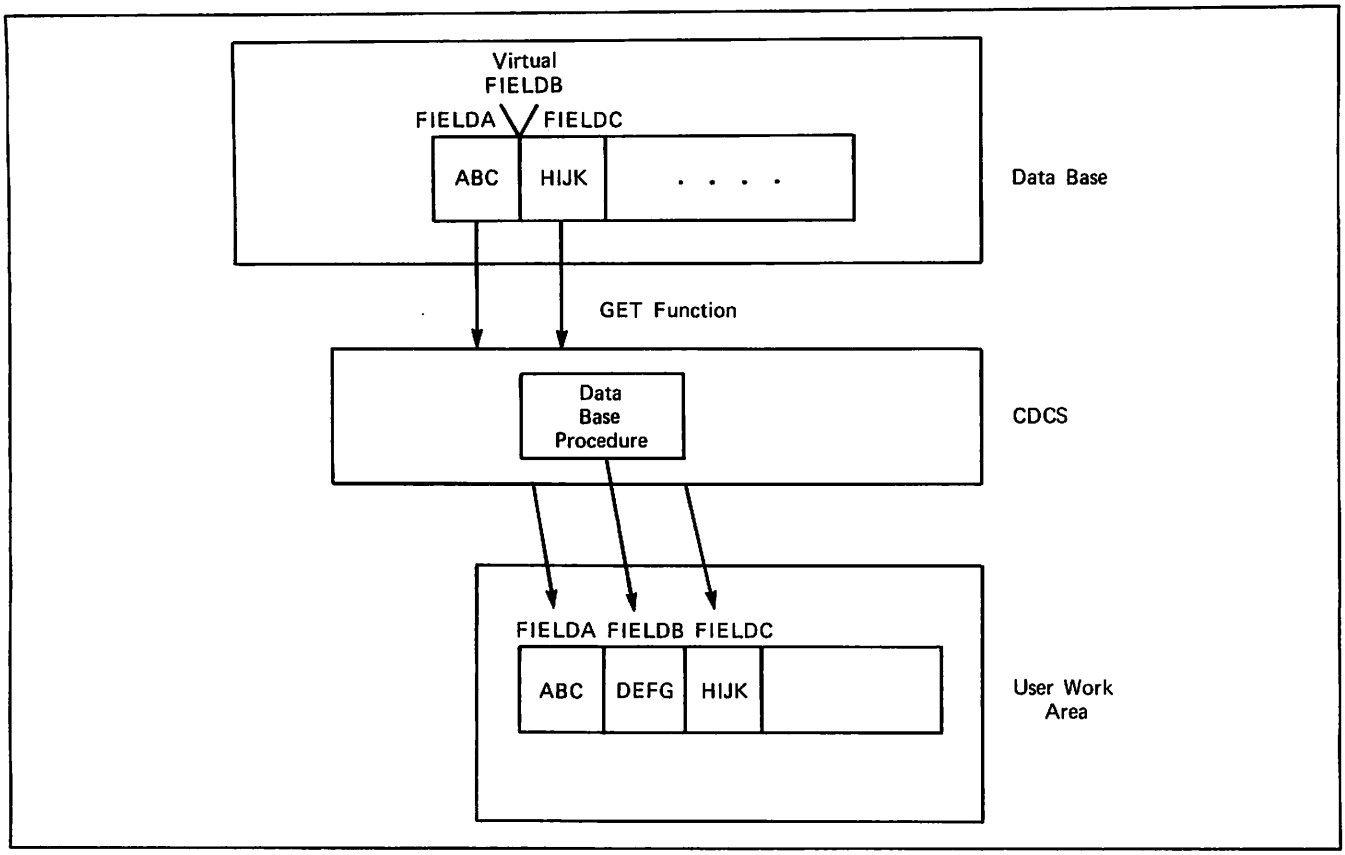


Figure 2-23. Virtual Data Item Processing

CHECK Clause

The CHECK clause imposes certain restrictions on the value of the data item. One or more of the following functions can be specified for the data item:

Inhibiting data conversion

Checking validity of the data item whenever a value is changed or added

Restricting the value of the data item

If the value of the data item does not satisfy the condition of the CHECK clause, an error is reported to the user. The format of the CHECK clause is shown in figure 2-24.

Each of the three options in the CHECK clause can be specified once. Multiple options can appear in any order. PIC is the legal abbreviation for PICTURE. The data item described with the CHECK clause must also be described with either the PICTURE clause or the TYPE clause; however, the data item cannot be described as TYPE COMPLEX.

The PICTURE option specifies that data conversion between the schema and the subschema is not allowed for the data item. The characteristics of the data item in the subschema must match the characteristics of the data item in the schema.

The data base procedure option causes the specified procedure to be executed when a value for the data item is added or changed. The specified procedure performs validity checking on the value. If both the data base procedure and VALUE options are specified, the VALUE check is performed first; the procedure is executed only if the value of the data item is valid.

```
CHECK IS { PICTURE }
         { PIC     }
         data-base-procedure
         VALUE [NOT] literal-1 [THRU literal-2] [literal-3 [THRU literal-4]] ...
```

Figure 2-24. CHECK Clause Format

The VALUE option places a restriction on the actual value of the data item. The actual value is compared with the values specified in the VALUE option. If the actual value satisfies the comparison and the NOT option is omitted, the value is valid. When VALUE NOT is specified, an actual value that satisfies the comparison is not valid. Values for the comparison are specified as individual values or as ranges of values. An individual value is specified by literal-1; a range of values is specified by literal-1 THRU literal-2. Literals must be specified in ascending order according to the collating sequence specified for the area. (Refer to the SEQUENCE clause in the data control entry.)

Validity checking is performed when a user function includes the data item. The specific function that causes the checking to be performed is determined as follows:

If the RESULT clause is not used to describe the data item, validity checking occurs when a STORE or MODIFY function (WRITE or REWRITE statement) adds or changes the value of the data item.

If the VIRTUAL RESULT clause is used to describe the data item, validity checking occurs when a GET function (READ statement) includes the data item.

If the ACTUAL RESULT clause is used to describe the data item, validity checking occurs when a new record is added, or when the data item is updated.

When the data item requires other procedures (such as a CDCS conversion routine or a data base procedure specified in a RESULT clause) the procedures are executed before validity checking occurs.

Literals specified in the VALUE option must correspond to the type or picture specification for the data item. Nonnumeric literals (enclosed in quotation marks) must be specified when the data item is described with the TYPE CHARACTER clause or with a PICTURE clause containing the character A or X. If the data item is described with a TYPE FIXED or TYPE FLOAT clause or with a PICTURE clause that does not contain any A or X characters, a numeric literal must be specified. Additional rules that apply to literals in the VALUE option are as follows:

The last character in a numeric literal must not be a decimal point.

The actual sign, not a sign overpunch, must be specified in the literal.

If the data item is not signed (the character T is not specified in the picture specification) and the literal is signed, the sign is ignored and the literal is assumed to be positive.

If the fractional part of a literal is greater than the data item specification, the literal is rounded off to the fractional length of the data item. For a data item described as PICTURE "99.999" or TYPE FIXED 5,3, the literal 20.5555 is rounded off to 20.556.

If the literal contains a fractional part and the data item is an integer, the literal is rounded off to an integer (1234.55 becomes 1235).

It is recommended that the literal conform to the data item specification so that the last three rules need not be invoked. Figure 2-25 illustrates valid literals for some typical data item specifications. The values shown for the CHECK VALUE clause are literals that could be specified for the PICTURE and TYPE specifications given.

ENCODING/DECODING Clause

The ENCODING/DECODING clause is used when the data item requires nonstandard conversion. The conversion is performed by a data base procedure when the data item is stored, retrieved, or updated. The format of the ENCODING/DECODING clause is shown in figure 2-26.

Both the ENCODING clause and the DECODING clause can be specified for the same data item. The data item described with the ENCODING/DECODING clause must be an elementary data item that is also described with either the TYPE clause or the PICTURE clause.

The ENCODING clause invokes the conversion procedure when a STORE function (WRITE statement) adds a new value for the data item to the data base, or when a MODIFY function (REWRITE statement) alters an existing value. The value is converted from the subschema representation to the schema representation. A data item described with the ENCODING clause cannot be described with either the ACTUAL RESULT or the VIRTUAL RESULT clause.

The DECODING clause invokes the conversion procedure when a GET function (READ statement) requests the data item from the data base. The value is converted from the schema representation to the subschema representation. A data item described with the DECODING clause cannot be described with the VIRTUAL RESULT clause.

The ALWAYS option can be included to specify that the data base procedure is to be executed even if the characteristics of the data item do not differ between the schema and the subschema. If ALWAYS is omitted from the clause, conversion occurs only when record mapping is required for the data item.

The specified data base procedure is executed instead of a standard conversion procedure. The representation of the value is changed, not the value itself.

CALL Clause (Data Description Entry)

The CALL clause in a data description entry causes a data base procedure to be executed when a specified user function is performed on the data item. A user function is a request from an application program to perform an operation that involves the data item. This clause can be specified only for an elementary data item. The procedure can be invoked before or after the function is performed or when an error is detected during the performance of the function. The same procedure can be specified in separate CALL clauses. The format of the CALL clause is shown in figure 2-27.

PICTURE or TYPE Clause	Examples of Literals for CHECK VALUE Clause
Numeric picture-specifications	
PICTURE "9999V999"	1234 +1234.12 12.1235 (rounded off to 12.124)
PICTURE "999.99T"	123.34 -123 +123.555 (actual sign must be used)
PICTURE "PPP9999"	.0001234, .0001 (decimal point and leading zeros must be specified)
PICTURE "999PPP"	123000, 1000, 0010000 (trailing zeros must be specified with no decimal point)
Nonnumeric picture-specifications	
PICTURE "X(5)9(5)"	"ABCDE12345", "A•X-$11111" (nonnumeric literals must be enclosed in quotation marks)
PICTURE "A(10)"	"AAAA" "ABCD EFGH"
Floating point numeric type	
TYPE FLOAT	281 99999.999
Fixed point numeric types	
TYPE FIXED 5,7	.0012345, .0000001, .0010 (decimal point and leading zeros must be specified)
TYPE FIXED 5,-2	1234500, 500 (trailing zeros must be specified with no decimal point)
TYPE FIXED 14	55555.55 (rounded off to 55556) -50000 any integer not greater than 14 digits in length
TYPE FIXED 14,2	any numeric literal not greater than 14 digits in length
TYPE FIXED 18	any integer not greater than 18 digits in length
TYPE FIXED 18,4	any numeric literal not greater than 18 digits in length
Nonnumeric type	
TYPE CHARACTER 5	"AAAAA", "A-123" (nonnumeric literals must be enclosed in quotation marks)

Figure 2-25. Examples of Valid Literals in the CHECK VALUE Clause

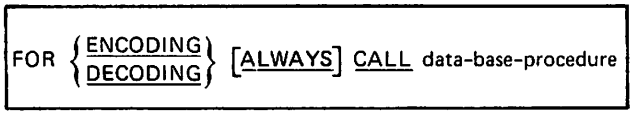


Figure 2-26. ENCODING/DECODING Clause Format

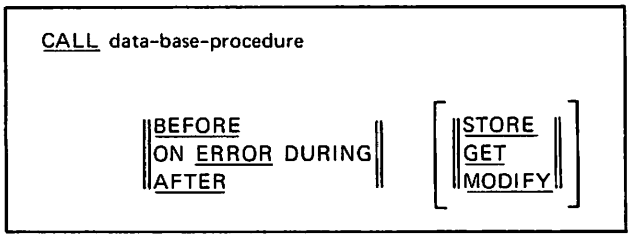


Figure 2-27. CALL Clause Format (Data Description Entry)

The BEFORE, ERROR, and AFTER options determine when the data base procedure is invoked.

BEFORE

The procedure is invoked immediately before record mapping occurs.

ERROR

The procedure is executed each time CDCS detects and reports an error during the performance of the specified function.

AFTER

The procedure is executed immediately after record mapping occurs.

At least one of these options must be specified; each option can be specified once in the CALL clause. If two or three options are included in the clause, the data base procedure is executed at each designated time. The only exception occurs when the ERROR and AFTER options are specified and an error causes execution of the data base procedure. In this situation, the data base procedure specified in the AFTER option is not executed. In the following example, the data base procedure SLEDIT is executed before the STORE function is performed, and whenever CDCS detects an error during the performance of the STORE function.

```
CALL SLEDIT BEFORE ERROR STORE
```

The user function specified in the CALL clause determines the type of data manipulation that causes the data base procedure to be executed at the designated time. Three functions can be specified:

STORE

The procedure is executed when a user function (WRITE statement) stores a new value for the data item.

GET

The procedure is executed when a user function (READ statement) places the data item in the user work area.

MODIFY

The procedure is executed when a user function (REWRITE statement) causes the value of the data item to be changed.

Each option can be specified once in a CALL clause. Separate CALL clauses for individual user functions can invoke the same or different data base procedures. If no user function is specified in the CALL clause, the data base procedure is executed at the designated time for each of the three user functions.

Multiple data base procedures can be executed at a designated time. When more than one CALL clause for the same user function specifies the same time to invoke a procedure (before, during, or after the user function is performed), the procedures are executed in the order the CALL clauses are stated in the data description entry. In the following example, the data base procedure GETCHK is executed, and then the procedure GETVER is executed before the GET function is performed.

```
CALL GETCHK BEFORE GET
CALL GETVER BEFORE GET
```

DATA CONTROL ENTRY

Certain attributes of the schema areas are specified in the data control entry. The data control entry follows the last record description entry in the schema. The format of the data control entry is shown in figure 2-28.

```
DATA CONTROL.

    {area control entry.} ...
```

Figure 2-28. Data Control Entry Format

The data control entry supplements information in FILE control statements for the areas in the schema. Information provided in a FILE control statement does not require corresponding information in the area control entry in the data control entry. For more information, refer to the FILE Control Statement subsection, which appears later in this section.

AREA CONTROL ENTRY (DATA CONTROL ENTRY)

The area control entry supplies special information related to an area of the schema. The following information can be specified in the entry:

The use of compression and decompression procedures

The data items representing the primary and alternate keys

The collating sequence

The means for distinguishing among multiple DDL record types within the area

An area control entry with a KEY clause is required for each area. The format of the area control entry is shown in figure 2-29.

```
AREA NAME clause

    [COMPRESSION/DECOMPRESSON clause]

    {KEY clause} ...

    [SEQUENCE clause]

    [RECORD CODE clause] .
```

Figure 2-29. Area Control Entry Format

AREA NAME Clause

The AREA NAME clause specifies the area for which the special information is being supplied. This clause must be the first clause in an area control entry. The format of the AREA NAME clause is shown in figure 2-30.

```
AREA NAME IS area-name
```

Figure 2-30. AREA NAME Clause Format

The area name specified in this clause must reference an area previously established in the schema by an area description entry. Subsequent clauses in the area control entry apply to the area name specified in the AREA NAME clause.

COMPRESSION/DECOMPRESSION Clause

The COMPRESSION/DECOMPRESSION clause provides for the reduction of record length and for the restoration of records to the original state. When the clause is omitted, no compression or decompression of records is performed. However, if either COMPRESSION or DECOMPRESSION is specified, the other must be specified in the same clause or in the clause immediately following. COMPRESSION and DECOMPRESSION can be specified only once in an area control entry. The format of the clause is shown in figure 2-31.

When COMPRESSION or DECOMPRESSION is specified, either SYSTEM or PROCEDURE must be specified. The SYSTEM option refers to a procedure supplied by CYBER RECORD Manager. The PROCEDURE option designates a data base procedure to perform the compression or decompression of records. When the SYSTEM option is used, it must be designated for both compression and decompression operations for the particular area. Similarly, if the PROCEDURE option is used, it must be designated for both compression and decompression operations for the area. However, either the same or different data base procedures can be designated. The procedure name can have a maximum of seven characters and must conform to all the rules governing procedure names as defined in this manual. When different procedures are specified, two clauses must be entered. The use of this option is shown in figure 2-32.

KEY Clause

Format 1 of the KEY clause, in figure 2-33, specifies the data items that are the primary and alternate keys for a record. A key clause specifying the primary key is required for each area. Key clauses specifying alternate keys are optional. Format 2, also shown in figure 2-33, can be used to specify the data items that are to be concatenated to produce the primary or alternate record key.

The data name specified in the KEY clause (format 1) designates a data item that is a record key. The data name must be qualified if it appears in more than one record type in the schema. The data item cannot be described as TYPE COMPLEX. Subscripting the data name is not allowed; if data name is a member of a repeating data item, different rules apply depending on whether a primary or alternate key is being defined. Refer to the following subsections about primary and alternate keys. The specified data name can be nested only one level in a repeating data item. If an area

Example 1

```
FOR COMPRESSION USE PROCEDURE CI01
FOR DECOMPRESSION USE PROCEDURE DCI01
```

Example 2

```
FOR COMPRESSION DECOMPRESSION USE SYSTEM
```

Figure 2-32. Examples of the COMPRESSION/DECOMPRESSION Clause

contains multiple record types, only data items from the first record type can be specified as keys for the area.

The data name refers to a data item that contains the record key. The data item must be a data item described in the area named in the preceding AREA NAME clause (primary or alternate key).

The maximum length allowed for a data item designated as a primary key depends on the file organization. For indexed sequential and direct access files, a primary key cannot exceed 240 characters. For actual key files, a primary key cannot exceed 8 characters. The maximum length allowed for a data item designated as an alternate key is the same for all the previously mentioned file organizations; an alternate key cannot exceed 240 characters.

FILE control statement parameters related to the key must be accurate descriptions of the data item; any parameter that does not agree with the data item description is overwritten with the data item description value, and a diagnostic is issued. Refer to the FILE Control Statement subsection which appears later in this section.

Primary Key

The KEY IS data name clause designates the primary key for a record; the primary key must be specified before any alternate keys are specified. The specified data name must refer to a data item in a record within the area named in the preceding AREA NAME clause. If the data name is a member of a repeating data item, the first occurrence of the data item is referenced as the key. A primary key for an area with actual key file organization must be described as TYPE FIXED with a range of 1 through 8. For example, a key with a length of five digits is described as TYPE FIXED 5.

The USING option can be used only in the primary key specification for an area with direct access file organization. This option specifies a data base procedure which performs a hashing operation, using the primary key, to derive the storage address of a record. The data base procedure name cannot exceed seven characters in length and must conform to all the rules governing procedure names as defined in this manual.

```
FOR [ COMPRESSION ] [ USE { SYSTEM                          }
    [ DECOMPRESSION ]     { PROCEDURE data-base-procedure   }
```

Figure 2-31. COMPRESSION/DECOMPRESSION Clause Format

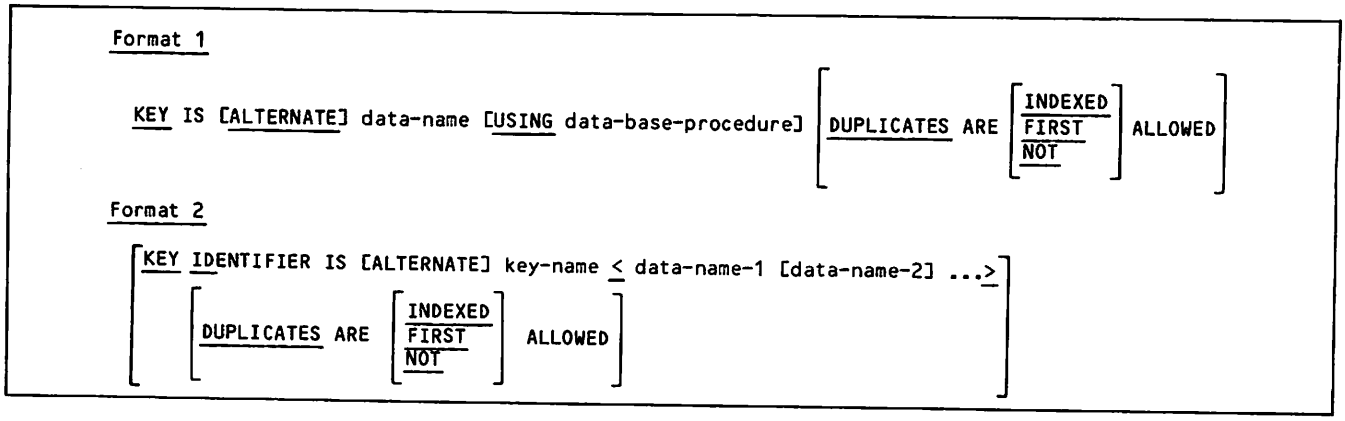


Figure 2-33. KEY Clause Format

Alternate Key

The KEY IS ALTERNATE data name clause designates an alternate key for a record; designation of any alternate keys must follow designation of the primary key. Multiple alternate keys can be specified by including a KEY clause for each alternate key. The data name must refer to a data item in a record within the area specified in the preceding AREA NAME clause. If the data name is a member of a repeating data item, all occurrences of the data item are referenced as a key.

NOTE

Refer to appendix F for recommendations on the use of alternate keys on repeating groups.

The DUPLICATES option can be specified for an alternate key. This option can be specified as follows:

DUPLICATES ARE INDEXED or
DUPLICATES ARE ALLOWED

Duplicate alternate keys are allowed and are stored in the index file in order according to the primary key to which the alternate key corresponds.

DUPLICATES ARE FIRST

Duplicate alternate keys are allowed and are stored in the index file in the order of occurrence (first in, first out).

DUPLICATES ARE NOT ALLOWED

Duplicate alternate keys are not allowed.

If the DUPLICATES option is not specified for an alternate key, duplicate keys are not allowed.

Concatenated Key

The KEY IDENTIFIER clause shown in format 2 (figure 2-33) designates a concatenated key for a record. The concatenated key can be a primary or an alternate key. A concatenated key cannot be used for actual key files.

The key name is a user-defined name used to identify the concatenated key. It represents a key in the specified area, whose size is equal to the sum of the sizes of the data items specified in the list of data names and whose beginning position is the beginning position of the first data item specified. Key name must conform to the rules governing data names defined in the User-Defined Names subsection which appears earlier in this section. Subscripting and qualification of the key name are not allowed. The specified key name must be unique among all key names and data names defined in the schema.

The data names specified in the KEY clause designate up to 64 contiguous elementary data items; the items cannot be repeating data items. All data names must be in the same record type in the area and must be specified in the same order in which they are defined in the record. A data name can be qualified by including the record name in which it appears. The data names specified in the list cannot be data items that are separated by padding caused by word-aligned values. A less than symbol (<) and a greater than symbol (>) must enclose the list of data names.

For file creation, the concatenated key is treated as a symbolic key for collating purposes. A symbolic key consists of 1 to 240 6-bit alphanumeric characters.

The ALTERNATE option designates an alternate concatenated key. Multiple alternate keys can be specified by including a KEY clause for each alternate key.

All keys in an area must be unique with respect to both size and position. For example, if two keys have the same starting position, they cannot have equal sizes.

SEQUENCE Clause

The SEQUENCE clause specifies the collating sequence for the data in the area. The format of the SEQUENCE clause is shown in figure 2-34.

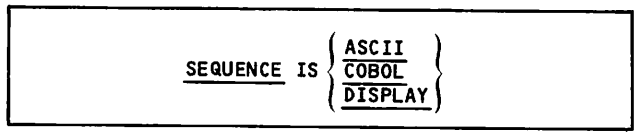


Figure 2-34. SEQUENCE Clause Format

The collating sequence for the area is specified as ASCII, COBOL, or DISPLAY. (Refer to appendix I for a listing of these collating sequences.) The DISPLAY option must be specified when the area is a Query Update catalog file. If the SEQUENCE clause is not specified, the COBOL collating sequence is assumed by default.

NOTE

Refer to appendix F for recommendations on the use of collating sequence.

RECORD CODE Clause

The RECORD CODE clause provides the means to distinguish between DDL record types when the area contains more than one record type. The record type can be determined by the value of a data item in the record, or by the execution of a data base procedure. This clause must be specified if the area has more than one record type. The format of the RECORD CODE clause is shown in the figure 2-35.

The BY data name option specifies the data item that contains the record code. The specified data name must refer to an elementary data item; data name cannot be subscripted, and it must appear in at least one record type in the area. If the data name appears in more than one record type, it must be qualified to provide a unique identifier. The data item containing the record code must be in the same location in each record type and must be of the same length and type (display fixed point, display numeric, etc.). The data name in each record type need not be the same.

The value that determines the record type must be specified for each record type in the area. The VALUE phrase is specified for each record type in the area (record-name-1, record-name-2, etc.); the actual value that identifies the record type is specified as literal-1, literal-2, and so forth. Each literal must have a unique value within the area and must be able to be converted to the type or picture specification of the data item. If the data item is described with the CHECK VALUE clause, the RECORD CODE literal must be the same as the CHECK VALUE literal; only one literal can be specified in the CHECK VALUE clause in each record. The literal in the RECORD CODE clause cannot exceed 240 characters.

The PROCEDURE option specifies a data base procedure that is performed to determine the record type. The data base procedure name can be a maximum of seven characters.

The specified data base procedure examines the record and returns to CDCS an integer value that designates the record type. The integer returned by the procedure is tested against the integers specified in the VALUE phrases. The VALUE phrase containing the matching integer designates the record name of the record type to be used. One VALUE phrase must be specified for each record type in the area. The integers must be in the range 1 through 1023. Each integer specified must be unique.

Figure 2-36 illustrates three methods of using the RECORD CODE clause.

NOTE

Refer to appendix F for recommendations on the use of multiple record descriptions.

CONSTRAINT ENTRY

A constraint defines a dependency condition between two areas or between data items within the same area. Update operations must satisfy the condition before updating an area is allowed. Thus, constraints protect the integrity of the data base. A constraint entry must be included in the schema source program for each constraint defined for the data base.

A constraint entry is an optional entry. If used, it must be included in the schema source program following the data control entry and preceding any relation entry.

A constraint entry begins with a constraint name statement. This is followed by a statement specifying the data items that associate the areas in a dependent condition. One data item establishes a record as being a parent record; the other data item establishes the other record as being a child record.

A constraint can involve only one area or can associate two areas. An area can be involved in more than one constraint. The maximum number of constraints allowed is 4095.

Figure 2-37 diagrams the dependency condition established between areas in constraints. The arrows between the areas indicate the dependency condition; the arrow points to the parent. Example 1 illustrates four constraints associating four areas. Example 2 illustrates two areas associated in two constraints. In this example, the dependency condition must be in the same direction in both constraints.

```
                                  ⎧BY data-name  VALUE FOR record-name-1 IS literal-1      ⎫
                                  ⎪              [VALUE FOR record-name-2 IS literal-2] ...⎪
RECORD CODE IS                    ⎨PROCEDURE data-base-procedure                          ⎬
                                  ⎪              VALUE FOR record-name-3 IS integer-3     ⎪
                                  ⎩              [VALUE FOR record-name-4 IS integer-4] ...⎭
```

Figure 2-35. RECORD CODE CLAUSE Format

Example 1

```
SCHEMA NAME IS MANUFACTURING-DB.
AREA NAME IS EMPLOYEE.
.
.
.
RECORD NAME IS EMPLOYEE-REC WITHIN EMPLOYEE.
01  REC-TYPE     PICTURE "A".
.
.
.
RECORD NAME IS PAYROLL-REC WITHIN EMPLOYEE.
01  REC-TYPE     PICTURE "A".
.
.
.
DATA CONTROL.
AREA NAME IS EMPLOYEE
    RECORD CODE IS BY REC-TYPE OF EMPLOYEE-REC
    VALUE FOR EMPLOYEE-REC IS "A"
    VALUE FOR PAYROLL-REC IS "B".
```

The record code is contained in the data item REC-TYPE, which is in the first character position in both records. An A in REC-TYPE identifies EMPLOYEE-REC; a B in REC-TYPE identifies PAYROLL-REC.

Example 2

```
SCHEMA NAME IS SCHLIST.
AREA NAME IS PRICELIST.
.
.
.
RECORD NAME IS PRICES WITHIN PRICELIST.
01  DATE-1       PICTURE "X(10)".
01  PRTAG        TYPE FIXED.
.
.
.
RECORD NAME IS INVENTORY-NUMBERS WITHIN PRICELIST.
01  DATE-1       PICTURE "X(10)".
01  INTAG        TYPE FIXED.
.
.
.
DATA CONTROL.
AREA NAME IS PRICELIST
    RECORD CODE IS BY PRTAG
    VALUE FOR PRICES IS 1
    VALUE FOR INVENTORY-NUMBERS IS 77.
```

The record code is contained in the second field, which begins in character position 11 in both records. A 1 in PRTAG identifies PRICES; a 77 in INTAG identifies INVENTORY-NUMBERS.

Example 3

```
SCHEMA NAME IS NURSERY-DB.
AREA NAME IS HISTORY.
.
.
.
RECORD NAME IS REGIONAL WITHIN HISTORY.
.
.
.
RECORD NAME IS NATIONAL WITHIN HISTORY.
.
.
.
RECORD NAME IS STATE WITHIN HISTORY.
.
.
.
DATA CONTROL.
AREA NAME IS HISTORY
    RECORD CODE IS PROCEDURE HSTCODE
    VALUE FOR REGIONAL IS 2
    VALUE FOR NATIONAL IS 1
    VALUE FOR STATE IS 3.
```

The record code is determined by the data base procedure HSTCODE. The procedure is executed and returns an integer that identifies the record type: a 1 designates NATIONAL, a 2 designates REGIONAL, or a 3 designates STATE.

Figure 2-36. Examples of the RECORD CODE Clause

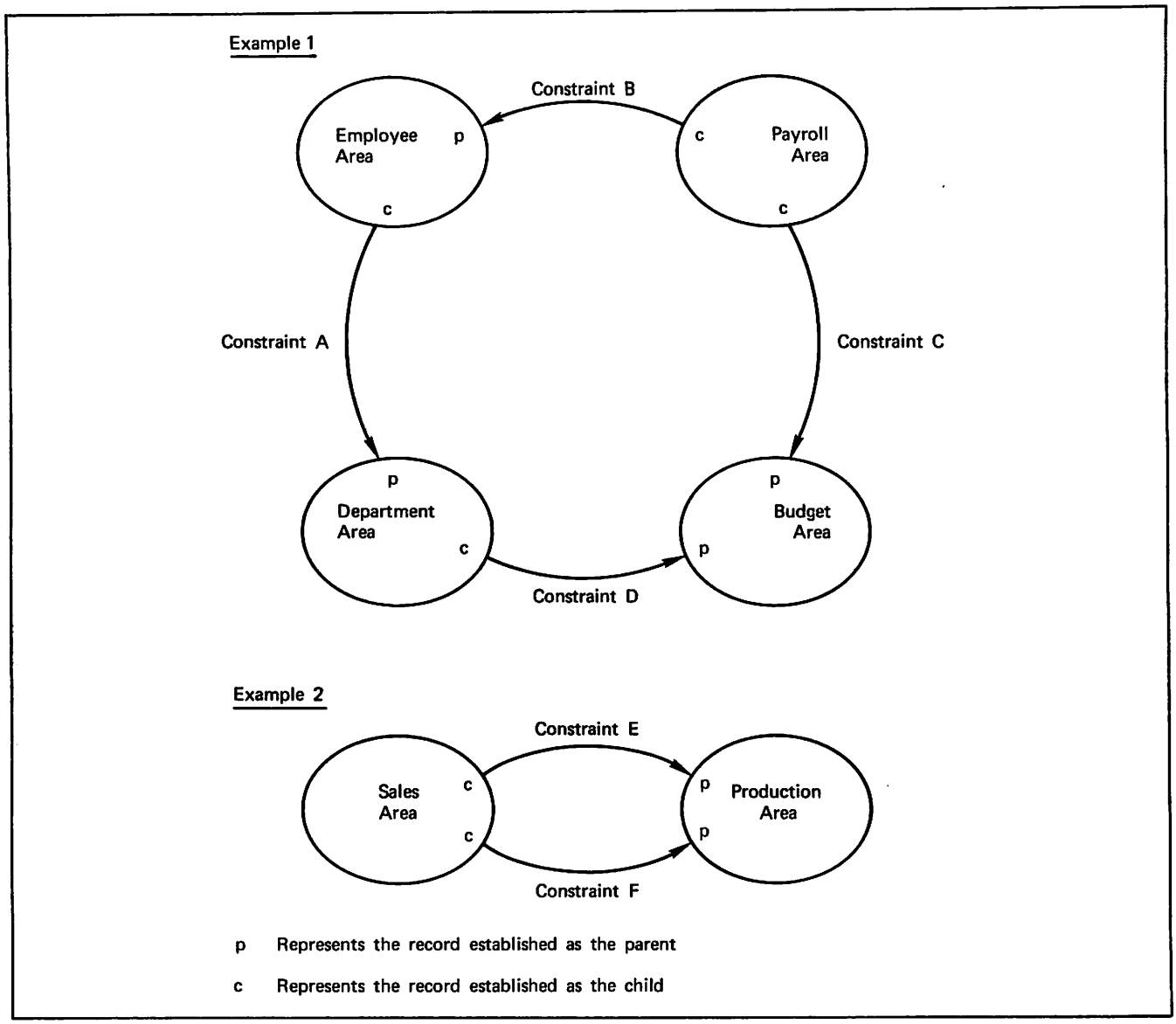


Figure 2-37. Dependency Conditions Established by Constraints

A series of constraints cannot result in a cycle. A cycle is a directed path starting from a record designated as a child and terminating at that starting record with it being a parent, also. If, in example 1 in figure 2-37, the dependency condition were reversed in constraint C, the series of constraints would result in a cycle, which is not allowed. Similarly in example 2, if one of the dependency conditions were reversed, the constraints would result in a cycle.

The format of the constraint entry is shown in figure 2-38. The constraint entry assigns a name to the constraint and specifies the data items involved in the constraint.

```
CONSTRAINT NAME IS constraint-name
    data-name-1 DEPENDS ON data-name-2.
```

Figure 2-38. Constraint Entry Format

CONSTRAINT NAME Clause

A constraint entry begins with the CONSTRAINT NAME clause. The name must be unique among all constraint names and area names in the schema. It can be up to 30 characters in length. The name can contain alphabetic and numeric characters and hyphens as long as the first character is alphabetic; consecutive hyphens are not allowed. The format of the CONSTRAINT NAME clause is shown in the constraint entry.

The name specified in this clause is associated with the dependency condition established by the items in the DEPENDS ON clause. An area that is involved in a constraint can contain only one DDL record type.

DEPENDS ON Clause

The DEPENDS ON clause is required when a CONSTRAINT name clause is specified. The DEPENDS ON clause designates the data items that are involved in the constraint. The format of the DEPENDS ON clause is shown in the constraint entry.

The order of the data items determines the direction of the dependency condition. The constraint condition established is that the record containing data-name-2 is the parent record and the record containing data-name-1 is the child record. However, data-name-1 and data-name-2 can be contained within the same record description.

The data items in the DEPENDS ON clause must meet a number of requirements.

They must be keys.

They must have identical characteristics: that is, they must be defined with identical TYPE or PICTURE clauses.

Data-name-1 must reference a data item designated as a primary key or an alternate key with duplicate values permitted.

Data-name-2 must reference a data item designated as a primary key or an alternate key with no duplicates permitted.

If one is a concatenated key, the other must be a concatenated key or a non-repeating group defined as a key.

If a data-name-1 or data-name-2 requires qualification, the name of the record to which it belongs must be used.

Figure 2-39 shows a constraint clause in a schema. Included are the entries necessary for supporting the constraint entry.

RELATION ENTRY

A relation defines a directed path joining areas described in the schema to allow retrieval of data from two or more areas at a time. An area can be joined only once in any one relation. Figure 2-40 illustrates five areas in a data base and several relations that could be defined for them; the arrows and relation sequences denote the directions of the relations.

A relation entry is included in the schema source program for each relationship defined in the data base. This entry assigns a name to the relation and specifies the data items to be used as join terms. The format of the relation entry is shown in figure 2-41.

```
SCHEMA NAME IS PERSONNEL.

AREA IS DEPARTMENT.
AREA IS EMPLOYEE.


RECORD IS DEPT-REC WITHIN DEPARTMENT.

01  DEPT-NO      PICTURE IS "X(5)".
        .
        .
        .


RECORD IS EMP-REC WITHIN EMPLOYEE.

01  EMP-NO       PICTURE IS "X(5)".
01  DEPT-NO      PICTURE IS "X(5)".
        .
        .
        .


DATA CONTROL.

AREA NAME IS DEPARTMENT
    KEY IS DEPT-NO OF DEPT-REC.

AREA NAME IS EMPLOYEE
    KEY IS EMP-NO
    KEY IS ALTERNATE DEPT-NO OF EMP-REC DUPLICATES
        ARE INDEXED.

CONSTRAINT NAME IS DEPARTMENT-EMPLOYEE
    DEPT-NO OF EMP-REC DEPENDS ON DEPT-NO OF DEPT-REC.
```

Figure 2-39. Constraint Entry Within a Schema

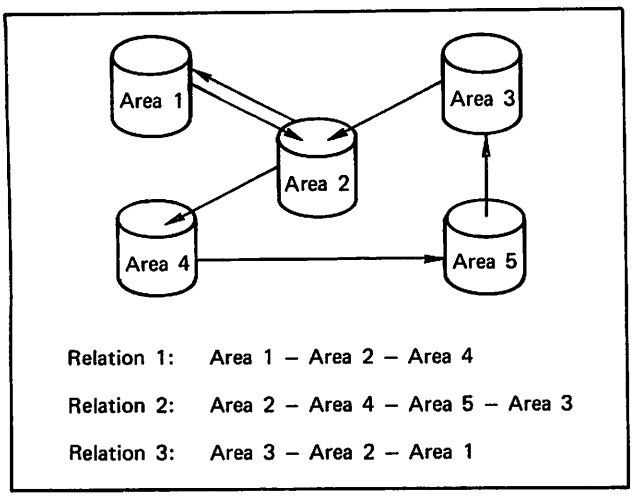


Figure 2-40. Relation Direction Example

```
RELATION NAME IS relation-name
    JOIN WHERE identifier-1 EQ identifier-2
        [identifier-3 EQ identifier-4] ... .
```

Figure 2-41. Relation Entry Format

The relation entry follows all other entries in the schema source program. A relation entry must be included for each relationship defined for the data base. The maximum number of relations allowed is 4095.

Each relation entry begins with a relation name statement. This is followed by a series of statements pairing the data item join term of one area with a corresponding data item in the area joined to it.

RELATION NAME Clause

The RELATION NAME clause assigns a name to the relation. It must be the first clause specified in the relation entry. The format of the RELATION NAME clause is shown in the relation entry.

The relation name must be unique among all relation names and area names in the schema. It can be up to 30 characters in length. The name can contain alphabetic and numeric characters and hyphens as long as the first character is alphabetic; consecutive hyphens are not allowed. The name specified in this clause is associated with all the areas joined in this relationship. An area can be included in more than one relation. An area that is joined in a relation cannot contain more than one DDL record type.

JOIN Clause

The JOIN clause specifies the data items that CDCS must inspect to join the areas in which the data items reside. The order in which the data items are specified determines the direction of the relationship, called a parent/child relationship. The JOIN clause is required if a RELATION NAME clause is present. The format of the JOIN clause is shown in the relation entry.

Within the JOIN clause a specified data item in one area is equated with an identical data item in another area; the two areas are thus related through specification of a common data item. The relational operator EQ must appear between each pair of identifiers included in the JOIN clause; the source identifier appears to the left of the EQ and the target identifier appears to the right.

In the relationship defined by the JOIN clause, the source identifier defines the parent; the target identifier defines the child. This relationship is valid for every source-target pair in the relation. Refer to section 6 for more information about relations.

The term identifier reflects a unique reference to a data name; it implies that the data name is referenced uniquely through a combination of subscripts and qualifiers. Refer to the Identifier subsection, which appears earlier in this section, for the identifier format.

Each source identifier (except the first) must be in the same area as the previous target identifier, so that a continuous path from one area to the next area is defined. Cycling is not allowed; an area cannot be joined to itself directly or indirectly.

The identifiers to the left and to the right of a relational operator must have identical characteristics in the picture specification or in the TYPE specification and must have identical synchronization and justification (if applicable). The position of the data items within the records need not be identical. The identifier must not specify a data item that is more than 255 characters long. Subscripting and qualification can be specified for any identifier.

Joining of areas in a relation is not guaranteed if a specified identifier is a group item with padding in it. Padding is used for word alignment of constituent items in a group item, and the content is unknown. Refer to figure 2-42 for an example. Padding must occur in group item H because item K must be on a word boundary. Since the padding content in H is unknown, data item G (1) in all probability will never equal data item H (1). An unusable relationship results.

The data item referenced by identifier can be any of the following types of data items:

A nonrepeating elementary item (no subscript).

A specific occurrence of a repeating elementary item or an occurrence of a repeating group item (up to three subscripts).

All occurrences of a repeating data item if it is an alternate key or the major part of an alternate key, and is specified to the right of the EQ in the JOIN clause (subscript ANY).

A concatenated key. (See KEY clause description.)

```
AREA NAME IS AREA1.
RECORD NAME IS R1 WITHIN AREA1.

02  G  OCCURS 2 TIMES.
    03 ITEM A  PIC  "X(5)".
    03 ITEM B  PIC  "9(10)".
    03 ITEM C  PIC  "9(3)V9(2)".

AREA NAME IS AREA2.
RECORD NAME IS R2 WITHIN AREA2.

02  H  OCCURS 1 TIMES.
    03 ITEM J  PIC  "X(2)".
    03 ITEM K TYPE FIXED 9,2.

RELATION NAME IS REL-1.
JOIN WHERE G (1) EQ H (1).
```

Figure 2-42. Alignment Example

If a repeating elementary item or group item is specified without a subscript, the first occurrence is used to join the relationship.

For efficient processing, the target identifier should be an item defined as a key field; retrieval of record occurrences from the file containing the target identifier is by that key. This rule holds for repeating or nonrepeating elementary items. Target identifiers defined as primary keys are most efficient; alternate keys are next most efficient. Only the first occurrence of a repeating item can be defined as a primary key. If a repeating group item or a repeating elementary item is defined as an alternate key, all occurrences of the item or group item are used as the key, rather than just the first occurrence. The ANY option used as a subscript identifies this type of alternate key field as a join term. If any one of the item occurrences satisfies the equality, the record occurrence is included in the relation occurrence. If an integer subscript is used instead of ANY on an alternate key, retrieval of record occurrences is by sequential access. If the target identifier is not a key, retrieval of record occurrences is by sequential access.

Figure 2-43 illustrates a record type in a schema. Based on the rules specified previously, the following data names from this record type could be used in a JOIN clause:

PART-NBR

STATE-ADDR (1,1) or PSOURCE (2)

SOURCE-NAME (ANY)

```
RECORD IS PARTS-LIST WITHIN INVENTORY.

01  PART-NAME PIC "X(20)".

01  PART-NBR PIC "X(25)".

01  PSOURCE OCCURS 10 TIMES.

    03  SOURCE-NAME PIC "X(40)".

    03  SOURCE-ADDR OCCURS 1 TIMES.

        05  STREET-ADDR PIC "X(20)".

        05  CITY-ADDR PIC "X(15)".

        05  STATE-ADDR PIC "X(5)".
```

Figure 2-43. Data Name Example

The identifier SOURCE-NAME must be to the right of the EQ operator in the JOIN clause and must be defined as an alternate key.

SCHEMA COMPILATION AND MAINTENANCE FACILITIES

The DDL compiler is a multifunctional compiler that generates the schema directory and uses that directory to verify compatibility of a recompiled schema with existing subschemas and to decode portions of the directory through the Exhibit facility. The particular operation performed by the compiler is selected by a parameter in the DDL3 control statement. Only one schema directory is allowed to reside on a file.

For field length requirements for schema compilation, refer to appendix G.

DDL3 CONTROL STATEMENT

The DDL3 control statement calls the DDL compiler. It specifies the function to be performed, the file that identifies the schema directory, the file that identifies the subschema library, the file that contains the source input for the DDL compiler, the file that is to receive output listings and diagnostics, and the memory allocation for the schema record buffer.

The format of the DDL3 control statement for schema compilation and maintenance facilities is shown in figure 2-44. The comma immediately following DDL3 can be replaced by a left parenthesis; the terminating period can be replaced by a right parenthesis.

All parameters of the DDL3 control statement are optional.

```
DDL3 [ [,DS] ] [,SC=lfn][,SB=lfn][,I=lfn][,L=lfn][,NI=nbr].
     [ [,EX] ]
```

Figure 2-44. DDL Control Statement Format for Schema Compilation and Maintenance

The EX and DS parameters select the function that is to be performed: The DS parameter selects schema compilation; the EX parameter selects execution of the Exhibit facility. These parameters are interpreted as follows:

omitted

The DS parameter is assumed; a DDL schema is compiled.

DS

A DDL schema is compiled.

EX

The Exhibit facility is executed. This parameter must be specified to execute the Exhibit facility.

The SC parameter identifies the local file that contains the schema directory. The parameter is interpreted as follows:

omitted

The first seven characters of the schema-name in the schema description entry of the source program identify the local file that contains the schema directory (the object schema).

SC=lfn

The specified local file name identifies the file that contains the schema directory (the object schema).

The SB parameter identifies the local file name of the subschema library file to be searched for checksum mismatches when a schema is recompiled. This parameter is interpreted as follows:

omitted

No subschema library search is performed.

SB=lfn

The local file name specified identifies the subschema library file to be searched.

The I parameter identifies the local file that contains source input for the DDL compiler. This parameter is interpreted as follows:

omitted

The local file INPUT is assumed to contain the source input for the DDL compiler.

I=lfn

The specified local file name identifies the file that contains source input for the DDL compiler.

The L parameter identifies the local file that receives listings and diagnostics generated by the DDL compiler. This parameter is interpreted as follows:

omitted

The local file OUTPUT receives the listings and diagnostics generated by the DDL compiler.

L=0

The local file OUTPUT receives only the diagnostics generated by the DDL compiler. This designation is not valid for execution of the Exhibit utility.

L=lfn

The specified local file name identifies the file that receives the listings and diagnostics generated by the DDL compiler.

The NI parameter specifies the number of 10-word blocks allocated for the schema record buffer. This parameter is interpreted as follows:

omitted

One-fourth of available memory is allocated for the schema record buffer.

NI=nbr

The number nbr specifies the number of 10-word blocks of memory that are allocated for the schema record buffer.

The default value for the NI parameter is one-fourth of the available memory. Available memory is defined as the memory available from the highest address for the DDL compiler (approximately 44000 octal) to the memory limit specified by an RFL control statement (NOS or NOS/BE operating system) or an EFL command (INTERCOM). For example, if the memory limit is specified as 54000 octal and available memory is 10000 octal words; the schema record buffer is 2000 octal words.

The schema record buffer must be large enough to contain all data description entries for the largest record type in the schema. One 10-word block can store a data description entry with a 10-character data name, a PICTURE clause, and a clause specifying a data base procedure. If an entry contains other clauses or a longer data name, it requires more than 10 words to store the entry.

SCHEMA COMPILATION

The schema must be coded according to the specifications in this manual. Schema source code can be entered through a terminal, processed by a text editor, and stored in a file.

The DDL compiler can be executed from a terminal and through a batch job in the following ways. When executed from a terminal, the schema program must reside on a file whose name is indicated by the I parameter of the DDL3 control statement. In a batch job in which the input file for the DDL compiler is assumed to be local file INPUT, the job stream must be structured so that the control statements precede the subschema source program. An end-of-record indicator must immediately precede the first line of schema source code to separate the schema program from the control statements.

Control statements in the job stream and the resulting source program are used by the DDL compiler to compile the schema directory and to store it as a permanent file. Only one schema directory should be stored on a file. The control statements provide information for the operating system and for the DDL compiler.

DDL3 Control Statement for Compilation

The DDL3 control statement must be included in the set of control statements preceding the DDL source program. It provides the DDL compiler with information related to the schema. The DDL3 control statement is used to request compilation and to provide the local file name of the schema directory. The DDL3 control statement for schema compilation is as follows:

```
DDL3,DS,SC=lfn.
```

The DS parameter and SC parameter are optional and have default values. The I, L, and NI parameters must be specified if default values are not applicable for input and output files and for memory allocation for the schema record buffer (refer to the DDL3 Control Statement subsection).

FILE Control Statement

The FILE control statement provides CYBER Record Manager with information that is required for file processing. A FILE control statement for each area in the schema must precede the DDL3 control statement. The information in the FILE control statements along with file information from the data control entry is placed in the schema directory.

The FILE control statement equates the area name with an operating system file name and associates the area with a particular file organization. Figure 2-45 illustrates the FILE control statement and indicates parameters required for schema compilation. Optionally, the comma immediately following FILE can be replaced by a left parenthesis; the terminating period can be replaced by a right parenthesis.

```
FILE,lfn,{FO=AK}
         {FO=DA}[,p] ... .
         {FO=IS}
```

Figure 2-45. FILE Control Statement Format

The local file name specified in the FILE control statement must be the name of an area defined in the schema or, if the area name exceeds seven characters, the first seven characters of the area name.

The FO parameter associates the area with the specified file organization. This parameter must be specified in one of the following forms:

FO=AK

The extended actual key file organization is associated with the area.

FO=DA

The extended direct access file organization is associated with the area.

FO=IS

The extended indexed sequential file organization is associated with the area.

The p indicates additional parameters. Depending on particular conditions, three additional parameters are required.

The XN parameter can be specified if alternate keys are specified for an area. The XN parameter associates the area with the file that contains the alternate key index. If not specified and alternate keys have been specified for the area, the DDL compiler automatically assigns a file name to reference an index file for the area and issues a trivial diagnostic. If specified, the XN parameter must be specified in the following form:

XN=lfn

The specified local file name identifies the alternate key index. The lfn must be one to seven characters in length, must contain only characters or digits, must begin with a letter, and must not duplicate another lfn indicated in the job stream.

The HMB parameter must be specified if the file organization associated with an area is extended direct access. The HMB parameter indicates the number of home blocks associated with the area. The HMB parameter must be specified in the following form:

HMB=nn

The number nn home blocks are associated with the area. The number nn must be an integer 1 through $2^{24}-1$. See the CYBER Record Manager Advanced Access Methods reference manual more information about home blocks.

The key length (KL) parameter must be specified if the file organization associated with an area is extended actual key. For the other file organizations, if KL is not specified then DDL determines the value of the KL parameter from the data descriptions. If a KL parameter specification does not agree with the data descriptions, DDL uses the value from the data descriptions and issues a trivial diagnostic. The KL parameter must be specified in the following form:

KL=nnn

For an extended actual key file, the number nnn must be an integer within the range of 1 through 8. For all other file organizations, the number cannot exceed 240 characters.

Additional parameters can provide other information (such as CYBER Record Manager record type and block type). Table 2-3 lists the FILE control statement parameters that can be specified. Refer to the CYBER Record Manager Advanced Access Methods reference manual for detailed descriptions of the parameters.

TABLE 2-3. FILE CONTROL STATEMENT PARAMETERS

Parameter	File Organization		
	Extended Indexed Sequential	Extended Direct Access	Extended Actual Key
BFS	X	X	X
CL	X	X	X
CP	X	X	X
DP	X		X
EFC	X	X	X
FL	X	X	X
FLM	X	X	
FO	X	X	X
HL	X	X	X
HMB		X	
IP	X		
KL	X	X	X
LL	X	X	X
LP	X	X	X
MBL	X	X	X
MNR	X	X	X
MRL	X	X	X
NL	X		
ORG	X	X	X
RB	X	X	X
RMK	X	X	X
RT	X	X	X
TL	X	X	X
XN	X	X	X

NOTE

Refer to appendix F for recommendations on access methods.

The FILE control statement for an area that has record keys specified can include additional parameters related to the key. Parameters related to the data item that is the record key should agree with the data item description. If the record key and data item descriptions do not agree, the DDL compiler uses the record key description and issues a trivial diagnostic.

If a FILE control statement is not included for an area, a diagnostic message is issued. The following defaults apply to FILE control statements that do not specify the applicable parameter.

IP no index padding

DP no data block padding (release default)

NL one level of index blocks (release default)

RT=U CYBER Record Manager undefined record type

MRL=size of maximum record length in the area

MNR=size of minimum record length in the area

If the MRL parameter (maximum record length) is specified in the FILE control statement and the value is less than the length of the longest record in the area, the DDL compiler overrides the FILE control statement MRL parameter, substituting a value equal to the size of the longest record. The DDL compiler then issues a diagnostic. If the specified value for the MNR parameter (minimum record length) is inadequate, the DDL compiler overrides the FILE control statement MNR parameter, substituting a value large enough to contain all keys defined for the area.

If the FILE control statement specifies RT=T (CYBER Record Manager trailer count record type), and the area contains a record with a variable length repeating data item, the DDL compiler computes the values for the HL, TL, CP, and CL parameters. A warning diagnostic message is issued if the FILE control statement contains these parameters with values different from the values computed by the DDL compiler.

Schema Compilation Example

The source program for a schema must be preceded by a set of control statements and followed by an end-of-information indicator. Figure 2-46 illustrates a job that compiles a schema and stores it as a permanent file. The FILE control statements must specify one of the allowed organizations: extended indexed sequential, extended direct access, or extended actual key. The DEFINE and REQUEST/CATALOG control statements specify the local file name of the file that contains the schema directory and assign it to a permanent file device. The DDL3 control statement specifies schema compilation and the local file name MANUFAC to identify the schema directory.

Schema Compilation Output

A listing of the DDL source program is provided whenever a schema is compiled and L=0 is not specified in the DDL3 control statement. Each line of the listing corresponds to one statement in the source program. The format and order of each line on the listing are identical to the format and order of the statements in the source program. Figure 2-47 is a sample source listing for a schema compilation.

NOS Operating System

```
Job statement
USER control statement
CHARGE control statement
DEFINE,MANUFAC=MANUFAC/CT=PU,M=R.
FILE(EMPLOYE,FO=IS,XN=IXEMP)
FILE(JOBDETA,FO=IS)
FILE(DEPARTM,FO=IS,XN=IXDEPT,RT=T)
FILE(PROJECT,FO=DA,XN=IXPROJ,HMB=7)
FILE(DEVELOP,FO=DA,XN=IXDEV,HMB=11)
FILE(TESTS,FO=IS,XN=IXTEST)
FILE(CDCSCAT,FO=IS)
DDL3,DS,SC=MANUFAC.
End-of-record
    Schema source input
End-of-information
```

NOS/BE Operating System

```
Job statement
ACCOUNT control statement
REQUEST,MANUFAC,PF.
FILE(EMPLOYE,FO=IS,XN=IXEMP)
FILE(JOBDETA,FO=IS)
FILE(DEPARTM,FO=IS,XN=IXDEPT,RT=T)
FILE(PROJECT,FO=DA,XN=IXPROJ,HMB=7)
FILE(DEVELOP,FO=DA,XN=IXDEV,HMB=11)
FILE(TESTS,FO=IS,XN=IXTEST)
FILE(CDCSCAT,FO=IS)
DDL3,DS,SC=MANUFAC.
CATALOG,MANUFAC,MANUFACTURINGDB,ID=owner.
End-of-record
    Schema source input
End-of-information
```

Figure 2-46. Schema Compilation Example

```
MANUFACTUR                  * SOURCE LISTING *   (82061)  DDL  3.2+564.


00001          SCHEMA NAME IS MANUFACTURING-DB.
00002
00003          AREA IS EMPLOYEE
00004             ACCESS-CONTROL LOCK FOR RETRIEVAL IS "EMP-READ"
00005             ACCESS-CONTROL LOCK FOR UPDATE IS "EMP-WRITE"
00006                     CALL OPENEMP BEFORE OPEN.
00007
00008          RECORD IS EMPREC WITHIN EMPLOYEE.
00009             01  EMP-ID                  PICTURE "X(8)".
00010             01  SALARY                  TYPE FIXED DECIMAL 8,2
00011                                             CALL EMPCHK BEFORE.
00012             01  EMP-LAST-NAME           PICTURE "A(20)".
00013             01  EMP-INITIALS            PICTURE "A(4)".
00014             01  DEPT                    PICTURE "X(4)".
00015             01  ADDRESS-NUMBERS         PICTURE "X(6)".
00016             01  ADDRESS-STREET          PICTURE "X(20)".
00017             01  ADDRESS-CITY            PICTURE "A(15)".
00018             01  ADDRESS-STATE-PROV      PICTURE "A(15)".
00019             01  POSTAL-CODE             PICTURE "X(10)".
00020             01  ADDRESS-COUNTRY         PICTURE "A(15)".
00021             01  PHONE-NO                PICTURE "9(10)".
00022             01  HIRE-DATE               PICTURE "9(6)".
00023             01  INSURANCE-NO            PICTURE "9(10)".
00024             01  NUM-DEPENDENTS          TYPE DECIMAL FIXED 2.
00025             01  JOB-CLASS               PICTURE "A"
00026                                             FOR ENCODING CALL DBPJC1
00027                                             FOR DECODING CALL DBPJC2.
00028             01  GRADE-LEVEL             PICTURE "9"
00029                                             CHECK VALUE 0 THRU 8.
00030
00031          /*    THE FOLLOWING ARE YEAR-TO-DATE TOTALS
00032          */
00033             01  GROSS                   TYPE DECIMAL FIXED 7,2.
00034             01  FED-TAX                 TYPE DECIMAL FIXED 7,2.
00035             01  STATE-TAX               TYPE DECIMAL FIXED 6,2.
00036             01  DISABILITY              TYPE DECIMAL FIXED 5,2.
00037             01  SS-INSURANCE            TYPE DECIMAL FIXED 4,4.
00038
00039
00040          AREA IS JOBDETAIL
00041             ACCESS-CONTROL LOCK IS "ABCDEFZ".
00042          RECORD IS JOBREC WITHIN JOBDETAIL
00043                     CALL DELCHK BEFORE DELETE.
                  .
                  .
                  .
```

Figure 2-47. Sample Schema Compilation Output Listing (Sheet 1 of 3)

```
00125          AREA NAME IS CDCSCAT
00126          ACCESS-CONTROL LOCK IS "PERMISSION*GRANTED".
00127          RECORD IS QUCATREC WITHIN CDCSCAT.
00128                 QUCAT-KEY              PICTURE "X(10)".
00129                 QUCAT-ITEM             PICTURE "X(1030)".
00130
00131          DATA CONTROL.
00132
00133          AREA NAME IS EMPLOYEE
00134                 KEY IS EMP-ID OF EMPREC
00135                     DUPLICATES ARE NOT ALLOWED
00136                 KEY IS ALTERNATE DEPT
00137                     DUPLICATES ARE INDEXED.
00138
00139          AREA NAME IS JOBDETAIL
00140                 KEY ID IS CONCATKEY < EMP-ID OF JOBREC
00141                                      SEQ-NO >
00142                     DUPLICATES ARE NOT ALLOWED.
00143
00144          AREA NAME IS DEPARTMENTS
00145                 KEY IS DEPT-NO
00146                     DUPLICATES ARE NOT ALLOWED
00147                 KEY IS ALTERNATE MGR-ID
00148                     DUPLICATES ARE FIRST.
00149
00150          AREA NAME IS PROJECT
00151                 KEY IS PROJECT-ID OF PROJREC
00152                     DUPLICATES ARE NOT ALLOWED
00153                 KEY IS ALTERNATE RESPONSIBILITY
00154                     DUPLICATES ARE ALLOWED.
00155
00156          AREA NAME IS DEVELOPMENT-PRODUCTS
00157                 KEY IS PRODUCT-ID OF DEVREC
00158                     DUPLICATES ARE NOT ALLOWED
00159                 KEY IS ALTERNATE PROJECT-ID OF DEVREC
00160                     DUPLICATES ARE ALLOWED
00161                 KEY IS ALTERNATE EVAL-ID
00162                     DUPLICATES ARE INDEXED.
00163
00164          AREA NAME IS TESTS
00165                 KEY IS TESTNO
00166                     DUPLICATES ARE NOT ALLOWED
00167                 KEY IS ALTERNATE TNAME
00168                     DUPLICATES ARE INDEXED
00169                 KEY IS ALTERNATE TESTER
00170                     DUPLICATES ARE FIRST
00171                 SEQUENCE IS ASCII.
00172
00173          AREA NAME IS CDCSCAT
00174                 KEY IS QUCAT-KEY OF QUCATREC
00175                     DUPLICATES ARE NOT ALLOWED
00176                 SEQUENCE IS DISPLAY.
00177
00178
00179          CONSTRAINT NAME IS MGR-CONST
00180                     MGR-ID OF DEPTREC DEPENDS ON EMP-ID OF EMPREC.
00181
00182          CONSTRAINT NAME IS PROJ-CONST
00183                     PROJECT-ID OF DEVREC DEPENDS ON PROJECT-ID OF PROJREC.
00184
00185
00186          RELATION NAME IS EMP-REL
00187                 JOIN WHERE EMP-ID OF JOBREC EQ EMP-ID OF EMPREC.
00188
00189          RELATION NAME IS TEST-REL
00190                 JOIN WHERE PRDCTNO OF TESTREC EQ PRODUCT-ID OF DEVREC.
00191
00192          RELATION NAME IS DPD-REL
00193                 JOIN WHERE MGR-ID OF DEPTREC EQ RESPONSIBILITY OF PROJREC
00194                            PROJECT-ID OF PROJREC EQ PROJECT-ID OF DEVREC.
```

Figure 2-47. Sample Schema Compilation Output Listing (Sheet 2 of 3)

```
*** AREA CHECKSUMS ***
        AREA NAME                              CHECKSUM
        EMPLOYEE                               56430552040035230620
        JOBDETAIL                              13626262424331006604
        DEPARTMENTS                            16574462455232633415
        PROJECT                                35175246562100113225
        DEVELOPMENT-PRODUCTS                   61470111011661762112
        TESTS                                  40015236474170366115
        CDCSCAT                                57033604647410770643
*** RELATION CHECKSUMS ***
        RELATION NAME                          CHECKSUM
        EMP-REL                                33541156402300070013
        TEST-REL                               10246503456523070027
        DPD-REL                                16223300077042601641
*** DATA-BASE PROCEDURE LIST FOR SCHEMA  MANUFACTURING-DB
        CALCCP
        CALCHR
        DATESP
        DBPJC1
        DBPJC2
        DELCHK
        EMPCHK
        OPENEMP
DDL COMPLETE.          0 DIAGNOSTICS.
  47300B CM USED.    0.419  CP SECS.
```

Figure 2-47. Sample Schema Compilation Output Listing (Sheet 3 of 3)

The DDL compiler assigns a line number to each input statement, beginning with 00001. The line numbers are printed on the source listing, starting in column 16. Diagnostic messages begin in column 3 of the listing. After the last input statement is listed, a checksum for each area and relation, together with the area or relation name, is printed. Following the checksums, an alphabetic list of all data base procedures specified in the schema is written to the output file. Lastly, a compilation summary is printed.

The source listing can be suppressed by specifying L=0 in the DDL3 control statement. In this situation, diagnostic messages and the compilation summary are the only listing that is produced.

RECOMPILATION GUIDELINES

The checksums generated by the DDL compiler for each area and relation in the schema determine when a subschema must be recompiled. If a checksum in a recompiled schema is different from the corresponding checksum in the previous schema, any subschema referencing that changed area or relation must be recompiled; a list of these subschemas is printed at the end of a schema compilation when the DDL3 control statement includes the SB parameter. The DDL3 control statement for compiling a schema and generating a list of subschemas that require recompilation is as follows:

```
DDL3,DS,SC=lfn,SB=lfn.
```

The DS parameter and SC parameters are optional and have default values. The SB parameter is required; it specifies the local file name of the file that contains the subschema library to be searched for checksum mismatches. The I, L, and NI parameters must be specified if default values are not applicable for the input and output files and for memory allocation for the schema record buffer (refer to the DDL3 Control Statement subsection).

In a job that recompiles a schema and checks for subschema checksum mismatches, the subschema library must be attached. The DDL compiler searches all the subschemas in the specified library and compares checksums. If checksum mismatches are found, the names of any subschemas that must be recompiled are printed. An example of this facility is shown in figure 2-48.

```
----- REPORT ON SUB-SCHEMA RECOMPILATIONS FOR SCHEMA  MANUFACTURING-DB
          FOLLOWING SUB-SCHEMAS REQUIRE RECOMPILATION

                MARKETING
                INVOICING
                PAYROLL
                    -----END REPORT-----
```

Figure 2-48. Recompilation List Example

EXHIBIT FACILITY

The Exhibit facility of DDL provides the capability of decoding a specific entry or group of entries contained in the schema directory. The output is similar in appearance to the initial DDL input. Certain output entries generated by the Exhibit facility are not associated with any specific source statement. They appear at the beginning of the output as comments and are enclosed by a preceding delimiter (/*) and a following delimiter (/*). Output is in the format of 50 characters per line.

Control Statement Format

The DDL3 control statement must be specified to execute the Exhibit facility. After the schema directory file is attached, Exhibit can be called. The control statement for executing the Exhibit facility is as follows:

```
DDL3,EX,SC=lfn.
```

The EX parameter is required; it specifies execution of the Exhibit facility. The SC parameter is required; it identifies the local file name of the schema directory. The I and L parameters must be specified if default values are not applicable for input and output files (refer to the DDL3 Control Statement subsection).

EXHIBIT Directive Format

Input specification directives for EXHIBIT must follow a specific directive format as shown in figure 2-49. Input for each directive is free-format and characters can appear in any columns from column 1 through column 72. Each directive must be contained within the 72 columns.

One of the keywords AREA, RECORD, ITEM, ALL-ENTRIES, CONSTRAINT, or RELATION must be specified to indicate the type of entry to be decoded and written to the output file. ALL-ENTRIES specifies that the names or clauses for all entries in the schema are to be exhibited. The option ONLY applies to record

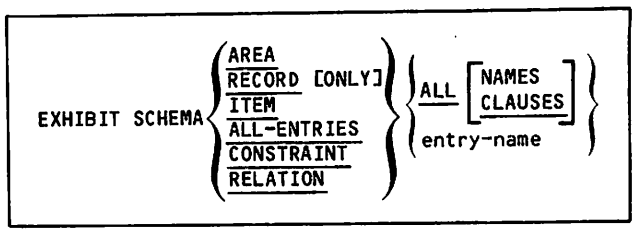


Figure 2-49. EXHIBIT Directive Format

entries and specifies that only the record name or clauses associated with the record entry are to be exhibited.

If ONLY is not specified, all data subentry names or clauses associated with the record entry are also decoded and written to the output file.

When ALL or ALL NAMES is specified, all entry names for the specified entry type are exhibited. Specification of ALL CLAUSES causes all entry names and clauses for the specified entry type to be displayed.

If an entry name is designated in the directive, it indicates the name of the area, record, item, constraint, or relation for which all the associated clauses are to be decoded and written to the output file. If entry name specifies the data name of an item, it can be qualified as specified in paragraphs describing Data Reference and Qualification in this section. If the data name of an item is not qualified in the specification and multiple entries for that data name exist, the first entry having the designated data name is exhibited.

To exhibit all schema clauses, the ALL-ENTRIES ALL CLAUSES specification is used. Control information maintained by the DDL compiler is also included in the output when this option is selected. Refer to figure 2-50 for several examples of the EXHIBIT directive.

Sample control statements and directive formats for the exhibit facility are shown with their respective generated output samples in figures 2-51 through 2-54.

Example 1

```
EXHIBIT SCHEMA RECORD ONLY ALL NAMES
```

This directive exhibits all record names in the schema.

Example 2

```
EXHIBIT SCHEMA AREA ALL CLAUSES
```

This directive exhibits all areas and their respective clauses.

Example 3

```
EXHIBIT SCHEMA RECORD EMPREC
```

This directive exhibits a record called EMPREC with all its data items and associated clauses.

Example 4

```
EXHIBIT SCHEMA ALL-ENTRIES ALL CLAUSES
```

This directive exhibits the whole schema.

Figure 2-50. EXHIBIT Examples

NOS Operating System

```
Job statement
USER control statement
CHARGE control statement
ATTACH,MANUFAC/UN=usernumber.
DDL3,EX,SC=MANUFAC.
End-of-record
    EXHIBIT SCHEMA RECORD ONLY ALL NAMES
End-of-information
```

NOS/BE Operating System

```
Job statement
ACCOUNT control statement
ATTACH,MANUFAC,MANUFACTURINGDB,ID=owner.
DDL3,EX,SC=MANUFAC.
End-of-record
    EXHIBIT SCHEMA RECORD ONLY ALL NAMES
End-of-information
```

Figure 2-51. Executing the EXHIBIT Utility, Example 1

```
/*
 EXHIBIT SCHEMA MANUFACTURING-DB
 TIME,DATE OF CREATION 14.14  82087
**********************************
*/
              00001                EXHIBIT SCHEMA RECORD ONLY ALL NAMES
RECORD EMPREC
RECORD JOBREC
RECORD DEPTREC
RECORD PROJREC
RECORD DEVREC
RECORD TESTREC
RECORD QUCATREC
```

Figure 2-52. Sample EXHIBIT Utility Output, Example 1

NOS Operating System

```
Job statement
USER control statement
CHARGE control statement
ATTACH,MANUFAC/UN=usernumber.
DDL3,EX,SC=MANUFAC.
End-of-record
    EXHIBIT SCHEMA AREA ALL CLAUSES
End-of-information
```

NOS/BE Operating System

```
Job statement
ACCOUNT control statement
ATTACH,MANUFAC,MANUFACTURINGDB,ID=owner.
DDL3,EX,SC=MANUFAC.
End-of-record
    EXHIBIT SCHEMA AREA ALL CLAUSES
End-of-information
```

Figure 2-53. Executing the EXHIBIT Utility, Example 2

```
/*
 EXHIBIT SCHEMA MANUFACTURING-DB
 TIME,DATE OF CREATION 14.14  82087
**********************************
*/
              00001                EXHIBIT SCHEMA AREA ALL CLAUSES
AREA EMPLOYEE
    CALL OPENEMP BEFORE OPEN RETRIEVAL UPDATE
    ACCESS-CONTROL FOR RETRIEVAL IS "EMP-READ"
    ACCESS-CONTROL FOR UPDATE IS "EMP-WRITE".
        .
        .
        .
AREA CDCSCAT
    ACCESS-CONTROL FOR RETRIEVAL UPDATE IS
        "PERMISSION*GRANTED".
```

Figure 2-54. Sample EXHIBIT Utility Output, Example 2

A COBOL subschema describes the portion of the data base that is needed by one or more COBOL application programs written in the COBOL 5 language. Similarly, a Query Update subschema in CYBER Database Control System (CDCS) data base access mode (hereafter referred to simply as a Query Update subschema) describes the portion of the data base that is needed by one or more Query Update applications. In the subschema, record structures and individual data items are defined in a manner acceptable to the COBOL compiler or to Query Update. The subschema data descriptions replace the COBOL or Query Update data descriptions that the COBOL application programmer or Query Update user would otherwise supply.

A subschema is written based on the schema description. Generally, the schema description of data can be changed in the subschema to meet the needs of the COBOL program or Query Update application; however, some limitations have been imposed on the variations between the schema and the subschema.

SUBSCHEMA STRUCTURING CONVENTIONS

COBOL and Query Update subschemas consist of three required divisions and two optional divisions:

Title Division

Names the subschema and identifies the schema.

Alias Division (optional)

Assigns alternate names to be used in place of names assigned in the schema.

Realm Division

Identifies the schema areas to be made available through the subschema.

Record Division

Describes the structure and content of each record type in the subschema.

Relation Division (optional)

Identifies the specific realm relationships that are to be used in the subschema and in the COBOL programs or Query Update applications referencing the subschema; also, specifies the qualification criteria that must be satisfied by records that are to be returned to the application programs from the data base.

All data from the data base that is to be accessible through COBOL or Query Update is described and organized in the Record Division. The structure of this division is similar to that of a record description entry in the Data Division of a COBOL program.

DATA DESCRIPTION

The basic unit of data description is the data description entry. The record description entry is the entire description of a data base record type. The record description entry consists of a record name entry followed by a series of data description entries that describe the specific data items within the record. Each data description entry has a level number, a data name, and a terminating period. In addition, the entry can have one or more clauses describing the data item. Data items are described in terms of size, class, and usage.

The level number designates the level of the entry relative to other entries in the record description. The highest level entry, the record name entry, is always level number 01. Other entries, which can be either group or elementary data items, are assigned level numbers 02 through 49 based on the position of the specific data item within the hierarchical structure of the record. Level numbers need not be consecutive but must be ordered so that the higher the number, the lower the entry in the hierarchy. Level numbers 66 and 88 are special level numbers that are used to specify two particular types of data description entries; they do not define the hierarchy of the item. A level 66 entry renames a group or elementary item. A level 88 entry provides a value or range of values to be associated with a condition name; level 88 entries cannot be used in a Query Update subschema.

With the exception of a data name being redefined or renamed, the data name in the entry must be one of the following:

A data name from the schema description

An alias assigned in the Alias Division

A user-defined name assigned to a nonrepeating group item

A user-defined name assigned to an entry in a repeating group that corresponds to a vector in the schema

Data names from the schema must appear exactly as they are specified in the schema description. If an alias is assigned to the schema data item, the alias must be used to reference that data item in the data description entry. Nonrepeating group items are defined under Schema/Subschema Compatibility.

At least one clause must be included in the data description entry for an elementary item. A group item entry can also have one or more clauses. The order of clauses is not important except where explicitly stated in the clause description.

The only punctuation required in a data description entry is the terminating period. Semicolons or commas can be used to separate clauses.

DATA ORGANIZATION

Data items in the Record Division are organized first into records and then into group and elementary items. Each record type corresponds to a record type in the schema and must be within the realms (schema areas) specified in the Realm Division. A record description entry begins with level number 01 and includes a series of subordinate data description entries with higher level numbers.

Group Items

A group item is a collection of related items organized in a hierarchical structure. The group name data description entry has the lowest level number within the group itself. Other items within the group item have successively higher level numbers ending with the most elementary item, the highest numbered item in the group. All data items in a group item can be referenced collectively by the data name of the group item. A description must be written for each data item in the group, and the group name data description entry must have at least a level number and a data name. A group item can be part of a larger group (nested group items).

Elementary Items

An elementary item is an item that cannot be further subdivided. If it is part of a group item, the elementary item has the highest level number of the group item (03-49) to which it belongs. An elementary item in the subschema must correspond to an elementary item in the schema. A schema elementary item cannot be described in the subschema as a group item even if the total number of characters represented in the schema and in the subschema are equal.

SCHEMA/SUBSCHEMA COMPATIBILITY

The subschema is created to accommodate the needs of a COBOL or Query Update application program. Some characteristics of the data in the data base are fixed by the schema and cannot be changed by the subschema; other characteristics specified in the schema can be different in the subschema. The limitations placed on the variations between the schema and the subschema are precise and must be carefully considered when the subschema is described.

OMISSION OF DATA ITEMS

The subschema normally describes only a portion of the data base. Data items that are not required by a COBOL or Query Update program are not included in the subschema description. Elementary items, group items, complete records, and entire areas in the schema can be omitted from the subschema. When a group item, a record, or an area is not included in the subschema description, all subordinate entries are automatically omitted and cannot be referenced in the subschema.

Each schema area that is to be included in the subschema is identified in the Realm Division. If a schema area is omitted from the subschema, no records or data items within that area can be included in the subschema. Omission of an area implies omission of all subordinate items.

Once a schema area is specified as a realm in the subschema, any or all of the records in that area can be included in the subschema. If a record is omitted, all data items within that record are also omitted from the subschema. Only those records within the realms (schema areas) identified in the Realm Division can be described in the subschema.

When a record is included in the subschema, data items within the schema record can be omitted from the subschema record description. Elementary data items that are not subordinate to a group data item can be included or omitted as needed in the subschema. If a group data item is omitted, all subordinate data items must also be omitted. For nested group items, the highest level group item must be included when a lower level group item is to be included in the record description. Primary key items cannot be omitted from the subschema record description.

Figure 3-1 illustrates a subschema record that omits several of the data items in the corresponding schema record. In this example, only two schema areas have been included in the subschema. The schema records in the CUSTOMER and SALES areas are the only records that can be specified in the subschema; the records in the PRODUCTS area are automatically omitted. The subschema record CUSTOMER-REC includes the data items CUSTOMER-ID, MONTHLY-ORDERS, and TOTAL-AMT; all the other data items have been omitted from the subschema and cannot be accessed by an application program. If the group item MONTHLY-ORDERS had been omitted, the subordinate item TOTAL-AMT could not have been included in the subschema.

ORDERING OF DATA ITEMS

Data items in a record selected for the subschema need not be organized in exactly the same sequence as specified in the schema. The relative hierarchy of the schema must be maintained when the order of data items is changed.

Only one restriction is placed on changing the order of data items in the subschema record description. Data items that are subordinate to a group item in the schema must be subordinate to the same group item in the subschema.

Figure 3-2 illustrates the reordering of data items from the schema to the subschema.

DEFINITION OF DATA ITEMS

Each data base data item that is to be made available to a COBOL or Query Update application program is described in a subschema record description entry. Data items not included in the subschema cannot be accessed by a program. The size and class of the data item can be exactly the same as described in the schema or, under certain circumstances, they can be different. Nonrepeating group items, which do not exist in the schema, can be specified in the subschema.

Schema

```
AREA NAME IS SALES.
AREA NAME IS CUSTOMER.
AREA NAME IS PRODUCTS.
.
.
.
RECORD NAME IS CUSTOMER-REC
    WITHIN CUSTOMER.

01  CUSTOMER-ID     ...
01  CUSTOMER-NAME   ...
01  ADDR            ...
01  CITY            ...
01  STATE           ...
01  ZIP-CODE        ...
01  PHONE-NUM       ...
01  MONTHLY-ORDERS OCCURS 12 TIMES.
    03  NUM-ORDERS...
    03  TOTAL-AMT ...
```

Subschema

```
REALM DIVISION.
RD  CUSTOMER, SALES.
RECORD DIVISION.
01  CUSTOMER-REC.
    03  CUSTOMER-ID     ...
    03  MONTHLY-ORDERS OCCURS 12
                          TIMES.
        05  TOTAL-AMT   ...
01  SALES-REC.
.
.
.
```

Figure 3-1. Omitting Schema Items From the Subschema

Schema

```
01  CUSTOMER-ID     ...
01  INVOICE-NUM     ...
01  YEAR            ...
01  MONTH           ...
01  DA              ...
01  CHARGE-NUM      ...
01  REMARKS         ...
01  AMOUNT-DUE      ...
01  AMOUNT-REC      ...
01  SALES-TAX       ...
01  NUM-ITEMS       ...
01  ITEM OCCURS     ...
    03  QUANTITY    ...
    03  DESC-A      ...
    03  UNIT-PRICE  ...
    03  EXT-PRICE   ...
```

Subschema

```
03  CUSTOMER-ID     ...
03  INVOICE-NUM     ...
03  AMOUNT-REC      ...
03  AMOUNT-DUE      ...
03  SALES-TAX       ...
03  MONTH           ...
03  DA              ...
03  YEAR            ...
03  REMARKS         ...
03  CHARGE-NUM      ...
03  NUM-ITEMS       ...
03  ITEM OCCURS     ...
    05  DESC-A      ...
    05  QUANTITY    ...
    05  UNIT-PRICE  ...
    05  EXT-PRICE   ...
```

Figure 3-2. Reordering Data Items

Data Size and Class

The size and class of data items in the subschema are specified by the PICTURE and USAGE clauses. The schema description designates the size and class through either the PICTURE or TYPE clause. When the PICTURE and USAGE clauses are written for the subschema, the schema description must be checked to determine that the subschema description does not conflict with the schema description.

The size of the data item is specified in the subschema by the number of character position designators (A, X, and 9) in the picture-specification of the PICTURE clause. Usually, when a data item is described in a picture-specification, the number of character positions specified in the subschema should not exceed the number of character positions specified for the item in the schema. For any subschema data item that is larger than the corresponding schema item, the DDL compiler issues an informative diagnostic message. The user must decide when the schema or subschema picture-specification should be corrected. Figure 3-3 has examples of size discrepancies that cause the DDL compiler to issue diagnostics. In example 3, the subschema picture size is five, but the internal storage size is one word, which is greater than the schema size of five. When a conversion error such as nonblank truncation occurs, the operation is aborted, CDCS writes a nonfatal error message to the dayfile, and processing continues.

	Schema	Subschema
Example 1		
	01 AAAA PIC "X" FOR ENCODING CALL DEPROC.	02 AAAA PIC X(5).
Example 2		
	01 BBBB PIC "999".	02 BBBB PIC 9(4).
Example 3		
	01 CCCC PIC "9(5)".	02 CCCC PIC 9(5) USAGE IS COMP-1.

Figure 3-3. Size Discrepancies of Data Items

Each data item falls into one of eight data class categories. The data class is determined by the description of the data item. The data classes and codes are as follows:

Code	Data Class
0	Display alphanumeric
1	Display alphabetic
3	Display integer
4	Display fixed-point
10	Coded binary integer
13	Coded floating-point normalized
14	Coded double-precision (Query Update subschema only)
15	Coded complex (Query Update subschema only)

The data class that can be specified in the subschema depends upon the class of the data item in the schema. Valid subschema data classes for each schema data class are indicated in table 3-1.

If the schema specifies a CHECK IS PICTURE clause, the data description in the subschema must match the data description in the schema. The CHECK IS PICTURE clause in the schema definition inhibits data conversion between the schema and the subschema. Identical schema and subschema data descriptions are also required for a primary key item embedded in a record of an area having actual key file organization.

The data class of an item described in the schema is determined by either a PICTURE clause or a TYPE clause. Table 3-2 lists each data class as it is represented in the schema. The PICTURE and USAGE clauses in the subschema description of a data item determine the subschema data class. The COBOL

TABLE 3-1. VALID SCHEMA TO SUBSCHEMA CLASS CONVERSIONS

Schema Data Class	Subschema Data Class							
	0	1	3	4	10	13	14†	15†
0	X	X	X					
1	X	X						
3	X		X	X	X	X	X	
4			X	X	X	X	X	
10			X	X	X	X	X	X
13			X	X	X	X	X	X
14			X	X	X	X	X	X
15					X	X	X	X

†Query Update subschema only

subschema representation of each valid data class is listed in table 3-3; table 3-4 lists the Query Update subschema representation of each valid data class. Refer to appendix G for a summary of data definition in DMS-170.

Repeating Data Items

Two types of repeating data items can be designated in the subschema: vectors and repeating groups. A vector is an elementary data item that is repeated a number of times in each record. A repeating group is a collection of data items that is repeated; the entire collection, not individual data items, is repeated a number of times in each record.

TABLE 3-2. DATA CLASS REPRESENTATION IN THE SCHEMA

Data Class	Schema PICTURE Clause	Schema TYPE Clause	Internal Representation
0	None	CHARACTER	Display code, alphanumeric
	Character-string (A X 9, not all 9s)	None	
1	Character-string (all As)	None	Display code, alphabetic
3	Numeric-picture integer (9 T)	None	Display code numeric can have sign overpunch in last character position
4	Numeric-picture fixed-point (9 V T P)	None	Display code numeric plus implicit or explicit decimal or scaling position
10	None	FIXED integer-1, integer-2 (where integer-1 $\leq$ 18)	Binary integer
13	None	FLOAT integer-1 (where integer-1 $\leq$ 14)	Signed, normalized floating-point (60-bit)
14	None	FLOAT integer-1 (where 15 $\geq$ integer-1 $\leq$ 29)	Two words of normalized floating-point
15	None	TYPE COMPLEX	Two words of a complex number

TABLE 3-3. DATA CLASS REPRESENTATION IN THE COBOL SUBSCHEMA

Data Class	COBOL Subschema PICTURE Clause	COBOL Subschema USAGE Clause	Internal Representation
0	Alphanumeric (A X 9) (A specification of mixed As and 9s is treated as all Xs)	DISPLAY (or none)	Display code, alphanumeric
1	Alphabetic (A)	DISPLAY (or none)	Display code, alphabetic
3	Numeric (9 S)	DISPLAY or COMP (or none)	Display code numeric, $\leq$ 18 characters; trailing sign overpunch if S is specified
4	Numeric (9 S V P)	DISPLAY or COMP	Display code numeric, $\leq$ 18 characters; trailing sign overpunch if S is specified; implicit decimal or scaling position
10	Numeric (9 S V P)	COMP-1 or INDEX†	Size < 15; binary integer
13	Numeric (9 S V P)	COMP-2	Normalized floating-point

†A PICTURE clause cannot be used to describe an item that specifies USAGE IS INDEX.

TABLE 3-4. DATA CLASS REPRESENTATION IN THE QUERY UPDATE SUBSCHEMA

Data Class	Query Update Subschema PICTURE Clause	Query Update Subschema USAGE Clause	Internal Representation
0	Alphanumeric (A X 9) (A specification of mixed As and 9s is treated as all Xs)	DISPLAY (or none)	Display code, alphanumeric
1	Alphabetic (A)	DISPLAY (or none)	Display code, alphabetic
3	Numeric (9 S and insertion and replacement characters)	DISPLAY or COMP (or none)	Display code numeric, $\leq$ 18 characters; trailing sign overpunch if S is specified
4	Numeric (9 S V P and insertion and replacement characters)	DISPLAY or COMP (or none)	Display code numeric, $\leq$ 18 characters; trailing sign overpunch if S is specified; implicit decimal or scaling position
10	Numeric (9 S V P and insertion and replacement characters)	COMP-1, INDEX†, or LOGICAL	Size < 15; binary integer
13	Numeric (9 S V P and insertion and replacement characters)	COMP-2	Normalized floating-point
14	Numeric (9 S V P and insertion and replacement characters)	DOUBLE	Normalized floating-point (2 words)
15	Numeric (9 S V P and insertion and replacement characters)	COMPLEX	Normalized floating-point (2 consecutive words)

†A PICTURE clause cannot be used to describe an item that specifies USAGE IS INDEX.

Repeating data items are specified by including the OCCURS clause in the data description entry. When a repeating data item in the schema is to be included in the subschema description, it must be described as a repeating data item and must conform to the following rules:

A subschema data item that is repeated a fixed number of times must correspond to a fixed occurrence data item in the schema.

The number of occurrences specified in the subschema OCCURS clause cannot be greater than the number of occurrences allowed by the schema description of the data item.

A subschema data item that is repeated a variable number of times must correspond to a variable occurrence data item in the schema.

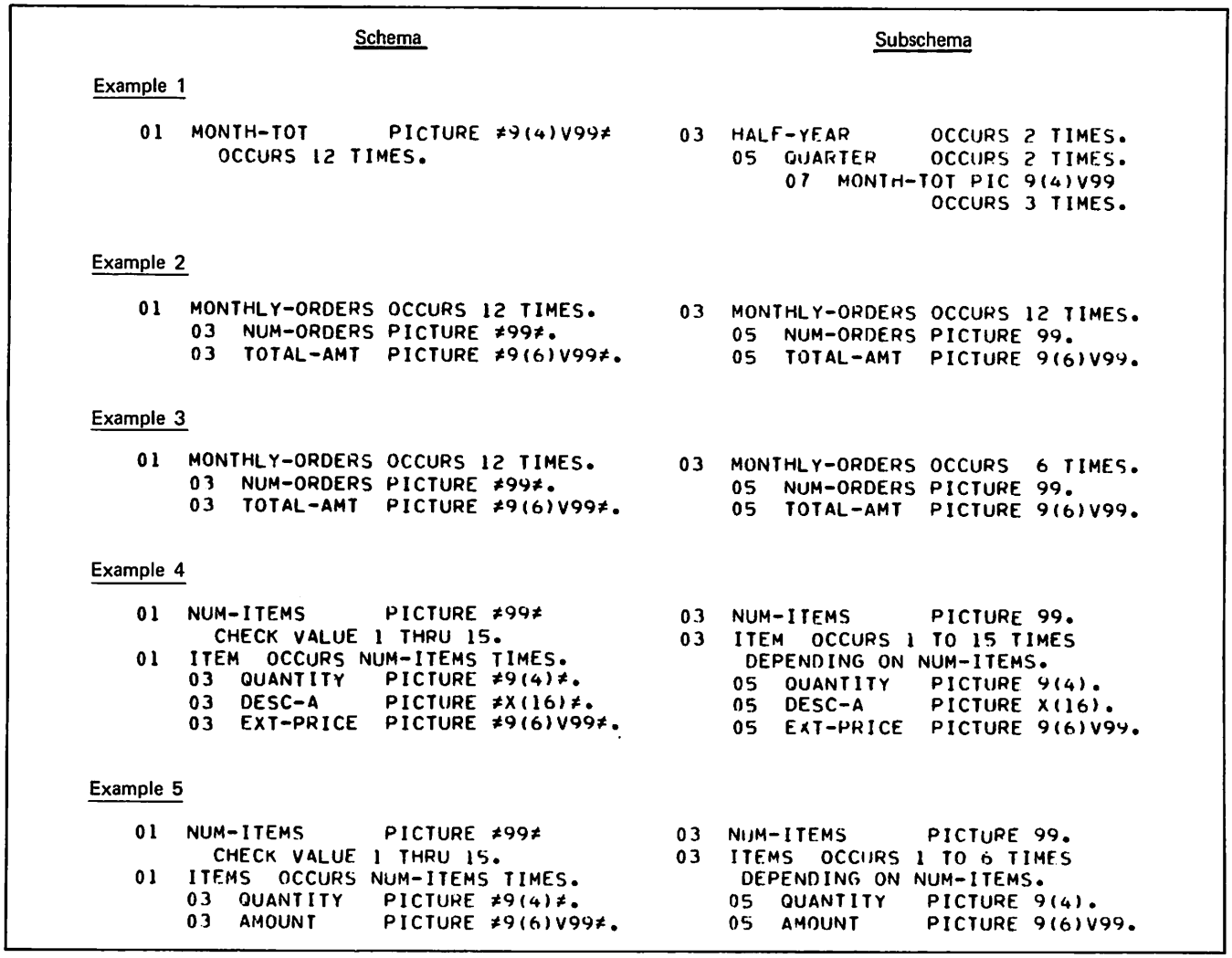
If a subschema data item corresponds to a schema data item that controls a variable occurrence data item, the subschema must include the variable occurrence data item.

A subschema variable occurrence data item must be the last item in the record.

When the OCCURS ... DEPENDING ON clause is used, the data name in the clause must refer to an elementary data item that precedes the entry containing the clause. The data item cannot be within a variable occurrence data item.

Vectors

A subschema vector is described with the OCCURS clause and the PICTURE or USAGE clause; in the schema, the OCCURS clause and either the TYPE or PICTURE clause are used to describe a vector. No data item can be subordinate to a vector. A schema vector that is included in the subschema can be described as a vector or as nested repeating groups. If the schema vector is described in the subschema as nested repeating groups, the elementary data item at the bottom of the hierarchy corresponds to the vector in the schema. Up to three levels of nested groups can be specified; the schema vector must be a fixed occurrence data item. Example 1 in figure 3-4 illustrates a schema vector described in the subschema as nested repeating groups.



Schema

Example 1

```
01  MONTH-TOT       PICTURE ≠9(4)V99≠
      OCCURS 12 TIMES.
```

```
03  HALF-YEAR       OCCURS 2 TIMES.
    05  QUARTER     OCCURS 2 TIMES.
        07  MONTH-TOT PIC 9(4)V99
                    OCCURS 3 TIMES.
```

Example 2

```
01  MONTHLY-ORDERS OCCURS 12 TIMES.
    03  NUM-ORDERS PICTURE ≠99≠.
    03  TOTAL-AMT  PICTURE ≠9(6)V99≠.
```

```
03  MONTHLY-ORDERS OCCURS 12 TIMES.
    05  NUM-ORDERS PICTURE 99.
    05  TOTAL-AMT  PICTURE 9(6)V99.
```

Example 3

```
01  MONTHLY-ORDERS OCCURS 12 TIMES.
    03  NUM-ORDERS PICTURE ≠99≠.
    03  TOTAL-AMT  PICTURE ≠9(6)V99≠.
```

```
03  MONTHLY-ORDERS OCCURS  6 TIMES.
    05  NUM-ORDERS PICTURE 99.
    05  TOTAL-AMT  PICTURE 9(6)V99.
```

Example 4

```
01  NUM-ITEMS      PICTURE ≠99≠
      CHECK VALUE 1 THRU 15.
01  ITEM  OCCURS NUM-ITEMS TIMES.
    03  QUANTITY   PICTURE ≠9(4)≠.
    03  DESC-A     PICTURE ≠X(16)≠.
    03  EXT-PRICE  PICTURE ≠9(6)V99≠.
```

```
03  NUM-ITEMS      PICTURE 99.
03  ITEM  OCCURS 1 TO 15 TIMES
      DEPENDING ON NUM-ITEMS.
    05  QUANTITY   PICTURE 9(4).
    05  DESC-A     PICTURE X(16).
    05  EXT-PRICE  PICTURE 9(6)V99.
```

Example 5

```
01  NUM-ITEMS      PICTURE ≠99≠
      CHECK VALUE 1 THRU 15.
01  ITEMS  OCCURS NUM-ITEMS TIMES.
    03  QUANTITY   PICTURE ≠9(4)≠.
    03  AMOUNT     PICTURE ≠9(6)V99≠.
```

```
03  NUM-ITEMS      PICTURE 99.
03  ITEMS  OCCURS 1 TO 6 TIMES
      DEPENDING ON NUM-ITEMS.
    05  QUANTITY   PICTURE 9(4).
    05  AMOUNT     PICTURE 9(6)V99.
```

Figure 3-4. Examples of Repeating Data Items

Repeating Groups

A repeating group is described with two or more data description entries. The first entry consists of the group data name and the OCCURS clause. Each additional entry is subordinate to the first entry and describes a repeating group, a vector, or an elementary data item. The vector or elementary data item contains the PICTURE clause. Up to three levels of nested groups can be specified. When repeating groups and vectors are nested, the vector is considered to be a repeating group and is included in the level count. Example 2 in figure 3-4 illustrates a repeating group data item.

NOTE

Refer to appendix F for recommendations on the use of repeating groups.

Fixed Occurrence Data Items

A subschema data item that occurs a fixed number of times must correspond to a fixed occurrence data item in the schema. Format 1 of the OCCURS clause specification is used to describe a fixed occurrence data item. (Refer to the OCCURS Clause subsection.)

The integer specified in the clause cannot be greater than the number of occurrences specified for the data item in the schema; it can be less than or equal to the schema number. Examples 2 and 3 in figure 3-4 illustrate fixed occurrence data items.

Variable Occurrence Data Items

A subschema data item that occurs a variable number of times must correspond to a variable occurrence data item in the schema. Format 2 of the OCCURS clause specification is used to describe a variable occurrence data item. (Refer to the OCCURS Clause subsection.)

The maximum number of times the data item can occur in the subschema must not be greater than the maximum allowed in the schema description. The OCCURS clause in the subschema specifies the minimum and maximum number of occurrences of the repeating data

item and also specifies the elementary data item that contains the actual number of occurrences for the record. The OCCURS clause in the schema specifies the elementary data item that contains the number of occurrences for the record; the data description entry for the controlling data item includes the CHECK clause, which indicates the minimum and maximum number of occurrences of the repeating data item.

The data item that designates the actual number of occurrences must be an elementary data item and it must precede the data description entry for the variable occurrence data item. If the schema data item that controls the variable occurrence data item is included in the subschema, the variable occurrence data item must also be included in the subschema. The controlling data item in the schema cannot be an independent data item in the subschema. In example 4 of figure 3-4, the data item NUM-ITEMS controls the variable occurrence data item ITEM. If NUM-ITEMS is included in the subschema, ITEMS must also be included.

The variable occurrence data item must be the last item in the record. Only subordinate data description entries can follow the entry containing the OCCURS clause. Variable occurrence data items are shown in examples 4 and 5 of figure 3-4.

Nonrepeating Group Items

The only group items in the schema are repeating groups. Nonrepeating group items can be specified in the subschema by designating a series of related data items as subordinate entries to a group data item that is inserted in the subschema description, which allows the application program to access several data items with one request.

A nonrepeating group item consists of three or more data description entries. The first entry does not correspond to an entry in the schema; it specifies the data name for the entire group data item. Each additional entry is subordinate to the first entry and describes a data item that corresponds to an entry in the schema. The only restriction placed on nonrepeating group items is that the relative hierarchy of the schema be preserved. Figure 3-5 illustrates the insertion of nonrepeating group items in the subschema.

A concatenated key is a primary or alternate record key that is declared in the schema and is composed of a series of contiguous elementary data items. To make use of this record key, a nonrepeating group item must be defined in the subschema with the same name as the key-name specified in the schema. The subordinate items for this group must be all of the concatenated data items (or the subschema aliases for these data-names) specified in the same order as in the schema KEY clause. Definition of other subordinate items for the group item is not allowed. An example of a concatenated key specification is shown in figure 3-6.

SUBSCHEMA PROGRAMMING CONVENTIONS

The COBOL or Query Update subschema source program consists of a series of statements that describe a portion of the data base. The rules, conventions, and hierarchical structures of DDL are similar to those of COBOL.

Schema	Subschema
01 EMPLOYEE-ID ...	03 EMPLOYEE-IDENT. ...
01 EMPLOYEE-NAME ...	05 EMPLOYEE-ID ...
01 SOC-SEC-NUM ...	05 EMPLOYEE-NAME ...
01 ADDR ...	05 SOC-SEC-NUM ...
01 CITY ...	03 EMP-ADDRESS. ...
01 STATE ...	05 ADDR ...
01 ZIP-CODE ...	05 CITY ...
	05 STATE ...
	05 ZIP-CODE ...

Figure 3-5. Insertion of Nonrepeating Group Items

Schema	Subschema
KEY ID IS CONCAT-KEY < ITEMA,ITEMB,ITEMC >	03 CONCAT-KEY.
	05 ITEMA ...
	05 ITEMB ...
	05 ITEMC ...

Figure 3-6. Concatenated Key Declaration

LANGUAGE ELEMENTS

DDL source statements are composed of clauses that contain reserved words, user-defined names, and literals. The use of these elements is described in the following paragraphs. The specific formats of the clauses are defined in the COBOL and Query Update Subschema Syntax subsection.

Reserved Words

Reserved words are English words, abbreviations, and acronyms that have special meaning to the DDL compiler. These words can be used only as shown in the format specifications. A reserved word must be spelled correctly; it cannot be replaced by another word. Over 350 words have been defined as reserved words. Appendix D contains a complete list of DDL reserved words used in COBOL and Query Update subschema definition.

Two types of reserved words are recognized by the DDL compiler: keywords and optional words. A keyword is a reserved word that must be used in a specific clause. Keywords are essential to convey the meaning of a clause to the compiler. An optional word is a reserved word that can be included in a clause to improve readability. Optional words are recognized by the compiler but are not needed to compile the object coding. In the format specifications, keywords are shown as uppercase words that are underlined; optional words are shown as uppercase words that are not underlined.

User-Defined Names

Many of the format specifications include names that the user supplies. User-defined names identify the schema, subschema, realms, records, data items, index names, and condition names. The type of name to be supplied is indicated in the format specification by a lowercase word.

The following rules apply to user-defined names except for some names used with Query Update (see the exceptions following the rules):

- A name can contain up to 30 characters.
- Letters (A-Z), digits (0-9), and hyphens (-) can be used.
- The first character must be a letter.
- A hyphen cannot be used to begin or end a name.
- Adjacent hyphens cannot be used.
- Spaces (blanks) cannot be used.
- A name cannot be spelled exactly the same as a reserved word.

In a Query Update subschema, realm and subschema names must conform to the above rules with the following exceptions:

- Realm and subschema names cannot contain any hyphens.
- If an alias is specified for a realm, then that name cannot contain hyphens.

Literals

In some formats, the user must supply a literal as part of the clause. A literal is a string of characters that represents a specific value. Literals are either numeric or nonnumeric.

Numeric Literals

A numeric literal can contain the numbers 0 through 9, the decimal point, and the plus or minus sign. A decimal point can be included in any character position except the rightmost position. A plus sign or a minus sign can precede the number. If a sign is not included, the literal is treated as a positive number.

The maximum size for a numeric literal is 30 digits; however, only 18 digits can be significant. Up to 12 leading or trailing zeros can be specified for input alignment.

Nonnumeric Literals

A nonnumeric literal is a string of up to 255 characters. The string must be enclosed in quotation marks. Any characters in the DDL character set, including the space, can be used in a nonnumeric literal. If a quotation mark is to be included in the literal, the quotation mark must be specified twice for each occurrence. For example, "A" "B" would yield the literal A"B.

Data Reference

Each user-defined name in the subschema must be capable of being uniquely referenced. Unless the name itself is unique because no other name has the identical spelling, a method for obtaining unique identification is necessary. Unique reference is recognized through the qualification, subscripting, and identifier concepts.

Qualification

Qualification is permitted in any clause, other than a RESTRICT clause, that references a data name. When a name exists within a hierarchy of one or more names, the higher level names can be used to make the name unique. The data name is written followed by the word OF or IN and the qualifier. The choice between OF or IN is based on readability; the two words are logically equivalent. Qualification mut be made to the level necessary to make the name unique; however, qualification can be used even when the name does not need to be qualified.

Subscripting

Subscripting within the DDL syntax is permitted only in the RESTRICT clause. Subscripts are used to indicate which occurrence of a repeating group or elementary item is to be referenced. The data name is written followed by a positive integer constant enclosed in parentheses. Specific rules for using subscripts are detailed in the following paragraphs.

Identifier

The term identifier is used to indicate a data name that is referenced uniquely through a combination of subscripts and qualifiers. When the term identifier appears in a RESTRICT clause, it assumes the format shown in figure 3-7.

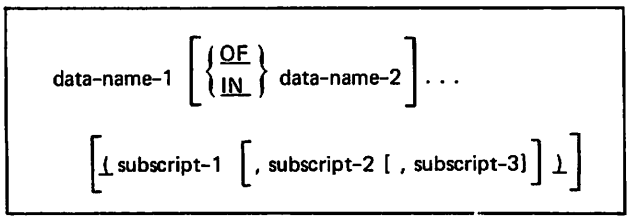


Figure 3-7. Identifier Format

Subscripting within the DDL syntax is permitted only in the RESTRICT clause. Qualifiers and subscripting are independent of each other. A maximum of five data-name-2 qualifiers can be specified. They must be listed in the order of innermost to outermost level of nesting of the group or record; hence the leftmost qualifier (data-name-2) is the term with the highest level number.

In the example in figure 3-8, valid references to item C are given for a COBOL subschema repeating group containing A, B, and C.

DDL Character Set

The set of characters recognized by the DDL compiler can be combined according to the specified rules to form names and values in the source program. The DDL character set consists of the letters A through Z, the numbers 0 through 9, and the following special characters:

	Blank or Space
+	Plus Sign
-	Minus Sign or Hyphen
,	Comma
;	Semicolon
*	Asterisk
.	Period or Decimal Point
"	Quotation Mark
(	Left Parenthesis
)	Right Parenthesis
$	Dollar Sign
/	Slash

Punctuation

Most punctuation marks in a DDL source program are optional. When punctuation marks are used, the rules are precise and must be followed exactly. The rules for using punctuation marks are as follows:

A period is required to terminate a division heading and each complete statement.

A period must be followed by at least one space. If the punctuation mark is in the last column of the statement area (column 72), a blank is assumed to exist immediately after column 72 and before column 73 of the program identification area.

A left parenthesis must not be followed by a space; a right parenthesis must not be preceded by a space.

At least one delimiter (a space, comma, or semicolon) must separate successive words in a statement.

Commas and semicolons can be used to separate clauses in a statement.

A period terminates the SS clause (Title Division), each AD clause (Alias Division), the RD clause (Realm Division), each entry introduced by a level number (Record Division), and the relation description entry (Relation Division).

DDL CODING

DDL source programs can be written on standard COBOL coding sheets. Coding a COBOL or Query Update subschema is similar to coding the Data Division of a COBOL application program. Columns 8 through 72 are used to write DDL statements.

Coding DDL Statements

The DDL source program defining a COBOL or Query Update subschema consists of up to five divisions; the Alias and Relation Divisions are optional. The

COBOL Subschema Repeating Group	Valid Identifiers for C
02 A OCCURS 3 TIMES.	C IN A (3, 5)
04 B OCCURS 6 TIMES.	C OF B OF A (3)
06 C PICTURE 999.	The default subscript for B is the value 1.
	C OF B OF A
	The default subscripts for both A and B have the value 1.

Figure 3-8. COBOL Subschema Qualification and Subscripting Example

division name begins in columns 8 through 11 and is followed by a space, the word DIVISION, and a period. The remainder of the line must be left blank.

A division heading is followed by one or more statements pertaining to that division. In the Title, Alias, Realm, and Relation Divisions, the statement begins in columns 8 through 72. In the Record Division, level number 01 begins in columns 8 through 72; all other level numbers begin in columns 12 through 72.

Sequence Numbers

A sequence number consisting of digits only can be entered in columns 1 through 6. The sequence number is optional and has no effect on the source program.

Continuation Lines

Words or literals can be continued from one line to the next. When a continuation line is written on the coding sheet, a hyphen must be entered in column 7. The continuation must begin in column 12.

Comment Lines

Comments can be printed on the source listing. A comment line is written on the coding sheet by entering an asterisk in column 7. The comment begins in column 8 through 72.

COBOL AND QUERY UPDATE SUBSCHEMA SYNTAX

The source program for a COBOL or Query Update subschema contains five divisions: Title Division, Alias Division, Realm Division, Record Division, and Relation Division. The following paragraphs define the format specifications for each clause that can be used in the source program.

The general format of a COBOL or Query Update subschema is shown in figure 3-9. For the rules governing the structure of the subschema, see the Subschema Structuring Conventions subsection.

```
TITLE DIVISION.
{title description entry}.

[ALIAS DIVISION.
 {alias description entry.} ...]

REALM DIVISION.
{realm description entry}.

RECORD DIVISION.
{record description entry.} ...

[RELATION DIVISION.
 {relation description entry.} ...]
```

Figure 3-9. General Format, COBOL and Query Update Subschema

TITLE DIVISION

The Title Division must be the first division in the DDL source program. It identifies the subschema being described and the schema to which the subschema applies. The format of the Title Division is as follows:

```
TITLE DIVISION.

{title description entry}.
```

The title description entry is a statement containing one clause that names the subschema and identifies the schema. One title description entry must be included in the Title Division. The DDL compiler accepts only one statement in this division.

SS Clause

The SS clause names the subschema being created and specifies the schema that describes the data base. COBOL or Query Update application programs reference the subschema by the name established in this clause. The format of the SS clause is shown in figure 3-10.

```
SS subschema-name WITHIN schema-name.
```

Figure 3-10. SS Clause Format

The subschema-name entered in this clause is the name used whenever the subschema is referenced after it has been compiled and stored in the subschema library. The name must be unique among subschemas associated with the designated schema.

The schema-name identifies the schema to which the subschema applies. The schema must exist within the system and be recognized by the CYBER Database Control System (CDCS). The schema describes the entire data base of which the subschema describes only a portion.

Other Clauses

Future software releases will provide other clauses that can be included in the title description entry. Only one clause is currently acceptable in the Title Division.

ALIAS DIVISION

The Alias Division is optional and, if included, must immediately follow the Title Division. This division provides the means to assign names that are to be used in the subschema in place of names defined in the schema. The format of the Alias Division is as follows:

```
ALIAS DIVISION.

{alias description entry.}...
```

The alias description entry is a statement that specifies a name to be used in the subschema and in the application programs instead of a name used in the schema. Each alias description entry assigns one alias. A separate entry is included for each name in the schema to be assigned an alias. One clause is used in an alias statement.

Assigning an alias in the subschema does not change the name in the schema; the alias is a substitute name that is used only in the subschema and in the application programs referencing that particular subschema. When an alias has been assigned, the name in the schema cannot be used in the subschema or in the application programs.

AD Clause

The AD clause assigns an alias to a name defined in the schema. The alias becomes the name recognized by the subschema and by the COBOL or Query Update application programs that reference the subschema. The AD clause eliminates the need to rewrite existing application programs that use names different from the names defined in the schema. The format of the AD clause is shown in figure 3-11.

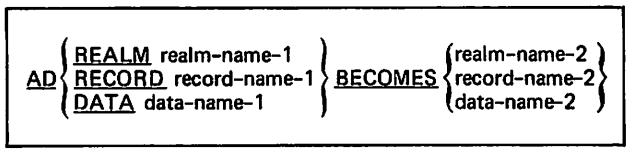


Figure 3-11. AD Clause Format

An alias can be assigned to a realm, a record, or a data item. A specific name in the schema can have only one alias in the subschema. Each realm-, record-, or data-name-1 must be a unique realm, record, or data name; however, data-name-1 can be qualified to make it unique. The same alias cannot be used for more than one realm-, record-, or data-name-2 entry. Data-name-2 can be qualified to make the name unique.

When a realm is assigned an alias, the REALM option is entered in the AD clause. The name entered as realm-name-1 must be defined in the schema as an area. The name entered as realm-name-2 is then the name used in the subschema and in the application programs to reference that area in the schema.

A record in the schema is assigned an alias by entering the RECORD option in the AD clause. The name entered as record-name-1 must be the name of a record defined in the schema. The name entered as record-name-2 then becomes the record name that is used in both the subschema and the application programs referencing the subschema.

A data item in the schema is assigned an alias by entering the DATA option in the AD clause. The name entered as data-name-1 must be the name of a data item in the schema. The name entered as data-name-2 then becomes the data name that is used in both the subschema and the application programs referencing the subschema.

The use of the AD clause in the Alias Division is illustrated in figure 3-12.

Other Clauses

Only the AD clause can be included in an alias description entry. No additional clauses are planned for the Alias Division.

REALM DIVISION

The Realm Division must be included in the DDL source program. If the Alias Division is included, the Realm Division immediately follows it; otherwise, the Realm Division follows the Title Division. The schema areas that are to be made available to the subschema as realms are specified in the Realm Division. The format of the Realm Division is as follows:

```
REALM DIVISION.

{realm description entry}.
```

The realm description entry is a statement consisting of one clause that identifies the schema areas to be used as realms in the subschema. One statement can specify as many realms as needed for the subschema.

RD Clause

The RD clause identifies the specific schema areas that are to be used in the subschema and in the COBOL or Query Update application programs referencing the subschema. An area in the schema is equivalent to a realm in the subschema. A realm is equivalent to a file in the application program. The format of the RD clause is shown in figure 3-13.

Schema

```
AREA NAME IS SALES.
RECORD NAME IS SALES-REC ...
01  CUSTOMER-ID  ...
01  INVOICE-NUM  ...
01  AMOUNT-DUE   ...
```

Subschema

```
ALIAS DIVISION.
AD REALM SALES BECOMES SALES-FILE.
AD RECORD SALES-REC BECOMES SALES.
AD DATA CUSTOMER-ID BECOMES CUST-NO.
AD DATA INVOICE-NUM BECOMES INV-NO.
AD DATA AMOUNT-DUE BECOMES AMT-DUE.
```

Figure 3-12. Assigning Aliases

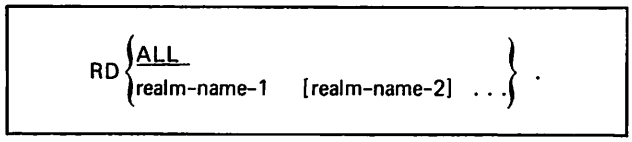


Figure 3-13. RD Clause Format

Realms for the subschema are selected from the areas in the schema. One or more realms are specified by entering the realm names in the RD clause. If an alias was assigned to a realm in the Alias Division, the alias is used in the RD clause. When all areas in the schema are to be made available to the subschema, the word ALL can be used in the RD clause. Unless ALL is specified, only those realms named in the RD clause are available to the subschema and to the programs referencing the subschema.

Other Clauses

Future software releases will provide other clauses that can be included in the realm description entry. Only one clause is currently acceptable in the Realm Division.

RECORD DIVISION, COBOL SUBSCHEMA

The Record Division immediately follows the Realm Division. It specifies the record type to be made available to a COBOL application program and describes the format of the data in each record type. The format of the Record Division is as follows:

```
RECORD DIVISION.

{record description entry.}...
```

The Record Division is similar to the File Section in the Data Division of a COBOL program. When a COBOL program specifies a subschema, the record description entries in the subschema Record Division replace the file description entries normally written in the COBOL program.

The records within the realms specified in the Realm Division are described in the Record Division. Only those records that are to be used by the COBOL programs are included. The record description entry consists of a series of statements (data description entries) that describe the data as it is used by the COBOL programs.

The first data description entry must be level number 01, the record name entry. Subsequent entries begin with level numbers 02 through 49 for group and elementary data items, level number 66 for renaming a data item, and level number 88 for defining values to be associated with a condition. The formats for data description entries are shown in figure 3-14.

JUSTIFIED Clause

The JUSTIFIED clause specifies nonstandard positioning of data within a receiving field. The format of the JUSTIFIED clause is shown in figure 3-15.

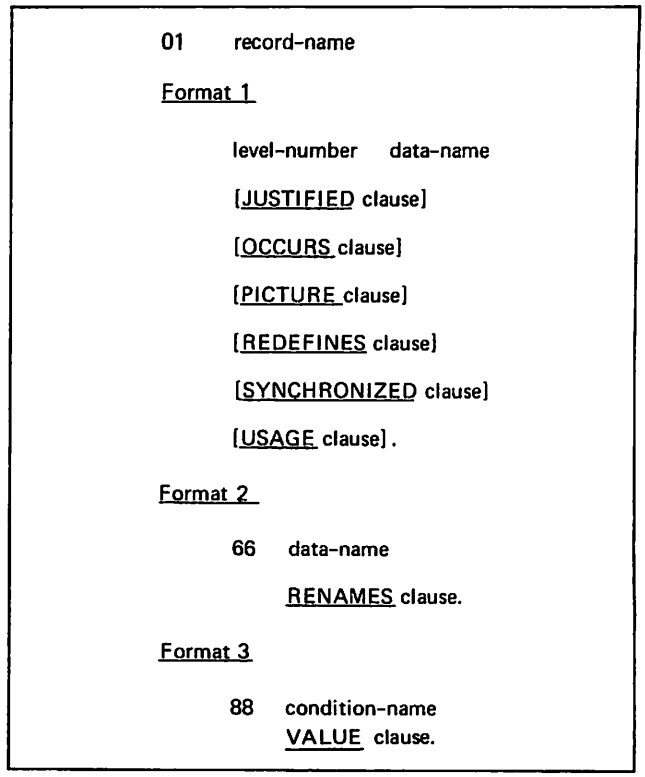


Figure 3-14. Formats of Data Description Entries, COBOL Subschema

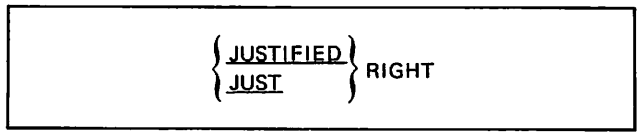


Figure 3-15. JUSTIFIED Clause Format, COBOL Subschema

This clause can be specified only for nonnumeric elementary data items. JUST is the legal abbreviation for JUSTIFIED.

The JUSTIFIED clause overrides the normal positioning of data when the size of the receiving field does not equal the number of characters in the data item. When the receiving field contains fewer character positions than the data item, positioning occurs as follows:

In normal positioning, the data item is aligned at the leftmost character position and truncated at the right.

If the JUSTIFIED clause is specified, the data item is aligned at the rightmost character position and truncated at the left.

When the receiving field contains more character positions than the data item, positioning occurs as follows:

In normal positioning, the data item is aligned at the leftmost character position and blank filled at the right.

If the JUSTIFIED clause is specified, the data item is aligned at the rightmost character position and blank filled at the left.

Figure 3-16 illustrates character positioning.

OCCURS Clause

The OCCURS clause is used to indicate a repeated data item where all occurrences of the data item are identical in every respect except value. The data item can be an elementary item or a group item. The format of the OCCURS clause is shown in figure 3-17.

NOTE

Refer to appendix F for recommendations on the use of repeating groups.

The KEY and INDEXED options are not used by CDCS; the options can be included in an OCCURS clause for use by the COBOL application program. The OCCURS clause must not be specified in a data description entry that has level number 66 or 88.

Picture	Justified	Data Item	Receiving Field	
9(5)	Right	1 2 3		Illegal; item is numeric.
X(5)		A B C	A B C Δ Δ	Left-justified normally.
X(5)	Right	A B C	Δ Δ A B C	Right-justified; blanks filled in.
X(2)		A B C	A B	Left-justified normally; right character truncated.
X(2)	Right	A B C	B C	Right-justified; left character truncated.

Figure 3-16. Character Positioning

Format 1

```
OCCURS integer-2 TIMES

[ { ASCENDING  } KEY IS data-name-2 [data-name-3] . . . ] . . .
  { DESCENDING }

[INDEXED BY index-name-1 [index-name-2] . . . ]
```

Format 2

```
OCCURS integer-1 TO integer-2 TIMES DEPENDING ON data-name-1

[ { ASCENDING  } KEY IS data-name-2 [data-name-3] . . . ] . . .
  { DESCENDING }

[INDEXED BY index-name-1 [index-name-2] . . ]
```

Figure 3-17. OCCURS Clause Format, COBOL Subschema

A data description entry with format 1 of the OCCURS clause can be subordinate to another entry with either format of the OCCURS clause. An entry with format 2 cannot be subordinate to an entry with the OCCURS clause. Up to three levels of nested data items can be specified with the OCCURS clause. When repeating groups and vectors are nested, the vector is considered to be a repeating group and is included in the level count.

Integer-1 and integer-2 must be positive numbers. In format 2, integer-2 must be greater than integer-1. The value of integer-1 can be zero; integer-2 must never be zero.

An elementary data item described with the OCCURS clause must also be described with the PICTURE or USAGE clause. A group data item cannot be described with both the OCCURS clause and the PICTURE clause.

When an item occurs a fixed number of times, format 1 is used and integer-2 specifies the exact number of occurrences. When an item occurs a variable number of times, format 2 is used and the number of occurrences for each record is determined as follows:

Integer-1 represents the minimum number of occurrences.

Integer-2 represents the maximum number of occurrences.

Data-name-1 references a data item whose current value represents the number of occurrences. The value of data-name-1 must be a positive value within the range of integer-1 through integer-2.

In format 2, data-name-1 names an elementary item that is unique or can be made unique by qualification. It cannot be subscripted and it should not be described as COMPUTATIONAL-2. The size of the data item cannot exceed six characters. The elementary data item must precede the group data item that references it.

A data item that occurs a variable number of times must be the last item in a record description entry. The data description entry that contains the OCCURS clause can only be followed by subordinate entries.

If the OCCURS clause is used with a group data item, any data-name belonging to the group must be referenced by subscripting or indexing whenever it is used as an operand, unless the data-name is the object of a REDEFINES clause. Other clauses specified with the OCCURS clause apply to each occurrence of the data item.

ASCENDING/DESCENDING KEY Option

When repeated data items are sequenced, the ASCENDING KEY or DESCENDING KEY option specifies the order according to the values of data-name-2, data-name-3, etc. This option can be included in the OCCURS clause for use by the COBOL program; it is not used by CDCS.

The data-names in the KEY option can be qualified. Data-name-2 references either the entry containing the OCCURS clause or a subordinate entry in a repeating group item. Data-name-3 and any additional data-names specify entries subordinate to the group item that contains the OCCURS clause.

If data-name-2 does not reference the entry containing the OCCURS clause, the following conditions apply to the key data-names:

All items identified by the data-names must be subordinate to the group item containing the OCCURS clause.

None of the key data-names can reference an item that contains an OCCURS clause.

No item between the entry containing the KEY option and the entries identified by the key data-names can be described with an OCCURS clause.

INDEXED BY Option

The INDEXED BY option is used when the entry containing the OCCURS clause or an item subordinate to it is referenced by indexing. This option is included only for use by the COBOL program; it is not used by CDCS.

The index-name specified in this option is not described anywhere else in the Record Division. It is not data and cannot be associated with any data hierarchy. The format and allocation of the index-name are hardware dependent. The index-name must be unique; it cannot be qualified.

PICTURE Clause

The PICTURE clause describes the general characteristics of an elementary data item in terms of its size and class. The location of an operational sign or an assumed decimal point can also be indicated in the clause. The format of the PICTURE clause is shown in figure 3-18.

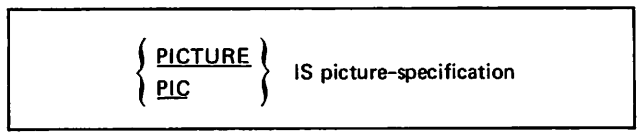


Figure 3-18. PICTURE Clause Format, COBOL Subschema

This clause can be specified only for elementary data items. It cannot be used with a level 66, level 88, or index data item. PIC is the legal abbreviation for PICTURE.

The class of a data item is determined by the type of characters in the picture-specification. The characters 9, S, V, and P are used to describe numeric data items. An alphabetic data item is described by the character A. The picture-specification for an alphanumeric data item contains the character X or any combination of the characters X, 9, and A.

The size of a data item is determined by the number of 9, X, or A characters in the picture-specification. The characters S, V, and P are not counted in determining the size. Consecutive identical characters in the string can be specified by a number in parentheses following the character. For example: 99999999 is equivalent to 9(8), XXXXXXXX is equivalent to X(8), and 9999AAAA is equivalent to 9(4)A(4); each indicates a data item with eight character positions.

The picture-specification can contain a maximum of 30 characters, including parentheses; however, a pictured item can be larger. A picture-specification containing the character A repeated 75 times is too long, but A(75) is a valid description for a data item with 75 alphabetic characters. Numeric data items can contain only 18 significant digits; additional leading or trailing zeros for decimal point alignment can be specified up to a total of 30 characters.

When a PICTURE clause is specified with a USAGE clause, the specifications must be compatible. For example, COMP usage must have a numeric picture specification.

Alphabetic Data Items

The picture-specification to describe an alphabetic data item can contain only the character A. The A can be specified as many times as necessary as long as the size of the item does not exceed 32767 characters.

The function of the characters in a PICTURE clause picture-specification for an alphabetic data item is as follows:

A Each A in the picture-specification represents a character position that can contain either a letter of the alphabet or a space (blank).

Some typical alphabetic data items are shown in figure 3-19.

Picture-Specification	Data Value	Display Code Stored
AAAAA or A(5)	COSTS	C O S T S
AAAA or A(4)	WXYZ	W X Y Z

Figure 3-19. Alphabetic Data Items

Numeric Data Items

The picture-specification to describe a numeric data item can contain a combination of the characters 9, S, V, and P. Each 9 represents a significant digit; a maximum of 18 significant digits can be specified. The character P is used to indicate leading or trailing zeros for decimal point alignment. The combined number of positions indicated by the characters 9 and P cannot exceed 30 characters.

Unsigned numeric data items consist of a combination of the digits 0 through 9. In a signed numeric data item, the minus sign is combined with the rightmost digit in the item; the plus sign is not carried in the number unless it already exists in the data. The representation of the minus sign in the rightmost digit is shown in figure 3-20.

Digit	0	1	2	3	4	5	6	7	8	9
Minus Representation	V	J	K	L	M	N	O	P	Q	R

Figure 3-20. Minus Sign Representation

The function of the characters in a PICTURE clause picture-specification for a numeric data item is as follows:

9 Each 9 in the picture-specification represents a digit position that can contain a number. The 9 is counted in determining the size of the data item.

S The character S is used in the picture-specification to indicate that the data item has an operational sign. The S must be the leftmost character in the picture-specification; it cannot appear more than once. The operational sign does not occupy a character position in the data item and is not counted in its size.

V The character V is used in the picture-specification to indicate the position of an assumed decimal point. A V as the rightmost character in the picture-specification is redundant. The V cannot appear more than once in the picture-specification. Since the assumed decimal point does not occupy a character position, the V is not counted in the size of the data item. An explicit decimal point is valid in the schema but not in the subschema.

P The character P in the picture-specification indicates an assumed decimal scaling position. It is used to specify an assumed decimal point when its position is not within the number that appears in the data item. If the assumed decimal point extends beyond the rightmost digit, one P is specified for each implied position between the rightmost digit and the assumed decimal point. Similarly, if the assumed decimal point extends beyond the leftmost digit, one P is specified for each implied position between the leftmost digit and the assumed decimal point. Since the P indicates an assumed decimal point, a V in the picturespecification would be redundant. The character P is not counted in determining the size of the data item; however, it is counted in determining the maximum number of digit positions (30) in numeric data items.

Some typical numeric data items are shown in figure 3-21.

Picture-Specification	Data Value	Display Code Stored
999	123	1 2 3
99V999	12345	1 2 3 4 5
S99V99	+1234	1 2 3 +4
PPP9999	.0001234	↑0 0 0 1 2 3 4
SPPP9999	-.0001234	↑0 0 0 1 2 3 −4
S999PPP	-123000.	1 2 −3 0 0 0↑

Figure 3-21. Numeric Data Items

Alphanumeric Data Items

The picture-specification to describe an alphanumeric data item contains either a combination of the characters 9, A, and X or only the character X. The size of the data item cannot exceed 32767 characters.

The function of the characters 9 and A is the same as for numeric and alphabetic data items; the function of the character X is as follows:

X Each X in the picture-specification represents a character position that can contain any character in the DDL character set.

Some typical alphanumeric data items are shown in figure 3-22.

REDEFINES Clause

The REDEFINES clause allows data to be described in an alternate format. The redefined data item is given a new name; since it is still the same data value, it occupies the same physical area in memory. The REDEFINES clause is not used by CDCS; it can be included in the subschema for use by the COBOL application program. The format of the REDEFINES clause is shown in figure 3-23.

```
level-number  data-name-1 REDEFINES data-name-2
```

Figure 3-23. REDEFINES Clause Format, COBOL Subschema

NOTE

Refer to appendix F for recommendations on the use of the REDEFINES clause.

This clause can be specified for both elementary and nonrepeating group data items. The level number and size of data-name-1 and data-name-2 must be identical. The REDEFINES clause cannot be used with a level 01, level 66, or level 88 data item.

The data description entry for data-name-2 cannot contain an OCCURS clause. Data-name-2 can be subordinate to an entry containing an OCCURS clause, but data-name-2 cannot be subscripted or indexed to reference a specific occurrence of the data item. Data-name-1 can be described as a repeating data item as long as the total number of character positions is equal to the number of characters in data-name-2.

Picture-Specification	Data Value	Display Code Stored
XXXXXXXX or X(8)	ABCD-***	A B C D - * * *
XXXXXXXX or X(8)	123.4567	1 2 3 . 4 5 6 7
AAAA999	ABCD123	A B C D 1 2 3
A(4)9(3)	ABCD123	A B C D 1 2 3

Figure 3-22. Alphanumeric Data Items

Data-name-2 must not be qualified even if it is not unique; data-name-2 can only refer to the previous entry with the same level number. If the REDEFINES clause is used for an elementary data item, no entries can be specified between the entry referenced by data-name-2 and the entry containing the REDEFINES clause. If a group data item is being redefined, only subordinate entries can be specified between the entry referenced by data-name-2 and the entry containing the REDEFINES clause.

Multiple redefinitions of the same data item are allowed; data-name-2 in each entry must reference the entry that originally defined the data item. Within the Procedure Division of the COBOL application program, the original data-name and the redefined data-names can be used to reference the data item.

The use of the REDEFINES clause is illustrated in figure 3-24.

SYNCHRONIZED Clause

The SYNCHRONIZED clause causes an elementary data item to start or end on a word boundary within the computer memory. The format of the SYNCHRONIZED clause is shown in figure 3-25.

This clause can only be used in a data description entry for an elementary data item. SYNC is the legal abbreviation for SYNCHRONIZED. If neither LEFT nor RIGHT is specified, LEFT is assumed and a warning diagnostic is issued.

Data items are normally packed without regard for machine words. The SYNCHRONIZED clause causes the data item to be allocated as many whole computer words as needed to contain the item. No other data item can occupy any of the character positions between the leftmost and rightmost word boundaries of the words allocated to the data item. If the data item does not use all the character positions between the boundaries, the unused character positions are included in:

The size of any group item to which the elementary item belongs

The character positions redefined when the data item is referenced in a REDEFINES clause

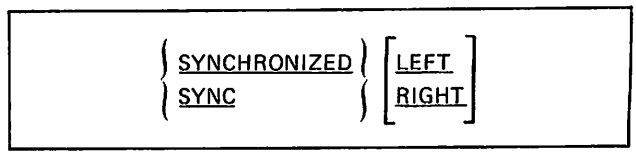


Figure 3-25. SYNCHRONIZED Clause Format, COBOL Subschema

SYNCHRONIZED LEFT places the leftmost character of the data item in the leftmost position of the first word allocated to the data item. Subsequent characters are placed in successive positions to the right. If more than one word is needed, consecutive words are allocated from left to right. The unused portion of the last word is not available.

SYNCHRONIZED RIGHT places the rightmost character of the data item in the rightmost position of the last word allocated to the data item. Preceding characters are placed in successive positions to the left. If more than one word is needed, consecutive words are allocated to the left. The unused portion of the first word is not available.

When a COBOL application program references a data item that has been described with the SYNCHRONIZED clause, the original size of the data item is used in determining any action that depends on size. The original size of the data item is the size indicated in the PICTURE clause.

The operational sign in a data item that is synchronized appears in its normal position. The LEFT or RIGHT option has no effect on the positioning of the operational sign.

When an entry is described with both the SYNCHRONIZED clause and the OCCURS clause, each occurrence of the data item is synchronized. This also applies to a synchronized item that is subordinate to an entry containing the OCCURS clause.

USAGE Clause

The USAGE clause specifies the internal representation of a data item in terms of the primary use of the data item. The format of the USAGE clause is shown in figure 3-26.

Schema	Subschema
Example 1	
01 STOCK-NUM PICTURE ≠X(6)≠.	03 STOCK-NUM PICTURE X(6). 03 STOCK-ID REDEFINES STOCK-NUM. 05 TYPE-ID PICTURE XX. 05 COLOR PICTURE X(4).
Example 2	
02 CHARGE-NUM PICTURE ≠X(10)≠.	02 CHARGE-ID REDEFINES CHARGE-NUM PIC XX OCCURS 5 TIMES.

Figure 3-24. Redefining Data Items

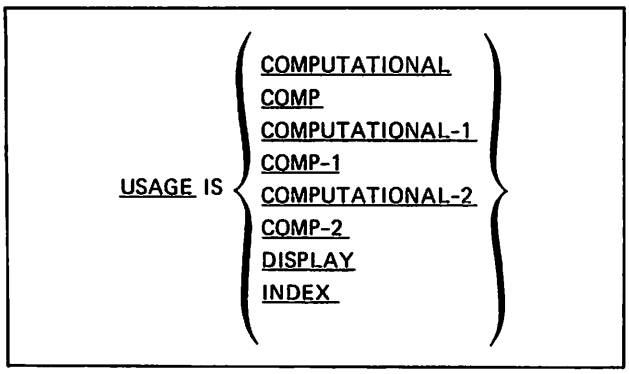


Figure 3-26. USAGE Clause Format, COBOL Subschema

This clause can describe a data item at any level. When the USAGE clause is specified for a group item, it applies to each subordinate item in the group. The USAGE clause of an elementary item in a group cannot contradict the USAGE clause of the group item. If the USAGE clause is not specified for an elementary item or for any group to which the elementary item belongs, DISPLAY is assumed. COMP, COMP-1, and COMP-2 are legal abbreviations for COMPUTATIONAL, COMPUTATIONAL-1, and COMPUTATIONAL-2.

A data item described as COMPUTATIONAL, COMPUTATIONAL-1, or COMPUTATIONAL-2 must be numeric with a size not exceeding 18 significant digits. If a group item is described as computational, the elementary items within the group are all computational; however, the group item itself is not computational and cannot be used in computations.

When a PICTURE clause is specified with a USAGE clause, the specifications must be compatible.

COMPUTATIONAL Option

A COMPUTATIONAL data item has a decimal numeric value. The PICTURE picture-specification can only contain the characters 9 (digit position), S (operational sign), V (implied decimal point), and P (assumed decimal scaling position).

COMPUTATIONAL-1 Option

The COMPUTATIONAL-1 format describes a fixed-point integer, which is represented internally as a 48-bit binary integer, right-aligned in the word.

A corresponding PICTURE clause must be numeric. If a PICTURE clause is not specified, the default is PIC 9.

COMPUTATIONAL-2 Option

A COMPUTATIONAL-2 data item is stored as a normalized floating-point binary representation of a decimal number. The decimal point location is carried in the data item itself as a binary exponent. The data item is single-precision and is stored in one computer word.

A corresponding PICTURE clause must be numeric. If a PICTURE clause is not specified, the default is PIC 9.

DISPLAY Option

A DISPLAY data item is stored in display code format. The data item can be alphabetic, numeric, or alphanumeric. When no USAGE clause is associated with a data item, DISPLAY is the default format.

INDEX Option

The USAGE IS INDEX clause specifies a data item that is used with indexed tables. The index data item contains a value that must correspond to an occurrence number of a table element. It cannot be a conditional variable (level 88 data item). The INDEX option is not used by CDCS; it can be included in the subschema description for use by the COBOL application program.

An index data item is an elementary item, one computer word in length, and binary in format. Its mode corresponds to an item described as COMPUTATIONAL-1; the item must be numeric and must not exceed 18 digits.

The SYNCHRONIZED, JUSTIFIED, and PICTURE clauses cannot be used to describe a group or elementary item that specifies USAGE IS INDEX.

RENAMES Clause

A level 66 data description entry uses the RENAMES clause. This clause permits alternate grouping and renaming of data items. The RENAMES clause is not used by CDCS; it can be included in the subschema description for use by the COBOL application program. The format of the RENAMES clause is shown in figure 3-27.

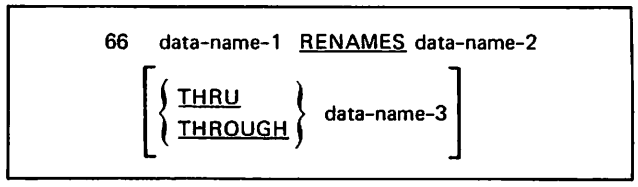


Figure 3-27. RENAMES Clause Format, COBOL Subschema

This clause is always used in a level 66 entry. No other clause can be included in a level 66 entry. The RENAMES clause cannot be used to rename another level 66 entry or a level 88 entry. THRU is the legal abbreviation for THROUGH.

More than one level 66 entry can rename the same data item. Level 66 entries must be the last entries defined in a record. No entry can be subordinate to a level 66 entry.

Data-name-1 cannot be used as a qualifier; data-name-2 and data-name-3 can be qualified. The entries referred to by data-name-2 and data-name-3 cannot be described with the OCCURS clause or be subordinate to an entry described with the OCCURS clause. When the THRU option is included, none of the data items within the specified range can be variable-occurrence data items. Data-name-2 and data-name-3 must be names of elementary items or groups of elementary items. Data-name-3 cannot be the same name as data-name-2 or subordinate to data-name-2.

The THRU option is used to rename a series of consecutive elementary or group items. Data-name-1 is a group item that includes all elementary items beginning with data-name-2 (or the first elementary item if it is a group item) and ending with data-name-3 (or the last elementary item if it is a group item).

If the THRU option is not used, data-name-1 can be either an elementary item or a group item. Data-name-1 is an elementary item if data-name-2 is an elementary item or a group item if data-name-2 is a group item.

The use of the RENAMES clause is illustrated in figure 3-28.

VALUE Clause

The VALUE clause specifies the values associated with a condition name. It is used in a level 88 data description entry. The VALUE clause is not used by CDCS; it can be included in the subschema description for use by the COBOL application program. The format of the VALUE clause is shown in figure 3-29.

This clause defines one or more values or ranges of values for a condition-name. It is always used in a level 88 entry and is the only clause that can be used in a level 88 entry. THRU is the legal abbreviation for THROUGH. The THRU option is used to specify a range of values. Literals that specify a range must appear in ascending order.

The condition-name is the name assigned to the values an item can assume. The following rules apply to a condition-name and its associated VALUE clause:

The condition-name can describe only an elementary data item; the name cannot be used to describe a repeating group or renamed item.

The condition-name must immediately follow the data item to which it refers.

Schema

```
01  JANUARY  PICTURE ≠X(31)≠.
01  FEBRUARY PICTURE ≠X(29)≠.
01  MARCH    PICTURE ≠X(31)≠.
.
.
.
01  DECEMBER PICTURE ≠X(31)≠.
```

Subschema

```
03  CURRENT-YEAR.
    05  JANUARY  PICTURE X(31).
    05  FEBRUARY PICTURE X(29).
    .
    .
    05  DECEMBER PICTURE X(31).
    66  FIRST-HALF RENAMES JANUARY
          THRU JUNE.
    66  SECOND-HALF RENAMES JULY
          THRU DECEMBER.
    66  FIRST-QUARTER RENAMES
          JANUARY THRU MARCH.
    .
    66  LAST-QUARTER RENAMES
          OCTOBER THRU DECEMBER.
```

Figure 3-28. Renaming Data Items

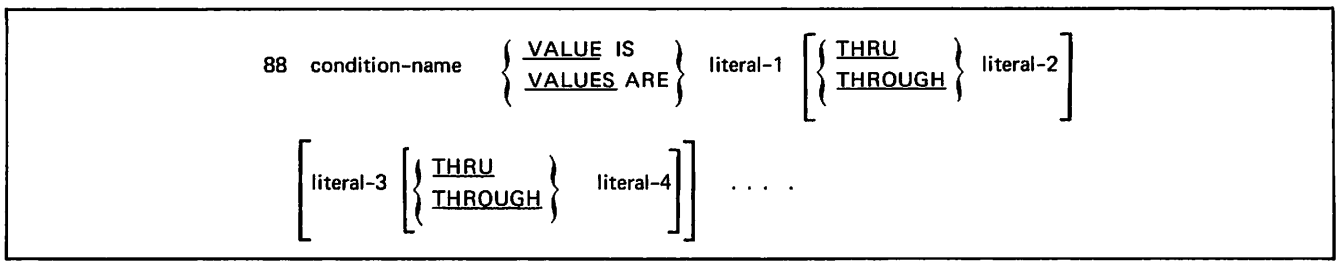


Figure 3-29. VALUE Clause Format, COBOL Subschema

The condition-name must be unique only when describing the same data item; the name can be duplicated for different data items. For example:

```
02  ITEM1 PIC X(20).
    88 NAME1 VALUE "ABCDE".
02  ITEM2 PIC 999.
    88 NAME1 VALUE 123.
```

Level 88 Literals

Specific rules apply to the use of literals in the VALUE clause. The rules are defined in the following text. The term target item is used to identify the elementary data item described by the condition-name.

Alphanumeric display target items (data class 0)

Literals specified for alphanumeric display target items must be nonnumeric and can contain a maximum of 255 characters and digits. The literal length must not exceed the target item picture length. If USAGE DISPLAY is specified with no PICTURE clause, literal length must not exceed one character. Literals must be enclosed in quotation marks. Example 1 in figure 3-30 illustrates valid literals for alphanumeric display target items.

Alphabetic display target items (data class 1)

Literals specified for alphabetic display target items must be nonnumeric and can contain a maximum of 255 characters. The literal length must not exceed the target item picture length. USAGE DISPLAY cannot be used because data class defaults to 0 rather than to 1. Literals must be enclosed in quotation marks. Example 2 in figure 3-30 illustrates valid literals for alphabetic display target items.

Integer display target items (data class 3)

Literals specified for integer display target items must be numeric and can contain a maximum of 18 digits. The literal length must not exceed the target item picture length. If USAGE COMP is specified with no PICTURE clause, literal length must not exceed one digit and a separate operational sign is not allowed. Example 3 in figure 3-30 illustrates valid literals for integer display target items.

Example 1

```
02  ITEMALPNUM.
    88  L88 VALUE ≠ABCD≠, ≠1234≠  THRU ≠56789≠.
```

Example 2

```
02  ITEMALPHA PIC A(5).
    88  L88 VALUE ≠ABCD≠, ≠ABCDE≠.
```

Example 3

```
02  ITEMNUM PIC  9999.
    88  L88 VALUE 1, 100, 5555.

02  ITEMNUM PIC  S9999.
    88  L88 VALUE -100, -20, 0, +100, 5555.

02  ITEMNUM USAGE COMP.
    88  L88 VALUE 0, 1, 5, 9.
```

Example 4

```
02  ITEMFXPT PIC 999V99.
    88  L88 VALUE 0, 100, 100.5, 100.55.

02  ITEMFXPT PIC S999V99.
    88  L88 VALUE -100.55, -100, 0, +100, 100.55.

02  ITEMFXPT PIC PPP999.
    88  L88 VALUE .0001, .00012, .000123.

02  ITEMFXPT PIC 999PPP.
    88  L88 VALUE 1000, 123.000.
```

Figure 3-30. Examples of Valid Level 88 Literals

Fixed-point, binary integer, and coded floating-point display target items (data class 4, 10, 13)

Literals specified for fixed-point, binary integer, and coded floating-point must be numeric. The literal length must not exceed the target item picture length. If scaling zeros (P) are not specified, picture length cannot exceed 18 characters excluding the decimal point. If scaling zeros are specified, picture length cannot exceed 30 characters and significant digit count cannot exceed 18 characters. If USAGE COMP-1 (data class 10) or USAGE COMP-2 (data class 13) is specified with no PICTURE clause, literal length must not exceed one character. Example 4 in figure 3-30 illustrates valid literals for these target items.

Level 88 Figurative Constants

A figurative constant is a fixed value with a predefined name. When the name is referenced in a source program, the constant associated with the name is automatically generated in the object program. A figurative constant can be substituted for a literal in the VALUE clause.

Specific rules apply to the use of figurative constants in the VALUE clause. The rules are defined in the following text. The term target item is used to identify the elementary data item described by the condition-name.

HIGH-VALUES

When the HIGH-VALUES figurative constant is specified, the target item must be alphanumeric. A PICTURE clause with the character X or USAGE DISPLAY must be specified. Example 1 in figure 3-31 illustrates valid use of HIGH-VALUES.

LOW-VALUES, SPACES

When the LOW-VALUES or SPACES figurative constant is specified, the target item must be either alphabetic or alphanumeric, or USAGE DISPLAY must be specified. Example 2 in figure 3-31 illustrates valid use of LOW-VALUES and SPACES.

ZERO, ZEROS, ZEROES

When the ZERO, ZEROS, or ZEROES figurative constant is specified, the target item must not be alphabetic (data class 1); all other data classes are allowed. Example 3 in figure 3-31 illustrates valid use of ZERO, ZEROS, and ZEROES.

ALL "literal"

When ALL "literal" or ALL figurative-constant is specified, the target item must be alphanumeric or USAGE DISPLAY must be specified. The length of the literal specified in the ALL clause must not exceed the target item picture length. If USAGE DISPLAY is specified with no PICTURE clause, literal length must not exceed one character.

Example 1

```
02  ITEMA PIC X(25).
    88  L88 VALUE HIGH-VALUES.

02  ITEMA USAGE DISPLAY.
    88  L88 VALUE HIGH-VALUES.
```

Example 2

```
02  ITEMA PIC X(25).
    88  L88 VALUE LOW-VALUES.

02  ITEMA USAGE DISPLAY.
    88  L88 VALUE SPACES.
```

Example 3

```
02  ITEMA PIC 999.
    88  L88 VALUE ZEROES.

02  ITEMA USAGE COMP.
    88  L88 VALUE ZERO.
    88  L881 VALUE ZEROS.
```

Example 4

```
02  ITEMA PIC X(5).
    88  L88 VALUE ALL ≠ABC≠.
    88  L881 VALUE ALL SPACES.
    88  L882 VALUE ALL HIGH-VALUES.
    88  L883 VALUE ALL LOW-VALUES.
```

Figure 3-31. Examples of Valid Level 88 Figurative Constants

RECORD DIVISION, QUERY UPDATE SUBSCHEMA

The Record Division immediately follows the Realm Division. It specifies the record type to be made available to a Query Update application program and describes the format of the data in each record type. The format of the Record Division is as follows:

```
RECORD DIVISION.

{record description entry.}...
```

The records within the realms specified in the Realm Division are described in the Record Division. Only those records that are to be used by the Query Update programs are included. The record description entry consists of a series of statements (data description entries) that describe the data as it is used by the Query Update programs.

The first data description entry must be level number 01, the record name entry. Subsequent entries begin with level numbers 02 through 49 for group and elementary data items, level number 66 for renaming a data item, and level number 88 for defining values to be associated with a condition. The formats for data description entries are shown in figure 3-32.

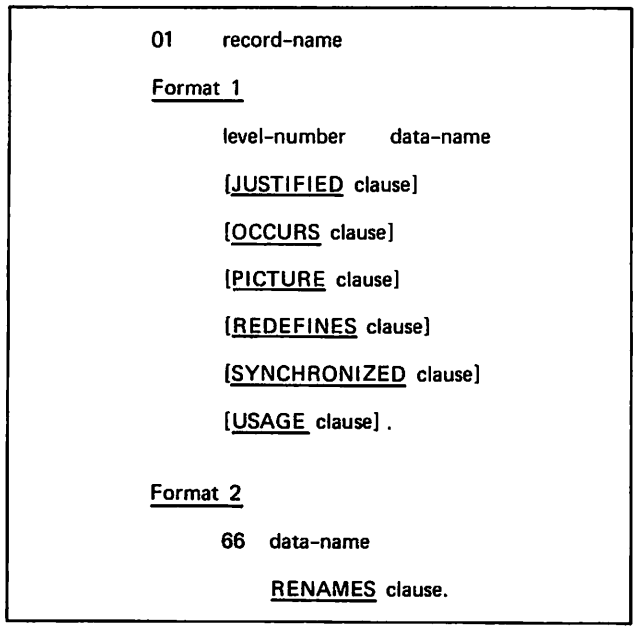


Figure 3-32. Formats of Data Description Entries, Query Update Subschema

JUSTIFIED Clause

The JUSTIFIED clause specifies nonstandard positioning of data within a receiving field. The format of the JUSTIFIED clause is shown in figure 3-33.

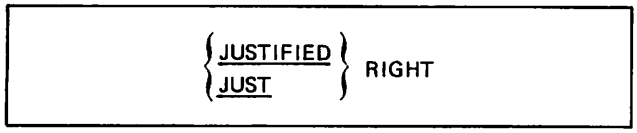


Figure 3-33. JUSTIFIED Clause Format, Query Update Subschema

This clause can be specified only for nonnumeric elementary data items. JUST is the legal abbreviation for JUSTIFIED.

The JUSTIFIED clause overrides the normal positioning of data when the size of the receiving field does not equal the number of characters in the data item. When the receiving field contains fewer character positions than the data item, positioning occurs as follows:

In normal positioning, the data item is aligned at the leftmost character position and truncated at the right.

If the JUSTIFIED clause is specified, the data item is aligned at the rightmost character position and truncated at the left.

When the receiving field contains more character positions than the data item, positioning is as follows:

In normal positioning, the data item is aligned at the leftmost character position and blank filled at the right.

If the JUSTIFIED clause is specified, the data item is aligned at the rightmost character position and blank filled at the left.

Refer to figure 3-16 for examples of character positioning.

OCCURS Clause

The OCCURS clause is used to indicate a repeated data item where all occurrences of the data item are identical in every respect except value. The data item can be an elementary item or a group item. The format of the OCCURS clause is shown in figure 3-34.

NOTE

Refer to appendix F for recommendations on the use of repeating groups.

A data description entry with format 1 of the OCCURS clause can be subordinate to another entry with either format of the OCCURS clause. An entry with format 2 cannot be subordinate to an entry with the OCCURS clause. Up to three levels of nested data items can be specified with the OCCURS clause. When repeating groups and vectors are nested, the vector is considered to be a repeating group and is included in the level count. The OCCURS clause must not be specified in a data description that has level number 66.

Integer-1 and integer-2 must be positive numbers. In format 2, integer-2 must be greater than integer-1. The value of integer-1 can be zero; integer-2 must never be zero.

An elementary data item described with the OCCURS clause must also be described with the PICTURE or USAGE clause. A group data item cannot be described with both the OCCURS clause and the PICTURE clause.

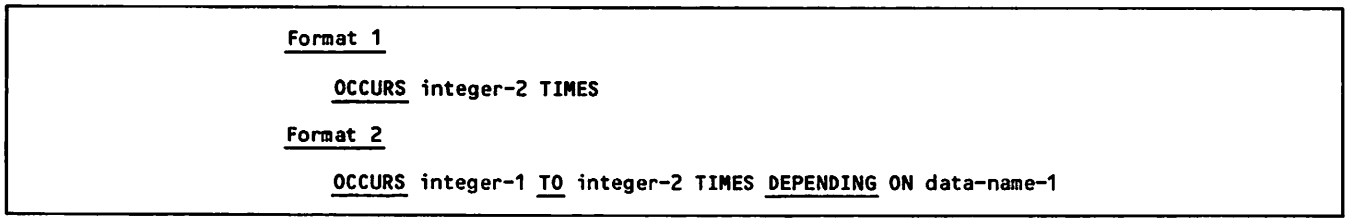


Figure 3-34. OCCURS Clause Format, Query Update Subschema

When an item occurs a fixed number of times, format 1 is used and integer-2 specifies the exact number of occurrences. When an item occurs a variable number of times, format 2 is used and the number of occurrences for each record is determined as follows:

Integer-1 represents the minimum number of occurrences.

Integer-2 represents the maximum number of occurrences.

Data-name-1 references a data item whose current value represents the number of occurrences. The value of data-name-1 must be a positive value within the range of integer-1 through integer-2.

In format 2, data-name-1 names an elementary item that is unique or can be made unique by qualification. It cannot be subscripted and it should not be described as COMPUTATIONAL-2. The size of the data item cannot exceed six characters. The elementary data item must precede the group data item that references it.

A data item that occurs a variable number of times must be the last item in a record description entry. The data description entry that contains the OCCURS clause can only be followed by subordinate entries.

If the OCCURS clause is used with a group data item, any data-name belonging to the group must be referenced by subscripting or indexing whenever it is used as an operand, unless the data-name is the object of a REDEFINES clause. Other clauses specified with the OCCURS clause apply to each occurrence of the data item.

PICTURE Clause

The PICTURE clause describes the general characteristics of an elementary data item in terms of its size and class. The location of an operational sign or an assumed decimal point can also be indicated in the clause. The format of the PICTURE clause is shown in figure 3-35.

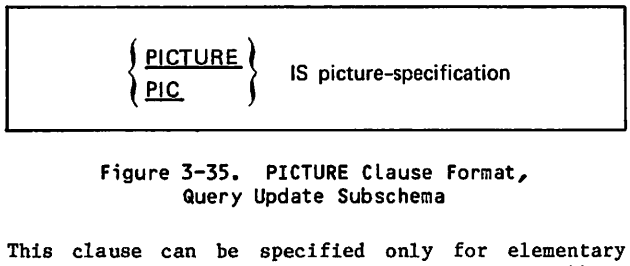


Figure 3-35. PICTURE Clause Format, Query Update Subschema

This clause can be specified only for elementary data items. It cannot be used with a level 66 or index data item. PIC is the legal abbreviation for PICTURE.

The class of a data item is determined by the type of characters in the picture-specification. The characters 9, S, V, and P are used to describe numeric data items. An alphabetic data item is described by the character A. The picture-specification for an alphanumeric data item contains the character X or any combination of the characters X, 9, and A.

The size of a data item is determined by the number of 9, X, or A characters in the picture-specification. The characters S, V, and P are not counted in determining the size. Consecutive identical characters in the string can be specified by a number in parentheses following the character. For example: 99999999 is equivalent to 9(8), XXXXXXXX is equivalent to X(8), and 9999AAAA is equivalent to 9(4)A(4); each indicates a data item with eight character positions.

The picture-specification can contain a maximum of 30 characters, including parentheses; however, a pictured item can be larger. A picture-specification containing the character A repeated 75 times is too long, but A(75) is a valid description for a data item with 75 alphabetic characters. Numeric data items can contain 18 digits, 14 of which are significant; additional leading or trailing zeros for decimal point alignment can be specified up to a total of 30 characters.

When a PICTURE clause is specified with a USAGE clause, the specifications must be compatible. For example, COMPUTATIONAL usage must have a numeric picture specification.

The application of format and punctuation to a numeric source data item at the time it is displayed is called editing. Editing characters are insertion characters or replacement characters.

The displayed format is referred to as a report item rather than as a source data item. Insertion and replacement characters are not included in the size of a numeric source item. Insertion and replacement characters must be included in the size of a report item.

Alphabetic Data Items

The picture-specification to describe an alphabetic data item can contain only the character A. The A can be specified as many times as necessary as long as the size of the item does not exceed 32767 characters.

The function of the characters in a PICTURE clause picture-specification for an alphabetic data item is as follows:

A Each A in the picture-specification represents a character position that can contain either a letter of the alphabet or a space (blank).

Refer to figure 3-19 for examples of alphabetic data items.

Numeric Data Items

The picture-specification to describe a numeric data item can contain a combination of the characters 9, S, V, and P. Each 9 represents a significant digit; a maximum of 18 digits can be specified (only 14 digits are significant). The character P is used to indicate leading or trailing zeros for decimal point alignment. The combined number of positions indicated by the characters 9 and P cannot exceed 30 characters.

Unsigned numeric data items consist of a combination of the digits 0 through 9. In a signed numeric data item, the minus sign is combined with the rightmost digit in the item; the plus sign is not carried in the number unless it already exists in the data. The representation of the minus sign in the rightmost digit is shown in figure 3-20.

The function of the characters in a PICTURE clause picture-specification for a numeric data item is as follows:

- 9 Each 9 in the picture-specification represents a digit position that can contain a number. The 9 is counted in determining the size of the data item.
- S The character S is used in the picture-specification to indicate that the data item has an operational sign. The S must be the leftmost character in the picture-specification; it cannot appear more than once. The operational sign does not occupy a character position in the data item and is not counted in its size.
- V The character V is used in the picture-specification to indicate the position of an assumed decimal point. A V as the rightmost character in the picture-specification is redundant. The character V cannot appear more than once in the picture-specification. Since the assumed decimal point does not occupy a character position, the V is not counted in the size of the data item. An explicit decimal point is valid in the schema, but it is not valid in the subschema.
- P The character P in the picture-specification indicates an assumed decimal scaling position. It is used to specify an assumed decimal point when its position is not within the number that appears in the data item. If the assumed decimal point extends beyond the rightmost digit, one P is specified for each implied position between the rightmost digit and the assumed decimal point. Similarly, if the assumed decimal point extends beyond the leftmost digit, one P is specified for each implied position between the leftmost digit and the assumed decimal point. Since the character P indicates an assumed decimal point, the character V in the picture-specification would be redundant. The character P is not counted in determining the size of the data item; however, it is counted in determining the maximum number of digit positions (30) in numeric data items.

Refer to figure 3-21 for examples of numeric data items.

Alphanumeric Data Items

The picture-specification to describe an alphanumeric data item contains either a combination of the characters 9, A, and X or only the character X. The size of the data item cannot exceed 32767 characters.

The function of the characters 9 and A is the same as for numeric and alphabetic data items; the function of the character X is as follows:

- X Each X in the picture-specification represents a character position that can contain any character in the DDL character set.

Refer to figure 3-22 for examples of alphanumeric data items.

Insertion Characters

Insertion characters in the picture of an item appear in the edited data item and are counted in the size of the report item. A maximum of 63 characters is allowed. Insertion characters for the PICTURE clause are as follows:

- $ A single dollar sign specified as the leftmost symbol in an item picture appears as the leftmost character in a report item.
- + A plus sign specified as the first or last symbol in an item picture is inserted in the indicated character position of the report item if the data contains a positive operational sign or is unsigned. If the data is negative, a minus sign is inserted in the indicated character position.
- - A minus sign specified as the first or last symbol in an item picture is inserted in the indicated character position of the report item if the data contains a negative operational sign. If the data is positive, a blank is inserted in the indicated character position.
- . A period is used in a report item picture to represent a decimal point. When it is used, a decimal point appears in the report item as a character in the indicated character position.
- , A comma used in the picture of an item is inserted in the corresponding character position of the report item.
- / A slash used in the picture of an item is inserted in the corresponding character position of the report item.
- 0 A zero used in the picture of an item is inserted in the corresponding character position of the report item.
- B The character B used in the picture of an item causes a blank to be inserted in the corresponding character position of the report item.
- CR The CR symbol represents credit and can be specified only at the rightmost position of the picture of an item. The symbol is inserted in the last two character positions of the report item if the data contains a negative operational sign. If the data is positive or unsigned, the last two character positions are set to blanks. Since the credit symbol always results in two characters - CR or blanks - it is included as two characters in the size of the report item.

DB The DB symbol represents debit and can be specified only at the rightmost position of the picture of an item. The debit symbol has the same result as the credit symbol.

Examples of insertion characters are shown in figure 3-36.

Replacement Characters

A replacement character in the picture of an item suppresses leading zeros in the source data and replaces them with the specified character or a blank in the report item. Only one type of replacement character can be used in a picture. Replacement characters for the PICTURE clause are as follows:

Z One character Z is specified as the leftmost symbol in an item picture for each leading zero to be suppressed and replaced by blanks. The character Z can be preceded by one of the insertion characters and interspersed with any of the insertion characters decimal point, comma, zero, or B.

No zeros are suppressed to the right of the first nonzero digit whether a Z is present or not, nor are any zeros to the right of an assumed or actual decimal point suppressed unless the value of the data is zero and all character positions in the item are described by a Z. In this special case, even the actual decimal point is suppressed and the edited item is all blanks.

If a $ + or - precedes the Z characters, it is inserted in the far left character position of the item even if succeeding zeros in the item are suppressed. In the special case where the value of the data is zero and all the character positions following the $ + or - are Zs, the $ + or - is replaced by blanks.

If a comma, zero, or B is encountered before zero suppression terminates, the character is not inserted in the edited data item. Rather, the character is suppressed and a blank is inserted in its place.

* The asterisk causes leading zeros to be replaced by an asterisk instead of a blank. It is specified in the same way as the editing character Z and follows the same rules, except an actual decimal point is not replaced by an asterisk when the value of the data is zero.

Picture-Specification	Data Value	Displayed Item
$99	48	$48
$99.99	4834	$48.34
9,999	4834	4,834
+999	292	+292
+999	+292	+292
+999	-292	-292
999-	-292	292-
-999	+292	△292
999-	292	292△
$BB999.99	24321	$△△243.21
$00999.99	24321	$00243.21
99.99CR	-1134	11.34CR
99.99CR	1134	11.34△△
99.99DB	-2376	23.76DB
99.99DB	2376	23.76△△

Figure 3-36. Examples of Insertion Characters

$ When the dollar sign is used as a replacement character to suppress leading zeros, it acts as a floating sign and is inserted directly preceding the first nonsuppressed character. One more dollar sign than the number of zeros to be suppressed must be specified. This dollar sign is always present in the edited data whether or not any zero suppression occurs. The remaining dollar signs act in the same way as the Z characters to suppress leading zeros.

\+ When a plus sign is used as a replacement character, it is a floating sign. The plus sign is specified one more time than the number of leading zeros to be suppressed. It functions in the same way as the floating dollar sign. A plus sign is placed directly preceding the first nonsuppressed character if the edited data is positive or unsigned; a minus sign is placed in this position if the edited data is negative.

\- When a minus sign is used as a replacement character, it is a floating sign. The minus sign is specified one more time than the number of leading zeros to be suppressed. It functions in the same way as the floating plus sign. A minus sign is placed directly preceding the first nonsuppressed character if the edited data is negative; a blank is placed in this position if the edited data is positive or unsigned.

Examples of replacement characters are shown in figure 3-37. Examples of picture editing are shown in figure 3-38.

Picture-Specification	Data Value	Displayed Item
ZZ999	00923	Δ Δ 9 2 3
ZZZ99	00923	Δ Δ 9 2 3
ZZZZ.ZZ	000000	Δ Δ Δ Δ Δ Δ Δ
S***.99	00923	S * * 9 . 2 3
SSS9.99	000824	Δ Δ S 8 . 2 4
- - - 9.99	00526̄	Δ Δ - 5 . 2 6
SSS.99	3265	S 3 2 . 6 5

Figure 3-37. Examples of Replacement Characters

Picture-Specification	Data Value	Displayed Item
ZZZ,999.99	12345	Δ 1 2 , 3 4 5 . 0 0
Z99,999.99	1234	Δ 0 0 , 0 1 2 . 3 4
$ZZZ,ZZ9.99	123	$ Δ Δ Δ Δ Δ Δ 1 . 2 3
$ZZZ,ZZZ.99	12	$ Δ Δ Δ Δ Δ Δ Δ . 1 2
$***,**9.99	1234	$ * * 1 , 2 3 4 . 0 0
$***,***.99	123456	$ 1 2 3 , 4 5 6 . 0 0
$***,***.99	123	$ * * * * * 1 . 2 3
+999,999	12	+ 0 0 0 , 0 1 2
-ZZZ,ZZZ	12̄	- Δ Δ Δ Δ Δ 1 2
$ZZZ,ZZ9.99CR	123456̄	$ 1 2 3 , 4 5 6 . 0 0 C R
$ZZZ,ZZ9.99DB	123	$ Δ Δ Δ Δ Δ Δ 1 . 2 3 Δ Δ
$(4),SS9.99	1234	Δ Δ Δ Δ $ 1 2 3 . 4 0
$(4),SS.99	0000	Δ Δ Δ Δ Δ Δ Δ $. 0 0
- - - -,- - - .99	12̄	Δ Δ Δ Δ Δ Δ Δ - . 1 2
BBBB,BBB.99	12	Δ Δ Δ Δ Δ Δ Δ Δ . 1 2
$SSS,SZZ.99	12	illegal picture
S99.99	12	illegal picture

Figure 3-38. Examples of Picture Editing

REDEFINES Clause

The REDEFINES clause allows data to be described in an alternate format. The redefined data item is given a new name; since it is still the same data value, it occupies the same physical area in memory. The REDEFINES clause is not used by CDCS; it can be included in the subschema for use by the Query Update application program. The format of the REDEFINES clause is shown in figure 3-39.

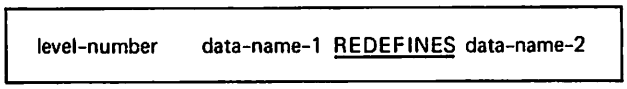


Figure 3-39. REDEFINES Clause Format, Query Update Subschema

NOTE

Refer to appendix F for recommendations on the use of the REDEFINES clause.

This clause can be specified for both elementary and nonrepeating group data items. The level number and size of data-name-1 and data-name-2 must be identical. The REDEFINES clause cannot be used with a level 01 or level 66 data item.

The data description entry for data-name-2 cannot contain an OCCURS clause. Data-name-2 can be subordinate to an entry containing an OCCURS clause, but data-name-2 cannot be subscripted or indexed to reference a specific occurrence of the data item. Data-name-1 can be described as a repeating data item as long as the total number of character positions is equal to the number of characters in data-name-2.

Data-name-2 must not be qualified even if it is not unique; data-name-2 can only refer to the previous entry with the same level number. If the REDEFINES clause is used for an elementary data item, no entries can be specified between the entry referenced by data-name-2 and the entry containing the REDEFINES clause. If a group data item is being redefined, only subordinate entries can be specified between the entry referenced by data-name-2 and the entry containing the REDEFINES clause.

Multiple redefinitions of the same data item are allowed; data-name-2 in each entry must reference the entry that originally defined the data item. Within the Query Update application program, the original data-name and the redefined data-names can be used to reference the data item.

Refer to figure 3-24 for examples of the use of the REDEFINES clause.

SYNCHRONIZED Clause

The SYNCHRONIZED clause causes an elementary data item to start or end on a word boundary within the computer memory. The format of the SYNCHRONIZED clause is shown in figure 3-40.

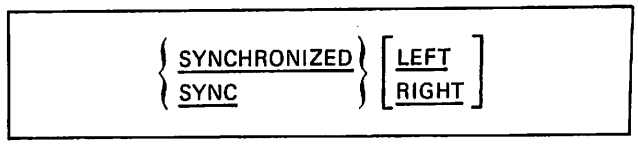


Figure 3-40. SYNCHRONIZED Clause Format, Query Update Subschema

This clause can only be used in a data description entry for an elementary data item. SYNC is the legal abbreviation for SYNCHRONIZED. If neither LEFT nor RIGHT is specified, LEFT is assumed and a warning diagnostic is issued.

Data items are normally packed without regard for machine words. The SYNCHRONIZED clause causes the data item to be allocated as many whole computer words as needed to contain the item. No other data item can occupy any of the character positions between the leftmost and rightmost word boundaries of the words allocated to the data item. If the data item does not use all the character positions between the boundaries, the unused character positions are included in:

The size of any group item to which the elementary item belongs

The character positions redefined when the data item is referenced in a REDEFINES clause

SYNCHRONIZED LEFT places the leftmost character of the data item in the leftmost position of the first word allocated to the data item. Subsequent characters are placed in successive positions to the right. If more than one word is needed, consecutive words are allocated from left to right. The unused portion of the last word is not available.

SYNCHRONIZED RIGHT places the rightmost character of the data item in the rightmost position of the last word allocated to the data item. Preceding characters are placed in successive positions to the left. If more than one word is needed, consecutive words are allocated to the left. The unused portion of the first word is not available.

When a Query Update application program references a data item that has been described with the SYNCHRONIZED clause, the original size of the data item is used in determining any action that depends on size. The original size of the data item is the size indicated in the PICTURE clause.

The operational sign in a data item that is synchronized appears in its normal position. The LEFT or RIGHT option has no effect on the positioning of the operational sign.

When an entry is described with both the SYNCHRONIZED clause and the OCCURS clause, each occurrence of the data item is synchronized. This also applies to a synchronized item that is subordinate to an entry containing the OCCURS clause.

USAGE Clause

The USAGE clause specifies the internal representation of a data item in terms of the primary use of the data item. The format of the USAGE clause is shown in figure 3-41.

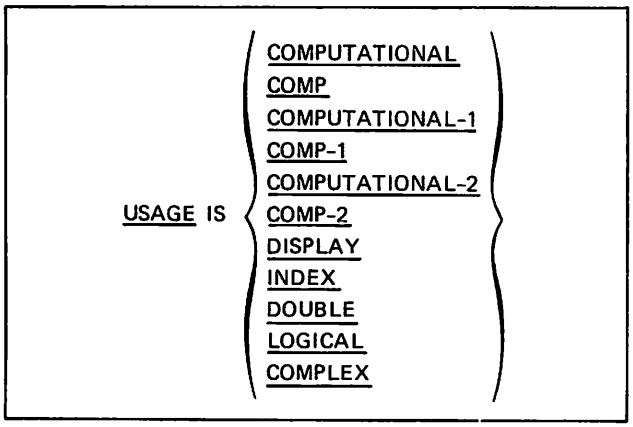


Figure 3-41. USAGE Clause Format, Query Update Subschema

This clause can describe a data item at any level. When the USAGE clause is specified for a group item, it applies to each subordinate item in the group. The USAGE clause of an elementary item in a group cannot contradict the USAGE clause of the group item. If the USAGE clause is not specified for an elementary item or for any group to which the elementary item belongs, DISPLAY is assumed. COMP, COMP-1, and COMP-2 are legal abbreviations for COMPUTATIONAL, COMPUTATIONAL-1, and COMPUTATIONAL-2.

A data item described as COMPUTATIONAL, COMPUTATIONAL-1, or COMPUTATIONAL-2 must be numeric with a size not exceeding 18 digits. If a group item is described as computational, the elementary items within the group are all computational; however, the group item itself is not computational and cannot be used in computations.

When a PICTURE clause is specified with a USAGE clause, the specifications must be compatible.

COMPUTATIONAL Option

A COMPUTATIONAL data item has a decimal numeric value. The PICTURE picture-specification can contain the characters 9 (digit position), S (operational sign), V (implied decimal point), and P (assumed decimal scaling position); editing characters can also be included.

COMPUTATIONAL-1 Option

The COMPUTATIONAL-1 format describes a fixed-point integer, which is represented internally as a 48-bit binary integer, right-aligned in the word.

A corresponding PICTURE clause must be numeric and can include editing characters. If a PICTURE clause is not specified, the default is PIC 9.

COMPUTATIONAL-2 Option

A COMPUTATIONAL-2 data item is stored as a normalized floating-point binary representation of a decimal number. The decimal point location is carried in the data item itself as a binary exponent. The data item is single precision and is stored in one computer word.

A corresponding PICTURE clause must be numeric and can include editing characters. If a PICTURE clause is not specified, the default is PIC 9.

DISPLAY Option

A DISPLAY data item is stored in display code format. The data item can be alphabetic, numeric, or alphanumeric. When no USAGE clause is associated with a data item, DISPLAY is the default format.

INDEX Option

The USAGE IS INDEX clause specifies a data item that is used with indexed tables. The index data item contains a value that must correspond to an occurrence number of a table element. The INDEX option is not used by CDCS; it can be included in the subschema description for use by the Query Update application program.

An index data item is an elementary item, one computer word in length, and binary in format. Its mode corresponds to an item described as COMPUTATIONAL-1; the item must be numeric and must not exceed 18 digits.

The SYNCHRONIZED, JUSTIFIED, and PICTURE clauses cannot be used to describe a group or elementary item that specifies USAGE IS INDEX.

DOUBLE Option

A DOUBLE data item is stored as a normalized floating-point number that occupies two computer words. The data item can be 29 digits.

A corresponding PICTURE clause must be numeric and can include editing characters. If a PICTURE clause is not specified, the default is PIC 9.

LOGICAL Option

A LOGICAL data item assumes only the values true or false. When this option is selected, Query Update displays the true/false condition rather than the value of the described data. A corresponding PICTURE clause must be numeric.

COMPLEX Option

A COMPLEX data item is an ordered pair of signed or unsigned real constants; the first represents the real part of the complex number and the second represents the imaginary part of the complex number.

A corresponding picture clause must be numeric and can include editing characters. If a PICTURE clause is not specified, the default is PIC 9.

RENAMES Clause

A level 66 data description entry uses the RENAMES clause. This clause permits alternate grouping and renaming of data items. The RENAMES clause is not used by CDCS; it can be included in the subschema description for use by the Query Update application program. The format of the RENAMES clause is shown in figure 3-42.

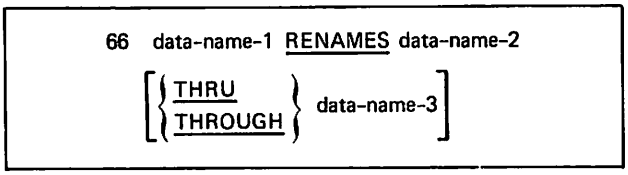


Figure 3-42. RENAMES Clause Format, Query Update Subschema

This clause is always used in a level 66 entry. No other clause can be included in a level 66 entry. The RENAMES clause cannot be used to rename another level 66 entry. THRU is the legal abbreviation for THROUGH.

More than one level 66 entry can rename the same data item. Level 66 entries must be the last entries defined in a record. No entry can be subordinate to a level 66 entry.

Data-name-1 cannot be used as a qualifier; data-name-2 and data-name-3 can be qualified. The entries referred to by data-name-2 and data-name-3 cannot be described with the OCCURS clause or be subordinate to an entry described with the OCCURS clause. When the THRU option is included, none of the data items within the specified range can be variable-occurrence data items. Data-name-2 and data-name-3 must be names of elementary items or groups of elementary items. Data-name-3 cannot be the same name as data-name-2 or subordinate to data-name-2.

The THRU option is used to rename a series of consecutive elementary or group items. Data-name-1 is a group item that includes all elementary items beginning with data-name-2 (or the first elementary item if it is a group item) and ending with data-name-3 (or the last elementary item if it is a group item).

If the THRU option is not used, data-name-1 can be either an elementary item or a group item. Data-name-1 is an elementary item if data-name-2 is an elementary item or a group item if data-name-2 is a group item.

Refer to figure 3-28 for examples of renaming data items.

RELATION DIVISION

The Relation Division is the last division in the DDL source program. It is optional and, if included, must immediately follow the Record Division. The Relation Division identifies the specific realm (area or file) relationships that are to be used in the subschema and in the COBOL or Query Update application programs referencing the subschema. The Relation Division also specifies the qualification criteria that must be satisfied by records that are to be returned to the application programs from the data base. The format of the Relation Division is as follows:

```
RELATION DIVISION.

[relation description entry.] ...
```

The relation description entry is a statement consisting of the RN and RESTRICT clauses that specify the relation name and the record qualification restrictions. Each relation description entry identifies one relation.

RN Clause

The RN clause specifies the name of the relation. It must be the first clause in the relation description entry. The format of the RN clause is shown in figure 3-43.

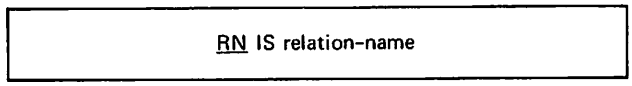


Figure 3-43. RN Clause Format

The relation-name must be unique among all relation and realm names in the subschema. It must be the same name specified in the RELATION NAME clause in the schema. The RN clause can be the only clause specified in the relation description entry if record qualification is not desired.

The subschema Record Division must contain record descriptions for all record types that are referenced within a relationship named in the subschema. All realms in which the record types are defined must be declared in the subschema Realm Division.

RESTRICT Clause

The RESTRICT clause specifies the record qualification criteria that must be satisfied by a record occurrence that is to be returned to the application program work area. Record qualification using the RESTRICT clause is optional; if a relationship is to be referenced by a program, the RN clause is required whether or not qualification is specified. The format of the RESTRICT clause is shown in figure 3-44.

Only one RESTRICT clause can be included for a given record. A maximum of 1024 entities (identifiers, operators, literals, or data-names) can appear in RESTRICT clauses for any one relation.

Record-name designates the record type to which the qualification restrictions pertain. The record type must be described in the Record Division of the subschema. The record name must be contained in a realm that has been joined by the relation named in the preceding RN clause.

The identifiers represent data items whose values are examined by CDCS to determine whether a record occurrence qualifies to be returned to the user's work area. They are termed qualifier identifiers. Refer to the Identifier subsection (under Data Reference).

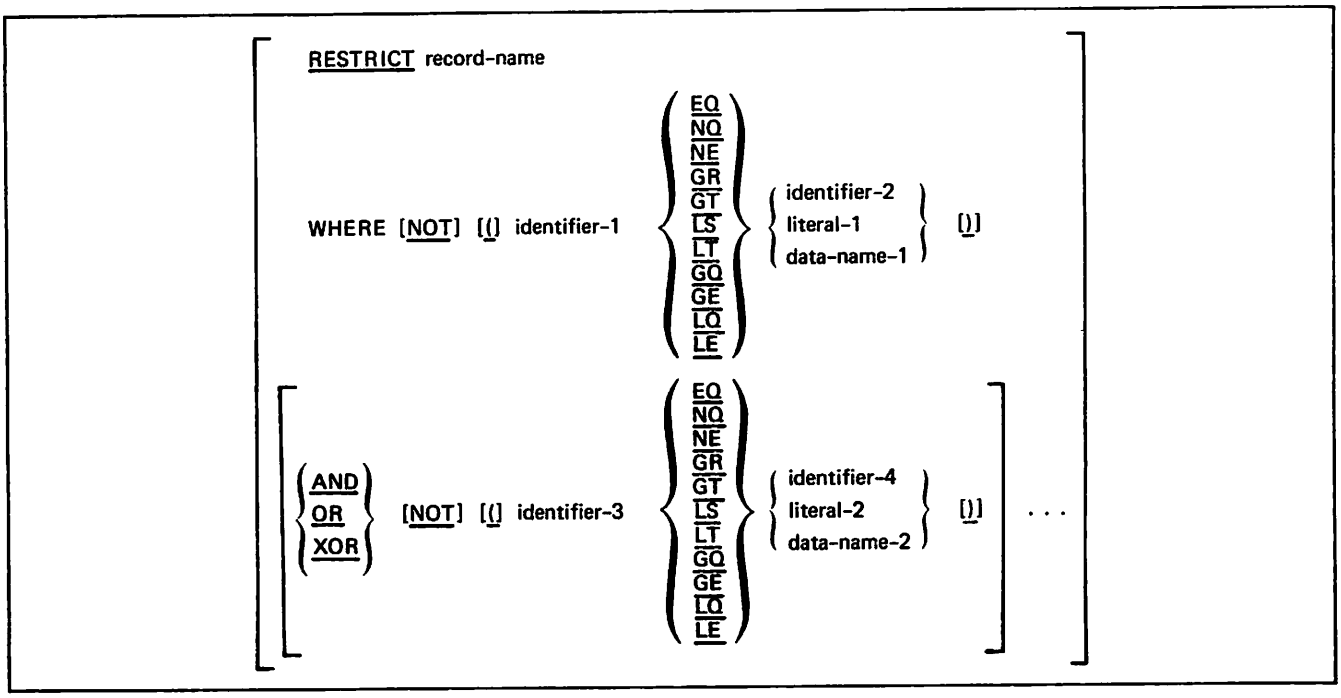


Figure 3-44. RESTRICT Clause Format

An identifier in one realm cannot be used as a qualifier identifier for records in another realm. All qualifier identifiers must be defined in the record type named in the RESTRICT clause. An identifier can be used as both a join term in the schema and a qualifier identifier in the subschema.

A qualifier identifier must be described in the subschema. It cannot be any of the following subschema data descriptions:

A data item that redescribes another item in a REDEFINES clause

A condition-name assigned to the values an item can assume in a level 88 data description entry (COBOL subschema only)

A data item that renames another data item in a RENAMES clause

In addition, the qualifier identifier must not have been defined in a VIRTUAL RESULT or a DECODING clause in the schema.

Identifiers to the left and to the right of the operator must be similar in type: both must be display coded data or both must be the same type of binary data. If two identifiers are different display coded data classes, they are compared as alphanumeric data. If two identifiers are binary data, both must be of the same data class. Refer to the Data Size and Class subsection for details regarding data class representation.

Qualifier identifiers must be elementary items. Only constant integers are allowed as item subscripts.

Literals specified in the RESTRICT clause must be compatible with the schema representation of the identifier corresponding to the literal. Table 3-5 lists valid identifier and literal combinations. Literals provide static qualification and cannot be changed from within a COBOL or Query Update program. In a COBOL subschema, literals cannot be figurative constants such as ZERO, SPACES, and so on.

TABLE 3-5. VALID IDENTIFIER AND LITERAL COMBINATIONS

Identifier Data Type	Literal Data Type				
	Integer	Fixed-Point	Floating-Point	Complex	Non-Numeric
Integer	X	X	X		
Fixed-Point	X	X	X		
Floating-Point	X	X	X		
Complex	X	X	X	X	
Non-Numeric					X

Data-name-1 in a COBOL subschema references a data item contained in the Data Division of the COBOL application program; such an item is initialized within the COBOL program. Data-name-1 in a Query Update subschema references a data item defined by the user within the Query Update session prior to the INVOKE (or USE) directive that specifies the subschema. In either subschema, data-name-1 must be unique among all names in the subschema. Use of a data-name in the RESTRICT clause allows for dynamic qualification of the relationship. The data-name specified to the right of the operator must have a data representation that is identical to the subschema representation of the identifier specified to the left of the operator.

SUBSCHEMA COMPILATION AND SUBSCHEMA LIBRARY MAINTENANCE

A COBOL or Query Update subschema source program is compiled by the DDL compiler. The compiler generates a subschema directory or object subschema from the subschema source program. A subschema directory is often simply called a subschema.

The DDL compiler is a multifunctional compiler. In addition to generating a subschema directory, the compiler stores the directory in a subschema library and performs maintenance operations on the subschema library. More than one subschema can be compiled with one control statement call to the DDL compiler. The particular operation to be performed by the DDL compiler is selected by a parameter in the DDL3 control statement.

The field length requirements for subschema compilation are indicated in appendix G.

SUBSCHEMA LIBRARY

One or more compiled subschemas are stored in a permanent file called the subschema library. The library is created when the first subschema is stored in it. Subsequent subschemas can be added to the library or can replace existing subschemas in the library. A subschema can be deleted from the library through use of the purge parameter in the DDL3 control statement. A subschema library that has had subschemas replaced or purged can be transferred to a new, compacted subschema library.

Data security can be maintained by creating more than one subschema library to control the availability of the subschemas to the application programs. Subschemas providing access to data that is restricted to specific applications can be stored in one subschema library; subschemas for general use can be stored in a different library. Each library is identified by a unique permanent file name. When a COBOL application program is compiled, a subschema library must be attached by specifying the permanent file name of the library. Subschemas stored in other libraries are not available to the COBOL program. When a Query Update INVOKE (or USE) directive names a subschema library, Query Update automatically attaches the subschema library file. Subschemas stored in other libraries are not available to the Query Update program. COBOL and Query Update subschemas can reside in the same library.

DDL3 CONTROL STATEMENT

The DDL3 control statement executes the DDL compiler. It provides the DDL compiler with information about a specific subschema, specifies the listings to be generated, and specifies the subschema library maintenance operations to be performed. The format of the DDL3 control statement is shown in figure 3-45. The comma immediately following DDL3 can be replaced by a left parenthesis; the terminating period can be replaced by a right parenthesis.

Either the C5 or QC parameter must be specified. All other parameters of the DDL3 control statement are optional.

The C5 and QC parameters select compilation for either a COBOL or Query Update subschema, respectively. The C5 and QC parameters are interpreted as follows:

C5

Specifies a COBOL subschema for use with a COBOL 5 application program.

QC

Specifies a Query Update subschema for use with a Query Update application program.

The SB parameter identifies the local file name of the subschema library file. This parameter is interpreted as follows:

omitted

Local file SBLFN is assumed to contain the subschema library.

SB=lfn

The local file name specified identifies the file that contains the subschema library.

```
DDL3 {,C5 / ,QC} [,SB=lfn] [ ,A / ,N / ,NL=lfn / ,P / ,R ] [,SC=lfn][,I=lfn][,L=lfn][,LO=op].
```

Figure 3-45. DDL3 Control Statement Format for COBOL and Query Update Subschemas

The A, N, NL=lfn, P and R parameters along with a default operation select subschema library maintenance operations. Only one of the operations can be performed in a single execution of the DDL compiler. If two or more of these parameters are specified, a control statement error is issued. If a library manipulation function is attempted on a nonempty file that does not contain a library, a diagnostic is issued and the job is aborted. The parameters are interpreted as follows:

omitted

Each subschema source program in the job stream is compiled and added to the subschema library identified by the SB parameter.

A

Audit parameter. A list of the subschemas and their corresponding schemas, together with their creation dates, is produced from the subschema library identified by the SB parameter.

N

Compile parameter. Each subschema source program in the job stream is compiled but not added to the subschema library identified by the SB parameter.

NL=lfn

New library parameter. The specified local file name identifies a new subschema library to which the active subschemas in the subschema library (identified by the SB parameter) are transfered. If the NL parameter is specified with no file name indicated, local file name NEWLIB is assumed.

P

Purge parameter. A subschema specified in the job stream is purged from the subschema library identified by the SB parameter. No compilation takes place.

R

Replace parameter. Each subschema source program in the job stream is compiled and replaces the existing subschema (identified by the Title Division) in the subschema library. The subschema library file is identified by the SB parameter. Replacement takes place only if no compilation errors other than informative diagnostics are encountered.

The SC parameter identifies the local file that contains the schema directory. A specified lfn overrides the default file name determined from the Title Division. This parameter is interpreted as follows:

omitted

The first seven characters of the schema name specified in Title Division of the subschema source program is assumed to identify the local file that contains the schema directory.

SC=lfn

The specified local file name identifies the file that contains the schema directory.

The I parameter identifies the local file that contains source input for the DDL compiler. This parameter is interpreted as follows:

omitted

The local file INPUT is assumed to contain the source input for the DDL compiler.

I=lfn

The specified local file name identifies the file that contains source input for the DDL compiler.

The L parameter identifies the local file that receives listings and diagnostics generated by the DDL compiler. This parameter is interpreted as follows:

omitted

The local file OUTPUT receives the listings and diagnostics generated by the DDL compiler.

L=0

The local file OUTPUT receives only the diagnostics generated by the DDL compiler.

L=lfn

The specified local file name identifies the file that receives the listings and diagnostics generated by the DDL compiler.

The LO parameter selects the listing produced by the DDL compiler. Two listing options (op) can be specified: a source listing (S option) and an object listing (O option). Source and object listings are written to the file specified by the L parameter. If L=0 is specified, no listing is produced except for error messages, regardless of LO specification. This parameter is interpreted as follows:

omitted

Same as LO=S; a source listing is produced.

LO

Same as LO=S; only a source listing is produced.

LO=op
LO=op_1/op_2

The specified listing option selects the listing produced. The listing options are as follows:

S Source listing

O Object listing; a listing of the code generated for a mapping capsule. If no mapping capsule is required for the subschema, there is no object listing.

SUBSCHEMA COMPILATION AND LIBRARY MAINTENANCE OPERATIONS

A COBOL or Query Update subschema must be coded according to the specifications in this manual. Subschema source code can be entered through a terminal, processed by a text editor, and stored in a file.

The DDL compiler can be executed from a terminal and through a batch job in the following ways. When executed from a terminal, the subschema program must reside on a file whose name is indicated by the I parameter of the DDL3 control statement. In a batch job, in which the input file for the DDL compiler is assumed to be local file INPUT, the job stream must be structured so that the control statements precede the subschema source program. An end-of-record indicator must immediately precede the first line of subschema source code to separate the subschema program from the control statements.

Control statements in the job stream and the resulting subschema source program are used by the DDL compiler either to compile the subschema and to store it in the subschema library or to perform a subschema library maintenance operation such as replacing or deleting subschemas, generating a new, compacted subschema library, or producing an audit listing. The parameters in the DDL3 control statement select the operation to be performed. Other control statements provide information for the operating system to make necessary information available for the DDL compiler.

The following paragraphs indicate the parameters required in the DDL3 control statement to accomplish compilation of COBOL and Query Update subschemas and to perform subschema library maintenance operations. Examples of batch jobs that accomplish the operations are shown. The DDL3 control statements used in the examples alternately indicate Query Update and COBOL subschemas. Control statements are shown for both the NOS and NOS/BE operating systems.

Compiling a Subschema

The source program for a subschema can be compiled without being stored in the subschema library. The form of the DDL3 control statement (shown with the QC parameter) needed to compile a Query Update subschema is as follows:

```
DDL3,QC,SB=lfn,N,SC=lfn.
```

Either the C5 or QC parameter must be specified: C5 for compilation of a COBOL subschema; QC for compilation of a Query Update subschema. The compile parameter (N) must also be specified. The SB, SC, I, and L parameters are optional and have default values (refer to the DDL3 Control Statement subsection).

Figure 3-46 illustrates a job that accomplishes subschema compilation only. The schema file must be attached.

Creating a Subschema Library

The subschema library is created when the first subschema is stored in it. The form of the DDL3 control statement (shown with the C5 parameter) needed to create a subschema library and store a compiled subschema in it is as follows:

```
DDL3,C5,SB=lfn,SC=lfn.
```

Either the C5 or QC parameter must be specified: C5 for compilation of a COBOL subschema; QC for compilation of a Query Update subschema. The SB, SC, I, L, and LO parameters are optional and have default values (refer to the DDL3 Control Statement subsection).

Figure 3-47 illustrates a job that compiles a subschema and creates a subschema library. In this example, the schema file SCHPAY is attached. The subschema library is identified by the local file name SUBSCH. The DEFINE and REQUEST/CATALOG control statements specify the local file name of the subschema library and assign it to a permanent file device. A password specified in the DEFINE or CATALOG statement controls subsequent access and use of the subschema library.

If only one subschema is to be stored in the library, the subschema name can be used for the subschema library file name.

Query Update-CDCS subschemas (QC option) and COBOL subschemas (C5 option) are not compatible with Query Update-CRM subschemas (QD option) and therefore cannot exist on the same library file.

NOS Operating System

```
Job statement
USER control statement
CHARGE control statement
ATTACH,SCHPAY.
DDL3,C5,SB=SBTST,N,SC=SCHPAY.
End-of-record
   DDL Subschema Source Input
End-of-information
```

NOS/BE Operating System

```
Job statement
ACCOUNT control statement
ATTACH,SCHPAY,ID=DDL.
DDL3,QC,SB=SBTST,N,SC=SCHPAY.
End-of-record
   DDL Subschema Source Input
End-of-information
```

Figure 3-46. Compiling a Subschema

NOS Operating System

```
Job statement
USER control statement
CHARGE control statement
ATTACH,SCHPAY.
DEFINE,SUBSCH/PW=DDL,CT=PU,M=W.
DDL3,QC,SB=SUBSCH,SC=SCHPAY.
End-of-record
   DDL Subschema Source Input
End-of-information
```

NOS/BE Operating System

```
Job statement
ACCOUNT control statement
ATTACH,SCHPAY,ID=DDL.
REQUEST,SUBSCH,PF.
DDL3,C5,SB=SUBSCH,SC=SCHPAY.
CATALOG,SUBSCH,ID=DDL,MD=DDL,EX=DDL,CN=DDLX.
End-of-record
   DDL Subschema Source Input
End-of-information
```

Figure 3-47. Creating a Subschema Library

Compiling Multiple Subschemas

A number of subschemas can be compiled with one control statement call to the DDL compiler. The parameters specified in the control statement apply to all the subschemas. The subschema library maintenance operations of creating a subschema library, adding (default operation) subschemas to the subschema library, replacing (R parameter) subschemas in the library, or just compiling (N parameter) subschemas can be used with this facility.

For the compilation of several subschemas with one control statement call to the DDL compiler, the subschema source statements must be contiguous, with no intervening end-of-record indicator or end-of-information indicator. The program source for each subschema must begin with the TITLE DIVISION statement. Each subsequent TITLE DIVISION statement immediately follows the last source statement of the preceding subschema. Each subschema is compiled in turn until end-of-information or end-of-record is encountered.

Compiling a Subschema and Adding to a Subschema Library

Once the subschema library has been created and saved on a permanent file (as described in the Creating a Subschema Library subsection), new subschemas can be added to the library. The form of the DDL3 control statement (shown with the QC parameter) needed to add a subschema to an existing subschema library is as follows:

```
DDL3,QC,SB=lfn,SC=lfn.
```

Either the C5 or QC parameter must be specified: C5 for compilation of a COBOL subschema; QC for compilation of a Query Update subschema. The SB, SC, I, L, and LO parameters are optional and have default values (refer to the DDL3 Control Statement subsection).

Figure 3-48 illustrates a job that adds a subschema to the subschema library created by the example in figure 3-47. The schema directory is attached. The subschema library is also attached with the applicable password and (for NOS only) access mode M=W specified.

Each subschema stored in the library must have a unique name. If the subschema being added to the library has the same name as a subschema already stored in the library, a diagnostic is issued and the job is aborted.

Replacing a Subschema

A new subschema can replace one that is stored in the subschema library. The form of the DDL3 control statement (shown with the C5 parameter) needed to replace a subschema in the subschema library is as follows:

```
DDL3,C5,SB=lfn,R,SC=lfn.
```

Either the C5 or QC parameter must be specified: C5 for compilation of a COBOL subschema; QC for compilation of a Query Update subschema. The replacement parameter (R) must be specified. The SB, SC, I, L, and LO parameters are optional and have default values (refer to the DDL3 Control Statement subsection).

NOS Operating System

```
Job statement
USER control statement
CHARGE control statement
ATTACH,SCHPAY.
ATTACH,SUBSCH/PW=DDL,M=W.
DDL3,C5,SB=SUBSCH.
End-of-record
   DDL Subschema Source Input
End-of-information
```

NOS/BE Operating System

```
Job statement
ACCOUNT control statement
ATTACH,SCHPAY,ID=DDL.
ATTACH,SUBSCH,ID=DDL,PW=DDL.
DDL3,QC,SB=SUBSCH.
End-of-record
   DDL Subschema Source Input
End-of-information
```

Figure 3-48. Compiling a Subschema and Adding to a Subschema Library

Figure 3-49 illustrates a job that replaces a subschema in the subschema library. The schema directory is attached. The subschema library file is also attached with the applicable password and (for NOS only) access mode M=W is specified. If the subschema to be replaced cannot be found in the subschema library, an informative diagnostic is issued and the new subschema is added to the library.

Deleting a Subschema

A subschema stored in the subschema library can be deleted from the library. The form of the DDL3 control statement (shown with the QC parameter) needed to purge a subschema from the subschema library is as follows:

```
DDL3,QC,SB=lfn,P.
```

Either the C5 or QC parameter must be specified. The purge parameter (P) must be specified. The SB, I, and L parameters are optional and have default values (refer to the DDL3 Control Statement subsection).

Figure 3-50 illustrates a job that purges a subschema from the subschema library. The subschema library file is attached with the applicable password and (for NOS only) access mode M=W is specified. The end-of-record indicator designates the end of the control statements.

The control statements are followed by statements that specify the subschemas to be deleted. The subschema name is entered anywhere from column 8 through column 72. If more than one subschema name is entered, a space or a comma must follow each subschema name.

No compilation occurs when a subschema is deleted from the subschema library.

Auditing a Subschema Library

After a subschema library is created, it can be searched and a list generated that includes the name of each subschema and the name of the schema it references, together with their creation dates. The form of the DDL3 control statement (shown with the C5 parameter) needed to generate the audit listing is as follows:

```
DDL3,C5,SB=lfn,A.
```

Either the C5 or QC parameter must be specified. The audit parameter (A) must be specified. The SB and L paramaters are optional and have default values (refer to the DDL3 Control Statement subsection).

Figure 3-51 illustrates a job that produces an audit listing of the subschema library. The subschema library is attached with the appropriate password specified. The listing produced by the DDL compiler when the audit is performed is shown in figure 3-52.

NOS Operating System

```
Job statement
USER control statement
CHARGE control statement
ATTACH,SCHPAY.
ATTACH,SUBSCH/PW=DDL,M=W.
DDL3,QC,SB=SUBSCH,R.
End-of-record
   DDL Subschema Source Input
End-of-information
```

NOS/BE Operating System

```
Job statement
ACCOUNT control statement
ATTACH,SCHPAY,ID=DDL.
ATTACH,SUBSCH,ID=DDL,PW=DDL.
DDL3,C5,SB=SUBSCH,R.
End-of-record
   DDL Subschema Source Input
End-of-information
```

Figure 3-49. Replacing a Subschema Library

NOS Operating System

```
Job statement
USER control statement
CHARGE control statement
ATTACH,SUBSCH/PW=DDL,M=W.
DDL3,C5,SB=SUBSCH,P.
End-of-record
        PAYROLL
        NEWHIRES
End-of-information
```

NOS/BE Operating System

```
Job statement
ACCOUNT control statement
ATTACH,SUBSCH,ID=DDL,PW=DDL.
DDL3,QC,SB=SUBSCH,P.
End-of-record
        PAYROLL
        NEWHIRES
End-of-information
```

Figure 3-50. Deleting Subschemas From the Library

NOS Operating System

```
Job statement
USER control statement
CHARGE control statement
ATTACH,SUBSCH/PW=DDL.
DDL3,C5,SB=SUBSCH,A.
End-of-information
```

NOS/BE Operating System

```
Job statement
ACCOUNT control statement
ATTACH,SUBSCH,ID=DDL,PW=DDL.
DDL3,QC,SB=SUBSCH,A.
End-of-information
```

Figure 3-51. Auditing a Subschema Library

```
                        * SOURCE LISTING *   (82061)  DDL  3.2+564.      82/04/05. 09.56.11.


      -----       BEGIN SUB-SCHEMA FILE MAINTENANCE              -----

    LIST OF SUB-SCHEMAS IN FILE

    SUB-SCHEMA                      CREATION        ?    SCHEMA                      CREATION
                                   DATE    TIME                                    DATE    TIME
  ----------                     ----------------   ?  --------                   ----------------
    C5SS-PRODUCT-PERSONNEL        82087   14.19          MANUFACTURING-DB         82087   14.14
    PAYROLL                       82087   14.19          MANUFACTURING-DB         82087   14.14
    PRODUCTS                      82087   14.19          MANUFACTURING-DB         82087   14.14
    PROJECTS                      82087   14.19          MANUFACTURING-DB         82087   14.14
    C5SS-PRODUCT-MANAGEMENT       82087   14.19          MANUFACTURING-DB         82087   14.14

      -----       END OF FILE MAINTENANCE              -----
DDL COMPLETE.          0 DIAGNOSTICS.
  45300B CM USED.     0.022  CP SECS.
```

Figure 3-52. Audit Listing of the Subschema Library

Compacting a Subschema Library

Subschema library compaction is an optional facility that generates a new subschema library by copying only the active subschemas from the specified subschema library. When this facility is executed, the DDL compiler generates a listing that includes the name of each transferred subschema and the schema it references, together with their creation dates. The form of the DDL3 control statement (shown with the C5 parameter) needed to create a new, compacted subschema library is as follows:

```
DDL3,C5,SB=lfn,NL=lfn.
```

Either the C5 or QC parameter must be specified. The new library parameter (NL) must be specified. The SB and L parameters are optional and have default values (refer to the DDL3 Control Statement subsection).

Figure 3-53 illustrates a job that creates a new, compacted subschema library. The subschema library is attached with the appropriate password. The DEFINE and REQUEST/CATALOG control statements specify the local file name of the new subschema library and assign it to a permanent file device. A password specified in the DEFINE or CATALOG statement controls subsequent access and use of the subschema library. The DDL3 control statement specifies the SB parameter with the local file name of the current subschema library to be compacted, and the NL parameter with the local file name of the new subschema library. The PURGE control statement destroys the old library file.

NOS Operating System

```
Job statement
USER control statement
CHARGE control statement
ATTACH,SUBSCH/PW=DDL, M=W.
DEFINE,NEWSUB/PW=DDL,CT=PU,M=W.
DDL3,QC,SB=SUBSCH,NL=NEWSUB.
PURGE,SUBSCH.
End-of-information
```

NOS/BE Operating System

```
Job statement
ACCOUNT control statement
ATTACH,SUBSCH,ID=DDL,PW=DDL.
REQUEST,NEWSUB,PF.
DDL3,C5,SB=SUBSCH,NL=NEWSUB.
CATALOG,NEWSUB,ID=DDL,MD=DDL,EX=DDL,CN=DDLX.
PURGE,SUBSCH,RB=1.
End-of-information
```

Figure 3-53. Compacting a Subschema Library

This facility is intended to be used on a subschema library that has had a number of subschemas purged or replaced and, therefore, contains wasted space. The DDL compiler eliminates the wasted space in the new subschema library. After the new subschema library is stored as a permanent file, the user should purge the old subschema library. This facility does not allow for compilation of subschemas.

COMPILATION OUTPUT

A listing of the DDL source program is provided whenever a COBOL or Query Update subschema is compiled. Each line of the listing corresponds to one source line in the source program. The format and order of each line of the listing are identical to the format and order of the statements in the source program. Figure 3-54 is a sample source listing for a subschema compilation.

The DDL compiler assigns a line number to each input statement, beginning with 00001. The line numbers are printed on the source listing, starting in column 16. Diagnostic messages begin in column 3 of the listing. After the last input statement is listed, a compilation summary is printed. When relation statistics are applicable, relation names and their traversed areas are included.

The source listing can be suppressed by specifying L=0 in the DDL3 control statement. Only diagnostic messages and the compilation summary are printed on the listing.

When a subschema is compiled, a cross-reference list and a data map are not printed. When a COBOL application program specifies the subschema, the COBOL compiler produces a cross-reference list and a data map containing the realm, record, and data item entries defined in the subschema.

RECOMPILATION GUIDELINES

The DDL compiler generates a checksum for each area and relation in the schema. These checksums are the means for determining the need to recompile a subschema. If a checksum in a recompiled schema is different from the corresponding checksum in the previous schema, any subschema referencing the changed area or relation must be recompiled. A list of subschemas that must be recompiled because of a schema change involving an area or relation can be obtained from a schema compilation. Refer to the Recompilation Guidelines subsection in section 2 for more information. Subschemas not referencing changed areas or relations need not be recompiled.

The DDL compiler generates a checksum for each subschema. This checksum is the means for determining the need to recompile application programs. When an application program is compiled, the checksum of the subschema it references is copied into the program binary output. When the application program is run, that checksum must be the same as a checksum of a subschema in the master directory. If the application program references an invalid checksum, CDCS aborts the program and issues a diagnostic.

```
                        * SOURCE LISTING *   (82061)  DDL  3.2+564.

00001                TITLE DIVISION.
00002                    SS  C5SS-PRODUCT-PERSONNEL WITHIN MANUFACTURING-DB.
00003
00004                ALIAS DIVISION.
00005                    AD  REALM JOBDETAIL BECOMES WORK-FILE.
00006                    AD  RECORD JOBREC BECOMES WORK-REC.
00007                    AD  RECORD EMPREC BECOMES EMP-REC.
00008                    AD  DATA LOC-CODE BECOMES LOCATION.
00009
00010                REALM DIVISION.
00011                    RD EMPLOYEE, WORK-FILE.
00012
00013                RECORD DIVISION.
00014                01  EMP-REC.                                    ** WITHIN EMPLOYE
00015                    03  EMP-ID                PICTURE X(8).     ** ORDINAL    1
00016                    03  SALARY                PICTURE 9(6)V99   ** ORDINAL    2
00017                                                   USAGE IS COMP-1.
00018                    03  EMP-LAST-NAME         PICTURE A(20).    ** ORDINAL    3
00019                    03  EMP-INITIALS          PICTURE A(4).     ** ORDINAL    4
00020                    03  DEPT                  PICTURE X(4).     ** ORDINAL    5
00021                    03  MAILING-ADDRESS.                        ** ORDINAL    6
00022                        05  ADDRESS-NUMBERS   PICTURE X(6).     ** ORDINAL    7
                             .
                             .
                             .
```

Figure 3-54. Sample COBOL Subschema Compilation Output Listing (Sheet 1 of 2)

```
             00035                01  WORK-REC.                                                  ** WITHIN WORK-FI
             00036                    03  CONCATKEY.                                             ** ORDINAL    1
             00037                        05  EMP-ID              PICTURE X(8).                  ** ORDINAL    2
             00038                        05  SEQ-NO              PICTURE X(4).                  ** ORDINAL    3
             00039                    03  PRODUCT-ID              PICTURE X(10).                 ** ORDINAL    4
             00040                    03  SECURITY-CODE           PICTURE X(2).                  ** ORDINAL    5
             00041                    03  MONTHLY-COMPENSATION    USAGE IS COMP-1                ** ORDINAL    6
             00042                                                    OCCURS 12 TIMES.
             00043                        05  REG-HOURS           PICTURE 9(3)V99.               ** ORDINAL    7
             00044                        05  REG-COMPENSATION    PICTURE 9(4)V99.               ** ORDINAL    8
             00045                        05  OT-HOURS            PICTURE 9(3)V99.               ** ORDINAL    9
             00046                        05  OT-COMPENSATION     PICTURE 9(4)V99.               ** ORDINAL   10
             00047                    03  HOURS-YTD               PICTURE 9(4)V99                ** ORDINAL   11
             00048                                                    USAGE IS COMP-1.
             00049                    03  COMPENSATION-YTD        PICTURE 9(8)V99                ** ORDINAL   12
             00050                                                    USAGE IS COMP-1.
             00051                    03  LOCATION                PICTURE X(4).                  ** ORDINAL   13
             00052
 ***314*     00041       WARNING: SS SIZE GR SCHEMA SIZE - MAY CAUSE TRUNCATION ERRORS AT EXECUTION TIME
PRIMARY KEY  00015       EMP-ID FOR AREA EMPLOYEE
ALTERNATE KEY 00020      DEPT FOR AREA EMPLOYEE
PRIMARY KEY  00036       CONCATKEY FOR AREA WORK-FILE
             *****       RECORD MAPPING IS NEEDED FOR REALM -   EMPLOYEE
             *****       RECORD MAPPING IS NEEDED FOR REALM -   WORK-FILE
             00053               RELATION DIVISION.
             00054                   RN IS EMP-REL
             00055                       RESTRICT WORK-REC WHERE SECURITY-CODE GE "B1".
             00056
             00057
             *****       END OF SUB-SCHEMA SOURCE INPUT

             *****               RELATION   STATISTICS                *****
RELATION 001                 EMP-REL JOINS            AREA - WORK-FILE
                                                      AREA - EMPLOYEE


             -----       BEGIN SUB-SCHEMA FILE MAINTENANCE             -----

        SUBSCHEMA                                CHECKSUM
        C5SS-PRODUCT-PERSONNEL                   07267725304252560742

             -----       END OF FILE MAINTENANCE              -----
DDL COMPLETE.          1 DIAGNOSTICS.
  50200B CM USED.      0.330  CP SECS.
```

Figure 3-54. Sample COBOL Subschema Compilation Output Listing (Sheet 2 of 2)

A FORTRAN subschema describes the portion of a data base to be used by one or more FORTRAN applications programs. Its descriptions link the descriptions found in the schema with the variables and arrays in the FORTRAN program. The subschema uses statements similar to FORTRAN specification statements to indicate the data type and dimensions of variables and arrays used in the applications program.

A FORTRAN subschema is coded and compiled for use by an application program of the FORTRAN 5 language version. The output from compilation of the subschema source program is a subschema directory or object subschema.

When a FORTRAN program containing Data Manipulation Language (DML) statements is processed, the file containing the subschema directory must be made available to the DML preprocessor. The DML preprocessor inserts FORTRAN statements derived from subschema definitions into the FORTRAN program.

The subschema is based on the schema; the schema must be compiled before the subschema. The subschema can include all or part of the entities defined in the schema. Any number of subschemas can use a given schema. In general, the schema description of data can be changed in the subschema to meet the needs of the applications program; however, some limitiations have been imposed. These are outlined in the following paragraphs.

SUBSCHEMA STRUCTURING REQUIREMENTS

FORTRAN subschemas consist of required and optional statements. The statements and their functions are as follows:

SUBSCHEMA statement specifies the name of the subschema and identifies the schema.

ALIAS statement assigns alternate names to be used in place of names assigned in the schema.

REALM statement identifies the schema areas to be made available to an application program through the subschema.

RECORD statement and subsequent type statements describe the structure and content of each record type in the subschema.

RELATION statement identifies a relation defined in the schema that is included in the subschema.

RESTRICT statement specifies the qualification criteria that must be satisfied by records that are returned through relation processing.

END statement indicates the end of the subschema source program.

The subschema source program must conform to structure requirements. Table 4-1 shows the order in which groups of FORTRAN Data Description Language (DDLF) statements must be included in a subschema. In group 4, the type statements that apply to the variables and arrays belonging to a record defined by a RECORD statement appear immediately after the RECORD statement. In group 5, the RESTRICT statements that apply to a relation defined by the RELATION statement appear immediately after the RELATION statement. A FORTRAN subschema must contain at least one realm, one record, and one type statement.

TABLE 4-1. ORDERING OF SUBSCHEMA STATEMENTS

Group	Statement
1	SUBSCHEMA
2	ALIAS (optional)
3	REALM
4	RECORD and type
5	RELATION (optional) RESTRICT (optional)
6	END

DATA DESCRIPTION

A record description entry is composed of a RECORD statement and the type statements immediately following it. The type statements following each RECORD statement specify the data items that are to be made available from the corresponding schema record. The ordering of type statements is independent of the ordering of items within the schema record.

Correspondence between schema and subschema items is based on the item name. Therefore, a data name in a type statement must be one of the following:

A data name from the schema description

An alias assigned in an ALIAS statement

Any data name in the schema longer than seven characters, or containing a hyphen, must be renamed in an ALIAS statement, because these data names are not allowed in FORTRAN. Once an alias is assigned to a schema name, the alias must be used in all FORTRAN DDL statements. Only elementary item names from the schema can be defined in a FORTRAN subschema. Each record type corresponds to a record type in the schema and must be within one of the realms specified in REALM statements. Each RECORD statement must be followed by at least one type statement.

All the data items defined in the subschema are included in the FORTRAN program by the DML preprocessor. Therefore, all data items named in the subschema are either variables or arrays in the FORTRAN applications program.

Character variables and arrays should be grouped together within a record in the subschema to minimize the number of common blocks that the DML preprocessor generates for a FORTRAN program. The variables and arrays are declared in common blocks in the same order as they are included in the subschema. One common block is generated for each realm. The name for that common block is in the form DBnnnn, where nnnn is the realm ordinal assigned by the DDLF compiler. Realm ordinals are assigned incrementally starting with 1 for each realm named in the subschema. Character data items cannot share the same common block with noncharacter items; therefore, a new common block is generated each time a type statement is encountered that is not compatible with the previous type statement. The name for each additional common block required for a realm is in the form Dnnnnxx, where nnnn is the realm ordinal and xx is a 2-letter identifier assigned incrementally from the series AA, AB, ..., ZZ.

VARIABLES

A variable declaration in a type statement associates a symbolic name of the specified type with a single data item. Because every data base data item that is to be referenced in the FORTRAN program must be declared in the subschema, implicit typing of variables is overridden. A variable defined in the subschema must correspond to a nonrepeating elementary item in the schema.

ARRAYS

The size, dimensions, and type of an array are defined in a type statement. Array declarations are identical in form and content to those in a FORTRAN program.

An array in a subschema corresponds to a repeating elementary item in the schema; that is, to an item containing an OCCURS clause. A repeating elementary item is called a vector. An array can correspond to either a fixed occurrence repeating item or a variable occurrence repeating item.

SCHEMA/SUBSCHEMA CORRESPONDENCE

The subschema is created to accommodate the needs of a FORTRAN applications program. Some characteristics of the data in the data base are fixed by the schema and cannot be changed by the subschema; other characteristics specified in the schema can be different in the subschema. The following paragraphs outline the cases in which differences are allowed between the schema and the subschema, and the actions taken by the DDLF compiler to resolve the differences in each case.

OMISSION OF DATA ITEMS

The subschema normally describes only a portion of the data base. Data items that are not required by the FORTRAN program are not included in the subschema description. Elementary items, complete records, and entire areas in the schema can be omitted from the subschema. When a record or an area is not included in the subschema description, all subordinate entries are automatically omitted and cannot be referenced in the subschema.

Unlike the COBOL subschema, the FORTRAN subschema has no mechanism for the description of group items. Therefore, these items must be omitted from the subschema. The elementary items making up the schema group item must also be omitted from the subschema.

Unlike the COBOL subschema, only one record type per realm is permitted in the FORTRAN subschema.

The primary key for a realm must be declared in the subschema; alternate keys are required only if they are actually used.

ORDERING OF DATA ITEMS

The order in which data items are specified in a record description need not match the order in the schema, unless the data items are constituent items of a concatenated key. The names of the items, however, must match those in the schema or in the ALIAS statement.

Data items of a concatenated key (primary or alternate) must be in the same order as in the schema and must also be contiguous items.

DEFINITION OF DATA ITEMS

Data items can differ in size, type, and number of array elements from those in the schema.

Data Size and Type

The size and type of data items in the subschema are specified in the type statements. In the schema they are specified in either the TYPE or PICTURE clause. Since the schema and subschema statements differ in format, rules have been established for conversion between subschema and schema specifications. In some cases, the types specified in the schema and subschema match exactly; no conversion is required. In other cases, the types differ, but a meaningful conversion is established by DDLF and carried out through mapping at execution time by CDCS. In still other cases, no conversion is possible and an error message is issued by the DDLF compiler.

Variables and Arrays

Complex and double precision variables occupy two words of storage each; all other variables occupy one word of storage. Complex and double precision arrays occupy two words of storage for each array element; all other arrays occupy one word of storage for each array element.

Conversion

Table 4-2 shows the allowed correspondence between types of items in the subschema and data class specifications in the schema. (Refer to section 2 for complete descriptions of these data classes.) For those cases where conversion is necessary, the table describes the method used.

If the schema specifies a CHECK IS PICTURE clause, the data description in the subschema must match the data description in the schema for both size and class. The CHECK IS PICTURE clause in the schema definition inhibits data conversion between the schema and the subschema. Table 4-2 indicates the data type required for a subschema item to correspond to a schema item defined with the CHECK IS PICTURE clause.

Concatenated Keys

Items in a concatenated key must conform to the schema description of the items in size and type. The size of concatenated key items in the subschema must be identical to the size of concatenated key items in the schema. The type of the concatenated key items must match the type of concatenated key items in the schema; key mapping is not allowed. All members of a concatenated key must be elementary items; they cannot be an element of a vector or an array. A maximum of 64 contiguous items is allowed for a concatenated key.

Because no key mapping is allowed, only the data types requiring no conversion or type checking from schema to subschema are accepted for concatenated key items. Table 4-3 shows the allowable schema item types and corresponding subschema item types for which no conversion or type checking is performed. The table entries correspond to the footnoted items in table 4-2 designating required subschema data types when CHECK IS PICTURE is used in the schema. Refer to the discussion of data size and class in section 2 for a description of these schema classes.

The effect of the CHECK IS PICTURE clause is to disallow conversion and type checking for the item. Therefore, it might be helpful for the data base administrator to have CHECK IS PICTURE specified in the schema description for all data items used in concatenated keys. The specification, however, inhibits conversion for COBOL and Query Update subschemas and might not be practical.

TABLE 4-2. SCHEMA/SUBSCHEMA MAPPING

Schema Class (type)	Subschema Type						
	INTEGER FORTRAN 5	CHARACTER FORTRAN 5	BOOLEAN FORTRAN 5	LOGICAL FORTRAN 5	REAL FORTRAN 5	DOUBLE PRECISION FORTRAN 5	COMPLEX FORTRAN 5
0 Display alpha-numeric	Not permitted.	No conversion required.†	Not permitted.	Not permitted.	Not permitted.	Not permitted.	Not permitted.
1 Display alphabetic	Not permitted.	No conversion required. Error if not alphabetic.	Not permitted.	Not permitted.	Not permitted.	Not permitted.	Not permitted.
3 Display integer	Evaluate character string representing integer and convert to binary.	No conversion required. Error if not numeric.	Evaluate character string representing integer and convert to binary.	Not permitted.	Evaluate character string representing value and float.	Same as for real; set least significant word to zero.	Not permitted.
4 Display fixed point	Evaluate character string representing integer and convert to binary.	Not permitted.	Evaluate character string representing integer and convert to binary.	Not permitted.	Evaluate character string representing value and float.	Same as for real; set least significant word to zero.	Not permitted.
10 Coded fixed point	No conversion required.†	Not permitted.	No conversion required.	No conversion required.	Float.	Float; set least significant word to zero.	Float; set imaginary part to zero.
13 Coded floating point normalized	Truncate real value.	Not permitted.	No conversion required.	Not permitted.	No conversion required.†	Set least significant word to zero.	Set imaginary part to zero.
14 Coded double precision	Drop least significant word and truncate value of most significant word.	Not permitted.	Drop least significant word.	Not permitted.	Drop least significant word.	No conversion required.†	Drop least significant word; set imaginary part to zero.
15 Coded complex	Drop imaginary part and truncate value of real part.	Not permitted.	Drop imaginary part.	Not permitted.	Drop imaginary part.	Drop imaginary part; set least significant word to zero.	No conversion required.

†Data type heading this column is required for a subschema item when the CHECK IS PICTURE clause is specified for the corresponding schema item.

TABLE 4-3. DATA TYPES FOR CONCATENATED KEYS

Schema Class (Type)	Subschema Type
	FORTRAN 5
Display Alphanumeric (Type Character; Picture X)	Character
Coded Fixed Point (Type Fixed)	Integer
Coded Floating Point Normalized (Type Float)	Real
Coded Double Precision (Type Float)	Double Precision

Array Declaration

Arrays are declared in the subschema by type statements in the same format as those in a FORTRAN program. The number of elements in an array is the product of all its dimensions.

An array is defined in the schema as a vector, or a repeating elementary item. Elementary data items can be repeated either a fixed or varying number of times. Arrays in the subschema correspond with these repeating data items.

A vector in the schema can correspond to a subschema array with up to seven dimensions for a FORTRAN 5 subschema. The number of elements in an array must be less than or equal to the number of occurrences of the schema vector. A repeating elementary item is called a fixed occurrence repeating item if the OCCURS clause which describes it in the schema specifies the precise number of occurrences of the item.

Figure 4-1 shows the schema and subschema declarations for two arrays. Both are fixed occurrence elementary items in the schema.

```
Schema

   RESULTS     TYPE FLOAT        OCCURS 40 TIMES.
   TESTNUM     PICTURE "9(10)"   OCCURS 20 TIMES.

Subschema

   REAL RESULTS(40)
   INTEGER TESTNUM(20)
```

Figure 4-1. Fixed Occurrence Elementary Items

The number of elements of the array declared in the subschema can fall short of (but cannot exceed) the number of occurrences of the item declared in the schema. When the number falls short, the initial elements of the array are matched to the initial occurrences of the item.

For example, in figure 4-2, the arrays BAYNAME and QUANT have the same number of elements as their counterparts in the schema, even though they are defined as 2-dimensional arrays in the subschema. The array TOP10 has fewer elements than its counterpart in the schema.

```
Schema

   BAYNAME     PICTURE "A(10)"   OCCURS 16 TIMES.
   QUANT       PICTURE "999"     OCCURS 2000 TIMES.
   TOP10       TYPE FIXED        OCCURS 100 TIMES.

FORTRAN 5 Subschema

   CHARACTER*10 BAYNAME(2,8)
   INTEGER QUANT(200,10), TOP10(10)
```

Figure 4-2. Schema/Subschema Differences in Array Size and Dimension

A repeating elementary item in the schema is called a variable occurrence repeating item if a data name is used in the OCCURS clause. In the schema, the occurrences of a variable occurrence repeating item (which correspond to the elements of the array in the subschema) are indexed by the elementary item referenced in the OCCURS clause. In a FORTRAN subschema, this item must be declared whenever the repeating item is declared. In the DML statements in the applications program, the variable is used to specify the number of occurrences of a repeating item when a record is written. When a record is read, CDCS sets the variable to the number of occurrences actually in the record.

If the subschema defines an item that is used in the schema to index the occurrences of a repeating item, the subschema must also define the elementary repeating item.

A subschema array that corresponds to a variable occurrence repeating item must be defined in the subschema to have the maximum number of elements possible (the upper limit of the CHECK IS VALUE clause).

In figure 4-3, the number of occurrences of SYMBOL can vary from 1 to 12, depending upon the value of ECOUNT. SYMBOL is a variable occurrence repeating elementary item in the schema. SYMBOL is declared as an array in the subschema with dimensions equal to the maximum value specified in the OCCURS clause. ECOUNT must also be declared in the subschema.

```
Schema

   ECOUNT      TYPE FIXED
                   CHECK VALUE 1 THRU 12.
   SYMBOL      PICTURE "A(10)
                   OCCURS ECOUNT TIMES.

FORTRAN 5 Subschema

   INTEGER ECOUNT
   CHARACTER*10 SYMBOL(12)
```

Figure 4-3. Variable Arrays in Schema and Subschema

Concatenated Key

A concatenated key is a primary or alternate record key that is defined in the schema and is composed of a series of contiguous elementary data items. For a FORTRAN program to use this record key, all the items in the concatenated key must be defined in the subschema. For a primary concatenated key, all the constituent items must be defined in the subschema. If an alternate concatenated key is used, all the constituent items must also be defined in the subschema.

Figure 4-4 gives an example of how a concatenated key is declared. By including the items in the concatenated key in the subschema, the key is automatically available for a program using the subschema. Because FORTRAN does not accept data names which have more than seven characters or which have hyphenation, the ALIAS statement is used in the subschema to assign alternate names in place of the names assigned in the schema. The concatenated key name is not used in FORTRAN statements and does not need to conform to FORTRAN syntax. In a FORTRAN program, the concatenated key name can only be specified in the FORTRAN DML READ and START statements.

SUBSCHEMA PROGRAMMING CONVENTIONS

The FORTRAN DDL subschema source program is composed of a series of statements that describe a portion of a data base. The rules for coding DDLF statements are similar to those for the version of FORTRAN specified in the DDLF control statement. The statements are described in the FORTRAN Subschema Syntax subsection that appears later in this section. The following paragraphs describe the format of the statements.

LANGUAGE ELEMENTS

FORTRAN DDL statements consist of keywords, user-defined names, constants, and operators. The operators are fully explained with the statements in the FORTRAN Subschema Syntax subsection; the remainder of the elements are described here.

Keywords

FORTRAN DDL keywords identify statements and options within statements. Each statement begins with a specific keyword, and other keywords are used within statements. When a keyword is used, it must be specified exactly as shown in the reference format statement that defines the syntax for the particular DDLF statement. Keywords are shown in uppercase in this manual. For a complete list of keywords see appendix D.

User-Defined Names

User-defined names identify the schema, subschema, realm, records, data items, and relations. They are indicated in the formats by lowercase words.

Schema

```
SCHEMA NAME IS CONCATBASE
  .
  .
  .
AREA NAME IS CONCTKY.
RECORD NAME IS KEYLIST-REC WITHIN CONCTKY.
   01  ITEM-A   PICTURE   "X(10)".
   01  ITEM-B   PICTURE   "X(10)".
   01  ITEM-C   PICTURE   "X(20)".
DATA CONTROL.
  .
  .
  .
AREA NAME IS CONCTKY
KEY ID IS CONCAT <ITEM-A, ITEM-B, ITEM-C>
```

Subschema Input

```
SUBSCHEMA FTCONCAT, SCHEMA=CONCATBASE
ALIAS (RECORD)  KEYLREC=KEYLIST-REC
ALIAS (ITEM)    ITEMA=ITEM-A
ALIAS (ITEM)    ITEMB=ITEM-B
ALIAS (ITEM)    ITEMC=ITEM-C

REALM CONCTKY

RECORD, KEYLREC

CHARACTER *10 ITEMA
CHARACTER *10 ITEMB
CHARACTER *20 ITEMC
```

Subschema Output

```
PRIMARY KEY CONCAT (ITEMA,ITEMB,ITEMC) FOR
              AREA CONCTKY
```

Figure 4-4. Declaring a Concatenated Key

All names are taken from the schema except for schema entities that are renamed in the subschema by the ALIAS statement. The rules for forming names differ between data item names and all other names as follows:

Data item names

Since these are used in the FORTRAN program as variable and array names, they must correspond to the FORTRAN rules. They must be from one to seven characters long, contain only letters and digits, and begin with a letter.

Other names

These are used only by DMS-170 and consequently follow the more lenient rules of the schema. They must contain from 1 to 30 letters, digits, or hyphens. The first character must be a letter, and hyphens cannot be used at the beginning or end or adjacent to each other.

Since each data item appears in a type statement, default types based on the first letter of the item name are not applicable in the subschema. Non-data base items (used in the RESTRICT statement) are not given types by DDLF, and consequently must appear in type statements in the FORTRAN program if other than default types are desired for them.

A number of variables are generated in the FORTRAN program by the DML preprocessor; definition or declaration of these variables in a FORTRAN program can lead to invalid results. A complete list of these variables can be found in table 4-4. The variable names indicated in this table must be specified as user-defined names in a subschema source program.

TABLE 4-4. NAMES OF VARIABLES AND COMMON BLOCKS GENERATED BY THE DML PREPROCESSOR

Name	Explanation
DBFnnnn	nnnn is 0001 through 9999
DBInnnn	nnnn is 0001 through 9999
DBNnnnn	nnnn is 0001 through 9999
DBREALM	
DBRELST	
DBRUID	
DBRnnnn	nnnn is 0001 through 9999
DBSCNAM	
DBSTAT	
DBSnnnn	nnnn is 0001 through 9999
DBTEMP	
DBTnnnn	nnnn is 0001 through 9999
DBnnnn	nnnn is 0001 through 9999
Dnnnnxx	nnnn is 0001 through 9999 and xx is AA through ZZ

Constants

With a few restrictions, constants appearing in DDLF statements follow the rules for FORTRAN constants. These rules are defined in the FORTRAN reference manual corresponding to the version of FORTRAN specified on the control statement used to compile the subschema. The constants allowed are restricted to integer or real constants without exponents, and to strings delimited by quotation marks or to strings delimited by apostrophes. The following forms are not allowed: double precision, complex, logical, octal, and hexadecimal constants; Hollerith constants in H, L, or R format; and real constants with exponents. The following items indicate constants that can be used:

Integer constants consist of an optional sign (+ or -) followed by 1 to 18 digits. For example:

```
0
-12345
+2000
000000000000000004
```

Real constants consist of an optional sign (+ or -) followed by a string of digits containing exactly one decimal point. The maximum number of digits is 15. The decimal point can be anywhere within the string; that is, the number can consist of a fractional part, a whole number part, or both. For example:

```
0.
-3.22
+4000.
-.5
.0
1.414
```

Rules for nonnumeric constants depend on the version of FORTRAN. In a subschema for a FORTRAN 5 program, the constant can be a character constant (a string of from 1 to 255 characters delimited by apostrophes) or a Hollerith constant (a string of from 1 to 10 characters delimited by quotation marks). If a delimiting character is to be used in the string, the character must be specified twice for each occurrence. For example, "A""B" would yield the constant A"B; 'C''D' would yield the constant C'D. For example:

```
FORTRAN 5 Hollerith constant
    "MAX10CHARS"

FORTRAN 5 character constant
    'CATS & DOGS'
```

FORTRAN DDL STATEMENT FORMAT

FORTRAN DDL statements occupy from 1 to 100 character positions (columns) in a source line. Table 4-5 shows the usage of the character positions within the source line.

TABLE 4-5. COLUMN USAGE IN FORTRAN DDL STATEMENTS

Column	Usage
1	C or * indicates a comment line.
1-5	Optional statement label containing one through five digits.
6	Character other than blank or zero indicates a continuation line; does not apply to comment lines.
7-72	Text of FORTRAN DDL statements.
73-100	Identification field, listed but not otherwise processed by DDL.

No statement can begin on a line that includes any part of the previous statement; the $ statement separator is not used.

The following paragraphs describe other aspects of coding FORTRAN DDL statements.

Character Set

The FORTRAN DDL character set is a subset of the FORTRAN character set. It consists of the letters A through Z, the digits 0 through 9, and the following special characters:

	Blank
=	Equal sign
,	Comma
(	Left parenthesis
)	Right parenthesis
.	Decimal point
"	Quotation mark
'	Apostrophe
+	Plus sign
-	Minus sign

In addition, any character (appendix A) can be used in character constants and in comments.

Blanks

Unlike FORTRAN statements, in which blanks are ignored (except in character constants), DDLF statements forbid or require blanks in specific cases. The rules are as follows:

Blanks are significant in character constants.

Keywords, user-defined names, relational operators, and constants cannot be interrupted by blanks.

Permitted:

```
RESTRICT RECA(ITEM1.EQ.5.0)
```

Not permitted:

```
REST RICT REC A(ITEM 1. EQ.5 .0)
```

Keywords must be separated from adjacent names by at least one blank. Blanks are not required when a keyword is set off by other special characters.

Permitted:

```
DOUBLE PRECISION D1, D2(10,20),D3
```

Not permitted:

```
DOUBLEPRECISIOND1,D2(10,20),D3
```

This example clarifies that DOUBLE and PRECISION are considered separate keywords, even though they are used together.

Continuation

Statements can be continued for more than one line. A character other than a blank or zero in column 6 indicates that the line is a continuation line. Columns 1 through 5 of a continuation line must contain blanks. A line with a C or * in column 1 and any character in column 6 is a comment line, not a continuation line. The maximum number of continuation lines in one statement is 19.

The END statement cannot be continued.

Statement Labels

As in FORTRAN, any statement can contain a label in columns 1 through 5. A label is a one to five digit integer. Labels do not affect DDLF processing, but can be included for documentary reasons. No diagnostic is issued if the same label is used more than once.

Labels on type statements become FORTRAN statement labels when the type statements are copied into the FORTRAN source program. In the FORTRAN program, type statements are specification statements and are not executable. If any label on the type statements duplicates a label on another FORTRAN statement, a diagnostic is issued by the FORTRAN compiler.

Comment Lines

A line with a C or * in column 1 is a comment line. Comment lines are copied to the DDLF output listing but are not otherwise processed. Comments can contain any character in the character set. Comments do not interrupt statement continuation.

Blank Lines

Blank lines can be used freely between statements to produce blank lines on the source listing. With FORTRAN 5, a blank line is considered a comment line and does not break the continuation sequence.

FORTRAN SUBSCHEMA SYNTAX

The source program for a FORTRAN subschema contains various statements. The following paragraphs define the format specifications for each statement that can be used in the FORTRAN subschema source program. The keywords used in each statement are indicated by capital letters. Keywords and punctuation must appear exactly as shown in the reference format statements.

The general format of a FORTRAN subschema is shown in figure 4-5.

For the rules governing the structure and coding of FORTRAN subschemas, refer to the Subschema Structuring Requirements subsection and to the Subschema Programming Conventions subsection. (Both subsections appear earlier in this section.)

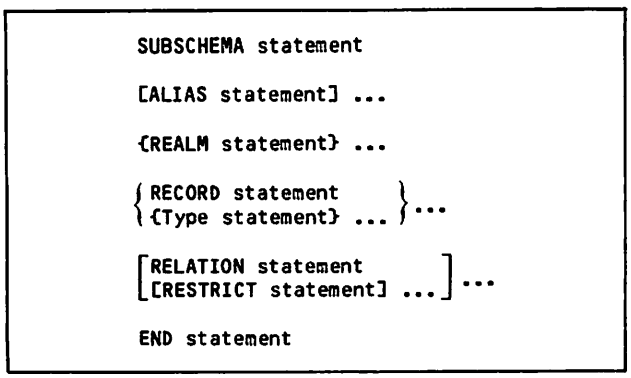


Figure 4-5. General Format, FORTRAN Subschema

SUBSCHEMA STATEMENT

The SUBSCHEMA statement must be the first statement in every FORTRAN DDL program. It names the schema and the subschema. Only one SUBSCHEMA statement can appear. The format of the SUBSCHEMA statement is shown in figure 4-6.

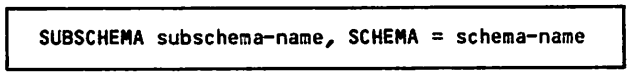


Figure 4-6. SUBSCHEMA Statement Format

The subschema named in the SUBSCHEMA statement is used whenever the subschema is referenced after it has been compiled and stored in the subschema library. The name must be unique among subschemas associated with the designated schema.

The schema-name identifies the schema to which the subschema belongs. The file containing the schema directory must be available to the compiler when the subschema is compiled.

ALIAS STATEMENT

The ALIAS statement is used to change the name of an entity from the name used in the schema to the name used in the subschema. The name to be changed can be a realm, a record, or a data item. The format of the ALIAS statement is shown in figure 4-7.

The ALIAS statement is optional; if used, it must immediately follow the SUBSCHEMA statement.

Assigning an alias in the subschema does not change the name in the schema; it is a substitute name that is used only in the subschema and in the FORTRAN programs referencing the subschema. The old name must appear in the schema; once an alias has been assigned, the new name, rather than the old name, must be used exclusively in the subschema and the FORTRAN program.

The ALIAS statement must specify a name type option; REALM, RECORD, and ITEM. The name type specified (for example, ITEM) applies to all names in that ALIAS statement.

Any number of ALIAS statements can be used, but no old name or new name can appear more than once within the set of ALIAS statements in a subschema.

The ALIAS statement must be used to change item names longer than seven characters or containing hyphens. Such names are valid in the schema but are not legal symbolic names in FORTRAN. The exception is a concatenated key name which can only be referenced in a DML READ statement or START statement. An ALIAS statement is not required to change a concatenated key name that is longer than seven characters or contains hyphens; in this case, it will be ignored if specified because the key name is not a data item. Examples of assigning aliases are shown in figure 4-8.

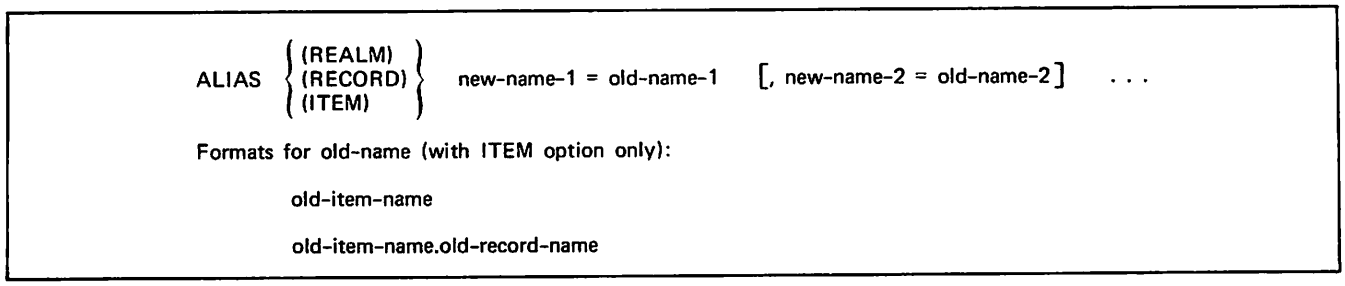


Figure 4-7. ALIAS Statement Format

```
Example 1

    Schema:

        01  LAST-IN-FIRST-OUT  PICTURE "X(10)".

    Subschema:

        ALIAS (ITEM) LIFO = LAST-IN-FIRST-OUT

Example 2

    Schema:

        RECORD NAME IS TOTAL-REC.
        01  ACCOUNT          PICTURE "99999".
        .
        .
        .
        RECORD NAME IS PARTIAL-REC.
        01  ACCOUNT          PICTURE "99999".

    Subschema:

        ALIAS (ITEM) ACNTTOT = ACCOUNT.TOTAL-REC,
       1    ACNTPAR = ACCOUNT.PARTIAL-REC
```

Figure 4-8. Assigning Aliases

The form old-item-name.old-record-name must be used to provide unique names within the FORTRAN program when the same name appears in two different records in the schema. Example 2 in figure 4-8 illustrates qualifying an old-item-name with an old-record-name.

REALM STATEMENT

The REALM statement identifies the specific schema areas that are used in the subschema and in the FORTRAN application programs referencing the subschema. Realm and area are subschema and schema terms, respectively, for what is known to the operating system and to FORTRAN as a file. The format of the REALM statement is shown in figure 4-9.

```
REALM {ALL
       realm-name-1 [, realm-name-2] ...}
```

Figure 4-9. REALM Statement Format

At least one REALM statement must appear in the subschema. If ALIAS statements are used, the REALM statements must immediately follow the last ALIAS statement; otherwise, they must immediately follow the SUBSCHEMA statement.

Any number of REALM statements can be included, but no realm can be named more than once. All realms named must appear in the schema. The ALL option specifies that all the realms defined in the schema are to be available to the subschema. Unless the ALL option is used, only those realms named in a REALM statement are available to the subschema and to the FORTRAN programs referencing the subschema.

RECORD DEFINITION

A record definition is composed of a RECORD statement followed by one or more type statements. The record definitions must immediately follow the REALM statements, and each subschema must contain at least one record definition. Each record definition corresponds to a record type defined in the schema.

The type statements following a RECORD statement specify those data items occurring within the record that are to be used in the subschema. These data items can be used as variables or arrays in the FORTRAN program. The type statements also define the FORTRAN data types of the items, as well as the number of dimensions and size of arrays. Data items not included in the subschema cannot be accessed by the FORTRAN program. The items defined for a record in the subschema can differ in order, number, and (in some cases) type from the equivalent definitions in the schema. For more information, refer to the Schema/Subschema Compatibility subsection which appears earlier in this section.

RECORD Statement

The RECORD statement identifies a schema record that is to be used in the subschema. Any number of RECORD statements, each followed by a group of type statements, can appear in the subschema, but no record name can be used more than once. The format of the RECORD statement is shown in figure 4-10.

```
RECORD record-name
```

Figure 4-10. RECORD Statement Format

Each record specified in a RECORD statement must appear in the schema. The schema links record types with areas; the area associated with a record in the subschema must appear in a REALM statement. Only one record type per realm in the subschema is permitted.

Type Statements

Type statements identify and describe the data items that are to be made available to the subschema and the applications program. The DDLF compiler recognizes the same type statements as the version of the FORTRAN compiler specified in the DDLF control statement. The rules governing type statements and the amount of storage allocated to each variable are described in the FORTRAN 5 reference manual. The formats of the type statements are shown in figure 4-11.

FORTRAN 5 allows the following type statements: CHARACTER, BOOLEAN, REAL, INTEGER, LOGICAL, COMPLEX, and DOUBLE PRECISION. FORTRAN 5 syntax requires that both the keywords DOUBLE and PRECISION be specified.

FORTRAN 5 Format

```
{ CHARACTER [*default-length [,]] item-name-1 [*len-1] [,item-name-2 [*len-2]] . . .
{ BOOLEAN          }
{ REAL             }
{ INTEGER          }  item-name-1 [,item-name-2] . . .
{ LOGICAL          }
{ COMPLEX          }
{ DOUBLE PRECISION }
```

In a CHARACTER type statement, default-length and len must be positive integer constants. If neither default-length nor len is specified, 1 is assumed.

In each type statement, the formats for item-name are:

item-name

item-name (d1 [,d2] . . .)

where d specifies the bounds of an array dimension. From one through seven bounds can be specified for an array.

The formats for d are:

upper-bound

lower-bound: upper-bound

where the bound specification must be an integer constant: positive, zero, or negative. The upper-bound must be greater than or equal to the lower-bound. (When the lower-bound is not specified, 1 is assumed.)

Figure 4-11. Type Statement Formats

Each data item referenced in the subschema must appear in a type statement. If it is an array, its dimensions are given in the same type statement. No item can appear in more than one type statement. If an item occurs in more than one record type in the schema, it must be renamed by the ALIAS statement in the subschema. Since each item appears in a type statement, implicit typing according to the first letter of the item name is not recognized by the DDLF compiler.

Each item appearing in a type statement must be defined in the schema. (It could be defined under a different name if it appears in the ALIAS statement in the subschema.) Each item defined as an array must be defined in the schema with the OCCURS clause. The number of elements in the subschema array must not exceed the number of occurrences in the schema vector.

At least one type statement must be associated with each RECORD statement, even if no items are used by the FORTRAN program (for example, a program that counts the number of records in a file). The variable that is the primary key for the realm must be included in the subschema; alternate keys must be included only if actually used in the program.

Figure 4-12 shows two record definitions. The variables A, B, and X and the array C are to be made available from the record type REC1; the variables D, E, and J and the array H are to be made available from the record type REC2.

```
RECORD REC1
INTEGER A, B, C(12)
REAL X

RECORD REC2
INTEGER D, E
COMPLEX H(2,3,4)
CHARACTER*20 J
```

Figure 4-12. Defining Records

For examples of correspondence between type statements in the subschema and item descriptions in the schema, refer to the Schema/Subschema Correspondence subsection which appears earlier in this section.

RELATION DEFINITION

A relation definition is composed of a RELATION statement optionally followed by any number of RESTRICT statements. Relation definitions are optional; if used, they must immediately follow the last group of RECORD and type statements.

The RELATION statements specify the schema relations to be made available to the FORTRAN program. RESTRICT statements restrict records accessed by a relation to those which satisfy a logical expression. The RESTRICT statements immediately following a RELATION statement describe restrictions on that relation.

RELATION Statement

The RELATION statement specifies that a relation defined in the schema is to be available to the FORTRAN program. Any number of RELATION statements can appear, but no relation can be named more than once. The format of the RELATION statement is shown in figure 4-13.

The relation-name must be the same name specified in the RELATION NAME clause in the schema. The schema specifies which realms and record types apply to a relation. When a relation is declared in the subschema, the associated realms and record types must also be declared in the subschema.

```
RELATION relation-name
```

Figure 4-13. RELATION Statement Format

RESTRICT Statement

The RESTRICT statement limits a relation by specifying criteria that must be satisfied by a record occurrence on a read before it is made available to the FORTRAN program. The format of the RESTRICT statement is shown in figure 4-14.

RESTRICT statements are optional; any number can appear with a RELATION statement. However, only one RESTRICT statement can be included for a given record. All records referenced must be defined in the schema and must be in a realm that is joined by the relation named in the preceding RELATION statement.

The entities to be compared in the logical expression of a RESTRICT statement are data base items, non-data-base items, and constants. A maximum of 1024 data base items, non-data-base items, constants, and operators can appear in the restrictions on any one relation. The following rules apply to the entities that can be compared in the logical expression of a RESTRICT statement:

A data base item must be defined in the subschema as part of the record. It must not be described in the schema through VIRTUAL RESULT or DECODING data base procedures. It must be a variable or an array element with a constant subscript. When two data base items are compared, they must be the same size and must be compatible as determined from the schema data class of each item. Items of schema data classes 0 through 4 (items stored as display code) are compatible. An item of any other schema data class is compatible only with an item of the same schema data class.

Non-data-base items are simple variables that appear in the FORTRAN program but are not defined in the subschema. Use of a non-data-base item in a RESTRICT statement allows dynamic qualification of a relation. Non-data-base items must agree in size and type with the subschema representation of the data base items to which they are being compared. Results are unpredictable if the size and type are not identical.

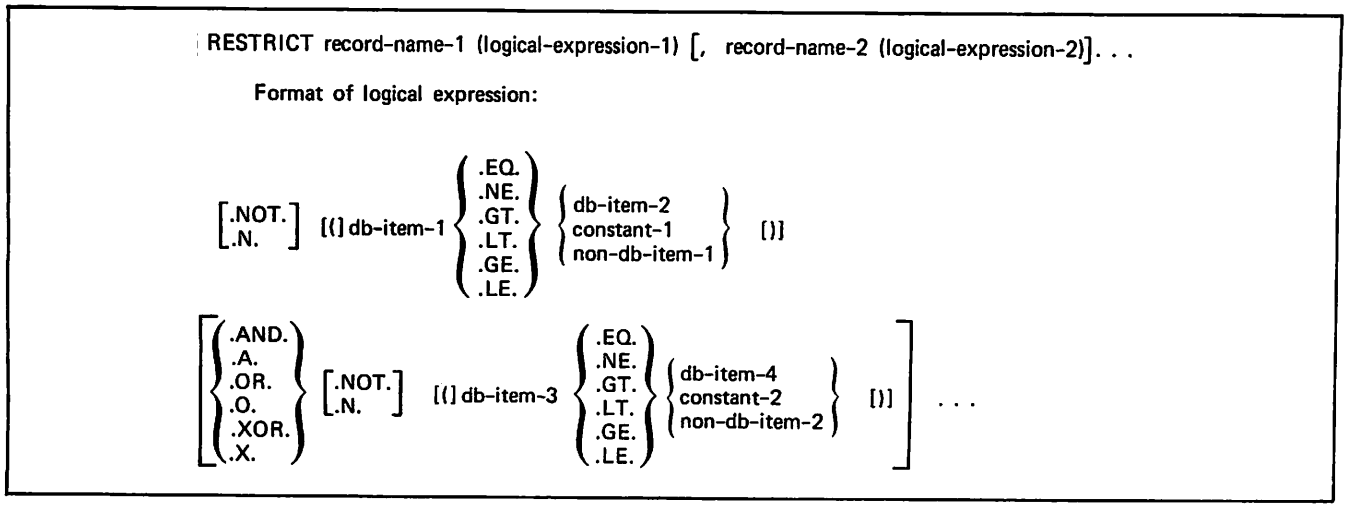


Figure 4-14. RESTRICT Statement Format

Constants are real, integer, Hollerith, or character constants. (Character constants are allowed only in FORTRAN 5.) The forms allowed for constants are shown in the Constants subsection which appears earlier in this section. A constant must be compatible with the schema representation of the data base item to which it is being compared.

When a data base item is compared to another data base item or to a non-data-base item, both items must have the same size and type. Complex and logical items are not allowed. Real, integer, and double precision items are compared by magnitude. If a data base item is described as an integer in the subschema and as a display item in the schema (schema classes 0 through 4), it is assumed to contain character data and can only be compared to a data base item similarly described, to a compatible non-data-base item, or to a Hollerith constant. A real or integer constant can be compared to a data base item of any numeric type (real, integer, or double precision); appropriate conversion is performed before the magnitudes are compared.

Logical expressions are interpreted the same as in FORTRAN. Valid and invalid logical expressions are shown in figure 4-15.

Valid Expressions

```
ITEM1 .GT. ITEM2

X .GE. 4.23 .AND.I .LT. 4

(FIRST .EQ. LAST) .AND. (SECOND .GT. THIRD)

I .OR. .NOT. FOURTH .GT. 0
```

Invalid Expressions

```
.NOT. ((I .GT. 2) .OR. (J .GT. 3))
    Two levels of nesting are not permitted.

LOGICAL L
A .GT. B .OR. L
    Logical variables are not allowed.

COMPLEX C,D
C .GT. D
    Complex variables are not allowed.
```

Figure 4-15. Examples of Logical Expressions

END STATEMENT

The END statement must be the last statement in every FORTRAN DDL subschema source program. The END statement cannot be continued. No statements or comment lines can appear after the END statement. The format of the END statement is shown in figure 4-16.

```
END
```

Figure 4-16. END Statement Format

SUBSCHEMA COMPILATION AND SUBSCHEMA LIBRARY MAINTENANCE

The FORTRAN DDL subschema source program is compiled by the FORTRAN DDL (DDLF) compiler. The DDLF compiler generates a subschema directory or object subschema from the subschema source program. A subschema directory is often simply called the subschema.

The DDLF compiler is a multifunctional compiler. In addition to generating a subschema directory, the compiler stores the directory in a subschema library and performs maintenance operations on the subschema library. A subschema can be used with a FORTRAN 5 application program, depending on the language version specified in the DDLF control statement. More than one subschema can be compiled with one control statement call to the DDLF compiler. The particular operation to be performed by the DDLF compiler is selected by a parameter in the DDLF control statement.

The field length requirements for subschema compilation are indicated in appendix G.

SUBSCHEMA LIBRARY

One or more compiled subschemas are stored in a permanent file called the subschema library. The library is created when the first subschema is stored in it. Subschemas for both language versions can reside in the same subschema library. Subsequent subschemas can be added to the library or can replace existing subschemas in the library. A subschema can be deleted from the library with a purge parameter on the DDLF control statement. A subschema library that has had subschemas replaced or purged can be transferred to a new, compacted subschema library.

Data security can be maintained by creating more than one subschema library in order to control the availability of the subschemas to the FORTRAN programs.

Subschemas providing access to data that is restricted to specific applications can be stored in one subschema library while subschemas for general use can be stored in a different library. Each library is identified by a unique permanent file name. When the application program is compiled, a subschema library must be attached by specifying the permanent file name of the library. Subschemas stored in other libraries are not available to the program.

DDLF CONTROL STATEMENT

The DDLF control statement calls the DDLF compiler for execution. It provides the DDLF compiler with information related to a specific subschema, specifies the listings to be generated, and specifies the subschema library maintenance operations to be performed. The format of the DDLF control statement is shown in figure 4-17. Optionally, the comma immediately following DDLF can be replaced by a left parenthesis; the terminating period can be replaced by a right parenthesis.

All the parameters of the DDLF control statement are optional.

The F5 parameter selects language version. This parameter is interpreted as follows:

omitted

F5 is the default that is assumed.

F5

Specifies a subschema for use with a FORTRAN 5 application program.

The SB parameter identifies the local file name of the subschema library file. This parameter is interpreted as follows:

omitted

Local file SBLFN is assumed to contain the subschema library.

SB=lfn

The local file name specified identifies the file that contains the subschema library.

The A, N, NL=lfn, P and R parameters along with a default operation select subschema library maintenance operations. If none of these parameters is selected, the following library maintenance operation is assumed: adding the compiled subschema to the subschema library. Only one of the operations selected by these parameters can be performed in a single execution of the DDLF compiler. If more than one of these parameters is specified, a control statement error is issued. If a library manipulation function is attempted on a nonempty file that does not contain a library, a diagnostic is issued and the job is aborted. The parameters are interpreted as follows:

omitted

Each subschema source program in the job stream is compiled and added to the subschema library identified by the SB parameter.

A

Audit parameter. A list of the subschemas and their corresponding schemas, together with their creation dates, is produced from the subschema library identified by the SB parameter.

N

Compile parameter. Each subschema source program in the job stream is compiled but not added to the subschema library identified by the SB parameter.

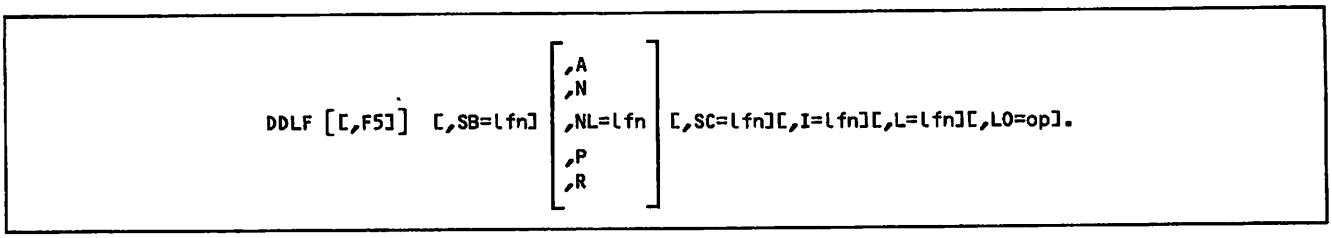


Figure 4-17. DDLF Control Statement Format for FORTRAN Subschemas

NL=lfn

New library parameter. The specified local file name identifies a new subschema library to which the active subschemas in the subschema library, identified by the SB parameter, are transfered. If the NL parameter is specified with no file name indicated, local file name NEWLIB is assumed.

P

Purge parameter. A subschema specified in the job stream is purged from the subschema library identified by the SB parameter. No compilation takes place.

R

Replace parameter. Each subschema source program in the job stream is compiled and replaces the existing subschema (identified by the SUBSCHEMA statement) in the subschema library (identified by the SB parameter). Replacement takes place only if no compilation errors other than informative diagnostics are encountered.

The SC parameter identifies the local file that contains the schema directory. A specified lfn overrides the default file name determined from the SUBSCHEMA statement. The parameter is interpreted as follows:

omitted

The first seven characters of the schema name specified in the SUBSCHEMA statement of the subschema source program is assumed to identify the local file that contains the schema directory.

SC=lfn

The specified local file name identifies the file that contains the schema directory.

The I parameter identifies the local file that contains source input for the DDLF compiler. This parameter is interpreted as follows:

omitted

The local file INPUT is assumed to contain the source input for the DDLF compiler.

I=lfn

The specified local file name identifies the file that contains source input for the DDLF compiler.

The L parameter identifies the local file that receives listings and diagnostics generated by the DDLF compiler. This parameter is interpreted as follows:

omitted

The local file OUTPUT receives the listings and diagnostics generated by the DDLF compiler.

L=0

The local file OUTPUT receives only the diagnostics generated by the DDLF compiler.

L=lfn

The specified local file name identifies the file that receives the listings and diagnostics generated by the DDLF compiler.

The LO parameter selects the listing produced by the DDLF compiler. Two listing options (op) can be specified: a source listing (S option) and an object listing (O option). Source and object listings are written to the file specified by the L parameter. If L=0 is specified, no listing is produced except for error messages regardless of LO specification. The parameter is interpreted as follows:

omitted

Same as LO=S; a source listing is produced.

LO

Same as LO=S; a source listing only is produced.

LO=op
LO=op_1/op_2

The specified listing option selects the listing produced. The listing options are as follows:

S Source listing

O Object listing

SUBSCHEMA COMPILATION AND LIBRARY MAINTENANCE OPERATIONS

The FORTRAN subschema must be coded according to the specifications in this manual. Subschema source code can be entered through a terminal, processed by a text editor, and stored in a file.

The DDLF compiler can be executed from a terminal and through a batch job in the following ways. When executed from a terminal, the subschema program must reside on a file whose name is indicated by the I parameter of the DDLF control statement. In a batch job in which the input file for the DDLF compiler is assumed to be local file INPUT, the job stream must be structured so that the control statements precede the subschema source program. An end-of-record indicator must immediately precede the first line of subschema source code to separate the subschema program from the control statements.

Control statements in the job stream and the resulting subschema source program are used by the DDLF compiler either to compile the subschema and to store it in the subschema library or to perform a subschema library maintenance operation such as replacing or deleting subschemas, generating a new, compacted subschema library, or producing an audit listing. The parameters in the DDLF control statement select the operation to be performed. Other control statements provide information for the operating system to make necessary information available for the DDLF compiler.

The following paragraphs indicate the parameters required in the DDLF control statement to accomplish compilation of FORTRAN subschemas and to perform subschema library maintenance operations. Examples of batch jobs that accomplish the operations are shown. The DDLF control statements used in the examples indicate FORTRAN 5 subschemas. Control statements are shown for both the NOS and NOS/BE operating systems.

Compiling a Subschema

The source program for a subschema can be compiled without being stored in the subschema library. The form of the DDLF control statement necessary to compile a subschema is as follows:

```
DDLF,F5,SB=lfn,N,SC=lfn.
```

The compile parameter (N) must be specified. The language version parameter (F5) is optional and has a default value. The SB, SC, I, L, and LO parameters are optional and have default values (refer to the DDLF Control Statement subsection).

Figure 4-18 illustrates a job that accomplishes subschema compilation only. The schema file must be attached.

Creating a Subschema Library

The subschema library is created when the first subschema is stored in it. The form of the DDLF control statement necessary to create a subschema library and store a subschema in it is as follows:

```
DDLF,F4,SB=lfn,SC=lfn.
```

The language version parameter (F5) is optional and has a default value. The SB, SC, I, L, and LO parameters are optional and have default values (refer to the DDLF Control Statement subsection).

Figure 4-19 illustrates a job that compiles a subschema and creates a subschema library. In this example, the schema file SCHPAY is attached. The subschema library is identified by the local file name SUBSCH. The DEFINE and REQUEST/CATALOG control statements specify the local file name of the subschema library and assign it to a permanent file device. A password specified in the DEFINE or CATALOG statement controls subsequent access and use of the subschema library.

If only one subschema is to be stored in the library, the subschema name can be used for the subschema library file name.

Compiling Multiple Subschemas

A number of subschemas can be compiled with one control statement call to the DDLF compiler. The parameters specified in the control statement apply to all the subschemas. The subschema library maintenance operations of creating a subschema library, of adding (default operation) subschemas to the subschema library, of replacing (R parameter) subschemas in the library, or of just compiling (N parameter) subschemas can be used with this facility.

For the compilation of several subschemas with one control statement call to the DDLF compiler, the subschema source statements must be contiguous, with no intervening end-of-record indicator or

NOS Operating System

```
Job statement
USER control statement
CHARGE control statement
ATTACH,SCHPAY.
DDLF,F5,SB=SBTST,N,SC=SCHPAY.
End-of-record
   DDL Subschema Source Input
End-of-information
```

NOS/BE Operating System

```
Job statement
ACCOUNT control statement
ATTACH,SCHPAY,ID=DDL.
DDLF,F5,SB=SBTST,N,SC=SCHPAY.
End-of-record
   DDL Subschema Source Input
End-of-information
```

Figure 4-18. Compiling a Subschema

NOS Operating System

```
Job statement
USER control statement
CHARGE control statement
ATTACH,SCHPAY.
DEFINE,SUBSCH/PW=DDL,CT=PU,M=W.
DDLF,F5,SB=SUBSCH,SC=SCHPAY.
End-of-record
   DDL Subschema Source Input
End-of-information
```

NOS/BE Operating System

```
Job statement
ACCOUNT control statement
ATTACH,SCHPAY,ID=DDL.
REQUEST,SUBSCH,PF.
DDLF,F5,SB=SUBSCH,SC=SCHPAY.
CATALOG,SUBSCH,ID=DDL,MD=DDL,EX=DDL,CN=DDLX.
End-of-record
   DDL Subschema Source Input
End-of-information
```

Figure 4-19. Creating a Subschema Library

end-of-information indicator. The program source for each subschema must begin with the SUBSCHEMA statement. Each subsequent SUBSCHEMA statement must immediately follow the END statement of the preceding subschema. Each subschema is compiled in turn until end-of-information or end-of-record is encountered.

Compiling a Subschema and Adding to a Subschema Library

Once the subschema library has been created and made a permanent file (as described in the Creating a Subschema Library subsection), new subschemas can be added to the library. The form of the DDLF control statement necessary to add a subschema to an existing subschema library is as follows:

```
DDLF,F5,SB=lfn,SC=lfn.
```

The SB and SC parameters are optional and have default values. The I, L, and LO parameters must be specified if default values are not applicable for input and output files or for the listing option (refer to the DDLF Control Statement subsection).

Figure 4-20 illustrates a job that adds a subschema to the subschema library created by the example in figure 4-19. The schema directory is attached. The subschema library is also attached with the applicable password and (for NOS only) access mode M=W specified.

Each subschema stored in the library must have a unique name. If the subschema being added to the library has the same name as a subschema already stored in the library, a diagnostic is issued and the job is aborted.

Replacing a Subschema

A new subschema can replace one that is stored in the subschema library. The form of the DDLF control statement necessary to replace a subschema in the subschema library is as follows:

```
DDLF,F5,SB=lfn,R,SC=lfn.
```

The replacement parameter (R) must be specified. The language version parameter (F5) is optional and has a default value. The SB, SC, I, L, and LO parameters are optional and have default values (refer to the DDLF Control Statement subsection).

Figure 4-21 illustrates a job that replaces a subschema in the subschema library. The schema directory is attached. The subschema library file is also attached with the applicable password, and (for NOS only) access mode M=W is specified. If the subschema to be replaced cannot be found in the subschema library, an informative diagnostic is issued and the new subschema is added to the library.

Deleting a Subschema

A subschema stored in the subschema library can be deleted from the library. The form of the DDLF control statement necessary to purge a subschema from the subschema library is as follows:

```
DDLF,F5,SB=lfn,P.
```

The purge parameter (P) must be specified. The language version parameter (F5) is optional and has a default value. The SB, I, and L parameters are optional and have default values (refer to the DDLF Control Statement subsection).

NOS Operating System

```
Job statement
USER control statement
CHARGE control statement
ATTACH,SCHPAY.
ATTACH,SUBSCH/PW=DDL,M=W.
DDLF,F5,SB=SUBSCH.
End-of-record
   DDL Subschema Source Input
End-of-information
```

NOS/BE Operating System

```
Job statement
ACCOUNT control statement
ATTACH,SCHPAY,ID=DDL.
ATTACH,SUBSCH,ID=DDL,PW=DDL.
DDLF,F5,SB=SUBSCH.
End-of-record
   DDL Subschema Source Input
End-of-information
```

Figure 4-20. Compiling a Subschema and Adding to a Subschema Library

NOS Operating System

```
Job statement
USER control statement
CHARGE control statement
ATTACH,SCHPAY.
ATTACH,SUBSCH/PW=DDL,M=W.
DDLF,F5,SB=SUBSCH,R.
End-of-record
   DDL Subschema Source Input
End-of-information
```

NOS/BE Operating System

```
Job statement
ACCOUNT control statement
ATTACH,SCHPAY,ID=DDL.
ATTACH,SUBSCH,ID=DDL,PW=DDL.
DDLF,F5,SB=SUBSCH,R.
End-of-record
   DDL Subschema Source Input
End-of-information
```

Figure 4-21. Replacing a Subschema Library

Figure 4-22 illustrates a job that purges a subschema from the subschema library. The subschema library file is attached with the applicable password and (for NOS only) access mode M=W is specified. The end-of-record indicator designates the end of the control statements.

The control statements are followed by statements that specify the subschemas to be deleted. The subschema name is entered anywhere from column 7 through column 72. If more than one subschema name is entered, a space or a comma must follow each subschema name.

No compilation occurs when a subschema is deleted from the subschema library.

Auditing a Subschema Library

After a subschema library is created, it can be searched and a list generated that includes the name of each subschema and the name of the schema it references, together with their creation dates. The form of the DDLF control statement necessary to have the audit listing generated is as follows:

```
DDLF,F5,SB=lfn,A.
```

The audit parameter (A) must be specified. The language version parameter (F5) is optional and has a default value. The SB and L parameters are optional and have default values (refer to the DDLF Control Statement subsection).

Figure 4-23 illustrates a job that produces an audit listing of the subschema library. The subschema library is attached with the appropriate password specified. The listing produced by the DDLF compiler when the audit is performed is shown in figure 4-24.

Compacting a Subschema Library

Subschema library compaction is an optional facility that generates a new subschema library by copying only the active subschemas from the specified subschema library. When this facility is executed, the DDLF compiler generates a listing that includes the name of each transferred subschema and the schema it references, together with their creation dates. The form of the DDLF control statement necessary to create a new, compacted subschema library is as follows:

```
DDLF,F5,SB=lfn,NL=lfn.
```

The new library parameter (NL) must be specified. The language version parameter (F5) is optional and has a default value. The SB and L parameters are optional and have default values (refer to the DDLF Control Statement subsection).

Figure 4-25 illustrates a job that creates a new, compacted subschema library. The subschema library is attached with the appropriate password. The DEFINE and REQUEST/CATALOG control statements specify the local file name of the new subschema library and assign it to a permanent file device. A password specified in the DEFINE or CATALOG statement controls subsequent access and use of the subschema library. The DDLF control statement specifies the SB parameter with the local file name of the current subschema library to be compacted, and the NL parameter with the local file name of the new subschema library. The PURGE control statement destroys the old library file.

This facility is intended to be used on a subschema library that has had a number of subschemas purged or replaced and, therefore, contains wasted space. The DDLF compiler eliminates the wasted space in the new subschema library. After the new subschema library is stored as a permanent file, the user should purge the old subschema library. This facility does not allow for compilation of subschemas.

NOS Operating System

```
Job statement
USER control statement
CHARGE control statement
ATTACH,SUBSCH/PW=DDL,M=W.
DDLF,F5,SB=SUBSCH,P.
End-of-record
        PAYROLL
        NEWHIRES
End-of-information
```

NOS/BE Operating System

```
Job statement
ACCOUNT control statement
ATTACH,SUBSCH,ID=DDL,PW=DDL.
DDLF,F5,SB=SUBSCH,P.
End-of-record
        PAYROLL
        NEWHIRES
End-of-information
```

Figure 4-22. Deleting Subschemas From the Library

NOS Operating System

```
Job statement
USER control statement
CHARGE control statement
ATTACH,SUBSCH/PW=DDL.
DDLF,F5,SB=SUBSCH,A.
End-of-information
```

NOS/BE Operating System

```
Job statement
ACCOUNT control statement
ATTACH,SUBSCH,ID=DDL,PW=DDL.
DDLF,F5,SB=SUBSCH,A.
End-of-information
```

Figure 4-23. Auditing a Subschema Library

```
                                    * SOURCE LISTING *   (82061)  DDLF 1.3+564.      82/04/05. 09.56.13.


          -----        BEGIN SUB-SCHEMA FILE MAINTENANCE                -----

        LIST OF SUB-SCHEMAS IN FILE

        SUB-SCHEMA                        CREATION          ?     SCHEMA                          CREATION
                                        DATE    TIME                                            DATE    TIME
        ----------                     ----------------     ?     --------                      ----------------
        F5SS-PRODUCT-MANAGEMENT        82089   09.09              MANUFACTURING-DB              82087   14.14
        F5SS-TESTS                     82089   09.09              MANUFACTURING-DB              82087   14.14
        F5SS-PRODUCTS                  82089   09.09              MANUFACTURING-DB              82087   14.14
        F5SS-PROJECTS                  82089   09.09              MANUFACTURING-DB              82087   14.14
        F5SS-PRODUCT-EVALUATION        82089   14.41              MANUFACTURING-DB              82087   14.14
        F5SS-TESTING                   82089   14.41              MANUFACTURING-DB              82087   14.14

          -----        END OF FILE MAINTENANCE                          -----
DDLF COMPLETE.          0 DIAGNOSTICS.
  466008 CM USED.     0.024  CP SECS.
```

Figure 4-24. Audit Listing of the Subschema Library

NOS Operating System

```
Job statement
USER control statement
CHARGE control statement
ATTACH,SUBSCH/PW=DDL,M=W.
DEFINE,NEWSUB/PW=DDL,CT=PU,M=W.
DDLF,F5,SB=SUBSCH,NL=NEWSUB.
PURGE,SUBSCH.
End-of-information
```

NOS/BE Operating System

```
Job statement
ACCOUNT control statement
ATTACH,SUBSCH,ID=DDL,PW=DDL.
REQUEST,NEWSUB,PF.
DDLF,F5,SB=SUBSCH,NL=NEWSUB.
CATALOG,NEWSUB,ID=DDL,MD=DDL,EX=DDL,CN=DDLX.
PURGE,SUBSCH,RB=1.
End-of-information
```

Figure 4-25. Compacting a Subschema Library

COMPILATION OUTPUT

A listing of the FORTRAN DDL source program is provided whenever a subschema is compiled. Each line of the listing corresponds to one source line in the source program. The format and order of each line on the listing are identical to the format and order of the statements in the source program. Figure 4-26 is a sample compilation output listing for a subschema compilation.

The DDLF compiler assigns a line number to each input statement beginning with 00001. The line numbers are printed on the source listing starting in column 16. Diagnostic messages begin in column 3 of the listing. After the last input statement is listed, a compilation summary is printed. When relation statistics are applicable, relation names and their traversed areas are included.

The source listing can be suppressed by specifying L=0 on the DDLF control statement. Diagnostic messages and the compilation summary are the only listing printed.

A cross-reference list and a data map are not printed when the subschema is compiled. When the FORTRAN applications program containing DML statements is compiled, the variables and arrays defined in the subschema or generated by the DML preprocessor are not listed unless the DS parameter is specified on the DML control statement.

RECOMPILATION GUIDELINES

The DDLF compiler generates a checksum (an identifying 1-word attribute) for each area and relation in the schema. These checksums are the means for determining the need to recompile a subschema. If a checksum in a recompiled schema is different from the corresponding checksum in the previous schema, any subschema referencing the changed element must be recompiled. A subschema not referencing a changed element does not need to be recompiled.

The DDLF compiler generates a single checksum for a subschema. This checksum is used for determining the need to recompile a FORTRAN-DML program. If a checksum of a recompiled subschema differs from the previous checksum, all application programs referencing that subschema must be processed again by the DML preprocessor and recompiled. When the FORTRAN-DML program is compiled, the checksum of the subschema it references is copied into the program binary output. When the program is executed, that checksum must be the same as a checksum of a subschema in the master directory. If the program references an invalid checksum, CDCS aborts the program and issues a diagnostic.

```
F5SS-PRODU                    * SOURCE LISTING *   (82061)  DDLF 1.3+564.

                00001                SUBSCHEMA F5SS-PRODUCT-EVALUATION, SCHEMA=MANUFACTURING-DB
                00002
                00003                ALIAS (REALM) PRODUCT-FILE = DEVELOPMENT-PRODUCTS
                00004                ALIAS (RECORD) PRODREC = DEVREC
                00005                ALIAS (ITEM) PRODUCT = PRODUCT-ID
                00006                ALIAS (ITEM)EVALID = EVAL-ID
                00007                ALIAS (ITEM) PROJECT = PROJECT-ID
                00008                ALIAS (ITEM) NTESTED = NUM-TESTED
                00009                ALIAS (ITEM) AVG = CUM-TEST-AVERAGE
                00010                ALIAS (ITEM) STATUS = STATUS-CODE
                00011
                00012                REALM PRODUCT-FILE, TESTS
                00013
                00014                RECORD PRODREC
** WITHIN PRODUCT
                00015                CHARACTER *10 PRODUCT
** ORDINAL    1
                00016                CHARACTER *2  CLASS
** ORDINAL    2
                00017                CHARACTER *20 EVALID(10)
** ORDINAL    3
                00018                CHARACTER *4  PROJECT
** ORDINAL    4
                00019                CHARACTER *1  STATUS
** ORDINAL    5
                00020                INTEGER PRICE, NTESTED
** ORDINAL    7
                00021                REAL AVG
                00022
** ORDINAL    8
                00023                RECORD TESTREC
** WITHIN TESTS
                00024                INTEGER TESTNO
** ORDINAL    1
                00025                CHARACTER *20 TNAME
** ORDINAL    2
                00026                CHARACTER *10 PRDCTNO
** ORDINAL    3
                00027                INTEGER N, TOTALCT
** ORDINAL    5
                00028                REAL PASPROB(100)
                00029
** ORDINAL    6
                00030                RELATION TEST-REL
 PRIMARY KEY    00015           PRODUCT FOR AREA PRODUCT-FILE
 ALTERNATE KEY  00018           PROJECT FOR AREA PRODUCT-FILE
 ALTERNATE KEY  00017           EVALID FOR AREA PRODUCT-FILE
 PRIMARY KEY    00024           TESTNO FOR AREA TESTS
 ALTERNATE KEY  00025           TNAME FOR AREA TESTS
                *****           RECORD MAPPING IS NEEDED FOR REALM -   PRODUCT-FILE
                *****           RECORD MAPPING IS NEEDED FOR REALM -   TESTS
                00031                RESTRICT PRODREC (STATUS .EQ. 'A' .OR. STATUS .EQ. 'R')
                00032
                00033                END
                *****           END OF SUB-SCHEMA SOURCE INPUT

                *****                    RELATION   STATISTICS                  *****
 RELATION 001                   TEST-REL JOINS            AREA - TESTS
                                                          AREA - PRODUCT-FILE


                -----           BEGIN SUB-SCHEMA FILE MAINTENANCE               -----

          SUBSCHEMA                                CHECKSUM
          F5SS-PRODUCT-EVALUATION                  41264663066177205611

                -----           END OF FILE MAINTENANCE                 -----
 DDLF COMPLETE.          0 DIAGNOSTICS.
   51000B CM USED.      0.322  CP SECS.
```

Figure 4-26. Sample FORTRAN Subschema Compilation Output Listing

A schema describes the data in a data base. The subschema describes only data that is accessible by a particular application program. The data described by the schema can be changed to meet the requirements of an application program by the inclusion, omission, reordering, redefinition, and renaming of data in the subschema.

The process of resolving the differences between the data types and structures in the schema and those in the subschema is called mapping. By using mapping, CYBER Data Base Control System (CDCS) can generate a record image conforming to the subschema format from a record in schema format, or can perform the conversion of data from subschema format to schema format.

STRUCTURING RESTRICTIONS

Some characteristics of the data in a data base are fixed by the schema and cannot be changed by the subschema; other characteristics specified in the schema can be different in the subschema. The following variations are allowed in the COBOL, Query Update, and FORTRAN subschemas with the exception of those pertaining to group items, which are not allowed in a FORTRAN subschema:

- Redefinition of the characteristics of data items to data types for which conversion is supported
- Omission or inclusion of specific areas, records, group items, elementary items, and relations
- Reordering of elementary items and group items
- Renaming of elementary items, group items, record names, area names, and others

The limitations placed on the variations between the schema and the subschema are precise and must be carefully considered when the subschema is described. Rules are given in sections 3 and 4.

DATA CONVERSION

By using mapping, CDCS can perform the transfer and conversion of individual data items from one data representation to another. The mapping process operates in a source-target mode. For example, for a read request, the source record obtained from the data base is in schema format and the target record produced by the mapping operation is in subschema format. The converse is true for a write operation: the source record is in subschema format and the target record produced is in schema format. Mapping occurs after a record is retrieved from a data base, but prior to writing a record.

The entire record image is transferred as a whole unless item-level operations are required for record restructuring, data validation, data item conversion, or item-level data base procedures. In this case, CDCS uses mapping to transfer and convert (if necessary) the source record, item by item, to the target record as described in the schema and subschema data descriptions. Data validation and calls to item-level data base procedures, if specified, are performed during the mapping of each data item.

If an error occurs during an item-level operation for a function such as record mapping, data validation, or item-level data base procedure processing, a value indicating the subschema item ordinal for the error, as well as the name of the file on which the error occurred, can be obtained by the application program through the data base status block. The status block, which is described in the CDCS 2 Application Programming reference manual, must have been activated at a previous point in the program in order to return the information. Query Update automatically returns error information and the subschema item ordinal in a diagnostic. A subschema ordinal is assigned to each item in a subschema when the subschema is compiled. The application programmer can use the subschema output listing to associate the subschema item ordinal returned in the data base status block with the item on which the error actually occurred.

Data validation is activated by the CHECK VALUE clause in the schema. Item-level data base procedures include those specified for ACTUAL RESULT and VIRTUAL RESULT clauses, the ENCODING/DECODING clause, the CHECK data base procedure clause, and the CALL clause for an input/output request involving an elementary item or a vector.

DATA CLASS

The characteristics of a data item are described in the PICTURE or TYPE clause in the schema, in the PICTURE and USAGE clauses in the COBOL and Query Update subschemas, and in the type statements in the FORTRAN subschema. The description of a data item determines its data class. CDCS supports eight Data Description Language (DDL) data classes at both the schema and subschema levels. The data classes, their respective codes, and the equivalent usage specification are given in table 5-1.

The data class that can be specified in the subschema depends upon the class of the data item in the schema. Table 5-2 shows the valid data conversions performed by CDCS. The conversion routines operate in a source-target mode. The data conversion rules that apply to the valid conversions marked in the matrix are given in appendix H. Appendix H also contains notes on conversions that do not produce conversion errors, but can cause alteration of data on a read and rewrite sequence of operations involving schema to subschema to schema data conversions.

TABLE 5-1. DATA CLASSES

Code	Data Class	COBOL USAGE Clause	Query Update USAGE Clause	FORTRAN 5 Type Statement
0	Display alphanumeric	DISPLAY	DISPLAY	CHARACTER
1	Display alphabetic	DISPLAY	DISPLAY	CHARACTER
3	Display integer	DISPLAY COMP	DISPLAY	None†
4	Display fixed point	DISPLAY COMP	DISPLAY COMP	None†
10	Coded binary integer	COMP-1 INDEX	COMP-1 INDEX LOGICAL	INTEGER LOGICAL BOOLEAN
13	Coded floating point normalized	COMP-2	COMP-2	REAL BOOLEAN
14	Coded double precision	None†	DOUBLE	DOUBLE PRECISION
15	Coded complex	None†	COMPLEX	COMPLEX

†No corresponding COBOL or FORTRAN type. Valid conversion is shown in table 5-2.

TABLE 5-2. VALID DATA CONVERSIONS

Source Data Class	Target Data Class							
	0	1	3	4	10	13	14	15
0	X	X	X					
1	X	X						
3	X		X	X	X	X	X	
4			X	X	X	X	X	
10			X	X	X	X	X	X
13			X	X	X	X	X	X
14			X	X	X	X	X	X
15					X	X	X	X

A data item specified as BOOLEAN in a FORTRAN 5 subschema is mapped as either a class 10 or class 13 item, depending on the class of the corresponding item in the schema.

Conversion to class 13, which employs normalized floating point binary representation, might incur loss of precision. The binary value cannot be guaranteed to be precise beyond the limits of the PICTURE clause or the dimensions of the item description. Class 13 is treated as a signed value in all cases.

The data class of an item described in the schema is determined by either a PICTURE clause or a TYPE clause. Table 5-3 lists each data class and its representation in the schema.

The PICTURE and USAGE clauses in the COBOL and Query Update subschema description of a data item and the type statements in the FORTRAN subschema description determine the subschema data classes. Table 5-4 lists each data class and its COBOL, Query Update, and FORTRAN subschema representations.

OMISSION OF DATA ITEMS

If data items described in the schema are omitted from the subschema description, the data items omitted are replaced on a write operation by null values before a call is issued to CYBER Record Manager Advanced Access Methods (AAM) to write the record. The actual null value placed in the omitted data item field depends on the data class of the data item, as shown in table 5-5.

TABLE 5-3. DATA CLASS REPRESENTATION IN THE SCHEMA

Data Class	Schema PICTURE Clause	Schema TYPE Clause	Internal Representation
0	None	CHARACTER	Display code, alphanumeric
	Character-string (A, X, 9; not all As or 9s)	None	
1	Character-string (all As)	None	Display code, alphabetic
3	Numeric-picture integer (9,T)	None	Display code numeric can have sign overpunch in last character position
4	Numeric-picture fixed point (9, V, T, P, .)	None	Display code numeric plus implicit or explicit decimal or scaling position
10	None	FIXED integer-1, integer-2 (where the integer value is 1 thru 18)	Binary integer
13	None	FLOAT integer-1 (where the integer value is 1 thru 14)	Signed, normalized floating point (60-bit)
14	None	FLOAT integer-1 (where the integer value is 15 thru 29)	Signed, normalized floating point (2 words)
15	None	COMPLEX	Floating point with real part and imaginary part (2 words)

TABLE 5-4. DATA CLASS REPRESENTATION IN THE SUBSCHEMA

COBOL Subschema			Query Update Subschema in CDCS Data Base Access Mode		FORTRAN 5 Subschema Type Statement	Internal Representation
Schema Data Class	PICTURE Clause	USAGE Clause	PICTURE Clause	USAGE Clause		
0	Alphanumeric (A X 9; not all As or 9s; mixed specification used as all Xs)	DISPLAY (or none)	Alphanumeric (A X 9; not all As or 9s; mixed specification used as all Xs)	DISPLAY (or none)	CHARACTER	Display code, alphanumeric
1	Alphabetic (A)	DISPLAY (or none)	Alphabetic (A)	DISPLAY (or none)	CHARACTER	Display code, alphabetic
3	Numeric (9 S)	DISPLAY COMP (or none)	Numeric (9 S and insertion and replacement characters)	DISPLAY COMP (or none)	None†	Display code numeric, $\leq$ 18 characters; trailing sign overpunch if S is specified

TABLE 5-4. DATA CLASS REPRESENTATION IN THE SUBSCHEMA (Contd)

COBOL Subschema			Query Update Subschema in CDCS Data Base Access Mode		FORTRAN 5 Subschema Type Statement	Internal Representation
Schema Data Class	PICTURE Clause	USAGE Clause	PICTURE Clause	USAGE Clause		
4	Numeric (9 S V P)	DISPLAY COMP (or none)	Numeric (9 S V P and insertion and replacement characters)	DISPLAY COMP (or none)	None†	Display code numeric, ≤ 18 characters; trailing sign overpunch if S is specified; implicit decimal or scaling position
10	Numeric (9 S V P)	COMP-1 INDEX††	Numeric (9 S V P and insertion and replacement characters)	COMP-1 INDEX†† LOGICAL	INTEGER LOGICAL BOOLEAN	Size < 15; binary integer
13	Numeric (9 S V P)	COMP-2	Numeric (9 S V P and insertion and replacement characters)	COMP-2	REAL BOOLEAN	Normalized floating point
14	None†	None†	Numeric (9 S V P and insertion and replacement characters)	DOUBLE	DOUBLE PRECISION	Normalized floating point (2 words)
15	None†	None†	Numeric (9 S V P and insertion and replacement characters)	COMPLEX	COMPLEX	Normalized floating point (2 consecutive words)

†No valid COBOL or FORTRAN type (whichever is applicable). Valid conversion is shown in table 5-2.

††No picture clause allowed.

TABLE 5-5. NULL VALUES FOR DATA CLASSES

Data Class	Null Value Stored
Display alphanumeric	Display code blanks
Display alphabetic	Display code blanks
Display integer	Display code zeros
Display fixed point	Display code zeros and optional decimal point
Coded binary integer	Binary zeros
Coded floating point normalized	Floating point zeros
Coded double precision	Floating point zeros
Coded complex	Floating point zeros

The facilities of CYBER Database Control System (CDCS) that provide for data structure and control of associated data are data base files, data base versions, relations, and constraints. The data accessed through CDCS resides in data base files. The data base version facility allows a schema to be associated with more than one group of data base files. The relation facility provides for retrieval of data from two or more files joined together logically in a relation. The constraint facility allows controls to be established and maintained during update operations on associated files or on associated items within a single file.

This section provides information about data base files, data base versions, relations, and constraints.

DATA BASE FILES

CYBER Record Manager (CRM) is the subsystem that provides the input/output interface between CDCS and the operating system routines that read and write files on hardware devices. The primary task of CRM is to reconcile logical records specified by the user with their physical representation on a particular hardware device. CRM is a generic term relating to the common products Basic Access Methods (BAM) and Advanced Access Methods (AAM). BAM is a file manager that processes sequential and word addressable file organizations. AAM is the DMS-170 file manager that processes indexed sequential, direct access, and actual key file organizations and supports the Multiple-Index Processor.

CRM supports five file organizations (three of which allow alternate key processing), four blocking types for sequential files, and eight record types. Every file opened is associated with a file information table (FIT) through which information about the file is communicated. The FIT holds details on file structure and processing and, in particular, contains descriptions of record and key size, record type, and blocking structure.

CDCS and the data base utilities use the services of CRM to process the following types of files:

- Conventional data files comprising a data base
- Log files
- Schema and subschema directories
- Master directory

The discussion in these paragraphs is limited to data files that comprise a data base. The paragraphs provide information on the data file organization and record types supported by CDCS and on the rules that govern data base files. For detailed information on CRM, refer to the CRM Advanced Access Methods reference manual.

CDCS AND CRM COMMUNICATION

Application programs access data files by issuing calls to CDCS, which in turn issues calls to CRM to handle the processing. Conventional input/output statements in COBOL are used to access data files and relations. These statements reference the files and records described in the COBOL subschema. In a FORTRAN program, FORTRAN Data Manipulation Language (DML) statements are used to access data files and relations. These statements, which are similar in syntax to FORTRAN statements, reference the files and records described in a FORTRAN subschema. In a Query Update program, Query Update directives can access data base files when a Query Update subschema for CDCS data base access is being used.

DATA BASE FILE DEFINITION

CDCS supports the three extended AAM file organizations: extended indexed sequential, extended direct access, and extended actual key. CDCS also supports multiple-index files for these organizations. Data base files must be permanent files on mass storage devices.

NOTE

Refer to appendix F for recommendations on the use of Advanced Access Methods.

Data base files must be defined in a way that allows them to be attached from the CDCS system control point. Under NOS, either the files must be defined with category PUBLIC (CT=PU in the DEFINE control statement) or file access must be permitted to the user number of the CDCS system control point job or the CDCS Batch Test Facility job. To permit file access to another user under NOS, the file owner can enter the PERMIT control statement and specify the user number of the other user and type of access to be allowed. For example by specifying DBCNTLX=W in a PERMIT control statement, the owner permits user DBCNTLX access to a file in write (W) mode.

CRM Record Types

CDCS supports seven CRM record types. Each schema area has a record type associated with it during schema compilation. CDCS associates the record type for the area with the file and uses that record type when interfacing with CRM.

NOTE

Refer to appendix F for recommendations on the use of CRM record types.

The following record types are supported:

F fixed length

Z zero byte

W control word

S system

T trailer count

D decimal character count

R record mark

CRM record types T, D, and R are less efficient than the others. For F, Z, and W type records, the record length (RL field of the FIT) is the length contained in the schema record description; RL defines a fixed length for each CRM record type. For T type records, the trailer item must have the same size and position in all schema record descriptions for one area. The data item that controls the number of occurrences of the trailer item must have the same size, position, and type in all schema record descriptions for an area.

File Control

The data items defined in the schema are organized into logical units called areas. The data control entry in the schema partly specifies certain file attributes for each area named in the schema. This entry defines the data items that represent the primary and alternate keys for extended AAM files, the collating sequence for the data in a named area, and the mechanism for distinguishing among multiple record types within an area.

The FILE control statement is used at schema compilation time to supplement the information in the data control entry of the schema. File organization, record type, and other pertinent information that is needed to describe fully the attributes of an area are specified in the FILE control statement.

An area is associated with an addressable storage unit (a permanent file) through information in the master directory. All the file attributes specified for an area are used by CDCS for processing the associated file.

Refer to section 2 for information on the data control entry and on the FILE control statement. Refer to the CYBER Record Manager Advanced Access Methods reference manual for detailed information on the FILE control statement.

Multiple-Index Files

One or more multiple-index files can exist as part of a data base. Alternate key processing for extended AAM files is supported. Alternate keys defined in the data control entry name the data items to be used for alternate key access. The characteristics of these data items are described in the schema and subschema record descriptions.

When an application program specifies processing via an alternate key, the COBOL compiler, the FORTRAN DML preprocessor, or Query Update set up appropriate calls to CDCS. During program execution, CDCS receives the calls from the program and, in turn, issues calls to CRM to perform the input/output operations. An alternate key retrieval returns a data record. When a record that has alternate keys is modified, the alternate key indexes are updated automatically by CRM.

Multiple Record Descriptions

An area can have more than one record description (DDL record type) in the schema and in a COBOL or Query Update subschema; only one record description per area is permitted in the FORTRAN subschema. All records in a given area must have the same CRM record type. The RECORD CODE clause in the schema data control entry defines the mechanism for determining the DDL record type; that is, which record description is to be used for a given data record. The record in the data base area must be matched with the appropriate record descriptions in the schema and in a COBOL or Query Update subschema so that record mapping, data validation, and other similar operations can be performed correctly. A control field or a data base procedure specifies the mechanism in this clause. If an area has multiple record descriptions, it cannot be joined in a relation.

NOTE

Refer to appendix F for recommendations on the use of multiple record descriptions.

The following rules apply to multiple record descriptions:

Key fields must have the same type, size, and position in all schema record descriptions for a given area.

A given COBOL or Query Update subschema area can contain all or some of the record types defined in the schema area; however, if an application program references a record type not specified in the subschema in use, COBOL or Query Update issues a diagnostic.

If a control field determines the DDL record type, it must be an elementary item of the same type, size, and position in all record descriptions for a given area.

Key Definitions

Primary and alternate keys are specified in the data control entry in the schema. The schema item declared as the primary key must have a corresponding subschema item. Duplicate primary keys are not allowed. Primary keys for extended indexed sequential and extended direct access files accessed through CDCS cannot exceed 240 characters in length; primary keys for extended actual key files cannot exceed 8 characters in length. Alternate keys for all extended AAM files cannot exceed 240 characters.

The schema and subschema compilation listings produced by the DDL compiler show the data items defined as keys. The COBOL compiler, the FORTRAN DML preprocessor, and Query Update verify the correctness of any key item specified in an input/output statement.

Embedded Key Items

Embedded keys are key items that are contained in the record. Keys for all files accessed through CDCS must be embedded. All embedded keys must be in the same position in all schema record descriptions for a given area. The following rules apply to embedded key items declared in the schema:

An embedded key item cannot be the item referenced in an ACTUAL RESULT or a VIRTUAL RESULT clause.

An embedded key item cannot be nested within more than one level of a repeating group; that is, only one subscript can apply to the item.

Data items used as key items in COBOL READ and START statements and in Query Update directives cannot be defined within a nested hierarchy of repeating items (only one subscript can apply). The subschema item used as a key in an input/output statement must be in one of the following two categories:

The data item must correspond to (1) a schema item or group of items that is declared as a key item or (2) a schema item or group of items that satisfies the requirements for a declared key item and has the same position in the record as a declared key item.

The data item must be within the span of a REDEFINES clause where the originally redefined item has the same position in the subschema record as the data item, corresponds to the data item described in (1), and is represented identically in the schema and subschema.

Major Key Processing

The leading portion of a primary or alternate key can be used for retrieval in COBOL, Query Update, and FORTRAN application programs. Using the leading portion of a key is called major key processing. To use major key processing in application programs, a concatenated key must be defined in the schema.

The subschema referenced by the COBOL or Query Update application program must define the concatenated key as a group item that includes only the data items included in the schema description of the concatenated key. The application program must use, as the major key, the data item or items that are the leading portion of the concatenated key. If more than one item is desired for the major key, these items must be in a group immediately subordinate to the key group. The major key is then referenced by the group name. For example if ITEMA, ITEMB, and ITEMC are items of a concatenated key, defined at the 01 level in the schema, the subschema groups might appear as follows:

```
03  CONCAT-KEY.
    05  MAJOR-KEY.
        07  ITEMA          PICTURE X(5).
        07  ITEMB          PICTURE X(3).
    05  ITEMC              PICTURE X(2).
```

where CONCAT-KEY is the group name that identifies the concatenated key as defined in the schema. MAJOR-KEY is the group name identifying the major key as defined in the subschema.

To perform major key processing, the COBOL programmer issues a START with the KEY EQUALS phrase and then follows with a READ NEXT AT END statement. Query Update automatically performs major key processing when the data item that is the leading portion of the concatenated key is specified as the subject of an IF directive or as the subject of a conditional statement.

The subschema referenced by the FORTRAN application program must define the concatenated key by defining all of the constituent data items of the key (or the subschema defined aliases of the these data items) in the same order as they are defined in the subschema. The concatenated key, the name of which appears in the subschema output (Figure 6-1), is then available for use.

To perform major key processing in a FORTRAN application program, the FORTRAN DML START statement is used. The major key item name (leading item of the concatenated key) or the list of names (leading contiguous items of the concatenated key) is specified in the statement's KEY clause.

In a FORTRAN program, the following rules and restrictions apply. The concatenated key name can only be specified in the READ and START statements. The major key can only be specified in the START statement. A major key composed of only one item must be the leading item of the concatenated key. A major key composed of more than one item must be contigous items beginning with the leading item of the concatenated key. The list can include as many of the items that make up the concatenated key as desired.

Figure 6-1 illustrates the definition of a concatenated key, called CAT-KEY, and its inclusion in a subschema. This is an example of a situation in which an employee works on several projects and information is retrieved by employee identification. The employee identification cannot be the primary key because CRM requires that each record have a unique primary key. By defining the primary key as the concatenation of employee and project identification (data items EMPLOYEE-ID and PROJECT-ID), the requirement of CRM is satisfied. Information can be retrieved by using EMPLOYEE-ID as the major key.

Use of a primary key for major key processing is available only for extended indexed sequential files. The use of an alternate key for major key processing is available for all three AAM file organizations supported by CDCS.

Schema

```
SCHEMA NAME IS FACTORY.
    .
    .
    .
AREA NAME IS WORK.
RECORD NAME IS WORK-REC WITHIN WORK.
    01  EMPLOYEE-ID  PICTURE "X(6)".
    01  PROJECT-ID   PICTURE "X(8)".
    01  HOURS        PICTURE "999V99T".
DATA CONTROL.
    .
    .
    .
AREA NAME IS WORK
KEY ID IS CAT-KEY <  EMPLOYEE-ID OF WORK-REC
                     PROJECT-ID OF WORK-REC >
KEY IS ALTERNATE PROJECT-ID OF WORK-REC
    DUPLICATES ARE ALLOWED.
```

Query Update/COBOL Subschema

```
TITLE DIVISION.
SS  ASUB WITHIN FACTORY.
REALM DIVISION.
RD  WORK ...
RECORD DIVISION.
01  WORK-REC.
    03 CAT-KEY.
        05  EMPLOYEE-ID     PICTURE X(6).
        05  PROJECT-ID      PICTURE X(8).
    03  HOURS               PICTURE S999V99.
    .
    .
    .
```

FORTRAN Subschema

```
SUBSCHEMA FTCONCT, SCHEMA=FACTORY
ALIAS (ITEM)  EMPID=EMPLOYEE-ID
ALIAS (ITEM)  PROJID=PROJECT-ID

REALM WORK

RECORD, WORK-REC

CHARACTER EMPID *6
CHARACTER PROJID *8
REAL HOURS
```

FORTRAN Subschema Output

```
PRIMARY KEY CAT-KEY(EMPID,PROJID) FOR AREA WORK
```

Figure 6-1. Concatenated Key Definition

Key Processing with Multiple Record Types

If record mapping is required for a particular record type, the subschema key items and the schema key items might not occupy corresponding positions because of reordering of items in the subschema. The key must be in the same position in each subschema record type. All keys must be declared in the first schema record description. A record description corresponding to the first schema record description must be included in the subschema; only those items in this subschema record description that correspond to the declared key items can be used as key items in input/output statements. All items to be used as keys must have invariant position across all subschema record descriptions.

Subschema areas, records, and items are said to correspond to schema areas, records, and items if they have the same names (or have been assigned aliases for corresponding schema names in the Alias Division of the COBOL or Query Update subschema).

Figure 6-2 illustrates a schema with two record descriptions, PRODUCT-REC and PART-REC, for an area called PRODUCTS. The primary and alternate key definitions are shown in the data control entry. The primary key is STOCK-NUM in PRODUCT-REC; the alternate key is PROD-NAME in PRODUCT-REC. PART-NUM or PRODUCT in PART-REC can be used as a key item in an input/output statement because each of these items corresponds exactly to a defined key.

```
SCHEMA NAME IS PRODBASE.
    .
    .
    .
AREA NAME IS PRODUCTS.
RECORD NAME IS PRODUCT-REC WITHIN PRODUCTS.
    01  STOCK-NUM     PICTURE "X(6)".
    01  PROD-NANE     PICTURE "X(10)".
    01  DESC-A        PICTURE "X(20)".
    01  PRODUCT-LOC   PICTURE "X(4)".
    01  RC1           PICTURE "9".
RECORD NAME IS PART-REC WITHIN PRODUCTS.
    01  PART-NUM      PICTURE "X(6)".
    01  PRODUCT       PICTURE "X(10)".
    01  PART-DESC     PICTURE "X(20)".
    01  PART-LOC      PICTURE "X(4)".
    01  RC2           PICTURE "9".
DATA CONTROL.
    .
    .
    .
AREA NAME IS PRODUCTS
KEY IS STOCK-NUM
KEY IS ALTERNATE PROD-NAME
RECORD CODE IS BY RC1
    VALUE FOR PRODUCT-REC IS 1,
    VALUE FOR PART-REC IS 2.
```

Figure 6-2. Key Definitions for Areas With Multiple Record Types

DATA BASE VERSIONS

The data base versions feature of CDCS allows the same schema and associated subschemas to be used with more than one group of files. Each group of files is called a data base version. During execution processing, CDCS can control concurrent processing on files in all data base versions.

Data base versions, if defined, interact with other facilities of CDCS. They affect logging, recovery, the operator interface, relations, and constraints. Refer to the descriptions of these facilities for information about the impact versions could have.

DEFINING DATA BASE VERSIONS

Data base versions are defined in the master directory. If data base versions are not specified in the master directory, one version (called MASTER) is assumed. If alternate data base versions are defined, a version called MASTER must be defined along with the other versions.

In the master directory, the first data base version defined must be version MASTER. Within version MASTER, each area is associated with a permanent file.

After version MASTER is defined, subsequent versions are defined. Within the definition of each subsequent version, each area is associated with a permanent file. The permanent file associated with an area can be either the same file associated with the area in version MASTER or can be a different permanent file. If the permanent file is the same file as in version MASTER, that file can also be associated with the same area in any version. If the permanent file is a different file, that permanent file cannot be used in any other version. The DBMSTRD utility ensures that only permanent files specified in version MASTER are shared among other versions.

ACCESS CONTROL LOCKS

Access control locks are specified for an area. When data base versions are defined, the access control locks specified for an area apply to that area in all data base versions. An application program, therefore, must specify the same access control keys to obtain access to a particular area in any version.

EXAMPLES OF ENVIRONMENTS FOR DATA BASE VERSIONS

In an operational environment, there are several situations where associating a schema with several groups of files is useful: for example, a testing situation and a situation in which a business has a number of branches.

Testing Situation

When new application programs are being tested, it is helpful to use the same schemas, subschemas and application programs that are to be used when the application is put into production. For obvious reasons, the data files actually used for production cannot be used. A group of files used for testing is needed. However, a test version of the data base version might not need to have its own copy of every permanent file; read-only files might be shared by the production version and test version.

An example of a master directory which defines a master version and a test version of a university data base is shown in figure 6-3. All areas defined in the schema (PROFESSOR, COURSE, STUDENT, CURRICULUM, and COSTS) are associated with permanent files in version MASTER. In the test version (called TESTDB), the areas PROFESSOR and COURSE are associated with the same files as used in version MASTER but different files are associated with the remaining areas.

```
SCHEMA NAME IS UNIVERSITY
        .
        .
        .
VERSION NAME IS MASTER
AREA NAME IS PROFESSOR
    PFN IS "PROFESS" UN IS "CDCS23"
    LOG BEFORE IMAGE BLOCKS
        AFTER IMAGE RECORDS
      INDEX FILE ASSIGNED
     PFN "PNDX" UN IS "CDCS23".
 AREA NAME IS COURSE
    PFN IS "COURSE" UN IS "CDCS23"
    LOG BEFORE IMAGE BLOCKS
        AFTER IMAGE RECORDS
      INDEX FILE ASSIGNED
     PFN "CRSNDX" UN IS "CDCS23".
AREA NAME IS STUDENT
    PFN IS "STUDENT" UN IS "CDCS23"
    LOG BEFORE IMAGE BLOCKS
        AFTER IMAGE RECORDS
      INDEX FILE ASSIGNED
     PFN "SNDX" UN IS "CDCS23".
AREA NAME IS CURRICULUM
    PFN IS "CURRICU" UN IS "CDCS23"
    LOG BEFORE IMAGE BLOCKS
        AFTER IMAGE RECORDS
      INDEX FILE ASSIGNED
     PFN "CRNDX" UN IS "CDCS23".
AREA NAME IS COSTS
    PFN IS "COSTS" UN IS "CDCS23".

VERSION NAME IS TESTDB
AREA NAME IS PROFESSOR
      SAME AS MASTER.
AREA NAME IS COURSE
      SAME AS MASTER.
AREA NAME IS STUDENT
    PFN IS "T1STD" UN IS "CDCS23"
      INDEX FILE ASSIGNED
      PFN IS "T1STDX" UN IS "CDCS23".
AREA NAME IS CURRICULUM
    PFN IS "T1CRC" UN IS "CDCS23"
      INDEX FILE ASSIGNED
      PFN IS "T1CRCX" UN IS "CDCS23".
AREA NAME IS COSTS
    PFN IS "T1CST" UN IS "CDCS23".
        .
        .
        .
```

Figure 6-3. Example of Version Definitions in Master Directory Input

Using the version facility to define the production and test version of the data base has the following advantages for processing:

- A single executing copy of CDCS is in the computer.
- Different logging options can be selected for the production and testing versions.
- Some files can be shared between the production and testing versions.

Branch Situation

In a branch situation, using the data base version facility can be helpful. Assume that a business consists of a number of branches or regional divisions with each branch operating relatively independently but in the same way as the other branches. Each branch can use the same schema, subschemas, and possibly a set of applications because the same kind of data and data manipulations are required to do business. Each branch, however, needs its own group of permanent files.

If a data base version is defined for each branch, concurrent processing could occur with each branch using its own permanent files. As in the testing situation, some of the files might be shared by several branches.

RELATIONS

The relation facility of CYBER Database Control System (CDCS) allows users to retrieve data from two or more files joined together logically. The logical structure created by the joining of several files is termed a relation.

A relation is defined through the facilities of the Data Description Language (DDL). A relation is described in the schema. Common data item fields that exist in areas are used to join the areas in a relation. The common items joining the areas are called join terms or identifiers and are specified in the schema. During processing, CDCS associates the areas and join terms with data base files and data items, respectively.

Record occurrences from files joined in a relation can be selectively retrieved. The subschema defines the qualification criteria for extracting selected records from the data base files joined in a relation.

CDCS controls the manipulation of the files joined in a relation and controls the selection of record occurrences through the use of information from the schema and subschema directories. The COBOL or FORTRAN application programmer codes a single READ statement specifying the relation that is to be read. CDCS processes the statement and returns a record occurrence from each file in the relation to the user's work area for each file. The Query Update programmer transmits a single query in which the data items specified are involved in a relation; Query Update automatically determines the relation and requests that CDCS perform relation processing.

The following paragraphs describe how files are joined and traversed, the restriction of records through record qualification, and the retrieval of records performed by CDCS during relation processing. This information is of interest to the data administrator because the order in which files are joined in a relation and qualification criteria affect access time used by application programs in retrieving records.

When designing relations, the data administrator should consider that data base relations should be designed with the least number of connections between the relations. When one file is linked to a file in one relation and is also joined to another file in another relation, the update operations on the common file must be monitored in order to preserve the meaning of each of the relations.

JOINING FILES

In the CDCS environment, one file can be accessed at a time, or (if two or more files can be related in some manner and defined as a relation) more than one file can be accessed at a time. Input/output operations for single or multiple access are performed by CYBER Record Manager (CRM) through calls from CDCS. CRM can access records in a file by using a record key that has been defined as the key into that file. A unique primary key value is associated with each record in the file. The multiple-index capability of CRM provides for the specification of alternate record keys in addition to the primary key. With alternate keys defined for a file, access to the records in that file is possible through the primary key or through any of the alternate keys.

When a data base is designed and created, it usually contains several files with common data item descriptions. The information in one file is related to the information in another file. When an application program processes information in one file, it might use a data item obtained from that file to access a second file. Information in the second file, in turn, might be used to access still a third file. A logical relationship can then be identified for these three files that are interrelated. For example, the EMPLOYEES file and the PRODUCTS file of an application are related by a common data item, PROJECT-NO. A PRODUCT record type and an EMPLOYEE record type are linked by PROJECT-NO, an alternate key in the EMPLOYEES file.

The PRODUCT record is read using the primary key PRODUCT-NO, and the EMPLOYEE record is accessed by alternate key using the value of PROJECT-NO obtained from the PRODUCT record. The application programmer can then use PROJECT-NO as a key to retrieve all the employee record occurrences for that project number. This relationship between project number in one record and project number in another record results in the identification of a number of employees. The diagram of this relationship in figure 6-4 shows it as a one-to-many relationship.

Without the CDCS relational data base facility, the application programmer would have to write the procedures to read the PRODUCT record initially and then, using the alternate key value obtained from the PRODUCT record occurrence, read the EMPLOYEE records.

The programming becomes more difficult when three records are participating in a relationship such as the one just described. For example, if a CONTRACT record were introduced into the above relationship and many records were retrieved for each alternate key value, the programming to handle this relationship would be more complicated than for the relationship joining only two files.

The relationship joining three files is illustrated in figure 6-5. The defined structure of the relationship begins with a contract number. The CONTRACTS file is accessed first. CONTRACT-NO from the CONTRACT record occurrence is used to access the PRODUCTS file via its alternate key CONTRACT-NO. A number of products exist for a given contract; that is, there are duplicate values in the contract number field.

Each PRODUCT record occurrence retrieved for a given contract supplies the project number value that can in turn be used to retrieve a number of EMPLOYEE record occurrences. PROJECT-NO is an alternate key in the EMPLOYEE record and, again, duplicate values exist.

Hierarchical Tree Structure

The dependency of the record occurrences within the files joined by a directed relationship can be schematically represented as a hierarchical tree structure like the one shown in figure 6-6. The root of the tree contains the record occurrences of the CONTRACTS file. The CONTRACT record occurrence branches to record occurrences in the PRODUCTS file. Likewise, the PRODUCT record occurrences

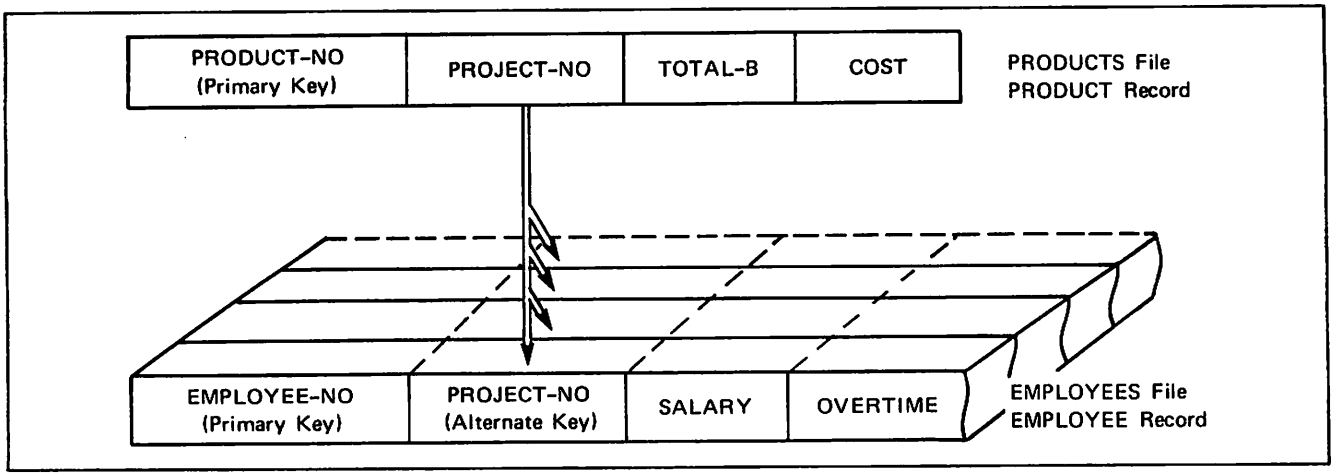


Figure 6-4. Two-File Relationship Example

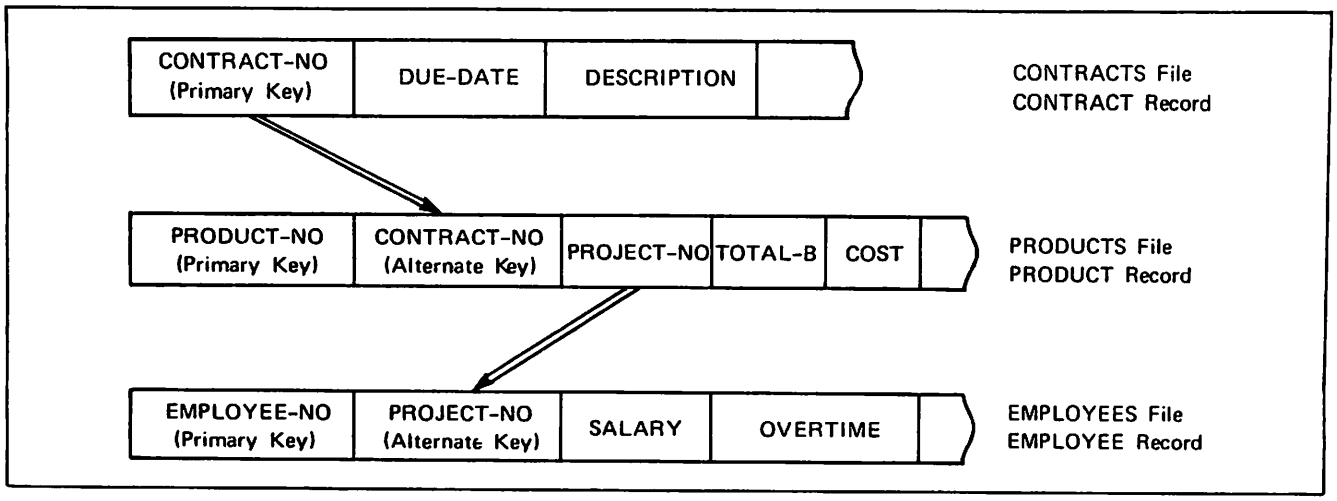


Figure 6-5. Three-File Relationship Example

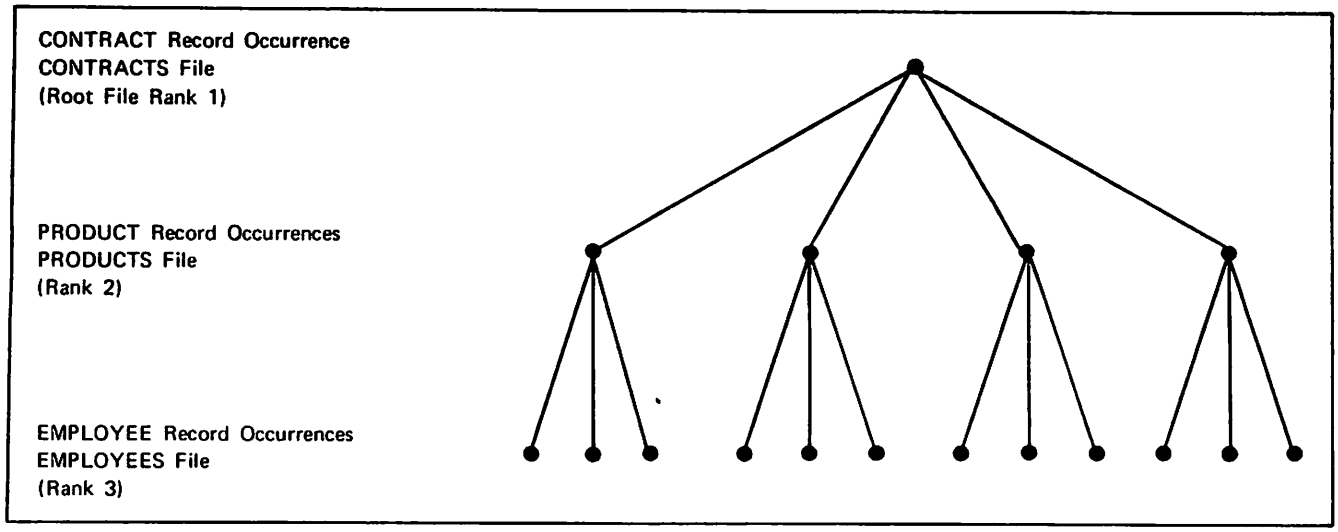


Figure 6-6. Tree Structure of CONTRACTS-PRODUCTS-EMPLOYEES Relationship

branch to record occurrences in the EMPLOYEE file. Extraction of record occurrences from each of the three files in the relationship is performed automatically for the application program through CDCS if a relation joining the CONTRACTS, PRODUCTS, and EMPLOYEES files has been defined in the schema and included in the subschema used by the application program.

Ranks of a Relation

A relation entry in the schema contains the relation name and a JOIN clause specifying the identifiers that are used to link the files in the relation. The files linked together in a relation are assigned a rank in the relation.

The identifiers in the JOIN clause specify a data item and an area to be involved in the relation. For detailed information on the relation entry, refer to section 2. For example, the relation entry that assigns the name CONTRACTS-PRODUCTS-EMPLOYEES to the relation shown in figure 6-5 and defines the relation is as follows:

```
RELATION NAME IS CONTRACTS-PRODUCTS-EMPLOYEES
JOIN WHERE
CONTRACT-NO OF CONTRACT EQ CONTRACT-NO OF
    PRODUCT
PROJECT-NO OF PRODUCT EQ PROJECT-NO OF EMPLOYEE
```

The order in which an area is included in a relation determines the rank in the relation of the associated file. The file associated with the first area included in the relation is assigned rank 1; the second rank 2, and so on, with newly assigned rank being an increment of 1 of the last rank assigned. In the example, the files are assigned ranks in the relation as follows: CONTRACTS, rank 1; PRODUCTS, rank 2; and EMPLOYEES, rank 3.

The lower rank a file has in a relation, the higher position the file has in the hierarchical tree structure. For example, the file of lowest rank (the root file, which is assigned rank 1) is pictured at the top of the tree structure.

Parent/Child Relationship

The joining of files in a directed relationship results in a dependency condition between record occurrences linked in the files. A record occurrence in the root file is termed the parent record occurrence for all related record occurrences (each termed a child record occurrence) in the file linked to it. A child record occurrence can also be a parent record occurrence when a subsequent file is joined in the relation and related record occurrences exist in that subsequent file.

In the CONTRACTS-PRODUCTS-EMPLOYEES relation, a record occurrence in CONTRACTS file is the parent record occurrence for related record occurrences in the PRODUCTS file. Likewise, a record occurrence in the PRODUCTS file is a parent of several record occurrences in the EMPLOYEES file. The record occurrences in the PRODUCTS files are the children of a record occurrence in the CONTRACTS file. Likewise, the record occurrences in the EMPLOYEES file are children of a record occurrence in the PRODUCTS file.

A parent record occurrence is one that has another record occurrence at the next numerically higher rank in the relation. A child record occurrence is one that has another record occurrence at a numerically lower rank in the relation.

RECORD QUALIFICATION

Record qualification is the method used to restrict which records are to be returned to the user. Record qualification is implemented by specifying criteria that must be satisfied by a record occurrence. Qualification is specified in the subschema for records in any file in the relationship. Use of qualification can greatly limit the number of record occurrences returned to the user's work area. For a better understanding of this statement, the CONTRACTS-PRODUCTS-EMPLOYEES relationship is reexamined.

A tree structure of record occurrences in the CONTRACTS-PRODUCTS-EMPLOYEES relation is illustrated in figure 6-7. A contract C1 is composed of four products: P1, P2, P3, and P4. Each product is developed by a number of employees and each employee works on only one product. For example, employees E1 and E2 develop product P1; employees E3, E4, E5, and E6 develop product P2. Proceeding from left to right following each path up the tree structure, twelve record occurrences of the relationship can be identified, namely C1P1E1, C1P1E2, C1P2E3, C1P2E4, and so on.

Qualification criteria are specified in the RESTRICT clause in the COBOL and Query Update subschemas and in the RESTRICT statement in the FORTRAN subschema; the criteria are used by CDCS to determine whether a record occurrence qualifies to be returned to the user's work area as part of the relation occurrence. For example, to retrieve the records of those employees working on product P4, qualification can be specified for the PRODUCT record type to indicate that the record occurrences should be restricted to those in which the value of the product number field is P4. CDCS reads and discards record occurrences P1, P2, and P3 before retrieving record occurrence P4 and its child record occurrences. Since P1, P2, and P3 do not qualify, no input/output operations are performed to retrieve their child record occurrences.

Without the facility to qualify records, each of the record occurrences to the left of the required ones (including P1, P2, and P3 and all their child record occurrences) would have to be extracted and returned to the user's work area (in many cases, after CDCS record mapping is performed). The user would then have to determine if the record occurrence was the one required or not.

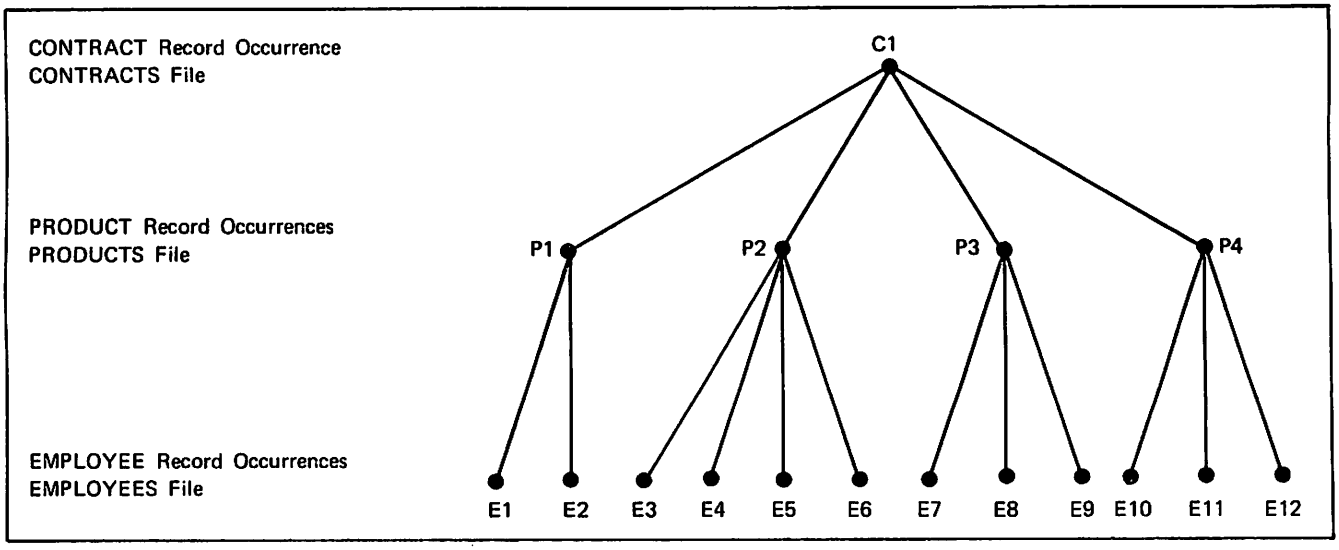


Figure 6-7. Complex Tree Structure for CONTRACTS-PRODUCTS-EMPLOYEES Relationship

Record qualification in the COBOL and Query Update subschemas is defined in the Relation Division. This division contains the relation name and the RESTRICT clause, which specifies record qualification. Refer to section 3 for further details on the Relation Division.

Record qualification in the FORTRAN subschema is defined in a relation definition. A relation definition contains the RELATION statement (which specifies the name of the relation) and the RESTRICT statement (which specifies record qualification). Refer to section 4 for further details on the relation definition.

CDCS RELATION PROCESSING

A relation defines a directed path joining several files. Join terms, called identifiers, are data items that link the files. The identifiers must be inspected by CDCS to traverse the relation and return a record occurrence from each file in the relation to the user's work area. The JOIN clause in the schema is used to specify the identifiers. Input/output statements in COBOL and in the FORTRAN Data Manipulation Language (DML) can be used to position and read a relation. Query Update performs relation processing if it is required to satisfy a query and a relation specified in the subschema.

The following paragraphs describe positioning and reading a relation. The statements described are those used in COBOL and FORTRAN application programs. For detailed information on relation processing using the application programming languages, see the CDCS 2 Application Programming reference manual.

Source and Target Identifiers

When CDCS traverses files in a relation, CDCS inspects the join terms that have been specified by the JOIN clause in the schema. The join term in one file (which contains parent record occurrences) is used to obtain a record occurrence from the file which is joined to it (which contains child record occurrences). The join term in the first file is termed the source identifier. The value of the source identifier must be identical to the value of the target identifier in the second file so that the record occurrences can be joined. The target identifier determines the retrieval method that CDCS must employ to read a record occurrence from the target file.

If the target identifier is a data item that was defined as a primary or an alternate key in the schema data control entry, retrieval is most efficient since the target file can be accessed randomly. If the target identifier is an item that is defined as the leftmost portion of a primary or alternate key, the target file can also be accessed randomly.

If the target identifier is a data item that is not a key, retrieval is inefficient. CDCS rewinds the file and searches it sequentially until a match occurs for the identifiers. The file must be searched until a match occurs on the contents of the join fields, or until the end-of-information on the file is reached.

Relation Positioning

A relation can be positioned for subsequent sequential read processing through the COBOL and FORTRAN DML relation START statements. In an application program, a START statement that specifies a relation name causes the root file of the relation to be positioned at the record whose key satisfies the condition specified in the KEY phrase of the START statement. The key specified in the START statement is established as the key of reference. The key of reference is always a key in the root file. After the root file has been positioned and the key of reference has been established, a sequential read of the relation returns to the user the record occurrence from the root file that satisfies the condition in the START statement. It also returns a record occurrence from each file at a higher rank in the relation. A sequential read loop following a relation start operation references the root file according to the key established by the operation.

Relation Read

The user specifies a relation read by using one of the READ statement formats. If the user specifies a key in the READ statement, the root file in the relation is accessed randomly according to the primary or alternate key in that file. If no key is specified, however, the root file is read sequentially by the current key of reference. Other files in the relation are accessed by CDCS depending on the identifiers specified for the files. If the target identifier is an alternate key for a file containing child record occurrences, that file is read randomly by alternate key. The diagram in figure 6-8 illustrates a group of record occurrences for a relation having three files.

If the relation is read sequentially, the order in which record occurrences are returned to the user is A, B, and C first, providing this set of record occurrences meets qualification criteria. Successive sequential reads of this relation by CDCS would return the remaining record occurrences that are child record occurrences of B (namely record occurrences D and E). FILEA and FILEB are not read again since CDCS expects the user's work areas to still contain record occurrences A and B. To retrieve the next record occurrence, CDCS returns to FILEB in the relation and retrieves record occurrence F, if F meets qualification criteria, and in turn retrieves the children of record occurrence F (record occurrences G and H).

The diagram in figure 6-9 identifies the record occurrences that are contained in the user's work areas after each read of the relation, assuming the record occurrences shown meet qualification criteria. The user's work areas contain record occurrences with record descriptions that are defined in the subschema used by the application program.

The relation is first read randomly if the user has specified a key in the READ statement. Successive sequential reads of the relation are specified by the user with the READ NEXT AT END statement in COBOL or the READ statement without the KEY option in FORTRAN; these are translated to retrievals on the appropriate files in the relation. Within the sequential read loop, the user should not try to reposition the files involved.

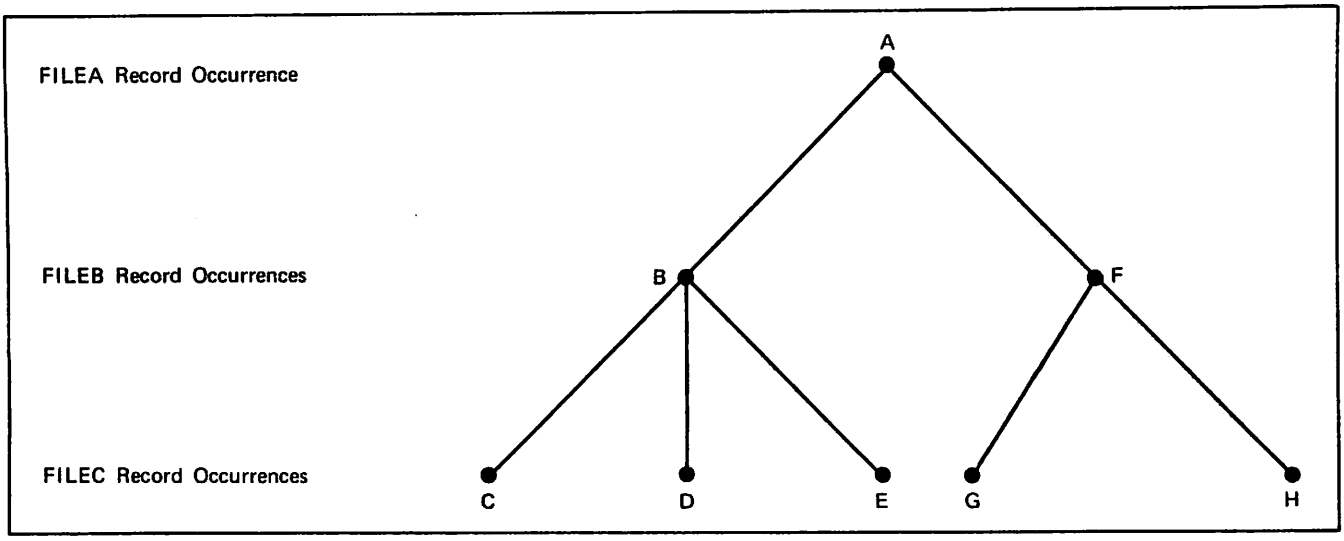


Figure 6-8. Record Occurrences for Three Related Files

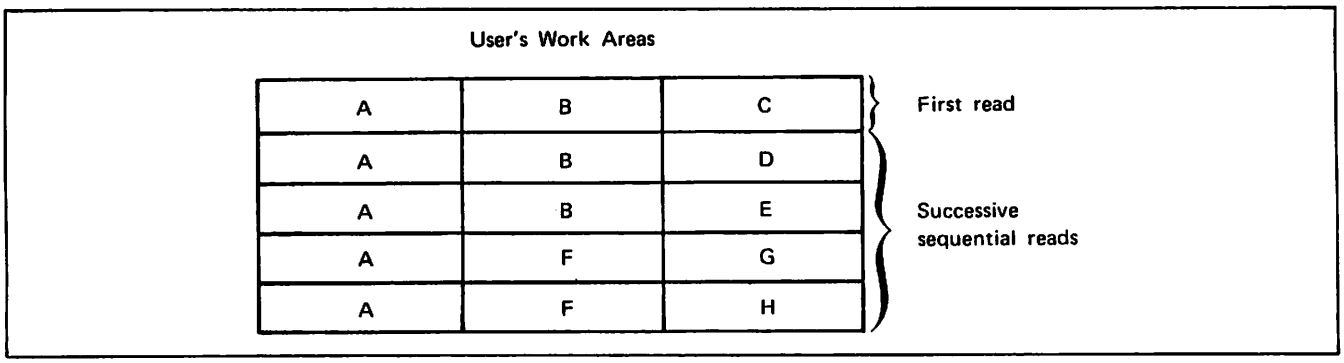


Figure 6-9. Record Occurrences in User's Work Areas After Reading

Order of Record Retrieval

The order in which record occurrences for a file are returned to the user work area as part of a relation occurrence depends on the target identifier for that file. When a relation is read sequentially, a group of child record occurrences for one parent record occurrence is ordered as follows:

If the target identifier is an alternate key with duplicate values, record occurrences are ordered according to the order in which alternate keys are stored in the index file. The order depends on what has been specified in the DUPLICATES option of the KEY clause in the schema data control entry.

If the target identifier is not a key, record occurrences are ordered in the sequence in which records containing a specific value for the target identifier are stored in the file.

If the target identifier is the major portion of a primary or alternate key, the record occurrences are ordered according to the portion of the key field that is not the target identifier.

If the target identifier is omitted from the subschema record description, it is not returned to the user's work area and the ordering of child record occurrences may depend on a value that is inaccessible by the user.

Informative Conditions

CDCS detects the occurrence of the following conditions:

Null record occurrence on a file

Control break on a file

A FORTRAN or COBOL application program can check for these conditions and determine the lowest ranked file on which the condition occurs if appropriate receiving fields are defined and used in the application program. Refer to the CDCS 2 Application Programming reference manual for detailed information on status checking.

Null Record Occurrence

If a parent record occurrence does not qualify for retrieval, any child record occurrences automatically do not qualify. In the example shown in figure 6-9, if B does not qualify, record occurrences C, D, and E automatically do not qualify; none of these record occurrences would be returned to the user's work areas for FILEB and FILEC. If, however, B does qualify, but C, D, and E do not, one null record occurrence is returned to the user's work area for FILEC to indicate that no child record occurrences of B qualify. The null record occurrence consists of a display code right bracket (]) in each character position; the number of character positions filled depends on the subschema description of the record type.

In general, a null child record occurrence status is returned to the user if all the children of a parent record occurrence that qualified do not qualify, or if no child record occurrences exist. Some examples of null record occurrences returned to the user's work area are illustrated in figure 6-10.

When null record occurrences are returned for all files in a relation, another READ statement must be executed to obtain the next set of record occurrences for the relation.

Control Break

The control break condition on a file signifies that a new record occurrence was read for the parent file in the relation. Control break status, however, is returned for the realm of the child. If a file in a relation has control break status after a sequential READ statement has been issued for the relation, the record occurrence read for this file is a child record occurrence for a new parent record occurrence.

Control break status is not set for a file if a null record occurrence status must be set.

Example of Null Record and Control Break Conditions

Record occurrences and control break conditions are shown in figure 6-11. The example shows the record occurrences A through M in files FILE1, FILE2, and FILE3 and the control break and null occurrence conditions that result from retrieval of each relation occurrence. No qualification criteria have been specified on any of the records for the files.

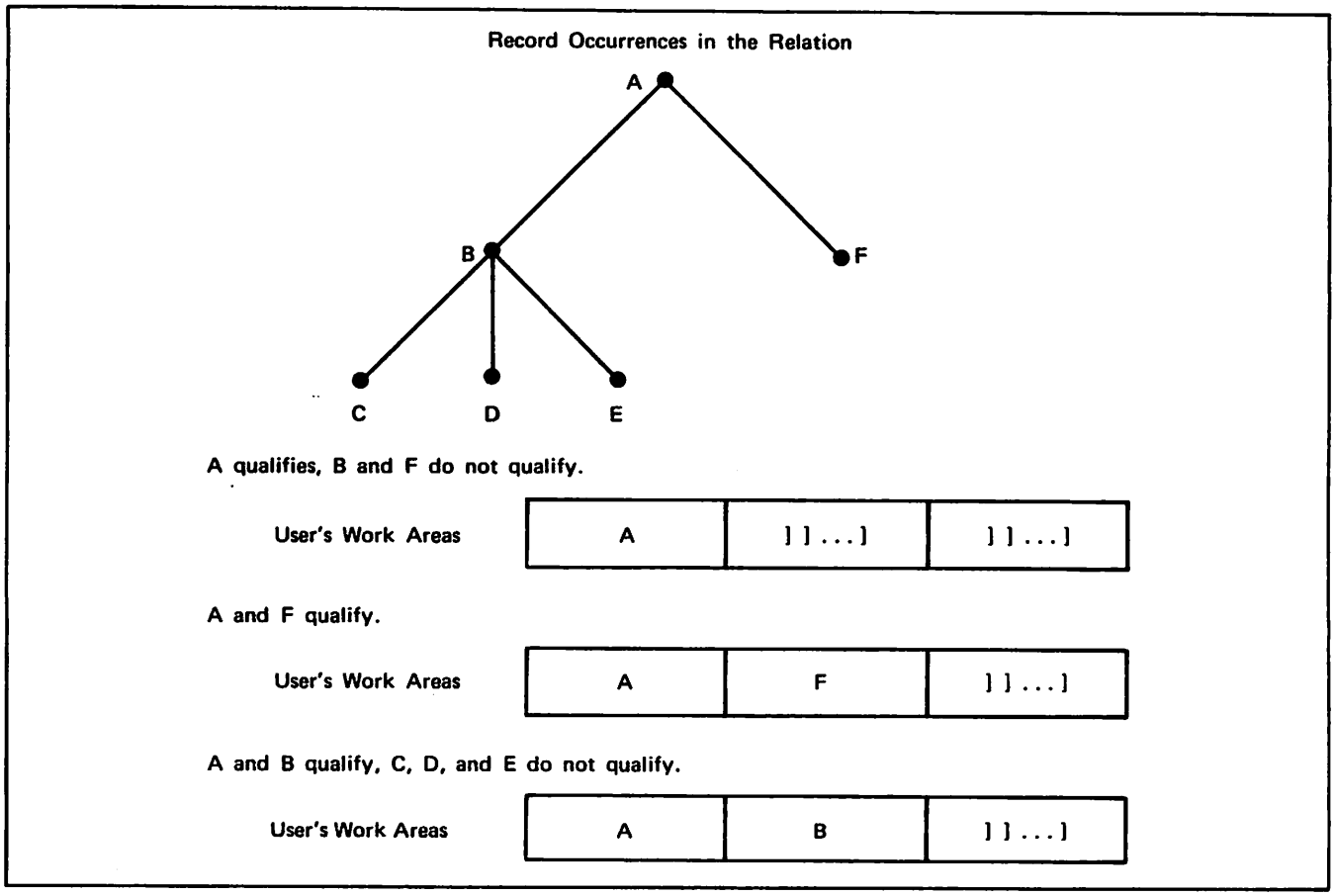
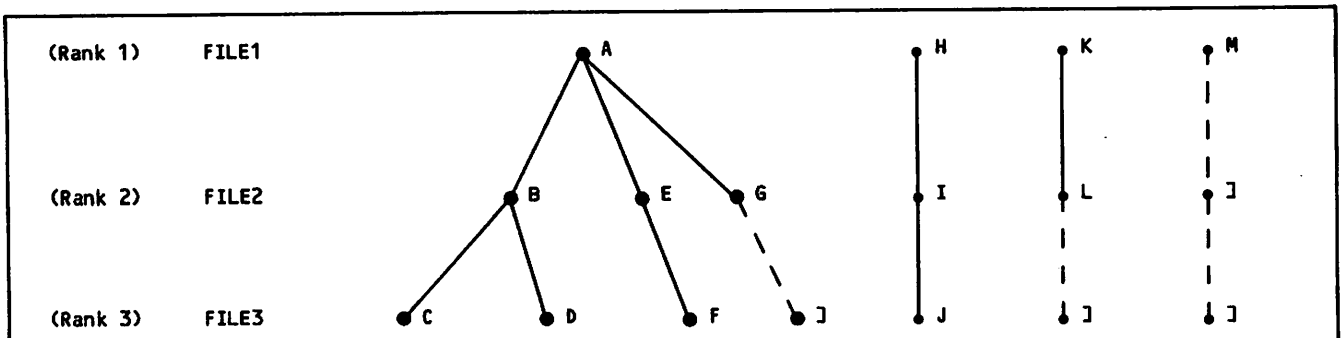


Figure 6-10. Null Record Occurrence Examples



Relation Occurrences			Data Base Status Block: Null Occurrence Rank	Data Base Status Block: Control Break Rank	Condition - File
A	B	C	0	2	Control break - FILE2 Control break - FILE3
A	B	D	0	0	
A	E	F	0	3	Control break - FILE3
A	G	null	3	0	Null occurrence - FILE3
H	I	J	0	2	Control break - FILE3 Control break - FILE2
K	L	null	3	2	Null occurrence - FILE3 Control break - FILE2
M	null	null	2	0	Null occurrence - FILE2

Figure 6-11. Example of Null Occurrence and Control Break Conditions

Effect of Versions on Relation Retrieval

A relation can join different groups of permanent files when data base versions exist. The files joined in a relation depend on the files associated with the version being used. Some files are used by several versions while other files are used by only one version. This means that relational reads can yield different results depending on the version used.

For example, figure 6-12 illustrates the CONTRACTS-PRODUCTS-EMPLOYEES relation being used with two data base versions, MASTER and UNIT1. Contrasted with previous examples of this three-level relation, the names CONTRACTS, PRODUCTS, and EMPLOYEES now refer only to the areas defined in the schema and not to the permanent files; the areas are associated with permanent files with different names.

In the example, versions MASTER and UNIT1 share two files: CMSTR (associated with the area CONTRACTS) and EMSTR (associated with the area EMPLOYEES). Each version uses a separate file associated with the area PRODUCTS.

A relational read using version MASTER could result in the following record occurrences being returned:

Record 1 of permanent file CMSTR
Record 1 of permanent file PMSTR
Record 1 of permanent file EMSTR

The same relational read using version UNIT1 could result in the following record occurrences being returned:

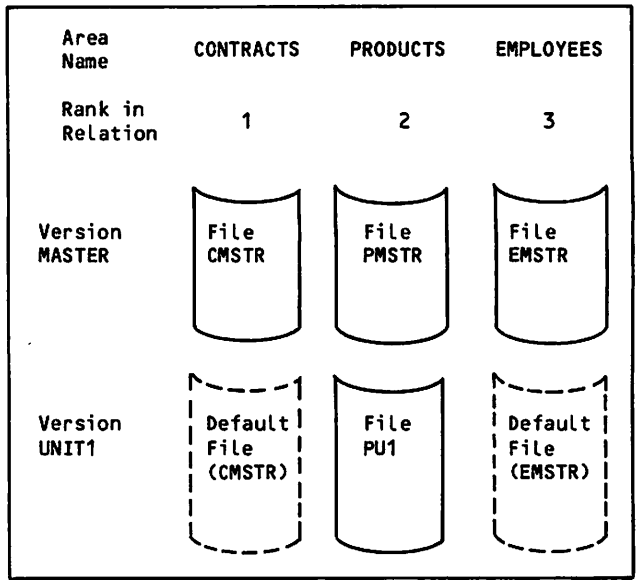


Figure 6-12. Example of Files Joined by a Relation and Grouped by Version

Record 1 of permanent file CMSTR
Record 1 of permanent file PU1
Record 3 of permanent file EMSTR

There are no restrictions on the use of relations with data base versions; however, the possibility of retrieving different records in relational reads that use different data base versions should be recognized.

COLLATING SEQUENCES

Files to be accessed through CDCS relation processing must be generated with one of the following collating sequences specified: COBOL, DISPLAY, or ASCII. The collating sequence is specified in the schema data control entry.

NOTE

Refer to appendix F for recommendations on the use of collating sequences.

If a file is built by using a user collating sequence that assigns more than one character to the same position in the collating sequence, the order of record retrieval on a relation read is unpredictable if the file is accessed by a data item that is not a key or by a major portion of a primary or an alternate key.

CONSTRAINTS

The constraint facility of CDCS allows controls to be established and maintained on update operations involving two logically associated files or two data items within the same file. A constraint is a means for imposing an integrity control on associated files or items within a single file.

Constraints are established for the purpose of protecting the integrity of data in a data base during update operations by application programs. When a program attempts to update a data base file that is involved in a constraint, CDCS evaluates the effect of the operation on the elements involved in the constraint before allowing the update to be performed. If the operation would violate the constraint, CDCS does not allow it; otherwise, CDCS allows the update to be executed normally.

An integrity constraint is defined through the facilities of DDL in the schema. Data items within two files or within a single file are used to establish a logically dependent condition. Files in a two-file constraint are associated by virtue of data items with common characteristics.

These paragraphs describe how a constraint is established in the schema and how CDCS monitors and enforces constraints when an application program is executed. Additionally, the restrictions that constraint definition imposes on data base versions are listed.

DEFINING CONSTRAINTS

In a data base environment, logical associations often exist among several of the files comprising the data base. The information in one file is associated with the information contained in another file through data item descriptions with identical characteristics. A typical example of this type of association might consist of department and employee files within a personnel data base that have in common a data item indicating department number. When an association such as this exists, the data administrator might find it desirable to impose a constraint on update operations involving the associated files so that the association is not altered and inconsistent data is not introduced into the data base as a result of an update.

A constraint can be defined for two associated files or for two associated items within a single file. In either case, the constraint entry that defines the constraint in the schema establishes a dependent condition for the elements involved.

Two-File Constraints

When a constraint is defined for two logically associated files, the data administrator establishes a dependent condition between record types in the files based on data items common to both records. In a constraint, one record type is defined as the dominant record and the other is defined as the dependent record. A dominant record occurrence corresponds to a dependent record occurrence if both records contain the same value for the item that connects them.

At execution time, CDCS uses the dependency condition established by the constraint in evaluating whether a particular update operation involving a dominant or dependent record should be allowed to take place. The general rule used to evaluate an update operation is that no dependent record can exist without a corresponding dominant record. Specifically, a dominant record occurrence can be deleted from a data base file only if there are no corresponding dependent record occurrences. A dependent record occurrence can be added to a data base file only if a corresponding dominant record occurrence exists. Modification of either dominant or dependent record occurrences must also follow these rules, if the data item that connects them is modified. (Processing of constraints by CDCS is described in detail later in this section.)

The diagram in figure 6-13 illustrates the dependency condition. A DEPARTMENTS file and an EMPLOYEES file in a data base both contain a data item DEPT-NO that indicates a department number within each file. A dependency condition is established between two record types by including a constraint entry such as the one in the figure in the schema source program. Refer to section 2 for details on the syntax of a constraint entry.

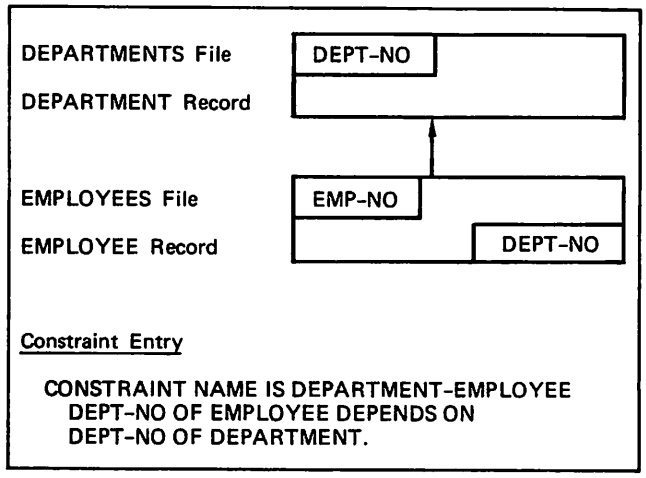


Figure 6-13. Two-File Constraint Example

The DEPARTMENT record type is assigned the dominant role and the EMPLOYEE record type is assigned the dependent role. The arrow in the figure indicates the dependency condition.

The data items used to associate files in a constraint need not have the same name, but they must have identical characteristics; that is, they must have identical TYPE or PICTURE clause specifications in the schema record descriptions. However, the position of the data items within the respective records need not be identical. The data item used to establish a dominant record type in a constraint must be a primary key in the file or an alternate key with no duplicate values allowed. The data item used to establish a dependent record type in a constraint must also be a primary key in its file or an alternate key that can have duplicate values. In the example in figure 6-13, the item DEPT-NO is a primary key in the DEPARTMENTS file (dominant role) and an alternate key in the EMPLOYEES file (dependent role).

A constraint is defined in the schema in a constraint entry. A constraint entry assigns a name to the constraint and specifies the dominant and dependent roles for record types within two data base files. Only two files can be associated in a single constraint entry. However, a file can be involved in more than one constraint. Thus constraints on associated files can include any number of levels. For instance, if a file named FILEA is logically associated with a file called FILEB by virtue of the occurrence of a data item named ITEMX in both files, a constraint could be imposed involving these files. ITEMX in FILEA and ITEMX in FILEB must have identical characteristics; they need not have the same name. Another constraint could be imposed on FILEB and a third file, FILEC, if these two files contain a data item, ITEMY, with the same characteristics. A third constraint could be defined for files FILEA and FILEC based on a connecting data item, ITEMZ.

Figure 6-14 shows the associations among FILEA, FILEB, and FILEC, along with the constraint entries in the schema that establish the dependency conditions. In the first constraint entry, record type A of FILEA is assigned the role of dominant record and record type B of FILEB is assigned the role of dependent record. In the second constraint entry, record type B is assigned the additional role of dominant record and record type C of FILEC is assigned the role of dependent record. Record type C also participates as a dependent record in the third constraint, with A being the dominant record.

When designing a series of constraints such as those described above, the data administrator should keep in mind two important factors. The first is that processing of constraints by CDCS at execution time might introduce a significant performance overhead when the data base is accessed. The data administrator should consider the trade-off of integrity protection versus performance overhead in establishing constraints. If constraints are to be specified, the number and complexity of the logical dependencies among files should be minimized as much as possible.

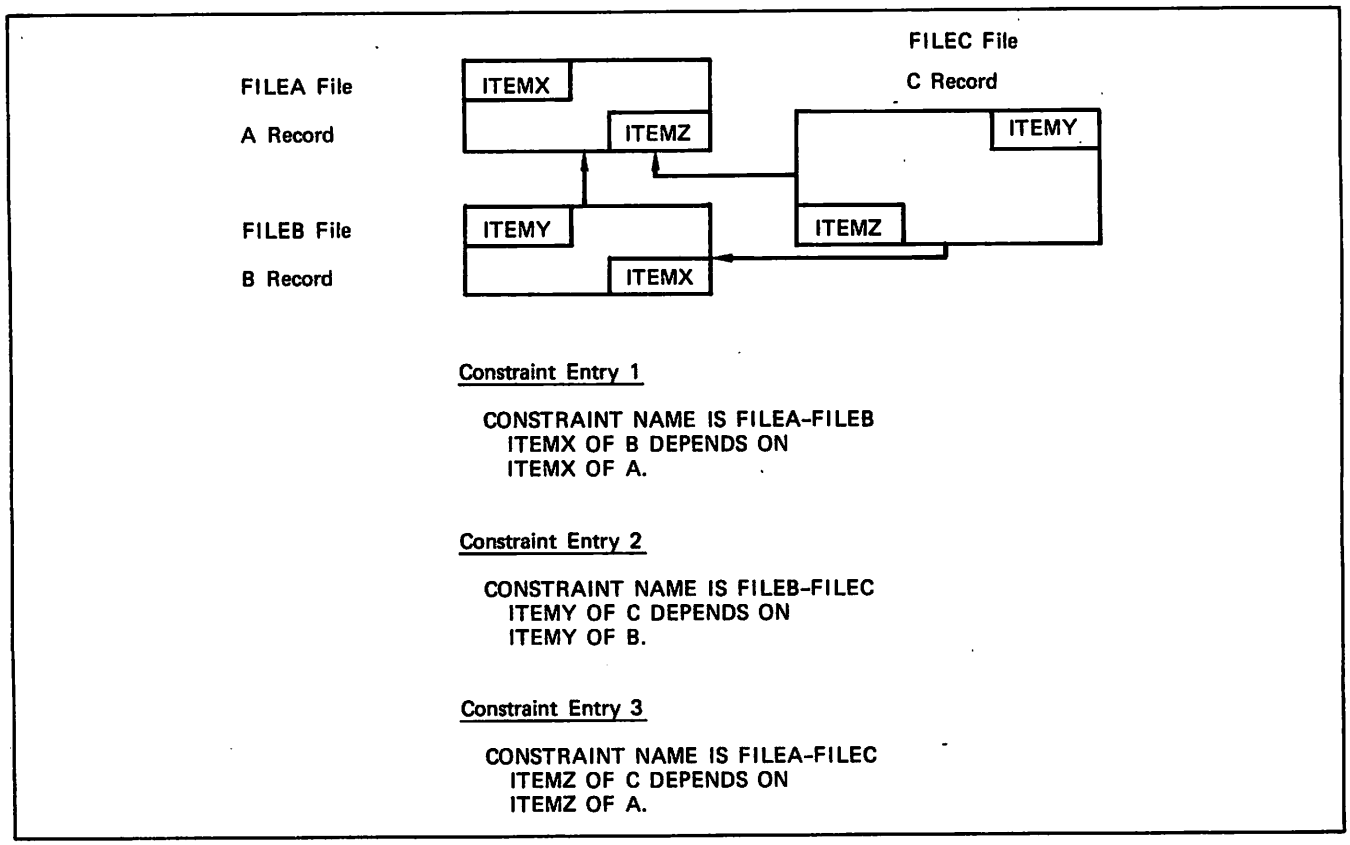


Figure 6-14. Three-File Constraint Example

The second factor to be considered when defining constraints is that a series of constraints cannot form a cycle. A cycle is a directed path that starts at a record designated as a dependent record in one constraint and ends at the same record which is designated as a dominant record in another constraint, thus forming a closed loop.

Modifications to the two previous constraint examples in this section clarify the concept of a cycle. If an additional constraint were specified for the files in figure 6-13 that would reverse the order of the dependency condition, these constraints would form a cycle. The directed path between the files would form a closed loop, with the DEPARTMENT record as a dominant record in the original constraint and as a dependent record in the additional one. Likewise, the EMPLOYEE record would participate in a dependent role in the original constraint and in a dominant role in the additional one.

The series of constraints in figure 6-14 would form a cycle if the dependency condition of constraint FILEA-FILEC were reversed: if record C were assigned the dominant role and record A the dependent role.

Single-File Constraints

A constraint can be defined for two logically associated items within a single file. A typical example is an employee file in which each record contains among other items an employee number and a manager number, where the manager number conforms to the structure of the employee number. A constraint might be imposed on these items that reflects the requirement that no employee record can be stored in the file if an employee record for the manager of the employee does not exist. This constraint also implies that an employee record for a manager cannot be deleted from the file if one or more employee records with the corresponding manager number exist.

In a single-file constraint, one data item in a record type is assigned the dominant role, and another data item in the same record type is assigned the dependent role. At execution time, CDCS uses the constraint in evaluating whether a particular update operation on a record occurrence in a data base file should be allowed to take place.

The diagram in figure 6-15 illustrates the concept of a single-file constraint. In an EMPLOYEES file, the EMPLOYEE record type contains the data item EMP-NO to indicate an employee number and the data item MNGR-NO to indicate the employee number of the manager to whom the employee reports. A constraint is defined for these items by including a constraint entry such as the one in the figure in the schema source program.

The item MNGR-NO is dependent on the item EMP-NO in the constraint EMPLOYEE-MANAGER. Two occurrences of the EMPLOYEE record are included in the diagram to indicate that it is actually the manager's employee number in an EMPLOYEE record occurrence for an employee (the MNGR-NO item) that is dependent on the manager's employee number in an EMPLOYEE record occurrence for a manager (the EMP-NO item). (The arrow in the diagram points from the dependent member to the dominant member.)

The data items involved in a single-file constraint must follow the same rules as those in a two-file constraint. The items must have identical characteristics. In addition, the dominant data item must be a primary key or an alternate key with no duplicate values allowed. The dependent item must be a primary key or an alternate key that can have duplicate values. In the example in figure 6-15, the item EMP-NO is a primary key and the item MNGR-NO is an alternate key with duplicate values.

A constraint entry in the schema can associate only two data items within a record type; however, a data item can be involved in more than one constraint. If a series of constraints is defined, the data administrator must ensure that the constraints do not form a closed loop, or cycle. If an additional constraint were specified for the data items in figure 6-15 which would reverse the order of the dependency, these constraints would form a cycle.

CDCS TWO-FILE CONSTRAINT PROCESSING

CDCS enforces the integrity constraints specified in the schema during update processing by application programs. CDCS does not permit a write, delete, or rewrite operation to be performed on

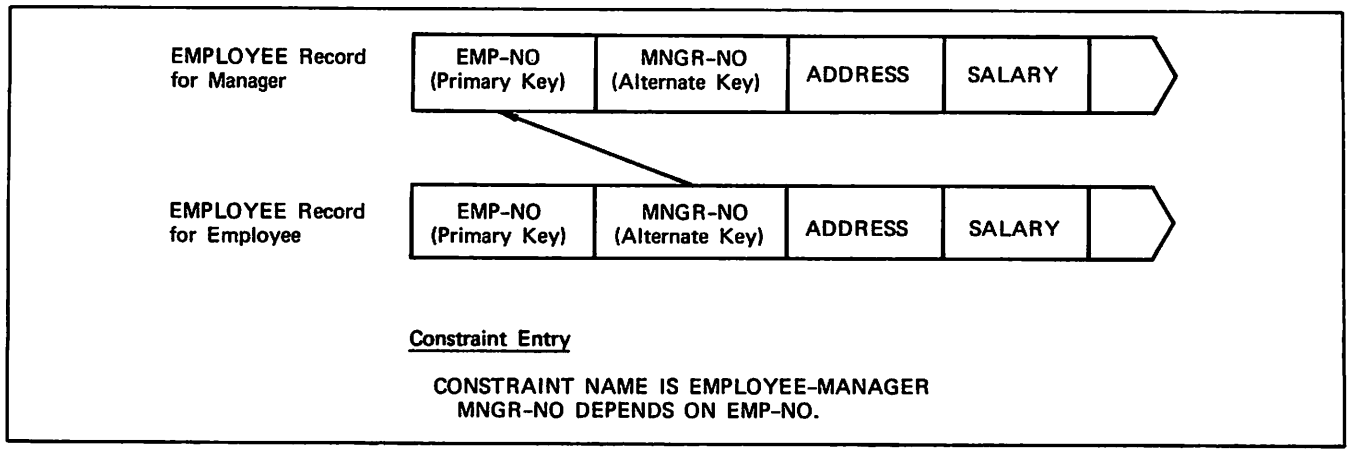


Figure 6-15. Single-File Constraint Example

records in data base files if the operation would alter the dependency condition established by the constraint and introduce inconsistent data into the data base.

The subschema being used by an application program might specify only one of the two files involved in a constraint in the schema. In order to process the constraint, CDCS must attach both files. The file that is not specified in the subschema is accessed through a read operation without checking of privacy locks and without most data base procedure calls. The only data base procedures that are executed for this file are the key hashing procedure for extended direct access files and the procedures for compression and decompression of data.

When a program attempts a write, delete, or rewrite that would violate a constraint, CDCS diagnoses the violation as a nonfatal error; the operation is terminated, but the program is permitted to continue processing. A CDCS error code is returned in the data base status block of the application program if the status block has been defined in the program. A diagnostic message is written to the user error file and to the CDCS output file indicating the name of the violated constraint, the operation that was attempted, and the name of the record being updated.

Guidelines for File Creation

Files involved in two-file constraints must be created in a particular order to ensure the integrity of the files. A file containing the dominant record type must be created (that is, opened for creation, provided with records, and closed) before the creation of any file that contains a dependent record type.

In the three-file constraint example (figure 6-14), FILEA must be created first. Record type A of FILEA is the dominant record for both record type B of FILEB and record type C of FILEC and must be established before any dependent record occurrences can be processed. FILEB must be the second file created. FILEB contains record type B, which has the dependent role for record type A of FILEA and has the dominant role for record type C of FILEC. The last file to be created must be FILEC, which contains record type C. Because FILEC is created last, CDCS has available the dominant record occurrences of record types A and B to process the dependent record occurrences of record type C.

Controlling Insertion Operations

Insertions into a data base file involved in a constraint are permitted by CDCS only if the dependency condition is not altered by the operation. A dependent record occurrence can be added to a file only if a corresponding dominant record occurrence exists. Dominant and dependent record occurrences correspond when the values of the data items used to associate the files to which the records belong are equal. An attempt to store a dependent record violates the integrity of the data base if there is no corresponding dominant record.

In the example of the DEPARTMENTS and EMPLOYEES files, in which the department number DEPT-NO has been used to assign the DEPARTMENT record the dominant role and the EMPLOYEE record the dependent role, a new EMPLOYEE record can be written to the EMPLOYEES file only if a DEPARTMENT record with the same department number exists in the DEPARTMENTS file. If there is no corresponding DEPARTMENT record, CDCS rejects the write request from the application program.

Controlling Deletion Operations

CDCS is responsible for determining whether or not an application program is allowed to delete a record from a file involved in a constraint. The determination depends on the role of the record being deleted and the effect of the operation on the dependent constraint condition. A dominant record occurrence can be deleted from a file only if there are no corresponding dependent record occurrences. CDCS disallows any attempt to delete a dominant record in a constraint when one or more corresponding dependent records exist.

For example, no DEPARTMENT record (the dominant record) can be deleted from the DEPARTMENTS file if any EMPLOYEE record (the dependent record) in the EMPLOYEES file has a department number matching that of the record to be deleted. If one or more corresponding EMPLOYEE records do exist, CDCS does not permit the DEPARTMENT record to be deleted.

Controlling Modification Operations

Modification of a record occurrence in a file on which a constraint has been imposed is restricted according to the rules for insertion and deletion, when the value of the data item that associates the files in the constraint is to be changed. A rewrite request from a user program on a file involved in a constraint must not introduce an inconsistent value for the data item. Modification of the associating item in a dominant record is rejected by CDCS if there is a dependent record with an item value equal to the current value of the item to be modified. Similarly, modification of the associating item in a dependent record is rejected if the new value is one that does not exist in a dominant record.

To modify the associating data item in a dominant record and to introduce the new value in the dependent records, the application programmer must perform a sequence of update operations. The operations performed depend on whether the data item is a primary key or an alternate key. If the item is a primary key, the following operations must be performed in the order shown:

1. Write the dominant record with the new value in the associating data item.
2. Read a dependent record; change the value of the associating data item to the new value contained in the dominant record; rewrite the dependent record. (Perform this step for each additional dependent record.)
3. Delete the dominant record with the old value.

If the data item to be modified in a dominant record is an alternate key, the following operations must be performed in the order given:

1. Write each dependent record containing the old value of the item to a temporary file.

2. Delete each dependent record containing the old value of the item from the data base file.

3. Read the dominant record; change the value of the data item to the new value; rewrite the dominant record.

4. Read a dependent record from the temporary file; change the value of the data item to the new value contained in the dominant record; write the dependent record to the data base file. (Perform this step for each additional dependent record.)

The DEPARTMENTS and EMPLOYEES files can be used once again to aid in understanding the concept of controlling file modifications through constraints. A rewrite operation cannot be performed on a DEPARTMENT record (the dominant record) if the operation would change the value of the item containing the department number, DEPT-NO, which is the primary key, and if one or more EMPLOYEE records (the dependent record) within the EMPLOYEES file have the same department number as the DEPARTMENT record to be modified.

A rewrite operation is not permitted on an EMPLOYEE record (the dependent record) if the updated value of the department number does not already exist in a DEPARTMENT record (the dominant record).

CDCS SINGLE-FILE CONSTRAINT PROCESSING

CDCS enforces a constraint placed on data items within a single data base file when an application program attempts to update the file. CDCS does not allow an update operation to be performed if the operation would violate the constraint and introduce inconsistent data into the data base.

Guidelines for File Creation

To create a file on which a single-file constraint has been imposed, the file must be created with record occurrences of dominant records. The dominant records in a single-file constraint are those in which the dominant item and dependent item have the same value.

This situation occurs in the single-file constraint example (figure 6-15) for the employee who has no manager. The record for this employee must have the same value for both EMP-NO and MNGR-NO.

For creating a file that is controlled by a single-file constraint, the following operations must be performed in the order shown:

1. Write the dominant records to the new file.

2. Close the file.

3. Reopen the file for input/output and add dependent records.

Controlling Insertion Operations

A write operation on a file on which a single-file constraint has been imposed is permitted by CDCS only if the condition specified in the constraint is satisfied by the operation. A record is added to a file based on the value of the dependent item in the record to be added and the values of the dominant item in records already in the file. If the value of the dependent item in the record to be written equals the value of the dominant item for one or more records present in the file, the new record is written to the file.

In the example of the single-file constraint EMPLOYEE-MANAGER (figure 6-15), the item EMP-NO in the EMPLOYEE record is defined as the dominant item of the constraint and the item MNGR-NO in the EMPLOYEE record is defined as the dependent item. A new EMPLOYEE record for an employee can be added to the EMPLOYEES file only if the value of the MNGR-NO item for the new record matches the value of the EMP-NO item in a manager's EMPLOYEE record in the file. In other words, a new EMPLOYEE record can be written only if an EMPLOYEE record for the employee's manager exists. If there is no EMPLOYEE record for the manager, CDCS rejects the write request from the application program and issues a nonfatal error message.

Controlling Deletion Operations

A delete operation on a file on which a single-file constraint has been imposed is evaluated by CDCS before the operation is allowed to take place. The value of the dominant item in the record to be deleted and the value of the dependent item in the remaining records in the file are used in determining whether or not a record can be deleted from a file. If one or more records exist in the file with a value for the dependent item that matches the value of the dominant item in the record to be deleted, the delete request is rejected. A record can only be deleted if there are no records in the file with dependent item values equal to the dominant item value of the record to be deleted.

For example, an EMPLOYEE record cannot be deleted from the EMPLOYEES file if any EMPLOYEE record has a value for the MNGR-NO item (the dependent item) that matches the EMP-NO item (the dominant item) of the record to be deleted. In general, an EMPLOYEE record for a manager can be deleted only if there are no EMPLOYEE records for the employees of that manager. If one or more EMPLOYEE records with the manager's employee number do exist, the EMPLOYEE record for the manager cannot be deleted. CDCS rejects the delete request and issues a nonfatal diagnostic.

Controlling Modification Operations

A rewrite operation in which one or both of the data items involved in a constraint are changed generally follows the rules for two-file constraints. For example, the employee number (EMP-NO) of the EMPLOYEE record for a manager cannot be changed unless the dependent records of the employees reporting to that manager are first deleted.

RESTRICTIONS FOR DATA BASE VERSIONS

For areas associated in two-file constraints, CDCS imposes restrictions on the assignment of permanent files to areas when data dase versions exist. CDCS enforces the restrictions when the DBMSTRD utility is run.

The restriction is imposed to control constraint processing when permanent files can be shared. Permanent files defined for version MASTER can be shared with other versions; permanent files defined for non-MASTER versions cannot be shared and, therefore, are for the exclusive use of that version.

The restrictions apply only to permanent file assignments in non-MASTER versions. The permanent files assigned to areas involved in a constraint must comply to the following rules:

- If one area involved in the constraint is assigned a permanent file that is the same file assigned for version MASTER, the other area involved in the constraint must also be assigned a permanent file that is the same file assigned for version MASTER.

- If one area involved in the constraint is assigned an exclusive permanent file, the other area involved in the constraint must also be assigned an exclusive permanent file.

For example, a situation in which a constraint restricts the files that can be included in a data base version is shown in figure 6-16. Areas A and B are involved in a constraint. Versions UNIT1 and UNIT3 are valid. Version UNIT2 is invalid because one file is exclusive and one file is shared.

If version UNIT2 were included in the source input for the master directory, the DBMSTRD utility would diagnose the error and no master directory would be created.

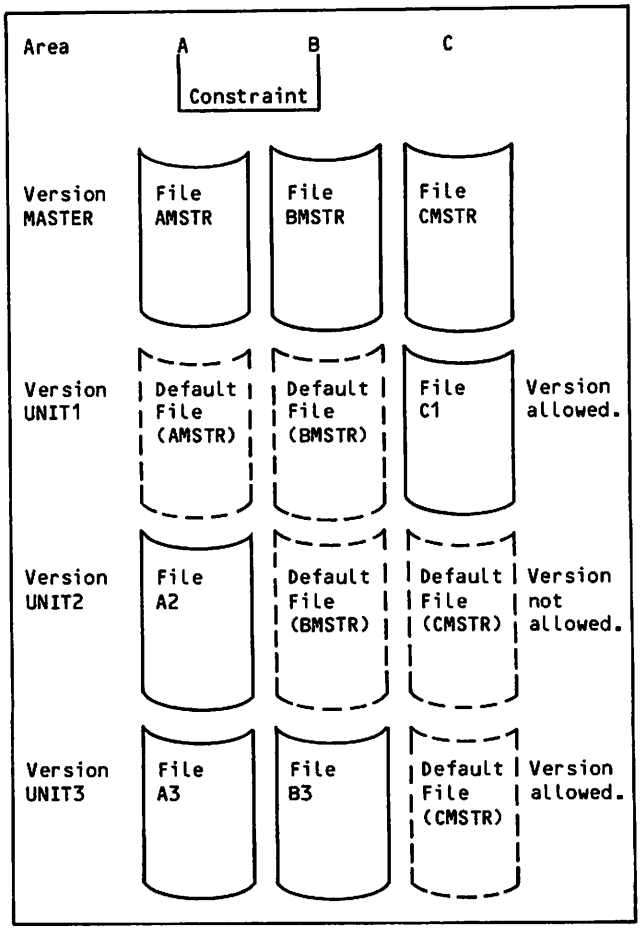


Figure 6-16. Example of Constraint Restrictions on Data Base Versions

DATA BASE PROCEDURES 7

Data base procedures are special subprograms written by the data administrator and called at execution time by CYBER Database Control System (CDCS) to perform any of the following functions:

Encoding and decoding of specified data items

Validation of data item values

Calculation of actual or virtual data item values

Compression and decompression of data

Additional processing on creation, retrieval, or update of data base records

Error condition handling

Privacy checking

Miscellaneous auxiliary functions

When data base procedures are implemented for encoding or decoding of data items, data conversions not supported by CDCS can be provided, or data can be enciphered for security purposes. The second function noted in the list, validity checking, ensures that valid data item values are stored in the data base or that a data item value falls within a range permitted for the item.

When a data base procedure is employed to calculate the value of a virtual data item, the value of the item is always determined by the data base procedure. The appropriate value is delivered to the working storage area of an application program but is not actually stored in the data base. For an actual item, the data base procedure is called to create and update the item; the computed value is actually stored in the data base.

Data base procedures can be employed for the process of reducing the amount of storage space required for a record in a data base file (compression) and for the process of restoring a compressed record to its original state (decompression). If a data base procedure is to be used for compression and decompression of data in a file, CDCS loads the procedure from the data base procedure library and passes the address of the procedure to CYBER Record Manager Advanced Access Methods (AAM). The procedure is called, as needed, by AAM.

Data base procedures can also be included for privacy checking or to perform other tasks not directly available as part of the CDCS services. In summary, the use of data base procedures can provide a well-defined method of tailoring the CDCS system to meet the needs of the particular installation.

DESCRIBING DATA BASE PROCEDURE USE

The use of data base procedures is defined through the facilities of the Data Description Language (DDL). The data base procedure name, the time at which the procedure is to be called, and the conditions governing the order of execution of data base procedures with respect to a CDCS function are specified via CALL clauses included in the schema definition. The following cases are covered by the CALL clause:

An area is opened or closed.

A specific function is performed on a record.

A specific function is performed on a data item.

An error occurs during the processing of any of the functions just mentioned.

Data base procedures can also be specified in the ENCODING/DECODING, RESULT, and CHECK clauses of the data description entry. The COMPRESSION and DECOMPRESSION clauses in the schema data control entry specify the names of data base procedures to be used for compression and decompression of records in a data base file. The RECORD CODE clause in the data control entry can contain the name of a data base procedure to be used to determine the DDL record type of a record belonging to an area that has multiple record types within it.

CDCS supports data base procedures for privacy checking. Privacy procedures are most useful for controlling access based on a program's job identification; CDCS does not control access in this manner. A privacy data base procedure uses a privacy key supplied by a program and the program's job name to determine whether or not the program should be allowed access to a particular area or areas. The program must supply a privacy key via the COBOL USE FOR ACCESS CONTROL statement, the FORTRAN DML PRIVACY statement, or the Query Update ACCESS directive in order for the procedure to be called, even if the key is a dummy value not actually used in the decision to allow access.

The PROCEDURE LIBRARY PFN clause in the master directory schema subentry is employed to inform CDCS of the name given to the user library in which data base procedures reside.

Section 2 contains more explicit details on describing data base procedure use in the schema.

LOADING OF DATA BASE PROCEDURES

Data base procedures are loaded and executed in managed memory. During execution, CDCS maintains in memory only those data base procedures that are actively being used, although information about the data base procedure library is retained in CDCS tables. Loading and unloading operations involving the system loader incur performance degradation; therefore, data base procedures producing large object code should be avoided. The size of data base procedures and the number of data base procedures that might be activated should be considered when selecting a running field length for CDCS.

All data base procedures residing in the procedure library must be in capsule form for loading by the Fast Dynamic Loading (FDL) facility. Refer to a later subsection entitled Procedure Library Preparation for more details.

WRITING DATA BASE PROCEDURES

Data base procedures are relocatable binary subprograms that can be written in COBOL, FORTRAN, COMPASS, or any other language meeting all of the following requirements:

Provides a subprogram capability.

Uses the standard address list calling sequence.

Supports all data representations to be referenced.

Applicable language reference manuals should be consulted for detailed information on writing and compiling subprograms.

CDCS assumes that data base procedures adhere to the restrictions set forth in this manual. In particular, input/output on data base areas from data base procedures is not supported. Any input/output statements in high-level languages, CYBER Record Manager (CRM) input/output statements, or other RA+1 calls (such as CIO calls) are not supported. Use of input/output operations could cause a system failure because of the presence of external references to resolve, and would cause suspension of processing for all other CDCS user jobs.

LINKAGE AND COMMUNICATION

CDCS and the data base procedure communicate through a parameter list. When CDCS calls the data base procedure, CDCS passes a parameter list in which information pertinent to a particular CDCS call is made available to the data base procedure. Upon completion of execution, the data base procedure returns a code to CDCS to indicate whether CDCS should proceed normally or take appropriate action on an error condition that has been encountered.

Data base procedures written in COMPASS obtain the address of the parameter list in register A1. COBOL data base procedures must use the Linkage Section and the USING phrase in the Procedure Division header to handle the parameter list. Within FORTRAN data base procedures, the SUBROUTINE statement must be used to handle the parameter list.

PARAMETER LIST

The parameter lists provided by CDCS for each type of data base procedure call are shown in table 7-1. The one-word parameter addresses are right-justified with leading zeros. The entire parameter list is terminated by a word of zeros.

Most of the parameters themselves are in integer format with the exception of the buffer parameters, which can contain items in various formats. The target item, source item, and record key parameters indicated in the table denote specific locations within the CDCS buffer that holds the current record.

Those parameters containing a size or a length in bits or characters are so designated in the table. Character positions are numbered from 0 through 9, where character position 0 is the leftmost character; likewise, bit positions are numbered from 0 through 59, where the leftmost bit is bit position 0.

The program ID parameter contains the 1- to 10-character user identification obtained from the PROGRAM-ID paragraph in the Identification Division of a COBOL main program. When the FORTRAN Data Base Facility is being used, the program ID is obtained from the program unit that initially invoked CDCS. It is taken from the PROGRAM statement of a FORTRAN main program or from the name of the subprogram. The field is left-justified; unused positions are blank filled.

CDCS passes the relation name parameter only for record-level data base procedure calls. The relation name can be 30 characters or fewer. The field is left-justified; unused positions are blank filled. If the current input/output request being processed is not a relation read, the relation name parameter contains blanks.

The element type included in the privacy-level call is an integer corresponding to one of the following:

0 all elements (areas)

1 area

The area name (parameter 6) for the privacy-level call can be 30 characters or less.

Only the parameters applicable to each type of call are passed to the data base procedure. Table 7-2 shows the valid parameters and applicable entry codes for each type of call.

Since the number of parameters contained in the list differs with the type of call, the data base procedure should check the entry code to determine how many parameters have been passed for the call. This check is particularly important when the data base procedure processes more than one type of call.

TABLE 7-1. PARAMETER LIST FORMATS

Area-Level Call		Record-Level Call		Item-Level Call		Privacy-Level Call	
Word	Address of	Word	Address of	Word	Address of	Word	Address of
1	Entry code	1	Entry code	1	Entry code	1	Entry code
2	Return code	2	Return code	2	Return code	2	Return code
3	Error code	3	Error code	3	Error code	3	Error code
4	FIT	4	FIT	4	FIT	4	Privacy key
5	Program ID	5	Buffer for current record	5	Buffer for current record	5	Element type
		6	Length of current record in characters	6	Length of current record in characters	6	Area name
		7	Buffer for current record primary key	7	Buffer for current record primary key	7	Reserved
		8	Starting character position of key (0-9)	8	Starting character position of key (0-9)	8	Program ID
		9	Length of key in characters	9	Length of key in characters		
		10	Buffer for record code	10	Buffer for target item		
		11	Program ID	11	Target item beginning bit position (0-59)		
		12	Relation name	12	Target item size in bits		
				13	Target item data class		
				14	Buffer for source item		
				15	Source item beginning bit position (0-59)		
				16	Source item size in bits		
				17	Source item data class		
				18	Program ID		

TABLE 7-2. APPLICABLE PARAMETERS FOR TYPE OF CALL

Type of Call	Parameters Passed	Applicable Entry Codes
Area level	1-5	1-6
Record level	1-12	7-21, 36
Item level	1-18	22-35
Privacy level	1-8	37

The CYBER Record Manager Advanced Access Methods reference manual should be consulted for a description of parameters for the procedures that perform the following operations: compression/decompression of data and hashing to determine the primary key of records in direct access files.

Entry Codes and Return Codes

Entry code values for parameter 1 of the parameter list are given in table 7-3. The entry code indicates the type of CDCS function being serviced. The column designation Opt refers to the time of entrance option, which governs the time of execution of a data base procedure, as specified in the schema:

B BEFORE

E ON ERROR DURING

A AFTER

TABLE 7-3. ENTRY CODES AND VALID RETURN CODES

Entry Code	Condition		Return Code			Entry Code	Condition		Return Code		
	Function	Opt	0	1	3		Function	Opt	0	1	3
1	OPEN	B	C	FA	PA	20	DELETE	E	CE	CE	PA
2	OPEN	E	CE	CE	PA	21	DELETE	A	C	FA	PA
3	OPEN	A	C	FA	PA	22	(Item) GET	B	C	FA	PA
4	CLOSE†	B	C	FA	PA	23	(Item) GET	E	CE	CE	PA
5	CLOSE	E	CE	CE	PA	24	(Item) GET	A	C	FA	PA
6	CLOSE†	A	C	FA	PA	25	(Item) STORE	B	C	FA	PA
7	(Record) GET	B	C	FA	PA	26	(Item) STORE	E	CE	CE	PA
8	(Record) GET	E	CE	CE	PA	27	(Item) STORE	A	C	FA	PA
9	(Record) GET	A	C	FA	PA	28	(Item) MODIFY	B	C	FA	PA
10	(Record) STORE	B	C	FA	PA	29	(Item) MODIFY	E	CE	CE	PA
11	(Record) STORE	E	CE	CE	PA	30	(Item) MODIFY	A	C	FA	PA
12	(Record) STORE	A	C	FA	PA	31	ENCODING	-	C	FA	PA
13	(Record) MODIFY	B	C	FA	PA	32	DECODING	-	C	FA	PA
14	(Record) MODIFY	E	CE	CE	PA	33	ACTUAL RESULT	-	C	FA	PA
15	(Record) MODIFY	A	C	FA	PA	34	VIRTUAL RESULT	-	C	FA	PA
16	FIND	B	C	FA	PA	35	CHECK	-	C	FA	PA
17	FIND	E	CE	CE	PA	36	RECORD CODE	-	C	FA	PA
18	FIND	A	C	FA	PA	37	PRIVACY	-	C	FA	PA
19	DELETE	B	C	FA	PA						

Note: The Opt codes denote entrance options as follows:

B Before
E ON ERROR DURING
A AFTER

Note: The return codes denote CDCS processing as follows:

C Continues processing input/output function.
CE Continues error processing.
FA Aborts input/output function.
PA Aborts applications program.

†See subsection entitled Error Processing.

The return code, parameter 2 of the parameter list, must be set by the data base procedure before transferring control back to CDCS. Table 7-3 indicates valid return codes for CDCS functions. The following codes denote the CDCS processing that occurs following the return of a particular code:

C Continues processing input/output function.

CE Continues error processing.

FA Aborts input/output function.

PA Aborts application program.

If a data base procedure supplies an invalid return code, CDCS aborts the application program, calls no other data base procedures, and issues diagnostic message 446 (676 octal). CDCS interprets the return codes as follows:

Return code 0

Proceed normally. For ENCODING, DECODING, ACTUAL RESULT, and VIRTUAL RESULT functions, CDCS assumes the result is stored in the target item parameter. For the functions GET, STORE, and MODIFY, the record in the parameter buffer should be processed in the normal manner.

Return code 1

Terminate the CDCS function being performed. If an ON ERROR DURING data base procedure returns the value 1, the error code returned by CDCS to the application program is the same code that caused the procedure to be called. If a data base procedure other than an ON ERROR DURING procedure returns the value 1, CDCS returns error code 443 (673 octal) to the application program. The error exit for the application program is taken. COBOL and FORTRAN application programs can obtain the CDCS error code if the appropriate field for retrieval of data base status information is defined and checked. (Refer to the CDCS 2 Application Programming reference manual for details.) Query Update issues the user a message that indicates the error code and the function being performed when the error occurred.

Return code 3

Abort the application program. CDCS termination processing follows to close data base areas and related log files.

Interpretation of Parameters

Parameters 2 and 10 are the only parameters for which modification by a data base procedure is supported. All other parameters are to be read only. Erroneous results may occur if other parameter values are altered.

Parameter 10 can be interpreted in two ways depending on whether it is used by a data base procedure called for item-level processing or one called for record-level processing. For item-level procedures, parameter 10 should be modified only by ENCODING, DECODING, ACTUAL RESULT, and VIRTUAL RESULT data base procedures. The value placed in the parameter by the data base procedure is part of the target record and must conform to the target item characteristics contained in parameters 11, 12, and 13.

At the record level, parameter 10 is set by the data base procedure called for the purpose of determining the DDL record type of a record in the CDCS buffer. When an area contains records with more than one record type, the RECORD CODE clause in the schema data control entry can name the data base procedure that is to be called to examine the record and determine its record type. The data base procedure stores the resultant record code in parameter 10 for subsequent use by CDCS.

Parameter 3 holds the CDCS code for a nonfatal error. If a data base procedure is called for ON ERROR DURING processing, it can obtain the error code from this parameter and can take appropriate action for the kind of error detected.

DATA BASE PROCEDURES FOR INPUT/OUTPUT FUNCTIONS

Data base procedures for input/output functions can have execution time options associated with them as defined in the schema. The data administrator can specify the time of entrance to a data base procedure by choosing one of the options: BEFORE, ON ERROR DURING, or AFTER. The time of execution of a data base procedure is governed by the type of input/output function being processed and the execution time option selected, as shown in table 7-4. The correspondence between the CDCS function designated in the schema CALL clause and the input/output statement used in the application program is also indicated in the table.

READ STATEMENT

CDCS processes a READ statement in the following manner. AAM delivers the requested record to the CDCS buffer. CDCS then calls RECORD CODE, BEFORE, and item-level data base procedures, in that order. CDCS performs record mapping next and then calls any data base procedures indicated for the AFTER option. If mapping is not required, the BEFORE and AFTER procedures are, in effect, called in direct sequence. CDCS calls ON ERROR DURING data base procedures whenever a nonfatal error occurs during the processing of a READ statement.

WRITE AND REWRITE STATEMENTS

CDCS processes the WRITE and REWRITE statements as follows. Any data base procedures for the STORE and MODIFY functions with the BEFORE option selected are executed first. Next CDCS generates the schema representation of the record through record mapping if necessary. CDCS then calls any data base procedures for the AFTER option and follows with a call to AAM. For a write, parameters 5 and 6, the current record and the record length, are meaningless to a data base procedure for the BEFORE option, since the record buffer is the CDCS buffer and its contents at this point in the execution of the request are invalid (no mapping has been performed as yet).

TABLE 7-4. DATA BASE PROCEDURE TIME OF EXECUTION

Input/Output Statement	CDCS Function Correspondence	Time of Entrance Option	Time of Execution
READ, WRITE, and REWRITE	GET, STORE, and MODIFY	BEFORE	Before CDCS record mapping
		ON ERROR DURING	Any time a nonfatal error condition occurs during the processing of the function
		AFTER	After CDCS record mapping
OPEN, CLOSE, START, and DELETE	OPEN, CLOSE, FIND, and DELETE	BEFORE	Before call to AAM
		ON ERROR DURING	Any time a nonfatal error condition occurs during the processing of the function
		AFTER	After call to AAM

DELETE AND START STATEMENTS

When CDCS processes a DELETE statement, BEFORE and AFTER data base procedures for the DELETE function are executed before and after calls to AAM. No record mapping is involved. Parameters 5 and 6 for record-level and item-level data base procedures are of no value to the data base procedures called for the BEFORE and AFTER options because no record image is in memory. Parameter 5 is the address of the current record or of an item defined in the current record. The use of parameter 5 can cause the CDCS system control point to abort if the address supplied by parameter 5 is not within the field length of CDCS.

For a FIND function (COBOL or FORTRAN DML START statement), data base procedures with BEFORE, AFTER, and ON ERROR DURING options should be specified for the first record type in an area if more than one record type exists. CDCS calls only those data base procedures associated with the first record type in an area for this function. Processing of data base procedures for the FIND function is similar to the processing described above for the DELETE function. Also, as described above, the use of parameter 5 with the FIND function can cause the CDCS system control point to abort.

RELATION READ STATEMENT

A relation READ statement retrieves a record from each file joined in a relation. The processing of data base procedures during a relation read is explained in the following paragraphs.

AAM reads a file and brings a record to the CDCS buffer for record qualification and mapping if required. If the record qualifies, the sequence of data base procedure execution is as follows:

CDCS executes data base procedures for the BEFORE option.

CDCS performs record mapping and executes any item-level data base procedures when required.

CDCS executes data base procedures for the AFTER option.

CDCS repeats the above sequence for each qualifying record that is retrieved during the read of a relation. Data base procedures specified for the ON ERROR DURING option are called whenever a nonfatal error is detected. CDCS does not call these procedures for informative diagnostics such as control break and null record occurrence.

ERROR PROCESSING

Appendix B contains diagnostics associated with data base procedure processing. CDCS reports nonfatal and fatal errors by issuing appropriate diagnostic messages. A discussion in the CDCS 2 Application Programming reference manual explains CDCS error processing as it relates to the application program. The information that follows pertains only to data base procedure error handling.

NONFATAL ERRORS

The ON ERROR DURING option selected for a function specifies that a data base procedure is to be called if a nonfatal error condition occurs when that function is being performed on a particular area, record, or item. CDCS does not call data base procedures specified with the AFTER option for that function after a call to an ON ERROR DURING data base procedure.

One exception exists, however, for calling ON ERROR DURING data base procedures. For item-level calls, CDCS calls these data base procedures only for errors that occur during the mapping of data items; that is, for errors with error code 445 (675 octal). Data base procedures that are called for the ON ERROR DURING option can obtain the error code in parameter 3.

FATAL ERRORS

When an application program is aborted by CDCS because a fatal error condition exists, CDCS processing closes all data base areas and empties log files that are still open. CDCS does not call data base procedures named for the CLOSE function when an application program is aborted.

RETURN CODES

When CDCS is to call more than one procedure for a particular time of entrance option, CDCS checks the return code after each data base procedure finishes execution. If the data base procedure returns a code of 1, CDCS terminates the function immediately; CDCS calls no other data base procedures except ON ERROR DURING data base procedures. If the procedure returns a code of 3, CDCS aborts the application program at that time and calls no other data base procedures.

A single difference exists when a data base procedure returns a code of 1 and has been called to satisfy the ON ERROR DURING option for a particular function. Since the function is tc be aborted anyway, a code of 1 returned by the data base procedure is repetitious and has no effect on subsequent CDCS processing (CDCS calls additional ON ERROR DURING data base procedures for that function). Thus a return code of 0 or 1 has the same effect for an ON ERROR DURING data base procedure.

A code of 1 returned by a data base procedure that has been called to fulfill a BEFORE or AFTER option causes the function to be aborted. In this case, if a data base procedure has been specified with the ON ERROR DURING option, CDCS calls the procedure and aborts the function.

PROCEDURE LIBRARY PREPARATION

All data base procedures residing in the procedure library specified in the master directory must be encapsulated. The encapsulation process forms object programs into capsules that can be loaded by the Fast Dynamic Loading (FDL) facility. The data administrator must prepare a job stream as shown in figure 7-1 to generate capsules and place them in a user library. The example uses COBOL data base procedures.

Data base procedures listed in the CAPSULE control statement must be in the same order in which they were compiled. Each capsule is a member of a capsule group. The FDL GROUP control statement requires a unique group name of seven characters. For CDCS, the group name must have P in the first character position and the schema ID assigned by the master directory utility (right-justified with leading zeros) in character positions two through seven. If a schema is deleted, then reinserted in the master directory, the schema ID should be compared with the previous ID. If the IDs do not match, the procedure library must be regenerated with the correct ID used in the FDL GROUP control statement.

Within a generated capsule, any required external references that are unsatisfied can cause CDCS to abort with a mode error. Because unsatisfied external references are not flagged as errors when the capsule is generated, the data administrator should insert the following control statement at the beginning of each load sequence for capsule generation:

```
LDSET(MAP=SBX)
```

The X option in this statement produces a listing of all unsatisfied external references for each capsule generated. The listing can be examined to determine whether or not all required external references have been properly satisfied.

If routines from the library SYSLIB are referenced by a capsule during the encapsulation process, SYSLIB must be added to the local library set: SYSLIB is not automatically searched for externals by FDL. The following control statement must be added to the load sequence for the generation of the capsule:

```
LDSET(LIB=SYSLIB)
```

The addition of this statement allows external references to be satisfied from the library SYSLIB during capsule generation.

NOS Operating System

```
Job statement
USER control statement
CHARGE control sttement
COBOL5,MSB.
REWIND,LGO.
GROUP(P000001)
CAPSULE(DBP0,DBP1,...)
CAPSULE(DBP11,DBP12,...)
LDSET(NOEPT)
LOAD,LGO.
NOGO,Q.
DEFINE,NEWLIB=PROCLIB.
LIBEDIT,P=0,B=Q,N=TEMP.
LIBGEN,F=TEMP,P=NEWLIB,NX=1.
End-of-record
    COBOL data base procedures
End-of-record
*B *,CAP/*
End-of-information
```

NOS/BE Operating System

```
Job statement
ACCOUNT control statement
COBOL5,MSB.
REWIND,LGO.
GROUP(P000001)
CAPSULE(DBP0,DBP1,...)
LDSET(NOEPT)
LOAD,LGO.
NOGO,Q.
REQUEST,NEWLIB,PF.
EDITLIB.
CATALOG,NEWLIB,PROCLIB...
End-of-record
    COBOL data base procedures
End-of-record
LIBRARY(NEWLIB,NEW)
ADD(*,Q)
FINISH.
ENDRUN.
End-of-information
```

Figure 7-1. Procedure Library Generation

SAMPLE DATA BASE PROCEDURES

This subsection contains information on COBOL, FORTRAN, and COMPASS data base procedures. A sample COBOL data base procedure illustrates one use of this CDCS facility.

COBOL DATA BASE PROCEDURES

COBOL data base procedures are main subprograms that follow the COBOL rules for called subprograms. The MSB parameter must be specified in the COBOL5 control statement when the subprogram is compiled. The Linkage Section describes the CDCS parameter list that is passed to the called data base procedure through the USING phrase in the Procedure Division header of the called program. No space is allocated in the data base procedure for the data described in the Linkage Section.

Parameters defined in the Linkage Section are included in the data base procedure, depending on the level of the call, the size of the record, and the type of processing performed by the procedure. All parameters need not be included if they are not necessary in processing the call.

Computational items require a PICTURE clause for COMP-1 items. COMP-1 items must have a size less than or equal to 14.

The current record and record key entries must be large enough to contain the largest record found in the area and the largest record key, respectively. Likewise, the target item and source item entries must be large enough to hold the largest item found in the record.

The sample COBOL data base procedure DBPMS1 encodes a data item called MARITAL-STATUS, which is part of an employee record, from the form of the item contained in the source item. Figure 7-2 shows the item descriptions that appear in the schema and subschema.

```
Schema Description

    01  MARITAL-STATUS PICTURE "A"
        FOR ENDCODING CALL DBPMS1
        FOR DECODING CALL DBPMS2.


Subschema Description

    03  MARITAL-STATUS PICTURE A (9).
```

Figure 7-2. ENDCODING/DECODING Description for COBOL Data Base Procedure

When an application program executes a STORE function, CDCS calls DBPMS1 to provide a one-character code for MARITAL-STATUS before storing the record in the data base. The nine-character item described in the subschema determines the code. Conversely, when a program retrieves the same record, CDCS calls an analogous data base procedure, DBPMS2, to decode MARITAL-STATUS. In this example, disk space conservation is the prime consideration for storing data in an encoded form.

Figure 7-3 shows the COBOL data base procedure DBPMS1. Source item and target item are known to be on word boundaries in this example. Parameters 11 and 15, which give the beginning bit positions of the source and target items, can be used to determine whether an item needs to be aligned to a word boundary and to perform the appropriate alignment.

FORTRAN DATA BASE PROCEDURES

FORTRAN data base procedures follow the rules for subroutine subprograms in FORTRAN. The SUBROUTINE statement must contain the parameter list arguments that CDCS passes to the data base procedure.

COMPASS DATA BASE PROCEDURES

COMPASS data base procedures normally perform tasks that can be accomplished more easily using an assembly language. The address of the parameter list is found in register A1.

```
CDC COBOL 5.3 - LEVEL 552       SOURCE LISTING OF DBPMS1

 1              IDENTIFICATION DIVISION.
 2              PROGRAM-ID.   DBPMS1.
 3             *    DATA BASE PROCEDURE TO ENCODE DATA ITEM
 4             *    MARITAL STATUS FOR SCHEMA REPRESENTATION.
 5              ENVIRONMENT DIVISION.
 6              CONFIGURATION SECTION.
 7              SOURCE-COMPUTER. CYBER-170.
 8              OBJECT-COMPUTER. CYBER-170.
 9              DATA DIVISION.
10              WORKING-STORAGE SECTION.
11              01  MARITAL-STATUS.
12                  05 SINGLE          PIC A   VALUE IS "S".
13                  05 MARRIED         PIC A   VALUE IS "M".
14                  05 DIVORCED        PIC A   VALUE IS "D".
15                  05 WIDOWED         PIC A   VALUE IS "W".
16                  05 SEPARATED       PIC A   VALUE IS "E".
17              LINKAGE SECTION.
18              01  ENTRY-CODE         PIC 9(10) USAGE COMP-1.
19              01  RETURN-CODE        PIC 9(10) USAGE COMP-1.
20              01  ERR-CODE           PIC 9(10) USAGE COMP-1.
21              01  FILE-INFO-TABLE.
22                  05  FIT            PIC X(10) OCCURS 35 TIMES.
23              01  CURRENT-RECORD.
24                  05 RECORD-AREA     PIC X(10) OCCURS 200 TIMES.
25              01  RECORD-LENGTH      PIC 9(10) USAGE COMP-1.
26              01  CURRENT-KEY.
27                  05 KEY-AREA        PIC X OCCURS 260 TIMES.
28              01  KEY-POS            PIC 9(10) USAGE COMP-1.
29              01  KEY-LENGTH         PIC 9(10) USAGE COMP-1.
30              01  TARGET-ITEM        PIC A.
31              01  TARGET-ITEM-POS    PIC 9(10) USAGE COMP-1.
32              01  TARGET-ITEM-SIZE   PIC 9(10) USAGE COMP-1.
33              01  TARGET-ITEM-CLASS PIC 9(10) USAGE COMP-1.
34              01  SOURCE-ITEM        PIC A(9).
35              01  SOURCE-ITEM-POS    PIC 9(10) USAGE COMP-1.
36              01  SOURCE-ITEM-SIZE   PIC 9(10) USAGE COMP-1.
37              01  SOURCE-ITEM-CLASS PIC 9(10) USAGE COMP-1.
38              01  PROG-ID            PIC X(10).
39              PROCEDURE DIVISION USING ENTRY-CODE, RETURN-CODE, ERR-CODE,
40                  FILE-INFO-TABLE, CURRENT-RECORD, RECORD-LENGTH, CURRENT-KEY,
41                  KEY-POS, KEY-LENGTH, TARGET-ITEM, TARGET-ITEM-POS,
42                  TARGET-ITEM-SIZE, TARGET-ITEM-CLASS, SOURCE-ITEM,
43                  SOURCE-ITEM-POS, SOURCE-ITEM-SIZE, SOURCE-ITEM-CLASS,
44                  PROG-ID.
45              100-START.
46                  IF SOURCE-ITEM IS EQUAL TO "SINGLE"
47                      MOVE SINGLE TO TARGET-ITEM
48                      GO TO END-DBP.
49                  IF SOURCE-ITEM IS EQUAL TO "MARRIED"
50                      MOVE MARRIED TO TARGET-ITEM
51                      GO TO END-DBP.
52                  IF SOURCE-ITEM IS EQUAL TO "DIVORCED"
53                      MOVE DIVORCED TO TARGET-ITEM
54                      GO TO END-DBP.
55                  IF SOURCE-ITEM IS EQUAL TO "WIDOWED"
56                      MOVE WIDOWED TO TARGET-ITEM
57                      GO TO END-DBP.
58                  IF SOURCE-ITEM IS EQUAL TO "SEPARATED"
59                      MOVE SEPARATED TO TARGET-ITEM
60                      GO TO END-DBP.
61                  MOVE 1 TO RETURN-CODE.
62                      GO TO END-DBP-A.
63              END-DBP.
64                  MOVE ZERO TO RETURN-CODE.
65              END-DBP-A.
66                  EXIT.
```

Figure 7-3. Sample COBOL Data Base Procedure

The master directory is a permanent file containing information on the data bases, schemas, and subschemas that are known to CDCS. The master directory is the source of all data base and media descriptions for CDCS. Miscellaneous information about the data bases, not specifiable in DDL syntax, is also contained in the master directory. Information derived from schemas and subschemas, logging specifications, and data base procedure information is included in the master directory. Before any data base areas can be processed by an applications program using CDCS, the master directory must be constructed by the data administrator.

The DBMSTRD utility creates and modifies the master directory. The DBMSTRD utility is called by a control statement. On a creation run, the DBMSTRD utility processes the entries from an input file to create the master directory. On a modification run, the DBMSTRD utility processes information in the existing master directory file and modification entries from an input file, and creates a new master directory. Modifications to the master directory are not made in place in the master directory file; rather, a new master directory file is created. A master directory can be built initially in a creation run and changed later in a modification run.

Master directories created by the DBMSTRD utility executing with CDCS 2.2 are not compatible with CDCS 2.3. The data administrator must regenerate master directories to upgrade to CDCS 2.3. Source input for a master directory under CDCS 2.2 is generally valid for CDCS 2.3 except for the SET and VSN clauses and for the clauses that specify log files.

MASTER DIRECTORY SYNTAX

All schemas, versions, areas, and subschemas that are to be used by applications programs must be identified in clauses in source input for the DBMSTRD utility. The clauses also specify logging requirements and identify log and recovery files, the data base procedure library, job control information, and permanent files associated with each area of each data base version.

Clauses in the source input file have a free-field format. Data appears in columns 1 through 72; columns 73 through 80 are not examined and can be used for sequencing or identification purposes. Spaces, commas, and semicolons are used equivalently as separators in the input to DBMSTRD. If a comma or semicolon is used, it need not be preceded or followed by a space. Periods must appear in source input as indicated in format statements. The format of source input for the DBMSTRD utility is summarized in appendix E.

Source input to DBMSTRD can be coded for a creation run or, if a master directory exists, for a modification run.

Source input for the master directory provides all information known to CDCS about permanent files except for permanent file information for the master directory file itself. The permanent file information subentry, which is included in clauses in a creation run or modification run, specifies permanent file information required by CDCS. This subentry is described in the following paragraphs and is referenced throughout the descriptions of master directory syntax.

PERMANENT FILE INFORMATION SUBENTRY

The permanent file information subentry identifies the following information: the permanent file name, the user name (UN for use on NOS) or the file identification (ID for use on NOS/BE), passwords, and device-specific information.

The permanent file information subentry provides the information required for CDCS to attach permanent files. The following permanent files are described by this subentry:

- Area
- Index
- Journal log
- Procedure library
- Restart identifier
- Quick recovery
- Transaction recovery

The permanent file information subentry immediately follows the keywords that specify the kind of file to which the permanent file information applies. There is one exception: for an area, the permanent file information subentry immediately follows the area name.

The format of the permanent file information subentry is shown in figure 8-1.

```
PFN IS literal-1 { UN } IS literal-2 [PW IS literal-3 [literal-4]...]
                 { ID }

[ FAMILY NAME IS family-name                    ]
[ PACK NAME IS pack-name                        ] [DEVICE TYPE IS dt]
[ SET NAME IS set-name  VSN IS literal-5        ]
```

Figure 8-1. Permanent File Information Subentry Format

Permanent file information usually must indicate a unique file in the master directory. The information that provides unique identification for a file is a combination of information as follows:

Permanent file name (literal-1)

UN/ID

FAMILY/SET/PACK

Literal-1 indicates the file name. UN or ID indicates the user name or user identification. FAMILY, SET, or PACK indicates device-specific information as applicable to the operating system. To provide unique identification for the file, this combination of the permanent file information must not be duplicated for any other permanent file known to the DBMSTRD utility. Permanent files known to DBMSTRD include the set of all permanent files specified in a creation run. In a modification run, the set includes all permanent files specified in a modification run and included in the existing master directory.

PFN Clause

The PFN clause (figure 8-1) identifies the permanent file name. This clause is required.

The literal-1 must be from one through seven characters in length (with two exceptions), must contain only letters and digits, must begin with a letter, and must be enclosed in quotation marks. The literal-1 for a transaction recovery file or a journal log file must be exactly six characters in length. (Refer to the subsections describing clauses for these files.)

UN/ID Clause

The UN/ID clause (figure 8-1) identifies the user number or user identification. Either the UN or ID can be specified. The literal-2 must be from one through seven characters in length, must contain only letters and digits, and must be enclosed in quotation marks. This clause is required.

PW Clause

The PW clause (figure 8-1) specifies passwords that CDCS uses to attach the permanent file. Up to five passwords can be specified for use on NOS/BE; for NOS, only the first password is used. The literal-2 and literal-3 must be from one through seven characters in length, must contain only letters and digits, and must be enclosed in quotation marks. This clause is optional.

FAMILY NAME Clause

The FAMILY NAME clause (figure 8-1) specifies a family name and is applicable only for use on NOS. The family-name can contain only letters or digits and must be from one through seven characters in length. This clause is optional. Refer to the NOS reference manual for detailed information about devices associated with a family name.

There is a restriction on the use of this clause with the Batch Test Facility (CDCSBTF). CDCSBTF cannot access files that have a FAMILY name specified. This applies to all files that use the permanent file information subentry. This problem occurs because CDCSBTF is not a system origin job, whereas CDCS is a system origin job when brought up at a system control point.

To use files from another family through CDCSBTF without using the FAMILY clause, the desired family name should be placed in the USER statement under which CDCSBTF is running. All of the files to be accessed by CDCSBTF should reside on that family. To avoid having two master directories, one for CDCSBTF and one for CDCS, the version feature can be used as follows:

For jobs using CDCSBTF, a version of the database can be defined without specifying FAMILY clauses.

For jobs interacting with CDCS at a system control point, a version can be defined with the FAMILY clauses specified.

PACK NAME Clause

The PACK NAME (figure 8-1) specifies an auxiliary device identifier and is applicable only for use on NOS. The pack-name can consist of letters or digits and must be from one through seven characters in length. This clause is optional.

SET NAME Clause

The SET NAME clause (figure 8-1) specifies a private device set and is applicable only for use on NOS/BE. The set-name can consist of letters or digits and must be from one through seven characters in length. This clause is optional.

VSN Clause

The VSN clause (figure 8-1) identifies the volume serial number of the master pack and is applicable only for use on NOS/BE. This clause is optional; however, if the SET NAME clause is specified for a permanent file, the VSN clause must also be specified. Literal-4 can contain letters or digits, must be from one through six characters in length, and must be enclosed in quotation marks.

DEVICE TYPE Clause

The DEVICE TYPE clause (figure 8-1) is optional. This clause is applicable only for use on NOS. The DEVICE TYPE clause provides for the assignment of an auxiliary mass storage device to the permanent file specified in the clause in which the DEVICE TYPE subentry also appears. An auxiliary device is any supported device which an installation defines as auxiliary. The dt specification must conform to the following rules:

Dt must be from two to three characters in length.

The first two characters must be letters.

The third character, if used, must be a digit.

If the specification of dt does not conform to these rules, the DBMSTRD utility issues a diagnostic; however, the DBMSTRD utility does not check for the validity of the mnemonic used. Refer to the NOS reference manual for information about the auxiliary device types supported by NOS.

SYNTAX FOR THE CREATION RUN

A master directory is generated when source input for a creation run is processed by the DBMSTRD utility. The general format for the creation run is shown in figure 8-2. A creation run can contain more than one creation entry; up to 4095 schemas can be specified in the master directory.

```
{creation entry} ...
```

Figure 8-2. General Format, Creation Run

The creation entry defines the following information for a data base: all log and recovery files, versions, areas, and subschemas associated with a particular schema. A series of subentries is required in the creation entry. The general format of a creation entry is shown in figure 8-3.

```
schema subentry
master version subentry
[alternate version subentry] ...
{subschema subentry} ...
```

Figure 8-3. General Format, Creation Entry

The subentries within the creation entry must appear in the order indicated in figure 8-3. The alternate version subentry and the subschema subentry can be repeated. A maximum of 4095 data base versions can be specified for a schema. Any number of subschemas can be associated with a particular schema in the master directory.

SCHEMA SUBENTRY

The schema subentry specifies the schema name, the log and recovery files, and the procedure library file used for the schema. Also, job control information for the CDCS-initiated job to dump the journal log file can be specified. The format of the schema subentry is shown in figure 8-4. A period must terminate the schema subentry.

Two clauses are required in the schema subentry: SCHEMA NAME and FILE NAME. The SCHEMA NAME clause must appear first in the subentry and must be followed by the FILE NAME clause. The other clauses are optional; they include the following:

PROCEDURE LIBRARY clause

TRANSACTION RECOVERY FILE clause

RESTART IDENTIFIER FILE clause

JOURNAL LOG FILE clause

```
SCHEMA NAME IS schema-name
        FILE NAME IS lfn
    [PROCEDURE LIBRARY
        permanent file information subentry]
    [TRANSACTION RECOVERY FILE
        permanent file information subentry
        [UNIT LIMIT IS integer-1]
        [UPDATE LIMIT IS integer-2]]
    [RESTART IDENTIFIER FILE
        permanent file information subentry]
    [JOURNAL LOG FILE
        permanent file information subentry]
    [QUICK RECOVERY FILE
        permanent file information subentry]
    [JOB CONTROL INFORMATION
        [TAPE (TYPE IS {MT|NT}
               DENSITY IS {LO|HI|HY|HD|PE|GE})]
        [UN IS literal-1]
        [PW IS literal-2]
        [FAMILY NAME IS family-name]
        [{ACCOUNT|CHARGE} IS literal-3]
        [CCLPROC NAME IS literal-4
            [ACCESS IS {INDIRECT|DIRECT}]]].
```

Figure 8-4. Schema Subentry Format

QUICK RECOVERY FILE clause

JOB CONTROL INFORMATION clause

The optional clauses can appear in any order within the schema subentry.

SCHEMA NAME Clause

The SCHEMA NAME clause (figure 8-4) specifies the name of a schema for a data base. This clause is required.

The schema name must be unique among all schema names in the master directory. The name specified here is the same schema name specified in the schema directory and listed on the schema compilation listing. A schema name can be from 1 through 30 characters in length.

FILE NAME Clause (Schema Subentry)

The FILE NAME clause (figure 8-4) identifies the name of the local file that contains the schema directory. This clause is required.

The local file name (lfn) must be unique among all local file names specified in the master directory. The lfn must be from one through seven characters in length, must contain only letters and digits, and must begin with a letter. The file containing the schema directory must be attached in the job stream with this lfn before execution of the DBMSTRD control statement.

PROCEDURE LIBRARY Clause

The PROCEDURE LIBRARY clause (figure 8-4) specifies the permanent file information for the file that contains the data base procedure library. This clause must be used if data base procedures have been defined in the schema; it must be omitted if data base procedures have not been defined in the schema.

The permanent file information for the procedure library file must be specified as described in the Permanent File Subentry subsection. The combination of permanent file information for permanent file name, UN/ID, and family/set/pack must be unique in the master directory. Additional permanent file information must be sufficient to enable CDCS to attach the permanent file with the correct permission.

All data base procedures for a schema must reside in the specified file. They must be in a form appropriate for loading by the fast dynamic loader (FDL). The permanent file must exist before CDCS accesses it during execution. For information about creating a data base procedure library, refer to section 7.

TRANSACTION RECOVERY FILE Clause

The TRANSACTION RECOVERY FILE clause (figure 8-4) selects transaction logging and identifies the file to which log records are written. This clause is optional, but it must be specified in order for logging to a transaction recovery file to be performed for the schema.

The permanent file information for the transaction recovery file must be specified as described in the Permanent File Subentry subsection. The literal that specifies the permanent file name must follow the rules for permanent file names except that it must be six characters in length. The permanent file name that CDCS assumes for the transaction recovery file begins with these six characters; the number 1 is the seventh character. The combination of permanent file information for permanent file name (six characters and the character 1), UN/ID, and family/set/pack must be unique in the master directory. The additional permanent file information must be sufficient to enable CDCS to attach the permanent file with the correct permission.

Refer to section 9 for information about the use of the transaction recovery file.

UNIT LIMIT Clause

The UNIT LIMIT clause specifies the maximum number of outstanding transactions that are allowed for a schema. An outstanding transaction is one that was begun but not committed. This clause is optional.

The integer-1 specifies the limit; it must be a positive decimal integer with a maximum value of 63.

If this clause is not specified, the value 15 is assumed.

UPDATE LIMIT Clause

The UPDATE LIMIT clause specifies the maximum number of updates that can be made within a particular transaction. An update is one of the following operations: delete, rewrite, or write. This clause is optional.

The integer-2 specifies the maximum number of updates; it must be a positive decimal integer with a maximum value of 63.

If this clause is not specified, the value 15 is assumed.

RESTART IDENTIFIER FILE Clause

The RESTART IDENTIFIER FILE clause (figure 8-4) selects restart identifier logging and identifies the file to which the restart identifiers and associated committed transactions are written. This clause is optional, but it must be specified in order for logging to a restart identifier file to be performed for the schema. If this clause is specified, the TRANSACTION RECOVERY FILE clause must also be specified.

The permanent file information for the restart identifier file must be specified as described in the Permanent File Subentry subsection. The combination of permanent file information for permanent file name, UN/ID, and family/set/pack must be unique in the master directory. The additional permanent file information must be sufficient to enable CDCS to attach the permanent file with the correct permission.

Refer to section 9 for information about the use of the restart identifier file.

JOURNAL LOG FILE Clause

The JOURNAL LOG FILE clause (figure 8-4) specifies logging to a journal log file and identifies two journal log files. This clause is optional, but it must be specified in order for logging to a journal log file to be performed for the schema. The LOG clause indicating the logging of either BEFORE RECORDS or AFTER RECORDS must be specified in the area subentry so that logging to the journal log

file can be performed for a given area of a given version.

The permanent file information for the journal log file must be specified as described in the Permanent File Subentry subsection. The literal that specifies the permanent file name must follow the rules for permanent file names except that it must contain exactly six characters. The permanent file names that CDCS assumes for the journal log files begin with these six characters. For one file, the number 1 is the seventh character; for the other file, the number 2 is the seventh character. The combination of permanent file information for permanent file name, UN/ID, and family/set/pack must be unique in the master directory. Additional permanent file information must be sufficient to enable CDCS to attach the permanent file with the correct permission.

This clause provides CDCS with two permanent file names for two separate journal log files: one name for the active file and one name for the alternate file. Refer to section 9 for information about the use of the journal log file.

QUICK RECOVERY FILE Clause

The QUICK RECOVERY FILE clause (figure 8-4) selects block logging and identifies the file to which the before-image blocks are written. This clause is optional, but it must be specified in order for logging to a quick recovery file to be performed for the schema. Additionally, the LOG clause and the BEFORE IMAGE BLOCKS logging option must be specified in an area subentry for logging to the quick recovery file to be performed for a given area of a given version.

The permanent file information for the quick recovery file must be specified as described in the Permanent File Subentry subsection. The combination of permanent file information for permanent file name, UN/ID, and family/set/pack must be unique in the master directory. The additional permanent file information must be sufficient to enable CDCS to attach the permanent file with the correct permission.

Refer to section 9 for information about the use of the quick recovery file.

JOB CONTROL INFORMATION Clause

The JOB CONTROL INFORMATION clause (figure 8-4) designates the job control information to be used by CDCS when constructing the online job that dumps the contents of the journal log files. This clause is optional, but it must be specified for the online dump of the journal log file to be performed. The optional clauses that follow the keywords JOB CONTROL INFORMATION provide CDCS with the necessary information to construct the job that performs the dump.

The appropriate optional clauses to specify in the source input to the DBMSTRD utility depend on the operating system used for executing CDCS. For NOS, all optional clauses can apply with the UN and PW clauses being a minimum specification. For NOS/BE, only the TAPE/CCLPROC and CHARGE/ACCOUNT clauses apply. The user must know the operating system restrictions that apply to information specified in these clauses. Refer to the NOS or NOS/BE reference manuals for further information.

TAPE Clause

The TAPE clause identifies the characteristics of the tape to which the journal log file is dumped. The TYPE options are MT (7-track tape) and NT (9-track tape). The DENSITY options that can be specified with each TYPE option are as follows:

MT (7-track tape)

- LO (200 bits per inch [bpi])
- HI (556 bpi)
- HY (800 bpi)

NT (9-track tape)

- HD (800 characters per inch [cpi])
- PE (1600 cpi)
- GE (6250 cpi)

If TYPE MT or NT is specified without a DENSITY option being specified, the installation default density for the type specified is assumed. If a DENSITY option is specified without a TYPE option being specified, the type is determined as indicated in the preceding list of allowed DENSITY options for each type. If the entire TAPE clause is omitted, TYPE NT is assumed with the default density of the installation.

CCLPROC Clause

The CCLPROC clause is an alternative to the TAPE clause. If neither is specified the TAPE option is assumed. The CCLPROC clause identifies a user defined CCL procedure that is called by the online job that dumps the contents of the journal log file.

Use of this option permits the installation to control whether the journal log file is dumped to a tape or to a permanent file. It is also possible to specify that the contents of the journal log file is appended to the dump file instead of over-writing it.

The following basic steps must be contained within the CCL procedure defined by Literal-4:

1. Specify an ATTACH, REQUEST or LABEL statement to obtain the file to which the journal log file is to be dumped. This file must be given the local file name of LOGDUMP.
2. The file, LOGDUMP, must be positioned at the beginning or at the end of information, as selected by the installation.
3. The CDCS master directory must be attached with the local file name MSTRDIR.
4. The utility progam, DBREC, must be called to dump the journal log file.

The ACCESS option is applicable only for use on NOS. The default is INDIRECT. If the procedure is on a direct permanent file, ACCESS IS DIRECT must be specified.

UN Clause

The UN clause identifies the user name and is applicable only for use on NOS. The literal-1 must be from one through seven characters in length, must contain only letters and digits, and must be enclosed in quotation marks.

PW Clause

The PW clause identifies the password associated with the user number specified in the UN clause and is applicable only for use on NOS. The literal-2 must be from one through seven characters in length, must contain only letters and digits, and must be enclosed in quotation marks.

FAMILY NAME Clause

The FAMILY NAME clause identifies the family name and is applicable only for use on NOS. The family-name can contain only letters or digits and must be one through seven characters in length.

CHARGE/ACCOUNT Clause

The CHARGE/ACCOUNT clause identifies the charge number. For NOS, literal-3 must be from 1 through 30 characters in length; the user must ensure that the literal does not exceed 30 characters because the DBMSTRD utility does not check the length. For NOS/BE, literal-3 must be from 1 through 70 characters in length. The literal must be enclosed in quotation marks.

MASTER VERSION SUBENTRY

The master version subentry designates (either explicitly or by default) the name MASTER to the version. It also associates each area of the schema with logging options and a permanent file (or two permanent files if an index file is required for the area). This subentry is required. The format of the master version subentry is shown in figure 8-5.

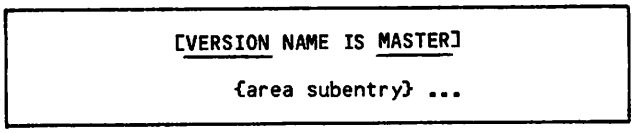


Figure 8-5. Master Version Subentry Format

The VERSION NAME clause in the master version subentry is optional. If this clause is not specified, the version name MASTER is assumed. If this clause is specified, it must immediately follow the schema subentry in the master directory source input.

A separate area subentry is required for each area defined in the schema. Only one area subentry per area can be specified within the master version subentry. Refer to the following subsection for the description of area subentry.

If the VERSION NAME clause is specified, the first area subentry must follow that clause. Otherwise, an area subentry must immediately follow the schema subentry.

AREA SUBENTRY (MASTER VERSION SUBENTRY)

The area subentry specifies an area defined in the schema and designates the name of the permanent file for the data file associated with the area. If an index file is required for the area, this subentry designates the name of the permanent file for the index file for the area. The area subentry also specifies the logging options selected for the area. The format of the area subentry is shown in figure 8-6. A period follows each area subentry.

AREA NAME Clause (Area Subentry)

The AREA NAME clause (figure 8-6) in the area subentry specifies the name of the area in the schema. This clause is required.

The area name specified must be the same as an area name specified in the schema directory and listed on the schema compilation listing. An area name must be from 1 through 30 characters in length. A particular area can be specified only once in the source input for DBMSTRD for the particular version of the schema.

Permanent File Information Subentry (Area Subentry)

The PFN literal-1 clause and subsequent clauses through SET and VSN (figure 8-6) actually make up a permanent file information subentry. These clauses are shown in figure 8-6 for the convenience of the user. The permanent file information subentry for the area file is required and must be specified as described in the Permanent File Information Subentry subsection. The combination of permanent file information for permanent file name, UN/ID, and family/set/pack must be unique in the master directory. The additional permanent file information must be sufficient to enable CDCS to attach the permanent file with the correct permission.

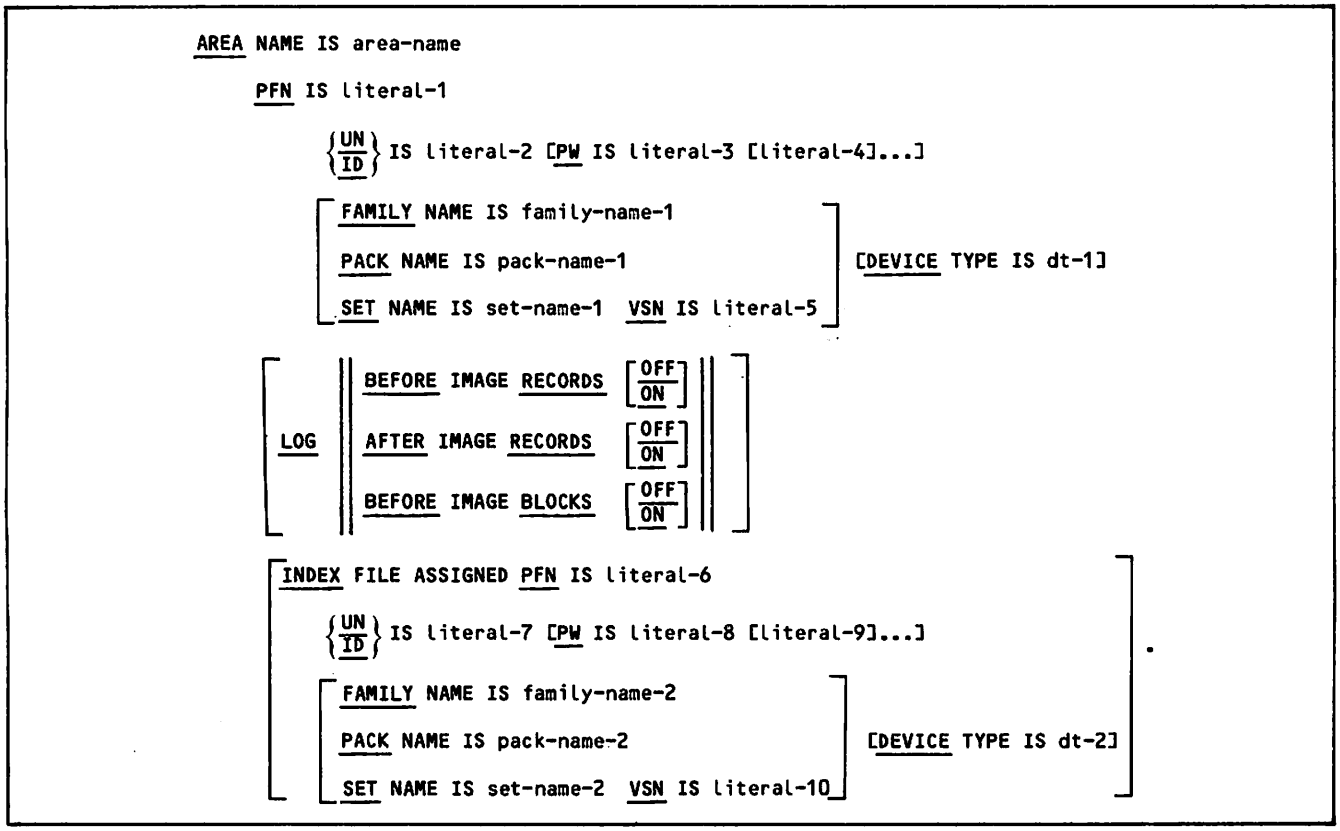


Figure 8-6. Area Subentry Format

LOG Clause (Area Subentry)

The LOG clause (figure 8-6) in the area subentry specifies the types of logging to be performed for the area. This clause is optional.

Three logging options can be specified. The BEFORE IMAGE RECORDS option selects logging of before-image records to a journal log file. (A before-image record is a copy of a record before it is updated.) The AFTER IMAGE RECORDS option specifies logging of after-image records to a journal log file. (An after-image record is a copy of a record after it is updated). The BEFORE IMAGE BLOCKS option selects logging of CRM blocks to a quick recovery file.

The options that can be specified depend on the log files selected for the schema. If a journal log file is selected for the schema, the BEFORE IMAGE RECORDS option and the AFTER IMAGE RECORDS option can be specified. If a quick recovery log file is selected for the schema, the BEFORE IMAGE BLOCKS option can be specified. All or none of the options can be specified if the log file requirements are met.

The keyword ON is assumed if a particular logging option is specified; ON can be specified for documentation purposes. Specification of the keyword OFF causes no logging of the option indicated to be performed. If no logging options are specified, the OFF option is assumed.

Refer to section 9 for details on the journal log file, the quick recovery file, and the logging options that must be selected for different recovery capabilities.

INDEX FILE Clause (Area Subentry)

The INDEX FILE clause (figure 8-6) in the area subentry specifies permanent file information for the index file for the area of the particular version. This clause is required if alternate key processing has been defined for the area and is not allowed if no alternate keys are defined.

The PFN literal-6 clause and subsequent clauses through SET and VSN actually make up a permanent file information subentry. These clauses are shown in figure 8-6 for the convenience of the user. The permanent file information subentry for the index file must be be specified as described in the Permanent File Information Subentry subsection. The combination of permanent file information for permanent file name, UN/ID, and family/set/pack must be unique in the master directory. The additional permanent file information must be sufficient to enable CDCS to attach the permanent file with the correct permission.

ALTERNATE VERSION SUBENTRY

An alternate version subentry designates the name of an alternate data base version. It also associates each area of the schema with logging options and a permanent file (or two permanent files if an index file is required for the area). This subentry is optional. The format of the alternate version subentry is shown in figure 8-7.

An alternate version subentry must be included in the master directory source input for each version defined in addition to version MASTER. A total of 4095 versions can be defined for a particular schema in a master directory.

The permanent files associated with an area of an alternate version can be either the same files associated with the same area in version MASTER or unique files.

For any two areas that are associated by a constraint, the permanent files associated with the areas in each non-MASTER version must conform to these rules:

- Files for both areas of the constraint must be the same as for version MASTER.
- Files for both areas of the constraint must be permanent files unique to the version.

VERSION NAME Clause (Alternate Version Subentry)

The VERSION NAME clause (figure 8-7) designates the name of an alternate data base version. This clause is required within an alternate version subentry.

The version name specifies the alternate version name. The version name must be from one through seven characters in length, must contain letters or digits, must begin with a letter, and must duplicate no other version name associated with the same schema.

AREA NAME SAME AS MASTER Clause (Alternate Version Subentry)

The AREA NAME SAME AS MASTER clause (figure 8-7) associates the specified area of the alternate version with the same permanent files and logging options as specified for the same area in version MASTER. This clause is optional. If this clause is not specified and no area subentry is specified for a particular area within the alternate version subentry, SAME AS MASTER is assumed.

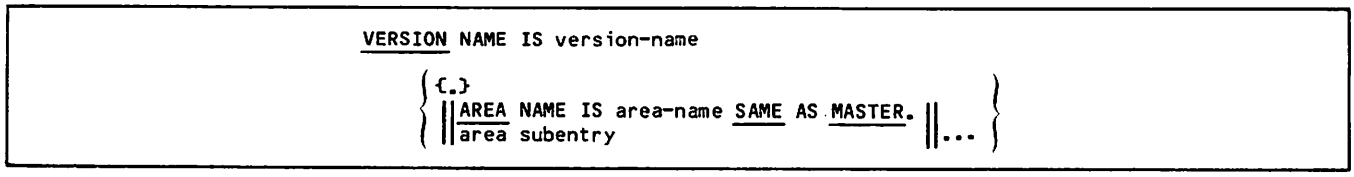


Figure 8-7. Alternate Version Subentry Format

The area name specified in the clause must be an area defined in the schema. The area name cannot duplicate an area name specified in either another AREA NAME SAME AS MASTER clause or in an area subentry within the alternate version subentry.

A period must terminate this clause.

Area Subentry (Alternate Version Subentry)

The area subentry specifies an area defined in the schema and designates the name of the permanent file for the data file associated with the area. If an index file is required for the area, the area subentry designates the name of the permanent file for the index file for the area. The area subentry also designates the logging options selected for the area. Refer to figure 8-6 for the format of the area subentry.

An area subentry is optional, but it is required to associate an area with permanent files other than those specified for the area in the master version subentry.

The area name specified in the area subentry cannot duplicate an area name specified either in another area subentry or in an AREA NAME SAME AS MASTER clause within the alternate version subentry.

An area subentry must be consistent with the area subentry specified for the same area in version MASTER. If the same area in version MASTER is assigned an index file, the area in the alternate version must also be assigned an index file. The logging options selected can be different for the same area in version MASTER and in the alternate version.

The permanent file information specified in the area subentry must identify unique permanent files. Refer to the Area Subentry in the Master Version Subentry subsection for the description of the clauses of the area subentry.

SUBSCHEMA SUBENTRY

A subschema subentry is included for each subschema that can be referenced by application programs using CDCS. At least one subschema subentry must be included for each schema. This subentry specifies the subschema name and the local file name of the file containing the subschema directory. The format of the subschema subentry is shown in figure 8-8.

```
SUBSCHEMA NAME IS subschema-name

FILE NAME IS lfn.
```

Figure 8-8. Subschema Subentry Format

SUBSCHEMA NAME Clause (Subschema Subentry)

The SUBSCHEMA NAME clause specifies the name of a subschema for a data base. This clause is required. The subschema name must be unique among all subschema names defined for the schema. The name specified here is the same subschema name specified in the DDL subschema definition. A subschema name must be from 1 through 30 characters in length.

FILE NAME Clause (Subschema Subentry)

The FILE NAME clause of the subschema subentry specifies the local file name (lfn) of the subschema directory or the subschema library. This clause is required.

The lfn must be from one through seven characters in length, must contain only letters and digits, and must begin with a letter. The file that contains the subschema directory or subschema library must be attached in the job stream with this lfn before the execution of the DBMSTRD control statement.

SYNTAX FOR MODIFICATION RUN

In a modification run, changes specified in the source input to the DBMSTRD utility are applied to an existing master directory, designated to DBMSTRD as the old master directory. The result of the modification run is a new master directory; the old master directory is not affected by the run. The modification run consists of three types of entries: add schema entry, delete schema entry, and modify schema entry. The general format of the modification run is shown in figure 8-9.

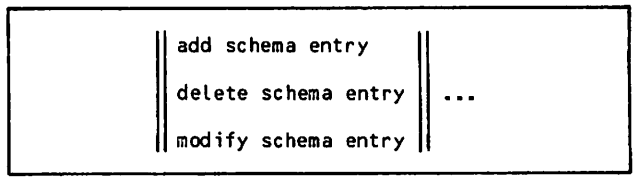


Figure 8-9. General Format, Modification Run

The entries of a modification run are optional and can appear in any order and as many times as desired, subject to a few restrictions:

- A schema can be named in a delete schema entry or in a modify schema entry only if information for it exists in the old master directory.
- A schema that is named in a modify schema entry cannot be named in a subsequent delete schema entry in the same run.
- A schema for which information is added by an add schema entry cannot be named in either a subsequent delete schema entry or modify schema entry in the same run.
- A schema for which information already exists in the old master directory can be deleted, and new information for a schema with the same name can be added.

The following subsections describe the entries for the modification run. The formats of the entries are summarized in appendix E.

ADD SCHEMA ENTRY

The add schema entry is used to add information for one or more schemas to the master directory. The add schema entry is composed of one or more creation entries with each creation entry composed of a schema subentry, master version subentry, alternate version subentries (optional), and subschema subentries. The format of the add schema entry is shown in figure 8-10.

```
ADD SCHEMAS.

    {creation entry} ...
```

Figure 8-10. Add Schema Entry Format

A creation entry must not be used for a schema with information already in the old master directory unless that schema is being deleted in a preceding delete schema entry. The series of entries of the creation entry must conform to the rules previously stated. Refer to the Schema Subentry, Master Version Subentry, Alternate Version Subentry, and Subschema Subentry subsections for the descriptions.

DELETE SCHEMA ENTRY

The delete schema entry is used to delete information for one or more schemas from the master directory. The format of the delete schema entry is shown in figure 8-11.

```
DELETE SCHEMAS.

    {SCHEMA NAME IS schema-name.} ...
```

Figure 8-11. Delete Schema Entry Format

The SCHEMA NAME clause identifies the schema being deleted. This clause is required.

The schema name must be a schema name that already exists in the old master directory. The schema name must not duplicate a schema name specified in the modify schema entry, or in an add schema entry that precedes this clause in the source input to DBMSTRD. The same schema name can be specified in an add schema entry that follows the delete schema entry.

MODIFY SCHEMA ENTRY

The modify schema entry identifies the modifications to be made to information for a schema that already exists in the old master directory. Only permanent file information for files that are already specified in the old master directory for the particular schema and associated areas can be changed. For example, a permanent file for a quick recovery file cannot be specified unless a quick recovery file is specified in the old master directory. Logging options selected for areas can be changed by use of the modify schema entry. Versions can be added and deleted; subschemas can be added and deleted. The format of the modify schema entry is shown in figure 8-12.

```
MODIFY SCHEMA NAME IS schema-name

    [FILE NAME IS lfn].

    | change procedure library subentry                  |
    | change transaction recovery file subentry          |
    | change restart identifier subentry                 |
    | change journal log file subentry                   |
    | change quick recovery file subentry                |
    | change job control information subentry            |
    | {change area subentry} ...                         |
    | {delete version subentry} ...                      |
    | {add version subentry} ...                         |
    | {delete subschema subentry} ...                    |
    | {add subschema subentry} ...                       |

END { MODIFICATIONS }.
    { MODS          }
```

Figure 8-12. Modify Schema Entry Format

The modify schema subentry must begin with the MODIFY SCHEMA NAME clause. The FILE NAME clause, if specified, must follow in the source input. The modify schema subentry must be terminated with the END MODIFICATIONS clause. The modification options can appear in any order in the source input with some restrictions for the add and delete subschema subentries. Those subentries that can be repeated (indicated by ellipsis in the format) can be repeated any number of times.

MODIFY SCHEMA NAME Clause

The MODIFY SCHEMA clause (figure 8-12) identifies the schema for which information is being modified. This clause is required.

The schema name must specify a schema name that already exists in the old master directory. The name specified here is the same schema name specified in the schema compilation. The schema name must be from 1 through 30 characters in length.

If the FILE NAME clause is omitted, a period must terminate the SCHEMA NAME clause.

FILE NAME Clause (Modify Schema Subentry)

The FILE NAME clause (figure 8-12) specifies the name of the local file containing the schema directory. This clause is required if an ADD SUBSCHEMA NAME clause is specified within a modify schema entry; otherwise, this clause is optional. When this clause is specified, it must immediately follow the MODIFY SCHEMA clause.

The lfn must be from one through seven characters in length, must contain only letters and digits, and must begin with a letter. The lfn must be unique in the modify run (that is, in the source input to the DBMSTRD utility). The file that contains the schema directory must be attached in the job stream with this lfn before execution of the DBMSTRD control statement.

A period must terminate the FILE NAME clause, if it is specifed; otherwise the period must terminate the MODIFY SCHEMA clause.

END Clause

The END clause is required to terminate the modification entry. The format of the end subentry is shown in the modify schema entry format (figure 8-12). Either the keyword MODIFICATIONS or the keyword MODS can be specified.

Changes to the new master directory, including any new CST (Condensed Schema/Subschema Table) that is built for CDCS, are not placed in the new master directory until the END clause is detected.

Change Procedure Library Subentry

The change procedure library subentry changes permanent file information for the procedure library file. This subentry is optional and can be specified only once within a modify schema entry. The format of the change procedure library subentry is shown in figure 8-13.

```
CHANGE PROCEDURE LIBRARY

     permanent file information subentry.
```

Figure 8-13. Change Procedure Library Subentry Format

The change procedure library subentry can be specified only if information about the procedure library file exists in the old master directory. If this subentry is specified for a schema with no procedure library information or is specified more than once in the modify schema entry, a fatal syntax error occurs in the DBMSTRD run.

The permanent file information for the procedure library file must be specified as described in the Permanent File Information Subentry subsection. The combination of permanent file information for permanent file name, UN/ID, and family/set/pack must not be duplicated in the old master directory or in the source input to the DBMSTRD utility with the exception of information being deleted in the modification run. The additional permanent file information must be sufficient to enable CDCS to attach the permanent file with the correct permission.

The information specified in this subentry replaces the information for the procedure library file in the old master directory. Only the permanent file information that is explicitly included in the change procedure library subentry is included in the new master directory after the DBMSTRD run. The omission of information contained in the old master directory about any optional clauses (namely, PW, FAMILY, PACK, DEVICE, SET, or VSN) in effect deletes that information in the new master directory.

The DBMSTRD utility does not require that the schema or procedure library file be available during a modification run to change the procedure library permanent file information.

Change Transaction Recovery File Subentry

The change transaction recovery file subentry changes permanent file information for the transaction recovery file. This subentry is optional and can be specified only once within a modify schema entry. The format of the change transaction recovery file subentry is shown in figure 8-14.

```
CHANGE TRANSACTION RECOVERY FILE

     [permanent file information subentry]

     [UNIT LIMIT IS integer-1]

     [UPDATE LIMIT IS integer-2]                .
```

Figure 8-14. Change Transaction Recovery File Subentry Format

The change transaction recovery file subentry can be specified only if information about the transaction recovery file for the specified schema exists in the old master directory. If this subentry is specified for a schema with no transaction recovery file selected or is specified more than once in the modify schema entry, a fatal syntax error occurs in the DBMSTRD run.

Three kinds of information can be changed by use of this subentry. The information changed depends on the optional clauses specified. The options include permanent file information (permanent file information subentry), unit limit (UNIT LIMIT clause), and update limit (UPDATE LIMIT clause). Each optional clause can be specified only once; the clauses can be specified in any order. Any one clause or all of the clauses can be specified in the subentry. The specification of any one of the clauses does not affect the existing information in the old master directory associated with any other optional clause of this subentry; the information for other clauses is transferred to the new master directory.

CHANGE TRANSACTION RECOVERY FILE Clause

The CHANGE TRANSACTION RECOVERY FILE clause is required within the subentry. This clause identifies the subentry.

Permanent File Information Subentry

The permanent file information subentry is required to change permanent file information for the transaction recovery file. The permanent file information for the transaction recovery file must be specified as described in the Permanent File Information Subentry subsection. The literal that specifies the permanent file name must follow the rules for permanent file names except that it must contain exactly six characters. The permanent file name that CDCS assumes for the transaction recovery file begins with these six characters; the number 1 is the seventh character. The combination of permanent file information for permanent file name, UN/ID, and family/set/pack must not be duplicated in the old master directory or in the source input to the DBMSTRD utility with the exception of information being deleted in the modification run. The additional permanent file information must be sufficient to enable CDCS to attach the permanent file with the correct permission.

The permanent file information specified in the permanent file information subentry replaces the information for the transaction recovery file in the old master directory. Only the permanent file information that is explicitly included in the permanent file information subentry is included in the new master directory after the DBMSTRD run. The omission of information for any optional clause (namely, PW, FAMILY, PACK, DEVICE, SET, or VSN) that is in the old master directory, in effect, deletes that information in the new master directory.

The DBMSTRD utility does not require that the schema or transaction recovery file be available during a modification run to change the permanent file information.

UNIT LIMIT Clause

The UNIT LIMIT clause is required to change the maximum number of outstanding transactions that are allowed for a schema. The information specified in this clause replaces the information designating unit limit in the old master directory. The integer-1 specifies the limit; it must be a positive integer with a maximum value of 63. If this clause is not specified in the change transaction recovery file subentry, the information in the old master directory is transferred to the new master directory.

UPDATE LIMIT Clause

The UPDATE LIMIT clause is required to change the maximum number of updates that can be made within a particular transaction. The information specified in this clause replaces the information designating update limit in the old master directory. The integer-2 specifies the maximum number of updates; it must be a positive integer with a maximum value of 63. If this clause is not specified in the change transaction recovery file subentry, the information in the old master directory is transferred to the new master directory.

Change Restart Identifier File Subentry

The change restart identifier file subentry changes permanent file information for the restart identifier file. This subentry is optional and can be specified only once within a modify schema entry. The format of the change restart identifier file subentry is shown in figure 8-15.

```
CHANGE RESTART IDENTIFIER FILE

    permanent file information subentry.
```

Figure 8-15. Change Restart Identifier File Subentry Format

The change restart identifier file subentry can be specified only if information about the restart identifier file for the specified schema exists in the old master directory. If this subentry is specified for a schema with no restart identifier file selected or is specified more than once in the modify schema entry, a fatal syntax error occurs in the DBMSTRD run.

The permanent file information for the restart identifier file must be specified as described in the Permanent File Information Subentry subsection. The combination of permanent file information for permanent file name, UN/ID, and family/set/pack must not be duplicated in the old master directory or in the source input to the DBMSTRD utility with the exception of information being deleted in the modification run. The additional permanent file information must be sufficient to enable CDCS to attach the permanent file with the correct permission.

The information specified in the permanent file information subentry replaces the information for the restart identifier file in the old master directory. Only the permanent file information that is explicitly included in this subentry is included in the new master directory after the DBMSTRD run. The omission of information for any optional clause (namely, PW, FAMILY, PACK, DEVICE, SET, or VSN) that is in the old master directory, in effect, deletes that information in the new master directory.

The DBMSTRD utility does not require that the schema or restart identifier file be available during a modification run to change the permanent file information.

Change Journal Log File Subentry

The change journal log file subentry changes permanent file information for the journal log file. This subentry is optional and can be specified only once within a modify schema entry. The format of the change journal log file subentry is shown in figure 8-16.

```
CHANGE JOURNAL LOG FILE

    permanent file information subentry.
```

Figure 8-16. Change Journal Log File Subentry Format

The change journal log file subentry can be specified only if information about the journal log file for the specified schema exists in the old master directory. If this subentry is specified for a schema with no journal log file selected or is specified more than once in the modify schema entry, a fatal syntax error occurs in the DBMSTRD run.

The permanent file information for the journal log file must be specified as described in the Permanent File Information Subentry subsection. The literal that specifies the permanent file name must follow the rules for permanent file names except that it must contain exactly six characters. The permanent file name that CDCS assumes for the journal log files begins with these six characters. For one file, the number 1 is the seventh character; for the other file, the number 2 is the seventh character. The combination of permanent file information for permanent file name, UN/ID, and family/set/pack must not be duplicated in the old master directory or in the source input to the DBMSTRD utility with the exception of information being deleted in the modification run. The additional permanent file information must be sufficient to enable CDCS to attach the permanent file with the correct permission.

The information specified in this subentry replaces the information for the journal log file in the old master directory. Only the permanent file information that is explicitly included in this subentry is included in the new master directory after the DBMSTRD run. The omission of information for any optional clause (namely, PW, FAMILY, PACK, DEVICE, SET, or VSN) that is in the old master directory, in effect, deletes that information in the new master directory.

The DBMSTRD utility does not require that the schema or journal log file be available during a modification run to change the permanent file information.

Change Quick Recovery File Subentry

The change quick recovery file subentry changes permanent file information for the quick recovery file. This subentry is optional and can be specified only once within a modify schema entry. The format of the change quick recovery file subentry is shown in figure 8-17.

```
CHANGE QUICK RECOVERY FILE

    permanent file information subentry.
```

Figure 8-17. Change Quick Recovery File Subentry Format

The change quick recovery file subentry can be specified only if information about the quick recovery file for the specified schema exists in the old master directory. If this subentry is specified for a schema with no quick recovery file selected or is specified more than once in the modify schema entry, a fatal syntax error occurs in the DBMSTRD run.

The permanent file information for the quick recovery file must be specified as described in the Permanent File Information Subentry subsection. The combination of permanent file information for permanent file name, UN/ID, and family/set/pack must not be duplicated in the old master directory or in the source input to the DBMSTRD utility with the exception of information being deleted in the modification run. The additional permanent file information must be sufficient to enable CDCS to attach the permanent file with the correct permission.

The information specified in this subentry replaces the information for the quick recovery file in the old master directory. Only the permanent file information that is explicitly included in this subentry is included in the new master directory after the DBMSTRD run. The omission of information for any optional clause (namely, PW, FAMILY, PACK, DEVICE, SET, or VSN) that is in the old master directory, in effect, deletes that information in the new master directory.

The DBMSTRD utility does not require that the schema or quick recovery file be available during a modification run to change the permanent file information.

Change Job Control Information Subentry

The change job control information subentry changes job control information for the CDCS-initiated job that dumps the journal log file. This subentry is optional and can be specified only once within a modify schema entry. The format of the change job control information subentry is shown in figure 8-18.

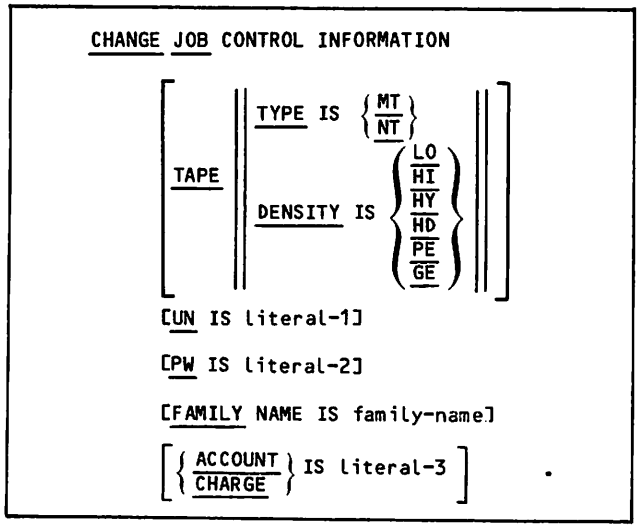


Figure 8-18. Change Job Control Information Subentry Format

This subentry can be specified only if job control information already exists for the specified schema in the old master directory. If this subentry is specified for a schema with no job control information specified or is specified more than once in the modify schema entry, a fatal syntax error occurs in the DBMSTRD run.

The information provided by use of this subentry replaces information in the old master directory.

The information changed depends on the optional clauses specified in the entry. Each optional clause can be specified once. Any one or all of the clauses can be specified. Only the information that is explicitly included in the optional clauses is included in the new master directory after the DBMSTRD run. The omission of a clause in effect deletes that information in the new master directory.

The clauses that are applicable depend on the operating system. All clauses are applicable for use on NOS. Only the TAPE and CHARGE/ACCOUNT clauses are applicable for use on NOS/BE. For the description of the information provided by these clauses and for the applicability of clauses to the operating system, refer to the JOB CONTROL INFORMATION Clause subsection that appears earlier in this section.

Change Area Subentry

The change area subentry in the modify schema entry changes logging options selected for the area and changes permanent file information for an area and index file. The VERSION clause of the subentry identifies the version affected by the change area subentry. The information specified in the change area subentry replaces the information for the indicated version and area in the old master directory. The format of the change area subentry is shown in figure 8-19.

The change area subentry can be specified only if information about the version and area is included in the old master directory. The information can be explicit or implicit. Explicit information is included because the information was provided in an area subentry. Implicit information is included because area information was either specified or assumed to be the same as master.

The combination of version name and area name specified in the change area subentry cannot be duplicated in another change area subentry in source input to DBMSTRD. If a change area subentry specifies a change in permanent file information for an area or version that is not included in the old master directory, a fatal syntax error occurs in the DBMSTRD run.

The change area subentry cannot be used to change the area name in the old master directory; this type of change requires modification to a DDL schema.

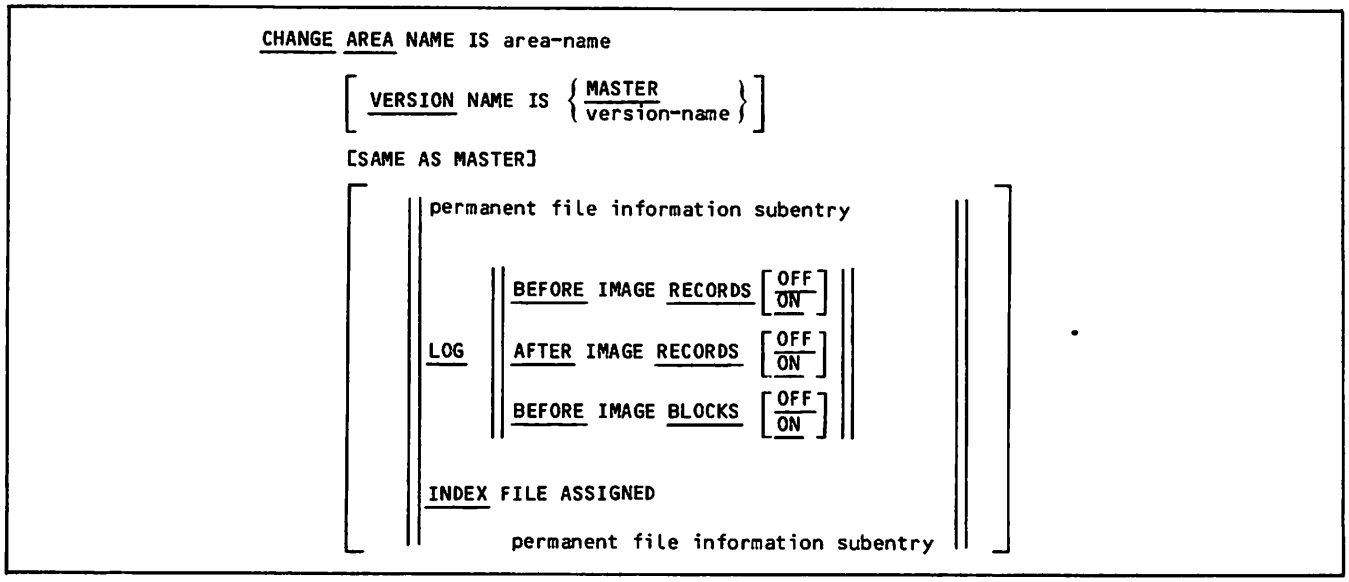


Figure 8-19. Change Area Subentry Format

The change area subentry allows the specification of area information to be the SAME AS MASTER or to be different from master. If the information is different, any or all of the following options can be specified to change area information: permanent file information for the area file (permanent file subentry for area file), permanent file information for the index file (INDEX clause), and logging for the area (LOG clause). If the above options are not specified, the information pertaining to the area in the old master directory is retained in the new master directory.

The change area subentry does not affect the area identification that is assigned by the DBMSTRD utility and listed in the second section of the DBMSTRD output. An area retains the identification number assigned when the area was previously specified for inclusion in the old master directory.

The DBMSTRD utility does not require that the schema or area and index files be available during a modification run to change area or index permanent file information or logging options.

CHANGE AREA Clause

The CHANGE AREA clause (figure 8-19) is required within the change area subentry. The area name identifies the area being changed. The area name must be an area specified in the old master directory for the schema. An area name must be 1 through 30 characters in length.

VERSION NAME Clause (Change Area Subentry)

The VERSION NAME clause (figure 8-19) specifies the name of the version that is associated with the area. This clause is optional. If this clause is not specified, version name MASTER is assumed.

The version name must already be specified in the old master directory. The restrictions in specifying a particular version name and associated area name are described in preceding paragraphs. The version name must not be duplicated in an add version subentry or in a delete version subentry.

SAME AS MASTER Clause (Change Area Subentry)

The SAME AS MASTER clause (figure 8-19) replaces all existing information for the area of the indicated version with the information associated with the same area for version MASTER. This clause is optional.

If this clause is specified, the following options cannot be specified: permanent file information subentry, LOG clause, and INDEX clause.

Permanent File Information Subentry (Change Area Subentry)

The first permanent file information subentry shown in the format (figure 8-19) causes the existing permanent file information for the area file to be replaced with the information specified in the subentry. This subentry is optional.

The permanent file information for the area data file must be specified as described in the Permanent File Information Subentry subsection. The combination of permanent file information for permanent file name, UN/ID, and family/set/pack must not be duplicated in the old master directory or in the source input to the DBMSTRD utility with the exception of information being deleted in the modification run. The additional permanent file information must be sufficient to enable CDCS to attach the permanent file with the correct permission.

Only the permanent file information that is explicitly included in the subentry is included in the new master directory after the DBMSTRD run. The omission of information for any optional clause (namely, PW, FAMILY, PACK, DEVICE, SET, or VSN) that is in the old master directory, in effect, deletes that information in the new master directory.

The specification of this clause has no effect on information associated with the LOG clause or INDEX clause.

LOG Clause (Change Area Subentry)

The LOG clause (figure 8-19) can cancel logging options selected in the old master directory and select new logging options. This clause is optional.

The LOG clause specifies the types of logging to be performed for the area. Any combination of options can be selected. The BEFORE IMAGE RECORDS option selects logging of before-image records to a journal log file. (A before-image record is a copy of a record before its modification.) The AFTER IMAGE RECORDS option specifies logging of after-image records to a journal log file. (An after-image record is a copy of a record after its modification.) The BEFORE IMAGE BLOCKS option selects block logging to a quick recovery file. Refer to section 9 for details on the journal log file, the quick recovery file, and the logging options that must be selected for different recovery capabilities.

To cancel the LOG options selected for an area in the old master directory, the keyword OFF must be specified for the same option in the source input to DBMSTRD.

To select new logging options, the option must be specified. The keyword ON does not have to be specified but can be for documentation purposes.

If no LOG options are specified in a change area subentry, the logging requirements specified in the old master directory remain in effect. The specification of the LOG clause has no effect on information associated with the INDEX clause or with the permanent file information subentry for the area file.

INDEX Clause (Change Area Subentry)

The INDEX clause (figure 8-19) causes the existing permanent file information for the index file to be replaced with the information specified in the permanent file information subentry. This clause is optional.

The permanent file information for the index file must be specified as described in the Permanent File Information Subentry subsection. The combination of permanent file information for permanent file name, UN/ID, and family/set/pack must not be duplicated in the old master directory or in the source input to the DBMSTRD utility with the exception of information being deleted in the modification run. The additional permanent file information must be sufficient to enable CDCS to attach the permanent file with the correct permission.

Only the permanent file information that is explicitly included in the INDEX clause and in the permanent file information subentry is included in the new master directory after the DBMSTRD run. The omission of information for any optional clause (namely, PW, FAMILY, PACK, DEVICE, SET, or VSN) that is in the old master directory, in effect, deletes that information in the new master directory.

The specification of this clause has no effect on information associated with the permanent file information subentry or with the LOG clause.

Delete Version Subentry

The delete version subentry specifies the name of a version to be deleted from the master directory. All area information associated with the version is also deleted. The format of the delete version subentry is shown in figure 8-20.

```
DELETE VERSION NAME IS version-name.
```

Figure 8-20. Delete Version Subentry Format

The DELETE VERSION clause identifies the version being deleted. This clause is required within the subentry.

The version name must specify an alternate version that already exists in the old master directory; version MASTER cannot be specified. The version name must not duplicate a version name specified in a change area subentry or in a preceding add version subentry. The version name can be specified in a following add version subentry.

Add Version Subentry

The add version subentry designates the name of an alternate data base version to be added to a data base. This subentry also associates each area of the schema with logging requirements and either a permanent file or two permanent files if an index file is required for the area. The format of the add version subentry is shown in figure 8-21.

A version name specified in this subentry must not be associated with the same schema in the old master directory or in the source input to DBMSTRD with one exception: the version name can appear in a preceding delete version subentry. The version name MASTER cannot be added.

The add version subentry is the same as the alternate version subentry with one exception: the keyword ADD precedes the keywords VERSION NAME IS. Therefore, the description of the alternate version subentry applies to this subentry with the exception of the restriction on version name described in the preceding paragraph. Refer to the subsection Alternate Version Subentry for the description of the clauses.

Delete Subschema Subentry

The delete subschema subentry specifies a subschema to be deleted from the master directory. The format of the delete subschema subentry is shown in figure 8-22.

```
DELETE SUBSCHEMA NAME IS subschema-name.
```

Figure 8-22. Delete Subschema Subentry Format

The DELETE SUBSCHEMA clause identifies the subschema being deleted. This clause is required within the subentry.

The subschema name must specify a subschema that is already included in the old master directory.

Add Subschema Subentry

The add subschema subentry specifies a subschema to be added to a data base. The format of the add subschema subentry is shown in figure 8-23.

```
ADD SUBSCHEMA NAME IS subschema-name
    FILE NAME IS lfn.
```

Figure 8-23. Add Subschema Subentry Format

```
ADD VERSION NAME IS version-name
    { ,                                                  }
    { | AREA NAME IS area-name SAME AS MASTER. | ...     }
      | area subentry
```

Figure 8-21. Add Version Subentry Format

The ADD SUBSCHEMA NAME clause identifies the subschema being added. This clause is required.

The subschema name must not duplicate a subschema name already associated with the same schema in the old master directory unless the same subschema is being deleted in a preceding delete subschema subentry. The subschema name must be the same subschema name specified in the subschema directory and listed on the subschema compilation output listing. A subschema name must be from 1 through 30 characters in length.

The FILE NAME clause specifies the local file name (lfn) of the subschema directory or the subschema library. This clause is required.

The lfn can be from one through seven characters in length, must contain only letters and digits, and must begin with a letter. The file that contains the subschema directory or subschema library must be attached in the job stream with this lfn before the execution of the DBMSTRD control statement.

MASTER DIRECTORY GENERATION

The master directory is generated by the DBMSTRD utility, which is called by the DBMSTRD control statement. The DBMSTRD utility generates a master directory under two different conditions: a creation run and a modification run.

DBMSTRD CONTROL STATEMENT

The DBMSTRD control statement is used to request the creation or modification of a master directory and to provide the DBMSTRD utility with the file names needed in the run.

The format of the DBMSTRD control statement is shown in figure 8-24. Optionally, the comma immediately following DBMSTRD can be replaced by a left parenthesis; the terminating period can be replaced by a right parenthesis.

All parameters of the control statement are optional. The parameters can appear in any order.

The I parameter identifies the local file that contains the source input for the utility. The I parameter is interpreted as follows:

omitted

Local file INPUT is assumed to contain the source input for the utility.

I

Local file COMPILE is assumed to contain the source input for the utility.

I=lfn-1

The specified file name identifies the local file that contains the source input for the utility.

I=0

Not allowed.

The L parameter identifies the local file that receives listings and diagnostics generated by the utility. The L parameter is interpreted as follows:

omitted

Local file OUTPUT receives the listings and diagnostics generated by the utility.

L

Local file OUTPUT receives the listings and diagnostics generated by the utility.

L=lfn-2

The specified local file name identifies the file that receives the listings and diagnostics generated by the utility.

L=0

Local file OUTPUT receives only warning and error messages. A listing is not produced.

The LD parameter controls generation of the master directory contents report that is produced at the end of the run. This parameter is interpreted as follows:

omitted

No master directory contents report is generated.

LD

The master directory contents report is written to the file specified by the L parameter.

The OMD parameter identifies the local file that contains the old (existing) master directory. The OMD parameter is interpreted as follows:

omitted

Local file OMSTRDR is assumed to contain the old master directory.

OMD

Local file OMSTRDR is assumed to contain the old master directory.

```
DBMSTRD[,I=lfn-1][,L=lfn-2][,LD][,OMD=lfn-3][,NMD=lfn-4].
```

Figure 8-24. DBMSTRD Utility Control Statement

OMD=lfn-3

The specified file name identifies the local file that contains the old master directory.

The NMD parameter identifies the local file that contains the master directory generated by the current run (the new master directory). The NMD parameter is interpreted as follows:

omitted

Local file NMSTRDR is assumed to contain the new master directory.

NMD

Local file NMSTRDR is assumed to contain the new master directory.

NMD=lfn-4

The specified file name identifies the local file that contains the new master directory.

CREATION RUN

To create the master directory, the schema directories and the subschema directories of any schema and subschema specified in the source input for the master directory must be available to the DBMSTRD utility. The file containing the schema directory must be attached in the job stream by the local file name specified in the creation entry of the source input. Similarly, the file containing the subschema library or the particular files that contain the subschema directories must be attached in the job stream with the local file names specified in the creation entry of source input.

The DBMSTRD control statement must be included in the set of control statements to generate the master directory. It provides the DBMSTRD utility with the local file name of the source input file and the local file name of the master directory that is generated. The DBMSTRD control statement for master directory creation is as follows:

```
DBMSTRD,LD,NMD=lfn.
```

The LD parameter and the NMD parameter are optional and have default values. The I and L parameters must be specified if default values are not applicable for input and output files (refer to the DBMSTRD Control Statement subsection).

The data administrator is responsible for ensuring that the master directory file generated by the DBMSTRD utility and contained in the file identified by the NMD parameter is made permanent. For use of the master directory file under NOS, the permanent file usually has to be made accessible to the user number under which CDCS or some other job that attaches the master directory executes. The file can be made accessible to other user numbers by execution of the PERMIT control statement. For example, if the user number of the CDCS job is not the owner of the master directory file, either the PERMIT control statement designating access to that user number must be executed or the master directory file must be made a public file (CT=PU).

MODIFICATION RUN

To modify the master directory, the existing master directory to be modified (called the old master directory) must be made available to the DBMSTRD utility. Therefore, the old master directory must be attached (read permission is required) in the job stream with the local file name specified by the OMD parameter of the DBMSTRD control statement. The file containing the schema directory must be available to the DBMSTRD utility if a schema is being added to the master directory or if a schema is being modified by having subschemas added. Similarly, the files containing the subschema directories must be available to the DBMSTRD utility if subschemas are being added to the master directory either through an add schemas entry or through a modify schemas entry. If the schema directory is needed, the file containing the schema directory must be attached in the job stream by the local file name specified in the source input. If the subschema directory is needed, the file containing the subschema library or the subschema directory must be attached in the job stream by the local file name specified in the source input.

The DBMSTRD control statement provides the DBMSTRD utility with the local file name of the existing master directory being modified (the old master directory), the local file name of the source input file, and the local file name of the master directory that is generated (the new master directory). The DBMSTRD control statement for master directory creation is as follows:

```
DBMSTRD,LD,OMD=lfn,NMD=lfn.
```

The LD parameter, the OMD parameter, and the NMD parameter are optional and have default values. The I and L parameters must be specified if default values are not applicable for input and output files (refer to the DBMSTRD control statement subsection).

The data administrator must ensure that the master directory file generated by the DBMSTRD utility and contained in the file identified by the NMD parameter is made permanent. For use of the master directory file under NOS, the permanent file usually has to be made accessible to the user number under which CDCS or some other job that attaches the master directory executes. The file can be made accessible to other user numbers by execution of the PERMIT control statement. For example, if the user number of the CDCS job is not the owner of the master directory file, either the PERMIT control statement designating access to that user number must be executed or the master directory file must be made a public file (CT=PU).

DBMSTRD OUTPUT

Output from the DBMSTRD utility is in two sections. The first section consists of a listing of the input file with any syntax errors identified and statistics about the CST (Condensed Schema/SubSchema Table) for each subschema for which information is added in the run. The second section, called the master directory contents report, lists the schemas, versions, areas, relations, and subschemas for which information exists in the

master directory; the checksum values for the areas, relations, and subschemas; and statistical information about the master directory. The second section appears only if the LD option was specified in the DBMSTRD utility control statement.

The syntax listing and CST statistics section indicates each card image read preceded by a sequence number. Syntax error messages in the listing refer to the sequence number of the line on which the error occurred. CST statistics appear at the end of the section for each schema.

The CST statistics can help the data administrator estimate the field length required by CDCS to service applications programs using a particular combination of schema and subschema. The number of words required in the CST for each portion of statistical information can indicate the CDCS memory overhead for use of various features and can help the data administrator determine ways to decrease memory requirements. The area, constraint, record, and relation entries increase in size when an increase occurs in the number and/or complexity of areas/realms, constraints, records, and relations defined in the schema or the subschema.

Within the CST statistics, a lock entry refers to access control locks or procedures defined in the schema. A procedure entry is included for each occurrence of a data base procedure at the area or record level. Area capsule entries indicate the size of the mapping capsules required for mapping of keys or relation data names. Record capsule entries indicate the size of the mapping capsules required for mapping of record items. No statistics are given for data base procedure capsules since they are not part of the CST. Figure 8-25 shows a sample of the first section of DBMSTRD output for a creation run.

The optional second section, the master directory contents report, consists of one entry for each schema for which information is contained in the master directory. The following information is given:

Schema name and identification assigned by DBMSTRD.

Version name MASTER.

Names of all areas in the schema, along with the identifications assigned by DBMSTRD and the area checksums assigned by DDL.

Names of additional versions with every version name followed by area information. The area information is the same as indicated in the item above and is listed only for those areas that have unique permanent files for that version.

Names of all relations in the schema and the relation checksums assigned by DDL.

Names of all subschemas for the schema and the subschema checksums assigned by DDL.

Number of words used for all master directory information.

The area, relation, and subschema checksums listed in the DBMSTRD output can be used in conjunction with DDL output listings to determine which subschemas and application programs must be recompiled as a result of the recompilation of a schema. A checksum is a one-word attribute generated by the DDL compiler for each area and relation in the schema and for each subschema. When a schema for which information exists in the master directory is recompiled, the data administrator should compare the area and relation checksums in the version of the schema that is already in the master directory with the checksums of the newly compiled schema. If the checksums do not agree, all subschemas referencing the elements with changed checksum values must be recompiled. (The DDL3 and DDLF compilers can be used to determine if subschemas in a subschema library need recompilation; refer to sections 3 and 4 for more information.) Before an applications program using a subschema is executed, the checksum of the subschema used to compile the program should be compared with the subschema checksum recorded in the master directory to verify that the program is consistent with the subschema it references. In addition, when a schema or subschema has been recompiled many times, the checksums listed in the DBMSTRD output can be used to determine which version of the schema or subschema is currently being used.

Figure 8-26 shows the master directory contents report generated by the DBMSTRD utility during a creation run.

```
DBMSTRD SOURCE LISTING   NMD LFN=MSTRDIR    OMD LFN=OMSTRDR  DBMSTRD V2.3(82061)

      1          SCHEMA NAME IS MANUFACTURING-DB
      2                FILE NAME IS MANUFAC
      3
      4                PROCEDURE LIBRARY
      5                    PFN IS "DB1PLIB" UN IS "CDCS23"
      6                TRANSACTION RECOVERY FILE
      7                    PFN IS "DB1TRF" UN IS "CDCS23"
      8                    UNIT LIMIT IS 50
      9                    UPDATE LIMIT IS 15
```

Figure 8-25. Sample First Section of DBMSTRD Output (Sheet 1 of 3)

```
10                  RESTART IDENTIFIER FILE
11                      PFN IS "DB1RIF" UN IS "CDCS23"
12                  JOURNAL LOG FILE
13                      PFN IS "DB1JLF" UN IS "CDCS23"
14                  QUICK RECOVERY FILE
15                      PFN IS "DB1QRF" UN IS "CDCS23"
16                  JOB CONTROL INFORMATION
17                      TAPE TYPE IS NT
18                           DENSITY IS PE
19                      UN IS "CDCS23"
20                      CHARGE IS "1982CHG".
21
22          VERSION NAME IS MASTER
23
24              AREA NAME IS EMPLOYEE
25                  PFN IS "MEMPL" UN IS "CDCS23"
26                      PW IS "OKCDCS2"
27                  LOG BEFORE IMAGE BLOCKS
28                      AFTER IMAGE RECORDS
29                  INDEX FILE ASSIGNED
30                      PFN "MXEMPL" UN IS "CDCS23".
31
32              AREA NAME IS JOBDETAIL
33                  PFN IS "MJOBD" UN IS "CDCS23"
34                      PW IS "OKCDCS2"
35                  LOG BEFORE IMAGE BLOCKS
36                      AFTER IMAGE RECORDS.
37
38              AREA NAME IS DEPARTMENTS
39                  PFN IS "MDEPT" UN IS "CDCS23"
40                  LOG BEFORE IMAGE BLOCKS
41                  INDEX FILE ASSIGNED
42                      PFN "MXDEPT" UN IS "CDCS23".
43
44              AREA NAME IS PROJECT
45                  PFN IS "MPROJ" UN IS "CDCS23"
46                  LOG BEFORE IMAGE BLOCKS
47                      AFTER IMAGE RECORDS
48                  INDEX FILE ASSIGNED
49                      PFN "MXPROJ" UN IS "CDCS23".
50
51              AREA NAME IS DEVELOPMENT-PRODUCTS
52                  PFN IS "MDEVE" UN IS "CDCS23"
53                  LOG BEFORE IMAGE BLOCKS
54                  INDEX FILE ASSIGNED
55                      PFN IS "MXDEVE" UN IS "CDCS23".
56
57              AREA IS TESTS
58                  PFN IS "MTEST" UN IS "CDCS23"
59                  LOG BEFORE IMAGE BLOCKS
60                      BEFORE IMAGE RECORDS
61                      AFTER IMAGE RECORDS
62                  INDEX FILE ASSIGNED
63                      PFN "MXTEST" UN IS "CDCS23".
64
65              AREA NAME IS CDCSCAT
66                  PFN IS "MQCAT" UN IS "CDCS23".
67
68          VERSION NAME IS TESTVRS
69
70              AREA NAME IS EMPLOYEE
71                  SAME AS MASTER.
72
73              AREA NAME IS JOBDETAIL
74                  PFN IS "TJOBD" UN IS "CDCS23".
75
```

Figure 8-25. Sample First Section of DBMSTRD Output (Sheet 2 of 3)

```
                    76              AREA NAME IS DEPARTMENTS
                    77                  SAME AS MASTER.
                    78
                    79              AREA NAME IS PROJECT
                    80                  PFN IS "TPROJ" UN IS "CDCS23"
                    81                  INDEX FILE ASSIGNED
                    82                      PFN "TXPROJ" UN IS "CDCS23".
                    83
                    84              AREA IS DEVELOPMENT-PRODUCTS
                    85                  PFN IS "TDEVE" UN IS "CDCS23"
                    86                  INDEX FILE ASSIGNED
                    87                      PFN "TXDEVE" UN IS "CDCS23".
                    88
                    89              AREA IS TESTS
                    90                  PFN IS "TTEST" UN IS "CDCS23"
                    91                  INDEX FILE ASSIGNED
                    92                      PFN "TXTEST" UN IS "CDCS23".
                    93
                    94              AREA NAME IS CDCSCAT
                    95                  PFN IS "TQCAT" UN IS "CDCS23".
                    96
                    97          SUBSCHEMA NAME IS C5SS-PRODUCT-PERSONNEL
                    98                  FILE NAME IS C5SSLIB.
                    99          SUBSCHEMA NAME IS C5SS-PRODUCT-MANAGEMENT
                   100                  FILE NAME IS C5SSLIB.
                   101          SUBSCHEMA NAME IS F4SS-PRODUCT-EVALUATION
                   102                  FILE NAME IS F4SSLIB.
                   103          SUBSCHEMA NAME IS F5SS-PRODUCT-MANAGEMENT
                   104                  FILE NAME IS F5SSLIB.
                   105          SUBSCHEMA NAME IS QUCREA6
                   106                  FILE NAME IS QUSSLIB.
                   107          SUBSCHEMA NAME IS QUPRODMGT
                   108                  FILE NAME IS QUSSLIB.

CST BUILDER STATISTICS FOR THE CST FOR
SUBSCHEMA C5SS-PRODUCT-PERSONNEL
OF SCHEMA MANUFACTURING-DB
CATEGORY     ENTRIES      ----WORDS IN THE CST-----
OF ENTRY     IN  CST      OCTAL    DECIMAL    PERCENT
AREA              3         36         30          3
AREA CAPSULE      1        143         99         10
CONSTRAINT        1          2          2          0
GENERAL           1         23         19          2
LOCK              3         14         12          1
PROCEDURE         2          1          1          0
REC               2         14         12          1
REC CAPSULE       4       1477        831         81
RELATION          1         23         19          2
TOTAL            18       2001       1025        100
THE CST BUILDER USED    0.038 SECONDS

CST BUILDER STATISTICS FOR THE CST FOR
SUBSCHEMA C5SS-PRODUCT-MANAGEMENT
OF SCHEMA MANUFACTURING-DB
                .
                .
                .
THE CST BUILDER USED    0.041 SECONDS

-    DBMSTRD COMPLETE     0 ERRORS

                          0 WARNINGS

     REQUIRED   45500B WORDS SCM
     RUN TIME     0.370 SECONDS
```

Figure 8-25. Sample First Section of DBMSTRD Output (Sheet 3 of 3)

```
MSTRDIR                                  PAGE     1

M A S T E R   D I R E C T O R Y   C O N T E N T S
   (S=SCHEMA, V=VERSION, A=AREA, R=RELATION, SB=SUBSCHEMA)
     NAME                                          ID
-
   S  MANUFACTURING-DB                              1
      CREATION DATE -  82087  TIME -  14.14
    V MASTER
      A  EMPLOYEE                                   1
          CHECKSUM -  56430552040035230620
      A  JOBDETAIL                                  2
          CHECKSUM -  13626262424331006604
      A  DEPARTMENTS                                3
          CHECKSUM -  16574462455232633415
      A  PROJECT                                    4
          CHECKSUM -  35175246562100113225
      A  DEVELOPMENT-PRODUCTS                       5
          CHECKSUM -  61470111011661762112
      A  TESTS                                      6
          CHECKSUM -  40015236474170366115
      A  CDCSCAT                                    7
          CHECKSUM -  57033604647410770643
    V TESTVRS
      A  JOBDETAIL                                  2
          CHECKSUM -  13626262424331006604
      A  PROJECT                                    4
          CHECKSUM -  35175246562100113225
      A  DEVELOPMENT-PRODUCTS                       5
          CHECKSUM -  61470111011661762112
      A  TESTS                                      6
          CHECKSUM -  40015236474170366115
      A  CDCSCAT                                    7
          CHECKSUM -  57033604647410770643
    R    EMP-REL
          CHECKSUM -  33541156402300070013
    R    TEST-REL
          CHECKSUM -  10246503456523070027
    R    DPD-REL
          CHECKSUM -  16223300077042601641
    SB   C5SS-PRODUCT-PERSONNEL
          CHECKSUM -  07267725304252560742
          CREATION DATE - 82087  TIME -  14.19
    SB   C5SS-PRODUCT-MANAGEMENT
          CHECKSUM -  34223316060544764172
          CREATION DATE - 82087  TIME -  14.19
    SB   F4SS-PRODUCT-EVALUATION
          CHECKSUM -  00070537057255605254
          CREATION DATE - 82087  TIME -  14.19
    SB   F5SS-PRODUCT-MANAGEMENT
          CHECKSUM -  16022653644161327023
          CREATION DATE - 82087  TIME -  14.20
    SB   QUCREA6
          CHECKSUM -  16231423336214473450
          CREATION DATE - 82087  TIME -  14.20
    SB   QUPRODMGT
          CHECKSUM -  01703020523731103263
          CREATION DATE - 82087  TIME -  14.20

   SUMMARY FOR MANUFACTURING-DB
      NUMBER OF VERSIONS              2
      NUMBER OF AREAS                 7
      NUMBER OF RELATIONS             3
      NUMBER OF SUBSCHEMAS            6
-
   OVERALL SUMMARY
      NUMBER OF SCHEMAS               1
      NUMBER OF SUBSCHEMAS
      NUMBER OF RELATIONS             3
      NUMBER OF VERSIONS              2
      NUMBER OF AREAS                 7
      DIRECTORY SIZE (WORDS)       8118
```

Figure 8-26. Sample Second Section of DBMSTRD Output (Master Directory Contents)

DATA BASE RECOVERY 9

Data base recovery facilities are grouped into two categories: automatic recovery and manual recovery. Automatic recovery is performed by CDCS and provides for recovery from system failure, for recovery from application program failure, and for restarting application programs after a failure. Manual recovery is performed by the data administrator, who uses data base utilities provided for manual recovery. The manual recovery utilities are intended for use in situations where automatic recovery is not in effect or is incapable of accomplishing the desired recovery. The automatic and manual recovery environments are shown in figures 9-1 and 9-2, respectively.

The recovery facilities are described within the following topics:

- Automatic recovery
- Data base utilities
- Logging
- Recovery considerations

Manual recovery involves recovery considerations and the use of data base utilities. Therefore, manual recovery is discussed in the subsections describing those topics.

Automatic recovery has two phases: the CDCS initialization phase and the application-service phase. The CDCS initialization phase provides for recovery from system failure. The application-service phase provides for data base recovery from application program failure and for restarting application programs after a failure. Automatic recovery depends on data base transactions being performed.

Data base utilities provide for the maintenance of log files and for manual recovery operations. The data base utilities and the functions they perform are as follows:

DBREC utility

> Initializes log files; transfers the contents of a journal log file to a tape file.

DBRCN utility

> Reconstructs a data base by using files reloaded from dumps and by using after-image records in a journal log file.

DBRST utility

> Restores a data base to a previous state by using a current copy of the data base and before-image records in a journal log file.

DBQRFA utility

> Applies the contents of the quick recovery file to a data base before a reconstruction or restoration operation is initiated.

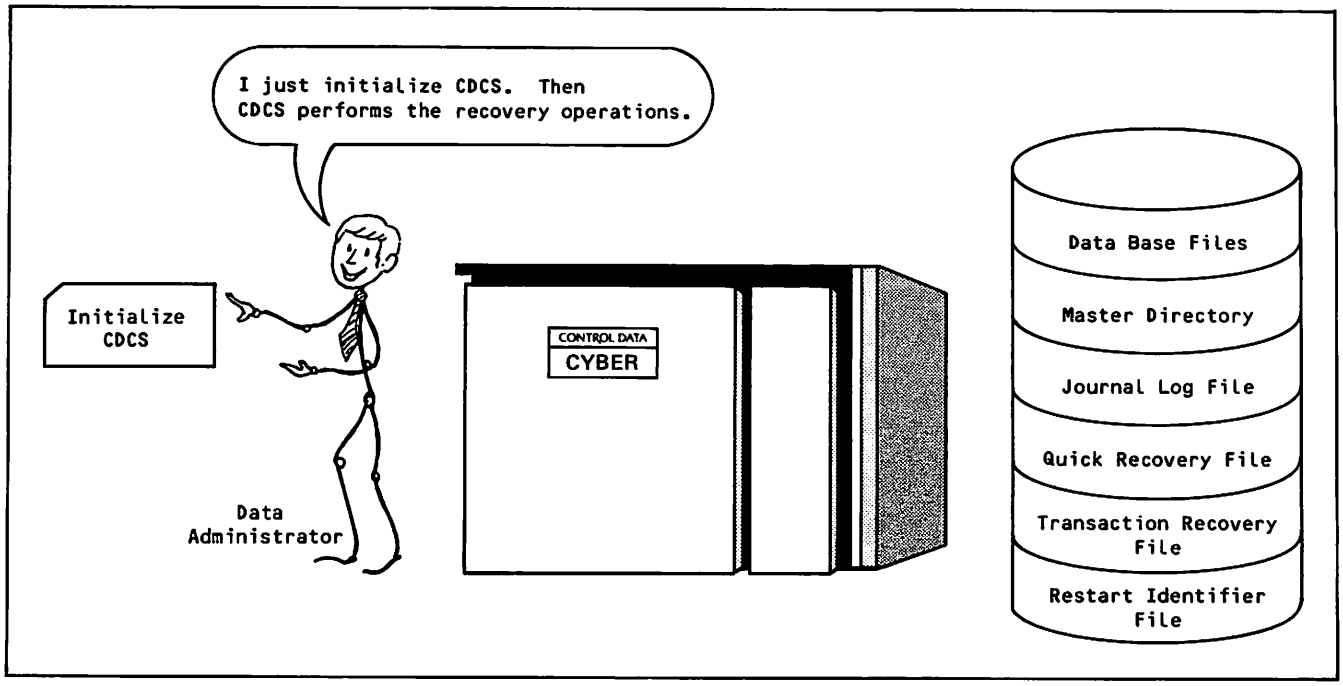


Figure 9-1. Automatic Recovery Environment

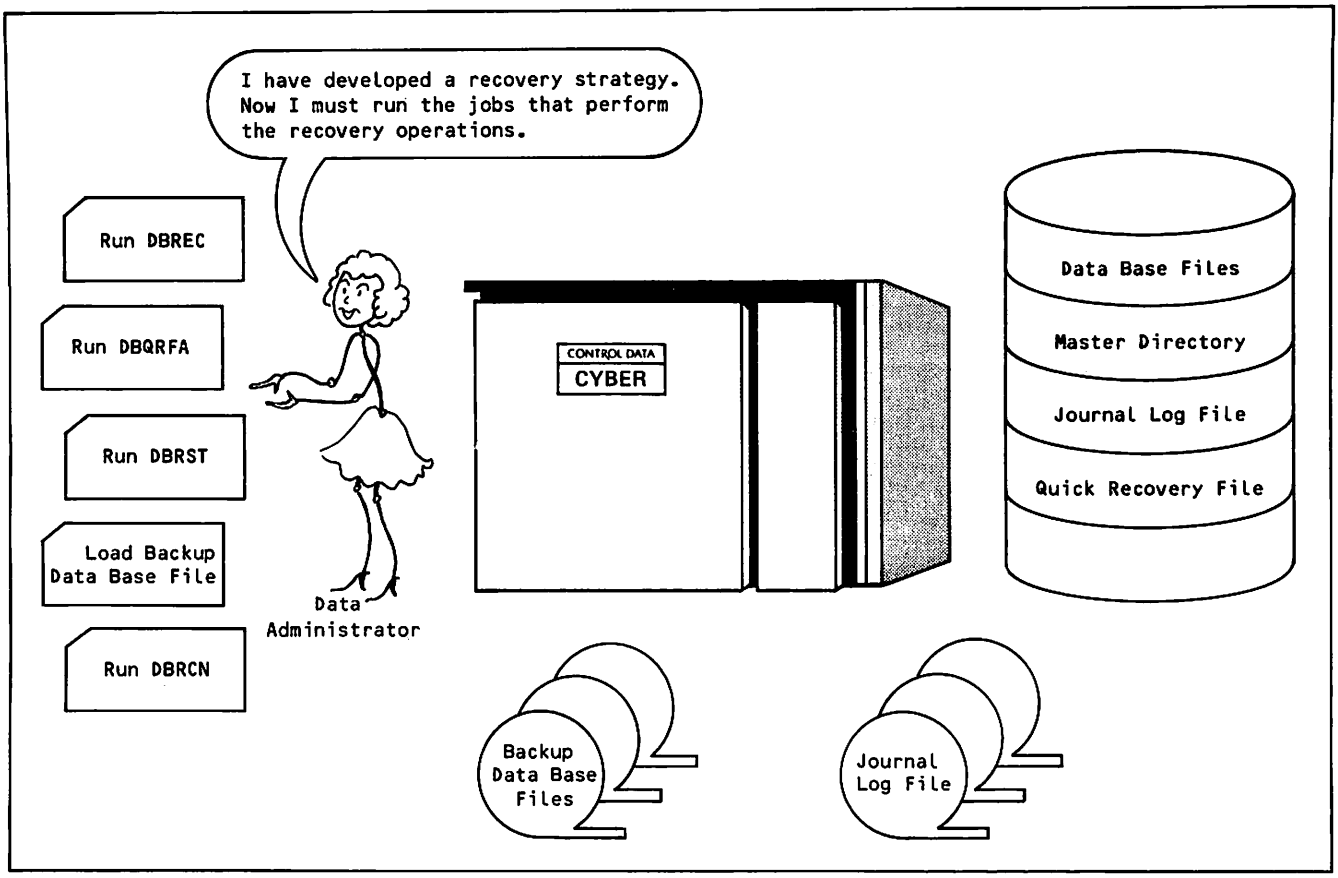


Figure 9-2. Manual Recovery Environment

The log processing component of CDCS records data on the following files (as selected in the master directory):

Journal log file

Contains before and after images (as selected) of records that are updated.

Quick recovery file

Contains before-image records of CRM blocks that have been updated.

Transaction recovery file

Contains information required for reversing incomplete data base transactions.

Restart identifier file

Contains restart identifiers for CDCS run-units and a transaction identifier of the last completed data base transaction.

Each of the preceding log files is on mass storage (disk).

Two application program interfaces to recovery are provided: data base transaction processing and recovery point definition. Transaction processing is a facility of automatic recovery and is described in the Automatic Recovery subsection. Recovery points are written to the journal log file when requested by an application program and initiated by CDCS in particular situations. Recovery points are described in the Logging subsection.

AUTOMATIC RECOVERY

The automatic recovery facility of CDCS permits automatic recovery of a data base (in whole or in part) following a system failure or an application program failure. The automatic recovery facility also provides for automatic maintenance of the journal log file and for restarting application programs. The application program interface to automatic recovery is transaction processing. In summary, the features of automatic recovery are as follows:

Data base transactions

Application program restart

Data base recovery from application program failure

Data base recovery from system failure

Automatic journal log maintenance

Data base logging is required to support automatic recovery. The data administrator must select the logging options by specifying the log and recovery files in the master directory and by initializing the files through the DBREC utility. During application-service phase processing, CDCS performs the logging to the selected files as application programs perform data base updates. For recovery purposes it is recommended that application programs perform updates that involve several associated records with a data base transaction.

The subsections that follow describe the requirements to implement automatic recovery and the Data Administrator's Role in Automatic Recovery.

DATA BASE TRANSACTION

A data base transaction (called, simply, a transaction) is a sequence of one or more updates that usually involve related files. The use of data base transactions is essential for automatic recovery. The data base transaction feature is available to COBOL and FORTRAN application programs. It is not available to Query Update users.

The application program specifies the beginning of the transaction, performs the update operations, and specifies the end of the transaction, which can be either a commit or a drop. With the specification to commit the transaction, CDCS makes the updates permanent. With the specification to drop the transaction, all the updates performed within the transaction are reversed by CDCS; therefore, the data base is restored to its state before the beginning of the transaction. If the application program fails to commit a transaction because of a system or program failure, automatic recovery is performed and the data base is restored to its state before the beginning of the transaction.

For example, a system failure occurs when an application program BETA is performing the third update in a transaction identified as TRAN689. Automatic recovery is performed when CDCS is reinitialized after the system failure. Each of the three records affected by the transaction is restored to its state before the program began processing the transaction.

The data base transaction feature also provides a restart capability for application programs after a system failure. Using this feature, the application program can determine the last transaction that was committed by the program before the failure occurred. For example in the situation just described, program BETA could determine that TRAN688 was the last committed transaction.

COBOL and FORTRAN data manipulation language (DML) statements provide for the operations involved in transaction processing. Five operations are involved in transaction processing. Three of the operations are directly involved in main-line code dealing with updates. Two of the operations are used for restart operations. The transaction processing operations are as follows:

Beginning a transaction

> The application program specifies the beginning of a transaction and communicates a transaction identifier to CDCS. This causes CDCS to begin processing subsequent updates by that program in transaction mode.

Committing a transaction

> The application program specifies the completion of the transaction. This causes all the updates made within the transaction to be made permanent.

Dropping a transaction

> The application program indicates that the current transaction is to be terminated and not committed. This causes each record updated within the transaction to be restored to the state it was in just before the beginning of the transaction.

Obtaining a restart identifier

> Before beginning transaction processing, the application program obtains a restart identifier from CDCS and must save the identifier for subsequent use in a restart operation.

Inquiring about the status of the last transaction

> In a restart operation, the application program communicates to CDCS a restart identifier and obtains the transaction identifier for the last completed transaction associated with that restart identifier.

DATA BASE RECOVERY FROM APPLICATION PROGRAM FAILURE

Recovery from application failure occurs automatically during CDCS application-service phase processing with no effect on other users. CDCS performs any required recovery of data base files before CDCS allows any user access to the records affected by the recovery operation.

CDCS performs the recovery when CDCS detects the termination of the application program from CDCS and the existence of an incompleted transaction. To accomplish recovery, CDCS applies the before-image records in the transaction recovery file to reverse any updates made within the incompleted transaction. This process restores the affected data base files to their respective states just before the program failure occurred.

Requirements must be met to implement automatic recovery. Refer to the subsection entitled Data Administrator's Role in Automatic Recovery for a discussion of the requirements.

RECOVERY FROM SYSTEM FAILURE

Recovery from system failure occurs automatically during CDCS initialization. CDCS performs any required recovery of data base files before CDCS allows any user access to the files. The recovery process involves the following steps for each schema referenced in the master directory:

Application of before-image blocks from the quick recovery file to reverse updates made to the data base since the last recovery point.

Application of after-image records from the journal log file from the last recovery point to the point of failure.

Application of before-image records in the transaction recovery file to reverse any updates made for transactions that were not committed.

This process restores all data bases to their respective states just before the failure occurred. The state of the data base is determined by the last completed update made by an application program processing in regular mode or by the last committed transaction made by an application program processing in transaction mode.

Requirements must be met to implement automatic recovery. Refer to the subsection entitled Data Administrator's Role in Automatic Recovery for a discussion of the requirements.

When the console operator enters a command to UP a schema, that schema is put through the same recovery process as is used for a recovery from a system failure. If the recovery process for the schema is not successful, then the schema is not changed to "up" status.

RECOVERY OF A SINGLE DATA BASE FILE

If there is a malfunction that causes a single database area to be placed in an "error down" status, that area is subjected to the first two steps of automatic recovery. The before-image blocks are applied and then the after-image records are applied.

The intent is that the application of the before-image blocks will restore the area to a usable state. If not, then the application of the after-image records will fail. In that case, the schema will be placed in an "error down" status.

If the recovery is successful, the area is still left in a down status. This forces recognition of the fact that a malfunction has occurred. It is necessary for the console operator to UP the area in order to resume its use.

Any user jobs that had the file open at the time of the malfunction are dropped. If any of those jobs had an uncommitted transaction in progress, those transactions are backed out.

AUTOMATIC JOURNAL LOG MAINTENANCE

The automatic journal log maintenance feature provides two functions:

Automatic switch to an alternate journal log file.

Automatic transfer of the contents of the journal log file to a storage file (called dumping the journal log file).

CDCS initiates these operations when one of the following conditions occurs:

A journal log file is full. This occurs when records have been written to the file and fill its designated length.

A fatal error is detected on the journal log file during a logging operation.

When CDCS detects one of these conditions, CDCS begins logging operations on the alternate journal log file and initiates a job that calls the DBREC utility. DBREC performs the dump operation. When that operation is complete, DBREC automatically reinitializes the journal log file for future use.

Requirements must be met to implement automatic journal log maintenance.

DATA ADMINISTRATOR'S ROLE IN AUTOMATIC RECOVERY

The data administrator selects recovery options by including particular clauses in the master directory that select the log files. Selected recovery options apply to a data base on a schema basis; other recovery options apply only to an area of a particular version. A summary of the clauses that implement features of automatic recovery is shown in table 9-1. The table also indicates that a parameter must be included in the CDCS control statement to provide information for attaching the master directory. For further information about the clauses of the master directory, refer to section 8. For further information about the CDCS control statement, refer to section 10.

Any log and recovery file used in automatic recovery must be initialized before CDCS can use the file. The DBREC utility provides for this initialization. The data administrator must construct the job that calls the DBREC utility and prepare source input for the DBREC utility. The DBREC utility is discussed later in this section.

The data administrator must establish certain size limits for log and recovery files. These limits are established either in master directory clauses or in DBREC clauses. These limits are described in the following items:

Transaction Recovery File

In input for the master directory, establish two limits: unit limit (the maximum number of outstanding transactions for all users of the schema) and update limit (the maximum number of updates allowed within a transaction).

Journal Log File

In input to the DBREC utility, specify the file size in physical record units.

Quick Recovery File

In input to the DBREC utility, specify the file size in physical record units.

One or more messages pertinent to the recovery process can appear in the CDCS dayfile and output file. If a data base cannot be recovered, CDCS downs the schema. When this occurs, the data administrator must manually restore the data base to an operational state.

If the CCLPROC sub-clause is specified in the JOB CONTROL clause of DBMSTRD, then the data administrator must define the corresponding CCL procedure.

DATA BASE UTILITIES

The data base utilities provide for maintenance and use of log and recovery files. The data administrator has the major role in determining the use of the utilities in the data base environment and in constructing the jobs that use the data base utilities. The data base utilities are described in following subsections in the the order listed:

DBREC utility

DBQRFA utility

DBQRFI utility

DBRCN utility

DBRST utility

TABLE 9-1. SUMMARY OF REQUIREMENTS OF DATA BASE DEFINITION FOR FEATURES OF AUTOMATIC RECOVERY

Requirements			Recovery From Application Program Failure	Application Program Restart	Recovery From System Failure	Automatic Journal Log Maintenance	Comments
Parameter or Clause	Location	Scope					
TRANSACTION RECOVERY FILE clause	Master directory	Schema	x	x	x		This clause must be specified for application programs to perform data base transactions.
RESTART IDENTIFIER FILE clause	Master directory	Schema		x			This clause cannot be specified unless a transaction recovery file is specified.
JOURNAL LOG FILE clause	Master directory	Schema			x	x	This clause provides two permanent file names for journal log files. For the automatic switch, both files must be established by the DBREC utility.
QUICK RECOVERY FILE clause	Master directory	Schema			x		The quick recovery file should be selected if a journal log file or a transaction recovery file is selected.
JOB CONTROL INFORMATION clause	Master directory	Schema				x	This clause must be specified for automatic dumping of the journal log file.
LOG AFTER IMAGE RECORDS option	Master directory	Area of a version			x		This option causes logging to the journal log file for the permanent file associated with the area and version. For recovery of all data base file, this option must be specified for each area of each version.
LOG BEFORE IMAGE BLOCKS option	Master directory	Area of a version			x		This option causes logging to the quick recovery file for the permanent file associated with the area and version. For recovery of all data base files, this option must be specified for each area of each version.
Parameters for attaching the master directory	CDCS control statement	All data bases in master directory				x	The parameters must be included for automatic dumping of the journal log file.

DBREC UTILITY

The DBREC utility provides for two operations: transferring the contents of the journal log file to a tape file (called dumping the journal log file) and initializing the files maintained for data base recovery. These files are the journal log file, the quick recovery file, the transaction recovery file, and the restart identifier file.

The operations which the DBREC utility will perform are specified by clauses in a source input file. The utility is called by the DBREC control statement.

The DBREC utility executes as a batch job at its own control point. DBREC can be executed either online or offline. In offline execution, the data administrator constructs a batch job that executes the DBREC utility. In online execution, CDCS initiates a batch job that executes the DBREC utility.

Schema Entry

The source input for the DBREC utility consists of one or more schema entries. Clauses included in the schema entry indicate operations to be performed on log and recovery files for the specified schema. The format of the schema entry is shown in figure 9-3.

The clauses of the schema entry identify the schema and specify any operation to be performed. One or more schemas can be specified in any order in schema entries.

Clauses in the input file have a free-field format. Data appears in columns 1 through 72; columns 73 through 80 are not examined and can be used for sequencing or identification purposes.

The DUMP and ALLOCATE clauses must refer to log or recovery files specified in the master directory for the same schema as identified in the SCHEMA NAME clause. DUMP and ALLOCATE clauses can appear in any order within a schema entry.

SCHEMA NAME Clause

The SCHEMA NAME clause identifies the schema. This must be the first clause specified in the schema entry.

Schema-name must be a valid schema name, 1 to 30 characters in length, that exists in the master directory attached during execution of the DBREC utility. Schema-name must not duplicate any other schema-name specified in the source input to the DBREC utility.

DUMP Clause

The DUMP clause provides for transferring the contents of a journal log file to a tape file and for reinitialization of the journal log file. The DUMP clause can be specified in a job for either online or offline execution. More than one DUMP clause can be specified in a job for offline execution; the job inititiated by CDCS for online execution includes only one DUMP clause.

Log-file-name specified in the DUMP clause is the 7-character permanent file name of the journal log file. Log-file-name must be specified as follows: the first six characters of the name must be the same as the 6-character permanent file name specified for the journal log file in the master directory; the seventh character must be the digit 1 or 2.

If CDCS initiates the job of dumping the journal log file, CDCS determines the appropriate log-file-name. If the data administrator initiates the job of dumping the journal log file, the data administrator must determine the appropriate log-file-name.

After the DBREC utility has dumped the journal log to tape, the DBREC utility causes the journal log file to be reinitialized as if that file were specified in an ALLOCATE clause.

ALLOCATE Clause

The ALLOCATE clause provides for the initialization of a specified file. If the file is a journal log, quick recovery, or transaction recovery file, storage is allocated for it during intialization. This process is called preallocation.

The ALLOCATE clause can be specified only for a job executing offline.

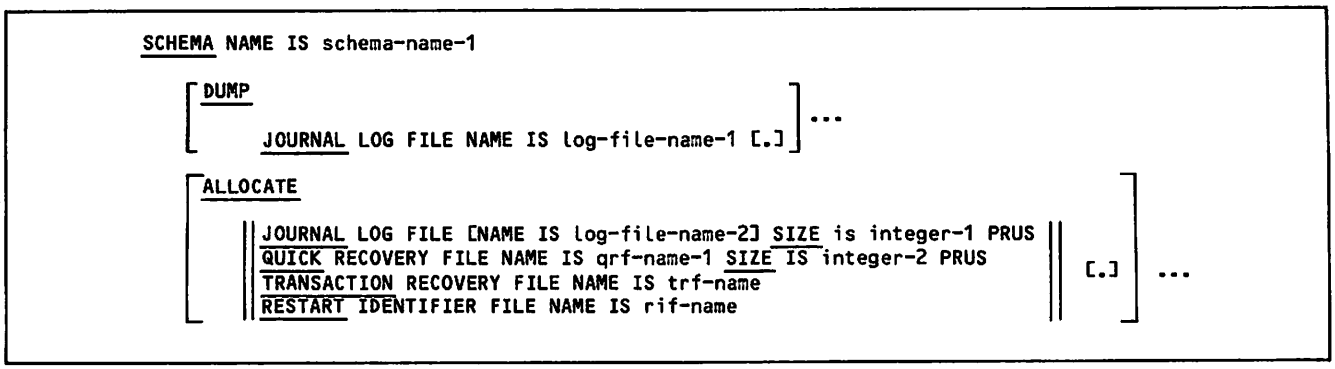


Figure 9-3. Schema Entry Format (DBREC Utility)

File options identify the file types and the permanent file names of the files to be initialized. The rules for specifying file options are as follows:

File options can be specified in any order in an ALLOCATE clause.

One ALLOCATE clause can specify one or more file options.

One or more ALLOCATE clauses can be included in a schema entry.

A particular file option can be specified only once in a schema entry.

JOURNAL LOG FILE Option

The JOURNAL LOG FILE option provides for the initialization of a journal log file. Every journal log file used by CDCS must be initialized by the DBREC utility.

Log-file-name identifies the permanent file name of the journal log file. One or two journal log files can be initialized by use of this clause. Log-file-name can either be omitted or a 7-character log-file-name can be specified.

If log-file-name is omitted, two log file names are assumed. The first six characters of both assumed names are characters specified for the permanent file name of the journal log file in the master directory. The seventh character of one name is the digit 1; the seventh character of the other name is the digit 2. When log-file-name is omitted, one or two files are initialized depending on whether one or two files have been established as permanent files by operating system control statements. Refer to the subsequent subsection Execution of DBREC for Allocation for further information.

If log-file-name is specified, the name must be one of the 7-character names that would be assumed by CDCS and must be the permanent file name of an established permanent file. To initialize the second log file in this situation, another ALLOCATE clause must be specified. Refer to the subsequent subsection Execution of DBREC for Allocation for further information.

Integer-1 specifies the size in physical record units (PRUS) of the journal log file; it must be a positive integer. CDCS utilizes the size of the journal log file to determine when to switch to the alternate file.

QUICK RECOVERY FILE Option

The QUICK RECOVERY FILE option provides for the initialization of the quick recovery file. This file must be initialized before it can be used. Use of the DBREC utility rather than the DBQRFI utility is recommended for initializing the quick recovery file.

The qrf-name specified in this option must specify the permanent file name of the quick recovery file. Qrf-name must be be the same permanent file name (a 1- through 7-character permanent file name) as the name specified for the quick recovery file in the master directory.

Integer-2 specifies the size in physical record units of the quick recovery file; it must be a positive integer. CDCS can write to the quick recovery file until the space provided by specification of the integer is filled. CDCS enforces the size specified by integer-2 by initiating a recovery point, which is described later in this section.

TRANSACTION RECOVERY FILE Option

The TRANSACTION RECOVERY FILE option provides for initialization of the transaction recovery file. This file must be initialized by the DBREC utility before the file can be used.

Trf-name specifies the permanent file name of the transaction recovery file. Trf-name must be specified as follows: the first six characters must be the six characters of the permanent file name specified for the transaction recovery file in the master directory; the seventh character must be the digit 1.

RESTART IDENTIFIER FILE Option

The RESTART IDENTIFIER option provides for the initialization of the restart identifier file. This file must be initialized by the DBREC utility, before the file can be used.

Rif-name specifies the permanent file name of the restart identifier file. Rif-name must be the same permanent file name (a 1- through 7-character permanent file name) as the name specified for the restart identifier file in the master directory.

DBREC Control Statement

Execution of the DBREC utility is called by the DBREC control statement. Figure 9-4 illustrates the format of the DBREC control statement. Optionally, the comma immediately following DBREC can be replaced by a left parenthesis; the terminating period can be replaced by a right parenthesis.

```
DBREC[,I=lfn-1][,L=lfn-2].
```

Figure 9-4. DBREC Control Statement

All parameters of the DBREC control statement are optional and can appear in any order.

The I parameter identifies the local file that contains source input for the DBREC utility. The I parameter is interpreted as follows:

omitted

Local file INPUT is assumed to contain the source input for the utility.

I

Local file COMPILE is assumed to contain the source input for the utility.

I=lfn-1

The specified file name identifies the local file that contains the source input for the utility.

I=0

Not allowed.

The L parameter identifies the local file that receives a listing and diagnostics generated by the DBREC utility. The L parameter is interpreted as follows:

omitted

Local file OUTPUT receives the listing and diagnostics generated by the utility.

L

Local file OUTPUT receives the listing and diagnostics generated by the utility.

L=lfn-2

The specified local file name identifies the file that receives the listing and diagnostics generated by the utility.

L=0

The local file OUTPUT receives only the diagnostics generated by the utility.

Execution of DBREC for Allocation

The job for execution of the DBREC utility for file allocation can only be initiated offline. A data base administrator can initiate the job when CDCS is either active or inactive.

All files that are specified in an ALLOCATE clause must have been established as permanent (direct access for NOS) files before execution of the DBREC utility. The following listing illustrates a series of control statements that, if executed for a particular permanent file name (pfn), establish the permanent file:

For NOS

```
DEFINE
RETURN
```

For NOS/BE

```
REQUEST
REWIND
CATALOG
RETURN
```

Table 9-2 indicates the permanent file name that must be specified in the operating system control statements to establish a pemanent file for each file option of the ALLOCATE clause. Additionally, table 9-2 summarizes all the contexts in which permanent file name is specified for allocation of a file.

Execution of the DBREC utility requires that the master directory file be attached by using the local file name MSTRDIR.

DBREC Execution to Dump the Journal Log File

The DBREC utility can be executed both online and offline for dumping the journal log file. The data administrator creates and initiates the job for offline execution; CDCS creates and initiates the job for online execution.

Much greater flexibility is available during online execution if the CCLPROC sub-clause is specified in the JOB CONTROL clause of DBMSTRD input. The storage file can be specified as either magnetic tape or a permanent file. The online job can be set up to copy the journal log file as an extension of the storage file instead of over writing the storage file on each execution.

The following steps must be contained within the CCL procedure:

1. Specify an ATTACH, REQUEST or LABEL statement to obtain the file to which the journal log file is to be dumped. This file must be given the local file name of LOGDUMP.

2. The file, LOGDUMP, must be positioned at the beginning or at the end of information, as selected by the installation.

3. The CDCS master directory must be attached with the local file name MSTRDIR.

4. The utility progam, DBREC, must be called to dump the journal log file.

These are the basic steps required. A recommended addition would be to place an EXIT statement after the DBREC statement. This can be used to notify the data administrator of a failure of the online dump job.

Offline Execution

The DBREC utility assumes the local file name LOGDUMP for the tape file. This local file must be associated with a tape volume serial number. The VSN control statement should be included in the job stream to associate the local file LOGDUMP with a tape. If no tape volume serial number is supplied in the job stream, the DBREC utility requests that the operator assign one. The tape type and density, if not specified in the master directory, are assumed to be type NT (nine-track) with the density of the installation default density.

Execution of the DBREC utility requires that the master directory be attached in the job stream with the local file name MSTRDIR.

Figure 9-5 is an example of a job for offline execution of the DBREC utility for dumping the journal log file.

Online Execution

When a journal log file is full or a bad file condition exists, CDCS can initiate a job to dump the journal log file. CDCS uses the ROUTE control statement to initiate execution of the job.

The following requirements must be met to provide CDCS with information necessary to initiate a job:

The master directory permanent file attach information must have been supplied as input to CDCS through the CDCS control statement.

The job control information required for initiation of the DBREC job must have been supplied as part of the input for the master directory.

During the online job, the tape to which the journal log file is dumped is specified by the operator. The data administrator must provide the operator with a list of blank-labeled tapes that can be used for journal log file dumps.

TABLE 9-2. SUMMARY OF PERMANENT FILE NAME SPECIFICATIONS FOR ALLOCATION OF FILES BY DBREC UTILITY

Context in Which Permafile Name (pfn) Is Specified	File Type			
	Journal Log File	Quick Recovery File	Transaction Recovery File	Restart Identifier File
Master directory	The 6-character literal that specifies pfn in the JOURNAL LOG FILE clause.	The literal (of 1 to 7 characters in length) that specifies the pfn in the QUICK RECOVERY FILE clause.	The 6-character literal that specifies pfn in the TRANSACTION RECOVERY FILE clause.	The literal (of 1 to 7 characters in length) that specifies pfn in the RESTART IDENTIFIER FILE clause.
Control statements (DEFINE and so forth for NOS or CATALOG and so forth for NOS/BE)	A 7-character pfn must be specified. The first six characters are the same as those specified in the clause indicated above. The seventh character must be the digit 1 or 2. The recommended procedure is to include control statements specifying both allowed pfn's to establish two files.	The same pfn as specified in the clause indicated above.	A 7-character pfn must be specified. The first six characters are the same as those specified in the clause above. The seventh character must be the digit 1.	The same pfn as specified in the clause indicated above.
File options of the ALLOCATE clause of the DBREC utility	Either the pfn is omitted from the JOURNAL LOG FILE option or a 7-character pfn must be specified in the option. The 7-character pfn must be the same pfn as the pfn specified in control statements indicated above. If pfn is omitted, two files with 7-character pfn's as indicated above are assumed.	The same pfn must be specified in the QUICK RECOVERY FILE option as specified in the master directory and in the control statements indicated above.	The same 7-character pfn must be specified in the TRANSACTION RECOVERY FILE option as specified in the control statements indicated above.	The same pfn must be specified in the RESTART IDENTIFIER FILE option as specified in the master directory and in the control statements indicated above.

DBQRFA UTILITY

The DBQRFA utility applies the contents of the quick recovery file to a data base before a reconstruction or restoration operation is initiated. This procedure ensures that the files are in a processable state for subsequent operations. The DBQRFA utility duplicates a recovery capability of automatic recovery.

Execution of the DBQRFA utility is called by the DBQRFA control statement. Figure 9-6 illustrates the format of the DBQRFA control statement. Optionally, the comma immediately following DBQRFA can be replaced by a left parenthesis; the terminating period can be replaced by a right parenthesis.

The parameters can appear in any order in the DBQRFA control statement. The SC parameter is required; the other parameters are optional.

The SC parameter identifies the schema associated with the quick recovery file being processed. The SC parameter is interpreted as follows:

SC=schema-id

Schema-id must be the 4-digit schema identification assigned by the DBMSTRD utility and listed in the master directory contents report.

NOS

```
Job statement
FAMILY control statement
USER control statement
CHARGE control statement
RESOURC(tape-type=integer)
ATTACH,MSTRDIR=pfn/UN=usernum,PW=passwd,M=R,
    R=device-type,PN=packname.
VSN (LOGDUMP=VSN)
DBREC.
End-of-record
    SCHEMA NAME IS SAMPSCH1
        DUMP JOURNAL LOG FILE DB1JLF1.
    SCHEMA NAME IS SAMPSCH2
        DUMP JOURNAL LOG FILE DB2JLF2.
End-of-Information
```

NOS/BE

```
Job statement
ACCOUNT control statement
VSN(LOGDUMP=vsn)
MOUNT(SN=set-name,VSN=vsn)
ATTACH,MSTRDIR,pfn,ID=id,PW=read-passwds,
    MR=1,SN=setname.
DBREC.
End-of-record
    SCHEMA NAME IS SAMPSCH1
        DUMP JOURNAL LOG FILE DB1JLF1.
    SCHEMA NAME IS SAMPSCH2
        DUMP JOURNAL LOG FILE DB2JLF2.
End-of-Information
```

Figure 9-5. Sample Off-line Execution of DBREC Utility to Dump the Journal Log File

```
DBQRFA,SC=schema-id[,MD=lfn-1][,QF=lfn-2].
```

Figure 9-6. DBQRFA Utility Control Statement

The MD parameter identifies the local file that contains the master directory. The MD parameter is interpreted as follows:

omitted

Local file MSTRDIR is assumed to contain the master directory.

MD

Not allowed.

MD=lfn-1

The specified file name identifies the local file that contains the master directory.

The QF parameter identifies the local file that contains the quick recovery file being processed. The QF parameter is interpreted as follows:

omitted

Local file QRFLOG is assumed to contain the quick recovery file.

QF

Not allowed.

QF=lfn-2.

The specified file name identifies the local file that contains the quick recovery file.

In the job stream for executing the utility, appropriate ATTACH control statements must precede the DBQRFA control statement to attach the master directory and the quick recovery file.

DBQRFI UTILITY

The DBQRFI utility initializes the quick recovery file for use in block logging. The quick recovery file must be created before the schema is accessed by CDCS. The DBQRFI utility duplicates a function of the DBREC utility; the use of the DBREC utility is recommended for compatibility with future CDCS enhancements.

Execution of the DBQRFI utility is called by the DBQRFI control statement. Figure 9-7 illustrates the format of the DBQRFI control statement. Optionally, the comma immediately following DBQRFI can be replaced by a left parenthesis; the terminating period can be replaced by a right parenthesis.

```
DBQRFI,lfn,pru-count,schema-id.
```

Figure 9-7. DBQRFI Utility Control Statement

The parameters can appear in any order in the DBQRFI control statement. All paramaters are required. The parameters are interpreted as follows:

lfn

The specified lfn identifies the local file name of the quick recovery file to be initiailzed.

pru-count

Pru-count specifies the size of the quick recovery file in physical record units. Pru-count must be a positive integer. The size must be sufficient to log at least one block for any area in the schema; for efficiency, the size should permit a number of blocks to be logged.

schema-id

Schema-id identifies the schema associated with the quick recovery file. Schema-id must be the 4-digit schema identification assigned by the DBMSTRD utility and listed in the master directory contents.

For use on NOS/BE, the control statements in the job stream for quick recovery file initialization include a REQUEST control statement, a DBQRFI control statement, and a CATALOG control statement. The REQUEST and CATALOG statements make the quick recovery file permanent. Under NOS, a DEFINE control statement and a DBQRFI control statement must be included in the control statements in the job stream.

DBRCN UTILITY

The DBRCN utility reconstructs a data base from file dumps and the journal log file. DBRCN is used when a data base has been destroyed through a disk or software failure. Only the destroyed files must be reinstated. The data base from which DBRCN works is obtained from file dumps. DBRCN updates this data base to the condition immediately preceding the failure or to a selected recovery point, by applying selected after-image log records obtained from the journal log file to the data base. The DBRCN utility duplicates a recovery capability of automatic recovery.

The DBRCN utility can also be used in conjunction with the DBQRFA utility to correct failures caused by the operating system or abnormal CDCS termination. This is done automatically during automatic recovery.

Format of Input for DBRCN Utility

Clauses in the input file have a free-field format. Data appears in columns 1 through 72; columns 73 through 80 are not examined and can be used for sequencing or identification purposes. Figure 9-8 shows the format of input for the DBRCN utility.

SCHEMA NAME Clause

The SCHEMA NAME clause identifies the schema. This clause must precede all other clauses. Schema-name must be a valid schema name, 1 through 30 characters in length, as specified in the master directory and recorded in the journal log file. Only one schema name can be specified in the source input to the utility. A period must terminate this clause.

SELECT FOR RECOVERY Clause

Options specified in the SELECT FOR RECOVERY clause determine the log records selected from the journal log file and applied to the data base. More than one selection criterion can be specified in the same recovery run. Multiple SELECT FOR RECOVERY clauses are allowed. Each log record read from the journal log file is evaluated against all selection criteria until a match occurs. A record qualifies if it meets one criterion.

The AREA NAME option identifies the area to be recovered. Area-name must be a valid area name, 1 through 30 characters in length, as specified in the master directory and recorded in the journal log file. The AREA NAME option and the VERSION NAME option together make up a single selection criterion against which records are selected. Refer to the VERSION NAME option for interpretations of these options.

The VERSION NAME option identifies a version to be recovered. Version name must be a valid version name, 1 through 7 characters in length, that is specified in the master directory attached during execution of the utility. The VERSION NAME option together with the AREA NAME option is interpreted as follows:

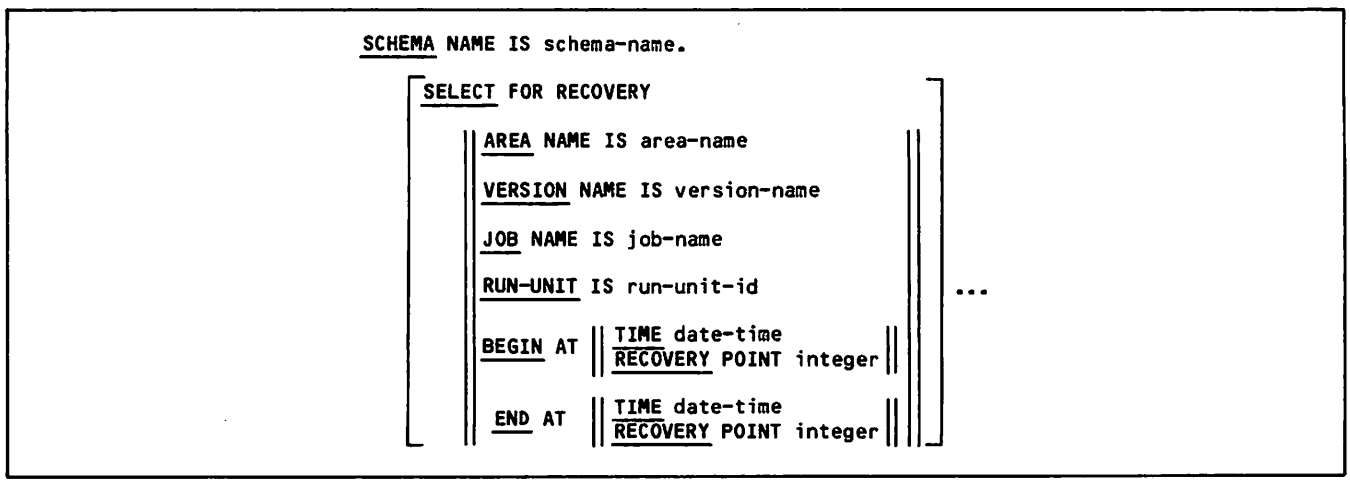


Figure 9-8. Format of Input for DBRCN and DBRST Utilities

If the AREA NAME option is specified without the VERSION NAME option being specified, all journal log records for all versions of the indicated area are selected.

If the VERSION NAME option is specified without the AREA NAME option being specified, all journal log records with the indicated version name are selected.

If the VERSION NAME option is specified with the AREA NAME option also specified, only those log records indicating both the specified version and area are selected and applied to the file. This combination has potential problems. If the area specified corresponds to a unique permanent file that is not shared by any other version, all log records for that area are selected. On the other hand, if the area corresponds to a permanent file for version MASTER that is shared by other versions, only log records of updates made by application programs using the specified version are selected; log records of updates made by application programs using any other version are not selected.

The JOB NAME option identifies a particular job as a selection criterion. Job-name must be the 4- or 7-character job name assigned by the operating system.

The RUN-UNIT option identifies a particular application program as a selection criterion. Run-unit must be the 1- to 10-character program identification (also called the program name) supplied by the application program.

The BEGIN and END options specify a range of entries on the journal log file to be considered for processing. The DBRCN utility selects entries that are within the range options; if no range is specified, the entire journal log file is considered. Either or both of the range options (time and a recovery point) can be specified. Within a given selection criterion, only one BEGIN and END range specification is permitted.

The TIME option identifies the date and time as a selection criterion. Date-time must be specified in the following form:

yydddhhmmss

The RECOVERY POINT option identifies a particular recovery point as a selection criterion. Integer must be a recovery point number (of 1 through 10 digits in length) recorded in the recovery point log record.

If both the TIME and RECOVERY POINT options are specified, the RECOVERY POINT specification must appear first in source input for the DBRCN utility. (The TIME option qualifies the RECOVERY POINT option.)

DBRCN Control Statement

Execution of the DBRCN utility is called by the DBRCN control statement. Figure 9-9 illustrates the format of the DBRCN control statement. Optionally, the comma immediately following DBRCN can be replaced by a left parenthesis; the terminating period can be replaced by a right parenthesis.

All parameters of the DBRCN control statement are optional and can appear in any order.

The I parameter identifies the local file that contains source input for the utility. The I parameter is interpreted as follows:

omitted

Local file INPUT is assumed to contain the source input for the utility.

I

Not allowed.

I=lfn-1

The specified file name identifies the local file that contains the source input for the utility.

I=0

Not allowed.

The L parameter identifies the local file that receives the listing and diagnostics generated by the utility. The L parameter is interpreted as follows:

omitted

Local file OUTPUT receives the listing and diagnostics generated by the utility.

L

Not allowed

L=lfn-2

The specified local file name identifies the file that receives the listing and diagnostics generated by the utility.

L=0

The local file OUTPUT receives only the diagnostics generated by the utility; no listing is produced.

DBRCN[,I=lfn-1][,L=lfn-2][,MD=lfn-3][,OL=lfn-4][,NL=lfn-5[/lfn-6]][,SM={TAPE / DISK}][,SF=lfn-7][,RCV].

Figure 9-9. DBRCN Utility Control Statement

The MD parameter identifies the local file that contains the master directory. The MD parameter is interpreted as follows:

omitted

Local file MSTRDIR is assumed to contain the master directory.

MD

Not allowed.

MD=lfn-3

The specified file name identifies the local file that contains the master directory.

The OL parameter identifies the local file that contains the journal log file to be used as input for the utility. The OL parameter is interpreted as follows:

omitted

Local file OLDLOG is assumed to identify the journal log file.

OL

Not allowed.

OL=lfn-4

The specified local file name identifies the journal log file.

The NL parameter identifies the local file names of the journal log files to which log records are written during recovery processing by the utility. The NL parameter is interpreted as follows:

omitted

No new journal log file is produced.

NL

Not allowed.

NL=lfn5

The specified local file name identifies the file to which journal log records are written.

NL=lfn5/lfn6

Specifies dual logging. Both specified local file names identify files to which journal log records are written.

The SM parameter identifies the medium to be used for the intermediate files produced by Sort/Merge. The SM parameter is interpreted as follows:

omitted

Same as SM=DISK.

SM

Same as SM=TAPE.

SM=DISK

A disk is used.

SM=TAPE

Tapes are used. Up to three tapes can be requested by Sort/Merge.

The SF parameter identifies the local file name of the sorted select file of records produced by the utility during recovery processing. If the local file is to be saved, it must be made a permanent file before the utility is executed. The SF parameter is interpreted as follows:

omitted

Output from the sort is not saved.

SF=lfn-7

The specified local file identifies the file that contains selected sorted records that were produced by the utility.

The RCV parameter identifies whether the current execution of the utility is an initial recovery run or a rerun. The parameter is interpreted as follows:

omitted

An initial recovery run is assumed. The journal log file, identified by the OL parameter, is assumed to have been produced by CDCS logging.

RCV

The current execution of the utility is a rerun of the utility. The journal log file, identified by the OL parameter, is assumed to have been produced by a previous execution of the utility when the file was specified in the SF parameter.

Execution of the DBRCN Utility

Usually, before the DBRCN utility is run, the contents of the quick recovery file for the data base being recovered are applied to the data base by calling the DBQRFA utility. The DBRCN utility checks for an empty quick recovery file if one is specified in the master directory. If for some reason the data administrator elects not to apply the contents of the quick recovery file to the data base, the quick recovery file must be reinitialized by the DBREC or DBQRFI utility before reconstruction is begun. The DBRCN utility logs to the quick recovery file while it is modifying the data base (if a quick recovery file is specified in the master directory).

The master directory and journal log files must be attached before calling the DBRCN utility. The master directory attached must be the one in use at the time the journal log files were produced. If a quick recovery file is specified in the master directory for the schema, the quick recovery file must be attached in the job stream by using the local file name QRFLOG. Data base files to be processed are automatically attached by the

utility. If the SF parameter is included in the DBRCN control statement, a new selected sorted journal log file is produced by execution of the utility. If the file is to be saved, it must be made permanent before the execution of the utility.

In addition to the DBRCN control statement, an input file is needed to specify selection criteria.

DBRST UTILITY

The DBRST utility restores a data base to a previous state using the journal log file. DBRST is used when a data base has been erroneously updated and must be reset to a previous state. Only those areas affected must be reset. DBRST selectively applies the before-image log records to the data base by processing the journal log file chronologically in reverse order (from a later to an earlier time). Before running DBRST, the contents of the quick recovery file must be applied to the data base. This operation can be performed either by a feature of automatic recovery or by execution of the DBQRFA utility.

The data administrator should be aware of the possible problems associated with restoring a data base. Erroneous data could still be in the data base after running DBRST because of the cascade effect (a problem of erroneous data being used; refer to the Cascade Effect subsection for more information). Updates to the bad data could have occurred at a later date and then could be lost after running DBRST. To solve problems like these, special recovery strategies, scheduling of jobs, and application-dependent reprocessing must be considered.

Format of Input for DBRST Utility

The clauses and format of source input for the DBRST utility are the same as for the DBRCN utility. Refer to figure 9-8 for the format of input for the DBRST utility and to the subsection Format of Input for DBRCN Utility for the descriptions of the clauses. The descriptions are applicable for the clauses of the DBRST utility with the exceptions of the BEGIN and END options. The BEGIN and END options of the DBRST utility are interpreted as follows:

BEGIN TIME option

All journal log record entries with a time-stamp less than or equal to the date-time supplied are considered by the DBRST utility.

END TIME option

All journal log record entries with a time-stamp greater than or equal to the date-time supplied are considered by the DBRST utility.

BEGIN RECOVERY POINT option

All journal log record entries prior to the specified recovery point are considered by the DBRST utility.

END RECOVERY POINT option

All journal log record entries after the specified recovery point are considered by the DBRST utility.

DBRST Control Statement

Execution of the DBRST utility is called by the DBRST control statement. Figure 9-10 illustrates the format of the DBRST control statement. Optionally, the comma immediately following DBRST can be replaced by a left parenthesis; the terminating period can be replaced by a right parenthesis.

All parameters of the DBRST control statement are optional and can appear in any order. The parameter descriptions are identical to the parameter descriptions for the DBRCN control statement. Refer to the subsection DBRCN Control Statement for the parameter descriptions of the DBRST control statement.

Execution of the DBRST Utility

Usually, before the DBRST utility is run, the contents of the quick recovery file are applied to the data base. (This operation is usually performed by the data administrator, who runs the DBQRFA utility.) The DBRST utility checks for an empty quick recovery file if one is specified in the master directory. If for some reason the data administrator elects not to apply the contents of the quick recovery file to the data base, the quick recovery file must be reinitialized by the DBREC or DBQRFI utilities before restoration is begun. The DBRST utility logs to the quick recovery file while it is modifying the data base (if a quick recovery file is specified in the master directory).

The master directory and journal log files must be attached before calling the DBRST utility. The master directory attached must be the one in use at the time the journal log file was produced. If a quick recovery file is specified in the master directory for the schema, the quick recovery file must be attached in the job stream by using the local file name QRFLOG. Data base files to be processed are automatically attached by the utility. If the SF parameter is included in the DBRST

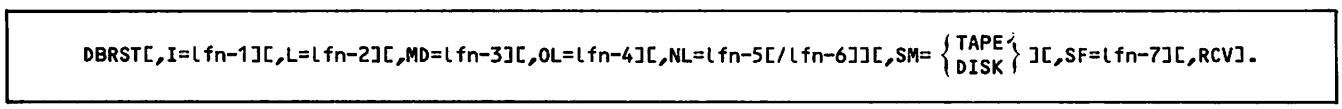


Figure 9-10. DBRST Utility Control Statement

control statement, a new selected sorted journal log file is produced by execution of the utility. To save this file, the file must be made a permanent file before execution of the utility.

In addition to the DBRST control statement, an input file is needed to specify selection criteria for the DBRST utility.

Recovery Example Using DBRST Utility

A sample restoration illustrates data base recovery using the DBRST utility. The run-unit IOD5 made erroneous updates to the area IOAREAD. Using a COBOL program to read the journal log file and extract information, the data administrator ascertained that no other run-units had updated the area while run-unit IOD5 was active. The DBRST utility was run with the schema name and the area name as input parameters.

Figure 9-11 shows the output listing produced by DBRST. The input to the utility is listed first in the DBRST output listing. Then information about the log records selected for use by DBRST is listed. The first column in this list shows the Julian date on which the log file record was written, the second column shows the time the log record was written, the third column shows the name of the job that generated the log record, and the fourth column shows the run-unit identification. The fifth column contains the schema identification and the area identification separated by a slash; the sixth column indicates whether the log record is a before- or after-image record and gives the operation on the file that generated the log record. In this example, the seventh column gives the key of the record. Processing statistics follow the list of log records. Log records and processing statistics appear for each area being processed. Processing statistics for the entire DBRST run appear at the end of the listing and on the dayfile.

The user should note that when DBRST selects a BEFORE WRITE log record for processing, DBRST must perform a delete operation to remove the effects of the selected write operation. The summary lines for DELETES PROCESSED therefore is a count of BEFORE WRITE log records selected.

LOGGING

CDCS performs logging during application-service phase processing to up to four different files for a schema. These files provide information needed for data base recovery. The log and recovery files are selected by the data administrator. The following files can be specified for logging:

Journal log file

Quick recovery file

Restart identifier file

Transaction recovery file

The journal log file can be a source of the collection of statistical information on the use of each data base file. User activity reports can be generated from entries in the journal log file since records in that file provide a historical record of users' interactions with data base areas. Logging of users' activities provides a comprehensive history of data base processing, including the frequency of data base access by a specific user and the kinds of operations performed on a data base.

The data administrator specifies the log and recovery files used for a data base by including in the source input for the master directory the clauses that specify particular files. Each file is defined for use on a schema basis; however, use of the journal log file and the quick recovery file must also be defined for each area and version for which logging to that file applies.

JOURNAL LOG FILE

The journal log file is assigned on a per-schema basis when specified in a clause in the source input for the master directory. If logging to a journal log file is specified in the schema and the journal log file is not available, no access to any area of the schema is allowed. The following rules apply to the journal log file:

The journal log file must be a permanent file and must be initialized by the DBREC utility.

Two journal log files are associated with a schema when journal logging is selected; however, logging can be performed on only one journal log file at a time. The other journal log file is an alternate to which an automatic online switch is initiated by CDCS if the journal log file in use becomes full or has an error condition.

A journal log file cannot be shared among different schemas.

When a journal log file is used, it is automatically attached at CDCS initialization time, returned, and automatically reattached for requests to use the schema. It is automatically returned when the schema is not being used and when CDCS system control point execution is terminated.

The structure of a journal log file is discussed in a subsequent subsection.

Journal Log File Logging Options

Although the journal log file is specified on a schema basis in the master directory, logging to that file must also be selected for each area of each version for which logging is to be performed. To select logging for an area, one or both of the following logging options must be specified for the area in the master directory:

BEFORE IMAGE RECORDS

AFTER IMAGE RECORDS

The options of recording before- and after-image records are intended for restoration and reconstruction. A before-image record is a copy of a record in a data base before the record has been updated (that is, stored, modified, or deleted); conversely, an after-image record is a copy of a record after it has been changed.

DBRST Output

```
        MD=MSTRDIR          SCHEMA=DUM1MDB                      DBRST V2.3(82020)


                  1             SCHEMA IS DUM1MDB.
                  2               SELECT FOR RECOVERY
                  3                 AREA NAME IS IOAREAD
                  4                   BEGIN AT TIME    82026163338
                  5                   END   AT TIME    82026163335
                  6                 .
        END OF SOURCE INPUT.


                                        LOG RECORDS SELECTED FOR UPDATE
 82026 16.33.36. C000001 IOD5       0001/0005          BEFORE WRITE    KEY=  :::::::A
 82026 16.33.36. C000001 IOD5       0001/0005          BEFORE WRITE    KEY=  :::::::B
 82026 16.33.36. C000001 IOD5       0001/0005          BEFORE WRITE    KEY=  :::::::C
 82026 16.33.36. C000001 IOD5       0001/0005          BEFORE WRITE    KEY=  :::::::D
 82026 16.33.36. C000001 IOD5       0001/0005          BEFORE WRITE    KEY=  :::::::E
 82026 16.33.36. C000001 IOD5       0001/0005          BEFORE WRITE    KEY=  :::::::F
 82026 16.33.36. C000001 IOD5       0001/0005          BEFORE WRITE    KEY=  :::::::G
 82026 16.33.36. C000001 IOD5       0001/0005          BEFORE WRITE    KEY=  :::::::H
 82026 16.33.36. C000001 IOD5       0001/0005          BEFORE WRITE    KEY=  :::::::I
 82026 16.33.36. C000001 IOD5       0001/0005          BEFORE WRITE    KEY=  :::::::J
     10  RECORDS SELECTED
  AREA NAME = IOAREAD
  SCHEMA ID = 0001   AREA ID = 0005
       WRITES PROCESSED =         0
       DELETES PROCESSED =       10
       REWRITES PROCESSED =       0
TOTAL WRITES PROCESSED =         0
TOTAL DELETES PROCESSED =       10
TOTAL REWRITES PROCESSED =       0
```

Dayfile

```
17.01.18.DBRST.
17.01.19.USER, ....
17.01.19.CHARGE, ...

17.01.19.ATTACH,QRFLOG=DUMQRF/UN=CDCS23,M=W.
17.01.19.ATTACH,MSTRDIR=CDCSMD.
17.01.20.VSN(OLDLOG=003487)
17.01.20.LABEL(OLDLOG,D=PE,PO=R)
17.06.58.NT54, ASSIGNED TO OLDLOG , VSN=003487.
17.06.58.DBRST.
17.07.00.      10  RECORDS SELECTED
17.07.00.***KEY EXTRACTION USED
17.07.00. ** INSERTIONS DURING INPUT   *********0
17.07.00. ** DELETIONS DURING INPUT    *********0
17.07.00. ** TOTAL RECORDS SORTED      ********10
17.07.00. ** INSERTIONS DURING OUTPUT  *********0
17.07.00. ** DELETIONS DURING OUTPUT   *********0
17.07.00. ** TOTAL RECORDS OUTPUT      ********10
17.07.00. ** MERGE ORDER USED          ********11
17.07.00. **END SORT RUN
17.07.03. TOTAL WRITES PROCESSED =         0
17.07.03. TOTAL DELETES PROCESSED =       10
17.07.03. TOTAL REWRITES PROCESSED =       0
17.07.03.     74600 MAXIMUM SCM WORDS USED
```

Figure 9-11. DBRST Utility Example

The selection of the options determines the kind of recovery possible for the area. The AFTER IMAGE RECORDS option must be selected for automatic recovery of the area and for recovery of the area by the DBRCN utility. The BEFORE IMAGE RECORDS option must be selected for recovery of the area by the DBRST utility.

Maintenance of the Journal Log File

The automatic journal log maintenance feature of CDCS provides for maintaining the disk file to which CDCS logs journal log records. If all requirements for this feature are met, CDCS initiates an online switch when the journal log file in use becomes full or has an error condition. After the switch, CDCS initiates a job that calls the DBREC utility to dump the journal log file to the storage file and reinitialize the journal log file for subsequent use. Refer to the Automatic Journal Log Maintenance subsection which appears earlier in this section. The data administrator should establish the appropriate administrative procedures for the maintenance of the tape files.

If an error condition occurs on the journal log file, the data administrator must determine the cause of the error and take corrective action.

If a master directory modification run involves adding or deleting information for schemas or areas, the schema and area identifications can be changed by the DBMSTRD utility. When schema and area identifications change, all log and recovery files containing the old schema and area identifications become invalid. The data base should be dumped so that a current backup copy can be obtained and all log files should be reinitialized. If a data base is restructured, schema and area identifications are changed also and the same procedure should be followed.

Journal Log File Structure

The journal log file is a CRM Basic Access Methods sequential (SQ) file with record type T (trailer) and block type C (character count). The journal log file to which CDCS logs during execution-time processing is partitioned; an end-of-record is written before every recovery point. The copy of the journal log file transferred to tape by the DBREC utility and the journal log file produced by DBRCN and DBRST logging are not partitioned.

The journal log file can contain the following types of log records:

Begin transaction
Commit transaction
Drop transaction
Invoke
Version change
Open
Privacy breach
Before-image
After-image
Close
Terminate
Recovery point

Journal Log Record Structure

The header (first seven words) of each log record contains information common to all log record types; the trailer, which is the remainder of the record, consists of information applicable to a specific log record type. The number of words contained in the log record varies according to the type of log record being generated with the exception of before-image and after-image records, which conform to the schema description of the record. All fields in the log record are in display code with the exception of before- and after-image records. The contents of all unused fields are undefined.

Log Record Header

The log record header, shown in figure 9-12, holds information such as the record type code (one of the twelve possible log record types); the directive code (one of the operations that caused the log record to be generated); the length of the trailer; the version name; and the schema, area, and user identifications.

An abbreviated description of the header is included in the description of each log record type.

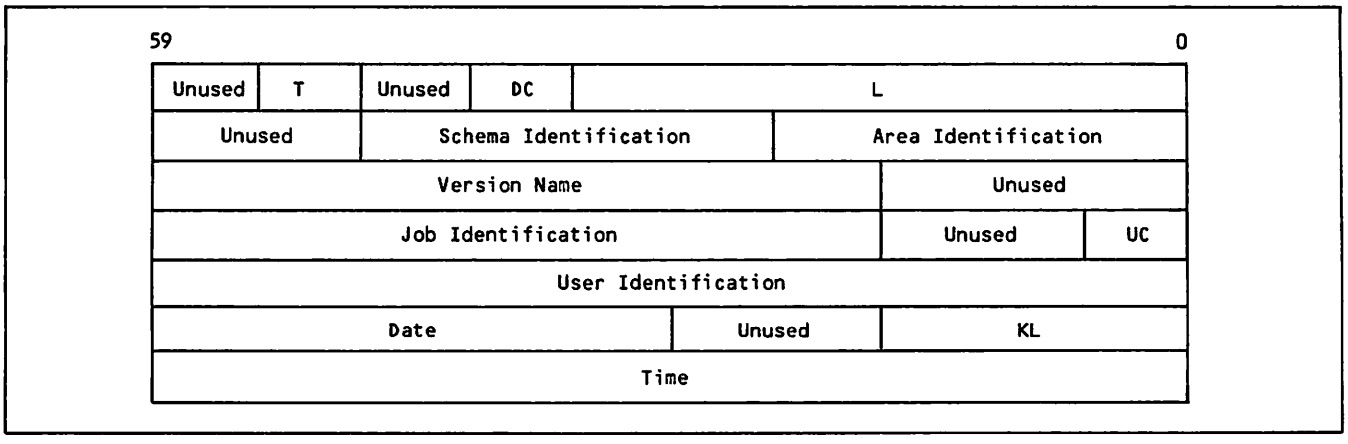


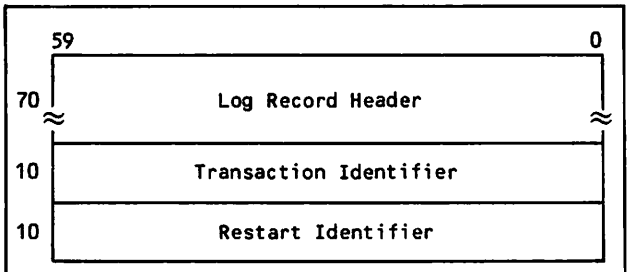
Figure 9-12. Log Record Header (Sheet 1 of 2)

Length in Characters	Description
1	Unused.
1	T: Log record type code; the applicable codes and their interpretations are as follows: A Begin transaction log record B Commit transaction log record C Drop transaction log record F Version change log record 1 Open log record 2 Privacy breach log record 3 Before image log record 4 After image log record 5 Close log record 6 Invoke log record 7 Terminate log record 8 Recovery point log record
1	Unused.
1	DC: Directive code; the applicable codes and their interpretations are as follows: A Open for retrieval B Open for update D Write E Rewrite F Delete (Codes applicable for log record types 1, 3, and 4.) Z Filler code (Code applicable for all other log record types.)
6	L: The number of characters in the trailer portion of the log record. The number is right justified; leading unused positions are zero filled.
2	Unused.
4	Schema Identification: The 4-digit schema identification number assigned by the DBMSTRD utility. The number is right justified; leading unused positions are zero filled.
4	Area Identification: The 4-digit area identification number assigned by the DBMSTRD utility. The number is right justified; leading unused positions are zero filled. The field is set to display code zeros if the record does not concern a particular area.
7	Version Name: The 1- to 7-character version name. The name is left justified in the field; unused positions are blank filled.
3	Unused.
7	Job identification: The job name assigned by NOS/BE, or the 4-character job sequence number assigned by NOS. The identifier is left justified in the word; unused positions are blank filled.
2	Unused.
1	UC: For log record types 3 and 4, the number of unused characters at the end of the record; for all other log record types, display code zero.
10	User identification: The program identification supplied by the application program.
5	Date: The date of the log entry in Julian format yyddd where the first two digits indicate the year, and the last three digits indicate the number of the day in the year.
2	Unused.
3	KL: For log record types 3 and 4, the number indicating the length of the record key; for all other log record types, display code zeros. The number is right justified; leading unused positions are zero filled.
10	Time: The time of the log entry in hours, minutes, and seconds in the format Δhh.mm.ss. where the first character is a blank.

Figure 9-12. Log Record Header (Sheet 2 of 2)

Transaction Log Record

The transaction log record is written when an application program begins, commits, or drops a transaction. The record format, shown in figure 9-13, is the same for each of these operations except for the log record type code, which indicates the operation performed.

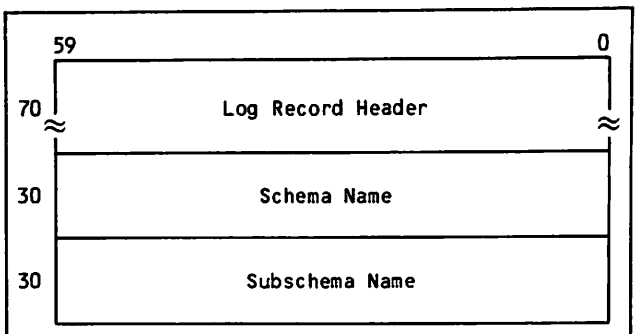


Length in Characters	Description
70	Log record header: Contains fields as described in figure 9-12. Particular fields contain values as follows: T A, B, or C as follows: A for a begin transaction log record; B for a commit transaction log record; C for a drop transaction log record. DC Z L 000020 UC 0 KL 000
10	Contains the 1- to 10-character transaction identifier assigned by the user. The identifier is left justified in the field; unused positions are blank filled.
10	Contains the 1- to 10-character restart identifier assigned by CDCS. The identifier is left justified in the field; unused positions are blank filled.

Figure 9-13. Transaction Log Record

Invoke or Version Change Log Record

The invoke or version change log record is written when an application program begins interaction with CDCS and when an application program specifies the use of a new data base version. The record format, shown in figure 9-14, is the same for both operations except for the log record type code, which indicates the operation performed.



Length in Characters	Description
70	Log record header: Contains fields as described in figure 9-12. Particular fields contain values as follows: T F or 6 as follows: F for a version change log record; 6 for an invoke log record. DC Z L 000060 UC 0 KL 000
30	Schema name: Contains the 1- through 30-character schema name. The name is left justified in the field; unused positions are blank filled.
30	Subschema name: Contains the 1- through 30-character subschema name. The name is left justified in the field; unused positions are blank filled.

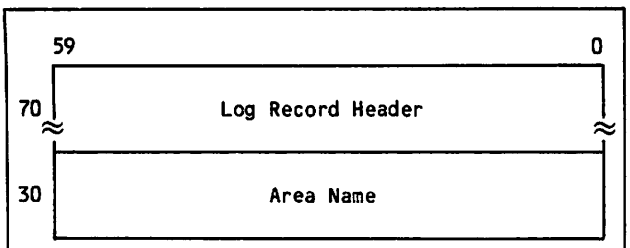
Figure 9-14. Invoke and Version Change Log Record

Open Log Record

The open log record, shown in figure 9-15, is written when an application program performs an open operation on the area.

Privacy Breach Log Record

The privacy breach log record is written to the journal log file when a user attempts to access an area but has failed to provide the appropriate access control key (privacy key). The log record includes the information shown in figure 9-16.



Length in Characters	Description
70	Log record header: Contains fields as described in figure 9-12. Particular fields contain values as follows: T 1 DC A or B as follows: A for open for retrieval; B for open for update. L 000030 UC 0 KL 000
30	Area name: Contains the 1- through 30-character area name. The name is left justified in the field; unused positions are blank filled.

Figure 9-15. Open Log Record

Before-Image and After-Image Log Records

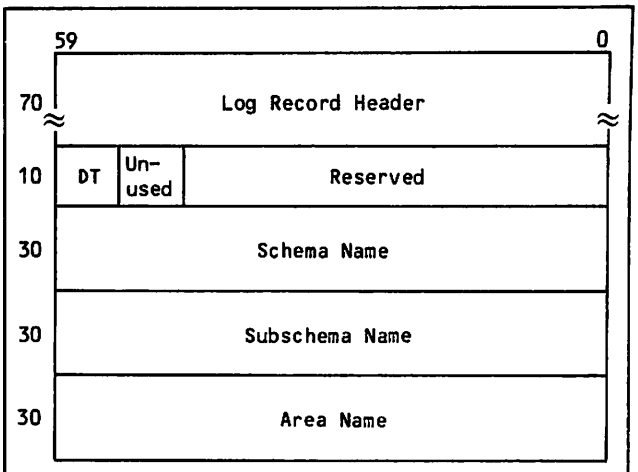
The before-image and after-image log records are written in response to an operation that updates the data base. The record format, shown in figure 9-17, is the same for both types of records except for the log record type code, which indicates the kind of record that was written. This is the only log record that can contain fields that are not display code; the record image portion of the record conforms to the schema description of the record.

Close Log Record

The close log record, shown in figure 9-18, is written when an application program performs a close operation on the area.

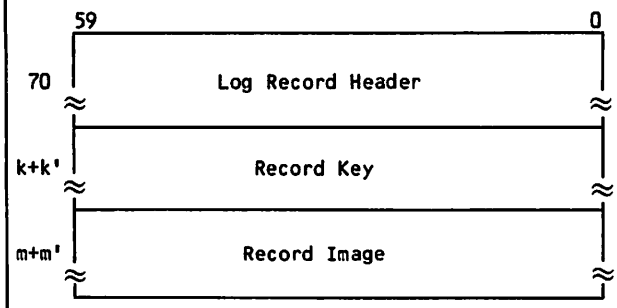
Terminate Log Record

The terminate log record is written when an application program terminates interaction with CDCS. The format of the terminate log record is shown in figure 9-19.



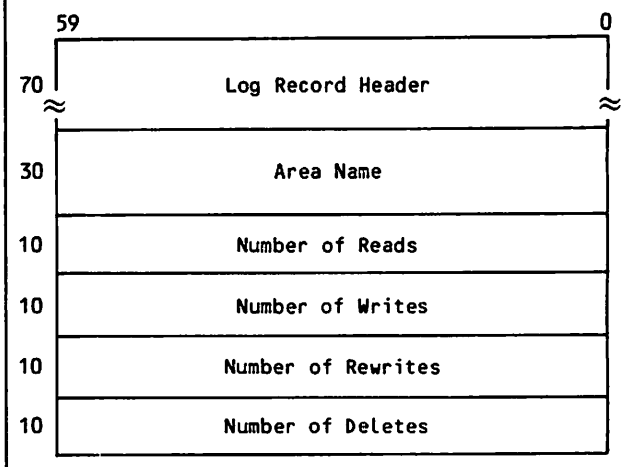
Length in Characters	Description
70	Log record header: Contains fields as described in figure 9-12. Particular fields contain values as follows: T 2 DC Z L 000100 UC 0 KL 000
1	DT; contains a code indicating the breach data type on which the breach attempt occured. The only applicable code is A, which indicates the area data type.
1	Unused.
8	Reserved.
30	Schema name: Contains the 1- through 30-character schema name. The name is left justified in the field; unused positions are blank filled.
30	Subschema name: Contains the 1- through 30-character subschema name. The name is left justified in the field; unused positions are blank filled.
30	Area name: Contains the 1- through 30-character area name. The name is left justified in the field; unused positions are blank filled.

Figure 9-16. Privacy Breach Log Record



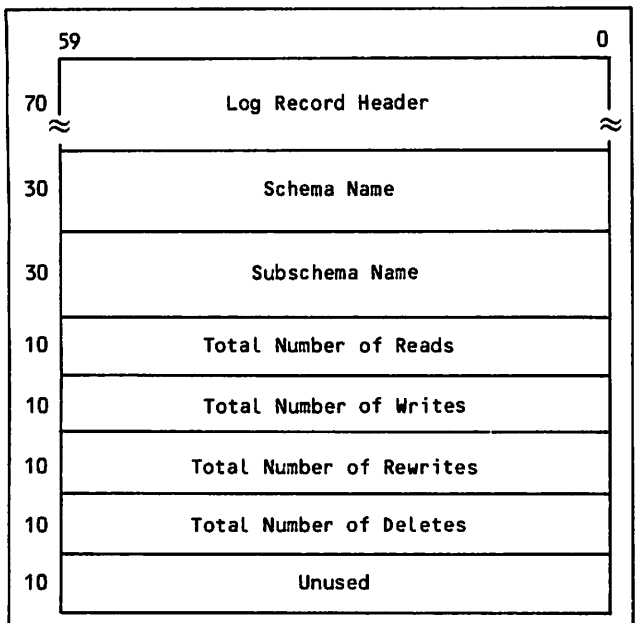
Length in Characters	Description
70	Log record header: Contains fields as described in figure 9-12. Particular fields contain values as follows: T 3 or 4 as follows: 3 for a before image log record; 4 for an after image log record. L n where n is the sum of k+k'+m+m'; these quantities are identified in the description of the record key and record image fields. UC m' where m' is number of characters added to the end of the data base record to make the length (in characters) of the log record a multiple of 10. KL k where k is the number of characters defined (in the schema definition) for the record key.
k+k'	Record key: Contains the primary key of the record and characters added to the end of the key to make number of characters in the field equal to a multiple of 10; k represents the length in characters of the key; k' represents the number of characters added. If k is a multiple of 10, k' equals 0.
m+m'	Record image: Contains the image of the data base record (as defined in the schema) and a number of characters added to the end of the record to make the number of characters in the field equal to a multiple of 10; m represents the length in characters of the record; m' represents the number of characters added. If m is a multiple of 10, m' equals 0.

Figure 9-17. Before Image and After Image Log Records



Length in Characters	Description
70	Log record header: Contains fields as described in figure 9-12. Particular fields contain values as follows: T 5 DC Z L 000070 UC 0 KL 000
30	Area name: Contains the 1- through 30-character area name. The name is left justified in the field; unused positions are blank filled.
10	Number of reads by the user: The number is right justified in the field; leading unused positions are zero filled.
10	Number of writes by the user: The number is right justified in the field; leading unused positions are zero filled.
10	Number of rewrites by the user: The number is right justified in the field; leading unused positions are zero filled.
10	Number of deletes by the user: The number is right justified in the field; leading unused positions are zero filled.

Figure 9-18. Close Log Record

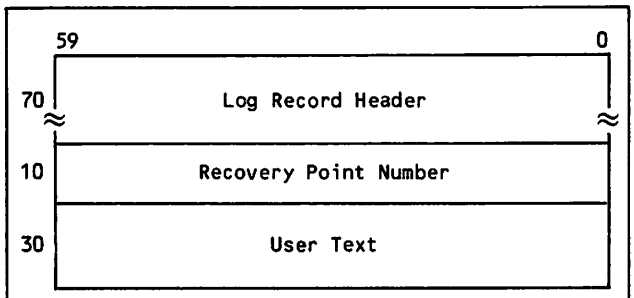


Length in Characters	Description
70	Log record header: Contains fields as described in figure 9-12. Particular fields contain values as follows: T 7 DC Z L 000110 UC 0 KL 000
30	Schema name: Contains the 1- through 30-character schema name. The name is left justified in the field; unused positions are blank filled.
30	Subschema name: Contains the 1- through 30-character subschema name. The name is left justified in the field; unused positions are blank filled.
10	Total number of reads by the user: The number is right justified in the field; leading unused positions are zero filled.
10	Total number of writes by the user: The number is right justified in the field; leading unused positions are zero filled.
10	Total number of rewrites by the user: The number is right justified in the field; leading unused positions are zero filled.
10	Total number of deletes by the user: The number is right justified in the field; leading unused positions are zero filled.
10	Unused: Contains zeros.

Figure 9-19. Terminate Log Record

Recovery Point Log Record

The recovery point log record, shown in figure 9-20, is written whenever a recovery point is requested by an application program or whenever CDCS generates an internal recovery point.



Length in Characters	Description
70	Log record header: Contains fields as described in figure 9-12. Particular fields contain values as follows: T 8 DC Z L 000040 UC 0 KL 000
10	Recovery point number: The number, which is assigned by CDCS, is right justified in the field; leading unused positions are zero filled.
30	User text: Contains the 1- through 30-character user-supplied text from the call to DB$RPT routine. The text is left justified in the field; unused positions are blank filled.

Figure 9-20. Recovery Point Log Record

QUICK RECOVERY FILE

The quick recovery file is assigned on a per-schema basis when specified in a clause in the source input for the master directory. If logging to a quick recovery file is specified in the schema and the quick recovery file is not available, no access to any area of the schema is allowed. The following rules apply to the quick recovery file.

File must be a permanent file.

File must be initialized before use. The DBREC utility can be used to initialize the file.

File cannot be shared among schemas.

When a quick recovery file is used, it is automatically attached at CDCS initialization time, returned, and automatically reattached for requests to use the schema. It is automatically returned when the schema is not being used and when CDCS system control point execution is terminated.

Although the quick recovery file is specified on a schema basis in the master directory, logging to that file must also be selected for each area of each version to which it applies. To select logging for an area, the logging option BEFORE IMAGE BLOCKS must be specified for the area in the master directory.

The log processing module writes before-image blocks to the quick recovery file before an area is updated if logging of before-image blocks has been specified for the area.

The size limit of the quick recovery file is adhered to by CDCS. When the quick recovery file is full, CDCS initiates a recovery point which causes the updated data base blocks for all areas in the schema to be force written, the quick recovery file for the schema to be emptied of before-image blocks, and a recovery point log record to be written on the journal log file if a journal log file is specified for the schema.

The quick recovery file is composed of information and blocks for internal use by CDCS. The data administrator need not be concerned with the actual structure of the file, but, as the first step in manual recovery from a system failure, should ensure that the contents of the quick recovery file are applied to the data base by calling the DBQRFA utility. If DBQRFA is not used, CDCS automatically applies the contents of the quick recovery file to the data base the next time CDCS is initialized.

If schema and area identifications change because a master directory is modified, the quick recovery file must be reinitialized.

RESTART IDENTIFIER FILE

The restart identifier file is assigned on a per-schema basis when specified in a clause in the source input for the master directory. If logging to a restart identifier file is specified for the schema and the restart identifier file is not available, no access to any area of the schema is allowed. The following rules apply to the restart identifier file.

File must be a random permanent file.

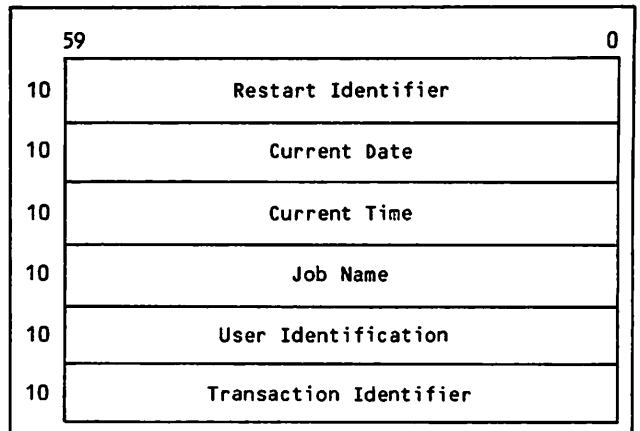
File must be initialized by the DBREC utility before use.

Only one file can be assigned to a schema.

File cannot be shared among schemas.

When a restart identifier file is used, it is automatically attached at CDCS initialization time, returned, and automatically reattached for requests to use the schema. It is automatically returned when the schema is not being used and when CDCS system control point execution is terminated.

The restart identifier file is an indexed sequential (IS) file whose primary key is the 10-character restart identifier. Each record in the restart identifier is 6 words (60 characters) long and has the format shown in figure 9-21.



Length in Characters	Description
10	Restart identifier: Contains the 1- to 10-character restart identifier, which is left justified in the field; unused positions are blank filled. This field is the primary record key of the restart identifier file.
10	Current date: Contains the current date. The format of the current date is dependent on the format specified for the operating system at installation time.
10	Current time: Contains the current time. The format of the current time is dependent on the format specified for the operating system at installation time.
10	Job name: Contains the first 7 characters of the job name assigned by NOS/BE, or the 4-character job sequence number assigned by NOS. The job name is left justified in the field; unused positions are blank filled.
10	User identification: Contains the 10-character program identification supplied by the application program. The user identification is left justified in the field; unused positions are blank filled.
10	Transaction identifier: Contains the 1- to 10-character transaction identifier, which is left justified in the field; unused positions are blank filled.

Figure 9-21. Restart Identifier File Record Format

A new record is written to the restart identifier file when an application program requests a restart identifier. The record is updated each time the application program commits a transaction. The record is retained in the restart identifier file until the application program associated with the restart identifier terminates normally.

A data administrator might want to read the restart identifier file to determine the number of application programs that terminate abnormally and have not been reassigned the restart identifier in a restart operation.

If the restart identifier file becomes very large, the data administrator should purge and reinitialize the file; however, purging the file could impact the restart of some application programs.

If the transaction recovery file is reinitialized, the restart identifier file should also be reinitialized. If not, CDCS could get a CRM error when CDCS accesses the restart identifier file.

TRANSACTION RECOVERY FILE

The transaction recovery file is assigned on a per-schema basis when specified in a clause in the source input for the master directory. If logging to a transaction recovery file is specified in the schema and the transaction recovery file is not available, no access to any area of the schema is allowed. The following rules apply to the transaction recovery file.

File must be a random permanent file.

File must be initialized by the DBREC utility before use.

Only one file can be designated per schema.

File cannot be shared among schemas.

When a transaction recovery file is used, it is automatically attached at CDCS initialization time, returned, and automatically reattached for requests to use the schema. It is automatically returned when the schema is not being used and when CDCS system control point execution is terminated.

The transaction recovery file contains the before-image records of the records updated during execution of a particular transaction. If the transaction is not completed, the records in this file are automatically applied to the data base files to restore the affected records to the state they were in just before the beginning of the transaction.

If schema and area identifications change because a master directory is modified, the transaction recovery file must be reinitialized by the DBREC utility.

JOURNAL LOG FILE REPORT GENERATION

Reports can be generated from records in the journal log file. The records provide a historical record of users' access to data base files and are a source of statistical information about data base use. A log record is generated each time an update, write, or delete operation occurs on an area for which logging of before- or after-image records has been selected. In addition, a log record is generated for each of the following operations if a journal log file has been specified for the schema: begin transaction, commit transaction, drop transaction, version change, open, close, privacy breach, invoke, terminate, and recovery point. The close log record contains information on the number of times records in an area have been read, written, rewritten, and deleted by a particular user. Terminate log records contain summary counts for all activities on all areas accessed by the user.

By using FORM, Query Update, or a COBOL program to read the journal log file and create a log report, the contents of the journal log file can be examined and the state of the data base can be determined before beginning a recovery or restoration sequence. Reports consisting of information by area, user, time, record type, or a combination of these can be generated.

If Query Update is used to generate a report, the journal log file should be described with the Query Update DESCRIBE directive. Then a formatted log report can be prepared from the data in the journal log file using Query Update report preparation facilities. The prepared report can be examined at a terminal or it can be printed. The Query Update reference manual contains specific information on describing files and generating reports.

FORM, a general-purpose file management utility for manipulating records and converting files, can be used to print the journal log file. Format options can be selected. The information needed to use this utility is located in the FORM reference manual.

A log report can also be produced through the use of COBOL. A COBOL program can be written to extract and format the pertinent information in the journal log file. Routines can be written to format report pages, or the Report Writer statements can be employed to define a report format and to generate the log report. The COBOL reference manual gives information on report generation through the Report Writer.

RECOVERY CONSIDERATIONS

Selection of the log files for recovery purposes is dependent on a number of factors, such as the sensitivity of the data to loss, available mass storage, system stability, and performance considerations. Maximum benefit is achieved when the full complement of log files and logging options is selected; however, there can be a significant penalty in terms of performance for some application environments. The data administrator must consider the available trade-offs when implementing the recovery strategy for a data base.

Backup data base files generated by dumps of the data files are required for some recovery strategies. The data administrator must determine which files require backup dumps and the frequency the dumps should be taken.

CDCS achieves coordination of logging to the transaction recovery file, quick recovery file, and journal file along with writing updates to data base files through recovery points. Therefore, an understanding of the sequence of events at recovery points can help the data administrator select appropriate logging options for the particular data base environment.

RECOVERY POINTS

A recovery point is a point in data base processing to which recovery can be carried with no loss of data. Recovery points are either system-generated or user-generated.

System-generated recovery points are initiated when particular events occur in processing; some of these events are mentioned in the following discussion.

User-generated recovery points are initiated by a direct call to the DB$RPT routine from a user program. Each application programming language is provided with a mechanism for calling this routine; refer to the CDCS 2 Application Programming reference manual for further information. For recovery purposes, the use of data base transactions are recommended for application programs rather than user-generated recovery points. The creation of a recovery point does incur system overhead, since CDCS activity halts for some users until recovery point processing is completed. Overuse of user-generated recovery points can severely decrease throughput.

Recovery points occur on both the quick recovery and journal log files.

Quick Recovery File Recovery Points

The following events occur at a quick recovery file recovery point:

- CRM data file buffers are force written to disk.
- The quick recovery file is reinitialized as an empty file.

No recovery point number is assigned for a quick recovery file recovery point.

The following events initiate a quick recovery file recovery point:

- A journal log file recovery point.
- A user request to commit a transaction.
- The end of CDCS automatic recovery operations.
- Any event that would cause a journal log file recovery point, if there is no journal log file assigned for the schema.

After a quick recovery file recovery point, the data base files are in a consistent state because all data base updates have been force written to disk.

Journal Log File Recovery Point

The following events occur at a journal log file recovery point:

- A quick recovery file recovery point is forced.
- A recovery point number is assigned and a recovery point record logged.
- The journal log file record header is rewritten with the recovery point number and location of the new recovery point log record.

The following events initiate a journal log file recovery point:

- A user request for a recovery point
- The termination of a user from CDCS
- A full-file condition for the journal log file
- A full-file condition of the quick recovery file

Recovery Point Processing With No Log Files

Recovery point processing is automatically performed by CDCS if no journal log or quick recovery file is used. At any event that would cause a journal log file recovery point, the CRM data file buffers are force written to disk.

LOGGING OPTIONS

Log files provide capabilities for both automatic and manual recovery. The data administrator must consider strategies for manual recovery along with requirements for automatic recovery when selecting logging options.

Logging to the Transaction Recovery File

Logging to this file must be selected to use the facilities of automatic recovery and to provide for data base transactions being performed by application programs.

Selection of this option guarantees recovery of the affected data base files to the last committed transaction before a failure. This guarantee assumes that the data file is usable after the failure so that automatic recovery can perform recovery operations.

Logging to the Restart Identifier File

Logging to this file must be selected for COBOL and FORTRAN application programs to be provided the restart capability of determining the last committed transaction before a failure. If logging to this file is selected, logging to the transaction recovery file must also be selected.

Logging to the Quick Recovery File

The purpose of the quick recovery file is to permit the recovery of a data base file after system failure has occurred while CRM was in the process of updating one of the files. A failure can leave a file unusable because of inconsistencies between index blocks, data blocks, and other file status blocks. Data base files are only rarely left in an unusable state. If a file is unusable, no user access to the file is allowed.

If a quick recovery file recovery point occurred just before a failure, data base files are in a consistent state and, therefore, protected from becoming unusable.

If quick recovery file logging is selected and a data file is unusable after a failure, CDCS can perform automatic recovery at CDCS initialization. CDCS attaches the quick recovery file and by using its contents restores the file to the state it was in at the time of the last quick recovery file recovery point.

The consequences of not selecting quick recovery log file logging could be temporary loss of a data base file. If the data file is unusable, automatic recovery cannot be performed on the file and no subsequent access is allowed on the file. Manual recovery must be performed.

A manual recovery strategy can be to select the after-image record journal logging option. If this option is selected, an unusable data file can be reconstructed by use of the DBRCN utility, which applies after-image records to a data base file that has been reloaded from a backup copy of the data file.

Another consequence of not selecting the quick recovery file option could be the loss of updates made to data base records. Any record updated outside of a data base transaction since the last quick recovery file recovery point could be lost.

On a highly stable system with a small data base that has a high volume of updates, the data administrator could decide not to select logging to a quick recovery file. The data base could still occasionally have to be reconstructed using the DBRCN utility or be reset to its state at the time of the last set of file dumps if power or system failures occur.

Logging After-Image Records to a Journal Log File

Journal log file after-image records are required to reconstruct the data base file. The after-image log records can be used during automatic recovery or for manual recovery using the DBRCN utility. The after-image record option does not provide for reversing updates to restore a data base to a previous state.

During automatic recovery when both the quick recovery file and the journal log file after-image records are logged for a file, CDCS applies the contents of the quick recovery file to return the file to the state it was in at the last quick recovery file recovery point. Then CDCS applies the after-image records of the journal log file to the data file to apply all the updates made to the file since the last quick recovery file recovery point.

The journal log file after-image records are not required to guarantee that the data base is updated with the last committed transaction. All committed transactions are a part of the data base after any automatic recovery.

Selection of the after-image record option provides for data base file reconstruction in the event the data base file is unusable because of some failure and there is no quick recovery file. This can be a good mode of operation on stable systems with a small but active data base. File dumps could be taken frequently to reduce the time of reconstruction.

The consequences of not selecting a quick recovery file or journal log file after-image records could be the loss of all processing since the last set of file dumps.

Logging Before-Image Records to a Journal Log File

Before-image records are necessary to restore a data base to a previous state. Restoration must be done through manual recovery by use of the DBRST utility. Before-image records are not used by automatic recovery. The before-image record option does not provide for reconstructing a data base.

MANUAL DATA BASE RECOVERY

Manual recovery is usually needed to reconstruct an unusable file or to reverse the effects of invalid application program logic. These recovery processes cannot be done automatically. Automatic recovery performs most of the necessary recovery tasks that ensure the validity of the data base following a failure by CDCS, the operating system, or hardware.

The data base utilities and the logging facilities provide the tools needed by the data administrator for manual data base recovery. The data base utilities apply selected entries from a journal log file to reconstruct or restore one or more data base areas (and versions, if applicable). The two utilities, DBRCN and DBRST, perform reconstruction and restoration, respectively.

DBRCN can recover one or more data base files that have been lost or damaged. This utility takes after-image records recorded in the journal log file and applies them to a data base backup copy that has been previously dumped.

DBRST restores a data base to a previous state. DBRST works from the current data base, applies before-image records in the journal log file, and reverses the results of invalid operations performed on the data base. When system failure occurs, the quick recovery file could have to be applied (by using the DBQRFA utility) to make the data base physically processable before DBRST can be used to restore the data base.

Both reconstruction and restoration can be performed selectively by area, version, user, date, and time. Reconstruction or restoration to a specific recovery point is also provided by the utilities.

RECOVERY CONDITIONS

The recovery sequence chosen by the data administrator is dependent upon the type of failure that has occurred and the logging techniques employed. Three examples of recovery from computer and software failure are given in the following paragraphs. If all recovery options have been selected, a data base can be recovered as described.

Physical Storage or Software Failure

When an entire data base and all disk resident log files are lost because of a disk malfunction or a software failure, every update to the data base since the last journal log file dump is lost. All updates performed before the last file dump are recorded on the journal log file.

The first step to take for recovery is to establish a data base from the latest set of data base file dumps. Then, using the DBRCN utility, the after-image records on the journal log file can be used to update this data base. This procedure should reconstruct the data base to its previous state before the last journal log file dump.

Cascade Effect

The cascade effect problem should be considered. This problem occurs when a user erroneously updates a data base and another user processes the erroneous data before the data administrator can initiate a restoration sequence to correct the data base. The restoration utility cannot detect this problem. This problem can be solved through forethought in the way applications are written, through consideration of the time of day at which they are executed, and through careful use of the DBRCN and DBRST utility by the data administrator. It might be necessary to initiate several consecutive executions of these utilities in order to recover a data base fully.

Program Logic Error

When a previously executed program has been found to contain a program logic error, all its updates to a data base are suspect. Using the DBRST utility and the before-image records on the journal log file from before the program first executed, all the incorrect updates can be removed from the data base.

Some problems are associated with this type of recovery. First, if the erroneous data in this data base has been used, the errors may have cascaded throughout the data base. Second, recovery is on a record basis. Using the DBRST utility causes updates to the affected records performed by other programs to be lost when they may be perfectly valid. The recovery utilities cannot solve this problem.

System Failure

If a failure occurs that causes the system to crash, a data base is still largely intact. If block logging is specified, CDCS performs automatic recovery to ensure the usability of the data base files. If, however, block logging has not been specified for affected data base files and the files are determined by CRM to be unprocessable, the data base files must be reloaded from file dumps and reconstructed if after-image records are available.

RECOVERY FROM RECOVERY FAILURES

When the DBRST or DBRCN utility does not terminate normally, several options are available to the data administrator.

If the recovery failure was not caused by media failure on the log file or the data base, the data administrator can rerun the utility provided the following conditions are met:

- The same input parameters are used.
- The original journal log file is used as input.
- Block logging to the quick recovery file was specified for the schema and for the affected data base files.

If the NL parameter in the utility control statement was specified, a new log file is generated during the new recovery run, but it will contain only records for updates that were successfully performed in the previous recovery run. DBRST could be run to remove updates logged during the utility run that aborted but need not be run if the recovery run is to be repeated.

If the failure was caused by or resulted in media failure, the data base must be reconstructed by DBRCN from file dumps and journal log files. Then, if the log files are intact from the original run, the original recovery run can be performed if necessary.

DATA ADMINISTRATOR AND OPERATING SYSTEM PROCEDURES

Data administration is a major concept in the data base environment. The data administrator is a person or group in a user installation responsible for defining the data bases, determining who has access to them, ensuring data base security and integrity, and managing the execution of CDCS.

The data administrator prepares the initialization procedures for CYBER Database Control System (CDCS) and defines the system procedure file used in initializing the system. Accounting charges for CDCS usage and data base input/output can be adjusted subject to the data administrator's discretion. This section details CDCS initialization and provides information the data administrator needs to establish operating procedures.

This section also indicates the data administrator's responsibilities and concerns in overseeing the data base environment. For an overview, the sequence of operations involved in establishing an active data base environment is shown in figure 10-1. The operations are shown along with resulting files and directories. (The operations performed by the data administrator are shaded.) An arrow indicates direct information transfer from the directory to the next unit. Files or directories shown above an operation need to be present for the operation. However, particular data base files need not be available for CDCS to be either initialized or active. If a particular data base file is not available, run-units that require the file cannot execute. Similarly, if log or recovery files and a data base procedure library are specified for a schema but the files are not available, a run-unit cannot invoke a subschema based on that schema because of unavailable files.

DATA ADMINISTRATOR'S RESPONSIBILITIES

The data administrator should be concerned with the following aspects of the data management system:

CDCS operational environment, including the console operator interface

Field length requirements for CDCS

Data base files

Data base procedures

Schema, subschema, and master directory creation and maintenance

Data base recovery, including log and recovery files

System limitations

Data privacy

Application program interface

System accounting

Information about many of these aspects is contained in other sections of this manual: the schema (section 2), subschemas (sections 3 and 4), master directory (section 8), data base procedures (section 7), and data base recovery (section 9). The CDCS environment is discussed in section 1 where processing flow, CDCS loading and execution, and the system control point concept are described. Additional information is contained in this section and in appendix G, which details field length requirements for CDCS and the data base utilities. Problems associated with inadequate field length are discussed also. The CDCS Batch Test Facility, which allows CDCS and an application program to be run at the same control point, is described in the CDCS 2 Application Programming reference manual. Data base files are described in sections 1 and 6; however, additional information about privacy of data base files and about the CYBER Record Manager (CRM) interface appears later in this section.

SYSTEM LIMITATIONS

System limitations imposed by the Data Description Language (DDL) limit to 4095 each the number of areas, records, items in a record, relations in a schema, constraints in a schema, or versions for a schema. In practice, the number of items in a record is smaller because of the presence of variable length subentries. The number of these entries that can be declared in a subschema is a function of DDL and CDCS internal table requirements. The number that can be accommodated, however, should be adequate for all practical cases. Another limit imposed by DDL concerns data base procedures: a maximum of 600 unique data base procedures is permitted in a schema.

DATA PRIVACY

Data privacy can be maintained by features of CDCS and by file security features of the operating system. It is the responsibility of the data administrator to implement the use of privacy features.

Privacy (Access Control) Checking

CDCS supports two types of privacy (access control) checking. The data administrator can supply a data base procedure for privacy checking, or access control locks can be declared in the schema for use in automatic access validation by CDCS. Checking of access control keys against lock values by CDCS has the advantages of simplicity, ease of implementation, and better performance. Access control locks are encrypted, which provides additional security. This feature is not available if data base procedures are used for privacy checking.

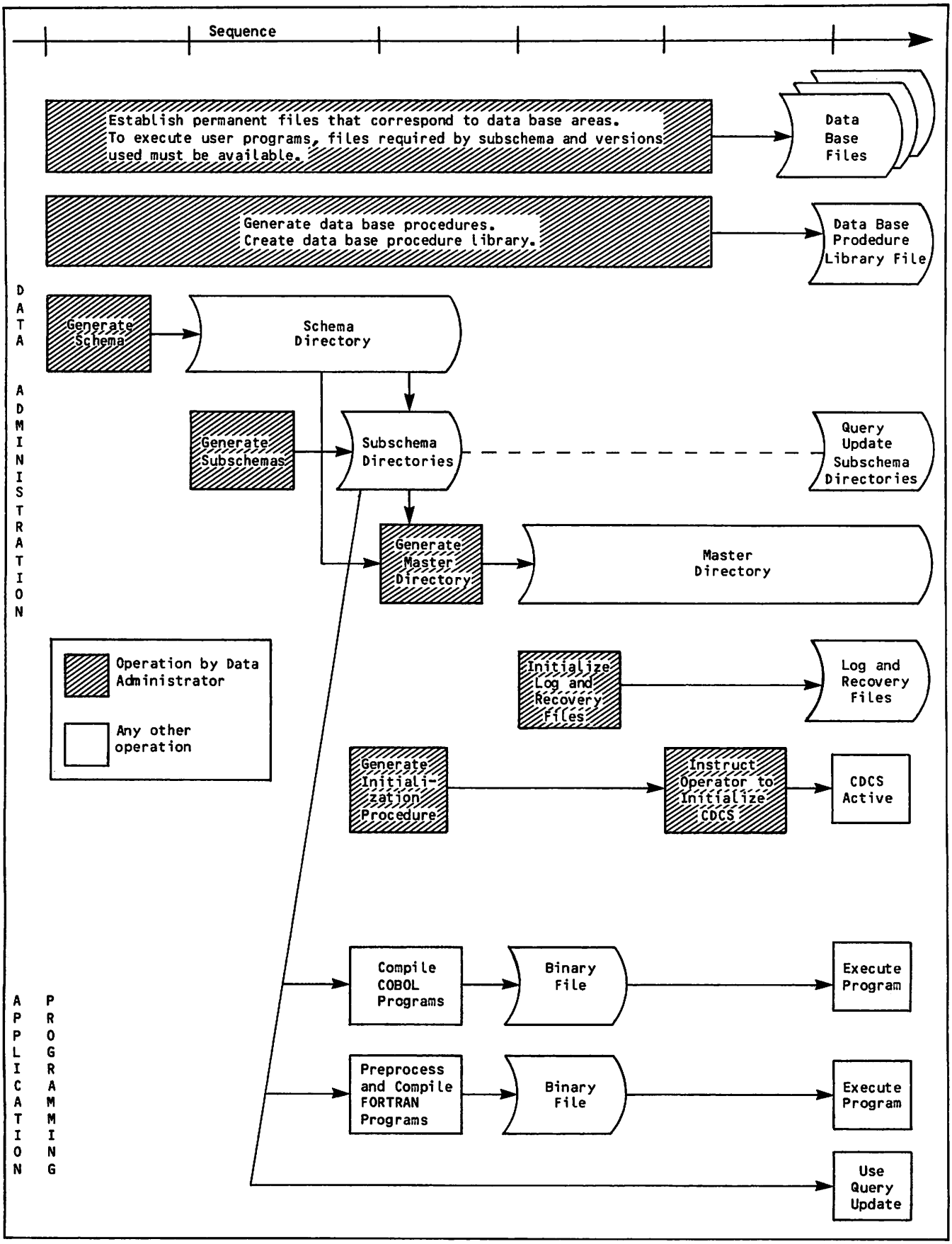


Figure 10-1. Sequence of Operations to Establish an Active Data Base Environment

Access control locks provide privacy at the area level. The ACCESS-CONTROL clause in the schema specifies the access control locks that apply to the use of an area. At execution time, the applications program must supply the appropriate access control key (also called privacy key) to gain access to a given area. If data base versions apply, the same access control lock is in effect for the same area of each version.

Data base procedures for privacy checking are most useful for controlling access based on a program's job identification, which CDCS does not do. The procedures make the decision whether to allow the use of a data base area or an input/output function for a data base area, based on the access control key supplied by a program and possibly on the job name of the program. The program must supply an access control key in order for the data base procedure to be called; the key can be a dummy key if it is not used in the decision made by the data base procedure to allow access. Data base procedures are loaded and executed at the system control point.

Operating System File Security

Data base files (that is, files associated with areas) must be established as permanent files according to operating system procedures. When establishing the permanent files, the data administrator should use the file security features provided by the operating system. For use under NOS, the files must be direct access files.

When used in the data base environment, the data base files are attached at the system control point by CDCS. CDCS, therefore, is the only job that provides user number or user identification, provides passwords, and specifies type of access for which the data base file is attached. CDCS uses the information in the master directory for attaching the data base files.

For NOS, one password is allowed. If a password is specified, it must be included in the permanent file information in the master directory if the user number under which CDCS executes is different from the user number under which the file is defined. For file security, it is recommended that the data base file have the access category of private (CT=PRIVATE, which is the default category). By use of the PERMIT control statement, access to the file can be specified for any other user name for which the data administrator permits file use. Refer to the NOS reference manual for detailed information.

For NOS/BE, up to five passwords can be specified for a file. If passwords are specified, the permanent file information in the master directory must include the appropriate passwords to obtain the permissions for the intended use of the file. Refer to the NOS/BE reference manual for detailed information.

APPLICATION PROGRAMMING INTERFACE

The information required by application programmers to perform data base processing is provided by the subschema compilation listing and by the data administrator. This discussion assumes that the programmer is provided with the subschema listing. Refer to the CDCS 2 Application Programming reference manual for discussions of information provided by these listings. The data administrator should provide application programmers with any other information required to code and execute user programs.

The data administrator must provide the following information when it is applicable.

Access control keys (privacy keys)

> If an area has been defined with an access control lock in the schema, the application program must provide the appropriate access control key to gain access to the area. Information about all access control keys required by an application program must be provided to the application programmer.

Constraints

> If constraints are defined in the schema and updates are likely to violate them, the application programmer should be provided with information about constraints.

Permanent file information for the subschema library

> The subschema directory must be available to compile a COBOL program, to preprocess a FORTRAN program with the Data Manipulation Language (DML) preprocessor, and to use Query Update for data base access. Therefore, the application programmer must have the information required to access the file containing the subschema directory.

Join items of the relation

> Usually, an application programmer does not need to know the join items in a relation. Often the programmer can determine the join items from the subschema listing. However, in some situations, the application programmer should be provided with information about these items.

Requirements imposed by a data base procedure

> If data base procedures are defined in the schema and require that the user program provide information, the application programmer must be provided with the information required by the data base porocedure.

Transaction update limit

> Application programmers should be told about the update limit allowed for data base transactions involving a particular schema so that the programmers can code application programs to handle the situation of the update limit being reached.

Version names

> If an application program is to use a data base version other than version MASTER, the application programmer must be provided with the version name.

CDCS CONTROL STATEMENT

The CDCS control statement initiates CDCS and provides information for the running of CDCS. The CDCS control statement format and parameters are shown in figure 10-2.

The CDCS control statement cannot exceed 80 characters in length and must conform to the operating system syntax rules for control statements. Optionally, the comma immediately following CDCS can be replaced by a left parenthesis; and the terminating period can be replaced by a right parenthesis.

All CDCS control statement parameters are optional. Parameters can be specified in a directive file.

DIRECTIVE FILE

An optional directive file can be used to contain parameters in addition to, or instead of, the parameters in the CDCS control statement. Use of the directive file allows specification of a parameter list longer than that allowed in the control statement.

Parameters entered in the directive file must conform to certain rules. Parameters can be specified in any order. The MFL and DIR parameters cannot be specified in the directive file; any other valid parameter can be specified. No parameter specified in the directive file can duplicate a parameter specified in the control statement. Parameters can be specified in columns 1 through 80. The first parameter specified in a line must begin in column 1. Parameters can either be placed in separate lines or combined in a line with commas acting as separators. Parameters cannot be split across lines. Blanks, which are ignored, can be used to improve readability.

PARAMETERS

All the parameters (figure 10-2) of the CDCS control statement are optional and can be specified in any order. The CDCS control statement parameters perform the following functions:

- Allocation of maximum pooled buffer space
- Adjustment of accounting charges
- Allocation of the maximum field length that CDCS is allowed to use
- Control of loading certain CDCS overlay capsules for the duration of CDCS execution
- Control of retention of certain CYBER Record Manager (CRM) capsules
- Specification of information required to attach the master directory if online dumping of journal log files is desired
- Specification of a directive file which can contain any of the CDCS control statement parameters except MFL and DIR.

CDCS[,p] ... .

where p is a parameter. The parameters are as follows:

DIR=lfn	Directive file for CDCS control statement parameters
BL=nn	Maximum pooled buffer space
CP=t1	Central processor average time
IO=t2	Input/output average time
MFL=fl	Maximum field length for CDCS
SBL=sbl	Small block limit for CDCS
SBI=sbi	Small block boundary increment for CDCS
CAP=ovcap-name1 [/ovcap-name2]...	Static load of CDCS overlay capsules where possible OVCAP names are as follows: CON DBP INV JLOG QRF REL TRAN
CRM=fs1[/fs2]...	Static load CRM capsules where fs is a file structure as follows: AK DA IS MIP or MP
MDPFN=pfn	
UN=user-name	
ID=user-id	
PW=pwrd1[/pwrd2]...	
FAM=family-name	permanent file information for master directory file
PN=pack-name	
SN=set-name	
VSN=serial-number	
DT=device-set	

NOTE: The OVCAP names and CRM capsule loading parameters can themselves be parameters of the control statement.

Figure 10-2. CDCS Control Statement

The DIR parameter specifies the directive file from which CDCS parameters are read. The DIR parameter is interpreted as follows:

omitted

No parameters are read from a directive file; the CDCS parameters are completely specified in the CDCS control statement.

DIR

The local file INPUT is assumed to contain the directive file for the CDCS control statement.

DIR=lfn

The specified local file name identifies the directive file for the CDCS control statement.

The BL parameter specifies in octal the maximum pooled buffer space that is allowed to be allocated by Advanced Access Methods (AAM) for all open data base files during a CDCS session. The BL parameter is interpreted as follows:

omitted

The maximum pooled buffer space is computed by CDCS. The allowed space is a little over half of the memory that is available between the initial load field length and the maximum field length.

BL

The maximum pooled buffer space is the pooled buffer space usually allowed by the AAM allocation scheme.

BL=nn

The maximum pooled buffer space is defined as the value specified by nn, which must be an octal integer. The value of nn must be less than the maximum field length established by the specification of the MFL parameter or by a default of that parameter.

The CP parameter specifies the average central processor time in micro seconds required by CDCS for a random read by primary key on an indexed sequential file. The Accounting Statistics subsection contains more information about this parameter. The CP parameter is interpreted as follows:

omitted

No accounting takes place unless the IO parameter is specified.

CP

The average central processor time is defined as the value specified at software installation time. (The value suggested in the released version of the software is 4000.) If this form of the parameter is chosen, the accounting charge table in the CDCS code is used for computing CDCS accounting charges.

CP=t1

The average central processor time is defined as the value specified by t1. The value of t1 cannot exceed seven decimal digits in length. If this form of the parameter is chosen, all the CDCS values in the CDCS accounting table are adjusted based on the CP time specified.

The IO parameter specifies the average input/output channel time in microseconds required by CDCS for a random read by primary key on an indexed sequential file. The Accounting Statistics subsection contains more information about this parameter. The IO parameter is interpreted as follows:

omitted

No accounting takes place unless the CP parameter is specified.

IO

The average input/output channel time is defined as the value specified at software installation time. (The value suggested in the released version of the software is 7000.) If this form of the parameter is chosen, the accounting charge table in the CDCS code is used for computing CDCS accounting charges.

IO=t2

The average input/output time is defined as the value specified by t2. The value of t2 cannot exceed seven decimal digits in length. When this form of the parameter is chosen, all the values in the CDCS accounting charge table are adjusted, based on the IO time specified.

The MFL parameter specifies the maximum field length in octal that CDCS is allowed to use. The MFL parameter is interpreted as follows:

omitted

The maximum field length is the system default maximum.

MFL

The maximum field length is 120000 octal.

MFL=fl

The maximum field length is specified by fl. The value of fl cannot exceed seven octal digits in length.

The SBL and SBI parameters are provided for reduction of memory fragmentation. CDCS establishes a boundary beyond which CYBER Memory Manager will not assign small blocks. A small block is one which is less than 240 octal words.

The SBL parameter specifies a maximum limit beyond which the boundary will not be moved. If SBI is not specified, SBL specifies a fixed boundary location. See appendix G for a discussion of field length requirements and optimizations. The SBL parameter is interpreted as follows.

omitted

The small block limit is equal to the maximum field length.

SBL

The small block limit is equal to the maximum field length.

SBL=sbl

The small block limit is specified by sbl. If this number is equal to or larger than the maximum field length, it will have no effect.

The SBI parameter specifies that CDCS should increase the small block boundary in increments of sbi. Each increment will be added only after CDCS and CRM have released any blocks that are releasable at that time. The small block boundary will not be advanced beyond the limit set by the SBL parameter. The SBI parameter is interpreted as follows:

omitted

The small block boundary is not varied from that specified by SBL.

SBI

The small block boundary is not varied from that specified by SBL.

SBI=sbi

The small block limit increment is specified by sbi.

The following CDCS OVCAP loading parameters control loading of certain CDCS overlay capsules during CDCS initialization. Specification of these parameters causes the overlay capsules to be kept in memory during the entire time CDCS is executing at a system control point. It is not necessary to specify one of these parameters in order to be able to use the corresponding feature. The following OVCAP parameters are available:

The CON parameter is interpreted as follows:

omitted

The constraint overlay capsules are loaded and unloaded according to use.

CON

The constraint overlay capsules are loaded at CDCS initialization time and remain loaded for the duration of the CDCS session.

The DBP parameter is interpreted as follows:

omitted

The data base procedure overlay capsules are loaded and unloaded according to use.

DBP

The data base procedure overlay capsules are loaded at CDCS initialization time and remain loaded for the duration of the CDCS session.

The INV parameter is interpreted as follows:

omitted

The invoke overlay capsules are loaded and unloaded according to use.

INV

The invoke overlay capsules are loaded at CDCS initialization time and remain loaded for the duration of the CDCS session.

The JLOG parameter is interpreted as follows:

omitted

The journal log file overlay capsules are loaded and unloaded according to use.

JLOG

The journal log file overlay capsules are loaded at CDCS initialization time and remain loaded for the duration of the CDCS session.

The QRF parameter is interpreted as follows:

omitted

The quick recovery file overlay capsules are loaded and unloaded according to use.

QRF

The quick recovery file overlay capsules are loaded at CDCS initialization time and remain loaded for the duration of the CDCS session.

The REL parameter is interpreted as follows:

omitted

The relations overlay capsules are loaded and unloaded according to use.

REL

The relations overlay capsules are loaded at CDCS initialization time and remain loaded for the duration of the CDCS session.

The TRAN parameter is interpreted as follows:

omitted

The data base transaction overlay capsules are loaded and unloaded according to use.

TRAN

The data base transaction overlay capsules are loaded at CDCS initialization time and remain loaded for the duration of the CDCS session.

The CAP parameter offers an alternate form in which to specify the CON, DBP, INV, JLOG, QRF, REL, and TRAN parameters. The CAP= portion of the parameter functions as documentation only. The CAP parameter is interpreted as follows:

omitted

Any of the CDCS overlay capsules that have not been specified by other CDCS OVCAP loading parameters are loaded and unloaded according to use.

CAP=ovcap-name1[/ovcap-name2]...

The CDCS overlay capsules specified by ovcap-name1 and ovcap-name2 are loaded at CDCS initialization and remain loaded for the duration of the CDCS session. The values defined for ovcap-name1 and ovcap-name2 are CON, DBP, INV, JLOG, QRF, REL, and TRAN. Refer to the description of the individual parameter for the interpretation of each option. If more than one CDCS OVCAP parameter is specified, the options must be separated by a slash.

The following CRM capsule loading parameters control loading of certain CYBER Record Manager (CRM) capsules during CDCS initialization. Specification of these parameters causes the capsules to be kept in memory during the entire time CDCS is active. It is not necessary to specify one of these parameters in order to be able to use the corresponding feature. Before specifying any of these parameters, refer to the Duration Loading subsection.

The AK parameter is interpreted as follows:

omitted

The extended actual key file structure capsules are loaded and unloaded according to use.

AK

The extended actual key file structure capsules are loaded at CDCS initialization and remain loaded for the duration of the CDCS session.

The DA parameter is interpreted as follows:

omitted

The extended direct access file structure capsules are loaded and unloaded according to use.

DA

The extended direct access file structure capsules are loaded at CDCS initialization and remain loaded for the duration of the CDCS session.

The IS parameter is interpreted as follows:

omitted

The extended indexed sequential file structure capsules are loaded and unloaded according to use.

IS

The extended indexed sequential file structure capsules are loaded at CDCS initialization and remain loaded for the duration of the CDCS session.

The MIP or MP parameter is interpreted as follows:

omitted

The extended multiple index processor file structure capsules are loaded and unloaded according to use.

MIP

The extended multiple index processor file structure capsules are loaded at CDCS initialization and remain loaded for the duration of the CDCS session. The extended indexed sequential capsules are also loaded, whether or not they are explicitly specified.

MP

Same as MIP.

The CRM parameter offers an alternate form in which to specify the AK, DA, IS, and MIP parameters. The CRM= portion of the parameter functions as documentation only. The CRM parameter is interpreted as follows:

omitted

Any of the CRM file structure capsules that have not been specified by other capsule loading parameters are loaded and unloaded according to use.

CRM=fs1[/fs2] ...

The CRM file structure capsules specified by fs1 and fs2 are loaded at CDCS initialization and remain loaded for the duration of the CDCS session. The values defined for fs1 and fs2 are AK, DA, IS, and MIP (MP). Refer to the description of the individual parameter for the interpretation of each option. If more than one file structure option is specified, the options must be separated by a slash.

The master directory file attach parameters consist of a series of parameters that can be used to specify the permanent file information required to attach the master directory file. These parameters must be specified if automatic online dumping of the journal log file is to be performed by CDCS. If the master directory file is attached externally prior to execution of the CDCS control statement, specification of these master directory file attach parameters causes CDCS execution to abort. All of the master directory file attach parameters, except for VSN and DT, must contain only letters or digits and must not exceed seven characters in length.

If automatic CDCS online dumping of the journal log file is desired, master directory file attach parameters must be specified. The parameters are used by CDCS to construct the DBREC job that dumps the log file. For the DBREC job, a minimum set of master directory file attach parameters must be specified. The minimum set of parameters for the NOS operating system consists of MDPFN and UN. The minimum set of parameters for the NOS/BE operating system consists of MDPFN and ID. The master directory attach parameters are defined as follows:

MDPFN=pfn

The permanent file name of the of the file containing the master directory.

UN=user-name

The user name of the owner of the permanent file containing the master directory (applicable for NOS).

ID=user-id

The name of the owner of the permanent file containing the master directory (applicable for NOS/BE).

PW=pwrd1[/pwrd2] ...

The operating system password (pwrd1) or passwords used to restrict access to the permanent file containing the master directory. Up to five passwords can be specified for NOS/BE; on NOS only the first password is used. If more than one password is used, the passwords must be separated by a slash.

FAM=family-name

The family name of the device on which the permanent file containing the master directory resides (NOS only). Refer to the NOS reference manaual for more information about devices associated with a family name.

PN=pack-name

The name of the auxiliary device on which the permanent file containing the master directory resides (NOS only). Refer to the NOS reference manual for more information about auxiliary devices.

SN=set-name

The name of the private device set on which the permanent file containing the master directory resides (NOS/BE only). If SN is specified, VSN must also be specified. If SN is specified, the master member of the set must have been mounted via a MOUNT control statement in the CDCS initialization procedure. Refer to the NOS/BE reference manual for more information about private devices.

VSN=serial-number

The volume serial number of the master pack of the private device set on which the permanent file containing the master directory resides (NOS/BE only). Refer to the NOS/BE reference manual for more informa- tion about this parameter.

DT=device-type

The auxiliary mass storage device type on which the permanent file containing the master directory resides (NOS only). An auxiliary device is any supported device that an installation defines as auxiliary.

ACCOUNTING STATISTICS

Accounting charges for CDCS central processor usage (CP) and data base input/output processing (IO) are reflected in the accounting statistics that appear in the user's control point dayfile. Specific charges are for central processor time and input/output processing time attributable to a specific user request. The CP and I/O times are charged to the user job by CDCS. Under NOS/BE they are added directly into the CP and I/O times that are accumulated by the user job. Under NOS, the CP and I/O times are accumulated together into another catagory called application unit charge. This catagory appears on the user job dayfile as UEAC units. Under NOS and NOS/BE, the following two informative messages are sent to the user control point dayfile when the program terminates from CDCS:

CP SECONDS CHARGED BY CDCS xxxxxx.xxx

IO SECONDS CHARGED BY CDCS yyyyyy.yyy

The first message indicates the central processor time in seconds used by CDCS in processing requests for the user. The second message indicates the channel time in seconds used by CDCS in processing requests for the user.

The application unit charges are transferred to the user job on each request. They are subject to rounding errors and conversion ratios used by the operating system, so they will not exactly match the total seconds shown in the dayfile messages.

The data administrator has three choices concerning accounting charges:

To use the accounting charge table included in the CDCS code

To tune accounting charges for the particular installation

To not calculate charges for the individual users

Specification of the IO or CP parameter in the CDCS control statement implements accounting procedures. If the data administrator wants accounting to be performed and wants to use the values in the accounting charge table included in the released version of the software, the CP and IO parameters must be specified in the CDCS control statement as shown in the following examples:

CDCS,CP,IO.

CDCS,CP.

CDCS,IO.

If the data administrator wants to tune accounting charges, the specific values desired must be specified in the CP and IO parameters in the CDCS control statement. If the data administrator wants no calculation of charges for CDCS users, the CP and IO parameters must not be specified in the CDCS control statement.

CDCS is released with an accounting charge table generated for a CYBER 170 Model 74. If the data administrator determines that charges made to users from this charge table require adjustment for the installation's needs, the parameters on the CDCS control statement can be changed so that the charging mechanism is more suitable. Since the accounting charges are all relative to the CP and IO time for an IS random read through CDCS, the entire table can be adjusted if new values for these times are specified.

The elements that affect accounting charges differ depending on the type of CDCS request. Charges for CDCS IO requests are based on the number of keys for the file, the file organization, the number of data base accesses, and logging. Charges for CDCS non-IO requests are a fixed cost plus possible logging charges per request.

When CDCS terminates, accounting statistics are sent to the CDCS system control point dayfile. These statistics can be used by the data administrator to determine what adjustments need to be made to the accounting charge table to minimize the discrepancy between amount charged and amount used.

Under NOS/BE and NOS, the following two messages are sent to the dayfile:

```
CDCS CHARGED      CP xxxxxx.xxx
                  IO yyyyyy.yyy

CDCS USED         CP aaaaaa.aaa
                  IO bbbbbb.bbb
```

The values denoted as lowercase letters are in seconds. The values returned in the CDCS CHARGED message represent the total charges made by CDCS for all requests CDCS users made during the time CDCS was active (that is, from the time CDCS was initialized until the time CDCS was terminated). The values returned in the CDCS USED message represent the total resources used by CDCS during the time CDCS was active.

Under NOS, an additional message is sent to the CDCS dayfile and appears as follows:

```
CDCS zz.zz PERCENT CPU USAGE
```

PERCENT CPU USAGE is defined as the ratio of the total CP time used to the real time elapsed while CDCS was active.

DURATION LOADING

The capsule loading parameters of the CDCS control statement allow specification of certain CDCS overlay capsules and CYBER Record Manager (CRM) capsules to be loaded during CDCS initialization and kept in memory during the entire time CDCS is active. This procedure is called duration loading. It is aimed at improving CDCS performance in a situation of heavy usage of a particular feature. The parameters need not be specified in order to use the corresponding feature. The parameters should be specified if the feature is in regular use because the corresponding capsules are loaded in a contiguous group and retained in memory during the entire execution of CDCS. In this case, specifying the parameters avoids the memory fragmentation and extra loading time that occurs when the capsules are each loaded as they are used. Specifying parameters for infrequently used CDCS overlay capsules or CRM capsules should be avoided because this causes memory for these capsules to be allocated for long time periods when the capsules are not in use.

CDCS LOADING OF OVERLAY CAPSULES

When the CDCS OVCAP loading parameters are specified, CDCS loads the overlay capsules for each feature specified at initialization time. If the OVCAP loading parameters are not specified, the overlay capsules which support a feature are loaded and unloaded according to use.

CDCS LOADING OF CRM CAPSULES

When the CRM capsule loading parameters are specified, CRM selects the capsules for each file structure, and loads them at initialization. CRM includes those capsules required for routine reading and updating; it excludes capsules that are required for exceptional processing such as block splits. The CRM capsule loading parameters need not be specified in order to use the corresponding file structure feature, such as MIP processing of alternate keys.

CLASSIFICATION AND EFFECT OF CRM ERRORS

CDCS classifies the CRM errors returned on data base files. A class 1 error indicates that the data base file can be structurally bad and cannot be used. If a class 1 error occurs, the data administrator must reconstruct the file. A class 2 error indicates an operation that is not allowed has been attempted on the data base file by a particular application program. If a class 2 error occurs, no action by the data administrator is required.

The action that CDCS takes when a CRM error occurs depends on the classification of the error. If a class 1 error is returned, CDCS aborts all application programs using the file, gives the file a down status and permits no further use of the file. CDCS also writes an error message to the CDCS output file indicating that the file could be structurally bad.

When a class 1 error is returned, the file must be recreated by the data administrator before subsequent access is attempted. The data administrator could develop administrative procedures to inquire about the status of data base areas in order to detect the down status of an area (file). Refer to the description of the STATUS command in the Operator Interface subsection for further information.

If a class 2 error is returned, CDCS returns the CRM error information through a CDCS diagnostic to the application program that caused the error and writes a message to the CDCS output file. The CDCS diagnostic returned depends on the severity of the CRM error. If the CRM error is a fatal error (as indicated in the CYBER Record Manager Advanced Access Methods reference manual), CDCS issues a nonfatal error and prohibits the application program from having further access to the file. If the CRM error is a nonfatal error, CDCS issues a trivial error. Upon issuing either a nonfatal or trivial error, CDCS terminates the request for data base access that caused the error.

For information about CRM error codes and messages, refer to the CYBER Record Manager Advanced Access Methods reference manual.

CRM ERROR FILE USE

The data administrator can use the CRM error file ZZZZZEG at the CDCS system control point to obtain statistics and other information about processing of data base files. However, the journal log file should be used instead of the error file because difficulties could arise when using the error file. The following paragraphs describe the problems that can occur when the CRM error file is used at the system control point.

When CDCS opens a data base area, an internal file name different from the schema area name is generated. This is the file name referenced in the ZZZZZEG error file at the system control point.

CRM refers to the DFC and EFC fields in the FIT to determine whether error messages or statistics/notes for an area are to be written to the dayfile or error file, or both. To control writing of error messages and statistics/notes, the DFC and EFC fields must be specified in the FILE control statement for an area at schema compilation time.

If file statistics are written to the error file from a number of user jobs running with CDCS, the error file can become very large.

CDCS LISTING

The CDCS output file contains a list of significant user messages issued during CDCS execution. Each message is identified by a time stamp and a user identification. CDCS writes the message list information on the file named OUTPUT. CDCSBTF writes the message list information on a file named CDCSOUT.

Each time twenty-five pages of messages are accumulated , the OUTPUT file is copied to another file and routed to a printer. If the CDCS user message listing is the first record on the OUTPUT file, the file is rewound after the copy. If there are other records ahead of the CDCS listing, the OUTPUT file is positioned at the first end-of-record. This preserves listing information that may have been placed on the OUTPUT file by CDCS initialization procedures.

SYSTEM OPERATOR GUIDELINES

The following subsection details the procedure for CDCS initialization and reviews some operations

information concerning the system control point. This information is given so that the data administrator can set up appropriate installation operating procedures for CDCS. Following the initialization discussion, information concerning the communication between CDCS and the system operator that can be achieved through the central programmable displays is presented. Information is communicated to CDCS by the system operator under the guidance of the data administrator. Refer to the appropriate operating system-related reference manuals and operator's guides for additional information.

CDCS INITIALIZATION

The system operator initializes CDCS at a system control point by typing in one of the following commands:

```
n.CDCSx.  (for NOS)

n.X CDCSx.  (for NOS/BE)
```

The n designation is the control point number; x can be one or two alphanumeric characters. These commands execute a system procedure file defined with the name CDCSx. The system procedure file must contain control statements to set system resource requirements. Exit processing should also be specified to provide diagnostic information in case of a CDCS failure. Information for attaching the master directory must be provided as indicated in the description of the system procedure file that follows. The method used to define the system procedure file differs under the NOS and NOS/BE operating systems.

The system operator should not adjust the time or date for the system if logging to journal log files is in progress.

If permanent file information in the CDCS control statement or in the master directory specifies the use of private packs or devices, the private packs or devices should be online at the time CDCS is initialized.

At CDCS initialization time, all log files and data base procedure library files specified in the master directory should be avaliable to CDCS. If one of these files is not available, the affected schema is marked with the down status and cannot be used.

NOS System Procedure File

The system procedure file under NOS is generated by creating a CYBER Control Language procedure with the name of CDCSx (where x is one or two alphanumeric characters). This procedure must contain appropriate control statements and must be saved as a permanent indirect access file with the same name as the procedure. The system operator moves the system procedure file to the system catalog by typing in the following:

```
x.MOVEPF(Fl=CDCSx/UI=a,DI=377777)
```

The specification a is the user index associated with the user number under which the system procedure file is saved. Any number of system procedure files can be created. The first three letters of the procedure file name must be CDC.

To give a user job access to CDCS, the MODVAL function must be used by the operator to set the CUCP bit in the user's validation file access word. This procedure allows the user job to access a system control point. Refer to the NOS installation handbook for a discussion of MODVAL.

The master directory can be attached either internally by CDCS using information provided in the master directory file attach parameters of the CDCS control statement or externally by specification of an ATTACH control statement prior to execution of the CDCS control statement. The internal method of attaching is recommended. If master directory file attach paramaters are specified in the CDCS control statement, the master directory file cannot be attached externally. A master directory attached externally by the system procedure file must be attached by using the local file name MSTRDIR.

Control statements for the journal log file, the transaction recovery file, the restart identifier file, the quick recovery file, and the data base procedure library must not be present in the system procedure file; the files are automatically attached by CDCS using permanent file information in the master directory.

A sample system procedure file for NOS is shown in figure 10-3. Either the MFL control statement or the MFL parameter on the CDCS control statement can be used to set maximum field length.

The USER control statement specifies the user name under which CDCS executes. This control statement is required if particular files used by CDCS are not defined as PUBLIC. (In this situation, a PERMIT control statement is used to specify that a file should be accessible to the user name under which CDCS will execute.) The USER control statement affects access for the following files: master directory file, data base and index files, and log and recovery files. If all of these files are PUBLIC, the USER control statement is not required.

NOS/BE System Procedure File

The system procedure file under NOS/BE is created by the data administrator using EDITLIB. The file is placed on the system library with an access level that allows it to be called from control statements. Any number of procedure files can be created. A sample system procedure file for NOS/BE is shown in figure 10-4.

On the CDCS control statement, the MFL parameter specifies the upper limit on the field length for CDCS. The field length, however, could be considerably less at any point during execution, depending on the number of users and the requests outstanding.

```
.PROC,CDCSx.                    Establishes the CYBER Control Language Procedure
USER,username.                  Specifies the user name for CDCS
RETURN,CDCSx.                   Returns the system procedure file
MFL,xxxxxxx.                    Establishes maximum field length
CDCS,MDPFN=xxxxxxx,UN=xxx.      Initiates CDCS; attaches master directory
EXIT.                           Establishes processing if error occurs
DMD.                            Prints exchange package dump
DMD,xxxxxx.                     Prints the contents of selected areas of memory
```

Figure 10-3. Sample NOS System Procedure File

```
.PROC,CDCSx.
CDCS,MDPFN=xxxxxxx,ID=xxx.      Initiates CDCS; attaches the master directory
EXIT.                           Establishes processing if error occurs
DMD,xxxxxx.                     Prints the contents of selected areas of memory
```

Figure 10-4. Sample NOS/BE System Procedure File

The master directory can be attached either internally by CDCS using information provided in the master directory attach parameters of the CDCS control statement or externally by specification of an ATTACH control statement prior to execution of the CDCS control statement. The internal method of attaching is recommended. If master directory file attach paramaters are specified in the CDCS control statement, the master directory file cannot be attached externally. A master directory attached externally in the system procedure file must be attached by using the local file name MSTRDIR.

Control statements for the journal log file, the transaction recovery file, the restart identifier file, the quick recovery file, and the data base procedure library must not be present in the system procedure file; the files are automatically attached by CDCS using permanent file information in the master directory.

OPERATOR INTERFACE

The system operator can communicate with the CDCS system control point by using the central programmable displays, the K display under NOS, or the L display under NOS/BE. These displays are available through the Dynamic System Display (DSD), one of two routines that control the operator console. Throughout the following discussion of programmable displays, the reference K/L display refers to the K display for NOS and to the L display for NOS/BE.

Operator communication with CDCS via the programmable display is allowed any time after CDCS initialization is complete. While CDCS is operating at a control point, it can use the display to place information on the console screen and to receive information from the keyboard.

Programmable Display Usage

CDCS initialization must be complete before the operator can use the programmable display to communicate with CDCS. In addition, the K/L display can be used only after it has been assigned to CDCS.

The operator can call the display by typing the following:

K,cpnum. (for NOS)

L=ord. (for NOS/BE)

Each call must be terminated with a carriage return. The cpnum specification in the NOS call to the K display indicates the control point number of CDCS, which can be found by looking at the B display. The ord specification in the NOS/BE call to the L display indicates the JDT ordinal, which can be found by looking at the B display.

The operator can terminate the display by replacing the K or L display with the A display. To do this under NOS, the operator enters the left blank key followed by a call to the A display. To do this under NOS/BE, the operator enters the right blank key.

CDCS uses three types of displays in its interface with the system operator: a main display, a minidisplay, and an error display. The main display is used for all operator commands. The minidisplay is viewable only for the moment between the time the K/L display is assigned to CDCS and the time the main display is brought up. When CDCS determines that the K display (for NOS) or L display (for NOS/BE) has been assigned to it, CDCS makes sure it is rolled in, then brings up the main display. (CDCS might roll out all but 1000 octal words of its field length if no user requests are active.) During this short interval, the minidisplay places a message on the console screen requesting the operator to wait a moment while CDCS is preparing its display.

The error display is used to communicate to the operator that the current operator command cannot be processed at this time due to memory usage by CDCS. The error display is viewable only when the maximum field length assigned to CDCS at initialization is insufficient for both the current CDCS user activity and for the operator command just entered. When CDCS activity subsides, the operator command can be retried.

The main display is the major means of communication between CDCS and the operator. When the K/L display is first brought up, the main display lists the commands available to the operator for use with CDCS. A list of the options available with the STATUS command can be viewed by entering the . (period) key. When another . key is entered, the display cycles back to the command list.

Use of the K/L display allows the operator to view up to 43 lines with 64 characters per line of displayed information. The main display prepared by CDCS always consists of the following fields of information:

CDCS version number

CDCS level number

CDCS build date in the form yydd

SCP name

SCP id for NOS/BE, or SCP priority for NOS

CDCS status: UP, IDLE, DOWN, or IDLE/TERM

User count since initialization

Active user count

Active schema count

Active area count, including areas used implicitly due to constraints

Most recent operator command entered

Most of the information displayed at the console remains the same each time the display is refreshed. This information represents a snapshot of the CDCS environment at the time the last command was processed. Five of the fields, however, are dynamically updated:

CDCS status

User count since initialization

Active user count

Active schema count

Active area count

Some commands entered by the operator might require CDCS to display more lines of information than can fit on the console screen at one time. In this case, CDCS displays as many lines as possible and prompts the operator that the + and - keys can be used to reach lines that are not currently displayed.

Operator Commands

The operator commands to be used with CDCS are as follows:

DOWN
IDLE
RETAIN
RETURN
STATUS
TERM
UP

The format and effect of each command are described in the following paragraphs. In general, a period can be used optionally to terminate a command. Within each command, spaces and commas are used equivalently as separators. Multiple occurrences of a space or a comma are treated as a single occurrence.

For some commands, the operator must consider the operating system limitations concerning the maximum number of characters that can be entered in a command: NOS/BE limits the input to a maximum of 47 typed characters; NOS allows 50 typed characters. In the commands that require the specification of a schema or area, specification of the schema and area names can cause the command to exceed input limit; however, identification codes are available to shorten a command. The schema can be specified by either the schema name or schema id; the area can be specified by either the area name or area id. The schema id and area id are 1- to 4-digit identifications assigned by the DBMSTRD utility and listed on the master directory contents report (the second section, which is an optional section of the DBMSTRD output). The operator must use the identifications instead of the names if the length of the names make the length of a command exceed the operating system limit. In the DOWN, IDLE, STATUS, and UP commands, the schema and area ids can be used. In the RETAIN and RETURN commands, the schema id can be used. In the RETAIN and RETURN commands, however, a subschema must be specified by the subschema name.

NOTE

The limit on input from the operator console of NOS/BE limits the size of the subschema name that can be specified in a RETAIN or RETURN command. If the schema is specified by a 4-digit schema id, the subschema name is limited to 29 characters for NOS/BE. A 30-character subschema name cannot be used.

Commands that affect the running status of CDCS are shown in the system dayfile and the system control point dayfile.

DOWN Command

The DOWN command specifies that usage of CDCS, of a particular schema, or of a particular area is to be shut down and that all active jobs using CDCS, the schema, or area are to be terminated. The format of the DOWN command is shown in figure 10-5. Options used with the parameters of the DOWN command are defined in table 10-1.

When DOWN,CDCS is specified, all programs currently using CDCS are aborted. Any program attempting an invoke call to CDCS after this command is issued is also aborted. The effect of this command can be removed for all programs except those that have already been aborted by specifying an UP,CDCS command. Each occurrence of a DOWN,CDCS command is echoed in the system dayfile and in the system control point dayfile.

When a schema name or schema id is specified in the DOWN command, all programs currently using the indicated schema are aborted. Any program attempting an invoke call using this schema after the command is issued is also aborted. The specified schema is given a down status; all areas of the schema are returned so that they can be attached for recovery purposes. When a schema is marked as down, this status is retained by CDCS during the time CDCS remains active, unless an UP command for the schema is issued to bring the schema up again. All knowledge of a schema with a down status is lost when CDCS is terminated.

When an area name or area id is specified in the DOWN command, all programs currently using the indicated area are aborted. After the command is issued, any program attempting an invoke call to a subschema that uses this area is also aborted. The specified area is given a down status and returned so that the area can be attached for recovery purposes. When an area is marked down, the down status is retained by CDCS during the time CDCS remains active, unless an UP command for the area is issued to bring the area up again. All knowledge of an area with a down status is lost when CDCS is terminated.

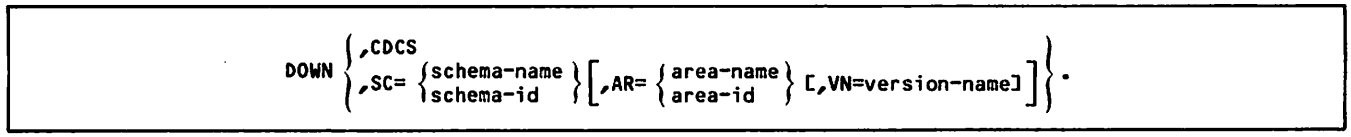


Figure 10-5. DOWN Command

TABLE 10-1. OPTIONS USED WITH PARAMETERS OF THE OPERATOR COMMANDS

Option	Definition
Area-id	Specifies the 1- to 4-digit area identification assigned by the DBMSTRD utility and listed in the master directory contents.
Area-name	Specifies the 1- to 30-character name of the area as recorded in the master directory.
Job-name	Specifies the unique 7-character job name assigned by NOS or NOS/BE and shown on the B display.
Schema-id	Specifies the 1- to 4-digit schema identification assigned by the DMSTRD utility and listed in the master directory contents.
Schema-name	Specifies the 1- to 30-character name of the schema, as recorded in the master directory.
Subschema-name	Specifies the 1- to 30-character name of the subschema, as recorded in the master directory.
Version-name	Specifies the 1- to 7-character version name, as recorded in the master directory. The version-name option can be used only in conjunction with either the area-name option or the area-id option.

When a version name is specified in the DOWN command, either the associated area name or the associated area id must also be specified. If MASTER is the only version defined in the master directory, the operator need not specify the version name. If the version name is required but is omitted, the following message is issued to the console:

VERSION NAME REQUIRED FOR AREA

The operator must then resubmit the command with the version name.

If an incorrect or unknown version name is entered, the following diagnostic message appears:

INVALID/UNKNOWN VERSION NAME

The operator must then resubmit the command with the proper version name.

Use of the DOWN command for a schema or area requires operator confirmation before the command can be executed. Verification of the command serves to prevent the unintentional termination of a program which could occur due to an error in the typing of the command. After the operator types in the DOWN command, the appropriate schema name and id, or area name and id, are displayed, along with a message requesting the operator to verify that the DOWN command for this schema or area should still be executed. The operator must respond appropriately. If the operator releases the K display (NOS) or L display (NOS/BE) without responding to the confirmation request, a flashing message is placed on the B display to remind the operator that an operator reply is required.

IDLE Command

The IDLE command specifies that usage of CDCS, of a particular schema, or of a particular area is to be shut down, but that all active jobs using CDCS or the schema or area are to be allowed to complete processing normally. The format of the IDLE command is shown in figure 10-6. Options used with parameters of the IDLE command are defined in table 10-1.

When IDLE,CDCS is specified, any program that subsequently attempts an invoke call to CDCS is aborted. All programs already using CDCS are permitted to continue processing until completion. The effects of an IDLE,CDCS command can be counteracted by specifying an UP,CDCS command. Each occurrence of an IDLE,CDCS command is echoed in the system dayfile and in the system control point dayfile.

When a schema name or schema id is specified in the IDLE command, any program that subsequently attempts an invoke call using the indicated schema is aborted. When the number of users of the schema becomes zero, the schema is given an idle status; all areas of the schema are returned so that they can be attached for recovery purposes. CDCS retains knowledge of an idle status for a schema while CDCS remains active; that knowledge is lost, however, when CDCS is terminated.

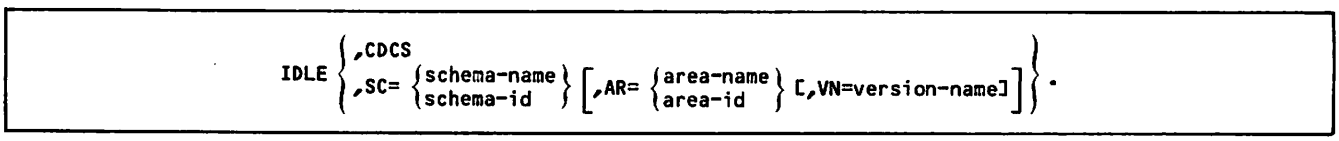


Figure 10-6. IDLE Command

When an area name or area id is specified in the IDLE command, any program that subsequently attempts an invoke call to a subschema that uses the indicated area is aborted. When the number of users of the area becomes zero, the area is given an idle status and returned so that the area can be attached for recovery purposes. CDCS retains knowledge of an idle status for an area while CDCS remains active; that knowledge is lost, however, when CDCS is terminated.

The requirements for usage of a version name with the IDLE command are the same as those described for the DOWN command. Refer to the description of the DOWN command for version name specification requirements.

As in the case of the DOWN command, operator confirmation of an IDLE command for a schema or area is required before the command can be executed. Refer to the description of the DOWN command for details on this requirement.

RETAIN Command

The RETAIN command causes CDCS to retain certain tables generated by CDCS and to keep data base files attached for CDCS use. Normally, when an application program invokes a subschema, CDCS generates tables and attaches the files required for executing the application program (unless the tables already exist and the files are already attached because a program executing concurrently has required them). Normally, when an application program terminates execution, CDCS returns the tables and files if they are not required by another program executing concurrently. When the RETAIN command is issued, the tables and files associated with the schema or subschema specified by the command are retained as long as CDCS remains active or until the schema or a retained file is given a down status or until a corresponding RETURN command is issued. The RETAIN command can specify that the retention of tables and files be on a schema or subschema basis. The format of the RETAIN command is shown in figure 10-7. Options used for the parameters of the RETAIN command are defined in table 10-1.

```
RETAIN,SC= { schema-name } [,SB=subschema-name] .
           { schema-id   }
```

Figure 10-7. RETAIN Command

The specification of just a schema in the RETAIN command causes the retention of all versions of the tables and files involved in CDCS processing using the specified schema. Therefore, all the tables and files involved in CDCS processing using any subschema of the specified schema are retained.

The specification of a schema and a subschema in the RETAIN command causes retention of the tables and files involved in CDCS processing using the specified subschema and version MASTER. If a RETAIN command specifying a schema and a subschema is entered when a RETAIN command specifying only that particular schema is in effect, the new command is ignored.

If the schema specified in a RETAIN command is downed, the RETAIN command is canceled, and the files associated with the schema are returned. If a file affected by a RETAIN command is downed, the file is returned. If the downed file is given an up status while CDCS remains active and the RETAIN command remains in effect, the file is retained after being accessed through CDCS.

When CDCS is initialized, retention of tables and files is not assumed; the RETAIN command must be specified to cause the tables and files to be retained.

The use of the RETAIN command can benefit CDCS processing; however, there are possible disadvantages in using the RETAIN command. If run-units continuously use a particular schema or if run-units using a particular subschema frequently invoke and terminate so that CDCS must repeatedly attach data base files, prepare them for processing, and return them, the specification of the RETAIN command improves the performance of CDCS. This situation normally occurs in the CDCS processing environment when the major use is by Query Update programs or by programs executing through the Transaction Facility (TAF). The improved performance in the execution of CDCS that results from the use of the RETAIN command is obtained at the expense of increased field length for CDCS. If a RETAIN command is in effect, files affected by the command could be available only to jobs executing with CDCS, depending on the permission with which CDCS has a file attached.

RETURN Command

The RETURN command cancels the retention of tables and files affected by a RETAIN command. A specification of this command causes the tables and files indicated by the command to be returned when the tables and files are not in use. The RETURN can specify the returning of tables and files on a schema or subschema basis. The format of the RETURN command is shown in figure 10-8. Options used for the parameters of the RETURN command are defined in table 10-1.

```
RETURN,SC= { schema-name } [,SB=subschema-name] .
           { schema-id   }
```

Figure 10-8. RETURN Command

If only a schema name is specified in the RETURN command, retention is canceled for all tables and files involved in CDCS processing using all versions of that schema. This command must be used to cancel a RETAIN command that specifies a schema only. Since this command specifies returning tables and files on a schema basis, the command can be used to cancel a RETAIN command that specifies both a schema and subschema. For example, the command RETURN,SC=AA must be used to cancel the command RETAIN,SC=AA. The command RETURN,SC=AA can also be used to cancel the command RETAIN,SC=AA, SB=ASUB; however, if other subschemas of schema AA were specified in other RETAIN commands, the tables and files used for processing with those subschemas are also returned.

If a subschema and a schema are specified in the RETURN command, retention is canceled for the tables and files involved in processing the specified subschema and version MASTER. This command can be used to cancel a corresponding RETAIN command that specifies the same schema and subschema. For example, the command RETURN,SC=BB,SB=BSUB can cancel the corresponding command RETAIN,SC=BB, SB=BSUB; the RETURN command is ignored if a RETAIN,SC=BB command is in effect.

STATUS Command

The STATUS command causes requested information concerning CDCS and the status of schemas, areas, and jobs to be displayed. In the command, STATUS can be specified by the abbreviation ST. The options in the command can be specified in any order. The format of the STATUS command is shown in figure 10-9. Options used for the parameters of the STATUS command are defined in table 10-1.

An error status can be indicated for a schema or area. An error status is effectively a down status. Error status for a schema usually indicates that a data base procedure file or log file cannot be attached or a fatal input/output error occurred on a log file. Error status on an area usually indicates that the permanent file associated with the area cannot be attached or that a fatal input/output error occurred on the file.

The options specified in the command determine the information that is displayed. The information displayed is divided into three categories as shown in table 10-2. The following list indicates the forms of the STATUS command and the categories of information that are displayed for each form. In the commands, the specification xxx indicates a schema name, schema id, area name, area id, version name or job name (whichever is appropriate in context).

STATUS,SC

Causes schema status information to be displayed for every schema currently in use.

STATUS,SC,ALL

Causes schema status information to be displayed for every schema for which information exists in the master directory.

STATUS,SC=xxx

Specifies a particular schema. This command causes schema status information to be displayed for the specified schema only.

STATUS,SC=xxx,AR=xxx[,VN=xxx]

Specifies a particular schema and area. Optionally, it also specifies a version. This command causes schema status information to be displayed for the specified schema and area status information to be displayed for the specified area.

STATUS,JOBS,SC=xxx

Specifies JOBS and a particular schema. This command causes schema status information to be displayed for the specified schema and job status information to be displayed for each job that is using CDCS and the specified schema.

STATUS,JOBS,SC=xxx,AR=xxx[,VN=xxx]

Specifies JOBS and a particular schema and area. Optionally, it also specifies a version. This command causes schema status information and area status information to be displayed for the specified schema and area, and job status information to be displayed for each job that is using CDCS and the specified area.

STATUS,JOBS

Causes job status information to be displayed for each job that is currently using CDCS.

STATUS,JOB=xxx

Specifies a particular job name. This command causes job status information to be displayed for the specified job.

TERM Command

The TERM command specifies that usage of CDCS is to be shut down, but that all active jobs using CDCS are to be allowed to complete processing normally. The format of the TERM command is as follows:

TERM.

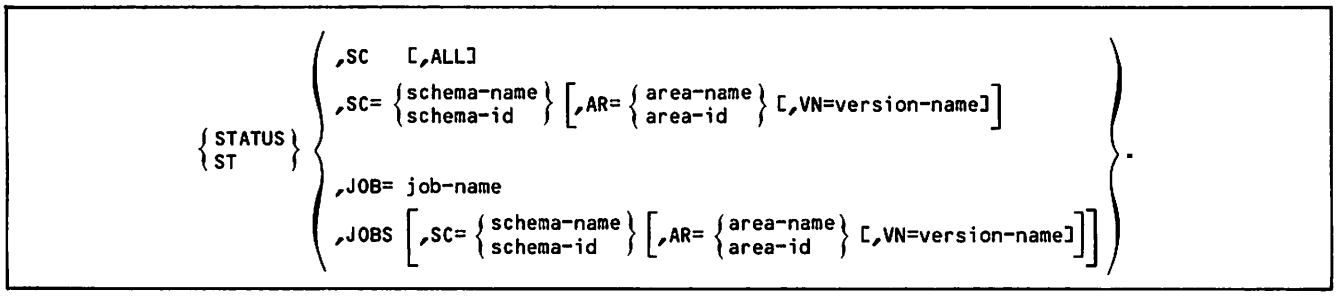


Figure 10-9. STATUS Command

TABLE 10-2. INFORMATION RETURNED BY THE STATUS COMMAND

Type of Information	Information Displayed
Schema Status Information	Schema id and name Schema status (DOWN, IDLE, UP, or ERROR) Number of schema users Current status of the log files indicated as follows: JLF journal log file QRF quick recovery file TRF transaction recovery file RIF restart identifier file
Area Status Information	Area id and name Area status (DOWN, IDLE, UP, or ERROR) Area activity (active or inactive) Number of users (indicated only if the area is active) Version name
Job Status Information	Job name Program name Task id for task executing through the Transaction Facility (TAF) Job activity (active, inactive, or rolled out) Schema id and name Subschema name Name of currently requested function Name of currently requested version

When the TERM command is specified, any program that subsequently attempts to issue an invoke call to CDCS is aborted. All programs already using CDCS are permitted to continue processing until completion. When the number of CDCS users reaches zero, CDCS is terminated. The effect of a TERM command can be counteracted by specifying an UP,CDCS command. Each occurrence of a TERM command is echoed in the system dayfile and in the system control point dayfile.

UP Command

The UP command specifies that usage of CDCS, of a particular schema, or of a particular area that has previously been shut down is to be restarted. The UP command can be used to allow access again to a schema or area that has had an error status. The format of the UP command is shown in figure 10-10. Options used for parameters in the UP command are defined in table 10-1.

An UP,CDCS command counteracts a DOWN,CDCS command, an IDLE,CDCS command, or a TERM command. All invoke calls to CDCS subsequent to the UP,CDCS command will be allowed. However, the down status of a schema or areas in a schema is not affected by an UP,CDCS command; all schemas or areas marked down remain down. All UP,CDCS commands are displayed in the system dayfile and in the system control point dayfile.

```
UP { ,CDCS
   { ,SC= { schema-name } [ ,AR= { area-name } [ ,VN=version-name] ] }
          { schema-id   }       { area-id   }
```

Figure 10-10. UP Command

An UP command that specifies a schema name or schema id removes the down status or error status of the indicated schema and its associated areas or terminates the effects of an IDLE command on the schema. All subsequent invoke calls using the schema will be allowed.

An UP command that specifies an area name or an area id removes the down status or error status of the indicated area or terminates the effects of an IDLE command on the area. All subsequent invoke calls to subschemas that use the area will be allowed if other required areas are usable.

The requirements for usage of a version name in the UP command are the same as those described for the DOWN command. Refer to the description of the DOWN command for version name specification requirements.

An UP command to counteract a DOWN command of a particular schema or area cannot be issued until the down status of that schema or area is complete.

Paging and Cycling Commands

The operator can use the + and - keys to display lines of information that are not currently displayed. (The programmable display affected is the K display for NOS or the L display for NOS/BE.) The . (period) key can be entered to cycle through the lists of available commands and command options. The keys and the actions they initiate are listed in table 10-3.

Sample Operator Interface

A series of operator commands illustrating the communication between the operator and CDCS is presented in figure 10-11. Use of a carriage return to terminate each command is assumed. After the K or L display has been called, a STATUS command is entered to determine which schemas are currently active. The operator should note that under NOS, the character K precedes only the first message typed to CDCS. DSD retains the K or L display for operator convenience. After examining the status of CDCS, the operator enters an IDLE command to terminate CDCS when all active jobs have completed processing. Finally, the operator replaces the K or L display with the A display. To do this under NOS, the operator enters the left blank key followed by a call to the A display. To do this under NOS/BE, the operator enters the right blank key.

NOS Example	NOS/BE Example
K,cpnum.	L=ord.
K.STATUS,SC.	STATUS,SC.
IDLE,CDCS.	IDLE,CDCS.
Left blank key	Right blank key
A.	

Figure 10-11. Sample Series of Operator Commands

TABLE 10-3. PAGING AND RECYCLING COMMANDS

Key	Action
+	Advances the K/L display to the next page of information. This key is valid only if the following message appears at the bottom of the currently displayed information: (ENTER + TO SEE NEXT PAGE). If this message does not appear, entering the + key has no effect. If the page currently displayed contains the last line of information, the message (END OF LIST) is shown at the bottom of the page.
-	Backs up the K/L display to the previous page of information. This key is valid only if the following message appears at the bottom of the currently displayed information (ENTER - TO SEE PREVIOUS PAGE). If this message does not appear, entering the - key has no effect. If the page currently displayed contains the first line of information, the message: (BEGINNING OF LIST) is displayed at the top of the page.
.	Cycles through command/option lists. When the K/L display is first assigned to CDCS, the main display shows a list of the commands available to the operator. Entering a period causes the display to cycle to a list of the options that can be specified in the STATUS command. Entering another period causes the display to cycle back to the command list.

CDCS TERMINATION

Two options are provided for termination of CDCS by the system operator. Either the special CDCS DOWN or TERM command or the operating system DROP command (STOP command in NOS) can be entered at the console to terminate CDCS. The DOWN and TERM commands, which have been described previously in this section, are preferred because they cause gradual shutdown; normally, DROP or STOP should not be used.

Under NOS/BE, the DROP command can be entered at the console to terminate CDCS. The DROP command has the following format:

```
n.DROP
```

The operator specifies the CDCS control point number for n.

Under NOS, the STOP command can be used to terminate CDCS. The STOP command is entered as follows:

```
n.STOP
```

The operator specifies the CDCS control point number for n.

With the DROP or STOP command, CDCS aborts. The operating system also aborts all user jobs. All active log files are returned.

Normally, this method of CDCS termination is not used because of the possible negative effect on currently active run-units and the implications for data base recovery. This method should be used only when CDCS processing must be terminated immediately.

The examples in this section show various phases of CDCS data base preparation for a sample manufacturing data base. In preparing the data base, the data administrator creates a schema, subschemas, and a master directory, and initializes log and recovery files. Sample jobs that the data administrator uses to accomplish these tasks are shown in this section.

The examples in this section reflect operation under the NOS operating system. Conversion to the NOS/BE operating system can be accomplished by making the following changes:

Substitute a NOS/BE ACCOUNT control statement for the NOS USER and CHARGE control statements.

Substitute the NOS/BE REQUEST and CATALOG control statements for the NOS DEFINE control statement.

Substitute the NOS/BE file identification parameter ID for the NOS file identification parameter UN.

Omit the NOS PERMIT control statements.

The examples are shown as batch jobs. The end-of-record that is required to separate control statements from source input in the job stream is shown in the examples by the symbol --EOR--.

SAMPLE SCHEMA

The first phase of data base preparation is schema creation. A sample schema is shown in figure 11-1. The schema is called MANUFACTURING-DB and includes seven data base areas: EMPLOYEE, JOBDETAIL, DEPARTMENTS, PROJECT, DEVELOPMENT-PRODUCTS, TESTS, and CDCSCAT.

The control statements required for the job include a FILE control statement for each area specified in the schema. The file organization specified for the area in the file control statement is eventually associated with the permanent file (data base file) through permanent file information in the master directory. The DEFINE control statement is also included in the job stream to establish the permanent file that contains the schema directory.

In the schema source input, access control locks are specified for each area. This schema illustrates using either one or two locks for an area. Two locks are specified for area EMPLOYEE, with one lock required for update and the other required for retrieval. A single lock applying for both update and retrieval is specified for area JOBDETAIL.

Area information is provided in the data control entry. The primary key for each area must be specified in the entry. EMP-ID of EMPREC (area EMPLOYEE) and a concatenated key CONCATKEY (area JOBDETAIL) are examples of primary keys specified for two areas. Alternate keys are optional and are specified for five areas.

Area CDCSCAT illustrates the description required for an area that can be used as a catalog file for a Query Update application using CDCS catalog mode. The primary key QUCAT-KEY is a 10-character, alphanumeric data item. The length of item QUCAT-ITEM corresponds to the Query Update's default transmission length (1030 characters). The collating sequence specified in the data control entry for the area is DISPLAY.

A constraint entry called PROJ-CONST associates the PROJECT and DEVELOPMENT-PRODUCTS areas through a common data item PROJECT-ID, with the record PROJREC being designated as the dominant record and DEVREC being designated as the dependent record in the constraint. CDCS enforces the constraint during update operations. A record in the PROJECT area cannot be deleted if the value of the PROJECT-ID also occurs in the DEVELOPMENT-PRODUCTS area. Similarly, the value of PROJECT-ID cannot be modified in a record in the PROJECT area if the modification references a value of PROJECT-ID occurring in both areas. Also, a record cannot be added to the DEVELOPMENT-PRODUCTS area if the value of the PROJECT-ID does not occur in the PROJECT area.

A relation entry called EMP-REL is used to join the areas EMPLOYEE and JOBDETAIL. The areas are joined by identical data items EMP-ID, which occurs in each area. This relation links employee information in both areas.

SAMPLE SUBSCHEMAS

The second phase of data base preparation is subschema creation. One sample of each of the following kinds of subschemas is shown in this section: COBOL, FORTRAN 5, and Query Update.

COBOL SUBSCHEMA

Sample COBOL subschema C5SS-PRODUCT-PERSONNEL is shown in figure 11-2. After compilation, the subschema directory resides on the subschema library C5SSLIB (the job creates the subschema library).

```
Job statement
USER control statement
CHARGE control statement
DEFINE,MANUFAC.
FILE(EMPLOYE,FO=IS,XN=IXEMP)
FILE(JOBDETA,FO=IS)
FILE(DEPARTM,FO=IS,XN=IXDEPT,RT=T)
FILE(PROJECT,FO=DA,XN=IXPROJ,HMB=7)
FILE(DEVELOP,FO=DA,XN=IXDEV,HMB=11)
FILE(TESTS,FO=IS,XN=IXTEST)
FILE(CDCSCAT,FO=IS)
DDL3,DS,SC=MANUFAC.
--EOR--

SCHEMA NAME IS MANUFACTURING-DB.

AREA IS EMPLOYEE
   ACCESS-CONTROL LOCK FOR RETRIEVAL IS "EMP-READ"
   ACCESS-CONTROL LOCK FOR UPDATE IS "EMP-WRITE"
           CALL OPENEMP BEFORE OPEN.

RECORD IS EMPREC WITHIN EMPLOYEE.
   01  EMP-ID                  PICTURE "X(8)".
   01  SALARY                  TYPE FIXED DECIMAL 8,2
                                   CALL EMPCHK BEFORE.
   01  EMP-LAST-NAME           PICTURE "A(20)".
   01  EMP-INITIALS            PICTURE "A(4)".
   01  DEPT                    PICTURE "X(4)".
   01  ADDRESS-NUMBERS         PICTURE "X(6)".
   01  ADDRESS-STREET          PICTURE "X(20)".
   01  ADDRESS-CITY            PICTURE "A(15)".
   01  ADDRESS-STATE-PROV      PICTURE "A(15)".
   01  POSTAL-CODE             PICTURE "X(10)".
   01  ADDRESS-COUNTRY         PICTURE "A(15)".
   01  PHONE-NO                PICTURE "9(10)".
   01  HIRE-DATE               PICTURE "9(6)".
   01  INSURANCE-NO            PICTURE "9(10)".
   01  NUM-DEPENDENTS          TYPE DECIMAL FIXED 2.
   01  JOB-CLASS               PICTURE "A"
                                   FOR ENCODING CALL DBPJC1
                                   FOR DECODING CALL DBPJC2.
   01  GRADE-LEVEL             PICTURE "9"
                                   CHECK VALUE 0 THRU 8.

/*    THE FOLLOWING ARE YEAR-TO-DATE TOTALS
*/
   01  GROSS                   TYPE DECIMAL FIXED 7,2.
   01  FED-TAX                 TYPE DECIMAL FIXED 7,2.
   01  STATE-TAX               TYPE DECIMAL FIXED 6,2.
   01  DISABILITY              TYPE DECIMAL FIXED 5,2.
   01  SS-INSURANCE            TYPE DECIMAL FIXED 4,4.
```

Figure 11-1. Sample Schema Named MANUFACTURING-DB (Sheet 1 of 4)

```
AREA IS JOBDETAIL
   ACCESS-CONTROL LOCK IS "ABCDEFZ".
RECORD IS JOBREC WITHIN JOBDETAIL
         CALL DELCHK BEFORE DELETE.
   01  EMP-ID                  PICTURE "X(8)".
   01  SEQ-NO                  PICTURE "X(4)".
   01  PRODUCT-ID              PICTURE "X(10)".
   01  SECURITY-CODE           PICTURE "X(2)".
   01  PROJECT-ID              PICTURE "X(10)".
   01  MONTHLY-COMPENSATION    OCCURS 12 TIMES.
       02  REG-HOURS           TYPE DECIMAL FIXED 5,2.
       02  REG-COMPENSATION    TYPE DECIMAL FIXED 6,2.
       02  OT-HOURS            TYPE DECIMAL FIXED 5,2.
       02  OT-COMPENSATION     TYPE DECIMAL FIXED 6,2.
   01  HOURS-YTD               TYPE DECIMAL FIXED 6,2
                                   VIRTUAL RESULT OF CALCHR.
   01  COMPENSATION-YTD        TYPE DECIMAL FIXED 10,2
                                   VIRTUAL RESULT OF CALCCP.
   01  START-DATE              PICTURE "9(6)".
   01  LOC-CODE                PICTURE "X(4)".
   01  ACTUAL-DATE             PICTURE "X(6)"
                                   ACTUAL RESULT OF DATESP.


AREA IS DEPARTMENTS
   ACCESS-CONTROL LOCK IS "VERY*PRIVATE".
RECORD IS DEPTREC WITHIN DEPARTMENTS.
   01  DEPT-NO                 PICTURE "X(4)".
   01  DEPT-NAME               PICTURE "X(20)".
   01  MGR-ID                  PICTURE "X(8)".
   01  MGR-NAME                PICTURE "X(20)".
   01  NUM-ITEM                PICTURE "9(3)"
                                  CHECK VALUE 5 THRU 25.
   01  ITEM                    OCCURS NUM-ITEM TIMES.
       02  LOC-CODE            PICTURE "X(4)".
       02  HEAD-COUNT          PICTURE "9(4)".
       02  EXPENSES-YTD        PICTURE "9(8)V99T".
       02  BUDGET              PICTURE "9(8)T".


AREA IS PROJECT
   ACCESS-CONTROL LOCK FOR RETRIEVAL IS "VERIFIED-INPUT"
   ACCESS-CONTROL LOCK FOR UPDATE IS "OKAYED-OUTPUT".
RECORD IS PROJREC WITHIN PROJECT.
   01  PROJECT-ID              PICTURE "X(10)".
   01  PROJ-DESCR              PICTURE "X(40)".
   01  BUDGET-TOTAL            PICTURE "9(9)V99".
   01  MONTHLY-BUDGET          PICTURE "9(7)V99" OCCURS 12 TIMES.
   01  SCHED-COMPLETE          PICTURE "X(10)".
   01  RESPONSIBILITY          PICTURE "X(8)".


AREA IS DEVELOPMENT-PRODUCTS
   ACCESS-CONTROL LOCK IS "ACCESS(/)OK".
RECORD IS DEVREC WITHIN DEVELOPMENT-PRODUCTS.
   01  PRODUCT-ID              PICTURE "X(10)".
   01  PRODUCT-DESCR           PICTURE "X(20)".
   01  CLASS                   PICTURE "9(2)"
                                  CHECK VALUE 0 THRU 99.
   01  PRICE                   TYPE DECIMAL FIXED 5,2.
   01  EVAL-ID                 TYPE CHARACTER 20 OCCURS 10 TIMES.
   01  PROJECT-ID              PICTURE "X(10)".
   01  NUM-TESTED              TYPE DECIMAL FIXED 4.
   01  CUM-TEST-AVERAGE        TYPE FLOAT.
   01  STATUS-CODE             PICTURE "A".
   01  SECURITY-CODE           PICTURE "X(2)".
   01  EST-COST                PICTURE "9(4)PPP".
   01  DEV-COST-YTD            TYPE DECIMAL FIXED 9,2.
```

Figure 11-1. Sample Schema Named MANUFACTURING-DB (Sheet 2 of 4)

```
AREA IS TESTS
   ACCESS-CONTROL LOCK FOR UPDATE IS "UP"
   ACCESS-CONTROL LOCK FOR RETRIEVAL IS "DOWN".
RECORD IS TESTREC WITHIN TESTS.
       TESTNO                  TYPE DECIMAL FIXED.
       TNAME                   TYPE CHARACTER 20.
       PRDCTNO                 TYPE CHARACTER 10.
       TESTER                  TYPE DECIMAL FIXED 8.
       TOTALCT                 TYPE DECIMAL FIXED 4.
       N                       TYPE DECIMAL FIXED 3
                                  CHECK VALUE 0 THRU 100.
       PASPROB                 TYPE FLOAT OCCURS 100 TIMES
                                  CHECK VALUE 0.0 THRU 1.0.


AREA NAME IS CDCSCAT
ACCESS-CONTROL LOCK IS "PERMISSION*GRANTED".
RECORD IS QUCATREC WITHIN CDCSCAT.
       QUCAT-KEY               PICTURE "X(10)".
       QUCAT-ITEM              PICTURE "X(1030)".

DATA CONTROL.

AREA NAME IS EMPLOYEE
       KEY IS EMP-ID OF EMPREC
           DUPLICATES ARE NOT ALLOWED
       KEY IS ALTERNATE DEPT
           DUPLICATES ARE INDEXED.

AREA NAME IS JOBDETAIL
       KEY ID IS CONCATKEY < EMP-ID OF JOBREC
                             SEQ-NO >
           DUPLICATES ARE NOT ALLOWED.

AREA NAME IS DEPARTMENTS
       KEY IS DEPT-NO
           DUPLICATES ARE NOT ALLOWED
       KEY IS ALTERNATE MGR-ID
           DUPLICATES ARE FIRST.

AREA NAME IS PROJECT
       KEY IS PROJECT-ID OF PROJREC
           DUPLICATES ARE NOT ALLOWED
       KEY IS ALTERNATE RESPONSIBILITY
           DUPLICATES ARE ALLOWED.

AREA NAME IS DEVELOPMENT-PRODUCTS
       KEY IS PRODUCT-ID OF DEVREC
           DUPLICATES ARE NOT ALLOWED
       KEY IS ALTERNATE PROJECT-ID OF DEVREC
           DUPLICATES ARE ALLOWED
       KEY IS ALTERNATE EVAL-ID
           DUPLICATES ARE INDEXED.

AREA NAME IS TESTS
       KEY IS TESTNO
           DUPLICATES ARE NOT ALLOWED
       KEY IS ALTERNATE TNAME
           DUPLICATES ARE INDEXED
       KEY IS ALTERNATE TESTER
           DUPLICATES ARE FIRST
       SEQUENCE IS ASCII.

AREA NAME IS CDCSCAT
       KEY IS QUCAT-KEY OF QUCATREC
           DUPLICATES ARE NOT ALLOWED
       SEQUENCE IS DISPLAY.
```

Figure 11-1. Sample Schema Named MANUFACTURING-DB (Sheet 3 of 4)

```
CONSTRAINT NAME IS MGR-CONST
          MGR-ID OF DEPTREC DEPENDS ON EMP-ID OF EMPREC.

CONSTRAINT NAME IS PROJ-CONST
          PROJECT-ID OF DEVREC DEPENDS ON PROJECT-ID OF PROJREC.


RELATION NAME IS EMP-REL
      JOIN WHERE EMP-ID OF JOBREC EQ EMP-ID OF EMPREC.

RELATION NAME IS TEST-REL
      JOIN WHERE PRDCTNO OF TESTREC EQ PRODUCT-ID OF DEVREC.

RELATION NAME IS DPD-REL
      JOIN WHERE MGR-ID OF DEPTREC EQ RESPONSIBILITY OF PROJREC
                 PROJECT-ID OF PROJREC EQ PROJECT-ID OF DEVREC.
```

Figure 11-1. Sample Schema Named MANUFACTURING-DB (Sheet 4 of 4)

```
Job statement
USER control statement
CHARGE control statement
DEFINE,C5SSLIB.
ATTACH,MANUFAC.
DDL3,C5,SB=C5SSLIB,SC=MANUFAC.
--EOR--

       TITLE DIVISION.
          SS  C5SS-PRODUCT-PERSONNEL WITHIN MANUFACTURING-DB.

       ALIAS DIVISION.
           AD  REALM JOBDETAIL BECOMES WORK-FILE.
           AD  RECORD JOBREC BECOMES WORK-REC.
           AD  RECORD EMPREC BECOMES EMP-REC.
           AD  DATA LOC-CODE BECOMES LOCATION.

       REALM DIVISION.
           RD EMPLOYEE, WORK-FILE.

       RECORD DIVISION.
       01  EMP-REC.
           03  EMP-ID                  PICTURE X(8).
           03  SALARY                  PICTURE 9(6)V99
                                           USAGE IS COMP-1.
           03  EMP-LAST-NAME           PICTURE A(20).
           03  EMP-INITIALS            PICTURE A(4).
           03  DEPT                    PICTURE X(4).
           03  MAILING-ADDRESS.
               05  ADDRESS-NUMBERS     PICTURE X(6).
               05  ADDRESS-STREET      PICTURE X(20).
               05  ADDRESS-CITY        PICTURE X(15).
               05  ADDRESS-STATE-PROV  PICTURE X(15).
               05  POSTAL-CODE         PICTURE X(10).
           03  PHONE-NO                PICTURE 9(10).
           03  PHONE                   REDEFINES PHONE-NO.
               05  AREA-CODE           PICTURE 999.
               05  PREFIX              PICTURE 999.
               05  DIGITS              PICTURE 9999.
           03  JOB-CLASS               PICTURE A.
           03  GRADE-LEVEL             PICTURE 9.
```

Figure 11-2. Sample COBOL Subschema Named C5SS-PRODUCT-PERSONNEL and Subschema Library Creation (Sheet 1 of 2)

```
01  WORK-REC.
    03  CONCATKEY.
        05  EMP-ID                 PICTURE X(8).
        05  SEQ-NO                 PICTURE X(4).
    03  PRODUCT-ID                 PICTURE X(10).
    03  SECURITY-CODE              PICTURE X(2).
    03  MONTHLY-COMPENSATION       USAGE IS COMP-1
                                       OCCURS 12 TIMES.
        05  REG-HOURS              PICTURE 9(3)V99.
        05  REG-COMPENSATION       PICTURE 9(4)V99.
        05  OT-HOURS               PICTURE 9(3)V99.
        05  OT-COMPENSATION        PICTURE 9(4)V99.
    03  HOURS-YTD                  PICTURE 9(4)V99
                                       USAGE IS COMP-1.
    03  COMPENSATION-YTD           PICTURE 9(8)V99
                                       USAGE IS COMP-1.
    03  LOCATION                   PICTURE X(4).

RELATION DIVISION.
    RN IS EMP-REL
        RESTRICT WORK-REC WHERE SECURITY-CODE GE "B1".
```

Figure 11-2. Sample COBOL Subschema Named C5SS-PRODUCT-PERSONNEL and Subschema Library Creation (Sheet 2 of 2)

The subschema includes realm definitions for schema areas EMPLOYEE and JOBDETAIL. Area JOBDETAI is assigned the alias realm name of WORK-FILE. This subschema is a basic example illustrating the manner in which items can be omitted, reordered, given alias names, and redefined (by the REDEFINES clause).

The subschema includes relation EMP-REL on which a restriction is placed. When records are read by use of the relation, only those records in which the value of data item SECURITY-CODE is greater than or equal to B1 are returned.

FORTRAN 5 SUBSCHEMA

Sample FORTRAN 5 subschema F5SS-PRODUCT-EVALUATION is shown in figure 11-3. After compilation, the subschema directory resides on the subschema library F5SSLIB (the job creates the subschema library).

The subschema includes definitions for two areas defined in the schema, areas DEVELOPMENT-PRODUCTS and TESTS. Area TESTS was included in the sample FORTRAN subschema shown previously. The two samples show differences in defining character data in FORTRAN 4 and FORTRAN 5 subschemas.

Many alias statements are included in this subschema. The area DEVELOPMENT-PRODUCTS is given an alias realm name PRODUCT-FILE. Many schema data names are given alias data item names because the data names in the schema exceed seven characters and include hyphens and, therefore, cannot be used in a FORTRAN program. The alias names specified in the ALIAS statements for data items are names acceptable to the FORTRAN compiler.

The subschema includes relation TEST-REL, which joins two realms. When an application program uses the relation in a DML READ statement, corresponding record occurrences from both realms are returned to the program. The restriction qualifies the relation and allows a record occurrence to be returned from realm PRODUCT-FILE only if the value of STATUS equals A.

```
Job statement
USER control statement
CHARGE control statement
DEFINE,F5SSLIB.
ATTACH,MANUFAC.
DDLF,F5,SB=F5SSLIB,SC=MANUFAC.
--EOR--

      SUBSCHEMA F5SS-PRODUCT-EVALUATION, SCHEMA=MANUFACTURING-DB

      ALIAS (REALM) PRODUCT-FILE = DEVELOPMENT-PRODUCTS
      ALIAS (RECORD) PRODREC = DEVREC
      ALIAS (ITEM) PRODUCT = PRODUCT-ID
      ALIAS (ITEM) EVALID = EVAL-ID
      ALIAS (ITEM) PROJECT = PROJECT-ID
      ALIAS (ITEM) NTESTED = NUM-TESTED
      ALIAS (ITEM) AVG = CUM-TEST-AVERAGE
      ALIAS (ITEM) STATUS = STATUS-CODE

      REALM PRODUCT-FILE, TESTS

      RECORD PRODREC
      CHARACTER *10 PRODUCT
      CHARACTER *2  CLASS
      CHARACTER *20 EVALID(10)
      CHARACTER *4  PROJECT
      CHARACTER *1  STATUS
      INTEGER PRICE, NTESTED
      REAL AVG

      RECORD TESTREC
      INTEGER TESTNO
      CHARACTER *20 TNAME
      CHARACTER *10 PRDCTNO
      INTEGER N, TOTALCT
      REAL PASPROB(100)

      RELATION TEST-REL
      RESTRICT PRODREC (STATUS .EQ. 'A' .OR. STATUS .EQ. 'R')

      END
```

Figure 11-3. Sample FORTRAN 5 Subschema Named F5SS-PRODUCT-EVALUATION and Subschema Library Creation

QUERY UPDATE SUBSCHEMA

Sample Query Update subschema QUPRODMGT is shown in figure 11-4. The control statements in the job stream show that the subschema directory is added to the subschema library QUSSLIB.

The subschema includes realm definitions for four areas defined in the schema: DEPARTMENTS (which is given the alias realm name DEPTAREA), PROJECTS, DEVELOPMENT-PRODUCTS (which is given the alias realm name PRODAREA), and CDCSCAT (a catalog file for use in CDCS catalog mode).

The subschema includes a three-area relation DPD-REL. The relation is used by Query Update (through CDCS) any time a query requires retrieval from the three areas joined in the relation; therefore, with a single query, the Query Update user can receive information about a department, the projects under control of the department, and the products involved in a particular project. A restriction is placed on the relation so that only records from realm PRODAREA in which the value of data name STATUS-CODE equals A are returned on a relational query.

```
Job statement
USER control statement
CHARGE control statement
ATTACH,QUSSLIB/M=W.
ATTACH,MANUFAC.
DDL3,QC,SB=QUSSLIB,SC=MANUFAC.
--EOR--

       TITLE DIVISION.
           SS  QUPRODMGT WITHIN MANUFACTURING-DB.

       ALIAS DIVISION.
           AD  REALM DEVELOPMENT-PRODUCTS BECOMES PRODAREA.
           AD  RECORD DEVREC BECOMES PRODREC.
           AD  DATA CUM-TEST-AVERAGE BECOMES CUMULATIVE-AVERAGE.
           AD  REALM DEPARTMENTS BECOMES DEPTAREA.

       REALM DIVISION.
           RD  DEPTAREA, PROJECT, PRODAREA, CDCSCAT.

       RECORD DIVISION.
       01  DEPTREC.
           03  DEPT-NO                 PICTURE X(4).
           03  DEPT-NAME               PICTURE X(20).
           03  MGR-ID                  PICTURE X(8).
           03  MGR-NAME                PICTURE X(20).
           03  NUM-ITEM                PICTURE 9(3).
           03  ITEM                    OCCURS 2 TO 25 TIMES
                                           DEPENDING ON NUM-ITEM.
               05  LOC-CODE            PICTURE X(4).
               05  HEAD-COUNT          PICTURE Z(4).
               05  EXPENSES-YTD        PICTURE Z(8).99.
               05  BUDGET              PICTURE Z(9).

       01  PROJREC.
           03  PROJECT-ID              PICTURE X(10).
           03  PROJ-DESCR              PICTURE X(40).
           03  BUDGET-TOTAL            PICTURE Z(9).99.
           03  RESPONSIBILITY          PICTURE X(8).

       01  PRODREC.
           03  PRODUCT-ID              PICTURE X(10).
           03  CLASS                   PICTURE Z9.
           03  PRICE                   PICTURE Z(5).99
                                           USAGE IS COMP-1.
           03  PROJECT-ID              PICTURE X(10).
           03  STATUS-CODE             PICTURE A.
           03  DEV-COST-YTD            PICTURE Z(8)9.99
                                           USAGE IS COMP-1.

       01  QUCATREC.
           03 QUCAT-KEY                PICTURE X(10).
           03 QUCAT-ITEM               PICTURE X(1030).

       RELATION DIVISION.
           RN IS DPD-REL
               RESTRICT PRODREC WHERE STATUS-CODE EQ "A".
```

Figure 11-4. Adding Sample Query Update Subschema Named QUPRODMGT to the Subschema Library

SAMPLE MASTER DIRECTORY CREATION RUN

The third phase of data base preparation is the master directory creation. A sample master directory creation run is shown in figure 11-5.

The control statements in the job stream show that the schema directory and subschema directories must be available to the DBMSTRD utility as it generates the master directory file. The master directory must be stored as a permanent file, MSTRDIR in this example.

The PERMIT control statement shown in the job stream applies to NOS only. This control statement makes the master directory file available with write (W) permission to CDCS or any job that executes under the user number DBCNTLX.

The sample master directory provides all the information necessary for CDCS to monitor processing of the manufacturing data base. Information specified in the master directory includes references to the schema and subschemas, specification of log and recovery files, specification of a data base procedure library, and permanent file information for areas. The schema entry in the master directory source input includes selection of a full set of log and recovery files and specification of job control information. Only one data base version (MASTER) is included in this creation run.

```
Job statement
USER control statement
CHARGE control statement
DEFINE,MSTRDIR.
PERMIT,MSTRDIR,DBCNTLX=W.
ATTACH,MANUFAC.
ATTACH,C5SSLIB.
ATTACH,F4SSLIB.
ATTACH,QUSSLIB.
DBMSTRD,NMD=MSTRDIR,LD.
--EOR--

SCHEMA NAME IS MANUFACTURING-DB
        FILE NAME IS MANUFAC

        PROCEDURE LIBRARY
            PFN IS "DB1PLIB" UN IS "CDCS23"
        TRANSACTION RECOVERY FILE
            PFN IS "DB1TRF" UN IS "CDCS23"
            UNIT LIMIT IS 50
            UPDATE LIMIT IS 15
        RESTART IDENTIFIER FILE
            PFN IS "DB1RIF" UN IS "CDCS23"
        JOURNAL LOG FILE
            PFN IS "DB1JLF" UN IS "CDCS23"
        QUICK RECOVERY FILE
            PFN IS "DB1QRF" UN IS "CDCS23"
        JOB CONTROL INFORMATION
            TAPE TYPE IS NT
                 DENSITY IS PE
            UN IS "CDCS23"
            CHARGE IS "1982CHG".

VERSION NAME IS MASTER

    AREA NAME IS EMPLOYEE
        PFN IS "MEMPL" UN IS "CDCS23"
            PW IS "OKCDCS2"
        LOG BEFORE IMAGE BLOCKS
            AFTER IMAGE RECORDS
        INDEX FILE ASSIGNED
            PFN "MXEMPL" UN IS "CDCS23".

    AREA NAME IS JOBDETAIL
        PFN IS "MJOBD" UN IS "CDCS23"
            PW IS "OKCDCS2"
        LOG BEFORE IMAGE BLOCKS
            AFTER IMAGE RECORDS.

        AREA NAME IS DEPARTMENTS
            PFN IS "MDEPT" UN IS "CDCS23"
            LOG BEFORE IMAGE BLOCKS
            INDEX FILE ASSIGNED
                PFN "MXDEPT" UN IS "CDCS23".

        AREA NAME IS PROJECT
            PFN IS "MPROJ" UN IS "CDCS23"
            LOG BEFORE IMAGE BLOCKS
                AFTER IMAGE RECORDS
            INDEX FILE ASSIGNED
                PFN "MXPROJ" UN IS "CDCS23".

        AREA NAME IS DEVELOPMENT-PRODUCTS
            PFN IS "MDEVE" UN IS "CDCS23"
            LOG BEFORE IMAGE BLOCKS
            INDEX FILE ASSIGNED
                PFN IS "MXDEVE" UN IS "CDCS23".

        AREA IS TESTS
            PFN IS "MTEST" UN IS "CDCS23"
            LOG BEFORE IMAGE BLOCKS
                BEFORE IMAGE RECORDS
                AFTER IMAGE RECORDS
            INDEX FILE ASSIGNED
                PFN "MXTEST" UN IS "CDCS23".

        AREA NAME IS CDCSCAT
            PFN IS "MQCAT" UN IS "CDCS23".


    SUBSCHEMA NAME IS C5SS-PRODUCT-PERSONNEL
            FILE NAME IS C5SSLIB.
    SUBSCHEMA NAME IS C5SS-PRODUCT-MANAGEMENT
            FILE NAME IS C5SSLIB.
    SUBSCHEMA NAME IS F4SS-TESTING
            FILE NAME IS F4SSLIB.
    SUBSCHEMA NAME IS F4SS-PRODUCT-MANAGEMENT
            FILE NAME IS F4SSLIB.
    SUBSCHEMA NAME IS QUCREA6
            FILE NAME IS QUSSLIB.
    SUBSCHEMA NAME IS QUPRODMGT
            FILE NAME IS QUSSLIB.
```

Figure 11-5. Sample Master Directory Creation Run

SAMPLE MASTER DIRECTORY MODIFICATION RUN

A possible fourth phase in data base preparation is modification of a master directory. A sample master directory modification run is shown in figure 11-6.

The control statements in the modification run include an ATTACH of the schema directory file and an ATTACH of the subschema library for a subschema being added. The old master directory, residing on file MSTRDIR, is also attached. The DBMSTRD control statement specifies the OMD parameter (which indicates the old master directory file) and the NMD parameter (which indicates the new master directory file). In a modification run, no changes are made to the old master directory file; a new, modified master directory is generated and written to a new master directory file.

This modification run changes the master directory as follows:

Changes permanent file information for the procedure library file.

Changes permanent file information and logging specifications for area TESTS (version MASTER).

Changes logging specifications for area CDCSCAT (version MASTER); however, no permanent file information is changed.

Adds data base version TESTVRS.

Deletes a subschema.

Adds two subschemas.

Data base version TESTVERS includes information for each area defined in the schema. Permanent files specified for this version can be unique or can be shared with version MASTER, depending on requirements for constraints (refer to section 6 for these requirements). Logging options were not selected because this is a test version, but they could have been selected.

SAMPLE DBREC UTILITY RUN

A final phase of data base preparation is the initialization of log and recovery files by the DBREC utility. A sample DBREC allocation run is shown in figure 11-7. The files allocated by DBREC have been specified for the schema in the master directory and must be allocated and available to CDCS before the schema can be used.

```
Job statement
USER control statement
CHARGE control statement
DEFINE,NEWMSTR.
PERMIT,NEWMSTR,DBCNTLX=W.
ATTACH,MSTRDIR.
ATTACH,MANUFAC.
ATTACH,F5SSLIB.
DBMSTRD,OMD=MSTRDIR,NMD=NEWMSTR,LD.
PURGE,MSTRDIR.
--EOR--

MODIFY SCHEMA IS MANUFACTURING-DB
        FILE NAME IS MANUFAC.

CHANGE PROCEDURE LIBRARY
    PFN IS "NEWPLIB" UN IS "CDCS23"
        PW IS "DMS170".

CHANGE AREA NAME IS TESTS
    VERSION NAME IS MASTER
    PFN IS "NEWFIL" UN IS "CDCS23"
        PW IS "DMS170"
    LOG BEFORE IMAGE BLOCKS
    INDEX PFN IS "NEWFILX" UN IS "CDCS23"
        PW IS "DMS170".

CHANGE AREA CDCSCAT
    LOG BEFORE IMAGE BLOCKS.

ADD VERSION NAME IS TESTVRS

    AREA NAME IS EMPLOYEE
        SAME AS MASTER.

    AREA NAME IS JOBDETAIL
        PFN IS "TJOBD" UN IS "CDCS23".

    AREA NAME IS DEPARTMENTS
        SAME AS MASTER.

    AREA NAME IS PROJECT
        PFN IS "TPROJ" UN IS "CDCS23"
        INDEX FILE ASSIGNED
            PFN "TXPROJ" UN IS "CDCS23".

    AREA IS DEVELOPMENT-PRODUCTS
        PFN IS "TDEVE" UN IS "CDCS23"
        INDEX FILE ASSIGNED
            PFN "TXDEVE" UN IS "CDCS23".

    AREA IS TESTS
        PFN IS "TTEST" UN IS "CDCS23"
        INDEX FILE ASSIGNED
            PFN "TXTEST" UN IS "CDCS23".

    AREA NAME IS CDCSCAT
        PFN IS "TQCAT" UN IS "CDCS23".

DELETE SUBSCHEMA NAME IS F4SS-PRODUCT-MANAGEMENT.

ADD SUBSCHEMA NAME IS F5SS-PRODUCT-MANAGEMENT
        FILE NAME IS F5SSLIB.

ADD SUBSCHEMA NAME IS F5SS-PRODUCT-EVALUATION
        FILE NAME IS F5SSLIB.

END MODIFICATIONS.
```

Figure 11-6. Sample Master Directory Modification Run

Job for DBREC Utility Run

```
Job statement
USER control statement
CHARGE control statement
DEFINE,DB1QRF,DB1PLIB.
DEFINE,DB1TRF1,DB1RIF.
DEFINE,DB1JLF1,DB1JLF2.
RETURN,DB1QRF,DB1PLIB.
RETURN,DB1TRF1,DB1RIF.
RETURN,DB1JLF1,DB1JLF2.
PERMIT,DB1QRF,DBCNTLX=W.
PERMIT,DB1PLIB,DBCNTLX=W.
PERMIT,DB1TRF1,DBCNTLX=W.
PERMIT,DB1RIF,DBCNTLX=W.
PERMIT,DB1TRF1,DBCNTLX=W.
PERMIT,DB1JLF1,DBCNTLX=W.
PERMIT,DB1JLF2,DBCNTLX=W.
ATTACH,MSTRDIR.
DBREC.
--EOR--

    SCHEMA NAME IS MANUFACTURING-DB
    ALLOCATE
      QUICK RECOVERY FILE IS DB1QRF SIZE IS 500 PRUS
      JOURNAL LOG FILE SIZE IS 10000 PRUS
      TRANSACTION RECOVERY FILE IS DB1TRF1
      RESTART IDENTIFIER FILE IS DB1RIF
```

DBREC Utility Output

```
        DBREC SOURCE LISTING                                    DBREC V2.3(82061)


                    1             SCHEMA NAME IS MANUFACTURING-DB
                    2             ALLOCATE
                    3               QUICK RECOVERY FILE IS DB1QRF SIZE IS 500 PRUS
                    4               JOURNAL LOG FILE SIZE IS 10000 PRUS
                    5               TRANSACTION RECOVERY FILE IS DB1TRF1
                    6               RESTART IDENTIFIER FILE IS DB1RIF
-   DBREC SYNTAX VALIDATION COMPLETE       0 ERRORS

         DBREC EXECUTION LISTING                                DBREC V2.3(82061)

 BEGIN PROCESSING DIRECTIVES FOR SCHEMA MANUFACTURING-DB.
        JOURNAL LOG FILE DB1JLF1 HAS BEEN ALLOCATED.
        JOURNAL LOG FILE DB1JLF2 HAS BEEN ALLOCATED.
        TRANSACTION RECOVERY FILE DB1TRF1 HAS BEEN ALLOCATED.
        QUICK RECOVERY FILE DB1QRF HAS BEEN ALLOCATED.
        RESTART IDENTIFIER FILE DB1RIF HAS BEEN ALLOCATED.
    ALLOCATE DIRECTIVE EXECUTION COMPLETE.
-   DBREC EXECUTION COMPLETE     0 ERRORS

                                 0 WARNINGS

    REQUIRED   53000B WORDS SCM
    RUN TIME     0.188 SECONDS
```

Figure 11-7. Sample DBREC Utility Run

The control statements shown in the job stream for the DBREC utility run indicate that the master directory file must be attached and the log and recovery files must be established as permanent files before the utility is executed. DBREC uses permanent file information provided in the master directory to process the files.

The figure also shows the output provided by the DBREC utility. The output indicates the permanent file names of the files that were allocated. The output also shows that two journal log files were allocated (the default allocation option for the journal log file processing).

STANDARD CHARACTER SETS

A

Control Data operating systems offer the following variations of a basic character set:

CDC 64-character set

CDC 63-character set

ASCII 64-character set

ASCII 63-character set

The set in use at a particular installation was specified when the operating system was installed.

Depending on another installation option, the system assumes an input deck has been punched either in 026 or in 029 mode (regardless of the character set in use). Under NOS/BE, the alternate mode can be specified by a 26 or 29 punched in columns 79 and 80 of the job statement or any 7/8/9 card. The specified mode remains in effect through the end of the job unless it is reset by specification of the alternate mode on a subsequent 7/8/9 card.

Under NOS, the alternate mode can be specified also by a 26 or 29 punched in columns 79 and 80 of any 6/7/9 card, as described above for a 7/8/9 card. In addition, 026 mode can be specified by a card with 5/7/9 multipunched in column 1, and 029 mode can be specified by a card with 5/7/9 multipunched in column 1 and a 9 punched in column 2.

Graphic character representation appearing at a terminal or printer depends on the installation character set and the terminal type. Characters shown in the CDC Graphic column of the standard character set table (table A-1) are applicable to BCD terminals; ASCII graphic characters are applicable to ASCII-CRT and ASCII-TTY terminals.

Standard collating sequences for the two printer character sets are shown in tables A-2 and A-3.

TABLE A-1. STANDARD CHARACTER SETS

Display Code (octal)	CDC			ASCII		
	Graphic	Hollerith Punch (026)	External BCD Code	Graphic Subset	Punch (029)	Code (octal)
00†	: (colon)††	8-2	00	: (colon) ††	8-2	072
01	A	12-1	61	A	12-1	101
02	B	12-2	62	B	12-2	102
03	C	12-3	63	C	12-3	103
04	D	12-4	64	D	12-4	104
05	E	12-5	65	E	12-5	105
06	F	12-6	66	F	12-6	106
07	G	12-7	67	G	12-7	107
10	H	12-8	70	H	12-8	110
11	I	12-9	71	I	12-9	111
12	J	11-1	41	J	11-1	112
13	K	11-2	42	K	11-2	113
14	L	11-3	43	L	11-3	114
15	M	11-4	44	M	11-4	115
16	N	11-5	45	N	11-5	116
17	O	11-6	46	O	11-6	117
20	P	11-7	47	P	11-7	120
21	Q	11-8	50	Q	11-8	121
22	R	11-9	51	R	11-9	122
23	S	0-2	22	S	0-2	123
24	T	0-3	23	T	0-3	124
25	U	0-4	24	U	0-4	125
26	V	0-5	25	V	0-5	126
27	W	0-6	26	W	0-6	127
30	X	0-7	27	X	0-7	130
31	Y	0-8	30	Y	0-8	131
32	Z	0-9	31	Z	0-9	132
33	0	0	12	0	0	060
34	1	1	01	1	1	061
35	2	2	02	2	2	062
36	3	3	03	3	3	063
37	4	4	04	4	4	064
40	5	5	05	5	5	065
41	6	6	06	6	6	066
42	7	7	07	7	7	067
43	8	8	10	8	8	070
44	9	9	11	9	9	071
45	+	12	60	+	12-8-6	053
46	-	11	40	-	11	055
47	*	11-8-4	54	*	11-8-4	052
50	/	0-1	21	/	0-1	057
51	(	0-8-4	34	(	12-8-5	050
52	)	12-8-4	74	)	11-8-5	051
53	$	11-8-3	53	$	11-8-3	044
54	=	8-3	13	=	8-6	075
55	blank	no punch	20	blank	no punch	040
56	, (comma)	0-8-3	33	, (comma)	0-8-3	054
57	. (period)	12-8-3	73	. (period)	12-8-3	056
60	≡	0-8-6	36	#	8-3	043
61	[	8-7	17	[	12-8-2	133
62	]	0-8-2	32	]	11-8-2	135
63	% ††	8-6	16	% ††	0-8-4	045
64	≠	8-4	14	" (quote)	8-7	042
65	←	0-8-5	35	_ (underline)	0-8-5	137
66	∨	11-0	52	!	12-8-7	041
67	∧	0-8-7	37	&	12	046
70	↑	11-8-5	55	' (apostrophe)	8-5	047
71	↓	11-8-6	56	?	0-8-7	077
72	<	12-0	72	<	12-8-4	074
73	>	11-8-7	57	>	0-8-6	076
74	≤	8-5	15	@	8-4	100
75	≥	12-8-5	75	\	0-8-2	134
76	¬	12-8-6	76	^ (circumflex)	11-8-7	136
77	; (semicolon)	12-8-7	77	; (semicolon)	11-8-6	073

†Twelve zero bits at the end of a 60-bit word in a zero byte record are an end of record mark rather than two colons.

††In installations using a 63-graphic set, display code 00 has no associated graphic or card code; display code 63 is the colon (8-2 punch). The % graphic and related card codes do not exist and translations yield a blank (55_8).

TABLE A-2. CDC CHARACTER SET COLLATING SEQUENCE

Collating Sequence Decimal	Collating Sequence Octal	CDC Graphic	Display Code	External BCD	Collating Sequence Decimal	Collating Sequence Octal	CDC Graphic	Display Code	External BCD
00	00	blank	55	20	32	40	H	10	70
01	01	≤	74	15	33	41	I	11	71
02	02	%	63 †	16 †	34	42	∨	66	52
03	03	[	61	17	35	43	J	12	41
04	04	→	65	35	36	44	K	13	42
05	05	≡	60	36	37	45	L	14	43
06	06	∧	67	37	38	46	M	15	44
07	07	↑	70	55	39	47	N	16	45
08	10	↓	71	56	40	50	O	17	46
09	11	>	73	57	41	51	P	20	47
10	12	≥	75	75	42	52	Q	21	50
11	13	¬	76	76	43	53	R	22	51
12	14	.	57	73	44	54	]	62	32
13	15	)	52	74	45	55	S	23	22
14	16	;	77	77	46	56	T	24	23
15	17	+	45	60	47	57	U	25	24
16	20	$	53	53	48	60	V	26	25
17	21	*	47	54	49	61	W	27	26
18	22	-	46	40	50	62	X	30	27
19	23	/	50	21	51	63	Y	31	30
20	24	,	56	33	52	64	Z	32	31
21	25	(	51	34	53	65	:	00 †	none†
22	26	=	54	13	54	66	0	33	12
23	27	≠	64	14	55	67	1	34	01
24	30	<	72	72	56	70	2	35	02
25	31	A	01	61	57	71	3	36	03
26	32	B	02	62	58	72	4	37	04
27	33	C	03	63	59	73	5	40	05
28	34	D	04	64	60	74	6	41	06
29	35	E	05	65	61	75	7	42	07
30	36	F	06	66	62	76	8	43	10
31	37	G	07	67	63	77	9	44	11

†In installations using the 63-graphic set, the % graphic does not exist. The : graphic is display code 63, External BCD code 16.

TABLE A-3. ASCII CHARACTER SET COLLATING SEQUENCE

Collating Sequence Decimal	Collating Sequence Octal	ASCII Graphic Subset	Display Code	ASCII Code	Collating Sequence Decimal	Collating Sequence Octal	ASCII Graphic Subset	Display Code	ASCII Code
00	00	blank	55	20	32	40	@	74	40
01	01	!	66	21	33	41	A	01	41
02	02	"	64	22	34	42	B	02	42
03	03	#	60	23	35	43	C	03	43
04	04	$	53	24	36	44	D	04	44
05	05	%	63†	25	37	45	E	05	45
06	06	&	67	26	38	46	F	06	46
07	07	'	70	27	39	47	G	07	47
08	10	(	51	28	40	50	H	10	48
09	11	)	52	29	41	51	I	11	49
10	12	*	47	2A	42	52	J	12	4A
11	13	+	45	2B	43	53	K	13	4B
12	14	,	56	2C	44	54	L	14	4C
13	15	-	46	2D	45	55	M	15	4D
14	16	.	57	2E	46	56	N	16	4E
15	17	/	50	2F	47	57	O	17	4F
16	20	0	33	30	48	60	P	20	50
17	21	1	34	31	49	61	Q	21	51
18	22	2	35	32	50	62	R	22	52
19	23	3	36	33	51	63	S	23	53
20	24	4	37	34	52	64	T	24	54
21	25	5	40	35	53	65	U	25	55
22	26	6	41	36	54	66	V	26	56
23	27	7	42	37	55	67	W	27	57
24	30	8	43	38	56	70	X	30	58
25	31	9	44	39	57	71	Y	31	59
26	32	:	00†	3A	58	72	Z	32	5A
27	33	;	77	3B	59	73	[	61	5B
28	34	<	72	3C	60	74	\	75	5C
29	35	=	54	3D	61	75	]	62	5D
30	36	>	73	3E	62	76	^	76	5E
31	37	?	71	3F	63	77	_	65	5F

†In installations using a 63-graphic set, the % graphic does not exist. The : graphic is display code 63.

DIAGNOSTICS B

Diagnostic messages issued by CYBER Database Control System (CDCS), the data base utilities, and the Data Description Language (DDL) compilers are contained in this appendix.

Messages issued by CDCS include diagnostics for schema and subschema errors, diagnostics for mapping errors, diagnostics for relation processing errors or conditions, and diagnostics for data base procedure processing errors. Messages issued by the data base utilities include diagnostics issued for syntax errors and diagnostics issued for execution-time errors. Messages issued by the DDL compiler include diagnostics issued during compilation of a schema or a subschema, diagnostics issued during execution of the EXHIBIT utility, and diagnostics issued during subschema library maintenence operations.

All messages in this appendix appear in the following order:

Messages beginning with an alphabetic character appear first.

Messages beginning with a number appear next.

Messages beginning with a variable are listed last.

Within a message, variables appear in lowercase to signify that the field is replaced by applicable text when the message is issued.

The diagnostics appear in this appendix in tables listed as follows:

B-1 CDCS Diagnostics

B-2 DBMSTRD Master Directory Diagnostics

B-3 DBQRFA and DBQRFI Diagnostics

B-4 DBRCN and DBRST Diagnostics

B-5 DBREC Diagnostics

B-6 DDL Schema Compilation and Exhibit Diagnostics

B-7 DDL Subschema Compilation and Library Maintenance Diagnostics for COBOL and Query Update

B-8 DDLF Subschema Compilation and Library Maintenance Diagnostics for FORTRAN

CDCS DIAGNOSTICS

All diagnostic messages that can be issued by CDCS are listed in table B-1. The general significance of the message and the action to be taken by the user accompany each message listed in the table. A destination column designates the dayfile or output file to which the message is written.

The severity level of the diagnostic messages listed in table B-1 can be of several types. The types of messages and their meanings are as follows:

C Critical error. The system control point is aborted and all user control point jobs are aborted.

F Fatal error. The run-unit is aborted, with the execption of a FORTRAN or COBOL program executing with the immediate return feature enabled, an application program executing through the Transaction Facility (TAF), or a Query Update application executing in interactive mode. (In these cases the program is terminated from CDCS processing. Control is returned to the program if the immediate return feature is enabled. Control is returned to TAF if the program is processing through TAF or to Query Update if the application is executing in interactive mode.)

N Nonfatal error. Processing continues.

I Informative message. Processing continues.

The CDCS diagnostic messages listed in table B-1 are categorized several ways. The following paragraphs describe the messages in categories which describe the destination of the message and the type of message sent to that destination.

The first category of CDCS diagnostics listed in table B-1 includes the unnumbered messages that are sent to the CDCS dayfile. These messages are of the following types:

C Critical error

N Nonfatal error

I Informative message

The second category of diagnostics includes the unnumbered messages issued by CDCS when it is operating as the CDCS Batch Test Facility (CDCSBTF). These messages are sent to the CDCSBTF control point dayfile, which is the user control point dayfile since CDCSBTF executes at the user control point. These messages are of the following types:

F Fatal error

I Informative message

The third category of CDCS messages includes the unnumbered messages that are sent to the user control point dayfile and the CDCS output file. These messages can be of the following types:

F Fatal error

I Informative message

The fourth category of CDCS messages include the unnumered messages that are sent to both the CDCS control point dayfile and the CDCS output file. These messages are usually issued during system recovery processing. They can also be issued at other times (for example, during invoke processing or when a transaction is reversed). These messages can be of the following types:

N Nonfatal error

I Informative message

The fifth category of CDCS messages includes the unnumbered messages that are sent to the user control point dayfile only. These messages can be of the following types:

F Fatal error

N Nonfatal error

I Informative message

The sixth category of messages include the unnumbered messages that are sent to the CDCS output file only. These error messages are usually accompanied by a fatal error message sent to the CDCS system control point dayfile. These messages can be of the following type:

I Informative message

The seventh category of CDCS message includes the messages that occur when the system operator communicates with CDCS through the K (NOS) or the L (NOS/BE) L display. These informative messages that indicate activities triggered by operator commands are sent to the CDCS system control point dayfile. In some cases, the message returned is an echo of the command entered by the operator. These messages are of the following type:

I Informative message

The last category of CDCS diagnostic messages includes the CDCS numbered execution-time messages for user errors. Each octally numbered message is accompanied by a conversion from the octal value of the CDCS error code to the equivalent decimal value. The decimal conversion of the error code (found in the column headed Dec. E. C. of table B-1) can be of assistance to a COBOL programmer using the data base status block to obtain error and status information because codes returned in the status block of a COBOL program must be referenced by their decimal values. The error and status codes returned in the status block of a FORTRAN program can be referenced by either their decimal or octal values. The octal value of codes are returned by Query Update in a diagnostic message. The destination of each message depends on the type. These messages can be of the following types:

F Fatal error. The message is sent to the CDCS output file, the data base status block if one exists, and the user control point dayfile. A message is sent to the ZZZZZEG file if the message is associated with a realm (area).

N Nonfatal error. The message is sent to the CDCS output file and the data base status block if one exists. The message is sent to the ZZZZZEG file if the message is associated with a realm (area).

T Trivial error. The message is sent to the data base status block if one exists. The message is also sent to the ZZZZZEG file if the message is associated with a realm (area).

I Informative message. The message is sent to the CDCS output file. The message is sent to the ZZZZZEG file if the message is associated with a realm (area).

Messages written by CDCS to the user's ZZZZZEG error file are known as data manager messages. When the ZZZZZEG file is processed, each data manager error message is preceded by the following information:

DM ERROR ec ON LFN fn EXIT ADDRESS ad

The fields ec, fn, and ad are replaced by applicable text to designate error code, schema file name, and address, respectively.

TABLE B-1. CDCS DIAGNOSTICS

Type	Dec. E.C.	Message	Significance	Action	Destination
N		ALL AREAS WERE DOWN, SCHEMA DOWN	None of the areas in the schema were in up status at the completion of system recovery. The schema is set to down status.	Correct the errors causing the areas to be down. Manually recover the areas in the schema.	CDCS output file CDCS system control point dayfile
F		AREA id VERSION vn NOT UP. SCHEMA sn	The area id of version vn (version MASTER if vn does not appear) was not in an up, idling, or downing status during processing for system recovery or CDCS transaction reversal. Schema sn is printed out for nonsystem recovery only.	If this error occurred during system recovery, manually recover the area. If transaction processing was in effect, manually reverse the uncommitted updates. Place the area in up status.	CDCS output file CDCS system control point dayfile

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
I		AREA RECOVERY IS COMPLETE. THE AREA IS LEFT IN DOWN STATUS. TO REGAIN USE OF THE AREA USE THE -UP- COMMAND.	The database area that suffered from an internal malfunction has been recovered. It has been left in a "DOWN" status to insure that the occurrence of the malfunction is noticed.	The console operator may change the area to "UP" status.	CDCS system control point dayfile
F		ATTACH ERROR nnn ON AREA id VERSION vn - AREA DOWN. SCHEMA sn	The attach error nnn occurred during system recovery processing or invoke processing on area id of version vn (version MASTER if vn does not appear). Schema sn is printed out for nonsystem recovery only.	If this error occurred during system recovery, manually recover the area. Otherwise, check that file is not attached elsewhere. For a non-PUBLIC file unavailable on NOS, ensure that a PERMIT control statement specifying the user name for CDCS or CDCSBTF execution is in effect or that the procedure to initialize CDCS includes a USER control statement specifying the user name for CDCS execution. Refer to the applicable permanent file error diagnostic in volume 4 of the NOS reference set, or in the description of the FDB macro in the NOS/BE reference manual.	CDCS output file CDCS system control point dayfile
F/N		ATTACH ERROR nnn ON CDCS file	The operating system attach error nnn occurred during system recovery or invoke processing on one of the following CDCS files: the JOURNAL LOG FILE, the PROCEDURE LIB FILE, the QUICK RECOVERY FILE, the RESTART IDENTIFIER FILE, or the TRANSACTION RECOVERY FILE. The schema is set to error status unless the file is busy.	If this error occurred during system recovery, manually recover the areas in the schema before placing the schema in up status. If this error occurred during invoke processing, resubmit the application program. Otherwise, check that file is not attached elsewhere. For a non-PUBLIC file unavailable on NOS, ensure that a PERMIT control statement specifying the user name for CDCS or CDCSBTF execution is in effect or that the procedure to initialize CDCS includes a USER control statement specifying the user name for CDCS execution. Refer to the applicable permanent file error diagnostic in volume 4 of the NOS reference set, or in the description of the FDB macro in the NOS/BE reference manual.	CDCS output file CDCS system control point dayfile

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
I		BL SET TO bbbbbb	In the absence of a BL parameter a value of bbbbb has been assigned.	None.	CDCS system control point dayfile
N		BOTH VSN AND SET NAME MUST BE GIVEN	In either the CDCS control statement or the directive file, if VSN is specified set name must be specified. If set name is specified VSN must be specified.	Specify both VSN and set name in the CDCS control statement or the directive file.	CDCS system control point dayfile
C		CANNOT GET num WORDS CM FOR jn	The 6-digit number indicates the number of words of memory that cannot be obtained to process the run-unit request. Under NOS/BE, jn is the 7-character job name; under NOS, jn is the 4-character sequence number of the system-assigned job name.	If several messages of this type appear, the data administrator should adjust the CDCS field length. If only one job is affected, rerun the job.	CDCS system control point dayfile
C		CDCS ABORT	Errors occurred while processing the CDCS control statement or the CDCS directive file.	Correct the control statement or the directive file.	CDCS system control point dayfile
C		CDCS ABORT, INVALID MASTER DIRECTORY	A master directory created and maintained by a previous version of CDCS is not compatible with CDCS 2.3.	Recreate the master directory by reintroducing all previous specifications and by using recompiled schemas and subschemas. Schemas and subschemas must be compiled with DDL 3.1 or DDL 3.2 for use with CDCS 2.3.	CDCS system control point dayfile
C		CDCS ABORT, NO SCHEMA ENTRIES IN MASTER DIRECTORY	Information for at least one schema must be included in master directory.	Check that information for the schema has not been deleted from the master directory. If deleted, add information for the schema in a master directory run.	CDCS system control point dayfile
C		CDCS ABORTED— RECOVERY FOR ALL SCHEMAS IMPOSSIBLE	All schemas are in error status after the CDCS system recovery phase.	Correct the schema errors and reinitiate CDCS execution.	CDCS system control point dayfile
I		CDCS CHARGED xxxxxx.xxx CP yyyyyy.yyy IO	CDCS accounting charged xxxxxx.xxx seconds of central processor time and yyyyyy.yyy seconds of I/O time to user jobs.	None.	CDCS system control point dayfile
I		CDCS DOWN COMPLETE	CDCS has been shut down by the system operator. No new invokes to CDCS are allowed.	None.	CDCS system control point dayfile

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
I		CDCS ERRORS	The CDCS control statement or the directive file contains errors. This message is followed by a list of the specific errors.	Correct the control statement or the directive file errors. Reinitialize CDCS.	CDCS system control point dayfile
F		CDCS FUNCTION CODE UNKNOWN DB$$SIM 1	This is an internal error.	Follow site-defined procedures for reporting software errors or operational problems.	CDCSBTF control point dayfile
I		CDCS IDLE COMPLETE	CDCS has been shut down by the system operator. No new invokes to CDCS are allowed.	None.	CDCS system control point dayfile
F		CDCS ILLEGAL REQUEST, DOUBLE INVOKE	More than one invoke was issued by the COBOL program. This is an internal error.	Follow site-defined procedures for reporting software errors or operational problems.	User control point dayfile CDCS output file
F		CDCS ILLEGAL REQUEST, ILLEGAL FUNCTION CODE	CDCS cannot process the request issued because it is an unrecognizable function code. This is an internal error.	Follow site-defined procedures for reporting software errors or operational problems.	User control point dayfile CDCS output file
F		CDCS ILLEGAL REQUEST, NOT INVOKED	CDCS cannot process the user program request because CDCS invocation has not yet occurred. This is an internal error.	Follow site-defined procedures for reporting software errors or operational problems.	User control point dayfile CDCS output file

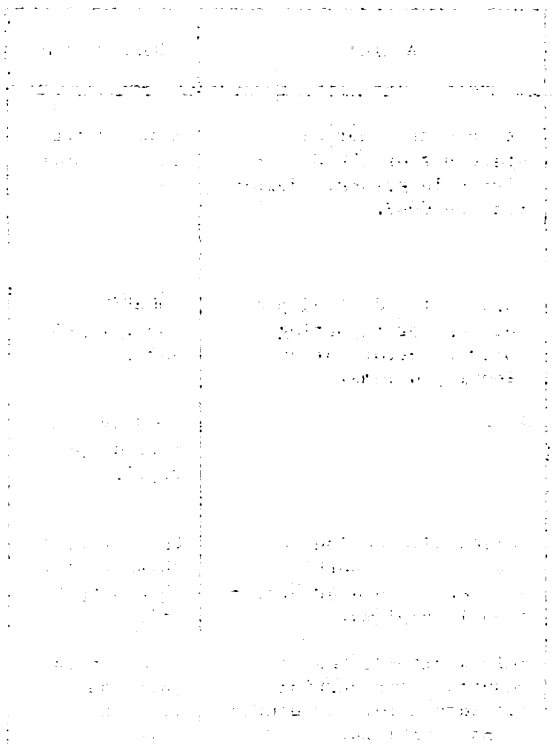


TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
F		CDCS ILLEGAL REQUEST, TWO OUTSTANDING REQUESTS	CDCS received another request from a user program before the previous request was satisfied. This is an internal error.	Follow site-defined procedures for reporting software errors or operational problems.	User control point dayfile CDCS output file
I		CDCS INITIALIZATION COMPLETE	CDCS is available at a system control point and is ready to start accepting requests from user programs.	None.	CDCS system control point dayfile
I		CDCS INTERFACE TERMINATED	CDCS termination processing has occurred.	None.	User control point dayfile CDCS output file
C		CDCS INTERNAL ERROR -- mdn	A CDCS internal error occurred in mdn; the first seven characters are the module name and the last three characters designate the location in the module where the error occurred.	Follow site-defined procedures for reporting software errors or operational problems.	CDCS system control point dayfile
I		CDCS INVOKED BY user-id	CDCS invocation has occurred by the job identified by user-id. The user-id was passed when CDCS was invoked.	None.	User control point dayfile CDCS output file
I		CDCS JOB ABORT OR ENDED BY SYSTEM	CDCS or the operating system aborted the job, or the job terminated without CDCS termination processing.	Determine cause of abort from other messages issued to the dayfiles.	User control point dayfile CDCS output file
N		CDCS JOURNAL LOG FILE NOT AVAILABLE	No journal log file in new or in use status can be found during system recovery or invoke processing. The schema is set to error status.	Analyze the reason for the journal log file being unavailable. If this error occurred during system recovery, manually recover the areas in the schema.	CDCS output file CDCS system control point dayfile
F		CDCS NOT ACTIVE AT SYSTEM CONTROL POINT	A job issued a request to CDCS when CDCS was not active at a system control point.	Check that CDCS is active at a system control point before resubmitting the application program.	User control point dayfile
I		CDCS RECOVERY COMPLETED	The CDCS system recovery phase is complete.	None.	CDCS system control point dayfile
I		CDCS RECOVERY STARTED	The CDCS system recovery phase is started.	None.	CDCS system control point dayfile
I		CDCS REPRIEVE PROCESSING DONE	CDCS processing for the abnormal termination of a job has been completed. All files attached by CDCS have been returned.	None.	CDCS system control point dayfile CDCS output file

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
F		CDCS REQUEST NOT IN FL	The job is aborted; the request cannot be processed by CDCS because of an invalid central memory address in the request.	Follow site-defined procedures for reporting software errors or operational problems.	User control point dayfile CDCS output file
I		CDCS SCP STATUS TERMINATED	CDCS has been terminated and is no longer at a system control point.	None.	CDCS system control point dayfile
I		CDCS USED aaaaaa.aaa CP SECONDS bbbbbb.bbb IO SECONDS	CDCS actually used aaaaaa.aaa central processor seconds and bbbbbb.bbb I/O seconds.	None.	CDCS system control point dayfile
I		CDCS xx.yy PERCENT CPU USAGE	CDCS CPU utilization is defined as the ratio of the total CP time used to the real time elapsed while CDCS was active (that is, from the time CDCS was initialized until the time CDCS was terminated).	None.	CDCS system control point dayfile
F		CDCSBTF ABORTED, 2 TRANSFER ADDRESSES ON USER LOAD FILE fn	The user's load file fn has multiple transfer addresses because two main programs are in the same file.	Change the load file so that it has only one main program.	CDCSBTF control point dayfile
F		CDCSBTF DEADLOCK, JOB ABORTED	CDCSBTF has determined that the user job and CDCS are both waiting for a particular event to occur, but that event is not occurring. CDCSBTF is aborted.	Correct the deadlock and resubmit the job.	CDCSBTF control point dayfile
I		CDCSBTF TERMINATED BEFORE ALL USERS COMPLETED	A CDCSBTF user program has placed END in RA+1 without referencing the external procedure SYS=.	Review each user program to be sure that it uses the normal termination statement provided by the language. For COMPASS, use the ENDRUN macro.	CDCSBTF control point dayfile
F		CDCSBTF TERMINATED, LOADER ERROR nnnnn ON USER LOAD FILE fn	A fatal loader error nnnnn occurred during the loading of user file fn.	Refer to the error message in the CYBER Loader reference manual for the appropriate action to correct the error.	CDCSBTF control point dayfile
F		CDCSBTF TERMINATED, NO PROGRAMS	No file name parameter was specified in the CDCSBTF control statement.	Specify the file name of the relocatable binary program as the parameter on the CDCSBTF control statement and resubmit the job.	CDCSBTF control point dayfile
N		CIO ERROR nnn DURING ROLLOUT	CIO encountered error nnn while rolling out part of the CDCS field length.	Follow site-defined procedures for reporting software errors or operational problems.	CDCS system control point dayfile

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
I		CIO ERROR nnn ON CDCS JOURNAL LOG FILE SCHEMA sn	CIO encountered error number nnn while performing an operation on the CDCS journal log file for the schema named sn. Some or all of the log records might have been lost. The CIO error causes the following events to occur: A memory dump is taken of the FET and of the I-O buffer at the time of the error. Requests in progress at the time of the error are aborted with error message 474. CDCS switches to an alternate log file if one is available.	To determine the disposition of the journal log file, refer to the CDCS initiated job that uses DBREC to dump the journal log file. It might be necessary to purge the journal log file and rerun DBREC to preallocate a new copy of this file. If this occurred during CDCS system recovery, manual recovery of the schema might be needed.	CDCS output file
I		CIO ERROR nnn ON QUICK RECOVERY FILESCHEMA sn	CIO encountered error number nnn while performing an operation on the quick recovery file for the schema named sn. The CIO error causes the following events to occur: A memory dump of the FET and a memory dump of the I-O buffer at the time of the error. Requests in progress at the time of the error are aborted with error message 474. CDCS sets the schema to error status and does not allow further access to it.	Purge the existing quick recovery file and rerun DBREC to initialize a new copy of this file before reinitiating CDCS. If this occurred during CDCS system recovery, manually recover the schema.	CDCS output file

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
I		CIO ERROR nnn ON TRANSACTION RECOVERY FILE-SCHEMA sn	CIO encountered error number nnn while performing an operation on the transaction recovery file for the schema named sn. The CIO error causes the following events to occur: A memory dump of the FET and a memory dump of the I-O buffer at the time of the error. Requests in progress at the time of the error are aborted with error message 474. CDCS sets the schema to error status and does not allow further access to it.	Purge the existing transaction recovery file and rerun DBREC to initialize a new copy of this file before reinitiating CDCS. If a restart identifier file exists for schema sn, it should also be initialized. Manually recover the areas affected by any uncommitted transactions. If this occurred during CDCS system recovery, manually recover the schema.	CDCS output file
F		CLTC ERROR	An internal error has occurred in clearing the long term connect.	Follow site-defined procedures for reporting software errors or operational problems.	CDCSBTF control point dayfile
I		CONTROL CARD PARAMETER xxxxx	The CDCS control statement parameter xxxxx is in error.	Correct the control statement error. Reinitialize CDCS.	CDCS system control point dayfile
I		CP SECONDS CHARGED BY CDCS xxxxxx.xxx	Central processor time of xxxxxx.xxx was used by CDCS at the system control point for processing requests for the particular user job.	The user job should be charged for xxxxxx.xxx seconds of central processor time.	User control point dayfile
I		CRM ERROR nnn ON CDCS RESTART IDENTIFIER FILE-SCHEMA sn	The indicated CRM error nnn occurred while performing an operation on the restart identifier file. If error 446 occurred, the transaction recovery file might have been reinitialized without also reinitializing the restart identifier file.	Correct the CRM error. If necessary, reinitialize the restart identifier file.	CDCS output file
C		CRM ERROR nnn WHILE ACCESSING MASTER DIRECTORY	The CRM error nnn occurred while reading the master directory.	Refer to the applicable CRM error diagnostic and correct as noted in the CYBER Record Manager Basic Access Methods reference manual.	CDCS system control point dayfile
I		DBREC JOB REQUEST	A journal log file is full. CDCS attempts to create a DBREC job to dump the log file to tape.	None.	CDCS system control point dayfile

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
I		DBREC JOB SUBMITTED JOBNAME = jn	A journal log file is full. CDCS has submitted a DBREC job with the job name jn to dump the full journal log file to tape.	None.	CDCS system control point dayfile
I		DIR FILE PARAMETER xxxxx	The CDCS directive file parameter xxxxx is in error.	Correct the directive file error and reinitialize CDCS.	CDCS system control point dayfile
I		DOWN,CDCS	The system operator has entered a DOWN,CDCS command.	None.	CDCS system control point dayfile
I		DOWN COMPLETE	The area or schema specified in a DOWN command by the system operator has been given a down status.	None.	CDCS system control point dayfile
I		DOWN IN PROGRESS	The area or schema specified in a DOWN command by the system operator is being shut down.	None.	CDCS system control point dayfile
I		DUMP CDCS JOURNAL LOG NAME = pfn	CDCS is in the process of creating a DBREC job to dump the journal log file pfn to tape.	None.	CDCS system control point dayfile
N		DUPLICATE PARAMETER	A parameter was specified more than once on the CDCS control statement or directive file.	Correct the control statement or the directive file so that the parameter is specified only once. Reinitialize CDCS.	CDCS system control point dayfile
I		END CDCSBTF RUN	CDCSBTF processing has terminated.	None.	CDCSBTF control point dayfile
N		EQUAL SIGN MUST BE PRESENT	In either the CDCS control statement or the directive file, a parameter requiring an equal sign was specified without an equal sign.	Correct the CDCS control statement or the directive file and reinitialize CDCS.	CDCS system control point dayfile

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
N		EQUAL SIGN NOT ALLOWED	In either the CDCS control statement or the directive file, a parameter that must not have an equal sign was specified with an equal sign.	Correct the CDCS control statement or the directive file. Reinitialize CDCS.	CDCS system control point dayfile
C		ERROR DURING ROLL IN--CDCS ABORTED	An error occurred while rolling in the CDCS field length that was previously rolled out for system recovery.	Follow site-defined procedures for reporting software errors or operational problems.	CDCS system control point dayfile
C		ERROR DURING ROLL OUT--CDCS ABORTED	An error occurred while rolling out that part of the CDCS field length that is unnecessary for system recovery.	Follow site-defined procedures for reporting software errors or operational problems.	CDCS system control point dayfile
F		ERROR FLAG n	The CDCS error flag has been set for user n.	Correct the error condition and return.	CDCSBTF control point dayfile
N		ERROR OR AREA id VERSION sn	During quick recovery file application processing for system recovery, an error occurred on area id.	Correct the error and manually recover the area.	CDCS system control point dayfile
I		ERROR nn WHILE REQUESTING A QUEUE DEVICE	During an attempt to route the CDCS OUTPUT file to the printer, the NOS/BE REQUEST call returned error code nn. Consult the NOS/BE Reference Manual for a description of the error code. The OUTPUT file is not routed to a printer, but no information is lost from the file. Continued writing to the OUTPUT file is not affected.	Immediate action is not required. If the problem persists Notify the data administrator.	CDCS system control point dayfile
I		ERROR nn WHILE ROUTING A PRINT FILE	During an attempt to route the CDCS OUTPUT file to the printer, the ROUTE call returned error code nn. Consult the appropriate operating system reference manual for a description of the error code. The OUTPUT file is not routed to a printer, but no information is lost from the file. Continued writing to the OUTPUT file is not affected.	Immedite action is not required. If the problem persists notify the data administrator.	CDCS system control point dayfile

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
F		FATAL CDCS ERROR -- RUN-UNIT ABORTED	A fatal error occurred during CDCS processing.	Correct the error and resubmit the application program.	User control point dayfile
F		FC COMPLETE	An SCP function was requested with the complete bit set. This is an internal error.	Follow site-defined procedures for reporting software errors or operational problems.	CDCSBTF control point dayfile
N		FDL LOAD ERROR nnn ON AREA id VERSION vn - AREA DOWN. SCHEMA sn	The FDL load error nnn occurred on the area id during processing for system recovery or CDCS transaction reversal. The area is set to error status. The schema sn is printed out for a nonsystem recovery error only.	Correct the error and manually recover the area.	CDCS output file CDCS system control point dayfile
I		FIELD LENGTH DUMP IS ON THE OUTPUT FILE	A dump of the CDCS field length has been written to the output file.	None.	CDCS system control point dayfile
N		FIRST TWO CHARACTERS MUST BE LETTERS	If the DT parameter is specified on the CDCS control statement or the directive file, the first two characters must be letters of the alphabet.	Correct the CDCS control statement or directive file and reinitialize CDCS.	CDCS system control point dayfile
I		IDLE,CDCS	The system operator has entered an IDLE,CDCS command.	None.	CDCS system control point dayfile
I		IDLE COMPLETE	The number of users of the area or schema specified in an IDLE command by the system operator has reached zero. The area or schema is given a down status.	None.	CDCS system control point dayfile
I		IDLE IN PROGRESS	The area or schema specified in an IDLE command by the system operator is being shut down. All active jobs using the area or schema are allowed to complete processing; all new jobs issuing invokes using the area or schema are aborted.	None.	CDCS system control point dayfile
I		INITIAL LOAD FL = iiiiii	Following CDCS initialization the system control point field length is iiiiii.	None.	CDCS system control point dayfile

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
F		INSUFFICIENT MEMORY FOR CDCS type	CMM overflow occurred during the following type of processing: QUICK RECOVERY FILE APPLICATION, ROLL FORWARD, or ROLL BACK. This processing (and all further recovery of the schema if in system recovery mode) is terminated. The schema is set to error status.	Manually recover all areas in the schema. Increase the CDCS field length limit.	CDCS output file CDCS system control point dayfile
I		INTERNAL DOWN COMPLETE	As a result of internal problems, CDCS has shut down the area or schema specified in the preceding THE AREA ID/NAME IS or THE SCHEMA ID/NAME IS message. An internal down of a schema occurs when the schema log files are down. An internal down of an area occurs as a result of a CRM error that is classified by CDCS as a class 1 error.	None.	CDCS system control point dayfile
N		INTERNAL ERROR - DB$MABT	An internal error has occurred while processing an operator command.	Follow site-defined procedures for reporting software errors or operational problems.	CDCS output file CDCS system control point dayfile
N		INVALID AREA IDENTIFIER id ON CDCS QUICK RECOVERY FILE	The area identifier id found on the CDCS quick recovery file during system recovery is not found in the master directory. The schema is set to error status.	Check that information for the area identified by id is present in the master directory. Manually recover the areas in the schema.	CDCS output file CDCS system control point dayfile
N		INVALID CHARACTER	An invalid character appears in one of the parameters in the CDCS control statement or directive file.	Correct the CDCS control statement or the directive and reinitialize CDCS.	CDCS system control point dayfile
N		INVALID DATA ON CDCS TRANSACTION FILE	Invalid data was found on the CDCS transaction recovery file during processing for system recovery or transaction reversal. The temporary updates made within the CDCS transaction remain in effect. The schema is set to error status.	Analyze the invalid data on the CDCS transaction recovery file. Manually reverse the updates made within any uncommitted transactions. If this error occurred during system recovery, manually recover the areas in the schema.	CDCS output file CDCS system control point dayfile

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
N		INVALID VERSION NAME vn	The version name vn was not found in the master directory during processing for system recovery or CDCS transaction reversal. Processing is terminated for the schema; the schema is set to error status.	Check that the version name vn is a valid version name in the master directory. Manually reverse the updates made within any uncommitted CDCS transactions. If this error occurred during system recovery, manually recover the areas in the schema.	CDCS output file CDCS system control point dayfile
I		IO SECONDS CHARGED BY CDCS yyyyyy.yyy	Channel time of yyyyyy.yyy was used by CDCS at the system control point for processing requests for the particular user job.	The user job should be charged for yyyyyy.yyy seconds of channel time.	User control point dayfile
F		I/O (CIO) ERROR nnn ON CDCS file	The indicated CIO error nnn occurred during processing for system recovery or CDCS transaction reversal on one of the following files: the JOURNAL LOG FILE, the QUICK RECOVERY FILE, or the TRANSACTION RECOVERY FILE. The schema is set to error status.	Correct the CIO error. If this error occurred during system recovery, manually reverse the updates made within the uncommitted CDCS transaction. If this error occurred during transaction reversal processing, manually finish reversing any uncommitted transactions.	CDCS output file CDCS system control point dayfile
F		I/O (CRM) ERROR nnn ON AREA id VERSION vn - AREA DOWN. SCHEMA sn	The CYBER Record Manager I/O error nnn occurred on the area during processing for system recovery or CDCS transaction reversal. The area is set to error status. The schema name is printed out for a nonsystem recovery error only.	Correct the error, and manually recover the area.	CDCS output file CDCS system control point dayfile
N		I/O (CRM) ERROR nnn ON CDCS RESTART IDENTIFIER FILE	The CYBER Record Manager error nnn occurred on the restart identifier file during system recovery. The schema is set to error status.	Correct the CRM error. Manually recover the areas in the schema.	CDCS output file CDCS system control point dayfile
F		JOB UNKNOWN TO SYSTEM	The CDCS request being processed is for a job that is no longer in the system.	Follow site-defined procedures for reporting software errors or operational problems.	CDCS output file

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
N		JOURNAL LOG FILE pfname FULL	The journal log file whose permanent file name is pfname has been released to be dumped to tape by DBREC. When both journal log files are in this state, jobs that use the schema associated with these files are delayed until a journal log file is available.	If both journal log files for the schema are in this state, determine if the CDCS-initiated DBREC jobs to dump the log files are executing. If they are executing, give them priority so they can complete the dump. If they are not executing, manually dump or replace the journal log files.	CDCS system control point dayfile
N		LAST RECOVERY POINT NOT FOUND	During system recovery, a matching recovery point record cannot be found in the journal log file to match the recovery point in the journal log file header. The schema is set to error status.	Analyze the journal log file. If this error occurred during system recovery, manually recover the areas in the schema.	CDCS output file CDCS system control point dayfile
N		LFN IS REQUIRED	No local file name was specified on the CDCSBTF control statement.	Add the local file name to the CDCSBTF control statement and resubmit the job.	CDCSBTF control point dayfile
N		MASTER DIRECTORY ALREADY ATTACHED	The master directory file specified on the CDCS control statement is already attached at the system control point.	Either attach the master prior to the CDCS call or specify the master directory attach information on the CDCS control statement or directive file.	CDCS system control point dayfile
C		MASTER DIRECTORY MUST BE A PERMANENT FILE	The master directory is not a permanent file, or there is not a permanent file named MSTRDIR at the CDCS control point.	The data administrator must make the master directory permanent after running the DBMSTRD utility, or must attach the master directory with the local file name MSTRDIR. The master directory can be attached in the control stream, or the master directory attach parameters can be supplied via the CDCS control statement. (The latter option is preferred.)	CDCS system control point dayfile
N		MASTER DIRECTORY NOT AVAILABLE	During CDCS initialization, an error occurred while attempting to attach the master directory specified in the CDCS control statement or directive file.	Check that the master specified is a valid permanent file, that the correct permissions have been specified, and that the master directory is not attached elsewhere. Reinitialize CDCS.	CDCS system control point dayfile

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
I		MESSAGE FOR USER nn =	Any normal user control point dayfile message produced by CDCSBTF is preceded by this message.	None.	CDCSBTF control point dayfile
N		MFL MUST BE GREATER THAN BL	In the CDCS control statement or directive file, the value specified for the MFL parameter must be greater than the value specified for the BL parameter.	Correct the error and reinitialize CDCS.	CDCS system control point dayfile
N		MUST BEGIN WITH LETTER	An alphanumeric value was specified that did not begin with a letter in the CDCS control statement or directive file.	Correct the error and reinitialize CDCS.	CDCS system control point dayfile
N		NO TERMINATOR ON CONTROL CARD	A terminator was not specified in the control CDCS statement.	Specify either a period or a right parenthesis as the last character in the CDCS control statement.	CDCS system control point
N		NOT ALLOWED ON FILE	A parameter was specified in the CDCS directive file that can only be specified in the CDCS control statement.	Move the incorrectly specified parameter from the directive file to the CDCS control statement.	CDCS system control point dayfile
I		OUTPUT FILE ROUTED TO A PRINTER	A portion of the CDCS listing has been released for printing.	None.	CDCS system control point dayfile
I		PF WAIT ON AREA an (vn)	Another job has exclusive file access for the area an of version vn (version MASTER if vn does not appear). The job trying to access the area must wait until control has been relinquished and CDCS can attach the file.	None.	User control point dayfile CDCS output file

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
N		PFM ERROR nnn ATTACHING pfn	The permanent file error nnn occurred while trying to attach file pfn.	Check that file is not attached elsewhere. For a non-PUBLIC file unavailable on NOS, ensure that a PERMIT control statement specifying the user name for CDCS or CDCSBTF execution is in effect or that the procedure to initialize CDCS includes a USER control statement specifying the user name for CDCS execution. Refer to the applicable permanent file error diagnostic in volume 4 of the NOS reference set, or in the description of the FDB macro in the NOS/BE reference manual.	CDCS system control point dayfile
I		RECOVERY COMPLETED FOR SCHEMA sn	The system recovery phase for schema sn has completed successfully.	None.	CDCS system control point dayfile
N		RECOVERY IMPOSSIBLE FOR SCHEMA sn	The schema sn is in error status after the system recovery phase.	Correct the schema errors. Manually recover the areas in the schema before placing the schema in up status.	CDCS system control point dayfile
I		RECOVERY STARTED FOR SCHEMA sn	The system recovery phase for schema sn has started.	None.	CDCS system control point dayfile
F		RUN UNIT ABORT, CDCS CMM OVERFLOW	The run-unit is aborted because not enough memory is available to load the CDCS modules required.	CDCS field length requirements and usage should be examined by the data administrator and adjusted if necessary.	User control point dayfile CDCS output file
I		RUN-UNIT TERMINATED BEFORE CDCS COMMIT OR DROP REQUEST - DROP ASSUMED	The run-unit aborted within a CDCS begin/commit sequence. The drop function is automatically performed.	None.	User control point dayfile CDCS output file
N		SCHEMA ID MISMATCH ON CDCS file	During system recovery, the schema id specified in the header of either the QUICK RECOVERY FILE or the TRANSACTION RECOVERY FILE does not match the corresponding schema id specified in the master directory. The schema is set to error status.	Analyze the master directory and system file to locate the cause of the mismatch. Manually recover the areas in the schema.	CDCS output file CDCS system control point dayfile

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
N		SCHEMA NOT UP - CDCS TRANSACTION ROLL BACK NOT COMPLETE SCHEMA sn	The schema is not in an up, idling or downing status during processing for system recovery or CDCS transaction reversal. The temporary updates made within the transaction remain in effect. Schema sn is printed out for non-system recovery only.	Manually reverse the updates made within the uncommitted transaction.	CDCS output file CDCS system control point dayfile
I		SCHEMA sn DBREC JOB REQUEST	A journal log file for schema sn is full. CDCS attempts to create a DBREC job to dump the log file to tape.	None.	CDCS system control point dayfile
F		SLTC ERROR	An internal error has occurred in setting the long term connect.	Follow site-defined procedures for reporting software errors or operational problems.	CDCSBTF control point dayfile
I		SMALL BLOCK BOUNDARY ADVANCED TO OR BEYOND ssssss nnnn TIMES	During CDCS execution the small block boundary has been advanced into the range of the small block increment beginning at ssssss on nnnn different occasions.	None	CDCS output file
I		TERM	The system operator has entered a TERM command.	None.	CDCS system control point dayfile
I		TERMINATE ENCOUNTERED BEFORE CDCS COMMIT OR DROP REQUEST - DROP ASSUMED	The application program issued a terminate within a CDCS begin/commit sequence. CDCS automatically performs the drop function to reverse uncommitted updates.	None.	User control point dayfile CDCS output file
I		THE AREA ID/NAME IS id/name	CDCS has changed the status of the specified area as a result of an operator command or an internal down. This message precedes all messages relating to the status of the area. The affected area is specified by name or identification where id is a 1- to 4-digit identification in the form ZZZ9 and name is a 1- to 30-character name in the form X(30).	None.	CDCS system control point dayfile

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
N		THE CDCS file IS EMPTY - MUST BE ALLOCATED	One of the following files was found to be empty during system recovery: the JOURNAL LOG FILE, the QUICK RECOVERY FILE, or the TRANSACTION RECOVERY FILE. The schema is set to error status.	Rerun DBREC to allocate the indicated file. Manually recover the areas in the schema.	CDCS output file CDCS system control point dayfile
I		THE SCHEMA/ID NAME IS id/name	CDCS has changed the status of the specified schema as a result of an operator command or an internal down. This message precedes all messages relating to the status of the schema. The affected schema is specified by name or identification where id is a 1- to 4-digit identification in the form ZZZ9, and name is a 1- to 30-character name in the form X(30).	None.	CDCS system control point dayfile
N		THIRD CHARACTER MUST BE NUMERIC	When the DT parameter is specified on the CDCS control statement or CDCS directive file, the third character must be a number between 0 and 9.	Correct the CDCS control statement or the CDCS directive file. Reinitialize CDCS.	CDCS system control point dayfile

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
N		TOO MANY CHARACTERS	The number of characters in a parameter value was greater than the maximum number allowed in the CDCS control statement or the CDCS directive file.	Correct the CDCS control statement or the CDCS directive file. Reinitialize CDCS.	CDCS system control point dayfile
N		TOO MANY PASSWORDS	More than five passwords for the master directory were specified in the CDCS control statement or the CDCS directive file.	Correct the CDCS control statement or the CDCS directive file. Reinitialize CDCS.	CDCS system control point dayfile
N		UN OR ID MUST BE GIVEN WITH MDPFN	The master directory name was specified without also specifying the UN or ID parameters in the CDCS control statement or CDCS directive file.	Correct the CDCS control statement or the CDCS directive file to specify the UN or ID parameter. Reinitialize CDCS.	CDCS system control point dayfile
N		UNABLE TO CREATE DBREC JOB - NO JOB/ATTACH DATA	Either the job control information was not specified in the master directory, or the attach data for the master directory was not specified in the CDCS control statement or the CDCS directive file.	Modify the master directory to include the job control information. Add the master directory attach data to the CDCS control statement or to the CDCS directive file. Manually dump and reallocate the full journal log file.	CDCS system control point dayfile
N		UNABLE TO SUBMIT DBREC JOB - I/O ERROR nnn	The I/O error nnn occurred on either the REQUEST statement (NOS/BE only) or the ROUTE statement of the DBREC job that CDCS was trying to submit.	Correct the I/O error. Manually dump and reallocate the full journal log file.	CDCS system control point dayfile
N		UNKNOWN PARAMETER	An invalid parameter was specified in the CDCS control statement or the CDCS directive file.	Correct the CDCS control statement or the CDCS directive file. Reinitialize CDCS.	CDCS system control point dayfile
I		UP,CDCS	The system operator has entered an UP,CDCS command. Subsequent invoke calls to CDCS will be allowed.	None.	CDCS system control point dayfile
I		UP COMPLETE	The system operator has removed the down status of an area or schema or terminated the effects of an IDLE command for an area or schema. Subsequent invokes using the area or schema will be allowed.	None.	CDCS system control point dayfile

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
I		UP IN PROGRESS	The area or schema is being restarted by the system operator following a previous DOWN or IDLE command.	None.	CDCS system control point dayfile
N		VALUE TOO LARGE	The value specified for the BL parameter was 1 larger than the maximum allowed (377777 octal) in the CDCS control statement or the CDCS directive file.	Specify a smaller value for the BL parameter. Reinitialize CDCS.	CDCS system control point dayfile
I		VERSION AFFECTED IS vn	The status of version vn has changed as a result of an operator command. This message precedes all messages relating to the status of the version.	None.	CDCS system control point dayfile
I		WAITING FOR CDCS	A user is trying to access CDCS when it is not active.	Contact the data administrator to initiate CDCS at a system control point.	User control point dayfile
F	384	600 - CHECKSUM MISMATCH SUBSCHEMA ssn	The checksum of the subschema ssn used to compile the applications program does not match the checksum of the subschema used at master directory run time.	Recompile and rerun program using correct subschema.	Data base status block User control point dayfile CDCS output file
N	385	601 - VIOLATION OF CONSTRAINT cn ON op OF RECORD rn	An attempt was made to perform the operation op (WRITE, DELETE, or REWRITE) on the record rn; the operation would have violated constraint cn.	Notify the data administrator.	Data base status block ZZZZZEG error file CDCS output file
N	386	602 - CRM ERROR nnn ON AREA an (vn) IN CONSTRAINT PROCESSING	A CRM error nnn occurred while performing an update operation on the area an of version vn (version MASTER if vn does not appear), which is involved in a constraint.	Correct the CRM error and resubmit the job.	Data base status block ZZZZZEG error file CDCS output file

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
N	387	603 - LOCKED RECORD/AREA-- REQUEST NOT PROCESSED	A TAF or interactive Query Update job has made a request to lock an area or to read a record in an area opened for input/output processing when the particular area or record in the area is locked by another user job. CDCS unlocks all the locks held by the TAF or interactive Query Update job and returns control to TAF or Query Update. If the TAF user is in CDCS transaction mode, the CDCS transaction is dropped.	The TAF user should include appropriate code in the application program to handle this diagnostic. Query Update issues the user a diagnostic and gives the user a choice about continuing processing.	Data base status block CDCS output file
N	388	604 - PF WAIT ON AREA an (vn)	Another job has exclusive file access to area an of version vn (version MASTER if vn does not appear) when a TAF or interactive Query Update job requested access to the area. CDCS returns control to TAF or Query Update. The TAF or Query Update job must wait for access until the other job returns the area file so that CDCS can attach the file.	TAF aborts the user job. Query Update issues the user a diagnostic and gives the user a choice about continuing processing.	Data base status block CDCS output file
N	389	605 - No message	CDCS is not active at a system control point.	Notify the data administrator.	Data base status block
F	390	606 - VERSION vn NOT IN SCHEMA sn	An invoke or version change operation specified a version name vn that cannot be found in the master directory for the schema sn.	Recompile and rerun the application program, using a correct version name for the schema sn.	Data base status block User control point dayfile CDCS output file
N	391	607 - AREA an (vn) IS NOT OPEN FOR I-O, op IS NOT ALLOWED	The operation op (REWRITE, or DELETE) is not allowed when the area an of version vn (version MASTER when vn does not appear) has not been opened for input/output processing.	Ensure that the updating program includes a statement that opens the area for both update and retrieval operations (that is, input/output processing) before the statement specifying the operation op.	Data base status block ZZZZZEG error file CDCS output file

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
N	392	610 - FOR AREA an (vn), KEY OF PRIOR READ MUST MATCH KEY ON op	The key of the record being written does not match the key of the record previously read from the area an of version vn (version MASTER if vn does not appear). The value of a data item that is a key cannot be modified; instead, the record containing the old key value must be deleted and a record containing the new key value must be written.	Resubmit the job specifying the statements to delete the old record and to write the new record.	Data base status block ZZZZZEG error file CDCS output file
F	393	611 - SCHEMA sn NOT IN MASTER DIRECTORY	The schema name sn used at execution time does not match the name of a schema for which information exists in the master directory.	Check that information for the schema has not been deleted from the master directory. If deleted, add information for the schema in a master directory run.	Data base status block User control point dayfile CDCS output file
F	394	612 - RESTART ID xxxxxxxxxx DOES NOT MATCH ID PREVIOUSLY ASSIGNED BY CDCS	The restart identifier specified in a DB$ASK or FINDTRAN request does not match the restart identifier that CDCS assigned to the run-unit earlier in a DB$GTID/ASSIGNID or DB$ASK/FINDTRAN request.	Verify the value of the restart identifier and resubmit the request.	Data base status block User control point dayfile CDCS output file
F	395	613 - AREA an NOT CLOSED BEFORE VERSION CHANGE	A version change operation was requested, but the area an was still open and had not been closed by the application program.	Recompile and rerun the application program, including the appropriate changes to the status of the open area before issuing the version change request.	Data base status block User control point dayfile CDCS output file
F	396	614 - SUBPROG SCHEMA/SUBSCHEMA NOT IDENTICAL TO MAIN PROGRAM	The subschema specified in the SUB-SCHEMA clause in the COBOL subprogram or in the SUBSCHEMA statement in the FORTRAN subprogram is not identical to the one specified in the main program.	Correct the SUB-SCHEMA clause in the COBOL subprogram and recompile, or correct the SUBSCHEMA statement in the FORTRAN subprogram and resubmit the program to the FORTRAN DML preprocessor.	Data base status block User control point dayfile CDCS output file
N	397	615 - LOCK MUST BE SET BEFORE READ ON AREA an (vn)	The request for a lock on area an of version vn (version MASTER if vn does not appear) was executed after a read was executed in the application program. When an area is open for input/output processing, an area lock (if specified) must be executed before a read is performed on the area.	Resubmit the job specifying an area lock before specifying a read request if the area is open for input/output processing.	Data base status block ZZZZZEG error file CDCS output file

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
F	398	616 - OUTSTANDING FATAL ERROR ON AREA an (vn)	A fatal error occurred on area an of version vn (version MASTER if vn does not appear) prior to the current request for input/output processing; CDCS denies the current request because of the previous error.	Check the data base status block, the user control point dayfile, or other status variables for information about the fatal error. Correct the error and resubmit the job.	Data base status block ZZZZZEG error file User control point dayfile CDCS output file
F	399	617 - NO VERSION CURRENTLY ATTACHED	A previous version change request resulted in a nonfatal error. The current request is not allowed since no files are attached.	Correct the version change request error. Recompile and rerun the application program.	Data base status block User control point dayfile CDCS output file
F	400	620 - CDCS TRANSACTION UPDATE NOT IN EFFECT sn	For a FORTRAN program an attempt was made to issue a BEGINTRAN, a COMMITTRAN, a DROPTRAN, or a FINDTRAN request for schema sn, when the master directory does not specify a transaction recovery file for this schema. For a COBOL program an attempt was made to issue a DB$BEG, a DB$CMT, a DB$DROP, or a DB$ASK request for schema sn, when the master directory for schema sn does not specify a transaction recovery file for this schema.	Modify the master directory to include the transaction recovery file clause.	Data base status block User control point dayfile CDCS output file
F	401	621 - CDCS TRANSACTION IDENTIFIER NON-BLANK	A blank CDCS transaction identifier was used in a BEGINTRAN request for a FORTRAN program or in a DB$BEG request in a COBOL program.	Recompile and rerun the application program, using a non-blank transaction identifier in the BEGINTRAN or DB$BEG request.	Data base status block User control point dayfile CDCS output file
N	402	622 - MAXIMUM NUMBER CDCS BEGIN/ COMMIT SEQUENCES EXCEEDED	The maximum number of outstanding CDCS transactions for all users of the schema has been exceeded.	Reissue the BEGINTRAN or DB$BEG request periodically. The master directory transaction unit limit might need increasing.	Data base status block CDCS output file
F	403	623 - NO OUTSTANDING CDCS BEGIN TRANSACTION REQUEST	Either a commit request or a drop request was attempted without previously issuing a begin request.	Correct and recompile the application program.	Data base status block User control point dayfile CDCS output file

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
F	405	625 - REQUEST rq NOT ALLOWED IN CDCS BEGIN/COMMIT SEQUENCE	The request rq was attempted within a CDCS begin/commit sequence.	Correct the program to issue the request outside of the begin/commit sequence.	Data base status block User control point dayfile CDCS output file
F	406	626 - ILLEGAL AREA NAME	The area name used by the application program does not match the name of an area for which information exists in the master directory.	Change the program to supply the correct area name.	Data base status block User control point dayfile CDCS output file
I	407	627 - No message	A null record occurrence was encountered on a file during relation processing.	Check the condition during processing if the application program is interested in recording null record occurances.	Data base status block C.DMRST object routine in COBOL or variable DBSTAT or DBSnnnn in FORTRAN
F	408	630 - ILLEGAL LOCK MODE	The lock mode specified in the lock request is an invalid lock mode; valid lock modes are: exclusive and protected.	Change the program to supply either the value EXCLUSIVE or PROTECTED.	Data base status block User control point dayfile CDCS output file
F	409	631 - NO CDCS RESTART IDENTIFIER FILE DEFINED	An attempt was made to either request a restart identifier or find the last committed transaction for a schema which does not have a restart identifier specified within the master directory.	Either remove the request from the application program, or define the restart identifier file within the master directory.	Data base status block User control point dayfile CDCS output file
I	410	632 - No message	A control break was encountered on a file during relation processing.	Check the condition during processing if the application program is interested in recording control break information.	Data base status block C.DMRST object routine in COBOL or variable DBSTAT or DBSnnnn in FORTRAN
N	411	633 - RESTART IDENTIFIER ALREADY ASSIGNED BY CDCS	An attempt is made to request a restart identifier after a restart identifier has already been assigned.	Correct and recompile the application program.	Data base status block CDCS output file
F	412	634 - MAXIMUM CDCS BEGIN/COMMIT UPDATE COUNT EXCEEDED	The maximum number of updates allowed within a CDCS begin/commit sequence has been exceeded.	Change the application program to issue fewer updates in a begin/commit sequence or increase the update count limit in the master directory.	Data base status block User control point dayfile CDCS output file

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
F	413	635 - SYSTEM FILE NOT AVAILABLE FOR SCHEMA sn	One of the following CDCS system files is not available: the journal log file, the transaction recovery file, the restart identifier file, the quick recovery file, or the procedure library file.	Verify that the file in question has been initialized by DBREC.	Data base status block User control point dayfile CDCS output file
F	414	636 - AREA an (vn) UNUSABLE DURING ROLL-BACK	An area error occurred applying a record to the area an version vn (version MASTER if vn does not appear) during processing to reverse a CDCS transaction. The temporary updates made within the transaction remain in effect.	Correct the problem causing the error and manually finish reversing the updates made within the CDCS transaction.	Data base status block ZZZZZEG error file User control point dayfile CDCS output file
F	415	637 - RUN ABORT, SCHEMA sn DOWN	The schema sn needed by the application program cannot be accessed because the schema sn has been marked down by the system operator. This message occurs during active processing using this schema.	Resubmit the job when the schema is brought up.	Data base status block User control point dayfile CDCS output file
F	416	640 - RUN ABORT, CDCS DOWN	CDCS is terminated; issued to active application programs.	Resubmit the job when CDCS is brought up again at a system control point.	Data base status block User control point dayfile CDCS output file
F	417	641 - SUBSCHEMA ssn NOT IN MASTER DIRECTORY	The subschema ssn used at execution time does not match the name of a subschema for which information exists in the master directory.	Check that information for the subschema is present in the master directory. If it is not present, add information for the subschema in a master directory run.	Data base status block User control point dayfile CDCS output file
F	418	642 - CDCS JOURNAL LOG FILES UNAVAILABLE SCHEMA sn	The CDCS journal log files specified for the schema sn have not been attached at CDCS initialization time, are down, or cannot be accessed for some other reason.	Check that journal log files are not in error status or are being dumped to tape by a DBREC job.	Data base status block User control point dayfile CDCS output file
F	419	643 - SCHEMA sn NOT AVAILABLE	The schema sn needed by the application program cannot be accessed. The message occurs at invocation time.	Resubmit the job when the schema is brought up again.	Data base status block User control point dayfile CDCS output file

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
F	420	644 - CDCS UNAVAILABLE--AN INVOKE IS NOT ALLOWED	The system operator is terminating, idling, or downing CDCS. CDCS no longer accepts requests from new run-units.	Resubmit job when CDCS is brought up again at a system control point.	Data base status block User control point dayfile CDCS output file
F	421	645 - ILLEGAL AREA ORDINAL	An illegal value was received from the application program.	Follow site-defined procedures for reporting software errors or operational problems.	Data base status block ZZZZZEG error file User control point dayfile CDCS output file
F	422	646 - RESTART IDENTIFIER xxxxxxxxxx	The restart identifier specified in a DB$ASK or FINDTRAN request is in use by another run-unit. This restart identifier can no longer be used in the request.	Verify and correct the old restart identifier. If this does not correct the problem, consult the data administrator.	Data base status block User control point dayfile CDCS output file
F	423	647 - PFM ERROR ec ON AREA an (vn)	The permanent file error ec occurred while attaching the area an of version vn (version MASTER if vn does not appear).	For a non-PUBLIC file unavailable on NOS, ensure that a PERMIT control statement specifying the user name for CDCS or CDCSBTF execution is in effect or that the procedure to initialize CDCS includes a USER control statement specifying the user name for CDCS execution. Refer to the applicable permanent file error diagnostic in volume 4 of the NOS reference set, or in the description of the FDB macro in the NOS/BE reference manual.	Data base status block User control point dayfile CDCS output file
F	424	650 - PFM ERROR ec ON AREA an (vn) INDEX FILE	The permanent file error ec occurred while attaching the index file for the area an of version vn (version MASTER if vn does not appear).	For a non-PUBLIC file unavailable on NOS, ensure that a PERMIT control statement specifying the user name for CDCS or CDCSBTF execution is in effect or that the procedure to initialize CDCS includes a USER control statement specifying the user name for CDCS execution. Refer to the applicable permanent file error diagnostic in volume 4 of the NOS reference set, or in the description of the FDB macro in the NOS/BE reference manual.	Data base status block User control point dayfile CDCS output file

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
F	425	651 - AREA an (vn) DOWN	The area an of version vn (version MASTER if vn does not appear) cannot be used because of physical storage failure, a fatal CRM error during area processing, or because the operator gives the area a down status. The area can be brought up again by issuing the operator command UP for the area or by reinitializing CDCS.	Notify the data administrator.	Data base status block ZZZZZEG error file User control point dayfile CDCS output file
N	426	652 - AREA an (vn) ALREADY OPEN	An open was attempted on area an of version vn (version MASTER if vn does not appear). This area has already been opened.	Correct and recompile the application program.	Data base status block ZZZZZEG error file CDCS output file

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
T/N	427	653 - CRM ERROR nnn DURING op ON AREA an (vn)	The indicated CRM error nnn occurred while performing the operation op on the area an of version vn (version MASTER if vn does not appear). If the file is structurally bad, CDCS marks the area down and does not allow further access to it.	Refer to the ZZZZZEG file and applicable CRM error diagnostic and correct as noted in the CRM Advanced Access Methods reference manual. If the file is structurally bad, reconstruct the file and give it an up status in order to allow access to the area.	Data base status block ZZZZZEG error file CDCS output file
N	428	654 - AREA an (vn) NOT OPEN	The indicated area an of version vn (version MASTER if vn does not appear) has not been opened.	Correct and recompile the application program.	Data base status block ZZZZZEG error file CDCS output file
F	429	655 - NO PROCEDURE LIBRARY SCHEMA sn	No data base procedure library has been defined in the master directory for the indicated schema sn. Data base procedures are specified in the schema definition.	Modify the master directory to include the procedure library specification.	Data base status block User control point dayfile CDCS output file
F	430	656 - INSUFFICIENT MEMORY FOR ROLLBACK SCHEMA sn	A CYBER Memory Manager (CMM) overflow condition occurred during processing to reverse a CDCS transaction. The temporary updates made within the transaction remain in effect. The schema sn is set to error status.	Manually reverse the updates made within any uncommitted transaction. Increase the memory limit for CDCS.	Data base status block User control point dayfile CDCS output file
N	431	657 - INCORRECT RECORD TYPE DURING op, AREA an (vn)	The DDL record type specified on the operation op does not match the record type determined by the RECORD CODE criteria for the area an of version vn (version MASTER if vn does not appear).	Check that record code is correct for record name given in the I/O statement.	Data base status block ZZZZZEG error file CDCS output file
N	432	660 - KEY MAPPING ERROR DURING op, AREA an (vn)	An illegal key value occurs during the operation op on the area an of version vn (version MASTER if vn does not appear). This could be a conversion error.	Correct the key value.	Data base status block ZZZZZEG error file CDCS output file
F	433	661 - FDL ERROR ec ON DBPROC pn	The indicated FDL error ec occurred while loading the indicated data base procedure pn.	Check that the data base procedure is in capsule form and has been properly put in a procedure library. Refer to the applicable FDL error code and correct as noted in the CYBER Loader reference manual.	Data base status block ZZZZZEG error file User control point dayfile CDCS output file

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
F	434	662 - ILLEGAL RECORD ORDINAL	An illegal value was received from the application program.	Recompile program. Follow site-defined procedures for reporting software errors or operational problems.	Data base status block ZZZZZEG error file User control point dayfile CDCS output file
N	435	663 - DEADLOCK ON AREA an (vn)	CDCS has unlocked all locks held by the program.	Provide appropriate code to handle recovery from a deadlock.	Data base status block ZZZZZEG error file CDCS output file
N	436	664 - ILLEGAL REQUEST ON AREA an (vn), READ OR FILE LOCK REQUIRED BEFORE op	The CDCS locking mechanism requires that the indicated operation op (REWRITE or DELETE) be preceded by a READ statement or a file lock to lock the record for area an version vn (version MASTER if vn does not appear).	Issue a READ statement in the program or lock the entire area using either the C.LOK or the DB$LKAR routine in COBOL or the DML LOCK statement in FORTRAN.	Data base status block ZZZZZEG error file CDCS output file
F	437	665 - PRIVACY BREACH ATTEMPT	The correct access control key to gain access to a given area was not supplied.	A USE FOR ACCESS CONTROL procedure must be coded in the COBOL program, a DML PRIVACY statement must be coded in the FORTRAN program, or an ACCESS directive must be entered to Query Update to supply the correct privacy key.	Data base status block ZZZZZEG error file User control point dayfile CDCS output file
F	438	666 - ERROR IN RELATION DATA NAME DEFINITION	The RESTRICT clause in the subschema references an improperly defined data name in the application program.	Correct data name and recompile the application program. If problem persists, follow site-defined procedures for reporting software errors or operational problems.	Data base status block ZZZZZEG error file User control point dayfile CDCS output file
N	439	667 - BAD RECTYPE CODE VALUE	The record type supplied by the VALUE option does not match one of the values expected by CDCS.	Correct the RECORD CODE information in the schema or the value stored in the record in the application program.	Data base status block ZZZZZEG error file CDCS output file
N	440	670 - CANNOT CHANGE REC TYPE ON op, AREA an (vn)	The record type of the data base record, as determined by the RECORD CODE criteria, cannot be modified during the indicated operation op for the area an of version vn (version MASTER if vn does not appear).	Check that the record code on the operation is the same as the original.	Data base status block ZZZZZEG error file CDCS output file

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
F	441	671 - ILLEGAL RELATION ORDINAL	An illegal value was received from the application program.	Recompile program. If problem persists, follow site-defined procedures for reporting software errors or operational problems.	Data base status block ZZZZZEG error file User control point dayfile CDCS output file
N	442	672 - BAD RECTYPE FROM DBPROC pn	The record type supplied by the indicated data base procedure pn does not match one of the values expected by CDCS.	Correct the data base procedure or the value stored in the record.	Data base status block ZZZZZEG error file CDCS output file
N	443	673 - op ABORT BY DBPROC pn	Data base procedure pn specified the indicated operation op should be aborted. Error might be intentional.	Check that data base procedure is being used correctly.	Data base status block ZZZZZEG error file CDCS output file
F	444	674 - RUN UNIT ABORT (DURING op) BY DBPROC pn	Data base procedure pn specified that the application program should be aborted during the indicated operation op. DURING op does not appear if the data base procedure was called on an error.	Check that data base procedure is being used correctly.	Data base status block ZZZZZEG error file User control point dayfile CDCS output file
N	445	675 - RECORD MAPPING ERROR DURING op ON RECORD rn	A record mapping error occurred on the indicated record rn during the specified operation op. The value of an item could not be converted or a data validation error occurred on that item (data stored into the item does not satisfy requirements in the CHECK clause). No data is transferred to or from the program working storage area. The subschema ordinal of the item in error is returned in the auxiliary status word of the data base status block.	Correct the value of the item to be converted.	Data base status block ZZZZZEG error file CDCS output file
F	446	676 - BAD RETURN CODE FROM DBPROC pn	The indicated data base procedure pn supplied a return code other than 0, 1, or 3.	Change the procedure to return a valid return code.	Data base status block ZZZZZEG error file User control point dayfile CDCS output file

TABLE B-1. CDCS DIAGNOSTICS (Contd)

Type	Dec. E.C.	Message	Significance	Action	Destination
F	447	677 - DBPROC pn NOT DEFINED FOR SCHEMA sn	The indicated data base procedure pn cannot be found in the data base procedure library for the schema sn.	Check that data base procedure is in the correct library and that the library has been specified in the master directory entry for the particular schema.	Data base status block ZZZZZEG error file User control point dayfile CDCS output file
F	472	730 - CDCS TRANSACTION RECOVERY FILE I/O ERROR DURING ROLL BACK, - SCHEMA DOWN	An error occurred reading the CDCS transaction recovery file while processing to reverse a CDCS transaction. The temporary updates made within the transaction remain in effect. The schema is set to error status.	Correct the problem causing the I/O error. Manually reverse the updates made within the uncommitted transaction.	Data base status block User control point dayfile CDCS output file
F	474	732 - CDCS SYSTEM FILE I-O ERROR	The requested function was not performed because of a CIO or CRM error on one of the CDCS logging files.	Notify the data administrator who has specific information about the error.	Data base status block User control point dayfile CDCS output file
N	475	733 - INQUIRE TRANSACTION RESTART IDENTIFIER ridname UNKNOWN TO CDCS	An attempt was made to find the CDCS transaction identifier associated with the restart identifier of a job that has terminated normally. Because the job terminated normally, the restart identifier cannot be found.	Assume the user job that was using the restart identifier has terminated normally.	Data base status block User control point dayfile CDCS output file
I		num CMM OVERFLOW CALLS MADE	The 6-digit number num indicates the number of CMM overflow calls that occurred during CDCS processing. A large number indicates excessive overhead.	Usually none. If an excessive number occurred, the data administrator should consider increasing the maximum field length for CDCS.	CDCS system control point dayfile
I		num MAXIMUM SCM WORDS USED	The 6-digit number num indicates the maximum number of central memory words used by CDCS.	None.	CDCS system control point dayfile

DBMSTRD UTILITY DIAGNOSTICS

The DBMSTRD utility diagnostic messages, listed in table B-2, are issued for syntax errors detected when reading the master directory input, for execution errors that cause the utility to abort, and for errors detected during the CST builder phase. In the following paragraphs, the DBMSTRD messages are discussed in categories that describe the type of message that is issued during a master directory creation run.

The first category is the unnumbered diagnostic messages beginning with an alphabetic character which are issued for fatal errors that cause the DBMSTRD run to abort. These messages are written to the output file and the dayfile. The following termination message also is written on the output file and the dayfile:

```
**** PROGRAM ABORT ****
```

The second category is the numbered diagnostic messages issued for syntax errors in the input file. Syntax messages are written to the output file and can be identified by a error code between 100 and 200.

The last category is the diagnostic messages that are issued if errors occur while the DBMSTRD utility is building the condensed schema/subschema (CST builder) table. These messages are written to the output file.

The DBMSTRD CST builder messages can be distinguished from the other DBMSTRD messages, listed in table B-2, in this way. A DBMSTRD CST builder message either begins with the word WARNING or begins with an error code between 7000 and 7999.

A generalized error message precedes a CST builder diagnostic when it is issued. The format of the generalized error message is shown in figure B-1. Format notations for brackets and braces are explained in the notations section earlier in this manual. The lowercase variables in the figure denote the following: subschema name (ssn), schema name (sn), a number (num), a data base element name in the subschema (nm), and the module name (mdn).

If a DBMSTRD CST builder message that is fatal to master directory processing at the schema level is issued, the following message is written to the output file:

```
***** FATAL SCHEMA ERROR, UNABLE TO
      CONTINUE PROCESSING FOR THIS SCHEMA
```

```
CST BUILDER ERROR FATAL TO {SUBSCHEMA / THE SCHEMA / MD UTILITY} FOUND WHEN PROCESSING

SUBSCHEMA ssn

OF SCHEMA sn

[WORKING ON {AREA ENTRY / THE RECORD / THE RELATN / CONSTRAINT} [WITH [A] SUBSCHEMA ORDINAL num DECIMAL]]

[{AND THE SUBSCHEMA / HAVING THE SCHEMA} ENTRY NAME nm]

[DURING {ITEM ENTRY PROCESSING FOR RECORD / CONSTRAINT PREPROCESSING FOR THE AREA}]

FATAL ERROR num IN mdn

Specific CST builder diagnostic message

THE CST BUILDER USED num SECONDS

where

   ssn    Subschema name

   sn     Schema name

   num    Number

   nm     Data base element name in the subschema

   mdn    Module name
```

Figure B-1. CST Builder Diagnostic Format

At the end of a run in which there are syntax errors or CST builder errors that are fatal to the creation of the schema or subschema entries, the following message is also written to the output file:

```
FATAL ERRORS PROHIBITED THE CREATION OF
    THE NEW MD
```

On the output file, this message is followed by run statistics.

When the preceding message is written to the dayfile, either another message that indicates execution termination is written to both the output file and the dayfile or the following message is written to the dayfile:

```
FATAL ERRORS, NO MD CREATED
```

The PROGRAM ABORT message always appears on the output file and the dayfile even when FATAL ERRORS, NO MD CREATED or some other termination message is issued.

TABLE B-2. DBMSTRD MASTER DIRECTORY DIAGNOSTICS

Message Code	Message	Significance	Action
	BAD CC PARAMETER = p	The parameter p in the utility control statement is invalid.	Correct the parameter and resubmit the job.
	BAD OLD MASTER DIRECTORY FILE	The file specified or implied by the OMD parameter is not acceptable as a valid old master directory file.	Correct the control statements and resubmit the job.
	CANNOT FIND FOLLOWING PFN ENTRY IN PFN TABLE pfn	This is an internal error.	Follow site-defined procedures for reporting software errors or operational problems.
	CRM ERR ON NEW MD, ERR CODE nnn	CRM error nnn occurred while processing the new master directory.	Refer to the applicable CRM diagnostic and correct as noted in the CRM Basic Access Methods reference manual.
	CRM ERR ON OLD MD, ERR CODE nnn	CRM error nnn occurred while processing the existing master directory.	Refer to the applicable CRM diagnostic and correct as noted in the CRM Basic Access Methods reference manual.
	CRM ERR ON SCHEMA, ERR CODE nnn	CRM error nnn occurred while processing the schema directory file.	Refer to the applicable CRM diagnostic and correct as noted in the CRM Basic Access Methods reference manual.
	CST BUILDER ERROR FATAL TO unit FOUND WHEN PROCESSING	A CST builder error has occurred; unit can be the SUBSCHEMA, SCHEMA or the MD UTILITY.	Refer to figure B-1 for the message format. Locate the specific CST builder error message later in this table.
	EMPTY INPUT FILE	A problem exists with the input file. The file might not have been rewound.	Resubmit the job with the input file included.
	ERROR IN LOADING OVL n,0	An error occurred while loading the indicated overlay n,0. The utility might have been built or installed incorrectly.	Follow site-defined procedures for reporting software errors or operational problems.
	ERROR LIMIT EXCEEDED - num ERRORS	The number (num) of errors found in syntax analysis exceeded the allowed limit.	Correct errors noted in other diagnostics and resubmit the job.
	FATAL ERROR ISSUED BY THE CST BLDR	CST builder errors prohibited master directory creation or modification. The run aborted.	Correct errors noted in other diagnostics and resubmit the job.

TABLE B-2. DBMSTRD MASTER DIRECTORY DIAGNOSTICS (Contd)

Message Code	Message	Significance	Action
	FATAL ERRORS, NO MD CREATED	Too many fatal errors occurred. A master directory could not be created. The message PROGRAM ABORT directly follows this message.	Correct errors noted in other diagnostics and resubmit the job.
	ILLEGAL CALL TO DB$MABT FROM numb	An internal error occurred. An illegal call was generated to DB$MABT from the address numb.	Follow site-defined procedures for reporting software errors or operational problems.
	ILLEGAL FILE TYPE ENCOUNTERED	This is an internal error.	Follow site-defined procedures for reporting software errors or operational problems.
	ILLEGAL OVL REQUESTED AT numb	An internal error occurred. An illegal overlay request was generated at the address numb.	Follow site-defined procedures for reporting software errors or operational problems.
	INTERNAL DBMSTRD ERROR	An internal error occurred in the DBMSTRD utility.	Follow site-defined procedures for reporting software errors or operational problems.
	INTERNAL ERROR -- DB$MABT	An internal error occurred.	Follow site-defined procedures for reporting software errors or operational problems.
	MANAGED MEMORY OVERFLOW, PROC pn	CMM overflow occurred during processing. The master directory utility procedure pn was involved.	The field length should be increased and/or the complexity of the schema and subschema should be decreased.
	MM BLOCK HANDLER ERR, PNTR AT numb	An internal error occurred in handling a managed memory block. The indicated address numb is the address of the CMM pointer for the block being manipulated.	Follow site-defined procedures for reporting software errors or operational problems.
	OLD MD FILE lfn NOT AT CP	The existing old master directory with the local file name lfn must be attached prior to calling DBMSTRD for a modification run.	Attach the existing master directory and resubmit the job.
	OLD MD LFN CANNOT BE THE SAME AS NEW MD LFN	The local file name for the existing master directory cannot be the same as the local file name for the new master directory.	Correct the local file names and resubmit the job.
	OLD MD lfn SPECIFIED WITH CREATION RUN INPUT	The OMD parameter must not be specified on a creation run.	Delete the parameter and resubmit the job.
	SCHEMA/AREA ID id TOO LARGE	The master directory contains too many schemas or areas within a schema. The indicated id exceeds the limit.	Reduce the number of schemas or areas included in a schema, correct the file, and resubmit the job.
	VARIABLE RANGE ERR/TBL ERR - tn	This is an internal error.	Follow site-defined procedures for reporting software errors or operational problems.
	WARNING, CST SIZE EXCLUDING CAPSULES NEAR MAXIMUM OF num OCTAL WORDS	The CST size in octal words is num. The number num can be from 1 to 6 digits.	None currently needed; however, if the schema and/or subschema are made more complex, subsequent master directory runs using them might result in other diagnostics.

TABLE B-2. DBMSTRD MASTER DIRECTORY DIAGNOSTICS (Contd)

Message Code	Message	Significance	Action
	WARNING, CST SIZE OF ALL CAPSULES IS NEAR MAXIMUM OF num OCTAL WORDS	The CST size of all capsules is num. The number num can be from 1 to 6 digits.	None currently needed; however, if the schema and/or subschema are made more complex, subsequent master directory runs using them might result in other diagnostics.
	WARNING, CURRENT SCHEMA MAY NOT BE PRIOR ONE USED FOR MD AND/OR SUB-SCHEMA	The schema in use might not be the same one used for a prior master directory run and/or the subschema compilation run.	Check the schema for consistency with the one used for prior master directory runs and/or the subschema compilation run.
	WARNING, FOR CDCS CORE LENGTH NOTE CST TOTAL SIZE OF num OCTAL WORDS	The total size of the CST is num. The number num can be from 1 to 6 digits.	None currently needed; however, consider this size when making CDCS field length estimates.
	wd NOT LEGAL FIRST INPUT WORD	The first word wd in the input file is not a recognizable word.	Correct the input file and resubmit the job.
101	sn IS AN INVALID SCHEMA NAME	The schema name sn is not syntactically correct.	Check requirements for schema names. Correct the input file and resubmit the job.
102	MISSING PERIOD	A period must be included to terminate the subentry.	Correct the input file and resubmit the job.
103	SCHEMA LFN WAS NOT SPECIFIED	A FILE NAME clause must be included to specify the local file name of the schema.	Correct the input file and resubmit the job.
104	FIT -XN- DID NOT INDICATE INDEX FOR AREA -an-	An INDEX FILE PFN clause is not needed for area an. The file is not a multiple-index file.	Correct the input file and resubmit the job.
105	AT LEAST ONE AREA MUST BE SPECIFIED FOR A SCHEMA	No area subentry was included for the schema.	Correct the input file and resubmit the job.
106	an IS AN INVALID AREA NAME	The area name an is not syntactically correct.	Check requirements for area names. Correct the input file and resubmit the job.
107	ALL REQUIRED PERMANENT FILE INFORMATION HAS NOT BEEN SPECIFIED	The maximum number of subclauses required to specify permanent file information was not specified for this file.	Check permanent file information requirements. Correct the input file and resubmit the job.
108	kw/an WAS ALREADY SPECIFIED FOR THE CURRENT SCHEMA	Either the area an or the keyword kw was already specified for this schema.	Correct the input file and resubmit the job.
109	AT LEAST ONE SUBSCHEMA MUST BE SPECIFIED FOR A SCHEMA	No subschema subentry was included for the schema.	Correct the input file and resubmit the job.
110	ssn IS AN INVALID SUBSCHEMA NAME	The subschema name ssn is not syntactically correct.	Check requirements for subschema names. Correct the input file and resubmit the job.
111	KEYWORD -FILE- IS MISSING	The keyword FILE to identify the FILE NAME clause is missing.	Correct the input file and resubmit the job.

TABLE B-2. DBMSTRD MASTER DIRECTORY DIAGNOSTICS (Contd)

Message Code	Message	Significance	Action
112	MAXIMUM NUMBER OF PASSWORDS EXCEEDED	More than five passwords were specified in the PW subclause.	Correct the input file and resubmit the job.
113	ILL FORMED MODIFY ADD OR DELETE HEADER	The add schemas entry, the delete schemas entry, or the modify schema entry has an incorrect entry header.	Correct the input file and resubmit the job.
114	KEYWORD -SCHEMA- IS MISSING	The keyword SCHEMA to identify the clause or header is missing.	Correct the input file and resubmit the job.
115	UNABLE TO FIND SCHEMA sn IN MD	The schema sn specified in the modify schema entry is not in the master directory.	Check the schema name for correct spelling. Correct the input file and resubmit the job.
116	wd IS AN UNKNOWN SOURCE WORD IN THE MODIFY CLAUSE	The indicated word wd is not a keyword that can be included in the modify schema entry.	Check the spelling of the word wd. Correct the input file and resubmit the job.
117	UNABLE TO FIND SUBSCHEMA ssn IN MASTER DIRECTORY	The subschema ssn specified in the DELETE SUBSCHEMA NAME clause is not in the master directory.	Check the subschema name for correct spelling. Correct the input file and resubmit the job.
118	wd IS AN INVALID SOURCE WORD IN THE ADD/DELETE CLAUSE	The word wd is not a keyword that is allowed in the context in which it is used.	Check the spelling of word wd or the requirements of the clause in which the word wd appears. Correct the input file and resubmit the job.
119	ILL FORMED END MODS CLAUSE	The END clause is not correct.	Correct the input file and resubmit the job.
120	lfn IS AN INVALID LOGICAL FILE NAME	The local file name lfn is not syntactically correct.	Check the requirements for local file names. Correct the input file and resubmit the job.
121	INVALID OR DUPLICATE CHARGE/ACCOUNT SPECIFIED	The charge/account specified in the job control information entry is greater than 70 characters, has not been properly specified, or has already been specified for this schema.	Check job control information requirements for charge/account. Correct the input file and resubmit the job.
122	pfn IS AN INVALID PERMANENT FILE NAME OR IS INAPPROPRIATE IN THE CONTEXT USED	The permanent file name pfn is not syntactically correct or has already been specified.	Check the requirements for permanent file names. Correct the input file and resubmit the job.
123	id IS AN INVALID ID/UN OR IS INAPPROPRIATE IN THE CONTEXT USED	The permanent file identification specified in the ID or UN subclause has incorrect syntax or has already been specified.	Check the requirements for permanent file identification. Correct the input file and resubmit the job.
124	pw IS AN INVALID PASSWORD	The password pw specified in the PW subclause is not syntactically correct.	Check the requirements for passwords. Correct the input file and resubmit the job.
125	wd IS IN ERROR IN THE LOG OPTIONS CLAUSE	The LOG clause contains an unrecognizable word wd.	Correct the input file and resubmit the job.

TABLE B-2. DBMSTRD MASTER DIRECTORY DIAGNOSTICS (Contd)

Message Code	Message	Significance	Action
126	NON-EXISTENT AREA NAME SPECIFIED	The area name specified for this version does not exist in the master version.	Review data base versions restrictions. Check that the correct area name has been used. Correct the input file and resubmit the job.
127	CONSTRAINTS INCONSISTENT FOR VERSION -vn-	Both permanent files for the areas involved in a constraint must either be associated with version MASTER or associated with the same alternate version.	Correct the input file and resubmit the job.
128	FIT -XN- INDICATES INDEX FOR AREA -an- BUT NO INDEX INPUT TO MD FOR AREA	An INDEX FILE PFN clause was not included in the area subentry for an area with alternate keys.	Correct the input file and resubmit the job.
129	vn IS AN INVALID VERSION NAME	The version name vn is not syntactically correct. Version name must consist of from 1 to 7 letters or digits; the first character must be alphabetic.	Correct the input file and resubmit the job.
130	VERSION NAME MASTER EXPECTED	If a VERSION NAME clause is diagnosed, VERSION MASTER is expected; permanent files of version MASTER must appear in the source input before permanent files of any alternate version. If a SAME AS MASTER clause is diagnosed, SAME AS MASTER is expected; only files of version MASTER can be shared with alternate data base versions.	Correct the input file and resubmit the job.
131	sn IS THE SCHEMA NAME IN THE SCHEMA DIRECTORY	The schema name specified in the input file does not match the schema name sn contained in the schema directory.	Check for correct schema directory file or check the spelling of the schema name. Correct the problem and resubmit the job.
132	sn IS NOT A UNIQUE SCHEMA NAME	The schema name sn is the name of a schema already in the master directory.	Correct the input file and resubmit the job.
133	-vn- IS A DUPLICATE VERSION NAME	The version name vn has already been specified for this schema; a version cannot be duplicated for a schema.	Correct the input file and resubmit the job.
134	num AREAS ARE IN SCHEMA, BUT MD INPUT DIFFERS	The schema directory contains the number of areas indicated (num), but the master directory input file does not contain the same number.	Correct the input file and resubmit the job.
135	UNABLE TO OPEN SCHEMA FILE	The schema directory file cannot be accessed properly for an open.	Check that the correct local file name has been used; correct the input file or the control statements and resubmit the job. If a CRM error diagnostic is indicated, correct as noted in the CRM Basic Access Methods reference manual.

TABLE B-2. DBMSTRD MASTER DIRECTORY DIAGNOSTICS (Contd)

Message Code	Message	Significance	Action
136	SCHEMA/SUBSCHEMA FILE IS NOT ATTACHED TO CP	The schema and subschema directories must be attached prior to calling the utility.	Attach the files and resubmit the job.
137	wd IS NOT A UNIQUE PFN/ID PACK COMBINATION	The indicated word wd is not part of a unique permanent file name/permanent file ID/permanent file pack combination.	Correct the input file and resubmit the job.
138	NO TRANSACTION RECOVERY FILE WAS FOUND FOR SCHEMA -sn-, CLAUSE INVALID	A CHANGE TRANSACTION clause has been specified for schema sn, which has no transaction recovery file.	Correct the input file and resubmit the job.
139	SAME AS MASTER INVALID FOR AREA IN MASTER VERSION	A CHANGE AREA SAME AS MASTER clause has been specified for an area in version MASTER.	Correct the input file and resubmit the job.
140	AREA an AND SCHEMA ENTRY LOGGING INFO MISMATCH	A LOG clause has been specified in the area entry requesting a form of logging not specified in the JOURNAL LOG or QUICK RECOVERY clause for the schema.	Correct the input file and resubmit the job.
141	UNABLE TO LOCATE AREA an IN THE SCHEMA DIRECTORY	The area subentry included for the schema is not one of the areas in the schema directory.	Check the area name for correct spelling. Correct the input file and resubmit the job.
142	wd IS AN INVALID SET/PACK/FAMILY NAME	The word wd given in the SET, PACK or FAMILY NAME subclause is not syntactically correct.	Check requirements for set, pack, or family names. Correct the input file and resubmit the job.
143	wd IS AN INVALID VOLUME SERIAL NUMBER	The word wd given in the VSN subclause is not syntactically correct.	Check the requirements for volume serial numbers. Correct the input file and resubmit the job.
144	INDEX FILE ALREADY SPECIFIED FOR AREA -an-	An INDEX FILE clause has already been specified for area an. An area file can be associated with only one index file.	Correct the input file and resubmit the job.
145	NO PROCEDURES IN SCHEMA, LIB SHOULD NOT BE SPECIFIED	The PROCEDURE LIBRARY PFN clause was specified for a schema that has no data base procedures specified.	Correct the input file and resubmit the job.
146	PROCEDURES IN SCHEMA, LIB MUST BE SPECIFIED	The PROCEDURE LIBRARY PFN clause must be specified for a schema that has data base procedures specified.	Correct the input file and resubmit the job.
147	INVALID TAPE TYPE SPECIFIED	A tape type other than MT or NT has been specified; specify MT or NT.	Correct the input file and resubmit the job.
148	SCHEMA sn HAS ALREADY APPEARED IN A MODIFY SCHEMA CLAUSE	A particular schema name can appear in only one MODIFY SCHEMA clause in a modification run.	Correct the input file and resubmit the job.

TABLE B-2. DBMSTRD MASTER DIRECTORY DIAGNOSTICS (Contd)

Message Code	Message	Significance	Action
149	DUPLICATE CHANGE CLAUSE	More than one CHANGE clause has been specified for the same entry of a particular schema. Only one change clause per entry is allowed.	Correct the input file and resubmit the job.
150	OPTION NOT SPECIFIED FOR SCHEMA sn, CHANGE CLAUSE INVALID	A CHANGE clause has been specified for an item in schema sn that was not specified in the creation of the existing master directory. Only those options included in the existing master directory can be specified in a CHANGE clause.	Correct the input file and resubmit the job.
151	INVALID CHANGE OPTION	The word following the keyword CHANGE in a modify schema entry is not a valid keyword. Valid keywords are PROCEDURE, TRANSACTION, RESTART, JOURNAL, QUICK, JOB, and AREA.	Correct the input file and resubmit the job.
152	AREA an HAS ALREADY APPEARED IN A CHANGE CLAUSE	A CHANGE AREA clause for a particular area can be specified only once in a modify schema entry.	Correct the input file and resubmit the job.
153	AREA an DOES NOT HAVE AN INDEX FILE SPECIFIED, CHANGE IS INVALID	The INDEX PFN clause can be specified only for a file for which information already exists in the master directory. The INDEX PFN clause is specified for area an, which has no index file.	Correct the input file and resubmit the job.
154	ILLEGAL OPTION IN CHANGE AREA CLAUSE	The option selected in the CHANGE AREA clause is not a valid option for the area.	Check the CHANGE AREA clause restrictions.
155	SETNAME WITHOUT VSN OR VSN WITHOUT SETNAME CLAUSE IS ILLEGAL	If a VSN subclause is specified, a SETNAME subclause must also be specified or vice versa.	Correct the input file and resubmit the job.
156	SCHEMA FILE CLAUSE NEEDED BEFORE ADD CLAUSE	If an ADD SUBSCHEMA clause is specified in a modify schema entry, the FILE NAME clause, which specifies the local file name of the schema, must be specified immediately following the MODIFY SCHEMA NAME clause.	Correct the input file and resubmit the job.
157	KEYWORD -PFN- IS MISSING	The keyword PFN to identify the PERMANENT FILE NAME clause is missing.	Correct the input file and resubmit the job.
158	INVALID TRANSACTION UNIT LIMIT	The value specified as a transaction unit limit is syntactically incorrect or greater than 63.	Specify a positive integer equal to or less than 63; correct the input file and resubmit the job.
159	INVALID TRANSACTION UPDATE LIMIT	The value specified as a transaction update limit is syntactically incorrect or greater than 63.	Specify a positive integer equal to or less than 63; correct the input file and resubmit the job.

TABLE B-2. DBMSTRD MASTER DIRECTORY DIAGNOSTICS (Contd)

Message Code	Message	Significance	Action
160	-dt- IS AN INVALID DEVICE TYPE	The device type selected in the TYPE option of the TAPE clause is syntactically incorrect. Only MT or NT is allowed.	Correct the input file and resubmit the job.
161	RESTART IDENTIFIER FILE CANNOT BE SELECTED WITHOUT TRANSACTION FILE	The RESTART IDENTIFIER clause cannot be specified in this schema because no TRANSACTION RECOVERY FILE clause has been specified.	Correct the input file and resubmit the job.
162	JOURNAL/TRANSACTION PFN MUST BE 6 CHARACTERS	The permanent file name selected for the journal log file or the transaction recovery file is syntactically incorrect.	Check permanent file name restrictions for journal log and transaction recovery files. Correct the input file and resubmit the job.
163	NON-EXISTENT VERSION NAME SPECIFIED	The version name specified in the CHANGE AREA clause of the modify schema entry is not included in the exisiting master directory.	Check version name for correct spelling. Correct the input file and resubmit the job.
164	MAXIMUM NUMBER OF VERSIONS EXCEEDED	The maximum number of versions that can be specified for a schema has been exceeded; the maximum is 4095.	Correct the input file and resubmit the job.
165	INSUFFICIENT JOB CONTROL INFORMATION SPECIFIED	The minimum number of subclauses required to specify job control information were not specified for this schema.	Check requirements for JOB CONTROL INFORMATION clause. Correct the input file and resubmit the job.
166	INVALID DENSITY OPTION SPECIFIED	A tape density option other than LO, HI, HY, HD, PE, or GE has been specified for this schema.	Correct the input file and resubmit the job.
167	INVALID TAPE/DENSITY COMBINATION	The tape type and tape density options in the JOB CONTROL INFORMATION clause that were selected for this schema are incorrect. Valid options are: For type NT, density HD, PE, or GE; for type MT, density LO, HI, or HY.	Correct the input file and resubmit the job.
168	DUPLICATE TAPE TYPE/DENSITY CLAUSE	The TAPE TYPE or TAPE DENSITY clause has already been specified for this schema.	Correct the input file and resubmit the job.
169	INVALID OR DUPLICATE JOB CONTROL USER NAME	The user name specified in the JOB CONTROL INFORMATION clause is syntactically incorrect or the UN clause has already been specified for this schema.	Correct the input file and resubmit the job.
170	INVALID OR DUPLICATE JOB CONTROL FAMILY NAME	The family name specified in the JOB CONTROL INFORMATION clause is syntactically incorrect or the FAMILY clause has already been specified for this schema.	Correct the input file and resubmit the job.
171	INVALID OR DUPLICATE JOB CONTROL PASSWORD	The password specified in the JOB CONTROL INFORMATION clause is syntactically incorrect or the PW clause has already been specified for this schema.	Correct the input file and resubmit the job.

TABLE B-2. DBMSTRD MASTER DIRECTORY DIAGNOSTICS (Contd)

Message Code	Message	Significance	Action
7001	SUBSCHEMA NOT FOUND - LFN IS lfn	The subschema directory could not be accessed. It must be attached prior to calling DBMSTRD. The local file name in the input file is lfn. The affected entry is the subschema subentry.	Check that correct local file name has been used. Correct the input file or control statements and resubmit the job.
7002	SUBSCHEMA HAS A SCHEMA NAME sn	The subschema directory has the name sn. The master directory input file indicates another name. The affected entry is the subschema subentry.	Check that the correct subschema directory has been used or correct the input file. Resubmit the job.
7003	SUBSCHEMA BAD DDL VERSION, LFN lfn	The subschema directory with local file name lfn was compiled using a previous version of DDL. DDL3 must be used. The affected entry is the subschema subentry.	Recompile the subschema and schema, if necessary. Resubmit the job and attach the new directories.
7004	SUBSCHEMA AREA COUNT EXCESS num DECIMAL	The subschema contains too many areas. The excess is a 1 to 4 digit decimal number num. The affected entry is the subschema subentry.	Reduce the number of areas and recompile the subschema and the schema. Correct the input file and resubmit the job.
7005	SUBSCHEMA REC COUNT EXCESS - num DECIMAL	The subschema contains too many records. The excess is a 1 to 4 digit decimal number num. The affected entry is the subschema subentry.	Reduce the number of records, recompile the subschema and the schema. Resubmit the job.
7006	SUBSCHEMA REL COUNT EXCESS - num DECIMAL	The subschema contains too many relations. The excess is a 1 to 4 digit decimal number num. The affected entry is the subschema subentry.	Reduce the number of relations and recompile the subschema and schema. Resubmit the job.
7007	SCHEMA CANNOT BE FOUND - LFN lfn	The schema directory could not be accessed. It must be attached prior to calling DBMSTRD. The local file name in the input file is lfn. The affected entry is the schema subentry.	Check that correct local file name has been used. Correct the input file or the control statements and resubmit the job.
7008	SCHEMA CONTAINS SCHEMA NAME sn	The schema directory has the schema name sn. The master directory input indicates another name. The affected entry is the schema subentry.	Check that both the correct schema directory and the correct input file have been used. Resubmit the job.
7009	SCHEMA BAD DDL VERSION - LFN lfn	The schema directory with local file name lfn was compiled using a previous version of DDL. DDL 3 must be used. The affected entry is the schema subentry.	Recompile both the schema and the subschema. Resubmit the job and attach the new directories.
7010	SCHEMA MAX ITEMS/REC EXCESS num DECIMAL	The schema contains too many items in a specific record. The excess is a 1 to 4 digit decimal number num. The affected entry is the schema subentry.	Reduce the maximum number of items in a record and recompile both the schema and the subschema. Resubmit the job.

TABLE B-2. DBMSTRD MASTER DIRECTORY DIAGNOSTICS (Contd)

Message Code	Message	Significance	Action
7011	SCHEMA MISMATCHED DBP COUNT num DECIMAL	The 1 to 4 digit decimal number num indicates the count of data base procedures specified in the current schema directory. The count is not the same as the count of data base procedures for the version of the schema in the old master directory file. The affected entry is the schema subentry.	Correct the inconsistencies between the different versions of the schema and resubmit the job.
7051	SCHEMA HAS NO ENTRY FOR NAME nm	The subschema directory contains an entry for the name nm and no corresponding name is found in the schema directory. The affected entry is the subschema subentry.	Check that the correct schema and subschema have been used. Resubmit the job.
7101	SCHEMA MISMATCHED AREA COUNT num DECIMAL	The 1 to 4 digit decimal number num indicates the count of areas in the schema directory. The number of areas described in the master directory input file is not the same. The affected entry is the schema subentry.	Correct the input file and resubmit job.
7102	RAISE MAXIMUM FL BY AT LEAST num OCTAL WORDS	The field length must be increased in order for the utility to run. The 1 to 6 digit number num indicates the minimum amount to increase the field length. More than this minimum might be needed. The affected entry is the subschema subentry.	Correct the field length parameter and resubmit the job.
7151	SCHEMA MISMATCHED REL COUNT num DECIMAL	The 1 to 4 digit decimal number num indicates the count of relations in the current schema directory. The count is not the same as the count of relations for the version of the schema in the old master directory file. The affected entry is the schema subentry.	Correct the inconsistencies between the different versions of the schema and resubmit the job.
7152	SCHEMA MISMATCHED AREA COUNT num DECIMAL	The 1 to 4 digit decimal number num indicates the count of areas in the current schema directory. The count is not the same as the count of areas for the version of the schema in the old master directory file. The affected entry is the schema subentry.	Correct the inconsistencies between the different versions of the schema and resubmit the job.
7153	SEARCH FAIL - SCHEMA AREA IS an	The area name an in the current schema directory was not found in the version of the schema in the old master directory file. The affected entry is the schema subentry.	Correct the inconsistencies between the different versions of the schema and resubmit the job.

TABLE B-2. DBMSTRD MASTER DIRECTORY DIAGNOSTICS (Contd)

Message Code	Message	Significance	Action
7154	SCHEMA BAD CHECKSUM FOR AREA an	The area name an in the current schema directory has a checksum that is not the same as the checksum of the area an for the version of the schema in the existing master directory file. The affected entry is the schema subentry.	Correct the inconsistencies between the different versions of the schema and resubmit the job.
7155	SEARCH FAIL - SCHEMA REL IS rn	The relation name rn in the current schema directory was not found in the version of the schema in the existing master directory file. The affected entry is the schema subentry.	Correct the inconsistencies between the different versions of the schema and resubmit the job.
7156	SCHEMA BAD CHECKSUM FOR REL rn	The relation rn in the current schema directory has a checksum that is not the same as the checksum of the relation rn for the version of the schema in the old master directory file. The affected entry is the schema subentry.	Correct the inconsistencies between the different versions of the schema and resubmit the job.
7157	SEARCH FAIL - SUBSCHEMA AREA an	The schema area name an in the current subschema directory was not found in the current schema directory. The affected entry is the subschema subentry.	Correct the inconsistencies between the subschema and the schema and resubmit the job.
7158	SUBSCHEMA BAD CHECKSUM, AREA an	The schema area an in the current subschema directory has a checksum that is not the same as the checksum of the area an in the current schema directory. The affected entry is the subschema subentry.	Correct the inconsistencies between the subschema and the schema and resubmit the job.
7159	SEARCH FAIL - SUBSCHEMA REL rn	The schema relation name rn in the current subschema directory was not found in the current schema directory. The affected entry is the subschema subentry.	Correct the inconsistencies between the subschema and the schema and resubmit the job.
7160	SUBSCHEMA BAD CHECKSUM - REL rn	The schema relation rn in the current subschema directory has a checksum that is not the same as the checksum of the relation rn in the current schema directory. The affected entry is the subschema subentry.	Correct the inconsistencies between the subschema and the schema and resubmit the job.
7201	ALLOCATE OVERFLOW POINTER AT num OCTAL ADDRESS	CMM overflow occurred during processing. The number num is a 1 to 6 digit number indicating the address of the pointer word for the block being allocated when overflow occurred. The affected entry is the subschema subentry.	Increase the maximum field length and/or decrease the complexity of the schemas and subschemas. Resubmit the job.

TABLE B-2. DEMSTRD MASTER DIRECTORY DIAGNOSTICS (Contd)

Message Code	Message	Significance	Action
7202	CMM GROW OVERFLOW POINTER AT num OCTAL ADDRESS	The CMM block size cannot be increased. The number num is a 1 to 6 digit number indicating the address of the pointer word for the block being manipulated when overflow occurred. The affected entry is the subschema subentry.	Increase the maximum field length or decrease the complexity of the schemas and subschemas. Resubmit the job.
7301	UNRECOGNIZABLE ERROR IDENT - id	The error identification id indicates an unrecognizable error. This is an internal problem. All entries are affected.	Follow site-defined procedures for reporting software errors or operational problems.
7331	CMM OVERFLOW, UP FL AT LEAST num OCTAL WORDS	CMM overflow occurred during processing. The number num indicates a 1 to 6 digit number by which the field length should be increased at a minimum. More than this minimum might be needed. All entries are affected.	Increase the maximum field length and/or decrease the complexity of the schema and subschema. Resubmit the job.
7371	MASTER DIRECTORY CRM ERROR - nnn OCTAL	The indicated CRM error nnn occurred while processing the new master directory. All entries are affected.	Refer to the applicable CRM error diagnostic and correct as noted in the CRM Basic Access Methods reference manual.
7372	SUBSCHEMA FILE CRM ERROR IS nnn OCTAL	The indicated CRM error nnn occurred while processing the subschema directory. The affected entry is the subschema subentry.	Refer to the applicable CRM error diagnostic and correct as noted in the CRM Basic Access Methods reference manual.
7373	SCHEMA DIRECTORY CRM ERROR - nnn OCTAL	The indicated CRM error nnn occurred while processing the schema directory. The affected entry is the schema subentry.	Refer to the applicable CRM error diagnostic and correct as noted in the CRM Basic Access Methods reference manual.
7401	ILLEGAL PROC TYPE OF OPTIONS typ OCTAL	The data base procedure type option is not one of the legal types. The type typ gives the illegal type. This is an internal problem. All entries are affected.	Follow site-defined procedures for reporting software errors or operational problems.
7402	NO OPTION FOR CST PROC ENTRY num DECIMAL	The data base procedure entry number num has no option. The number num is a 1 to 4 digit number. The DDL compiler might not have generated an option value. The affected entry is the schema subentry.	Recompile the schema and resubmit the job.
7403	CST PROC TABLE LEN CANNOT BE num OCTAL WORDS	The CST data base procedure table length has exceeded the limit. The number num is a 1 to 6 digit number. The affected entry is the subschema subentry.	Decrease the number of data base procedures specified and recompile both the schema and the subschema. Resubmit the job.

TABLE B-2. DBMSTRD MASTER DIRECTORY DIAGNOSTICS (Contd)

Message Code	Message	Significance	Action
7404	CST PROC LIST SIZE MUST BE num OCTAL WORDS	The length of the list of data base procedures in the CST must be less than the limit indicated by the number num. The number num is a 1 to 6 digit number. The affected entry is the schema subentry.	Decrease the number of data base procedures specified, and recompile both the schema and the subschema. Resubmit the job.
7451	SEARCH FAIL - SCHEMA DB PROC pn	The data base procedure pn was not found in the schema data base procedure list in the master directory file. The schema entry in the CST cannot be completed. The affected entry is the schema subentry.	Correct the inconsistencies between the different versions of the schema (perhaps by making a master directory run to delete old information for the schema). Resubmit the job.
7452	SEARCH FAIL - SUBSCHEMA DBP pn	The data base procedure pn was not found in the schema data base procedure list in the master directory file. The subschema entry in the CST cannot be completed. The affected entry is the subschema subentry.	Correct the inconsistencies between the different versions of the schema (perhaps by making a master directory run to delete old information for the schema) and/or recompile the subschema if the wrong schema was used. Resubmit the job.
7501	ILLEGAL LOCK DB ELEMENT TYPE typ OCTAL	The data base element indicated type typ is illegal for an access control lock. This is an internal error. All entries are affected.	Follow site-defined procedures for reporting software errors or operational problems.
7502	SCHEMA BAD PRIVACY LOCK TYPE typ OCTAL	The access control lock type typ in the schema directory is invalid. The schema directory file might be in error. The affected entry is the schema subentry.	Recompile the schema and resubmit the job.
7503	SUBSCHEMA HAS BAD LOCK TYPE typ OCTAL	The access control lock type typ in the subschema directory is invalid. The subschema directory file might be in error. The affected entry is the subschema subentry.	Recompile the subschema and resubmit the job.
7504	NO OPTION FOR CST LOCK ENTRY num DECIMAL	The access control lock option is missing. The lock number entry in the CST is num. The schema directory file might be in error. The affected entry is the schema subentry.	Recompile the schema and resubmit the job.
7505	NO OPTION FOR CST LOCK ENTRY num DECIMAL	The access control lock option is missing. The lock number entry in the CST is num. The subschema directory file might be in error. The affected entry is the subschema subentry.	Recompile the subschema and resubmit the job.
7601	CST NON-CAPSULE EXCESS SIZE num OCTAL WORDS	The CST size (not including capsules) has exceeded the size limit. The number num indicates the amount over the limit. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. Resubmit the job.

TABLE B-2. DBMSTRD MASTER DIRECTORY DIAGNOSTICS (Contd)

Message Code	Message	Significance	Action
7602	CST MAP CAPSULE EXCESS SIZE num OCTAL WORDS	The CST mapping capsules have exceeded the size limit. The number num indicates the amount over the limit. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. Resubmit the job.
7603	TRUNCATION - CST REC POINTER num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7604	TRUNCATION - RSB REC POINTER num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7605	TRUNCATION - CST REL POINTER num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7606	TRUNCATION - RSB REL POINTER num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7607	TRUNCATION - PROC OPTION PNT num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7608	TRUNCATION - PRIVACY LOCK PT num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7609	TRUNCATION - RSB LOCK POINTR num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7610	TRUNCATION - RSB SEARCH PNTR num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.

TABLE B-2. DBMSTRD MASTER DIRECTORY DIAGNOSTICS (Contd)

Message Code	Message	Significance	Action
7611	TRUNCATION - RSB DATA POINTR num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7612	TRUNCATION - RSB JOIN POINTR num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7613	TRUNCATION - RSB STACK POINT num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7614	TRUNCATION - RSB TOTAL SIZE num OCTAL WORDS	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7615	TRUNCATION - NUMBER OF AREAS num DECIMAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the number of areas in the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7616	TRUNCATION - NUMBER OF RECS num DECIMAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the number of records in the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7617	TRUNCATION - NUMBER OF RELS num DECIMAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the number of relations in the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7618	TRUNCATION - SUBSCHEMA MAXI REC num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the record size of records in the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7619	TRUNCATION - SCHEMA MAX REC num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the record size of records in the schema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.

TABLE B-2. DBMSTRD MASTER DIRECTORY DIAGNOSTICS (Contd)

Message Code	Message	Significance	Action
7620	TRUNCATION - MAP CAPSULE PNT num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7621	TRUNCATION - MAP CAPSULE LEN num OCTAL WORDS	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7622	TRUNCATION - MAX CAPSULE LEN num OCTAL WORDS	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7623	TRUNCATION - NUMBER CAPSULES num DECIMAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7624	TRUNCATION - CST CONSTRNT PT num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7625	TRUNCATION - RSB CONSTRNT PT num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7626	TRUNCATION - NUMBER EXT AREA num DECIMAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the number of constraints in the schema in which areas in the subschema are involved. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7627	TRUNCATION - NUMBER CONSTRNT num DECIMAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the number of constraints in the schema in which areas in the subschema are involved. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.

TABLE B-2. DBMSTRD MASTER DIRECTORY DIAGNOSTICS (Contd)

Message Code	Message	Significance	Action
7628	TRUNCATION - EXTEND MAX REC num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the record size of records in schema areas involved in constraints in which areas in the subschema are involved. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7701	SUBSCHE NEXT= CURRENT ADDRESS num OCTAL	In the subschema directory, the next address to be accessed is the same as the current address num. The affected entry is the subschema subentry.	Recompile the subschema and resubmit the job. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7702	SUBSCHEMA HAS BAD AREA TYPE typ OCTAL	An invalid value type typ exists in the subschema directory for the area type. The affected entry is the subschema subentry.	Recompile the subschema and resubmit the job. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7703	SCHEMA HAS THE BAD AREA TYPE typ OCTAL	An invalid value type typ exists in the schema directory for the area type. The affected entry is the schema subentry.	Recompile the schema and resubmit the job. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7704	SEARCH FAIL - SCHEMA AREA IS an	A problem occurred in obtaining area information for area an from the master directory file. The affected entry is the schema subentry.	Check consistency between the current schema and the version in the existing master directory file. Resubmit the job. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7705	TRUNCATION - AREA NAME SIZE num OCTAL WORDS	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the length of the area name. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7706	TRUNCATION - KEY CAPSULE LEN num OCTAL WORDS	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7707	TRUNCATION - KEY CAPSULE PNT num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7708	TRUNCATION - KEY TABLE POINT num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.

TABLE B-2. DBMSTRD MASTER DIRECTORY DIAGNOSTICS (Contd)

Message Code	Message	Significance	Action
7709	TRUNCATION - CODE TABLE PNTR num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7710	TRUNCATION - LITERAL POINTER num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7711	SUBSCHEMA BAD REC CODE SIZE num OCTAL WORDS	An invalid record code table size num exists in the subschema directory. The affected entry is the subschema subentry.	Recompile the subschema and resubmit the job. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7712	TRUNCATION - WORK BLOCK SIZE num OCTAL WORDS	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7713	TRUNCATION - DEPEND TABLE PT num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7751	SUBSCHEMA HAS BAD AREA TYPE typ OCTAL	An invalid value type typ exists in the subschema directory for the area type. The affected entry is the subschema subentry.	Recompile the subschema and resubmit the job. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7752	SCHEMA HAS THE BAD AREA TYPE typ OCTAL	An invalid value type typ exists in the schema directory for the area type. The affected entry is the schema subentry.	Recompile the schema and resubmit the job. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7753	SCHEMA BAD CONSTRAINT TYPE - typ OCTAL	An invalid value type typ exists in the schema directory for the constraint type. The affected entry is the schema subentry.	Recompile the schema and resubmit the job. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7754	SCHEMA BAD DATA - CONSTRAINT cn	Invalid area information exists in the schema directory for the constraint cn. The affected entry is the schema subentry.	Recompile the schema and resubmit the job. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.

TABLE B-2. DBMSTRD MASTER DIRECTORY DIAGNOSTICS (Contd)

Message Code	Message	Significance	Action
7801	SUBSCHEMA NEXT= CURRENT ADDRESS num OCTAL	In the subschema directory, the next address to be accessed is the same as the current address num. The affected entry is the subschema subentry.	Recompile the subschema and resubmit the job. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7802	SUBSCHEMA BAD RECORD TYPE IS typ OCTAL	An invalid value type typ exists in the subschema directory for the record type. The affected entry is the subschema subentry.	Recompile the subschema and resubmit the job. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7803	SCHEMA HAS A BAD RECORD TYPE typ OCTAL	An invalid value type typ exists in the schema directory for the record type. The affected entry is the schema subentry.	Recompile the schema and resubmit the job. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7804	TRUNCATION - RECORD NAME LEN num OCTAL WORDS	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the length of the record name. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7805	TRUNCATION - SUBSCHEMA ITEMS num DECIMAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the number of items in the record in the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7806	TRUNCATION - SUBSCHEMA REC LEN num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the length of the record in the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7807	TRUNCATION - SCHEMA REC ITEM num DECIMAL	An overflow condition exists in a CST field. The affected entry is the schema subentry.	Decrease the number of items in the record in the schema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7808	TRUNCATION - SCHEMA REC SIZE num OCTAL	An overflow condition exists in a CST field. The affected entry is the schema subentry.	Decrease the length of the record in the schema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7809	TRUNCATION - READ CAPSL SIZE num OCTAL WORDS	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.

TABLE B-2. DBMSTRD MASTER DIRECTORY DIAGNOSTICS (Contd)

Message Code	Message	Significance	Action
7810	TRUNCATION - READ CAPSL PNTR num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7811	TRUNCATION - WRITE CAPSL LEN num OCTAL WORDS	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7812	TRUNCATION - WRITE CAPSL PNT num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7813	SCHEMA GET ERROR - ITEM NAME nm	Information about item nm could not be obtained from the schema directory. The affected entry is the subschema subentry.	Check that consistent versions of the schema and subschema are being used. Recompile the schema and/or the subschema and resubmit the job. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7814	TRUNCATION - ITEM LOCK POINT num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7815	TRUNCATION - WORK BLOCK SIZE num OCTAL WORDS	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7901	SUBSCHEMA NEXT= CURRENT ADDRESS num OCTAL	In the subschema directory, the next address to be accessed is the same as the current address num. The affected entry is the subschema subentry.	Recompile the subschema and resubmit the job. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7902	SUBSCHEMA BAD RELATION TYPE typ OCTAL	An invalid value type typ exists in the subschema directory for the relation type. The affected entry is the subschema subentry.	Recompile the subschema and resubmit the job. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7903	TRUNCATION - RELATN MAX RANK num DECIMAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the maximum rank in the relation. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.

TABLE B-2. DBMSTRD MASTER DIRECTORY DIAGNOSTICS (Contd)

Message Code	Message	Significance	Action
7904	TRUNCATION - REL NAME LENGTH num OCTAL WORDS	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the length of the relation name. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7905	TRUNCATION - SEARCH TABLE PT num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7906	BAD RELATION QUAL TABLE TYPE typ OCTAL	An invalid value type typ exists in the relation qualification table in the subschema directory. The affected entry is the subschema subentry.	Recompile the subschema and resubmit the job. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7907	TRUNCATION - QUAL TABLE PNTR num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7908	TRUNCATION - QUAL MAX RANK - num DECIMAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the maximum qualified rank in the relation. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7909	TRUNCATION - QUAL TABLE SIZE num OCTAL WORDS	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the qualification information for the relation. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7910	TRUNCATION - WORK BLOCK SIZE num OCTAL WORDS	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7911	BAD QUAL TABLE STACK TYPE IS typ OCTAL	An invalid value type typ exists in the qualification stack in the subschema directory. The affected entry is the subschema subentry.	Recompile the subschema and resubmit the job. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7912	TRUNCATION - QUAL STACK SIZE num OCTAL WORDS	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the qualification information for the relation. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.

TABLE B-2. DBMSTRD MASTER DIRECTORY DIAGNOSTICS (Contd)

Message Code	Message	Significance	Action
7913	TRUNCATION - ATTRIBUTE POINT num OCTAL	An overflow condition exists in a CST field. The affected entry is the subschema subentry.	Decrease the complexity of the schema and the subschema. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7914	BAD REL QUAL TABLE AREA RANK num DECIMAL	An invalid rank value num exists in the relation qualification table in the subschema directory. The affected entry is the subschema subentry.	Recompile the subschema and resubmit the job. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7915	QUAL ABOVE MAX RANK FOR RANK num DECIMAL	An invalid maximum rank value num exists in the relation qualification table in the subschema. The affected entry is the subschema subentry directory.	Recompile the subschema and resubmit the job. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7916	BAD QUAL STACK ENTRY POINTER num OCTAL	An invalid pointer num exists in the relation qualification table in the subschema directory. The affected entry is the subschema subentry.	Recompile the subschema and resubmit the job. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7917	BAD RELATION QUAL TABLE SIZE num OCTAL WORDS	An invalid relation qualification table size num exists in the subschema directory. The affected entry is the subschema subentry.	Recompile the subschema and resubmit the job. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.
7918	BAD REL QUAL TABLE MAX RANK num DECIMAL	An invalid maximum qualified rank value num exists in the relation entry in the subschema directory. The affected entry is the subschema subentry.	Recompile the subschema and resubmit the job. If the problem persists, follow site-defined procedures for reporting software errors or operational problems.

DBQRFA AND DBQRFI DIAGNOSTICS

The diagnostic messages for the DBQRFA and DBQRFI utilities are listed in table B-3. All messages are written to the dayfile. Messages issued by the DBQRFA utility can be distinguished from messages issued by the DBQRFI utility by referring to the ISSUED BY column. These messages are of the following types:

F Fatal error message. The DBQRFA or the DBQRFI utility run is aborted.

N Nonfatal error message. The DBQRFA utility continues processing.

TABLE B-3. DBQRFA AND DBQRFI DIAGNOSTICS

Type	Message	Significance	Action	Issued By
N	AREA an CANNOT BE RESTORED	The area an is in an unprocessable state. The DBRST or DBRCN utility cannot be run to restore or recover this area.	This area can be reloaded from a file dump and recovered or restored from a journal log file. Some application programs might have to be rerun.	DBQRFA
F	BAD CONTROL CARD PARAMETER = p	The parameter p in the DBQRFA control statement is invalid.	Correct the parameter and resubmit the job.	DBQRFA
F	CANNOT FIND SCHEMA IN MASTER DIRECTORY	The schema identification specified in the control statement parameter SC is not found in the master directory specified by the MD parameter.	Correct the parameters and resubmit the job.	DBQRFA
F	CIO ERROR ON QRF FILE	A CIO error occurred while reading the quick recovery file.	The file cannot be updated because the quick recovery file is unprocessable. Use a backup copy of the data base and the DBRCN utility to recover.	DBQRFA
N	CIO ERROR nnn WHILE RESTORING FILE	The CIO error nnn occurred while processing the file that is specified in a previous message. The CIO error (REWRITE BEYOND EOI) frequently occurs on the NOS/BE system and can be ignored.	The file cannot be updated because it is unprocessable. Use a backup copy of the data base to recover.	DBQRFA
F	CRM ERROR nnn WHILE ACCESSING MASTER DIRECTORY	The CRM error nnn occurred while reading the master directory.	Refer to the applicable CRM error diagnostic and correct as noted in the Basic Access Methods reference manual.	DBQRFA
N	DBQRFA COMPLETE	The DBQRFA utility applied the contents of the quick recovery file to the data base.	None.	DBQRFA
F	ERROR NUMBER nnn IN DBQRFA	The internal error nnn occurred in the DBQRFA utility.	Follow site-defined procedures for reporting software errors or operational problems.	DBQRFA
N	FILE fn NOT AT CONTROL POINT	The file fn needed for processing is not at the control point.	Attach the required file and resubmit the job.	DBQRFA
N	FILE CANNOT BE RESTORED	The file that is specified in a previous message is in an unprocessable state.	The file can be reloaded from a file dump and recovered or restored from a journal log file.	DBQRFA
F	ILLEGAL NUMBER OF CONTROL CARD PARAMETERS	The DBQRFI control statement must contain three parameters.	Correct the control statement and resubmit the job.	DBQRFI
F	ILLEGAL PRU SIZE CONTROL CARD PARAMETER	The PRU size specified is not a positive decimal number.	Correct the control statement and resubmit the job.	DBQRFI

TABLE B-3. DBQRFA AND DBQRFI DIAGNOSTICS (Contd)

Type	Message	Significance	Action	Issued By
F	ILLEGAL SCHEMA ID CONTROL CARD PARAMETER	The schema identification must be the four-character identification assigned by the DBMSTRD utility.	Correct the control statement and resubmit the job.	DBQRFI
F	INCORRECT SCHEMA ID ON QRF HEADER	The schema identification on the quick recovery file header does not match the schema identification specified by the SC parameter.	Check that the correct quick recovery file has been attached. Correct the problem and resubmit the job.	DBQRFA
N	LAST RECOVERY POINT IS num	The last recovery point number is num. The data base has been restored to this point.	None.	DBQRFA
N	MESSAGE FOR AREA id VERSION vn	The following message or messages refer to the area an and version vn.	None.	DBQRFA
F	PROGRAM ABORTED **	The fatal errors noted in other diagnostics issued caused the program to abort.	Correct the errors noted and resubmit the job.	DBQRFA
F	QRF BLOCK CONTAINS INVALID AREA ID	The area identification contained in the quick recovery file block does not match the area identification contained in the master directory.	Check that the correct master directory has been attached. Correct the problem and resubmit the job.	DBQRFA
F	QRF BLOCK CONTAINS INVALID VERSION NAME	The version name specified in the quick recovery file block is not found in the master directory.	Check that information for the version name specified in the quick recovery file exists in the master directory.	DBQRFA
F	QRF BLOCK READ FOR UNDEFINED INDEX FILE	The quick recovery file contains a block for the index file of an area that does not have an index file.	Check that correct master directory has been attached. Any updates performed prior to the abort should be analyzed since files have been modified improperly and must be reloaded. Correct the problem and resubmit the job.	DBQRFA
N	QRF EMPTY	The quick recovery file does not contain any blocks. The data base is intact and can be processed.	None.	DBQRFA
F	REQUIRED FILES NOT AT CONTROL POINT	The files to be processed by the utility have not been attached and are not at the control point.	Check that correct master directory has been attached and that the files exist. Correct the problems and resubmit the job.	DBQRFA
N	num BLOCKS REWRITTEN	For the file that is specified in a previous message, the indicated number num of blocks were rewritten from the quick recovery file.	None.	DBQRFA
N	num INDEX BLOCKS REWRITTEN	For the file that is specified in a previous message, the indicated number num of index blocks were rewritten from the quick recovery file.	None.	DBQRFA

DBRCN AND DBRST DIAGNOSTICS

The diagnostic messages for the DBRCN and DBRST utilities are listed in table B-4. A destination column designates the dayfile or output file to which the message is written. These messages include the following types:

F Fatal error message. The DBRCN or DBRST utility run is aborted.

N Nonfatal error message. The DBRCN or DBRST utility continues processing.

TABLE B-4. DBRCN AND DBRST DIAGNOSTICS

Type	Message	Significance	Action	Destination
F	BAD PARAMETER = p	The parameter in the utility control statement is invalid.	Correct the parameter and resubmit the job.	Dayfile
F	CIO ERROR ON QRF	A CIO error occurred while reading from or writing to the quick recovery file.	Correct the CIO error and resubmit the job.	Dayfile
F	CRM ERROR ON OPEN OF OLD LOG FILE	A CRM error occurred while opening the input journal log file.	Refer to the applicable CRM error diagnostic and correct as noted in the CRM Basic Access Methods reference manual.	Dayfile Output file
F	CRM ERROR ON OPEN OF SELECT FILE	A CRM error occurred while opening the sorted select file produced by the utility.	Refer to the applicable CRM error diagnostic and correct as noted in the CRM Basic Access Methods reference manual.	Dayfile Output file
F	CRM ERROR ON SELECT FILE	A CRM error occurred while reading the sorted select file produced by the utility. The select file might be invalid if it is generated during a previous utility run that aborted.	Refer to the applicable CRM error diagnostic and correct as noted in the CRM Basic Access Methods reference manual.	Dayfile Output file
N	CRM ERROR nnn ON JOURNAL LOG FILE	The CRM error nnn occurred on the new log file fn produced by the utility.	Refer to the applicable CRM error diagnostic and correct as noted in the CRM Basic Access Methods reference manual.	Output file
N	CRM ERROR nnn ON OLD LOG FILE	The CRM error nnn occurred while processing the input journal log file.	Refer to the applicable CRM error diagnostic and correct as noted in the CRM Basic Access Methods reference manual.	Output file
F	CRM ERROR nnn ON SELECTION FILE	The CRM error nnn occurred while writing the sorted select file produced by the utility.	Refer to the applicable CRM error diagnostic and correct as noted in the CRM Basic Access Methods reference manual.	Dayfile Output file
N	CRM ERROR nnn ON PROCESSING fn	The CRM error nnn occurred while processing the data base file during the utility run.	Refer to the applicable CRM error diagnostic and correct as noted in the CRM Advanced Access Methods reference manual.	Output file
F	CRM ERROR nnn WHILE ACCESSING MASTER DIRECTORY	The CRM error nnn occurred while accessing the master directory.	Refer to the applicable CRM error diagnostic and correct as noted in the CRM Basic Access Methods reference manual.	Dayfile
N	ERR MSG WHICH FOLLOWS HAS TYPE CODE nn	The syntax error in the input file has the internal type code nn. This message appears if an internal error exists.	Follow site-defined procedures for reporting software errors and operational problems.	Output file

TABLE B-4. DBRCN AND DBRST DIAGNOSTICS (Contd)

Type	Message	Significance	Action	Destination
F	ERROR LIMIT EXCEEDED- num ERRORS	The number num of errors found in the syntax analysis exceeded the allowed limit.	Correct the errors noted in other diagnostics issued and resubmit the job.	Dayfile Output file
F	ERROR NUMBER nnn IN DB$RSEL	The error nnn occurred in the recovery module DB$RSEL.	Follow site-defined procedures for reporting software errors or operational problems.	Dayfile Output dayfile
F	ERROR - ILLEGAL RECORD TYPE ON SELECT FILE	A select file created by one utility has been used with the other utility; for example, a select file created by DBRST was used in a DBRCN run.	Correct the SF parameter on the utility control statement and resubmit the job.	Dayfile Output file
F	ERROR - INCORRECT RECORD LENGTH DATA IN LOG RECORD	The DBRCN or DBRST utility cannot compute the correct length of the image of the data base record because the information about the record length is erroneous in the log record.	Follow site-defined procedures for reporting software errors or operational problems.	Dayfile Output file
F	ERRORS DETECTED DURING SELECTION	Nonfatal errors were detected during selection; the data base was not updated.	Correct the problems on the select file and resubmit the job using the corrected select file.	Dayfile Output file
N	FILE fn NOT AT CONTROL POINT	The file fn specified in the utility control statement has not been attached and is not at the user control point.	Attach the file and resubmit the job.	Dayfile
F	INPUT ERROR(S) - PROGRAM ABORTED	The input parameters provided in the input file are in error. Processing cannot continue.	Correct the input parameters and resubmit the job.	Dayfile
F	INTERNAL DBMSTRD ERROR ILLEGAL CALL TO DB$MABT FROM numB	The master directory cannot be accessed because of an internal error. An illegal call was generated to DB$MABT from the address numB.	Follow site-defined procedures for reporting software errors or operation problems.	Dayfile Output file
N	INVALID RECORD LENGTH FOR LOG RECORD	The internal structure of the log file is incorrect.	Analyze the records in the log file to determine whether a logically correct update of the data base can be produced from this log file.	Output file
N	JOURNAL LOG CLOSED FOR SCHEMA sn	The journal log files have been closed for the schema sn.	None.	Dayfile
N	LOG FILES OPENED FOR SCHEMA sn	The journal log files have been opened for the schema sn.	None.	Dayfile
	NOTE - SELECTION PHASE SUPPRESSED. SELECT CLAUSE WILL HAVE NO EFFECT	The RCV parameter is specified on the utility control statement and a SELECT clause is present in the input file. The SELECT clause is ignored.	None.	Output file
N	PFN ERROR ec ATTACHING fn	The permanent file error ec occurred while attaching the file fn.	For a non-PUBLIC file unavailable on NOS, ensure that a PERMIT control statement specifying the user name under which DBRCN or DBRST executes is in effect. Refer to the applicable permanent file error diagnostic in volume 4 of the NOS reference set, or in the description of the FDB macro in the NOS/BE reference manual.	Dayfile

TABLE B-4. DBRCN AND DBRST DIAGNOSTICS (Contd)

Type	Message	Significance	Action	Destination
F	PROGRAM ABORTED**	The fatal errors noted in other diagnostics issued caused the program to abort.	Correct the errors noted and resubmit the job.	Dayfil Output file
F	QRF IS NOT A PERMANENT FILE	The quick recovery file is not cataloged as a permanent file.	Initialize a quick recovery file using the DBQRFI utility. It must be a cataloged random permanent file.	Dayfile
F	QRF IS NOT EMPTY INITIALLY	The quick recovery file is not empty. The DBQRFA utility must be run before DBRCN or DBRST can execute.	Run the DBQRFA utility; if the data base has been reloaded, run DBQRFI to initialize the quick recovery file.	Dayfile
F	RECOVERY ABORT, INVALID MASTER DIRECTORY	The file specified or implied by the MD parameter is not acceptable as a valid master directory. The DBRCN/DBRST run is aborted.	Check that the file specified in the MD parameter is for a valid master directory.	Dayfile
F	RECOVERY ABORT - WRITE ERROR	An error occurred while writing a record to the journal log file.	Correct the error and resubmit the job.	Dayfile
N	RECOVERY POINT num NOT FOUND	The recovery point number num specified in the input parameter is not found on the journal log file.	Check the journal log file for the correct recovery point number. Correct the input file and resubmit the job.	Dayfile Output file
F	RECOVER/RESTORE ABORT, JOB REPRIEVED	The DBRCN or DBRST utility run aborted while modifying the data base; the particular utility was reprieved by the system.	Correct the errors noted with the other diagnostics and resubmit the job.	Dayfile Output file
F	REQUIRED FILE(S) MISSING, PROGRAM ABORTED	The input files needed for the recovery or restore job are not present at the control point. The job is aborted.	Attach the required files and rerun the job.	Dayfile
F	SCHEMA ID MISMATCH ON QRF FOR SCHEMA	The identification for the schema found in the quick recovery file does not match the schema identification assigned by the DBMSTRD utility.	Verify that the correct quick recovery file has been attached.	Dayfile
F	SCHEMA NAME/ID ON LOGFILE DOES NOT MATCH MD	The schema name and identification found in the journal log file records do not match the schema name and identification in the master directory.	Attach the same master directory that was used when the journal log file was created and specify its local file name in the utility control statement.	Dayfile
N	SELECTION ERRORS, UPDATE PHASE SUPPRESSED	Errors occurred when the log records were selected from the journal log file. The data base was not recovered or restored.	If the SF parameter was specified on the utility control statement, the select file is saved and can be used when the job is rerun (the RCV parameter must be specified on the rerun).	Dayfile Output file
N	TIME yyddd hh.mm.ss NOT FOUND BEFORE EOI ****	The data and time specification in the input parameter is not found on the journal log file.	Check the date and time specification. Correct the error and resubmit the job.	Dayfile Output file

TABLE B-4. DBRCN AND DBRST DIAGNOSTICS (Contd)

Type	Message	Significance	Action	Destination
F	VARIABLE RANGE ERR/TBL-tn	This is an internal error.	Follow site-defined procedures for reporting software errors or operational problems.	Dayfile Output file
F	VERSION ON SELECT FILE DOES NOT MATCH MD	The version name specified does not match the corresponding version name specified in the master directory.	Check that information for the version name specified in the select file exists in the master directory.	Output file
N	101 - sn IS AN INVALID SCHEMA NAME	The schema name sn supplied in the input parameter must be 1 to 30 characters in length.	Correct the schema name in the input parameter and resubmit the job.	Output file
N	102 - MISSING PERIOD	A period does not terminate the input parameters in the input file.	Correct the input file and resubmit the job.	Output file
N	106 - an IS AN INVALID AREA NAME	The area name an supplied in the parameter must be 1 to 30 characters in length.	Correct the area name in the input file and resubmit the job.	Output file
N	114 - KEYWORD - SCHEMA - IS MISSING	The keyword SCHEMA is not the first word in the input file.	Correct the input parameter and resubmit the job.	Output file
F	115 - UNABLE TO FIND SCHEMA sn IN MD	Information for schema sn is not found in the master directory.	Check the schema name for correct spelling; correct the input parameter and resubmit the job. Check that the correct master directory is in use.	Output file
F	141 - UNABLE TO LOCATE AREA an IN THE SCHEMA DIRECTORY	The area name an specified in the input file is not in the schema directory.	Check the spelling of area name, or check that the correct schema is in use. Correct the input parameter and resubmit the job.	Output file
N	200 - NO INPUT FILE	No input file has been specified for the recovery run.	Resubmit the job with the input file included.	Output file
N	201 - INVALID OR MISSING KEYWORD - kw	The keyword kw in the input file is invalid or missing.	Correct the input parameter and resubmit the job.	Output file
N	202 - un IS AN INVALID JOBNAME	The jobname jn specified in the input file is not a valid jobname.	Correct the input parameter and resubmit the job.	Output file
N	203 - id IS AN INVALID RUN-UNIT-ID	The run-unit identification id specified in the input file is not valid.	Correct the input parameter and resubmit the job.	Output file
N	204 - t IS AN INVALID TIME FIELD	The data and time specification t in the input file is not valid.	Correct the input parameter and resubmit the job.	Output file
N	205 - num IS AN INVALID RECOVERY POINT NUMBER	The recovery point number num in the input file is not valid.	Correct the input parameter and resubmit the job.	Output file
F	207 - DUPLICATE KEYWORD - kw	The keyword kw has already been specified once in the input file.	Correct the input parameter and resubmit the job.	Output file
F	209 - BEGIN/END TIMES OUT OF ORDER	The BEGIN/END times specified in the input file are not in order.	Correct the order of the input parameters and resubmit the file.	Output file

DBREC UTILITY DIAGNOSTICS

The diagnostic messages issued by the DBREC utility are listed in table B-5. The messages are of the following types:

F Fatal error message. DBREC execution is aborted. If the message refers to a syntax error on the input file, the DBREC run does not abort until the rest of the input syntax has been checked.

N Nonfatal error message. Processing continues.

I Informative message. Processing continues.

If a fatal message causes DBREC to abort, the following message is written to the output file and or the dayfile:

**** DBREC ABORT ****

Both messages of the type fatal and nonfatal and messages with error codes between 500 and 599 are written to the output file and the dayfile. All other messages are written to the output file.

TABLE B-5. DBREC DIAGNOSTICS

Type	Message Code	Message	Significance	Action
I		ALLOCATE DIRECTIVE EXECUTION COMPLETE	The DBREC ALLOCATE directive has completed execution, either successfully or unsuccessfully.	None.
F		BAD PARAMETER = p	The parameter p in the DBREC control statement is invalid.	Correct the parameter and resubmit the job.
I		BEGIN PROCESSING DIRECTIVES FOR SCHEMA sn	Directives for schema sn are being processed.	None.
I		CDCS DETECTED A FATAL ERROR ON JOURNAL LOG FILE fn	CDCS has detected an error on journal log file fn. DBREC attempts to dump as much of the journal log file as possible, assuming DBREC also detects an error on the file. The journal log file is not reinitialized even if no error occurred while dumping the journal log file to tape.	None.
I		CHARACTERISTICS OF THE LAST LOG RECORD DUMPED - USER ID OF RECORD id PROGRAM ID OF RECORD id TYPE OF RECORD type DATE OF ENTRY date TIME OF ENTRY time	Either an error occurred while dumping the journal log file to tape or CDCS has detected an error on the journal log file. Information about the last record written to the tape file is indicated.	None.
F		CRM ERROR nnn WHILE ACCESSING MASTER DIRECTORY	The CRM error nnn occurred while reading the master directory. DBREC is aborted.	Refer to the applicable CRM error diagnostic Methods in the Basic Access Methods reference manual. Correct the error and resubmit the job.
F		DBREC INTERNAL ERROR -- mdnl	A DBREC internal error occurred in module mdn; the first eight characters are the module name and the last character designates the location l in the module where the error occurred. DBREC is aborted.	Follow site-defined procedures for reporting software errors or operational problems.

TABLE B-5. DBREC DIAGNOSTICS (Contd)

Type	Message Code	Message	Significance	Action
F		DBREC INTERNAL ERROR VARIABLE RANGE ERR/TBL ERR - TN	An internal error occurred while checking the DBREC input directives. DBREC is aborted.	Follow site-defined procedures for reporting software errors or operational problems.
I		DUMP DIRECTIVE EXECUTION COMPLETE	The DBREC DUMP directive has completed execution, either successfully or unsuccessfully.	None.
F		FILE MSTRDIR NOT AT CONTROL POINT -- RUN ABORTED	The master directory was not attached with the local file name of MSTRDIR. DBREC is aborted.	Change the ATTACH statement and rerun the job.
I		JOURNAL LOG FILE fn HAS BEEN ALLOCATED	The journal log file fn has been successfully allocated.	None.
I		JOURNAL LOG FILE fn HAS BEEN DUMPED TO TAPE	The journal log file fn has been successfully dumped to tape.	None.
I		JOURNAL LOG FILE HAS BEEN REINITIALIZED	The journal log file has been dumped to tape and successfully reinitialized.	None.
I		QUICK RECOVERY FILE fn HAS BEEN ALLOCATED	The quick recovery file has been successfully allocated.	None.
I		RESTART IDENTIFIER FILE fn HAS BEEN ALLOCATED	The restart identifier file has been successfully allocated.	None.
I		TRANSACTION RECOVERY FILE fn HAS BEEN ALLOCATED	The transaction recovery file has been successfully allocated.	None.
F	101	sn IS AN INVALID SCHEMA NAME	The schema name sn is not syntactically correct.	Check requirements for schema names. Correct the input file and resubmit the job.
F	115	UNABLE TO FIND SCHEMA sn IN MD	The schema sn is not in the master directory.	Check the schema name for correct spelling. Correct the input file and resubmit the job.
F	122	pfn IS AN INVALID PERMANENT FILE NAME OR IS INAPPROPRIATE IN THE CONTEXT USED	The permanent file name pfn is not syntactically correct.	Correct the input file and resubmit the job.
F	300	KEYWORD -JOURNAL- IS MISSING	The keyword JOURNAL has not been specified in the DUMP clause.	Correct the input file and resubmit the job.
F	301	SCHEMA sn WAS PREVIOUSLY SPECIFIED IN A SCHEMA CLAUSE	A particular schema name can appear in only one SCHEMA clause in the source input.	Correct the input file and resubmit the job.
F	302	LOG FILE NAME MUST BE 7 CHARACTERS LONG	The permanent file name of the journal log file must be exactly 7 characters long.	Correct the input file and resubmit the job.
F	303	KEYWORD -SIZE- IS MISSING	The keyword SIZE has not been specified in the ALLOCATE clause for either the journal log file or the quick recovery file.	Correct the input file and resubmit the job.

TABLE B-5. DBREC DIAGNOSTICS (Contd)

Type	Message Code	Message	Significance	Action
F	304	num IS AN INVALID SIZE	The size num must be an integer of 1 through 1073741823.	Correct the input file and resubmit the job.
F	305	AT LEAST ONE FILE MUST BE SPECIFIED IN THE ALLOCATE ENTRY	No file option was included in the ALLOCATE clause.	Correct the input file and resubmit the job.
F	306	JOURNAL LOG FILE WAS PREVIOUSLY SPECIFIED	The JOURNAL LOG FILE option can be specified only once in an ALLOCATE clause.	Correct the input file and resubmit the job.
F	307	NO TRANSACTION RECOVERY FILE EXISTS FOR THE CURRENT SCHEMA	A TRANSACTION RECOVERY FILE option has been specified for a schema for which no transaction recovery file is specified in the master directory.	Correct the input file and resubmit the job.
F	308	INVALID TRANSACTION RECOVERY NAME	The first 6 characters of the transaction recovery file name do not match the permanent file name of the transaction recovery file specified in the master directory.	Check the transaction recovery file name for correct spelling. Correct the input file and resubmit the job.
F	309	NO QUICK RECOVERY FILE EXISTS FOR THE CURRENT SCHEMA	A QUICK RECOVERY FILE option has been specified for a schema for which no quick recovery file is specified in the master directory.	Correct the input file and resubmit the job.
F	310	INVALID QUICK RECOVERY FILE NAME	The quick recovery file name must be the same as the quick recovery file name specified in the master directory.	Check the quick recovery file name for correct spelling. Correct the input file and resubmit the job.
F	311	NO RESTART IDENTIFIER FILE EXISTS FOR THE CURRENT SCHEMA	A RESTART IDENTIFIER FILE option has been specified for a schema for which no restart identifier file is specified in the master directory.	Correct the input file and resubmit the job.
F	312	INVALID RESTART IDENTIFIER FILE NAME	The restart identifier file name must be the same as the restart identifier file name specified in the master directory.	Check the restart identifier file name for correct spelling. Correct the input file and resubmit the job.
F	313	NO JOURNAL LOG FILES EXIST FOR THE CURRENT SCHEMA	A JOURNAL LOG FILE option has been specified for a schema for which no journal log files have been specified in the master directory.	Correct the input file and resubmit the job.
F	314	INVALID JOURNAL LOG FILE NAME	The first 6 characters of the journal log file name do not match the permanent file name of the journal log file in the master directory.	Check the journal log file name for correct spelling. Correct the input file and resubmit the job.
F	315	LAST CHARACTER IN JOURNAL LOG FILE NAME MUST BE 1 OR 2	The seventh character of the permanent file name of the journal log file is not a 1 or a 2.	Correct the input file and resubmit the job.

TABLE B-5. DBREC DIAGNOSTICS (Contd)

Type	Message Code	Message	Significance	Action
F	316	TRANSACTION RECOVERY FILE WAS PREVIOUSLY SPECIFIED	The TRANSACTION RECOVERY option can be specified only once in an ALLOCATE clause.	Correct the input file and resubmit the job.
F	317	QUICK RECOVERY FILE WAS PREVIOUSLY SPECIFIED	The QUICK RECOVERY FILE option can be specified only once in an ALLOCATE clause.	Correct the input file and resubmit the job.
F	318	RESTART IDENTIFIER FILE WAS PREVIOUSLY SPECIFIED	The RESTART IDENTIFIER FILE option can be specified only once in an ALLOCATE clause.	Correct the input file and resubmit the job.
F	319	EMPTY INPUT FILE	A problem exists with the input file. The file might not have been rewound.	Resubmit the job with the input file included.
F	320	ILLEGAL FIRST WORD ON INPUT	The first word on the input file must be SCHEMA.	Correct the input file and resubmit the job.
F	321	NO DIRECTIVES WERE VALIDATED FOR SCHEMA sn	The SCHEMA clause must be followed by at least one clause: either a DUMP or an ALLOCATE clause.	Correct the input file and resubmit the job.
F	322	wd IS AN INVALID SOURCE WORD OR INAPROPRIATE IN THE CONTEXT USED	The word wd is not a keyword that is allowed in the context in which it is used.	Check spelling of word wd or the requirements of the clause in which the word wd appears. Correct the input file and resubmit the job.
F	323	TRANSACTION RECOVERY FILE NAME MUST BE 7 CHARACTERS LONG	The permanent file name of the transaction recovery file must be exactly 7 characters long.	Correct the input file and resubmit the job.
N	502	CIO ERROR nnn OCCURRED DURING REWIND OF FILE fn	The CIO error nnn occurred during the rewind of file fn.	Refer to the appropriate operating system reference manual for a description of the CIO error codes.
N	503	CIO ERROR nnn OCCURRED WHILE WRITING TO FILE fn	The CIO error nnn occurred while writing to file fn.	Refer to the appropriate operating system reference manual for a description of the CIO error codes.
N	504	PFM ERROR ec OCCURRED WHILE EXTENDING FILE fn	The permanent file manager (PFM) error ec occurred while attempting to extend the NOS/BE permanent file fn.	Refer to the NOS/BE reference manual for a description of the FDB macro and the associated list of PFM errors.
N	505	PFM ERR ec OCCURRED WHILE ATTACHING FILE fn	The permanent file manager (PFM) error ec occurred while attaching file fn.	For a non-PUBLIC file unavailable on NOS, ensure that a PERMIT control statement specifying the user name under which DBREC executes is in effect. Refer to the applicable permanent file error diagnostic in volume 4 of the NOS reference set, or in the description of the FDB macro in the NOS/BE reference manual.

TABLE B-5. DBREC DIAGNOSTICS (Contd)

Type	Message Code	Message	Significance	Action
N	506	CRM ERROR nnn OCCURRED DURING THE OPEN OF FILE fn	The CRM error nnn occurred during the open of file fn.	Refer to the applicable CRM reference manual for a description of the error and for action needed to correct the error.
N	507	CRM ERROR nnn OCCURRED WHILE WRITING TO FILE fn	The indicated CRM error nnn occurred while executing a CRM PUT to file fn.	Refer to the applicable CRM reference manual for a description of the error and for action needed to correct the error.
N	508	CRM ERROR nnn OCCURRED DURING THE CLOSE OF FILE fn	The CRM error nnn occurred during the close of file fn.	Refer to the applicable CRM reference manual for a description of the error and for action needed to correct the error.
N	509	CIO ERROR nnn OCCURRED WHILE READING FILE fn	The CIO error nnn occurred while reading file fn.	Refer to the appropriate operating system reference manual for a description of the CIO error codes.
N	510	JOURNAL LOG FILE fn CAN NOT BE DUMPED -- STATUS IS *INACTIVE*	The journal log file fn has not been used by CDCS, and therefore the log file contains only data placed there by DBREC during processing to allocate the file.	None.
N	511	NO MD JOB INFO SPECIFIED FOR SCHEMA sn	The JOB CONTROL INFORMATION clause is not specified in the master directory for schema sn. The job control information must exist for a schema in the master directory before the DUMP clause can be used.	Add the job control information to the master directory and rerun the DBREC dump job.
N	512	ERROR nnn OCCURRED ON THE REQUEST OF THE JOURNAL LOG FILE DUMP FILE	The indicated error nnn occurred during the NOS/BE request to dump the logdump tape file.	Refer to the description of the REQUEST macro in the NOS/BE reference manual for a list of error codes.
N	513	MANAGED MEMORY OVERFLOW -- DBREC ABORTED	DBREC is aborted because there is not enough memory to continue execution.	Increase the field length of DBREC and rerun the job.
	514	ERROR nnn OCCURRED ON THE LABEL OF JOURNAL LOG FILE DUMP FILE	The indicated error occurred during a NOS label of the logdump tape file.	Refer to the description of the local file manager error processing in the NOS reference manual for a list of error codes.

DDL SCHEMA COMPILATION AND EXHIBIT DIAGNOSTICS

The diagnostic messages that can be issued either during compilation of a schema source program or during execution of the EXHIBIT utility are listed in table B-6. These messages are written to the source listing where each message is preceded by the error code enclosed in asterisks. The messages are of the following types:

C Catastrophic error message. Compilation is terminated due to an error in the source program syntax.

E Error message. Compilation is not necessarily terminated although an error has been discovered in the source program syntax.

T Trivial error message. Compilation is not terminated although an error has been discovered in the source program syntax.

TABLE B-6. DDL SCHEMA DIAGNOSTICS

Error Code	Type	Message	Significance	Action
100	C	SCHEMA DECLARATION IS INCORRECT	The schema name entry, which must be the first statement in the DDL schema input file, is incorrect.	Check the rules for structuring the schema identification entry, and recompile.
101	C	EMPTY INPUT FILE	There is no information in the DDL schema input file.	Check the input file, and recompile.
102	C	INVALID SCHEMA NAME	The schema name can contain only alphabetic and numeric characters and embedded hyphens.	Correct the schema name, and recompile.
105	E	DATA CONTROL CLAUSE INCOMPLETE	The DATA CONTROL clause must contain the words DATA CONTROL, followed by a period.	Correct the format of the DATA CONTROL clause, and recompile.
106	E	INVALID RECORD NAME	A record name can contain 1 to 30 alphabetic and numeric characters and hyphens.	Correct the record name, and recompile.
107	C	PROCEDURE NAME TABLE OVERFLOW, **DDL ABORTED**	The number of data base procedures that can be specified is limited by the number of entries allowed in the procedure name table. When there is no more space for entries, DDL aborts. A maximum of 600 entries is allowed in the schema.	Reduce the number of data base procedures and recompile.
108	E	PERIOD MISSING	A period was expected and was not found.	Insert a period in the appropriate place, and recompile.
111	E	AREA UNSPECIFIED OR UNRECOGNIZABLE KEYWORD	In the data control entry, an area control entry was not found when one was expected.	Check the conditions under which an area control entry is expected, and make the appropriate corrections.
112	E	AREA NOT UNIQUE	The first seven characters of the area name must be unique among all area names.	Make the area name unique, and recompile.
114	E	INCOMPLETE PROCEDURE CLAUSE	The clause referencing a data base procedure is not complete.	Check the format of the clause in error, make necessary corrections, and recompile.

TABLE B-6. DDL SCHEMA DIAGNOSTICS (Contd)

Error Code	Type	Message	Significance	Action
116	E	LITERAL VALUES NOT UNIQUE	The literal values in a RECORD CODE clause must be different for each record name.	Correct the literal values in the record code entry, and recompile.
119	E	PROCEDURE UNSPECIFIED	The CALL clause does not specify a data base procedure.	Correct the format of the CALL clause, and recompile.
120	E	RECORD NAME NOT UNIQUE	The record name must be unique among all record names in the schema.	Change the record name, and recompile.
121	E	LAST ITEM IN RECORD MUST BE AN ELEMENTARY ITEM	The last item in a record description entry must be an elementary item, not a repeating group.	Restructure the record description entry, and recompile.
122	E	VARIABLE DIMENSION GROUP MUST BE THE LAST ITEM ENTRY IN A RECORD	A variable dimension group can be followed only by items subordinate to it.	Make the variable dimension group the last item entry in a record, and recompile.
126	E	INVALID AREA IN WITHIN CLAUSE	The area name in the WITHIN clause is not a legal area name.	Correct the WITHIN clause, and recompile.
127	E	AREA UNDEFINED IN WITHIN CLAUSE	The area named in the WITHIN clause has not been previously defined in the schema.	Define the area name in an area description entry, and recompile.
129	E	INVALID AREA NAME	The area name can contain 1 to 30 alphabetic and numeric characters; embedded hyphens can occur after the first seven characters. The first character must be alphabetic.	Correct the area name, and recompile.
133	E	WORD ADDRESSABLE FILE ORGANIZATION ILLEGAL	File organization must be extended indexed sequential, extended direct access, or extended actual key.	Recompile the schema, omitting any files with word addressable file organization.
136	E	LEVEL NUMBER NOT GREATER THAN 0 AND NOT LESS THAN 100	Level numbers in data description entries must be in the range 1 through 99.	Correct the level number, and recompile.
138	E	TYPE CLAUSE OVERRIDES THE PICTURE SPECIFICATON FOR THIS ITEM ENTRY	When both TYPE and PICTURE clauses are specified in a data description entry, the TYPE clause takes precedence; the PICTURE clause is ignored.	Take no action; or omit the TYPE clause so that the PICTURE clause takes precedence. Refer to the discussion of TYPE clause to determine which clause is preferable, and recompile.
139		ALTERNATE KEY SPECIFIED FOR SEQUENTIAL FILE	An alternate key cannot be specified for a sequential file. Sequential file organization is not supported.	Change the file organization to indexed sequential.

TABLE B-6. DDL SCHEMA DIAGNOSTICS (Contd)

Error Code	Type	Message	Significance	Action
140	E	OCCURS CLAUSE INVALID ON A RESULT CLAUSE	An OCCURS clause cannot be used in the same data description entry as a RESULT clause.	Omit either the OCCURS clause or the RESULT clause, and recompile.
141	E	MULTIPLE RESULT CLAUSES	Only one RESULT clause can be specified for a data item.	Omit all but one valid RESULT clause, and recompile.
143	E	PROCEDURE UNSPECIFIED FOR RESULT CLAUSE	A data base procedure has not been specified in the RESULT clause.	Include the appropriate data base procedure in the RESULT clause, and recompile.
144	E	INVALID PROCEDURE NAME - NAME TRUNCATED	A data base procedure name must be seven characters or less; longer names are truncated to seven characters.	Make the necessary correction to the data base procedure name, and recompile.
145	E	INVALID DATANAME FOR KEY	The data name for a key must be 1 to 30 alphabetic and numeric characters and hyphens.	Make the necessary correction to the data name, and recompile.
146	E	INVALID PROCEDURE NAME	The data base procedure name must be one to seven alphabetic and numeric characters; the first character must be alphabetic.	Make the necessary correction to the data base procedure name, and recompile.
147	E	USING CLAUSE VALID ONLY FOR DA FILE, PRIMARY KEY	In a KEY clause where ALTERNATE is not specified, the USING option can be specified only for a direct access file.	Make the necessary corrections to the key clause/file organization for the use of the USAGE option, and recompile.
148	E	PRIMARY KEY DEFINED FOR AK FILE MUST BE OF TYPE FIXED INT-1 (NO INT-2)	The TYPE clause specified for a primary key defined for an actual key file must have the FIXED option. An integer-2 specification to designate an assumed decimal point must not be specified.	Correct the TYPE clause format for the key item, and recompile.
149	E	ALTERNATE KEY CANNOT BE SPECIFIED BEFORE THE PRIMARY KEY	Alternate keys have been specified before the primary key definition. Alternate keys must be specified after the primary key definition.	Correct the order of key specification, and recompile.
150	E	KEY LENGTH CANNOT BE GREATER THAN 240 CHARACTERS	The maximum length allowed for a data item designated as a key cannot exceed 240 characters.	Correct the item size or key specification, and recompile.
151	E	INVALID PROCEDURE-NAME IN CHECK CLAUSE - NAME TRUNCATED	The data base procedure name in the CHECK clause must be one to seven characters; longer names are truncated to seven characters.	Change the data base procedure name, and recompile.
152	T	DUPLICATES NOT ALLOWED ON PRIMARY KEYS - DEFAULTED TO NOT ALLOWED	In the KEY clause, duplicates are not allowed for a primary key.	Remove the duplicate key specifications or correct the alternate key designation, and recompile.

TABLE B-6. DDL SCHEMA DIAGNOSTICS (Contd)

Error Code	Type	Message	Significance	Action
153	E	DUPLICATES WITH FIRST OR INDEXED VALID ONLY FOR ALTERNATE KEY	In the KEY clause, duplicates can be specified with the FIRST or INDEXED option only for an alternate key.	Remove the duplicate key specifications or correct the alternate key designation, and recompile.
154	E	1ST LITERAL RANGE OPTION INVALID	In the CHECK clause, the first literal specified for value is not a valid literal.	Correct the literal in the CHECK clause, and recompile.
156	E	2ND LITERAL RANGE OPTION INVALID	In the CHECK clause, the literal following THRU in a value specification is not a valid literal.	Correct the literal in the CHECK clause, and recompile.
157	E	CHECK OPTION UNSPECIFIED	The CHECK clause must specify either the VALUE, PICTURE, or data base procedure option.	Change the CHECK clause to include either the VALUE, PICTURE, or data base procedure option, and recompile.
160	E	ONLY VALID SEQUENCES ARE ASCII, COBOL OR DISPLAY	The SEQUENCE clause must specify ASCII, COBOL, or DISPLAY for the collating sequence.	Correct the SEQUENCE clause to specify ASCII, COBOL, or DISPLAY for the collating sequence.
161	E	NO VALID WITHIN CLAUSE SPECIFIED IN THE CURRENT RECORD	The record description entry being processed does not contain the required WITHIN clause or errors exist in the WITHIN clause.	Add a WITHIN clause to the record description entry or correct the existing WITHIN clause, and recompile.
162	E	CALL UNSPECIFIED IN ENCODING	The ENCODING clause must include the word CALL.	Correct the ENCODING clause to include the word CALL, and reinput.
163	E	UNSPECIFIED ENCODING PROCEDURE	The ENCODING clause must specify a data base procedure.	Correct the ENCODING clause to include a data base procedure, and recompile.
164	E	INVALID ENCODING PROCEDURE NAME - NAME TRUNCATED	The data base procedure name contains more than seven characters; the name is truncated to seven characters.	No action; or change the data base procedure name, and recompile.
165	E	RESULT NOT SPECIFIED IN ACTUAL/VIRTUAL CLAUSE	The ACTUAL or VIRTUAL RESULT clause must include the word RESULT.	Add the word RESULT to the RESULT clause, and recompile.
166	E	VIRTUAL RESULT CANNOT BE SPECIFIED WITH EITHER ENCODING/DECODING OR CALL CLAUSES	When the VIRTUAL option is used to describe a data item, neither the ENCODING/DECODING clause nor the CALL clause can be used.	Eliminate the ENCODING, DECODING, or CALL clause from the data description entry, and recompile.
170	E	NAME GREATER THAN 30 CHARACTERS - NAME TRUNCATED	The specified name (record, area, or data item) contains more than 30 characters; the name is truncated to seven characters.	Correct the name, and recompile.
171	E	CALL PROCEDURE-NAME UNSPECIFIED	The CALL clause must specify a data base procedure.	Correct the format of the CALL clause, and recompile.

TABLE B-6. DDL SCHEMA DIAGNOSTICS (Contd)

Error Code	Type	Message	Significance	Action
172	E	RECORD CODE CLAUSE INCOMPLETE	The two words RECORD CODE must be specified for the RECORD CODE clause.	Correct the format of the RECORD CODE clause, and recompile.
173	E	INVALID OR NON-EXISTENT LEVEL NUMBER	The level number specified violates one of the following rules: 1) it must be an integer in the range 1 through 99; 2) it must be greater than or equal to the level number of the first item in the record; 3) if less than the level number of the preceding item, the level number must have been specified for a previous item in the record.	Correct the level number, and recompile.
176	E	INVALID NAME OR INTEGER	A data name or an integer value must be specified in the OCCURS clause.	Correct the format of the OCCURS clause, and recompile.
187	E	MISSING BEFORE, AFTER, ON ERROR DURING OPTION IN CALL STATEMENT	The CALL clause must specify either the BEFORE, AFTER, or ON ERROR DURING option.	Correct the format of the CALL clause, and recompile.
191	E	BAD INTEGER SUBSCRIPT	Subscripts are not valid in DDL clauses for the schema.	Remove all the subscripts, and recompile.
192	E	SEARCHING FOR A PERIOD	The system requires a period to terminate an entry.	Insert a period in the appropriate place, and recompile.
194	E	INVALID RECORD QUALIFICATION	The qualifier following OF or IN cannot exceed 30 alphabetic and numeric characters and hyphens; the first character must be alphabetic.	Correct the qualifier, and recompile.
195	E	PERIOD FOUND, SCAN RESUMES	An error has occurred and caused DDL to skip over all information up to the next period. The period has been found and normal syntax analysis begins again.	Correct the previous error, and recompile.
196	E	INVALID DATANAME AFTER BY	The data name following BY in the RECORD CODE clause must be 1 through 30 alphabetic and numeric characters and hyphens; the first character must be alphabetic.	Correct the data name in the RECORD CODE clause, and recompile.
197	E	UNDEFINED DATANAME AFTER BY	The data name following BY in the RECORD CODE clause has not been previously defined as a data item in the schema. Qualification by the record name might be required to define the data item.	Define the data name in the schema, and recompile.

TABLE B-6. DDL SCHEMA DIAGNOSTICS (Contd)

Error Code	Type	Message	Significance	Action
198	E	DATANAME FOLLOWING BY IS OCCURRING ITEM	The data name following BY in the RECORD CODE clause must be an elementary item.	Change the data name to an elementary item in the RECORD CODE clause, and recompile.
199	E	VALUE MUST BE SPECIFIED FOR RECORD CODE	In the RECORD CODE clause, the word VALUE must follow the BY data name or PROCEDURE procedure name option.	Correct the format of the RECORD CODE clause, and recompile.
200	E	kkkkkk INVALID IN FOLLOWING CLAUSE, SEARCHING FOR PERIOD	The specified word or literal is invalid; normally, a diagnostic precedes this message to indicate what is invalid. All further source input is skipped until a period is found. If the character string kkkkkk is blank, a required terminating period has probably been omitted.	Correct the invalid word or literal, or insert the required period, and recompile.
201	E	PICTURE SPECIFICATION MUST BE ENCLOSED IN QUOTES	Quotation marks must surround the picture specification character string.	Enclose the picture specification character string in quotes, and recompile.
202	E	RECORD NOT CONTAINED IN THIS AREA	The record name specified in the RECORD CODE clause is not contained within the area for which the record code is being processed.	Change the record name to a valid one in the RECORD CODE clause, and recompile.
203	E	INVALID LITERAL IN VALUE CLAUSE	A literal (either numeric or nonnumeric) must follow record name IS in the RECORD CODE clause.	Correct the format of the RECORD CODE clause, and recompile.
204	E	NOT ENOUGH ROOM FOR THIS LITERAL IN INTERNAL TABLE	The internal table used to store literals for the RECORD CODE clause is full.	Use fewer record types or a shorter record code data name.
205	E	-BY- OR -PROCEDURE- MUST FOLLOW RECORD CODE	The keywords RECORD CODE must be followed by either BY or PROCEDURE.	Correct the format of the RECORD CODE clause, and recompile.
206	E	INTEGER MUST BE SPECIFIED FOR PROCEDURE VALUE	An integer value must be specified for the PROCEDURE option in the RECORD CODE clause.	Correct the format of the RECORD CODE clause, and recompile.
207	E	VALID RANGE FOR INTEGER VALUE IS 1 TO 1023	The integer specified in the PROCEDURE option of the RECORD CODE clause must be in the range 1 through 1023.	Correct the value of the integer in the PROCEDURE option of the RECORD CODE clause, and recompile.
208	E	NON-EMBEDDED DATANAME ILLEGAL FOR THIS FILE TYPE	A key that is not defined as a data name in a record description entry for this area is illegal for direct access files.	Define the key as a data name in the record description entry, and recompile.

TABLE B-6. DDL SCHEMA DIAGNOSTICS (Contd)

Error Code	Type	Message	Significance	Action
209	E	INTEGER VALUES NOT UNIQUE FOR PROC	The integer values specified in the PROCEDURE option of the RECORD CODE clause must be unique.	Correct the integer values specified in the PROCEDURE option of the RECORD CODE clause, making them unique, and recompile.
210	E	FILE CARD FOR AREA MISSING	A FILE control statement has not been specified for this area.	Include a FILE control statement, and recompile.
211	E	NO RECORD CODE SPECIFIED FOR MULTIPLE RECORD TYPES	The area has more than one record description entry, and no RECORD CODE clause has been specified.	Include a RECORD CODE clause, and recompile.
212	E	DATANAME FOR KEY NOT DEFINED IN RECORD WITHIN THIS AREA	The data name defined to be the key has not been specified in a record description entry for this area.	Include a record description entry for the data name, and recompile.
213	E	RECORD CODE ALREADY SPECIFIED FOR THIS AREA	A RECORD CODE clause has already been specified for this area.	Delete the repetitive RECORD CODE clause, and recompile.
214	E	DATANAME NOT DEFINED IN ANY OF THE SPECIFIED RECORDS	The data name following BY in the RECORD CODE clause is not defined in any of the records for which values are listed.	Correct the format of the RECORD CODE clause so that records in which the data name is defined are included in the VALUE FOR portion of the clause.
215	E	ONE AND ONLY ONE PRIMARY KEY MUST BE SPECIFIED	An area can have only one primary key in the KEY clause.	Correct the KEY clause, and recompile.
216	E	DATA CONTROL ALREADY SPECIFIED FOR THIS AREA	Duplicate data control entries cannot be specified for the area.	Eliminate the duplicate data control entry, and recompile.
217	E	RECORD CODE DATANAME SIZE MUST BE LESS THAN 240 CHARS	The literal in the RECORD CODE clause cannot exceed 240 characters.	Correct the literal in the RECORD CODE clause, and recompile.
218	T	MNR VALUE INCORRECT ON FILE CARD FOR THE ABOVE AREA - DEFAULT VALUE USED	The value of the MNR (minimum record length) parameter used in the FILE control statement is inadequate. DDL has substituted a default value.	No action, or change the value of the MNR parameter on the FILE control statement.
219	E	KEY ITEM CANNOT BE TYPE COMPLEX	A data item defined as a key cannot be described as TYPE COMPLEX.	Correct the KEY clause or the data item, and recompile.
220	T	XN VALUE MISSING ON FILE CARD FOR ALTERNATE KEYS - DEFAULT VALUE USED	The XN (index file name) parameter was not specified on a FILE control statement for an area with alternate keys defined. DDL has assigned a default value.	No action, or specify an XN parameter on the FILE control statement.

TABLE B-6. DDL SCHEMA DIAGNOSTICS (Contd)

Error Code	Type	Message	Significance	Action
221	T	ALTERNATE KEYS ARE NOT SPECIFIED - XN VALUE ON FILE CARD IS IGNORED	The XN (index file name) parameter was specified on a FILE control statement for an area with no alternate keys. The XN value is ignored.	No action, or eliminate the XN parameter from the FILE control statement.
222	E	NO AREA SPECIFIED IN SCHEMA	No area description entry has been specified in the schema.	Include an area description entry in the schema for each area in the data base.
223	T	HMB VALUE MISSING ON FILE CARD FOR A DA FILE - DEFAULT VALUE OF 512 USED	The HMB (home blocks) parameter value has not been specified on the FILE control statement for an area with direct access file organization. A default value of 512 has been set for the file.	No action if the value of 512 is acceptable; otherwise, supply the correct value of HMB on the FILE control statement and recompile.
225	E	ACTUAL/VIRTUAL RESULT INVALID ON DATA-AGGREGATES OR THEIR COMPONENTS	The ACTUAL or VIRTUAL RESULT clause cannot be specified for groups or items within a group.	Eliminate the ACTUAL or VIRTUAL RESULT clause for the group or the group item.
226	E	CHECK VALUE OPTION IS NOT ALLOWED FOR COMPLEX DATA ITEMS	A COMPLEX data item cannot have a CHECK VALUE clause associated with it.	Correct the data item description, and recompile.
231	E	CONVERSION NOT POSSIBLE FOR CHECK LITERAL	The literal specified in the CHECK clause cannot be converted to the picture or type defined for the data item.	Check the list of examples of valid literals for the CHECK VALUE clause, change the value to a valid literal, and recompile.
232	E	PICTURE, VALUE PHRASE, OR PROCEDURE MAY BE SPECIFIED ONCE EACH IN CHECK CLAUSE	In the CHECK clause, each option (PICTURE, VALUE, or data base procedure) can be specified only once.	Correct the CHECK clause, and recompile.
233	E	TABLE OVERFLOW - TOO MANY LITERALS SPECIFIED, **DDL ABORTED**	The table used to store literals for the CHECK clause is full.	Increase the field length, and recompile.
234	E	NO PRIMARY KEY SPECIFIED	A nonsequential file must have a key specified in the area control entry.	Specify the key for the nonsequential file in the area control entry, and recompile.
236	E	LITERALS SPECIFIED IN CHECK VALUE CLAUSE MUST BE IN ASCENDING ORDER	The literals in the CHECK IS VALUE clause must be in ascending numeric order.	Arrange the literals in the correct numeric order in the CHECK IS VALUE clause, and recompile.
237	E	INTEGER-2 INVALID	The second number in the TYPE FIXED clause must be an integer.	Change the second number in the TYPE FIXED clause to an integer, and recompile.
238	E	DATA-BASE-DATA-NAME IS NOT UNIQUE WITHIN THIS RECORD	The data name in the given line has already been used for a data name in this record.	Change the data name, and recompile.

TABLE B-6. DDL SCHEMA DIAGNOSTICS (Contd)

Error Code	Type	Message	Significance	Action
239	E	THE ITEM ASSOCIATED WITH THIS CLAUSE MUST HAVE TYPE OR PICTURE CLAUSE SPECIFIED	Either the TYPE clause or the PICTURE clause must be specified for a nonrepeating data item.	Correct the format for the CHECK clause, and recompile.
240	E	ITEM ASSOCIATED WITH THIS CLAUSE CANNOT BE DESCRIBED WITH A VIRTUAL RESULT CLAUSE	A data item cannot be described with both an ACTUAL RESULT and a VIRTUAL RESULT clause.	Eliminate either the ACTUAL RESULT or the VIRTUAL RESULT clause, and recompile.
246	E	THE OCCURS CANNOT BE USED WITH RESULT CLAUSE	A repeating data item cannot be described with the ACTUAL or VIRTUAL RESULT clause.	Use either the OCCURS clause to describe a repeating data item, or the RESULT clause for establishing the value of the data item by the execution of a data base procedure.
247	E	INTEGER MUST BE GREATER THAN 0	The integer in an OCCURS integer-1 TIMES clause must be greater than zero.	Correct the value of the integer in the OCCURS clause, and recompile.
248	E	ITEM NOT DEFINED IN THIS RECORD	The data name in an OCCURS data name TIMES clause must have been previously defined in the same record.	Define the data name in the same record, and recompile.
249	E	ITEM MUST BE AN INTEGER	The data name in an OCCURS data name TIMES clause must be defined as an integer data item.	Describe the data item as an integer in the TYPE clause.
250	E	ITEM IS A DATA AGGREGATE COMPONENT OF A REPEATING GROUP OF VARIABLE DIMENSION	A variable occurrence data item cannot be nested within another variable occurrence data item.	Eliminate the nested variable occurrence data item, describe it in a separate OCCURS clause, and recompile.
251	E	ITEM CANNOT BE A VIRTUAL RESULT	The data name in an OCCURS data name TIMES clause cannot be defined with a VIRTUAL RESULT clause.	Eliminate the VIRTUAL RESULT clause, define the data name with a CHECK IS VALUE clause, and recompile.
253	E	CONVERSION ERROR ON RECORD CODE LITERAL	Each record code literal must have a unique value within the area and must be able to be converted to the type or picture specification of the data item.	Check the type or picture specification clause for the data item.
254	E	CHECK VALUE CLAUSE NOT SPECIFIED IN DEPENDING ON ITEM	The data name in an OCCURS data name TIMES clause must be defined with a CHECK IS VALUE clause.	Define the data name with a CHECK IS VALUE clause, and recompile.
255	E	DEPENDING ON ITEM IS A COMPONENT OF A REPEATING GROUP OF VARIABLE DIMENSION	The data name in an OCCURS data name TIMES clause must not be subordinate to a group item that is defined with an OCCURS data name TIMES clause.	Correct the format of the OCCURS clause, and recompile.

TABLE B-6. DDL SCHEMA DIAGNOSTICS (Contd)

Error Code	Type	Message	Significance	Action
256	E	FOR AK FILE, KL MUST BE SET IN FILE CARD AND MUST BE LESS THAN 9	For an area with actual key file organization, the KL parameter in the FILE control statement for the area must specify the key length in characters. This length must be in the range 1 through 8.	Correct the FILE control statement, and recompile.
258	E	NESTING OF REPEATING GROUPS MORE THAN THREE LEVELS DEEP NOT SUPPORTED	A maximum of three levels of nested groups can be specified.	Eliminate the excess levels of nesting, and recompile.
259	E	CHECK VALUE FOR DEPENDING ON ITEM MUST BE GREATER THAN ZERO	The value in the CHECK clause must be greater than zero.	Correct the value in the CHECK clause, and recompile.
260	E	ENCODING/DECODING CLAUSE CANNOT BE SPECIFIED WITH GROUP ITEMS	The ENCODING/DECODING clause cannot be used with repeating groups.	Eliminate the ENCODING/DECODING clause from the repeating group data description entry.
261	E	CALL CLAUSE CANNOT BE SPECIFIED WITH GROUP ITEMS	The CALL clause can be specified only for an elementary group item.	Correct the CALL clause, and recompile.
262	E	CHECK CLAUSE CANNOT BE SPECIFIED WITH GROUP ITEMS	The CHECK clause cannot be used with repeating groups.	Correct the CHECK clause, and recompile.
264	E	ACCESS-CONTROL ENTRIES MUST BE CONTIGUOUS	All ACCESS-CONTROL clauses must follow each other in the area description entry. CALL clauses are not allowed between access control entries.	Restructure the area description entry, and recompile.
265	E	ACCESS-CONTROL UPDATE PREVIOUSLY SPECIFIED FOR THIS AREA	Each usage mode can be used only once for an area.	Eliminate the duplicate usage mode specification, and recompile.
266	E	ACCESS-CONTROL RETRIEVAL PREVIOUSLY SPECIFIED FOR THIS AREA	A usage mode can be specified only once for an area.	Eliminate the duplicate usage mode specification, and recompile.
267	E	INVALID ACCESS-CONTROL DBP NAME	The data base procedure name does not conform to the rules specified for user-defined names.	Check the rules for names, correct the name, and recompile.
268	E	ACCESS-CONTROL LITERAL LONGER THAN 30 CHARACTERS - INVALID	The literal specified in the ACCESS-CONTROL clause is longer than the allowed maximum of 30 characters.	Reduce the number of characters in the literal, and recompile.
269	E	ACCESS-CONTROL DBP NAME LONGER THAN 7 CHARACTERS - INVALID	The data base procedure name specified in the ACCESS-CONTROL clause is longer than the allowed maximum of seven characters.	Change the data base procedure name, and recompile.
270	E	INVALID/MISSING ACCESS-CONTROL LOCK	The ACCESS-CONTROL clause is not complete according to the format specified in this manual.	Check the format, correct it, and recompile.

TABLE B-6. DDL SCHEMA DIAGNOSTICS (Contd)

Error Code	Type	Message	Significance	Action
279	E	UNQUALIFIED ITEM NAME IS CONSIDERED UNDEFINED	A data name that is specified in more than one record description entry must be qualified with the record name in order to be fully defined.	Reformat the record description entry to include the record name, and recompile.
280	E	**DDL ABORTED, TOO MANY SYNTAX ERRORS	The DDL compilation is aborted when 200 fatal diagnostics have been issued.	Correct the diagnostics, and recompile.
281	E	MISSING CLOSING PARENTHESIS	The closing parenthesis is missing on a subscript.	Include the parenthesis, and recompile.
282	T	KEY INFORMATION FROM FILE CARD INCONSISTENT WITH KEY ITEM DECLARATION, ITEM DECLARATION VALUES USED	The KL, KT, or KP parameter in the FILE control statement for an area is not consistent with the length, type, or position of the data name defined as the key. The value for the data name is used.	Change the applicable parameter on the FILE control statement to be consistent with the data name defined as the key; or take no action.
283	E	KEY MAY BE NESTED AT MOST ONE LEVEL	A data name defined as a key can be nested only one level.	Eliminate the additional levels of nesting, and recompile.
284	E	MAXIMUM RECORD ENTRY LENGTH EXCEEDED - SIMPLIFY SOURCE	DDL fixed internal table overflow.	Simplify the source program, and recompile.
285	E	MAXIMUM ITEM ENTRY LENGTH EXCEEDED - SIMPLIFY SOURCE	DDL fixed internal table overflow.	Simplify the source program, and recompile.
286	E	ITEM SIZE EXCEEDS THE MAXIMUM OF 32767	The maximum item size is 32767 characters.	Reduce the item size, and recompile.
287	E	MAXIMUM RECORD SIZE EXCEEDED FOR THE PREVIOUS RECORD	The maximum record size is 81870 characters.	Reduce the record size, and recompile.
288	E	MAXIMUM DATA CONTROL ENTRY LENGTH EXCEEDED - SIMPLIFY SOURCE	DDL fixed internal table overflow.	Reduce the size of the data control entry, and recompile.
289	E	MAXIMUM ITEM COUNT EXCEEDED	A maximum of 81870 items can be declared.	Reduce the number of items, and recompile.
290	E	MAXIMUM RECORD COUNT EXCEEDED	A maximum of 4095 records can be declared.	Reduce the number of records, and recompile.
291	E	MAXIMUM AREA COUNT EXCEEDED	A maximum of 4095 areas can be declared.	Reduce the number of areas, and recompile.
292	E	MAXIMUM RELATION COUNT EXCEEDED	A maximum of 4095 relations can be declared.	Reduce the number of relations, and recompile.
293	E	MAXIMUM ITEM COUNT FOR A RECORD EXCEEDED	The number of items defined for a record has exceeded the allowed limit of 4095 items.	Reduce the number of items, and recompile.
294	E	SEQUENTIAL FILE ORGANIZATION IS NOT SUPPORTED	Sequential file organization is not supported.	Change the file organization, and recompile.

TABLE B-6. DDL SCHEMA DIAGNOSTICS (Contd)

Error Code	Type	Message	Significance	Action
295	E	MAXIMUM RELATION ENTRY LENGTH EXCEEDED - SIMPLIFY SOURCE	The length of the entry is too long; too many areas have been joined in one relation.	Simplify the relation entry by specifying more relations, and recompile.
296	E	FATAL ERRORS, NO SCHEMA CREATED - **DDL ABORTED**	Fatal syntax errors have inhibited generation of the schema directory; DDL is aborted.	Correct the syntax errors, and recompile.
297	E	ONLY ONE CHECK CLAUSE IS ALLOWED	Only one CHECK clause can be specified for a data item.	Eliminate the additional CHECK clause, and recompile.
298	E	TYPE, PICTURE, OR OCCURS MUST BE SPECIFIED FOR DATA SUB-ENTRY	A data item must be defined with at least one of the following clauses: TYPE, PICTURE, or OCCURS.	Define the data item with one of these three clauses, and recompile.
300	E	INVALID PICTURE LENGTH---- GREATER THAN 18 CHARACTERS	A numeric picture can have only 18 significant digits with up to 30 characters specified.	Correct the picture specification, and recompile.
301	E	INVALID CHARACTER IN A NUMERIC PICTURE SPECIFICATION	A numeric picture specification can contain only the characters 9, V, P, T, and . (decimal point).	Correct the numeric picture specification, and recompile.
302	E	INVALID CHARACTER IN A FLOATING POINT PICTURE-SPECIFICATION	A floating point picture specification can contain only the characters 9, V, P, T, and . (decimal point).	Correct the floating point picture specification, and recompile.
303	E	INVALID BINARY DIGIT IN A DECIMAL NUMERIC PICTURE-SPECIFICATION	The character 1 is not allowed in a picture specification.	Eliminate the character 1, and recompile.
304	E	TWO'S COMPLEMENT NOT SUPPORTED IN THIS RELEASE OF DDL	The character 2 is not allowed in a picture specification.	Check the requirements for valid character representation, correct the picture specification, and recompile.
305	E	ONE'S COMPLEMENT NOT SUPPORTED IN THIS RELEASE OF DDL	The character 3 is not allowed in a picture specification.	Check the requirements for valid character representation, correct the picture specification, and recompile.
306	E	DOUBLY DEFINED DECIMAL POINT	Only one decimal point can be specified in a picture specification.	Correct the format of the picture clause, and recompile.
307	E	DECIMAL POINT IS INVALID IN AN ALPHA OR ALPHANUMERIC PICTURE SPECIFICATION	A decimal point cannot be specified in a picture specification that also contains the character A or X.	Correct the format of the picture clause, and recompile.
308	E	POINT SPECIFIER IS INVALID IN THE EXPONENT PART OF A FLOATING-POINT NUMERIC PICTURE SPECIFICATION	A decimal point should not be specified in the exponent portion of a picture specification.	Correct the format of the picture clause, and recompile.

TABLE B-6. DDL SCHEMA DIAGNOSTICS (Contd)

Error Code	Type	Message	Significance	Action
309	E	SIGN SPECIFIER IS INVALID IN AN ALPHA OR ALPHANUMERIC PICTURE SPECIFICATION	The characters S and T cannot be included in a picture specification that contains the character A or X.	Check the requirements for valid character representation, correct the picture specification, and recompile.
311	E	INVALID SIGN SPECIFIER IN A FLOATING-POINT PICTURE SPECIFICATION	The character T cannot be used as a sign specifier in a floating point picture specification.	Check the requirements for valid character representation, correct the picture specification, and recompile.
312	E	EXPONENT SPECIFIER IS INVALID IN AN ALPHA OR ALPHANUMERIC PICTURE SPECIFICATION	The characters E and K cannot be included in a picture specification that also contains the character A or X.	Check the requirements for valid character representation, correct the picture specification, and recompile.
313	E	DOUBLY DEFINED EXPONENT SPECIFIER	The character E or K can be used only once in a picture specification.	Check the requirements for valid character representation, correct the picture specification, and recompile.
314	E	REPETITION SPECIFIER IS IN AN INVALID LOCATION IN THE PICTURE SPECIFICATION	A repetition specifier can be used only after the characters A, X, P, and 9.	Correct the format of the picture clause, and recompile.
315	E	INVALID PICTURE SPECIFICATION	The picture specification is illegal.	Check the requirements for the picture specification, correct the format of the picture clause, and recompile.
316	E	FLOATING-POINT PICTURE SPECIFICATION IS MISSING EXPONENT PART	The exponent portion of a floating point picture specification has been omitted.	Correct the format of the picture clause, and recompile.
317	E	ILLEGAL CHARACTERS IN A PICTURE LITERAL	The only valid characters in a picture specification are A, X, P, 9, V, S, T, and . (decimal point).	Check the requirements for the picture specification, correct the picture clause format, and recompile.
318	E	DOUBLY DEFINED SIGN IN THE PICTURE SPECIFICATION	The character S or T can be included only once in a picture specification.	Check the requirements for valid character representation, correct the picture specification, and recompile.
319	E	INVALID REPETITION CHARACTER IN THE PICTURE SPECIFICATION	A repetition specifier must be an integer number enclosed in parentheses.	Correct the format of the picture specification, and recompile.
320	E	SIGN SPECIFIER IS NOT THE RIGHT OR LEFT MOST CHARACTER IN THE PICTURE SPECIFICATION	The character T must be the rightmost character in a numeric picture specification.	Check the requirements for valid character representation, correct the picture specification, and recompile.
321	E	DOUBLY DEFINED SYMBOL	A data name has been used twice as the same type of name (record name, item name, etc.) in the same record.	Eliminate the duplicate data name, substituting a unique name, and recompile.

TABLE B-6. DDL SCHEMA DIAGNOSTICS (Contd)

Error Code	Type	Message	Significance	Action
322	E	SEPARATE SIGN IS NOT SUPPORTED BY CDCS	The character S in a picture specification is not supported by CDCS.	Check the requirements for valid character representation, correct the picture specification, and recompile.
323	E	INTEGER-1 IS ILLEGAL SIZE	Integer-1 in the TYPE clause must be in the range 1 through 18.	Correct integer-1 in the TYPE clause, and recompile.
324	E	EXPONENT SPECIFIERS (E OR K) ARE NOT SUPPORTED BY CDCS 1.0	The picture specification characters E and K are not supported by CDCS.	Check the requirements for valid character representation, correct the picture specification, and recompile.
325	E	RECORD BUFFER OVERFLOW, ***DDL ABORTED***	The schema record buffer has overflowed; the job should be run with a larger field length or with an NI parameter in the DDL control statement to specify a larger record buffer.	Run the job with a larger field length or with an NI parameter in the DDL control statement to specify a larger record buffer.
326	E	SIGN OVER-PUNCH (T) MUST OCCUPY THE RIGHT-MOST DIGIT POSITION	The character T must be specified as the rightmost digit in a picture specification.	Check the requirements for the picture specification, correct the format, and recompile.
327	E	FATAL ERRORS OCCURRED - - FOLLOWING STATEMENTS IGNORED	Errors have occurred that prevent DDL from processing; statements not processed are listed.	Correct previous errors, and recompile.
330	E	INVALID/MISSING KEY-NAME	The key name identifying the concatenated key in the KEY clause is invalid or missing.	Check the rules governing valid concatenated keys, correct the key, and recompile.
331	E	CONCATENATED KEY-NAME IS DOUBLY DEFINED	The key name used to identify the concatenated key can be defined only once.	Eliminate the duplicate concatenated key definition, and recompile.
332	E	DATA-NAMES BRACKET (</>) MISSING	A less than symbol (<) and a greater than symbol (>) must enclose the list of data names composing the concatenated key.	Correct the format of the KEY IDENTIFIER clause, and recompile.
333	E	DATA-NAMES MUST BE CONSECUTIVE AND CONTIGUOUS (NO PADDING)	Data names in the KEY must follow each other consecutively in the record description.	Correct the format of the KEY clause, and recompile.
334	E	kkkkkk IS UNDEFINED	The stated data item used as part of the concatenated key must be defined.	Define the data item in the data description entry, and recompile.
335	E	kkkkkk IS A REPEATING ITEM - NOT ALLOWED	The stated data item used as part of the concatenated key cannot be a repeating data item.	Change the data item, and recompile.

TABLE B-6. DDL SCHEMA DIAGNOSTICS (Contd)

Error Code	Type	Message	Significance	Action
336	E	AT LEAST ONE DATA-NAME MUST BE SPECIFIED FOR A CONCATENATED KEY	When a concatenated key is specified, at least one data name is required.	Insert at least one data name in the KEY IDENTIFIER clause, and recompile.
337	E	MORE THAN 64 DATA-NAMES NOT ALLOWED IN A CONCATENATED KEY	A maximum of 64 contiguous elementary data items are allowed in the KEY clause for the concatenated key.	Correct the format of the KEY IDENTIFIER clause, and recompile.
338	E	KEYS MUST BE DEFINED IN FIRST RECORD TYPE WITHIN THE AREA	Keys for an area with multiple record types must be defined in the first record type in the area.	Rearrange the order of the records in the area, and recompile.
340	E	COMPRESSION PREVIOUSLY SPECIFIED FOR THIS AREA	Only one COMPRESSION clause is allowed in an area control statement.	Eliminate the additional COMPRESSION clause, and recompile.
341	E	DECOMPRESSION PREVIOUSLY SPECIFIED FOR THIS AREA	Only one DECOMPRESSION clause is allowed in an area control statement.	Eliminate the additional DECOMPRESSION clause, and recompile.
342	E	SYSTEM OR DATA-BASE-PROCEDURE MUST BE USED FOR BOTH USAGE MODES	In the COMPRESSION/ DECOMPRESSION specification for an area, either the SYSTEM or PROCEDURE option can be specified; both SYSTEM and PROCEDURE cannot be used in an area.	Correct the COMPRESSION or DECOMPRESSION clause, and recompile.
343	E	COMPRESSION OR DECOMPRESSION CLAUSE MISSING	If either a COMPRESSION or DECOMPRESSION clause is specified, the other must be specified also.	Add the missing COMPRESSION or DECOMPRESSION clause, and recompile.
344	E	SYSTEM OR DATA-BASE-PROCEDURE NOT SPECIFIED	When COMPRESSION OR DECOMPRESSION is specified, either SYSTEM or PROCEDURE procedure-name must be specified.	Correct the COMPRESSION and DECOMPRESSION clauses, and recompile.
345	E	COMPRESSION/DECOMPRESSION CLAUSE NOT CONTIGUOUS	A DECOMPRESSION clause must immediately follow a COMPRESSION clause, or vice versa.	Correct the syntax error, and recompile.
346	E	INVALID TYPE OPTION - - SKIPPING TO PERIOD	In the TYPE clause, the keywords used to specify the type option are incompatible or duplicates exist.	Correct the TYPE clause, and recompile.
400	E	ALL OR ENTRY NAME EXPECTED, NOT SPECIFIED	Either the word ALL or a valid entry name was not found when one was expected.	Check the syntax, and make the necessary corrections.
400	E	INVALID/NON-UNIQUE RELATION NAME	A relation name specified in the RELATION NAME clause can be used only once in the schema.	Substitute a unique name for the relation, and recompile.
401	T	RELATION NAME LONGER THAN 30 CHARACTERS---NAME TRUNCATED	A relation name cannot be longer than 30 characters. Any characters following the first 30 are truncated.	Reduce the number of characters in the relation name, or take no action.

TABLE B-6. DDL SCHEMA DIAGNOSTICS (Contd)

Error Code	Type	Message	Significance	Action
401	E	UNKNOWN ENTRY NAME	The specified name could not be found in the directory of entry names.	Check the specified name used, and make the appropriate correction.
402	E	RESERVED WORD JOIN MISSING	The reserved word JOIN is required in the JOIN clause format.	Correct the format of the JOIN clause, and recompile.
402	E	MISSING KEYWORD IN EXHIBIT SYNTAX	One of the keywords, AREA, RECORD ITEM, ALL-ENTRIES, or RELATION, was not specified in the EXHIBIT directive.	Correct the EXHIBIT directive, and recompile the schema.
403	C	INVALID/MISSING SCHEMA NAME	Either the schema name was not a legal file name, or it was omitted. Schema name is required.	Correct the schema name entry, and recompile.
403	E	kkkkkk DBI IS INVALID/ UNDEFINED	The specified identifier is undefined and/or invalid.	Check the rules for the join identifier, make the appropriate changes so that the data item is valid and is defined, and recompile.
404	E	RELATION ENTRIES MISSING IN SCHEMA	When the keyword RELATION was specified in the EXHIBIT CLAUSE, NO RELATION ENTRIES could be found in the schema.	Check to see that the correct schema is being used. Make any necessary corrections to the relation entry.
404	E	kkkkkk DBI SUBSCRIPTS IN ERROR	The specified identifier subscripts are in error.	Check the rules for subscripting, correct the format of the JOIN clause, and recompile.
405	E	CONSTRAINTS ENTRY MISSING IN SCHEMA	When the keyword constraints was specified in the EXHIBIT clause, no constraint entries could be found in the schema.	Check to see that the correct schema is being used. Make any necessary corrections to the constraint entry.
405	E	SUBSCRIPT ANY CANNOT BE USED FOR SOURCE DBIS	The subscript ANY cannot be used for source identifiers.	Correct the format of the JOIN clause, and recompile.
406	E	kkkkkk DBI SIZE EXCEEDS THE MAXIMUM OF 255 CHARACTERS	The identifier must not specify a data item that is more than 255 characters long.	Change the data item used as the identifier to one having the appropriate number of characters, and recompile.
407	E	SOURCE AND TARGET DBI PICTURE/ TYPE CHARACTERISTICS MUST BE IDENTICAL	The identifiers to the left and to the right of a relational operator must have identical characteristics in the picture specification or in the TYPE specification.	Change the PICTURE or TYPE clause so that both identifiers have identical characteristics.
408	E	SOURCE AND TARGET DBIS DIFFER IN SIZE	Source and target identifiers must be the same size.	Change the size of the data item(s) in the PICTURE or TYPE clause so that both are the same size.

TABLE B-6. DDL SCHEMA DIAGNOSTICS (Contd)

Error Code	Type	Message	Significance	Action
409	E	SUBSCRIPT ANY ON A TARGET DBI CAN BE SPECIFIED ONLY FOR ALTERNATE KEYS	The subscript ANY on a target identifier can be specified only for alternate keys.	Check the rules for subscripting, make the appropriate change, and recompile.
410	E	FILE CANNOT BE JOINED TO ITSELF (RECURSION)	An area (file) cannot be joined to itself in a relation.	Correct the structure of the relation, and recompile.
411	E	ILLEGAL PATH-CYCLING OR DISCONTINUITY	See requirements for JOIN clause identifiers.	Correct the format of the JOIN clause, and recompile.
412	E	RANK EXCEEDS MAXIMUM OF 64 FILES	The number of files being joined in a relation has exceeded the limit.	Reduce the number of files joined in the relation so that the maximum is not exceeded, and recompile.
413	E	kkkkkk DBI BELONGS TO AREA WITH MULTIPLE RECORDS -- NOT ALLOWED IN A RELATION PATH	The specified identifier cannot belong to an area with multiple record descriptions.	Check the rules for structuring a relation.
414	E	RESERVED WORD EQ MISSING	The reserved word EQ is required between each pair of identifiers included in the JOIN clause.	Correct the format of the JOIN clause, and recompile.
415	E	UNABLE TO COMPLETE ENTRY - ENTRY WILL BE IGNORED - SEARCHING FOR A PERIOD	A period is required to terminate the relation entry.	Insert the period, and recompile.
416	E	ILLEGAL PATH - SOURCE DBI MUST BE IN SAME AREA AS PREVIOUS TARGET DBI	Each source identifier (except the first) must be in the same area as the previous target identifier in the JOIN clause.	Check the rules for structuring a relation, correct the identifiers, and recompile.
500	E	INVALID CONSTRAINT NAME	A constraint name can contain 1 to 30 alphabetic and numeric characters and hyphens.	Change the constraint name so that it is valid, and recompile.
501	E	CONSTRAINT NAME NOT UNIQUE	A constraint name specified in the CONSTRAINT NAME clause duplicates another constraint name or an area name.	Sustitute a unique name for the constraint, and recompile.
502	E	CONSTRAINT DATA ITEM NOT A PRIMARY OR ALTERNATE KEY	The data name used in a constraint entry must be designated as a primary or alternate key in a KEY clause.	Correct the KEY clause or the data name in the constraint entry, and recompile.
503	E	CONSTRAINT DATA ITEMS DIFFER IN CHARACTERISTICS	The data names in the DEPENDS ON clause must be defined with identical TYPE or PICTURE clauses.	Change the TYPE or PICTURE clauses so that the data names have identical characteristics, and recompile.
504	E	RESERVED WORD DEPENDS MISSING	The word DEPENDS must appear between two data names in the constraint entry.	Insert the word DEPENDS in the constraint entry, and recompile.

TABLE B-6. DDL SCHEMA DIAGNOSTICS (Contd)

Error Code	Type	Message	Significance	Action
505	E	CYCLE DETECTED IN CONSTRAINTS	A series of constraints makes a cycle.	Change the constraint entry that makes a cycle, and recompile.
506	E	CONSTRAINT DATA ITEM (PARENT) CANNOT HAVE DUPLICATE VALUES	Data-name-2 in the constraint entry must be an item designated as an alternate or primary key with no duplicates allowed.	Make an appropriate change in the constraint entry or KEY clause, and recompile.
507	E	CONSTRAINT NAME LONGER THAN 30 CHARACTERS - NAME TRUNCATED	The maximum length for a constraint name is 30 characters. Characters following the first 30 are truncated.	Reduce the number of characters in the constraint name, or take no action.
508	E	DBI IN ABOVE CONSTRAINT CANNOT DEPEND ON ITSELF	The same data item has been specified as data-name-1 and data-name-2 in a constraint entry; different items must be specified.	Correct the constraint entry, and recompile.
991	E	KEY CHARACTERISTICS NOT THE SAME IN ALL RECORD TYPES FOR AREA aaaaaaa	In an area with multiple record types, key items do not appear in the same position or are not the same size and class in each record type.	Correct the key items, and recompile.
992	E	DATA CONTROL ENTRY FOR AREA aaaaaaa MISSING	A data control entry is required for each area in the data base.	Include a data control entry in the source program for that area, and recompile.
993	E	RECORD CODE VALUE IS NOT EQUAL TO CHECK VALUE LITERAL OF CORRESPONDING ITEM FOR REC rrrrrrrrrr, IN AREA aaaaaaa	The data item used for a record code value is defined in the record description entry with a CHECK IS VALUE clause. The literal specified in the CHECK clause must be the same as the literal specified in the RECORD CODE VALUE clause.	Correct the CHECK IS VALUE clause, and recompile.
994	E	REC rrrrrrrrrr IN AREA aaaaaaa DOES NOT CONTAIN ITEM CORRESPONDING TO DATANAME IN LOCATION, SIZE, AND CLASS	The designated record does not contain a data item corresponding to the data name specified in a RECORD CODE clause. Each record type in the area must have a data item in the same position with the same size and class.	Restructure the record(s) so that the data name specified in the RECORD CODE clause has the same position and size throughout the area.
995	E	LITERALS SPECIFIED ON LINE nnnnnn ARE NOT IN ASCENDING ORDER	Nonnumeric literals in the CHECK IS VALUE clause must be specified in ascending order according to the collating sequence specified for the area in which the data item is contained.	Correct the format of the CHECK IS VALUE clause, and recompile.

TABLE B-6. DDL SCHEMA DIAGNOSTICS (Contd)

Error Code	Type	Message	Significance	Action
996	E	VARIABLE DIMENSION GROUP OR GROUP CONTROL ITEM INCONSISTENT ACROSS RECORD TYPES FOR AREA aaaaaaa	A variable occurrence data item must be defined for each DDL record type in the area. The data item must be at the same location in each record type and must be the same length for one occurrence of the data item. The data item that controls the number of occurrences must also be in the same location and have the same length and type in each record type.	Make the necessary correction, and recompile.
997	T	FILE CARD VALUES FOR HL, TL, CP, CL, MNR, OR MRL OVERWRITTEN BY SCHEMA VALUES FOR AREA aaaaaaa	One of the values specified in an area FILE control statement has been overwritten by values computed during syntax analysis of the area. The values that can be overwritten include header length, trailer length, character position and character length for CYBER Record Manager T-type records, minimum record length, and maximum record length.	No action, or change the parameter(s) in the FILE control statement.
998	T	VARIABLE DIMENSION GROUP WITH RECORD TYPE NOT T, AREA aaaaaaa	The specified area contains a record type with a variable occurrence data item and the FILE control statement does not specify a CYBER Record Manager record type of T (trailer count).	Change the FILE control statement, and recompile.

DDL SUBSCHEMA COMPILATION AND LIBRARY MAINTENANCE DIAGNOSTICS FOR COBOL AND QUERY UPDATE

The DDL diagnostic messages that can be issued either during compilation of a COBOL or Query Update subschema source program or during library maintenance operations are listed in table B-7.

The library maintenance messages begin with an alphabetic character. The subschema compilation messages begin with an error code between 100 to 449 (these error codes are enclosed in asterisks and written to the source listing directly preceding the message). These messages do not necessarily indicate termination of DDL compilation; unless otherwise stated compilation continues but a subschema is not created.

TABLE B-7. DDL SUBSCHEMA COMPILATION AND LIBRARY MAINTENANCE DIAGNOSTICS FOR COBOL AND QUERY UPDATE

Error Code	Message	Significance	Action
	DID NOT LOCATE aaaaaa.....-- PURGE NOT POSSIBLE	The subschema name specified for the purge could not be located in the subschema library.	Specify the correct subschema name and recompile.
	DID NOT LOCATE SUB-SCHEMA TO BE REPLACED---NEW SUB-SCHEMA HAS BEEN ADDED	The subschema to be replaced in the library could not be located; the current subschema has been added to the library.	Specify correct library or subschema name and recompile.
	EMPTY INPUT FILE---PURGE NOT POSSIBLE	The names of the subschemas to be purged must be in the input file.	Make appropriate corrections to the input file and recompile.
	EMPTY SUB-SCHEMA FILE, DDL ABORTED	While executing an audit, collect, or purge of a subschema library, DDL found it empty and terminated execution.	Specify the correct library and recompile.
	ILL-FORMATTED LIBRARY--NOT UPDATABLE, DDL ABORTED	The subschema library contains an error and cannot be updated. DDL terminates.	Check for empty library or file that is not a library. Correct the error and recompile.
	OLD SUB-SCHEMA FILE BAD, SUB-SCHEMA LENGTH IS ZERO	During execution of the facility for compacting the subschema library, DDL found that a subschema in the old library had a length of zero, which indicates an error in the file.	Specify the correct subschema library and recompile.
	SUB-SCHEMA WITH THE SAME NAME AS THE NEW SUB-SCHEMA ALREADY EXISTS--FILE NOT UPDATED	The current subschema could not be added to the library since the library contains a subschema with the same name.	Change the subschema name and recompile.
	WARNING---EMPTY SUB-SCHEMA FILE	The subschema library contains no subschemas. The subschemas could have been purged prior to this compilation. The new subschema is added to the library and is the only subschema in the library.	If subschemas have been purged, create a new library.
100	EMPTY INPUT FILE	The input file is empty and the compilation is terminated.	Create a new file and recompile.
101	TITLE DIVISION DECLARATION INCORRECT	Reserved word TITLE or DIVISION is either misspelled or missing. Compilation is aborted.	Correct the error and recompile.
103	SUB-SCHEMA DECLARATION IS INCORRECT	A reserved word in the SS clause is either misspelled or missing. Compilation is aborted.	Correct the error and recompile.
104	INVALID SCHEMA NAME	The schema name is either the same as a reserved word or does not conform to the naming conventions. Compilation is aborted.	Correct the error and recompile.

TABLE B-7. DDL SUBSCHEMA COMPILATION AND LIBRARY MAINTENANCE DIAGNOSTICS FOR COBOL AND QUERY UPDATE (Contd)

Error Code	Message	Significance	Action
105	ALIAS DIVISION DECLARATION IS INCORRECT	Reserved word DIVISION is either misspelled or missing.	Correct the error and recompile.
106	AD DECLARATION IS INCORRECT	A reserved word in the Alias Division is misspelled.	Correct the error and recompile.
107	INVALID NAME IN ALIAS DIVISION	The specified name does not conform to the naming conventions.	Correct the error and recompile.
108	RESERVED WORD -BECOMES- IS MISSING	Reserved word BECOMES is not included in the AD clause.	Correct the error and recompile.
109	DUPLICATE ALIAS NAME	Assigned alias names must be unique.	Correct the error and recompile.
110	ALIAS-NAME-1 UNKNOWN	The name immediately following AD does not exist in the schema.	Correct the error and recompile.
111	INVALID QUALIFIER NAME	The specified qualifier name is either the same as a reserved word or does not conform to the naming conventions.	Correct the error and recompile.
112	REALM DIVISION NOT SPECIFIED, DEFAULTED TO -ALL-	The Realm Division statement could not be located; all the areas in the schema are copied to the subschema. Compilation is aborted.	Insert the Realm Division statement and recompile.
113	REALM DIVISION TITLE IS INCORRECT	Reserved word DIVISION is either missing or misspelled.	Correct the error and recompile.
114	REALM DESCRIPTION IS INCORRECT, DEFAULTED TO -ALL-	Reserved word RD is misspelled or missing; the clause is ignored. All the areas in the schema are copied to the subschema.	Correct the error and recompile.
115	INVALID REALM NAME	The specified name is either the same as a reserved word or does not conform to the naming conventions. All the areas in the schema are copied to the subschema.	Correct the error and recompile.
116	DUPLICATE REALM NAME	All realm names must be unique.	Correct the error and recompile.
117	RECORD DIVISION TITLE IS INCORRECT	Reserved words RECORD and DIVISION are either missing or misspelled.	Correct the error and recompile.
118	LEVEL NUMBER NOT SPECIFIED, DEFAULTED TO -01-	No level number was specified after the RECORD DIVISION statement; therefore, the current entry is assumed to be a record entry and the level number is defaulted to 01.	Correct the error and recompile.
119	FIRST ENTRY IN RECORD DIVISION IS NOT A RECORD ENTRY, UNABLE TO CONTINUE-COMPILATION ABORTED	The first entry specified after the RECORD DIVISION statement is expected to be a record entry. Compilation is terminated on the subschema missing this entry.	Correct the error and compile the uncompiled subschema.
120	INVALID RECORD NAME	The specified record name is either the same as a reserved word or does not conform to the naming conventions.	Correct the error and recompile.
121	DUPLICATE RECORD NAME	All record names must be unique.	Correct the error and recompile.

TABLE B-7. DDL SUBSCHEMA COMPILATION AND LIBRARY MAINTENANCE DIAGNOSTICS FOR COBOL AND QUERY UPDATE (Contd)

Error Code	Message	Significance	Action
122	CANNOT LOCATE OWNER REALM IN THE SCHEMA	The schema area in which this record is defined is not defined in the subschema Realm Divsion.	Correct the error and recompile.
123	INVALID LEVEL NUMBER	The level number is not an integer value.	Correct the error and recompile.
124	INVALID DATA-NAME	The specified item name is either a reserved word or does not conform to the naming conventions.	Correct the error and recompile.
125	DATA-NAME NOT UNIQUE	The specified item name is not unique within its dominant item.	Correct the error and recompile.
126	VALUE OF LEVEL NUMBER IS LESS THAN THE FIRST ITEM DEFINED IN THE RECORD	Level numbers cannot be lower than the first level number specified in the record.	Correct the error and recompile.
127	PICTURE LITERAL GREATER THAN 30 CHARACTERS, RIGHT MOST CHARACTERS TRUNCATED	A picture literal cannot be greater than 30 characters; the rightmost characters are truncated.	Correct the error and recompile.
128	INVALID USAGE TYPE, DEFAULTED TO -DISPLAY-	The specified usage type is either missing or misspelled.	Correct the error and recompile.
129	USAGE DOES NOT AGREE WITH THE USAGE IN THE GROUP ITEM	The usage specified at the elementary item level differs from the usage specified at the group level. The usage at the group level overrides the elementary item usage.	Change usage to correct specification.
130	INVALID OCCURRENCE VALUE	The specified occurrence value is not an integer value.	Correct the error and recompile.
131	THE SECOND OCCURRENCE VALUE IS LESS THAN THE FIRST OCCURRENCE	The second occurrence value must not be less than the first occurrence value.	Correct the error and recompile.
133	DEPENDING ON DATA-NAME IS INVALID	The specified DEPENDING ON name is either misspelled or does not conform to the naming conventions.	Correct the error and recompile.
134	DEPENDING ON DATA-NAME IS UNDEFINED IN CURRENT RECORD	The DEPENDING ON name must be defined in the same record and appear prior to the item entry that is referencing it.	Correct the error and recompile.
135	A VARIABLE-OCCURRENCE DATA ITEM IS NOT ALLOWED TO BE A SUBORDINATE OF A DATA ITEM WITH AN OCCURS CLAUSE	The specified DEPENDING ON name must not be subordinate to a repeating group.	Correct the error and recompile.
136	USAGE TYPE OF THE DEPENDING ON NAME IS INVALID	Usage type of the DEPENDING ON name must not be DISPLAY or COMP-2.	Correct the error and recompile.
137	INVALID KEY NAME	The specified key name is either the same as a reserved word or does not conform to the naming conventions.	Correct the error and recompile.
138	INVALID IDEX NAME	The specified index name is either the same as a reserved word or does not conform to the naming conventions.	Correct the error and recompile.
139	INDEX NAME IS NOT UNIQUE	Index names must be unique within the subschema.	Correct the error and recompile.

TABLE B-7. DDL SUBSCHEMA COMPILATION AND LIBRARY MAINTENANCE DIAGNOSTICS FOR COBOL AND QUERY UPDATE (Contd)

Error Code	Message	Significance	Action
140	LEVEL NUMBER IS OUT OF RANGE, HAS BEEN READJUSTED TO AN -02- LEVEL	Valid level numbers are 1 through 49, 66, and 88; 88 level numbers are not valid for Query Update subschemas.	If assumed level not acceptable, change to correct level and recompile.
141	DATA DESCRIPTION ENTRY WAS NOT TERMINATED WITH A PERIOD	A data description entry must be terminated with a period.	Correct the error and recompile.
142	ALIAS ENTRY TYPE NOT SPECIFIED	Alias entry type REALM, RECORD, or DATA was not specified; the current alias entry is ignored.	Specify entry type and recompile.
143	DATA IS THE ONLY ENTRY TYPE THAT CAN BE QUALIFIED	Record and realm names must be unique; therefore, there is no need to qualify them.	Correct the error and recompile.
144	UNKNOWN QUALIFIER NAME	The qualifier entry could not be located; the alias clause is ignored.	Correct the error and recompile.
145	parameter INVALID IN THE FOLLOWING CLAUSE	The indicated parameter is invalid.	Correct the error and recompile.
146	TITLE DIVISION NOT SPECIFIED - COMPILATION ABORTED	The Title Divsion is a required entry in the DDL source program. Compilation is terminated on the subschema missing this entry.	Correct the error and compile the uncompiled subschema.
147	SS - KEYWORD MISSING	Reserved word SS in the subschema declaration is missing.	Correct the error and recompile.
148	INVALID SCHEMA OR SUB-SCHEMA NAME	The specified schema or subschema name is either the same as a reserved word or does not conform to the naming conventions.	Correct the error and recompile.
149	WITHIN - KEYWORD MISSING	Reserved word WITHIN is not included in the SS clause.	Correct the error and recompile.
150	UNABLE TO COMPLETE ENTRY - ENTRY WILL BE IGNORED - SEARCHING FOR A PERIOD	Improper coding conventions have been used.	Check entry for missing period, correct the error, and recompile.
151	PERIOD FOUND	A terminating period has been located for the previous entry.	Check previous entry for missing period; correct the error and recompile.
152	TITLE DIVISION TITLE IS INCORRECT	Reserved word DIVISION is either missing or misspelled.	Correct the error and recompile.
153	UNKNOWN SOURCE WORD	The item entry could not be located.	Correct the error and recompile.
154	UNDEFINED KEY	The record key defined in the schema does not appear in the subschema.	Define the appropriate key and recompile.
155	SCHEMA NAME SPECIFIED IN THE SUB-SCHEMA DOES NOT MATCH THE SCHEMA NAME IN THE SCHEMA	The schema name specified in the Title Division must be the same as the name specified in the schema declaration in the schema.	Correct the error and recompile.

TABLE B-7. DDL SUBSCHEMA COMPILATION AND LIBRARY MAINTENANCE DIAGNOSTICS FOR COBOL AND QUERY UPDATE (Contd)

Error Code	Message	Significance	Action
156	REDEFINING A REDEFINES ENTRY IS INVALID, MUST USE THE DATA NAME OF THE ENTRY THAT ORIGINALLY DEFINED THE AREA	A redefined item cannot be referenced in another redefines entry.	Correct the error and recompile.
157	EDITING CHARACTERS NOT ALLOWED IN PICTURE	Editing and insertion characters are not valid in a COBOL subschema PICTURE clause.	Correct the error and recompile.
158	ILLEGAL CHARACTER IN PICTURE	A, X, 9, P, S, and V are the only valid picture characters for a COBOL subschema.	Correct the error and recompile.
159	PICTURE SIZE EXCEEDS MAXIMUM OF 32767 CHARACTERS	The value specified in the picture string exceeds the maximum number of characters allowed.	Correct the error and recompile.
160	SYNTACTICAL ERROR IN PICTURE	An invalid character appears in the picture string.	Correct the error and recompile.
161	EXCESS REPEAT COUNT IN EDITED PICTURE	Maximum character count is 65535 for alphanumeric picture and 30 for numeric picture.	Correct the error and recompile.
162	MAXIMUM OF 63 INSERTION CHARACTERS ALLOWED IN PICTURE	A Query Update subschema picture string contains more than 63 insertion characters.	Correct the error and recompile.
163	NON NUMERIC WITHIN PARENTHESES	Only numeric values are allowed within parentheses in the picture string.	Correct the error and recompile.
164	NUMERIC ITEM GREATER THAN 18	Maximum number of digits that can be specified in a picture string is 18.	Correct the error and recompile.
165	IN THE PRIOR ITEM ENTRY THE USAGE AND/OR PICTURE CLAUSE MUST BE SPECIFIED	Either the USAGE clause or PICTURE clause must be specified for an elementary item.	Correct the error and recompile.
166	INVALID SUBSCRIPT IN CLASS MATRIX	This is an internal error.	Follow site-defined procedures for reporting software errors or operational problems.
167	ALPHABETIC OR ALPHANUMERIC PICTURE NOT ALLOWED WITH USAGE IS COMP, COMP-1, OR COMP-2	Only a numeric picture is allowed if usage is COMP, COMP-1, or COMP-2.	Correct the error and recompile.
168	PICTURE CLAUSE IS NOT ALLOWED WITH USAGE IS INDEX	A PICTURE clause is legal only when usage is computational or display.	Correct the error and recompile.
169	EMPTY SCHEMA DIRECTORY	No Information is in the schema directory.	Recompile the schema.
170	IN THE PRIOR ITEM ENTRY THE PICTURE CLAUSE IS NOT ALLOWED IN A GROUP ITEM	The picture string is ignored.	Correct the error and recompile.
171	INSUFFICIENT FIELD LENGTH, INCREASE YOUR FL, DDL ABORTED	Not enough field length was specified to complete the compilation. The job is aborted.	Use the RFL control statement to increase field length, and recompile.
172	REDEFINED NAME IS EITHER MISSING OR INVALID	The specified redefined name is either the same as a reserved word or does not conform to the naming conventions; the item entry is ignored.	Correct the error and recompile.

TABLE B-7. DDL SUBSCHEMA COMPILATION AND LIBRARY MAINTENANCE DIAGNOSTICS FOR COBOL AND QUERY UPDATE (Contd)

Error Code	Message	Significance	Action
173	REDEFINED NAME DOES NOT EXIST	The original item entry that is to be redefined could not be located; the redefines entry is ignored.	Correct the error and recompile.
174	ORIGINALLY DEFINED ITEM CANNOT CONTAIN AN OCCURS	The original item entry that is to be redefined must not contain an OCCURS; the redefines entry is ignored.	Correct the error and recompile.
175	INVALID LEVEL NUMBER FOR REDEFINES	Level numbers 2 through 49 are the only valid level numbers for redefines.	Correct the error and recompile.
176	LEVEL NUMBER OF DATA-NAME IS NOT IDENTICAL TO DATA-NAME-2	The redefines entry level number must be the same as the original item entry that is being redefined; the level number is replaced with the correct level number.	Change redefines entry level number.
177	REDEFINED ITEM CANNOT CONTAIN AN OCCURS OF VARIABLE DIMENSION	The redefines entry is ignored.	Correct the error and recompile.
178	THE SIZE OF THE REDEFINED ITEM IS NOT THE SAME AS THE ORIGINALLY DEFINED ITEM	The size of the redefined item entry (if group, including all subordinate items) must be the same size as the originally defined item.	Correct the error and recompile.
179	UNABLE TO COPY AREA ENTRIES FROM THE SCHEMA, SCHEMA IS INVALID	The specified schema did not contain valid data, and area entries could not be read.	Correct and recompile the schema.
180	RESERVED WORD -RENAMES- IS MISSING	Reserved word RENAMES is either missing or misspelled; level 66 item entry is ignored.	Correct the error and recompile.
181	UNABLE TO FIND DATA-NAME-2	The original item entry being renamed could not be located; the level 66 item entry is ignored.	Correct the error and recompile.
182	A 66 LEVEL ENTRY CANNOT RENAME A 66 OR 88 LEVEL ITEM	The level 66 item entry is ignored.	Correct the error and recompile.
183	DATA-NAME-1 AND DATA-NAME-3 CANNOT CONTAIN AN OCCURS CLAUSE OR BE SUBORDINATE TO AN ITEM WITH AN OCCURS	The renames entry is ignored.	Correct the error and recompile.
184	NONE OF THE ITEMS WITHIN THE RANGE CAN BE VARIABLE-OCCURRENCE DATA-ITEMS	When two original item entries are specified, a range of item entries is indicated. None of the item entries (from the first through the last) can be repeating with variable occurrence.	Correct the error and recompile.
185	DATA-NAME-2 AND DATA-NAME-3 CANNOT BE THE SAME NAME	When THRU is specified in the renames clause, the first data name of the original item entry must not be the same as the second data name of the original entry.	Correct the error and recompile.
186	DATA-NAME-3 IS DEFINED BEFORE DATA-NAME-2	When THRU is specified, the first data name of the original item entry must be specified before the second data name.	Correct the error and recompile.
187	UNABLE TO FIND DATA-NAME-3	Data-name-3 was not previously defined; the renames entry is ignored.	Correct the error and recompile.

TABLE B-7. DDL SUBSCHEMA COMPILATION AND LIBRARY MAINTENANCE DIAGNOSTICS FOR COBOL AND QUERY UPDATE (Contd)

Error Code	Message	Significance	Action
188	DATA-NAME-3 CANNOT BE SUBORDINATE TO DATA-NAME-2	The renames entry is ignored.	Correct the error and recompile.
189	TARGET DATA ITEM MUST BE AN ELEMENTARY ITEM OR VECTOR	The item entry with which the condition name is associated cannot be a group item.	Correct the error and recompile.
190	KEYWORD VALUE NOT SPECIFIED FOR CONDITION-NAME ENTRY	Reserved word VALUE is either misspelled or missing; level 88 entry is ignored.	Correct the error and recompile.
191	LITERAL IS EITHER MISSING OR INVALID	The literal does not agree with the target data item specifications. The literal is a different data type or is longer than the target item; or the literal specifies a decimal point or sign that does not exist in the target item. The level 88 item is ignored.	Correct the error and recompile.
192	LITERALS ARE NOT IN ASCENDING ORDER	The level 88 item entry is ignored.	Correct the error and recompile.
193	ITEM IS NOT SUBORDINATE TO THE GROUP THAT DEFINED IT AS A KEY	All of the items identified by the data-names in the KEY IS phrase must be subordinate to the group item that is the subject of this entry.	Correct the error and recompile.
194	KEY ITEMS CANNOT CONTAIN AN OCCURS CLAUSE	Key data-names cannot reference an item that contains an OCCURS clause.	Correct the error and recompile.
195	UNABLE TO LOCATE KEY ENTRY	The data-name specified in the KEY clause is either misspelled or not defined as subordinate to the group item that is the subject of this entry.	Correct the error and recompile.
196	LEVEL 66 ITEM ENTRIES MUST BE DEFINED AT THE END OF THE RECORD DESCRIPTION	Level 66 items must be the last items specified in the record entry.	Correct the error and recompile
199	MAXIMUM OF 14 CHARACTERS ARE ALLOWED FOR COMP-1	COMP-1 items are limited to 14 characters.	Correct the error and recompile.
200	SYNCHRONIZED LEFT IS ASSUMED	Informative message only. Compilation is continued and the subschema is created.	Use SYNCHRONIZE clause to specify correct alignment.
201	AN ALIASED NAME MUST BE REFERENCED BY THE ALIAS-NAME SPECIFIED IN THE ALIAS DIVISION	When an alias is assigned, the alias and not the schema-assigned name must be referenced.	Correct the error and recompile.
202	MAXIMUM AREA COUNT EXCEEDED	A maximum of 4095 areas is allowed.	Correct the error and recompile.
203	MAXIMUM RECORD COUNT EXCEEDED	A maximum of 4095 records is allowed.	Correct the error and recompile.
204	MAXIMUM ITEM COUNT EXCEEDED	A maximum of 4095 items can be declared.	Correct the error and recompile.
205	MAXIMUM ITEM COUNT FOR A RECORD EXCEEDED	A maximum of 4095 items per record is allowed.	Correct the error and recompile.
206	FATAL ERRORS OCCURRED--FOLLOWING STATEMENTS IGNORED	Due to the occurrence of one or more fatal errors, subsequent statements are not processed by the compiler.	Correct the error and recompile.

TABLE B-7. DDL SUBSCHEMA COMPILATION AND LIBRARY MAINTENANCE DIAGNOSTICS FOR COBOL AND QUERY UPDATE (Contd)

Error Code	Message	Significance	Action
207	REDEFINES ENTRY DOES NOT IMMEDIATELY FOLLOW ENTRY BEING DEFINED	An entry containing a REDEFINES clause must immediately follow the description of the area being redefined, with no intervening entries that define new storage areas.	Correct the error and recompile.
300	RECORD NOT FOUND IN SCHEMA	The record defined in the subschema is not defined in the schema.	Correct the error and recompile.
301	MAXIMUM OCCURS IN SUBSCHEMA GREATER THAN IN SCHEMA	The number of times a data item can occur in the subschema must be less than or equal to the maximum allowed in the schema; it cannot be greater.	Correct the error and recompile.
302	SUBSCHEMA ITEM DOES NOT APPEAR IN SCHEMA	Only new nonrepeating group data items or new repeating group items created when converting a vector to a repeating group are allowed in the subschema. All other subschema entries must match a schema entry. This diagnostic is also issued when a subordinate data item is specified and its dominant item (group) is not.	Correct the error and recompile.
303	ITEM NOT IN SCHEMA CANNOT HAVE VARIABLE OCCURS	A variable occurrence data item in the subschema must correspond to a variable occurrence data item in the schema.	Correct the error and recompile.
304	KEY ITEM FOR REALM NOT FOUND IN SUBSCHEMA	When an area is defined in both the schema and subschema, a primary key defined in the schema must also be defined in the subschema.	Correct the error and recompile.
305	EMBEDDED PRIMARY KEY FOR AK REALM NOT IDENTICAL	AK keys must be COMP-1 and must be the same type, size, and picture in the schema and the subschema.	Correct the error and recompile.
306	TYPE OF REPEATING ITEM DOES NOT AGREE WITH THE TYPE OF REPEATING ITEM IN SCHEMA	A repeating group or vector of fixed dimension in the schema can correspond only to a repeating group or vector of fixed dimension in the subschema; and a repeating group or vector of variable dimension in the schema can correspond only to a repeating group or vector of variable dimension in the subschema.	Correct the error and recompile.
307	ILLEGAL ITEM CONVERSION	Data type specified in the schema cannot be converted to data type specified in the subschema.	Correct the error and recompile.
308	REALM NOT FOUND IN SCHEMA	The realm-name specified in the subschema cannot be located in the schema.	Correct the error and recompile.
309	DEPENDS ON CLAUSE DOES NOT AGREE WITH SCHEMA	The DEPENDING ON item specified in the schema is not the same DEPENDING ON item in the subschema.	Correct the error and recompile.
310	INSUFFICIENT FL FOR DDL PASS 2	Not enough field length was specified to complete the compilation. The job is aborted.	Use RFL control statement to increase field length.

TABLE B-7. DDL SUBSCHEMA COMPILATION AND LIBRARY MAINTENANCE DIAGNOSTICS FOR COBOL AND QUERY UPDATE (Contd)

Error Code	Message	Significance	Action
311	NEW GROUP ITEM NAME MUST NOT APPEAR IN SCHEMA OR REPEATING GROUPS DO NOT AGREE	New nonrepeating group item names specified in the subschema must be unique and must not appear in the schema. The only time a new repeating group is allowed in the subschema is when a vector in the schema is defined as a part of a repeating group in the subschema.	Correct the error and recompile.
312	HIERARCHY OF SUB-SCHEMA ITEM DIFFERS FROM CORRESPONDING ITEM IN SCHEMA	The dominant item of the subschema item differs from the schema item.	Correct the error and recompile.
313	ITEM item-name DOES NOT QUALIFY AS A VALID CONCATENATED KEY	The concatenated key items in the subschema are not identical to the concatenated key items in the schema or are not in the same order.	Correct the error and recompile.
314	WARNING: SS SIZE GR SCHEMA SIZE - MAY CAUSE TRUNCATION ERRORS AT EXECUTION TIME	The warning message occurs when the size of the subschema item is greater than the size of the corresponding schema item and indicates that truncation errors may occur during CDCS execution time.	Ignore the warning if the size is correctly represented for the application program, or correct the picture size of the subschema item and recompile.
315	SUB-SCHEMA ITEM CANNOT DIFFER FROM SCHEMA ITEM WITH CHECK IS PICTURE OPTION	When the subschema item is defined with a CHECK IS PICTURE clause, it must have identical characteristics (size and data class) as those of the schema item.	Correct the subschema item characteristics and recompile.
316	KEY CHARACTERISTICS NOT THE SAME IN ALL RECORD TYPES FOR AREA aaaaaaa	In an area with multiple record types, key items do not appear in the same position or are not the same size and class in each record type.	Correct the error and recompile.
400	RN DECLARATION IS INCORRECT	Reserved word RN is either misspelled or missing.	Correct the error and recompile.
401	INVALID RELATION NAME	The specified relation name is either the same as a reserved word or does not conform to the naming conventions.	Correct the error and recompile.
402	RELATION NAME IS NOT UNIQUE	The relation name must be unique among all relation and realm names in the subschema.	Correct the error and recompile.
403	UNABLE TO FIND CORRESPONDING RELATION ENTRY IN THE SCHEMA	The relation name specified in the subschema does not appear in the schema.	Correct the error and recompile.
404	AREA parameter TRAVERSED IN THE SCHEMA MUST BE SPECIFIED IN THE RD DIVISION	The indicated area is joined in a relationship within the schema, but is not specified in the subschema RD clause.	Correct the error and recompile.
405	INVALID RECORD NAME	The record name specified in a RESTRICT clause is either the same as a reserved word or does not conform to the naming conventions.	Correct the error and recompile.

TABLE B-7. DDL SUBSCHEMA COMPILATION AND LIBRARY MAINTENANCE DIAGNOSTICS FOR COBOL AND QUERY UPDATE (Contd)

Error Code	Message	Significance	Action
406	UNABLE TO FIND RECORD ENTRY	A record referenced in a RESTRICT clause does not appear in the subschema.	Correct the error and recompile.
407	parameter IS AN INVALID DATA-BASE-IDENTIFIER NAME	The indicated parameter is not a valid identifier name in a RESTRICT clause.	Correct the error and recompile.
408	DATA-BASE-IDENTIFIER parameter IS UNDEFINED	The indicated identifier in a RESTRICT clause does not appear in the subschema.	Correct the error and recompile.
409	DATA-BASE-IDENTIFIER parameter IS UNDEFINED IN THE SCHEMA	The indicated identifier parameter does not appear in the schema.	Correct the error and recompile.
410	CONVERSION NOT POSSIBLE FOR LITERAL	Arithmetic operations cannot be performed when data items are not compatible.	Correct the error and recompile.
411	INVALID USE OF DBI parameter DOES NOT BELONG TO RESTRICTED RECORD	The indicated identifier parameter does not apply to the record referenced in a RESTRICT clause.	Correct the error and recompile.
413	SOURCE DBI DOES NOT HAVE COMPATIBLE DATA REPRESENTATION WITH TARGET DBI	Source and target data base indentifiers in a RESTRICT clause are not compatible. In the schema definition, both must be display coded data or both must be the same type of binary data.	Correct the error and recompile.
414	QUALIFICATION NOT ALLOWED ON RESTRICT DATA-NAMES	A data name appearing in a RESTRICT cannot be qualified; it must be unique within the hierarchy.	Correct the error and recompile.
417	parameter UNKNOWN SOURCE WORD	The indicated parameter in the Relation Division is not recognized because improper coding conventions have been followed.	Correct the error and recompile.
418	ENTRY WAS NOT TERMINATED BY A PERIOD	A terminating period for the previous entry was omitted.	Correct the error and recompile.
419	SEARCHING FOR A PERIOD	Scanning for the terminating period.	Check statement for missing period.
420	PERIOD FOUND, COMPILATION RESUMED	A terminating period has been located for the previous entry.	Check statement for missing or misplaced period.
421	SUBSCRIPT SPECIFIED FOR NON REPEATING ITEM parameter IS INVALID	Subscripts are used for repeating group or elementary items only. The indicated parameter is not a subscriptable item.	Correct the error and recompile.
422	DUE TO FATAL ERRORS COMPILATION IS ABORTED	Too many fatal errors have occurred.	Correct as many errors as possible and recompile.
424	VIRTUAL RESULT AND DECODE DATA-BASE-IDENTIFIERS (IN SCHEMA) CANNOT BE USED AS QUALIFIERS	A qualifier identifier must not be defined in a VIRTUAL RESULT or a DECODING clause in the schema.	Correct the error and recompile.
425	LEVEL 66, LEVEL 88 AND REDEFINED DATA-BASE-IDENTIFIERS CANNOT BE USED AS QUALIFIERS	An identifier cannot be a data item that redefines or renames another data item, or references a level 88 data description entry.	Correct the error and recompile.

TABLE B-7. DDL SUBSCHEMA COMPILATION AND LIBRARY MAINTENANCE DIAGNOSTICS FOR COBOL AND QUERY UPDATE (Contd)

Error Code	Message	Significance	Action
426	OWNER AREA OF THE RESTRICTED RECORD WAS NOT SPECIFIED IN THE RELATION ENTRY (IN THE SCHEMA)	The area associated with a record named in a RESTRICT clause is not reference in the schema relation entry.	Correct the error and recompile.
427	A RELATIONAL OPERATOR MUST BE SPECIFIED	A relational operator is missing or is not one of the six valid operators.	Correct the error and recompile.
428	EXCEEDED FIELD LENGTH, INCREASE YOUR FIELD LENGTH	Not enough field length was specified to complete the compilation. The job is aborted.	Use the RFL control statement to increase field length.
429	RECORD ENTRY parameter WAS PREVIOUSLY SPECIFIED IN RESTRICT CLAUSE	Only one RESTRICT clause can be included for a given record. The indicated parameter is illegal.	Correct the error and recompile.
430	INVALID QUALIFIER NAME	The specified qualifier name is either the same as a reserved word or does not conform to the naming conventions.	Correct the error and recompile.
431	RELATION DIVISON TITLE IS INCORRECT	Reserved words TITLE and DIVISION are either missing or misspelled.	Correct the error and recompile.
432	TITLE STATEMENT WAS NOT TERMINATED BY A PERIOD	A division statement requires a terminating period.	Correct the error and recompile.
433	SKIPPING TO THE NEXT RESTRICT CLAUSE	An invalid RESTRICT clause causes the system to scan for the next RESTRICT clause.	Correct the error and recompile.
434	RESTRICT CLAUSE FOUND	The scan following an invalid RESTRICT clause located another RESTRICT clause.	Correct the error and recompile.
435	COULD NOT FIND ANY MORE RESTRICT CLAUSES FOR THE CURRENT RELATION ENTRY	The scan following an invalid RESTRICT clause did not locate another RESTRICT clause.	Correct the error and recompile.
436	parameter IS AN INVALID SUBSCRIPT	Subscripts must be positive integer constants. The indicated parameter is not valid.	Correct the error and recompile.
437	EXCEEDED THE MAXIMUM NUMBER OF SUBSCRIPTS ALLOWED (3)	A maximum of three subscripts can be declared in the DDL syntax.	Correct the error and recompile.
438	TERMINATING DELIMITER FOR SUBSCRIPT IS MISSING	The closing parenthesis of a subscript is missing.	Correct the error and recompile.
439	THE SUBSCRIPT VALUE EXCEEDED THE OCCURS VALUE FOR parameter	The indicated parameter determines the maximum value for a subscript.	Correct the error and recompile.
440	DID NOT SPECIFY SUBSCRIPT FOR REPEATING ITEM parameter	The indicated parameter requires subscripting.	Correct the error and recompile.
441	MAXIMUM RELATION COUNT EXCEEDED	A maximum of 511 relations can be declared.	Correct the error and recompile.

DDLF SUBSCHEMA COMPILATION AND LIBRARY MAINTENANCE DIAGNOSTICS FOR FORTRAN

The DDLF diagnostic messages that can be issued either during compilation of a FORTRAN subschema source program or during library maintenance operations are listed in table B-8. The library maintenance messages begin with an alphabetic character. The subschema compilation messages begin with an error code between 100 to 449 (these error codes are enclosed in asterisks and written to the source listing directly preceding the message). These messages do not necessarily indicate termination of DDLF compilation; unless otherwise stated compilation continues but a subschema is not created.

TABLE B-8. DDLF SUBSCHEMA COMPILATION AND LIBRARY MAINTENANCE DIAGNOSTICS FOR FORTRAN

Error Code	Message	Significance	Action
	DID NOT LOCATE SUB-SCHEMA TO BE REPLACED---NEW SUB-SCHEMA HAS BEEN ADDED	The subschema to be replaced in the library could not be located; the current subschema has been added to the library.	Specify correct library or subschema and recompile.
	DID NOT LOCATE parm, PURGE NOT POSSIBLE	The subschema name specified for the purge could not be located in the subschema library.	Specify the correct subschema name and recompile.
	EMPTY INPUT FILE---PURGE NOT POSSIBLE	The input file contains no subschemas. The input file might have been purged prior to this operation.	If the input file is in error, create a new file.
	EMPTY SUB-SCHEMA FILE, DDLF ABORTED	The subschema library contains no subschemas; issued during a schema compaction, audit, or purge. The subschemas might have been purged prior to this compilation.	If the subschemas have been purged, create a new library.
	ILL-FORMATTED LIBRARY--NOT UPDATABLE, DDL ABORTED	The subschema library contains an error and cannot be updated. DDLF terminates.	Check for empty library or file that is not a library. Correct the error and recompile.
	OLD SUB-SCHEMA FILE BAD, SUBSCHEMA LENGTH IS ZERO. DOLF ABORTED	The subschema library contains a subschema of length zero, indicating a bad library file; issued during a subschema compaction. DDLF terminates.	Re-create the library.
	SUB-SCHEMA WITH THE SAME NAME AS THE NEW SUB-SCHEMA ALREADY EXISTS---FILE NOT UPDATED	The current subschema could not be added to the library since the library contains a subschema with the same name.	Change the subschema name and recompile.
	WARNING---EMPTY SUBSCHEMA FILE	The subschema library contains no subschemas. The subschemas might have been purged prior to this compilation. The new subschema is added to the library and is the only subschema in the library.	If subschemas have been purged, create a new library.
099	SOURCE WORD LONGER THAN 255 CHARACTERS, UNABLE TO CONTINUE COMPILATION - DDLF ABORTED	The compiler is unable to interpret the input stream.	Correct the error and recompile.
100	EMPTY INPUT FILE	The input file is empty and the compilation is terminated.	Create a new file and recompile.
107	INVALID NAME IN ALIAS STATEMENT	The specified name does not conform to the naming conventions.	Correct the error and recompile.

TABLE B-8. DDLF SUBSCHEMA COMPILATION AND LIBRARY MAINTENANCE DIAGNOSTICS FOR FORTRAN (Contd)

Error Code	Message	Significance	Action
108	EQUAL SIGN MISSING	The syntax rules for this statement require an equal sign.	Correct the error and recompile.
111	INVALID QUALIFIER NAME	The specified qualifier name does not conform to the naming conventions.	Correct the error and recompile.
114	INVALID STATEMENT - REALM OR ALIAS STATEMENT EXPECTED	If ALIAS statements are used, they must immediately follow the SUBSCHEMA statement and precede any REALM statement. If ALIAS statements are not used, the REALM statements must immediately follow the SUBSCHEMA statement.	Correct the error and recompile.
115	INVALID REALM NAME	The specified name does not conform to the naming conventions.	Correct the error and recompile.
116	DUPLICATE REALM NAME	All realm names must be unique.	Correct the error and recompile.
119	RECORD STATEMENT NOT SPECIFIED, UNABLE TO CONTINUE - COMPILATION ABORTED	The subschema must include at least one RECORD statement. Compilation is terminated.	Correct the error and recompile.
120	INVALID RECORD NAME	The specified record name does not conform to the naming conventions.	Correct the error and recompile.
121	DUPLICATE RECORD NAME	All record names must be unique.	Correct the error and recompile.
122	CANNOT LOCATE OWNER REALM IN THE SCHEMA	The schema area in which this record is defined is not specified in a subschema REALM statement.	Correct the error and recompile.
124	INVALID ITEM NAME	The specified item name does not conform to the naming conventions.	Correct the error and recompile.
125	ITEM NAME NOT UNIQUE	All item names must be unique.	Correct the error and recompile.
126	ITEM SIZE GREATER THAN MAXIMUM SIZE ALLOWED	The maximum size allowed for a CHARACTER item is 32767 characters.	Correct the error and recompile.
127	INVALID ITEM LENGTH	The length for a CHARACTER item must be specified in the form *len, where len is a positive integer.	Correct the error and recompile.
128	LENGTH SPECIFIED FOR NON-CHARACTER ITEM	The specification for length, *len (where len is an integer), is allowed only for type CHARACTER items.	Correct the error and recompile.
130	INVALID DIMENSION BOUND	The specified dimension value is not an integer.	Correct the error and recompile.

TABLE B-8. DDLF SUBSCHEMA COMPILATION AND LIBRARY MAINTENANCE DIAGNOSTICS FOR FORTRAN (Contd)

Error Code	Message	Significance	Action
131	THE UPPER DIMENSION BOUND IS LESS THAN THE LOWER BOUND	When the user specifies a dimension of an array, the value of the upper bound must be greater than or equal to the value of the lower bound.	Correct the error and recompile.
133	EXCEEDED THE MAXIMUM NUMBER OF DIMENSIONS ALLOWED	In a subschema for a FORTRAN 4 program, an array can have a maximum of three dimensions; in a subschema for a FORTRAN 5 program, an array can have a maximum of seven dimensions.	Correct the error and recompile.
138	DOUBLE PRECISION MUST BE SPECIFIED IN FORTRAN 5	The keyword PRECISION must be specified in the type statement for a double precision item when the F5 parameter is specified in the DDLF control statement.	Correct the error and recompile.
142	ALIAS ENTRY TYPE NOT SPECIFIED	Alias entry type REALM, RECORD, or ITEM was not specified; the current entry is ignored.	Specify entry type and recompile.
143	AN ITEM IS THE ONLY ALIAS TYPE THAT CAN BE QUALIFIED	Record and realm names must be unique; therefore, there is no need to qualify them.	Correct the error and recompile.
144	UNKNOWN QUALIFIER NAME	The qualifier entry could not be located; the ALIAS statement is ignored.	Correct the error and recompile.
145	parm INVALID IN THE FOLLOWING STATEMENT	The indicated parameter is invalid.	Correct the error and recompile.
146	SUBSCHEMA STATEMENT NOT SPECIFIED - COMPILATION ABORTED	The SUBSCHEMA statement is required in every FORTRAN DDLF source program. Compilation is terminated.	Correct the error and recompile.
147	SCHEMA KEYWORD MISSING	The keyword SCHEMA is required in the SUBSCHEMA statement.	Correct the error and recompile.
148	INVALID SCHEMA OR SUBSCHEMA NAME	The specified schema or subschema name does not conform to the naming conventions.	Correct the error and recompile.
155	SCHEMA NAME SPECIFIED IN THE SUB-SCHEMA DOES NOT MATCH THE SCHEMA NAME IN THE SCHEMA	The schema name specified in the SUBSCHEMA statement must be the same as the name specified in the schema declaration in the schema.	Correct the error and recompile.
166	INVALID SUBSCRIPT IN CLASS MATRIX	This is an internal error.	Follow site-defined procedures for correcting software errors or operational problems.
169	EMPTY SCHEMA DIRECTORY	No information in the schema directory.	Recompile the schema.
171	INSUFFICIENT FIELD LENGTH - INCREASE YOUR FL --- DDL ABORTED	Not enough field length was specified to complete the compilation. The job is aborted.	Use the RFL control statement to increase field length.

TABLE B-8. DDLF SUBSCHEMA COMPILATION AND LIBRARY MAINTENANCE DIAGNOSTICS FOR FORTRAN (Contd)

Error Code	Message	Significance	Action
179	UNABLE TO COPY AREA ENTRIES FROM THE SCHEMA, SCHEMA IN INVALID	The specified schema did not contain valid data, and area entries could not be read.	Correct and recompile the schema.
200	RECORD STRUCTURE CAUSES MAXIMUM NUMBER OF COMMON BLOCKS TO BE EXCEEDED	The maximum number of common blocks allowed is 500. FORTRAN DDL generates common blocks sequentially according to the subschema source program: one common block is generated for each area; an additional common block is generated each time FORTRAN DDL encounters a data type incompatible with the previous data type. (Character data is incompatible with all other data types.)	Group data items of the same type together within a record to minimize the number of common blocks generated, and recompile.
201	AN ALIASED NAME MUST BE REFERENCED BY THE ALIAS-NAME SPECIFIED IN THE ALIAS STATEMENT	When an alias is assigned, the alias and not the schema-assigned name must be referenced.	Correct the error and recompile.
202	MAXIMUM AREA COUNT EXCEEDED	A maximum of 4095 areas is allowed.	Correct the error and recompile.
203	MAXIMUM RECORD COUNT EXCEEDED	A maximum of 4095 records is allowed.	Correct the error and recompile.
204	MAXIMUM ITEM COUNT EXCEEDED	A maximum of 4095 items can be declared.	Correct the error and recompile.
205	MAXIMUM ITEM COUNT FOR A RECORD EXCEEDED	A maximum of 4095 items per record is allowed.	Correct the error and recompile.
206	FATAL ERRORS OCCURRED -- FOLLOWING STATEMENTS IGNORED	Due to the occurrence of one or more fatal errors, subsequent statements are not processed by the compiler.	Correct the error and recompile.
210	A RECORD HAS BEEN PREVIOUSLY DEFINED FOR THIS REALM	Only one record can be defined for a realm.	Correct the error and recompile.
211	NO DATA ITEMS DEFINED FOR THIS RECORD	At least one data item must be defined for each record.	Correct the error and recompile.
212	INVALID STATEMENT - END STATEMENT EXPECTED	Only an END statement is allowed at this position in the input stream.	Correct the error and recompile.
213	INVALID STATEMENT FOLLOWING END STATEMENT	A SUBSCHEMA statement is the only statement that can follow an END statement.	Correct the error and recompile.
214	NO END STATEMENT	The last statement in a FORTRAN DDL program must be an END statement.	Correct the error and recompile.
215	COMMA OR EQUAL MISSING	The syntax rules for this statement require a comma or equal sign.	Correct the error and recompile.
216	STATEMENT CONTAINS EXTRANEOUS INFORMATION	Data was found beyond the end of the statement.	Correct the error and recompile.

TABLE B-8. DDLF SUBSCHEMA COMPILATION AND LIBRARY MAINTENANCE DIAGNOSTICS FOR FORTRAN (Contd)

Error Code	Message	Significance	Action
217	INTERNAL DDLF ERROR	This is an internal error.	Follow site-defined procedures for reporting software errors or operational problems.
218	TYPE CHARACTER IS NOT VALID IN FORTRAN 4	An item cannot be defined as type CHARACTER if the F4 parameter is specified on the DDLF control statement.	Change usage to one permitted in FORTRAN 4, and recompile.
219	THE LOW:HIGH BOUNDS FEATURE IS NOT VALID IN FORTRAN 4	The low:high bounds feature is allowed only in FORTRAN 5. In FORTRAN 4, the low bound of an array dimension is assumed to be 1; the user specifies only the high bound.	Correct the error and recompile.
220	TYPE BOOLEAN IS NOT VALID IN FORTRAN 4	An item cannot be described as type BOOLEAN if F4 is specified on the DDLF control statement. Type BOOLEAN is allowed only in FORTRAN 5.	Correct the error and recompile.
300	RECORD parm NOT FOUND IN SCHEMA	The record named is not defined in the schema.	Correct the error and recompile.
301	MAXIMUM ARRAY ELEMENTS IN SUB-SCHEMA GREATER THAN IN SCHEMA	The dimension value specified in the subschema must be less than or equal to the maximum occurs value in the schema.	Correct the error and recompile.
302	SUBSCHEMA ITEM DOES NOT APPEAR IN SCHEMA	The item specified in the subschema is not defined in the schema.	Correct the error and recompile.
304	KEY ITEM FOR REALM parm NOT FOUND IN SUBSCHEMA	The primary key must be defined for every realm in the subschema.	Correct the error and recompile.
306	TYPE OF REPEATING ITEM DOES NOT AGREE WITH TYPE OF REPEATING ITEM IN SCHEMA	An array must correspond to a vector in the schema. A variable must correspond to a nonrepeated data item in the schema.	Correct the error and recompile.
307	ILLEGAL ITEM CONVERSION	The data type specified in the schema cannot be converted to the data type specified in the subschema.	Correct the error and recompile.
308	REALM parm NOT FOUND IN SCHEMA	The realm name specified in the subschema cannot be located in the schema.	Correct the error and recompile.
310	INSUFFICIENT FL FOR DDLF PASS 2	Not enough field length was specified to complete the compilation. The job is aborted.	Use the RFL control statement to increase field length.
312	HIERARCHY OF SUB-SCHEMA ITEM DIFFERS FROM CORRESPONDING ITEM IN SCHEMA	A subschema item corresponds to a schema item that is part of a repeating group. Items in a repeating group cannot be referenced in FORTRAN DDL.	Convert schema repeating group into vectors or omit item from subschema, and recompile.

TABLE B-8. DDLF SUBSCHEMA COMPILATION AND LIBRARY MAINTENANCE DIAGNOSTICS FOR FORTRAN (Contd)

Error Code	Message	Significance	Action
313	ITEM item-name DOES NOT QUALIFY IN A VALID CONCATENATED KEY	The indicated item in the subschema is not identical in name, size, or type to the concatenated key item in the schema; or the item is not in the same order.	Correct the error and recompile.
314	WARNING: SS SIZE GR SCHEMA SIZE - MAY CAUSE TRUNCATION ERRORS AT EXECUTION TIME	An execution error can result from nonzero or nonblank truncation.	Increase schema size of item, or ensure that the data is within the schema-defined limit.
315	SUBSCHEMA ITEM CANNOT DIFFER FROM SCHEMA ITEM WITH CHECK IS PICTURE OPTION	The CHECK IS PICTURE option in the schema disallows data conversion.	Refer to the CHECK IS PICTURE option for type requirements; correct the error and recompile.
316	THE PRIMARY CONCATENATED KEY ITEM item-name NOT FOUND IN SUBSCHEMA	A constituent data item of a primary concatenated key was not defined in the subschema description.	Correct the error and recompile.
401	INVALID RELATION NAME	The specified relation name does not conform to the naming conventions.	Correct the error and recompile.
402	RELATION NAME NOT UNIQUE	The relation name must be unique among all relation and realm names in the subschema.	Correct the error and recompile.
403	UNABLE TO FIND CORRESPONDING RELATION ENTRY IN THE SCHEMA	The relation name specified in the subschema does not appear in the schema.	Correct the error and recompile.
404	AREA parm TRAVERSED IN THE SCHEMA MUST BE SPECIFIED IN A REALM STATEMENT	The indicated area is joined in a relationship within the schema, but is not specified in a REALM statement.	Correct the error and recompile.
405	INVALID RECORD NAME	The record name specified in a RESTRICT statement does not conform to the naming conventions.	Correct the error and recompile.
406	UNABLE TO FIND RECORD ENTRY	A record referenced in a RESTRICT statement does not appear in the subschema.	Correct the error and recompile.
407	parm IS AN INVALID DATA-BASE-IDENTIFIER NAME	The indicated parameter is not a valid data base item in a RESTRICT statement.	Correct the error and recompile.
408	DATA-BASE-IDENTIFIER parm IS UNDEFINED	The indicated data base item in a RESTRICT statement does not appear in the subschema.	Correct the error and recompile.
409	DATA-BASE-IDENTIFIER parm IS UNDEFINED IN THE SCHEMA	The indicated identifier parameter does not appear in the schema.	Correct the error and recompile.
410	CONVERSION NOT POSSIBLE FOR LITERAL	Arithmetic operations cannot be performed on a literal.	Correct the error and recompile.
411	INVALID USE OF DBI parm - DOES NOT BELONG TO RESTRICTED RECORD	The indicated data base item does not apply to the record referenced in a RESTRICT statement.	Correct the error and recompile.

TABLE B-8. DDLF SUBSCHEMA COMPILATION AND LIBRARY MAINTENANCE DIAGNOSTICS FOR FORTRAN (Contd)

Error Code	Message	Significance	Action
413	SOURCE DBI DOES NOT HAVE A COMPATIBLE SCHEMA DATA REPRESENTATION WITH THE TARGET DBI	Source and target data base items used in a RESTRICT statement must be compatible as determined from the schema data class of each item. Items of schema data classes 0 through 4 (items stored as display code) are compatible. An item of any other schema data class is compatible only with an item of the same schema data class.	Correct the error and recompile.
417	INVALID STATEMENT - END STATEMENT EXPECTED	Only an END statement is allowed at this position in the input stream.	Correct the error and recompile.
418	PERIOD MISSING	A period was expected in this statement and was not found.	Correct the error and recompile.
419	EXTRANEOUS DATA IN STATEMENT	Data was found beyond the end of the statement.	Correct the error and recompile.
420	NUMBER OF DIMENSIONS SPECIFIED DOES NOT EQUAL THE NUMBER OF DIMENSIONS DEFINED FOR ARRAY parm	The number of dimensions specified for the item in the RESTRICT clause does not conform to the subschema description of the item.	Correct the error and recompile.
421	SUBSCRIPT SPECIFIED FOR ITEM WHICH IS NOT AN ARRAY	A subscript is specified for an item that is not defined as an array in the subschema.	Correct the error and recompile.
422	DUE TO FATAL ERRORS, COMPILATION IS ABORTED	Too many fatal errors have occurred.	Correct as many errors as possible and recompile.
426	OWNER AREA OF THE RESTRICTED RECORD WAS NOT SPECIFIED IN THE RELATION ENTRY (IN THE SCHEMA)	The area associated with a record named in a RESTRICT statement is not referenced in the schema relation entry.	Correct the error and recompile.
427	INVALID OR MISSING RELATIONAL OPERATOR	A relational operator is missing or is not one of the six valid operators.	Correct the error and recompile.
428	EXCEEDED FIELD LENGTH, INCREASE YOUR FIELD LENGTH	Not enough field length was specified to complete the compilation. The job is aborted.	Use the RFL control statement to increase field length.
429	RECORD ENTRY parm WAS PREVIOUSLY SPECIFIED IN A RESTRICT CLAUSE	Only one RESTRICT statement can be included for a given record. The indicated parameter is not allowed.	Correct the error and recompile.
430	INVALID LOGICAL OPERATOR	A logical operator is missing or is not one of the allowed operators.	Correct the error and recompile.
431	NO END STATEMENT	The last statement in a FORTRAN DDL program must be an END statement.	Correct the error and recompile.

TABLE B-8. DDLF SUBSCHEMA COMPILATION AND LIBRARY MAINTENANCE DIAGNOSTICS FOR FORTRAN (Contd)

Error Code	Message	Significance	Action
432	INVALID STATEMENT FOLLOWING END STATEMENT	A SUBSCHEMA statement is the only statement that can follow an END statement.	Correct the error and recompile.
433	SKIPPING TO THE NEXT RESTRICT STATEMENT	An invalid RESTRICT statement causes the system to scan for the next RESTRICT statement.	Correct the error and recompile.
434	RESTRICT STATEMENT FOUND	The scan following an invalid RESTRICT statement located another RESTRICT statement.	Correct the error and recompile.
435	COULD NOT FIND ANY MORE RESTRICT STATEMENTS FOR THE CURRENT RELATION ENTRY	The scan following an invalid RESTRICT statement did not locate another RESTRICT statement.	Correct the error and recompile.
436	parm IS AN INVALID SUBSCRIPT	Subscripts must be positive integer constants. The indicated parameter is not valid.	Correct the error and recompile.
438	TERMINATING DELIMITER FOR SUBSCRIPT IS MISSING	The closing parenthesis of a subscript is missing.	Correct the error and recompile.
439	THE SUBSCRIPT VALUE IS NOT WITHIN THE BOUNDS OF ARRAY DECLARATION	The subscript specified is greater than the declared upper bound or less than the declared lower bound.	Correct the error and recompile.
440	DID NOT SPECIFY SUBSCRIPT FOR ARRAY parm	The indicated parameter requires subscripting.	Correct the error and recompile.
441	MAXIMUM RELATION COUNT EXCEEDED	A maximum of 4095 relations can be declared.	Correct the error and recompile.
442	LEFT PARENTHESIS MISSING	Left-right parentheses must match.	Correct the error and recompile.
443	RIGHT PARENTHESIS MISSING	Left-right parentheses must match.	Correct the error and recompile.

GLOSSARY C

The glossary contains terms unique to the description of the components of DMS-170 and terms common within the data processing industry that have special connotations within the context of DMS-170.

AAM -
See Advanced Access Methods.

Access Control -
Protection of data from unauthorized access or modification. Also called privacy.

Access Control Key -
The value an application program must supply to CDCS in order to gain access to a particular data base area. Also called a privacy key.

Access Control Lock -
The value associated with a data base area which must be known by the application program if the program is to gain access to the area.

Actual Item -
An item for which the value is materialized when the record is stored or updated. The value is determined by a data base procedure and is physically stored in the record.

Actual Key -
A file organization in which records are identified by system-assigned keys.

Advanced Access Methods (AAM) -
A file manager that processes indexed sequential, direct access, and actual key file organizations and supports the Multiple-Index Processor. See CYBER Record Manager.

After-Image -
A copy of a data base record after it has been updated. Contrast with Before-Image.

Alias -
A data name used in a COBOL or Query Update subschema in place of a schema data name; a data item used in a FORTRAN subschema in place of a schema data name.

Alphanumeric -
Any character in the computer character set defined in appendix A.

Alternate Key -
A data item for which the value can be used to randomly access a record in a CRM file.

Application Program -
A COBOL program, a FORTRAN program, or a Query Update application that interfaces with CDCS. The FORTRAN source program must have DML statements that provide for the CDCS interface.

Area -
A uniquely named schema data base subdivision that contains data records; identified in the subschema as a realm; associated in the master directory with at least one permanent file.

Array -
A data item consisting of a set of elements of the same type that is defined by a single name; FORTRAN subschema data structure. See Elementary Item.

Attach -
The process of making a permanent file accessible to a job by specifying the proper permanent file identification and passwords.

Automatic Recovery -
CDCS initiated recovery operations that make a data base usable and consistent after some type of software or hardware failure.

Backup Dump -
A copy of all or selected portions of a data base, which is produced on a regularly scheduled basis for the explicit purpose of data base recovery.

BAM -
See Basic Access Methods.

Basic Access Methods (BAM) -
A file manager that processes sequential and word addressable file organizations. See CYBER Record Manager.

Batch Test Facility -
An absolute program residing on the CDCS system library that allows CDCS to run at the same control point as a user program.

Before-Image -
A copy of a data base record before it has been updated. Contrast with After-Image.

Beginning-of-Information (BOI) -
As defined by CRM, the start of the first user record in a file. System-supplied information, such as an index block or control word, does not affect beginning-of-information. Any label on a tape exists prior to beginning-of-information.

Block -
On tape, information between interrecord gaps on a tape. CRM defines several blocks depending on file organization, as shown in table C-1.

TABLE C-1. BLOCK TYPES

Organization	Blocks
Indexed sequential	Data block; index block
Direct access	Home block; overflow block
Actual key	Data block
Sequential	Block type I, C, K, E

Cascade Effect -
A phenomenon causing the indirect propagation of erroneous data in a data base. Erroneous data is processed by a properly functioning program and made part of another previously correct record.

CDCS -
See CYBER Database Control System.

Checksum -
A one-word attribute generated by DDL for each area and relation in a schema and for each subschema. Checksums are stored in the schema and subschema directories, and in the master directory. CDCS references them to check the validity of using a previously compiled subschema with the current schema or of using a previously compiled application program with a current subschema.

Child Record Occurrence -
A record occurrence that has another record occurrence (the parent record occurrence) at the next numerically lower rank in a hierarchical tree structure of the relation. Contrast with Parent Record Occurrence.

Compression -
The process of condensing a record to reduce the amount of storage space required. Compression can be performed by either a system-supplied or a user-supplied routine. Contrast with Decompression.

Concurrency -
Simultaneous access to the same data in a data base by two or more application programs during a given span of time.

Condensed Schema/Subschema Table (CST) -
A table comprised of information from the schema and subschema directories. A CST is part of the master directory and is generated for every schema and subschema combination.

Constant -
A fixed value, explicitly written in a source statement. In a FORTRAN subschema, the term corresponds to a literal in the schema.

Constraint -
A control imposed on records in interdependent files or on items in a single file for the purpose of protecting the integrity of data in a data base during update operations. A constraint is defined in the schema and is based on common data items in the records or within a record.

Control Break -
A condition during a relation read which signifies that a new record occurrence was read for the parent file.

Control Word -
A system-supplied word that precedes each W type record in storage.

Conversion -
The process of changing data characteristics between the schema and the subschema.

CRM -
See CYBER Record Manager.

CYBER Database Control System (CDCS) -
The DMS-170 controlling module that provides the interface between an application program and a data base.

CYBER Record Manager (CRM) -
A generic term relating to the common products BAM and AAM, which run under the NOS and NOS/BE operating systems and allow a variety of record types, blocking types, and file organizations to be created and accessed. The execution time input/output of COBOL, FORTRAN, Sort/Merge 4, Sort/Merge 5, ALGOL, and the DMS-170 products is implemented through CRM. Neither the input/output of the NOS or NOS/BE operating systems themselves nor any of the system utilities such as COPY or SKIPF is implemented through CRM. All CRM file processing requests ultimately pass through the operating system input/output routines.

Data Administrator -
A person or a group who defines the format and organization of a data base and is responsible for maintaining and monitoring a data base.

Data Base -
A systematically organized, central pool of information; organization is described by a schema.

Data Base Procedure -
A special-purpose routine that performs a predefined operation; its use is specified in a schema and is initiated by CDCS.

Data Base Status Block -
An area of memory defined within an application program to which CDCS returns information concerning the status of operations on data base files and relations. The status block is updated after each CDCS operation.

Data Base Transaction -
See Transaction.

Data Base Version -
A set of data files that is described by a schema. Data base versions are defined in the master directory. When data base versions are used, a schema (the description of the data base) can be used with more than one set of files (each set of files being a data base version).

Data Description Language (DDL) -
The language used to structure a schema and a subschema.

Data Integrity -
Validity of data. Checking the validity of data items on a data base update can be performed by means of the CHECK clause or data base procedures.

Data Item -
Smallest unit of data within a record; can be an elementary or group data item (for FORTRAN, an elementary item only).

Data Manipulation Language (DML) -
A language, patterned after application language statements, that provides access to the data base. In the DMS-170 environment, DML is a part of COBOL statements; FORTRAN DML statements must be incorporated in a FORTRAN source program and translated by the DML preprocessor into statements acceptable to the FORTRAN compiler.

Data Name -
Used in a schema, in a COBOL or Query Update subschema, and in a COBOL or Query Update application program; a name identifying a group or elementary data item in the data base; can contain up to 30 letters, digits, or embedded hyphens, but must contain at least one letter. Similar to item name.

Deadlock -
A situation that arises in concurrent data base access when an application programs is contending for a resource that is locked by another program, and neither program can proceed without that resource.

Decompression -
The process of expanding a compressed record to restore it to its original size. The user can supply a decompression routine or use a system-supplied routine. Contrast with Compression.

Dependent Record Occurrence -
A record occurrence that is the dependent member of a condition defined by a constraint. Contrast with Dominant Record Occurrence.

Direct Access -
In the context of CRM, one of the five file organizations. It is characterized by the system hashing of the unique key within each file record to distribute records randomly in blocks called home blocks of the file.

In the context of NOS permanent files, a direct access file is a file that is accessed and modified directly, as contrasted with an indirect access permanent file.

Directed Relationship -
The logical relational structure that defines the specific order in which the files in a relation are traversed and the order in which the record occurrences are retrieved. The relational structure is formed by the join terms declared in the schema.

Directory -
A file that contains area and record attributes of the data base; created when the schema or subschema is compiled; an object schema or subschema.

Dominant Record Occurrence -
A record occurrence that is the dominant member of a condition defined by a constraint. Contrast with Dependent Record Occurrence.

Duration Loading -
A procedure that allows certain CDCS overlay capsules and CRM capsules to be loaded during CDCS initialization and kept in memory during the entire execution time of CDCS. Performance is improved in situations of heavy usage by avoiding memory fragmentation and the extra time that occurs when the capsules are each loaded as they are used.

Elementary Item -
A data item that is not subdivided into other data items. An elementary item that is part of a group item has the highest level number in the group item. A nonrepeating elementary item in the schema corresponds to a variable in a FORTRAN subschema; a repeating elementary item corresponds to an array.

End-of-Information (EOI) -
Defined by CRM in terms of the file organization and file residence as shown in table C-2.

TABLE C-2. END-OF-INFORMATION BOUNDARIES

File Organization	File Residence	Physical Position
Sequential	Mass storage	After the last user record.
	Labeled tape in SI, I, S, or L format	After the last user record and before any file trailer labels.
	Unlabeled tape in SI or I format	After the last user record and before any file trailer labels.
	Unlabeled tape in S or L format	Undefined.
Word Addressable	Mass storage	After the last word allocated to the file, which might be beyond the last user record.
Indexed Sequential, Actual Key	Mass storage	After the record with the highest key value.
Direct Access	Mass storage	After the last record in the most recently created overflow block or home block with the highest relative address.

Figurative Constant -
A fixed value with a predefined name.

File -
A collection of records treated as a unit; an area in the schema; a realm in the subschema.

Fixed Occurrence Data Item -
A data item that is repeated the same number of times in all records.

Floating Point Literal -
A string of digits with a decimal point and an optional exponent.

Flushing -
A process of force writing log file and data buffers.

FORM -
A general-purpose file management utility for manipulating records and creating and converting files.

FORTRAN DML -
See Data Manipulation Language.

Group Item -
A data item that is subdivided into other data items; a collection of data items. Group items cannot be referenced in a FORTRAN subschema.

Hierarchical Tree Structure -
A representation that commonly illustrates record occurrences for files joined in a directed relation. The root of the tree is a record occurrence in the root file and each successive level represents the record occurrences in each joined file.

Home Block -
Mass storage allocated for a file with direct access organization at the time the file is created.

Identifier -
A data name that is referenced uniquely through a combination of subscripts and qualifiers; used in a schema and COBOL and Query Update subschemas.

Indexed Sequential -
A file organization in which records are stored in ascending order by key.

Interrelated Files -
Those data base files that are connected through a relation defined in the schema using join terms.

Invocation -
Preparation by CDCS to handle input/output requests from application programs. The invocation call is generated by the COBOL compiler, by the FORTRAN DML INVOKE statement, or by Query Update when executing particular directives. The invocation call must occur prior to any other CDCS processing requests.

Item Name -
Used in a FORTRAN subschema or application program; a name identifying an elementary item in the data base; can contain up to seven letters or digits; must begin with a letter. Similar to data name.

Join Terms -
The identifiers that are used to join two files in a relation.

Joining Files -
The logical linkage of one file to another in a relation through the use of data items called join terms or identifiers.

Journal Log File -
An independent sequential permanent file (not a data base area) assigned for the purpose of collecting designated information to be used to reconstruct or restore a data base.

Keyword -
A reserved word that is required in a source program clause of a schema, of a COBOL or Query Update subschema, or as input to a data base utility; a word that is required in a source program statement of a FORTRAN subschema.

Level -
For system-logical-records, an octal number 0 through 17 in the system-supplied 48-bit marker that terminates a short or zero-length PRU.

Level Number -
Used in a schema and in a COBOL or Query Update subschema; a number defining the structure of data within a record; if not specified in data description entry in a schema, level number 01 is assumed by default.

Literal -
A constant completely defined by its own identity; called a constant in a FORTRAN subschema.

Local File Name-
The 1 to 7 display code alphabetic or numeric characters by which the operating system recognizes a file. Every local file name in a job must be unique and begin with a letter.

Logging -
The facility of CDCS through which historical records are kept of operations performed by users on data base areas. Logging information is used in data base recovery and restoration operations.

Logical Record -
Under NOS, a data grouping that consists of one or more PRUs terminated by a short PRU or zero-length PRU. Equivalent to a system-logical-record under NOS/BE.

Mapping -
The process by which CDCS produces a record or item image conforming to the schema or subschema description.

Master Directory -
A file containing information used by CDCS in processing. This information consists of schema and subschema tables, media parameters, and data base procedure library and logging specifications.

Nested Group Item -
A group item that is subordinate to another group item. Up to three levels of nested groups can be specified in a schema.

Noise Record -
The number of characters the tape drivers discard as being extraneous noise rather than a valid record. Value depends on installation settings.

Nonrepeating Group Item -
A COBOL subschema data item that contains subordinate data items. This group item occurs only once in each record occurrence; used to identify a series of related data items.

Null Record Occurrence -
A record occurrence composed of the display code right bracket symbol in each character position. It is used in a relation occurrence to denote that no record occurrence qualifies or that a record occurrence does not exist at a given level in the relation.

Operation -
A particular function performed on units of data; for instance, opening or closing an area, or storing or deleting a record.

Overflow Block -
Mass storage the system adds to a file with direct access organization when records cannot be accommodated in the home block.

Overlay Capsules -
A special type of capsule called OVCAP, designed for use with overlays, and called into memory by an overlay (see Duration Loading).

Parent Record Occurrence -
A record occurrence that has another record occurrence at the next numerically higher rank in a hierarchical tree structure of the relation. Contrast with Child Record Occurrence.

Partition -
As defined by CRM, a division within a file with sequential organization. Generally, a partition contains several records or sections. Implementation of a partition boundary is affected by file structure and residence, as shown in table C-3.

Notice that in a file with W type records, a short PRU of level 0 terminates both a section and a partition.

Permanent File -
A file on a mass storage permanent file device that is protected against accidental destruction by the system and can be protected against unauthorized access or destruction.

Physical Record Unit (PRU) -
Under NOS and NOS/BE, the amount of information transmitted by a single physical operation of a specified device (see table C-4).

A PRU that is not full of user data is called a short PRU; a PRU that has a level terminator but no user data is called a zero-length PRU.

Primary Key -
A key that must be defined for an indexed sequential, direct access, or actual key file when the file is created. Random access of a record depends on the use of the primary key.

Privacy -
See Access Control.

TABLE C-3. PARTITION BOUNDARIES

Device	Record Type (RT)	Block Type (BT)	Physical Boundary
PRU device	W	I	A short PRU of level 0 containing a one-word deleted record pointing back to the last I block boundary, followed by a control word with a flag indicating a partition boundary.
	W	C	A short PRU of level 0 containing a control word with a flag indicating a partition boundary.
	D,F,R, T,U,Z	C	A short PRU of level 0 followed by a zero-length PRU of level 17 octal.
	S	-	A zero-length PRU of level number 17 octal.
S or L format tape	W	I	A separate tape block containing as many deleted records of record length 0 as required to exceed noise record size, followed by a deleted one-word record pointing back to the last I block boundary, followed by a control word with a flag indicating a partition boundary.
	W	C	A separate tape block containing as many deleted records of record length 0 as required to exceed noise record size, followed by a control word with a flag indicating a partition boundary.
	D,F,T, R,U,Z	C,K,E	A tapemark.
	S	-	A tapemark.
Any other tape format	-	-	Undefined.

TABLE C-4. PRU SIZES

Device	Size in Number of 60-Bit Words
Mass storage (NOS and NOS/BE only).	64
Tape in SI format with coded data (NOS/BE only).	128
Tape in SI format with binary data.	512
Tape in I format (NOS only).	512
Tape in any other format.	Undefined.

Privacy Key -
The value an application program must supply to CDCS in order to gain access to a particular data base area. Also called an access control key.

Procedure Library -
A permanent file containing the data base procedure referenced in a schema.

PRU Device -
Under NOS and NOS/BE, a mass storage device or a tape in SI or I format, so called because records on these devices are written in PRUs.

Qualification -
The method whereby a nonunique name can be made unique. If the name exists within a hierarchy of names, it can be made unique by mentioning one or more of the higher levels (normally the record name) of the hierarchy.

Qualifier Identifier -
A data item that restricts which record occurrences are to be retrieved when reading a relation. It is specified in the subschema. This data item can be the same data item specified as a join field in the schema.

Quick Recovery File -
A random permanent file used internally by CDCS to restore the structural integrity of the data base. It contains a copy of all data blocks for a data base which have been updated since the last recovery point.

Random File -
In the context of CRM, a file with word addressable, indexed sequential, direct access, or actual key organization in which individual records can be accessed by the values of their keys; in the context of the NOS and NOS/BE operating systems, a file with the random bit set in the file environment table in which individual records are accessed by their relative PRU numbers.

Rank -
The rank of a file in a DMS-170 relation corresponds to the position of the file in the schema definition of the relation. The ranks of the files joined in a relation are numbered consecutively, with the root file having a rank of 1.

Realm -
A uniquely named DMS-170 subschema data base subdivision that contains data records; identified in the schema as an area; a file.

Realm Ordinal -
A unique identifier assigned to each realm in a DMS-170 subschema when the subschema is compiled. Realm ordinals for a FORTRAN subschema are used in conjunction with the FORTRAN status variables.

Reconstruction -
Re-creation of all or specified portions of a DMS-170 data base utilizing a backup dump of the data base and the after-image record entries from the journal log file.

Record -
As defined by CRM, a group of related characters. A record or a portion thereof is the smallest collection of information passed between CRM and a user program. Eight different record types exist, as defined by the RT field of the file information table.

For CDCS processing, a record is equivalent to a record occurrence.

Other parts of the operating systems and their products can have additional or different definitions of records.

Record Code -
A unique value that identifies a specific record type in an area with multiple record types. The value is either contained in a data item in the record or derived by execution of a data base procedure.

Record Mapping -
See Mapping.

Record Occurrence -
The actual data base record, which conforms in the data base file to a record type described in the schema and conforms for use in the application program to the record description in the subschema.

Record Qualification -
The method used to restrict which records are to be returned to the user by specifying criteria that must be satisfied by a record occurrence. Record qualification is allowed only for relation reads.

Record Type -
A term that can have one of several meanings, depending on the context. CRM defines eight record types established by an RT field in the file information table. Tables output by the loader are classified as record types such as text, relocatable, or absolute, depending on the first few words of the tables.

In DDL, record type is defined in the record description entry of the schema. It is a description of the attributes of a record and the items included in the record which serves as a template by which record occurrences are interpreted; an arbitrary number of record occurrences can exist for each record type.

Recovery -
A process that makes a data base useful after some type of software or hardware failure has occurred.

Recovery Point -
A user-generated or system-generated point to which CDCS guarantees recovery with no loss of data. User-generated recovery points are initiated by COBOL calls to DB$RPT, by FORTRAN calls to DMLRPT, or by transmission of the Query Update directive RECOVERY. System-generated recovery points are initiated by CDCS when certain conditions occur, such as a full quick recovery file, a termination of the CDCS interface by a user program, or the committing of a data base transaction.

Relation -
The logical structure formed by the joining of files for the purpose of allowing an application program to retrieve data from more than one file at the same time. The structure is declared in the schema and is based on common identifiers in the files.

Relation Occurrence -
The logical concatenation of a record occurrence from each record type specified in the relation. Each read of the relation yields a relation occurrence. The record descriptions of the records that compose the relation occurrence are contained in the subschema.

Relational Data Base -
A data base of files joined in relations through data item identifiers.

Repeating Group -
A collection of data items which occurs a number of times within a record occurrence; can consist of elementary items, group items, and vectors; cannot be referenced in a FORTRAN subschema.

Restart Identifier
A unique identifier for a run-unit that is maintained by CDCS for program restart operations in transaction processing.

Restart Identifier File
A random permanent file used internally by CDCS to support program restart operations for programs that request a restart identifier.

Restoration -
Resetting of a data base to a previous state by applying the before-image record entries from the journal log file to the data base in its current state.

Result Item -
An item whose value is determined by a data base procedure. The time at which the value is determined depends on whether the item is an actual or virtual result item. The value of the item is generated by a data base procedure operating on the value of other items in the same record.

Root File -
The file that has the rank 1 in a relation; its record occurrences are pictured at the root of a tree in a hierarchical tree structure.

Run-Unit -
In the CDCS Environment, the execution of a user job at a control point. The user job becomes a run-unit at invocation and ceases to be a run-unit at termination. The user job can be an application program (a main program with associated subprograms), a Query Update application, or a sequence of TAF tasks.

Schema -
A detailed description of the internal structure of a data base.

Schema Identification Entry -
A schema source program statement that assigns a name to the schema.

Section -
As defined by CRM, a division within a file with sequential organization. Generally, a section contains more than one record and is a division within a partition of a file. A section terminates with a physical representation of a section boundary (see table C-5).

The NOS and NOS/BE operating systems equate a section with a system-logical-record of level 0 through 16 octal.

Sequential -
A file organization in which records are stored in the order in which they are generated.

Short PRU -
A PRU that does not contain as much user data as the PRU can hold and is terminated by a system terminator with a level number.

Under NOS, a short PRU defines EOR; under NOS/BE, a short PRU defines the end of a system-logical-record. In the CRM context, a short PRU can have several interpretations depending on the record and blocking types.

Source Data Item -
The data item received by CDCS. For a CRM GET function, the data item is in the schema format; for a CRM STORE function or REWRITE function, the data item is in the subschema format. Contrast with Target Data Item.

Source Identifier -
An identifier that links a parent record type in one file to a child record type in another file in a relation.

Subschema -
A detailed description of the portion of a data base to be made available to one or more application programs.

TABLE C-5. SECTION BOUNDARIES

Device	Record Type (RT)	Block Type (BT)	Physical Representation
PRU device	W	I	A deleted one-word record pointing back to the last I block boundary followed by a control word with flags indicating a section boundary. At least the control word is in a short PRU of level 0.
	W	C	A control word with flags indicating a section boundary. The control word is in a short PRU of level 0.
	D,F,R, T,U,Z	C	A short PRU with a level less than 17 octal.
	S	-	Undefined.
S or L format tape	W	I	A separate tape block containing as many deleted records of record length 0 as required to exceed noise record size, followed by a deleted one-word record pointing back to the last I block boundary, followed by a control word with flags indicating a section boundary.
	W	C	A separate tape block containing as many deleted records of record length 0 as required to exceed noise record size, followed by a control word with flags indicating a section boundary.
	D,F,R, T,U,Z	C,K,E	Undefined.
	S	-	Undefined.
Any other tape format	-	-	Undefined.

Subschema Item Ordinal -
A unique identifier within a record assigned to each item in a subschema when the subschema is compiled. Subschema item ordinals are used in conjunction with the data base status block.

Subschema Library -
A permanent file containing one or more subschemas.

System-Logical-Record -
Under NOS/BE, a data grouping that consists of one or more PRUs terminated by a short PRU or zero-length PRU. These records can be transferred between devices without loss of structure.

Equivalent to a logical record under NOS.

Equivalent to a CRM S type record.

Target Data Item -
The data item transferred by CDCS. For a CRM GET function, the data item is in the subschema format; for the CRM STORE or MODIFY function, the data item is in the schema format. Contrast with Source Data Item.

Target Identifier -
An identifier in a child record type which must be identical to the source identifier in the parent record type for two files in a relation to be joined.

Transaction -
A series of update operations identified by a user-assigned transaction identifier. A transaction is bracketed by a begin transaction operation and either a commit or drop operation.

Transaction Facility (TAF) -
A network product that controls transaction processing, called TAF-transaction processing in this manual. See Transaction Processing.

Transaction Processing-
In the context of CDCS, a process that provides control for interrelated updates to data base files by means of a transaction; also provides a program restart capability which can be used for restarting an application program after a system failure.

In the context of TAF (called TAF-transaction processing in this manual), a manipulation of a data base by high speed handling or repetitive executions of a relatively small number of jobs called tasks.

Transaction Recovery File
A random permanent file used internally by CDCS to perform automatic recovery. It contains before-image records of records updated within an incompleted transaction.

Traversing Files -
The process by which CDCS goes from one to another of the files joined in a relation. Record occurrences are retrieved based on identifiers. A record occurrence from each file is returned to the user.

Type
The storage format of a data item which determines permitted values, length, and arithmetic meaning.

User Function -
A request from an application program to open or close an area or to perform a specific operation on a record or data item.

User Work Area -
The collection of record areas allocated by the COBOL compiler, the FORTRAN DML preprocessor, or Query Update within a run-unit, for record types defined in the subschema used by a run-unit.

Variable -
A single named data item in the FORTRAN subschema data structure.

Variable Occurrence Data Item -
A data item that is repeated a specific number of times in each record occurrence. The number of occurrences is controlled by a preceding elementary data item; the data name option of the OCCURS clause in a schema; or format 2 of the OCCURS clause in a COBOL or Query Update subschema.

Vector -
An elementary data item that is repeated a number of times in each record occurrence.

Version -
A data base version (see Data Base Version); a Query Update directive (VERSION) that specifies a catalog file.

Virtual Item -
An item that is part of a record when it is retrieved, but was not part of the record (had no value) when it was stored. The value of the item is determined by a data base procedure.

W Type Record -
One of the eight record types supported by CRM. Such records appear in storage preceded by a system-supplied control word. The existence of the control word allows files with sequential organization to have both partition and section boundaries.

Zero-Byte Terminator -
12 bits of zero in the low order position of a word that marks the end of the line to be displayed at a terminal or printed on a line printer. The image of cards input through the card reader or terminal also has such a terminator.

Zero-Length PRU -
A PRU that contains system information, but no user data. Under CRM, a zero-length PRU of level 17 is a partition boundary. Under NOS, a zero-length PRU defines EOF.

RESERVED WORDS AND FORTRAN DDL KEYWORDS D

Reserved words are English words and abbreviations that have special meanings to the DDL compiler and the data base utilities. These words must not be used as names or literals within the source input to either the compiler or the utilities.

FORTRAN DDL keywords identify statements and options within statements to the FORTRAN DDL compiler.

The reserved words and the FORTRAN DDL keywords are listed in alphabetical order under the applicable subsection of this appendix.

SCHEMA RESERVED WORDS

ACCESS-CONTROL
ACTUAL
AFTER
ALLOWED
ALTERNATE
ALWAYS
ANY
ARE
AREA
ASCII
BEFORE
BY
CALL
CHAR
CHARACTER
CHECK
CLOSE
COBOL
CODE
COMPRESSION
CONSTRAINT
CONTROL
DATA
DEC
DECIMAL
DECODING
DECOMPRESSION
DELETE
DEPENDS
DISPLAY
DUPLICATES
DURING
ENCODING
EQ
ERROR
FIND
FIRST
FIXED
FLOAT
FOR
GET
ID
IDENTIFIER
IN
INDEXED
IS
JOIN
KEY
LOCK
MODIFY
NAME
NOT
OCCURS
OF
ON
OPEN
OR
PIC
PICTURE
PROCEDURE
REAL
RECORD
RELATION
RESULT
RETRIEVAL
SCHEMA
SEQUENCE
STORE
SYSTEM
THRU
TIMES
TO
TYPE
UPDATE
USE
USING
VALUE
VIRTUAL
WHERE
WITHIN

COBOL/QUERY UPDATE SUBSCHEMA RESERVED WORDS

ACCEPT
ACCESS
ACTUAL KEY
AD
ADD
ADDRESS
ADVANCING
AFTER
ALIAS
ALL
ALPHABETIC
ALPHANUMERIC
ALPHANUMERIC-EDITED
ALSO
ALTER
ALTERNATE
AND
APOSTROPHE
APPLY
ARE
AREA
AREAS
ASCENDING
ASSIGN
AT
AUTHOR
BECOMES
BEFORE
BEGINNING
BITS
BLANK
BLOCK
BOTTOM
BY
CALL
CANCEL
CD
CF
CH
CHARACTER
CHARACTERS
CLOCK-UNITS
CLOSE
COBOL
CODE
CODE-SET
COLLATING
COLUMN
COMMA
COMMON-STORAGE
COMMUNICATION
COMP
COMP-1
COMP-2
COMP-3
COMPLEX
COMPUTATIONAL
COMPUTATIONAL-1
COMPUTATIONAL-2
COMPUTATIONAL-3
COMPUTE
CONFIGURATION
CONTAINS
CONTROL
CONTROLS
CONVERSION
COPY
CORR
CORRESPONDING
COUNT
CURRENCY
DATA
DATE
DATE-COMPILED
DATE-WRITTEN
DAY
DAY-OF-WEEK
DE
DEADLOCK
DEBUG-CONTENTS
DEBUG-ITEM
DEBUG-LINE
DEBUG-NAME
DEBUG-SUB-1
DEBUG-SUB-2
DEBUG-SUB-3
DEBUGGING
DECIMAL-POINT
DECLARATIVES
DELETE
DELIMITED
DELIMITER
DEPENDING
DESCENDING
DESTINATION
DETAIL
DIRECT
DISABLE
DISPLAY
DIVIDE
DIVISION
DOUBLE
DOWN
DUPLICATES

COBOL/QUERY UPDATE SUBSCHEMA RESERVED WORDS (CONTD)

DYNAMIC
EGI
ELSE
EMI
ENABLE
END
END-OF-PAGE
ENDING
ENTER
ENVIRONMENT
EOP
EQ
EQUAL
EQUALS
ERROR
ESI
EVERY
EXCEEDS
EXCEPTION
EXIT
EXTEND
EXTERNAL
FD
FILE
FILE-CONTROL
FILES
FILLER
FINAL
FIRST
FOOTING
FOR
FROM
GE
GENERATE
GIVING
GO
GQ
GR
GREATER
GROUP
GT
HASHED-VALUE
HASHING
HEADING
HIGH-VALUE
HIGH-VALUES
HOLD
I-O
I-O-CONTROL
IDENTIFICATION
IF
IN
INDEX
INDEXED
INDICATE
INITIAL
INITIALIZE
INITIATE
INPUT
INPUT-OUTPUT
INSPECT
INSTALLATION
INTO
INVALID
IS
JUST
JUSTIFIED
KEY
LABEL
LAST
LE
LEADING
LEFT
LENGTH
LESS
LIMIT
LIMITS
LINAGE
LINAGE-COUNTER
LINE
LINES
LINKAGE
LOCALLY
LOCK
LOGICAL
LOW-VALUE
LOW-VALUES
LQ
LS
LT
MEMORY
MERGE
MESSAGE
MODE
MODULES
MOVE
MULTIPLE
MULTIPLY
NATIVE
NE
NEGATIVE
NEXT
NO
NOT
NQ
NUMBER
NUMERIC
NUMERIC-EDITED
OBJECT-COMPUTER
OBJECT-PROGRAM
OCCURS
OF
OFF
OMITTED
ON
ONLY
OPEN
OPTIONAL
OR
ORGANIZATION
OTHER
OUTPUT
OVERFLOW
PAGE
PAGE-COUNTER
PERFORM
PF
PH
PIC
PICTURE
PLUS
POINTER
POSITION
POSITIVE
PRINTING
PROCEDURE
PROCEDURES
PROCEED
PROCESS
PROCESSING
PROGRAM
PROGRAM-ID
QUEUE
QUOTE
RANDOM
RD
READ
REALM
REALMS
RECEIVE
RECORD
RECORDING
RECORDS
REDEFINES
REEL
REFERENCES
RELATION
RELATIVE
RELEASE
REMAINDER
REMOVAL
RENAMES
REPLACING
REPORT
REPORTING
REPORTS
RERUN
RESERVE
RESET
RESTRICT
RETAINING
RETURN
REVERSED
REWIND
REWRITE
RF
RH
RIGHT
RN
ROUNDED
RUN
SAME
SD
SEARCH
SECONDARY-STORAGE
SECTION
SECURITY
SEGMENT
SEGMENT-LIMIT
SELECT
SEND
SENTENCE
SEPARATE
SEQUENCE
SEQUENTIAL
SET
SIGN
SIZE
SORT
SORT-MERGE
SOURCE
SOURCE-COMPUTER
SPACE
SPACES
SS
STANDARD
STANDARD-1
START
STATUS
STOP
STORE
STRING
SUB-QUEUE-1
SUB-QUEUE-2
SUB-QUEUE-3
SUB-SCHEMA
SUBTRACT
SUM
SUPERVISOR
SUPPRESS
SUSPEND
SYMBOLIC
SYNC
SYNCHRONIZED
TABLE
TALLYING
TAPE
TENANT
TERMINAL
TERMINATE
TEXT
THAN
THROUGH
THRU
TIME
TIMES
TITLE
TO
TOP
TRACE-OFF
TRACE-ON
TRAILING
TYPE
UNEQUAL
UNIT
UNSTRING
UNTIL
UP
UPON
USAGE
USE
USING
VALUE
VALUES
VARYING
WHEN
WHERE
WITH
WITHIN
WORD-ADDRESS
WORDS
WORKING-STORAGE
WRITE
XOR
ZERO
ZEROES
ZEROS

FORTRAN DDL KEYWORDS

.A.	END	.LT.	REALM
ALIAS	.EQ.	.N.	RECORD
ALL	.GE.	.NE.	RELATION
.AND.	.GT.	.NOT.	RESTRICT
BOOLEAN	INTEGER	.O.	SCHEMA
CHARACTER	ITEM	.OR.	SUBSCHEMA
COMPLEX	.LE.	PRECISION	.X.
DOUBLE	LOGICAL	REAL	.XOR.

DBMSTRD UTILITY RESERVED WORDS

ACCOUNT	FILE	LO	RECORDS
ADD	GE	MASTER	RECOVERY
AFTER	HD	MODIFICATIONS	RESTART
AREA	HI	MODIFY	SAME
AS	HY	MODS	SCHEMA
ASSIGNED	ID	MT	SCHEMAS
BEFORE	IDENTIFIER	NAME	SET
BLOCKS	IMAGE	NT	SUBSCHEMA
CHANGE	INFORMATION	OFF	TAPE
CHARGE	INDEX	ON	TRANSACTION
CONTROL	IS	PACK	TYPE
DELETE	JOB	PE	UN
DENSITY	JOURNAL	PFN	UNIT
DEVICE	LOG	PROCEDURE	UPDATE
END	LIBRARY	PW	VERSION
FAMILY	LIMIT	QUICK	VSN

DBREC UTILITY RESERVED WORDS

ALLOCATE	IS	PRU	RECOVERY
DUMP	JOURNAL	PRUS	SCHEMA
FILE	LOG	QUICK	SIZE
IDENTIFIER	NAME	RESTART	TRANSACTION

DBRCN AND DBRST UTILITIES RESERVED WORDS

AREA	FOR	POINT	SELECT
AT	IS	RECOVERY	TIME
BEGIN	JOB	RUN-UNIT	VERSION
END	NAME	SCHEMA	

SUMMARY SYNTAX FOR THE SCHEMA, SUBSCHEMAS, MASTER DIRECTORY, AND DATA BASE UTILITIES E

The syntax for the components of DMS-170 listed below are summarized in this appendix. Two forms of notations appear in these summaries. One form follows the COBOL convention, and the other form follows the FORTRAN convention. Refer to the Notations section for information about these conventions. The syntax is shown for the following components:

Schema

Schema exhibit utility

COBOL and Query Update subschemas

FORTRAN subschema

Master directory

DBREC utility

DBRCN and DBRST utility

The syntax is shown in the order indicated above. Detailed information for each format is referenced by page number. Within the summary for each element, a general format of source input (if applicable) precedes the statements.

SCHEMA SYNTAX

schema identification entry

{area description entry} ...

{record description entry} ...

{data control entry}

[constraint entry] ...

[relation entry] ...

SCHEMA IDENTIFICATION ENTRY

SCHEMA NAME IS schema-name. 2-9

AREA DESCRIPTION ENTRY

AREA NAME IS area-name. 2-9

```
[ CALL data-base-procedure [ BEFORE | ON ERROR | AFTER ] DURING [ [ OPEN | CLOSE ] [ FOR [ UPDATE | RETRIEVAL ] ] ] ]   2-9

[ ACCESS-CONTROL LOCK [ FOR [ UPDATE | RETRIEVAL ] ] IS { literal-1 | PROCEDURE data-base-procedure-1 }   2-10

    [ OR { literal-2 | PROCEDURE data-base-procedure-2 } ] ... ] .
```

Schema

RECORD DESCRIPTION ENTRY

```
RECORD NAME IS record-name                                                    2-11

    WITHIN area-name                                                          2-11

    [ CALL data-base-procedure || BEFORE      || DURING || [ STORE  ] || ] ... .   2-11
                               || ON ERROR    ||        || [ DELETE ] ||
                               || AFTER       ||        || [ MODIFY ] ||
                                                        || [ FIND   ] ||
                                                        || [ GET    ] ||

    [data description entry.] ...                                             2-12
```

Data Description Entry

```
[level-number] data-name

    [ { PICTURE } IS "picture-specification" ]                                 2-12
    [ { PIC     }                            ]

    [ TYPE IS ( || [ DECIMAL ] ||                [integer-1[integer-2]] )     2-14
              ( || [ DEC     ] ||                                       )
              ( || [ FIXED   ] ||                                       )
              ( || [ FLOAT   ] ||                                       )
              (    REAL                                                 )
              ( { CHARACTER }                   [integer-3]             )
              ( { CHAR      }                                           )
              ( || [ DECIMAL ] ||                                       )
              ( || [ DEC     ] ||                                       )
              ( || COMPLEX     ||                                       ) ]

    [ OCCURS { integer   } TIMES ]                                            2-16
             { data-name }

    [ IS { ACTUAL  } RESULT OF data-base-procedure ]                          2-16
         { VIRTUAL }

    [ CHECK IS || PICTURE                                                     2-17
               || PIC
               || data-base-procedure
               || VALUE [NOT] literal-1 [THRU literal-2] [literal-3 [THRU literal-4]] ... || ]

    [ FOR { ENCODING } [ALWAYS] CALL data-base-procedure] ...                 2-19
          { DECODING }

    [ CALL data-base-procedure || BEFORE   || DURING || [ STORE  ] || ] ... .   2-19
                               || ON ERROR ||        || [ GET    ] ||
                               || AFTER    ||        || [ MODIFY ] ||
```

DATA CONTROL ENTRY

```
DATA CONTROL.

    [area control entry.] ...                                          2-21
```

Area Control Entry

```
AREA NAME IS area-name                                                 2-21

[ FOR ||COMPRESSION|| USE {SYSTEM                                      2-22
      ||DECOMPRESSION||    {PROCEDURE data-base-procedure} ]

[ KEY IS [ALTERNATE] data-name [USING data-base-procedure]
      [ DUPLICATES ARE [INDEXED] ALLOWED ]                             2-22
                       [FIRST  ]
                       [NOT    ]                                  ]  ...
[ KEY IDENTIFIER IS [ALTERNATE] key-name < data-name-1 [data-name-2] ...> ]  2-22
      [ DUPLICATES ARE [INDEXED] ALLOWED ]                             2-22
                       [FIRST  ]                                   ]  ...
                       [NOT    ]                                  ]

[ SEQUENCE IS {ASCII   } ]                                             2-23
              {COBOL   }
              {DISPLAY }

[                    {BY data-name  VALUE FOR record-name-1 IS literal-1
                     {                    [VALUE FOR record-name-2 IS literal-2] ...
[ RECORD CODE IS     {PROCEDURE data-base-procedure                                  2-24
                     {                    VALUE FOR record-name-3 IS integer-3
                     {                    [VALUE FOR record-name-4 IS integer-4  ... ]
```

CONSTRAINT ENTRY

```
CONSTRAINT NAME IS constraint-name                                     2-24

    data-name-1 DEPENDS ON data-name-2.
```

RELATION ENTRY

```
RELATION NAME IS relation-name                                         2-27

    JOIN WHERE identifier-1 EQ identifier-2

        [identifier-3 EQ identifier-4] ... .
```

SCHEMA EXHIBIT DIRECTIVE FORMAT

```
EXHIBIT SCHEMA ( AREA                  )  ( ALL [ NAMES   ] )
               ( RECORD [ONLY]         )  ( [ CLAUSES ] )
               ( ITEM                  )  (             )
               ( ALL-ENTRIES           )  ( entry-name  )
               ( CONSTRAINT            )  (             )
               ( RELATION              )  (             )
```

2-36

COBOL AND QUERY UPDATE SUBSCHEMA SYNTAX

```
TITLE DIVISION.
{title description entry}.

[ALIAS DIVISION.
[alias description entry.]]...

REALM DIVISION.
{realm description entry}.

RECORD DIVISION.
{record description entry.} ...

[RELATION DIVISION.
{relation description entry.}]...
```

TITLE DIVISION

```
TITLE DIVISION.

SS sub-schema-name WITHIN schema-name.
```

3-11

ALIAS DIVISION

```
ALIAS DIVISION.

[ AD { REALM realm-name-1   } BECOMES { realm-name-2  } . ] ...
     { RECORD record-name-1 }         { record-name-2 }
     { DATA data-name-1     }         { data-name-2   }
```

3-12

REALM DIVISION

```
REALM DIVISION.

RD { ALL                          } .
   { realm-name-1 [realm-name-2] ... }
```

3-12

RECORD DIVISION, COBOL SUBSCHEMA

```
RECORD DIVISION.

{record-description-entry.} ...
```

3-13

Record Description Entry, COBOL Subschema

```
01   record-name.

{data-description-entry.} ...
```

3-13

Data Description Entry, COBOL Subschema

FORMAT 1

```
level-number   data-name                                                    3-13

[ {JUSTIFIED} RIGHT ]                                                       3-13
  {JUST     }

[ OCCURS integer-2 TIMES                                                    3-14
    [ {ASCENDING } KEY IS data-name-1 [data-name-2] ... ] ...
    {DESCENDING }
    [INDEXED BY index-name-1 [index-name-2] ...]
]

[ OCCURS integer-1 TO integer-2 TIMES DEPENDING ON data-name-1              3-14
    [ {ASCENDING } KEY IS data-name-2 [data-name-3] ... ] ...
    {DESCENDING }
    [INDEXED BY index-name-1 [index-name-2] ...]
]

[ {PICTURE} IS picture-specification ]                                      3-15
  {PIC    }

[REDEFINES data-name-1]                                                     3-17

[ {SYNCHRONIZED} [ {LEFT } ] ]                                              3-18
  {SYNC        }   {RIGHT}

[ USAGE IS { COMPUTATIONAL   }                                              3-19
           { COMP            }
           { COMPUTATIONAL-1 }
           { COMP-1          }
           { COMPUTATIONAL-2 }
           { COMP-2          }
           { DISPLAY         }
           { INDEX           } ]
```

FORMAT 2

```
66 data-name-1 RENAMES data-name-2 [ {THRU   } data-name-3 ].               3-19
                                    {THROUGH}
```

FORMAT 3

```
88 condition-name {VALUE IS} literal-1 [ {THRU   } literal-2 ]              3-20
                  {VALUES ARE}         {THROUGH}

    [ literal-3 [ {THRU   } literal-4 ] ] ... .
                {THROUGH}
```

RECORD DIVISION, QUERY UPDATE SUBSCHEMA

RECORD DIVISION. 3-22

{record-description-entry.} ...

Record Description Entry, Query Update Subschema

01 record-name. 3-22

{data-description-entry.} ...

Data Description Entry, Query Update Subschema

FORMAT 1

level-number data-name 3-23

[{ JUSTIFIED / JUST } RIGHT] 3-23

[OCCURS integer-2 TIMES] 3-23

[OCCURS integer-1 TO integer-2 TIMES DEPENDING ON data-name-1] 3-23

[{ PICTURE / PIC } IS picture-specification] 3-24

[REDEFINES data-name-1] 3-28

[{ SYNCHRONIZED / SYNC } [{ LEFT / RIGHT }]] 3-28

[USAGE IS { COMPUTATIONAL / COMP / COMPUTATIONAL-1 / COMP-1 / COMPUTATIONAL-2 / COMP-2 / DISPLAY / INDEX / DOUBLE / LOGICAL / COMPLEX }] 3-28

FORMAT 2

66 data-name-1 RENAMES data-name-2 [{ THRU / THROUGH } data-name-3] . 3-30

RELATION DIVISION

```
RELATION DIVISION.

RN IS relation-name                                                    3-30

[ RESTRICT record-name                                                 3-30
                                                                          ]
    WHERE [NOT][(] identifier-1 { EQ | NQ | NE | GR | GT | LS | LT | GQ | GE | LQ | LE }
                                { identifier-2 | literal-1 | data-name-1 } [)]
                                                                          ...
[[ { AND | OR | XOR } [NOT][(] identifier-3 { EQ | NQ | NE | GR | GT | LS | LT | GQ | GE | LQ | LE }
                                           { identifier-4 | literal-2 | data-name-2 } [)] ] ... ]
```

FORTRAN SUBSCHEMA SYNTAX

```
SUBSCHEMA statement

[ALIAS statement] ...

{REALM statement} ...

{ RECORD statement    }
{{type statement} ... } ...

[ RELATION statement     ]
[[RESTRICT statement] ...] ...

END statement
```

SUBSCHEMA STATEMENT

```
SUBSCHEMA subschema-name, SCHEMA = schema-name                         4-9
```

ALIAS STATEMENT

```
ALIAS { (REALM)  } new-name-1 = old-name-1 [, new-name-2 = old-name-2] ...   4-9
      { (RECORD) }
      { (ITEM)   }
```

FORTRAN Subschema

REALM STATEMENT

```
REALM  { ALL                                       }          4-10
       { realm-name-1 [, realm-name-2] ...         }
```

RECORD STATEMENT

```
RECORD record-name                                            4-10
```

TYPE STATEMENT - FORTRAN 5

```
{ CHARACTER [*default-length [,]] item-name-1 [*len-1] [, item-name-2 [*len-2]] ... }
{ { BOOLEAN          }                                                                }
{ { REAL             }                                                                }
{ { INTEGER          }                                                                }   4-11
{ { LOGICAL          } item-name-1 [, item-name-2] ...                                }
{ { COMPLEX          }                                                                }
{ { DOUBLE PRECISION }                                                                }
```

RELATION STATEMENT

```
RELATION relation-name                                        4-12
```

RESTRICT STATEMENT

```
RESTRICT record-name-1 (logical-expression-1) [, record-name-2 (logical-expression-2)] ...   4-12
```

Format of logical expression:

```
                          { .EQ. }
                          { .NE. }  { db-item-2    }
[ .NOT. ] [(] db-item-1   { .GT. }  { constant-1   }  [)]
[ .N.   ]                 { .LT. }  { non-db-item-1}
                          { .GE. }
                          { .LE. }

[ { .AND. }                             { .EQ. }                      ]
[ { .A.   }                             { .NE. }  { db-item-4     }   ]
[ { .OR.  }  [ .NOT. ] [(] db-item-3    { .GT. }  { constant-2    } [)] ] ...
[ { .O.   }  [ .N.   ]                  { .LT. }  { non-db-item-2 }   ]
[ { .XOR. }                             { .GE. }                      ]
[ { .X.   }                             { .LE. }                      ]
```

END STATEMENT

```
END                                                           4-13
```

MASTER DIRECTORY SYNTAX

Master directory syntax is shown in two subsections, syntax for a creation run and syntax for modification run. The syntax is ordered according to the general format for each run. Subentries that are referenced in the format appear following the last subentry (the subschema subentry) of the creation run.

SYNTAX FOR A CREATION RUN

General format for the creation run appears below.

{creation entry} ...

where the skeleton of a creation entry is as follows:

schema subentry
master version subentry
[alternate version subentry] ...
{subschema subentry} ...

Schema Subentry

```
SCHEMA NAME IS schema-name                                          8-3
     FILE NAME IS lfn

[ PROCEDURE LIBRARY                                                 8-4
       permanent file information subentry ]

[ TRANSACTION RECOVERY FILE                                         8-4
       permanent file information subentry
       [UNIT LIMIT IS integer-1]
       [UPDATE LIMIT IS integer-2] ]

[ RESTART IDENTIFIER FILE                                           8-4
       permanent file information subentry ]

[ JOURNAL LOG FILE                                                  8-4
       permanent file information subentry ]

[ QUICK RECOVERY FILE                                               8-4.1
       permanent file information subentry ]

[ JOB CONTROL INFORMATION                                           8-4.1
       [ TAPE [ TYPE IS { MT } ]
              [ TYPE IS { NT } ]
              [ DENSITY IS { LO } ]
              [ DENSITY IS { HI } ]
              [ DENSITY IS { HY } ]
              [ DENSITY IS { HD } ]
              [ DENSITY IS { PE } ]
              [ DENSITY IS { GE } ] ]
         [UN IS literal-1]
         [PW IS literal-2]
         [FAMILY NAME IS family-name]
         [ {ACCOUNT} IS literal-3 ]
         [ {CHARGE } ]                                                .
```

Master Directory

Master Version Subentry

```
[VERSION NAME IS MASTER]                                          8-5
    {area subentry} ...
```

Alternate Version Subentry

```
VERSION NAME IS version-name                                      8-8

    { { . }                                                       }
    { ||AREA NAME IS area-name SAME AS MASTER.|| ...              }
    { ||area subentry                         ||                  }
```

Subschema Subentry

```
SUBSCHEMA NAME IS subschema-name                                  8-8
    FILE NAME IS lfn.
```

Area Subentry

```
AREA NAME IS area-name                                            8-6
    PFN IS literal-1                                              8-6

        {UN} IS literal-2 [PW IS literal-3 [literal-4]...]
        {ID}

        [FAMILY NAME IS family-name-1                             ]
        [PACK NAME IS pack-name-1                                 ] [DEVICE TYPE IS dt-1]
        [SET NAME IS set-name-1  VSN IS literal-5                 ]

    [                                                             ]
    [ LOG ||BEFORE IMAGE RECORDS [OFF]                            ]                  8-7
    [     ||                     [ON ]                            ]
    [     ||AFTER IMAGE RECORDS  [OFF]                            ]
    [     ||                     [ON ]                            ]
    [     ||BEFORE IMAGE BLOCKS  [OFF]                            ]
    [     ||                     [ON ]                            ]

    [ INDEX FILE ASSIGNED PFN IS literal-6                        ]                  8-7
    [                                                             ]
    [    {UN} IS literal-7 [PW IS literal-8 [literal-9]...]       ]
    [    {ID}                                                     ]
    [                                                             ]
    [    [FAMILY NAME IS family-name-2                            ]
    [    [PACK NAME IS pack-name-2                                ] [DEVICE TYPE IS dt-2]
    [    [SET NAME IS set-name-2  VSN IS literal-10               ]
    [                                                             ]
```

Permanent File Information Subentry

```
PFN IS literal-1 {UN} IS literal-2 [PW IS literal-3 [literal-4]...]   8-1
                 {ID}

    [FAMILY NAME IS family-name                                   ]
    [PACK NAME IS pack-name                                       ] [DEVICE TYPE IS dt]
    [SET NAME IS set-name  VSN IS literal-5                       ]
```

SYNTAX FOR MODIFICATION RUN

General format for the modification run appears below. Syntax for these entries reference syntax shown for the creation run.

```
|| add schema entry    ||
|| delete schema entry || ...
|| modify schema entry ||
```

Add Schema Entry

```
ADD SCHEMAS.                                                    8-9
    {creation entry} ...
```

Delete Schema Entry

```
DELETE SCHEMAS.                                                 8-9
    {SCHEMA NAME IS schema-name.}  ...
```

Modify Schema Entry

```
MODIFY SCHEMA NAME IS schema-name                               8-9
    [FILE NAME IS lfn-1].

    [ CHANGE PROCEDURE LIBRARY                          ]       8-10
    [     permanent file information subentry.          ]

    [ CHANGE TRANSACTION RECOVERY FILE                  ]       8-10
    [    [permanent file information subentry]          ]
    [    [UNIT LIMIT IS integer-1]                      ]
    [    [UPDATE LIMIT IS integer-2]                    ]  .

    [ CHANGE RESTART IDENTIFIER FILE                    ]       8-11
    [    permanent file information subentry.           ]

    [ CHANGE JOURNAL LOG FILE                           ]       8-12
    [    permanent file information subentry.           ]

    [ CHANGE QUICK RECOVERY FILE                        ]       8-12
    [    permanent file information subentry.           ]
```

```
[CHANGE JOB CONTROL INFORMATION                                        ]   8-12
[     [        || TYPE IS {MT}                        || ]             ]
[     [        ||         {NT}                        || ]             ]
[     [ TAPE   ||                  (LO)               || ]             ]
[     [        ||                  (HI)               || ]             ]
[     [        || DENSITY IS       (HY)               || ]             ]
[     [        ||                  (HD)               || ]             ]
[     [        ||                  (PE)               || ]             ]
[     [        ||                  (GE)               || ]             ]
[     [UN IS literal-1]                                                ]
[     [PW IS literal-2]                                                ]
[     [FAMILY NAME IS family-name]                                     ]
[     [{ACCOUNT} IS literal-3]                                         ] .
[     [{CHARGE }              ]                                        ]

[CHANGE AREA NAME IS area-name                                         ]   8-13
[        [VERSION NAME IS {MASTER      }]                              ]
[        [                {version-name}]                              ]
[        [SAME AS MASTER]                                              ]
[        [||permanent file information subentry                   ||]  ]
[        [||                                                      ||]  ]
[        [||      || BEFORE IMAGE RECORDS [OFF] ||                ||]  ]
[        [||      ||                      [ON ] ||                ||]  ]
[        [|| LOG  || AFTER IMAGE RECORDS  [OFF] ||                ||]  ] .
[        [||      ||                      [ON ] ||                ||]  ]
[        [||      || BEFORE IMAGE BLOCKS  [OFF] ||                ||]  ]
[        [||      ||                      [ON ] ||                ||]  ]
[        [||                                                      ||]  ]
[        [|| INDEX FILE ASSIGNED                                  ||]  ]
[        [||      permanent file information subentry             ||]  ]

[DELETE VERSION NAME IS version-name.] ...                                 8-15

[ADD VERSION NAME IS version-name                                  ]       8-15
[   { { . }                                                 }      ]
[   { ||AREA NAME IS area-name SAME AS MASTER.  ||          } ...  ]
[   { ||area subentry                           ||          }      ]

[ADD SUBSCHEMA NAME IS subschema-name-1]                                   8-15
[   FILE NAME IS lfn-2.                ] ...

[DELETE SUBSCHEMA NAME IS subschema-name-2.] ...                           8-15

END {MODIFICATIONS} .                                                      8-10
    {MODS         }
```

DBREC SYNTAX

```
SCHEMA NAME IS schema-name-1

[ DUMP                                                          ]
[     JOURNAL LOG FILE NAME IS log-file-name-1 [.]              ]  ...

[ ALLOCATE                                                                                  ]
[     || JOURNAL LOG FILE [NAME IS log-file-name-2] SIZE is integer-1 PRUS                  ||
[     || QUICK RECOVERY FILE NAME IS qrf-name-1 SIZE IS integer-2 PRUS                      ||  [.]  ...
[     || TRANSACTION RECOVERY FILE NAME IS trf-name                                         ||
[     || RESTART IDENTIFIER FILE NAME IS rif-name                                           ||
```

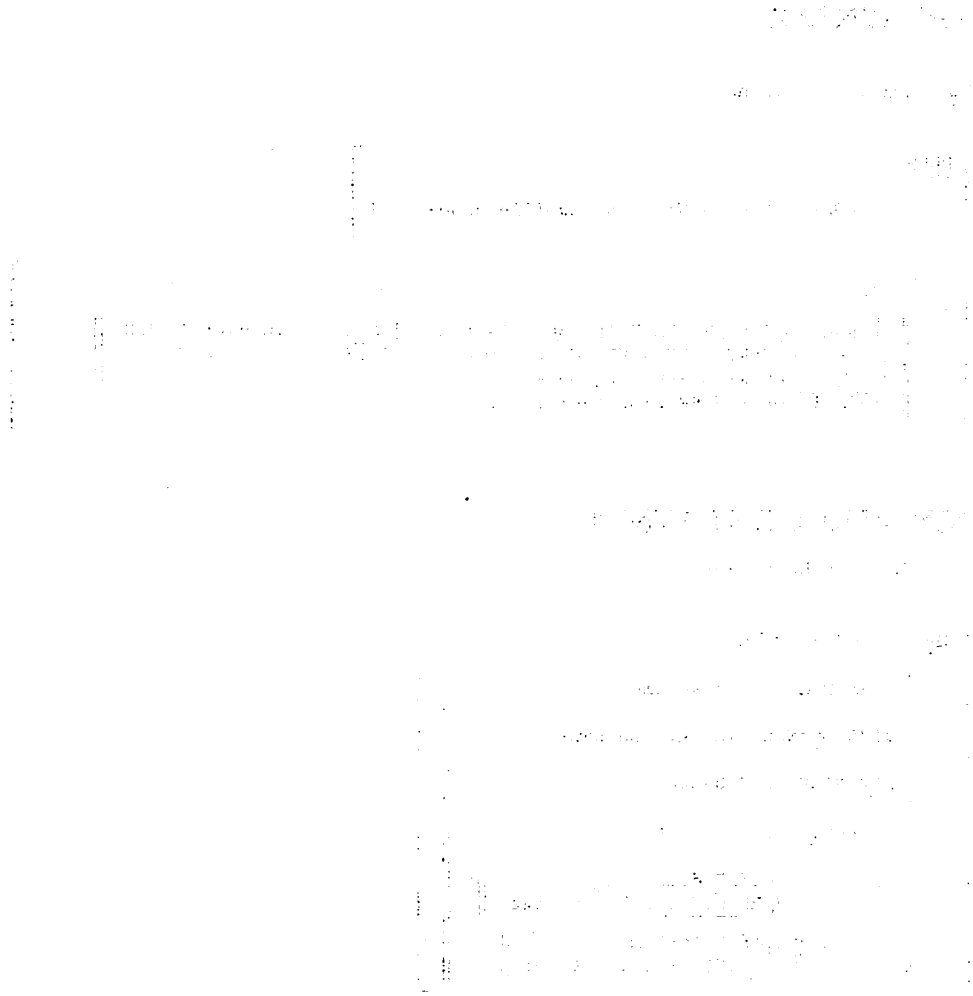
9-6

DBRCN AND DBRST SYNTAX

```
SCHEMA NAME IS schema-name.

[ SELECT FOR RECOVERY                                                                  ]
[     || AREA NAME IS area-name                                                        ||
[     || VERSION NAME IS version-name                                                  ||
[     || JOB NAME IS job-name                                                          ||
[     || RUN-UNIT IS run-unit-id                                                       ||  ...
[     || BEGIN AT || TIME date-time                                                    ||
[     ||          || RECOVERY POINT integer                                            ||
[     || END AT   || TIME date-time                                                    ||
[     ||          || RECOVERY POINT integer                                            ||
```

9-11



FUTURE SYSTEM MIGRATION GUIDELINES F

This appendix contains programming practices recommended by CDC for users of the software described in this manual. When possible, application programs based on this software should be designed and coded in conformance with these recommendations.

Two forms of guidelines are given. The general guidelines minimize application program dependence on the specific characteristics of a hardware system. The feature use guidelines ensure the easiest migration of an application program to future hardware or software systems.

GENERAL GUIDELINES

Programmers should observe the following practices to avoid hardware dependency:

Avoid programming hardcoded constants. Manipulation of data should never depend on the occurrence of a type of data in a fixed multiple such as 6, 10, or 60.

Do not manipulate data based on the binary representation of that data. Characters should be manipulated as characters, rather than as octal display-coded values or as 6-bit binary digits. Numbers should be manipulated as numeric data of a known type, rather than as binary patterns within a central memory word.

Do not identify or classify information based on the location of a specific value within a specific set of central memory word bits.

Avoid COMPASS in application programs. COMPASS and other machine-dependent languages can complicate migration to future hardware or software systems. Migration is restricted by continued use of COMPASS for stand-alone programs, by COMPASS subroutines embedded in programs using higher-level languages, and by COMPASS owncode routines used with CDC standard products. COMPASS should only be used to create part or all of an application program when the function cannot be performed in a higher-level language or when execution efficiency is more important than any other consideration.

FEATURE USE GUIDELINES

The recommendations in the remainder of this appendix ensure the easiest migration of an application program for use on future hardware or software systems. These recommendations are based on known or anticipated changes in the hardware or software system, or comply with proposed new industry standards or proposed changes to existing industry standards.

ADVANCED ACCESS METHODS

The Advanced Access Methods (AAM) offer several features within which choices must be made. The following paragraphs indicate preferred usage.

Access Methods

The recommended access methods are indexed sequential (IS), direct access (DA), and multiple-index processor (MIP).

Record Types

The recommended record types are either F for fixed length type records, or W for variable length records. Record length for W records is indicated in the control word; the length must be supplied by the user in the RL FIT field on a put operation and is returned to the user in RL on a get operation.

FORTRAN Usage

The following machine-independent coding practices are encouraged for a FORTRAN programmer using AAM:

Initialize the FIT by FILExx calls or by the FILE control statement.

Modify the FIT with STOREF calls.

Use the FORTRAN 5 CHARACTER data type when working with character fields rather than octal values of display code characters; specify lengths of fields, records, and so forth, in characters rather than words.

DMS-170

DMS-170 offers several features among which choices must be made. The following paragraphs indicate preferred usage of CDCS, DDL, and of Query Update in support of CDCS.

Multiple Record Descriptions

Do not include multiple record descriptions on a single file.

Repeating Groups

Avoid the use of the OCCURS clause, repeating groups, or arrays within records; as an alternative, the repeating data can be normalized into separate records on a different file. If repeating

data must be used, limit usage to fixed length groups (no OCCURS DEPENDING ON clause) and to simple (unnested) OCCURS clauses.

Alternate Keys on Repeating Groups

Avoid the specification of alternate keys on repeating groups. The data can be normalized as indicated under Repeating Groups.

Collating Sequence

Use the default collating sequence or the ASCII collating sequence.

REDEFINES Clause

Use the REDEFINES clause only for alphanumeric-to-alphanumeric redefinitions, where the term alphanumeric has the meaning assigned by COBOL to data. In general, avoid the use of REDEFINES where use is based on a knowledge of the internal representation of data (floating-point layout, number of characters per word, and so forth).

Query Update Syntax

Use the new directives INVOKE, STORE, MODIFY, and REMOVE, instead of the directives USE, INSERT, UPDATE, and DELETE.

FIELD LENGTH REQUIREMENTS G

This appendix discusses field length requirements for the following:

Schema generation

Subschema generation

CYBER Database Control System (CDCS) execution

Data base utilitites execution

Additional information concerning CDCS memory requests and job aborts is also given.

FIELD LENGTH REQUIREMENTS FOR SCHEMA GENERATION

The minimum field length required for generating a schema is approximately 50000 octal. For complex source input, additional field length of approximately 12 octal words is required for each data item contained in the record description entries.

FIELD LENGTH REQUIREMENTS FOR SUBSCHEMA GENERATION

The minimum field length required for generating a minimum FORTRAN, COBOL, or Query Update subschema is 50000 octal; a minimum subschema contains one realm, one record, and one item.

Field length requirements for a larger subschema can be estimated by increasing the field length by 12 octal words for each additional element in the subschema; an element is a realm, record, item, or relation.

FIELD LENGTH REQUIREMENTS FOR CDCS EXECUTION

CDCS resides at a system control point. After CDCS is initiated as an active system control point, the remainder of the field length represents managed memory under the control of Common Memory Manager (CMM). An applications program using CDCS resides at a user control point.

A basic load for CDCS execution requires sufficient space for the CDCS code and overlay capsules when in use, plus adequate space for buffers, tables, mapping capsules, and data base procedures as indicated in the estimates below. The space required for CDCS code can be determined from the load map generated during CDCS installation. The load map also shows the space required for each CDCS overlay capsule (OVCAP).

The code for many optional CDCS features is contained in overlay capsules. When a feature is used, the space for the corresponding overylay capsules must be included in the space for CDCS. Overlay capsules are used for the following features:

Constraint processing

Data base procedure processing

Data base version processing

Logging to the log and recovery files

Operator console displays

Relation processing

User-requested accounting

I/O buffers, various internal tables generated by CDCS, CDCS overlay capsules, mapping capsules, Advanced Access Methods (AAM) code capsules, and data base procedures are maintained in managed memory. The sizes of the buffers and tables are a function primarily of the maximum data record size, file block size, and the complexity of individual schemas and subschemas. The following are guidelines for determining a suitable CDCS execution field length:

I/O buffers

A minimum of 2 or 4 buffers per file, depending on whether or not an index file exists. (Buffer size is determined by the maximum block length (MBL) specification when the file was created.) In addition, 200 octal words are needed for the FSTT, and 100 octal words for each user of the file.

Pooled buffer space

The buffer space that can be allocated by AAM. The BL parameter on the CDCS control statement determines the buffer space that can be allocated by AAM. A value for the BL parameter can be specified to reduce the space that would be allocated by default. Specifying a value of less than 40000 octal is not usually recommended because specifying a lower value can cause degradation in the performance of CDCS processing.

Tables per user

Determined from the number of concurrent users, the size of the CST table, and the degree of sharing of CST-related data. Generally, the additional space per user above the CST is less than the CST size, except when the CST size is less than 200 octal words.

AAM/BAM basic routines

16000 octal words.

AAM IS

6000 octal words.

AAM DA

3000 octal words.

AAM AK

5000 octal words.

AAM MIP

6000 octal words.

Mapping capsules

Determined from the Data Description Language (DDL) subschema output or from CST statistics in the DBMSTRD output listing.

Data base procedures

Determined from capsule load map.

CST tables

Reported by the DBMSTRD utility.

A factor to be considered in deriving CDCS memory requirements is the degree to which sharing of subschemas for a given data base occurs. When users are sharing the same subschema, CDCS links all users to a single copy of the tables for that subschema.

When CDCS makes a memory request that cannot be satisfied, various procedures are executed to attempt to free up sufficient space so that the request can be satisfied. For example, data base procedures are unloaded and user tables are swapped out. Other modules (for example, AAM) also release unneeded blocks of memory through their own memory overflow procedures. If adequate memory is still not available, the user's job might be aborted.

If space cannot be obtained, a user's job might be aborted under the following circumstances:

Invocation of CDCS. Space is required for subschema and other tables.

Execution of a large data base procedure or mapping capsule. Space is required to load the capsule.

Some overflow conditions are not recoverable, even after all possible memory is released. In these cases, CDCS and all jobs using CDCS are aborted. This situation indicates that CDCS requires more field length to perform its basic functions. The maximum field length parameter on the CDCS control statement must be increased.

CDCS should normally be allowed to request sufficient memory to process requests from the maximum number of concurrent user programs that can be in memory at one time. If CDCS is constrained to a smaller amount of memory, conflicts result and occasionally jobs are aborted.

When CDCS is executing at a system control point and has no user requests to process, it rolls out its field length to an RMS swap file, leaving a field length of approximately 700 octal words at the control point. When a new user request or a request for the system L display is received, CDCS rolls itself back into memory.

CDCS MEMORY MANAGEMENT PARAMETER GUIDELINES

CDCS provides several control statement parameters that are aids in controlling the growth of the CDCS field length. There are no exact formulas for the best use of these parameters, but the following steps and guidelines will help achieve some initial estimates for these parameters.

First, review the available parameters that are provided for permanently loading CRM capsules and CDCS overlay capsules. When any of these are expected to be used on a regular basis, it is much more efficient to load them only once during CDCS initialization, than to permit them to be loaded and unloaded repeatedly during CDCS execution. This also avoids one of the possible sources of memory fragmentation.

The next step is to select the value of the MFL parameter. This value is often selected as a compromise between the minimum CDCS requirements and the maximum that the installation can permit CDCS to use within the environment of that machine. A larger MFL will permit a more efficient execution of CDCS. A small MFL will require frequent execution of memory overflow procedures and occasionally cause jobs or even CDCS to be aborted for lack of sufficient central memory. The preceding paragraph contains some basis for estimating a minimum requirement.

A value that should be known while selecting these parameter values is the initial load FL of CDCS. This value is given by the day file message "INITIAL LOAD FL" that is written to the CDCS dayfile immediately following CDCS initialization. Be sure that the CRM and CAP parameters are the same as are to be used in future CDCS executions.

To select a value for the BL parameter, take the difference between the MFL parameter and the initial load FL. This is the memory available for CDCS tables, mapping capsules, database procedures, CRM tables and I-O buffers. Half of this available memory should be made available for the I-O buffers. That is a value that is a reasonable estimate for the BL parameter. If the BL parameter is not specified, CDCS will select a value that is slightly larger than this calculation.

Increasing BL increases CDCS efficiency, but CRM may use an unwarrented amount of central memory buffer space. Decreasing BL will cause CRM to flush its buffers more frequently, resulting in more I/O.

SBL and SBI are parameters that are available to help to combat the memory fragmentation that occurs within CDCS. A small block is defined as any memory block of less than 240 octal words. Both CDCS and the Record Manager assign many memory blocks that fall within the small block category. As a general rule, these small blocks are assigned at a fixed location and cannot be released or moved by any of the memory overflow procedures.

There are provisions for releasing most of the larger blocks. If the large blocks are released but leave small blocks that were nested among them, then the resulting memory is fragmented.

It is possible to establish a boundary beyond which CMM does not assign any small blocks. When the small blocks are prevented from entering an area reserved for large blocks, the large blocks can be exchanged with each other more freely. When the SBI (small block boundary increment) is specified, the boundary is originally established at the initial load field length plus two increments.

When it is not possible to assign a small block within the small block boundary, CMM calls the memory overflow procedures that are defined by CDCS and by CRM. These memory overflow routines will first attempt to make memory available by releasing memory that is held. The releasable memory blocks consist of capsules, overlay capsules, I-O buffers and the RSB and CST tables. Each of these releasable memory blocks usually fall into the the large block category but may have been assigned within the small block boundary.

If releasing memory blocks does not make memory available within the small block boundary, the boundary is advanced by one increment. If the memory is not available within the enlarged boundary, the boundary is advanced just temporarily to include the full field length. Then after the block has been assigned the boundary is restored to its previously enlarged value.

When the large blocks that have been released are reassigned, they will not find any available space within the small block boundary and will be reassigned beyond the boundary. There, where the immobile small blocks will not interfere, the large blocks will exchange with each other more freely.

The small block boundary is reduced by 100 octal each time that a CDCS area is closed.

It is recommended that SBI be set to a value in the range of 2000 to 14000. Selection of a smaller value will cause the CMM overflow routines to be executed more frequently. This can help to contain the CDCS FL but incurs a large overhead.

The SBL parameter specifies an upper limit beyond which the small block boundary will not be advanced. If SBL is specified without specifying SBI, the boundary is fixed at SBL. Small blocks will be assigned beyond the SBL if all overflow procedures have failed to make space available within the limit. However, when they are being assigned beyond the limit, all of the memory overflow procedures will be tried before assigning each small block.

The SBL parameter can be used as a more flexible substitute for the MFL parameter to control the advance of the field length. Before establishing an SBL value it would be valuable to first specify only an SBI and observe the small block boundary summary messages that are written to the CDCS output file when CDCS is terminated. This will provide a better idea of what a reasonable limit is.

USE A UNIFORM CRM BLOCK SIZE

Another important consideration for reducing memory fragmentation is to attempt to define the memory blocks of uniform size so that they will easily interchange with each other. When specifying the MBL for each of the database files, consider that CRM will often be releasing a block from one file to allocate a block from another file. This works best if the blocks are the same size. When it is necessary to define blocks of different sizes, select just two or three uniform sizes to which all files can conform.

FIELD LENGTH REQUIREMENTS FOR THE DATA BASE UTILITIES

The minimum execution field lengths for jobs executing the data base utilities are the following:

DBRCN	70000 octal words
DBRST	70000 octal words
DBMSTRD	45000 octal words
DBQRFA	20000 octal words
DBQRFI	5000 octal words
DBREC	50000 octal words

DATA CONVERSION RULES H

The data conversion rules that apply to CDCS record mapping described in section 5 are covered in this appendix. Conversion sequences that can cause alteration of data without conversion diagnostics are noted following the conversion rules.

Valid conversions are listed with source and target data class codes, expected source field contents, conversion operation performed, and error conditions. Information that applies to a set of conversion rules is identified by the source data class and an X denoting all target data classes (for example, 3-X) and precedes the set to which it pertains. A specific conversion is referenced by giving the source and target data classes separated by a hyphen. CDCS diagnostic messages noted for error conditions are contained in appendix B.

The data classes and their respective codes are as follows:

Code	Meaning
0	Display alphanumeric
1	Display alphabetic
3	Display integer
4	Display fixed point
10	Coded binary integer
13	Coded floating point normalized
14	Coded double precision
15	Coded complex

CONVERSION RULES

Source Data Class: 0

Target Data Class: 0

Source Field: Must contain display code alphanumeric characters.

Operation: Source field is moved to the target field, and left-justified with blank fill to the right. If right-justification is specified for the source field, leading blanks are shifted off left circular before the move. If right-justification is specified for the target field, trailing blanks are shifted off right circular after the move.

Error Condition: Truncation of a nonblank character terminates the conversion with diagnostic 432 (660 octal) or 445 (675 octal).

Source Data Class: 0

Target Data Class: 1

Source Field: Same as 0-0.

Operation: Same as 0-0.

Error Condition: Same as 0-0, except that if a character other than a blank or an alphabetic character is present, conversion is terminated with diagnostic 432 (660 octal) or 445 (675 octal).

Source Data Class: 0

Target Data Class: 3

Source Field: Must contain display code alphanumeric characters. Any of the following can be present:

Optional leading blanks

Option sign (display code + or -)

Optional intervening blanks (one contiguous string of blanks before the first numeric character)

Numeric display code characters of which the last character can be sign overpunch (characters <, A through R, or ∨)

Optional trailing blanks

Operation: Source field is shifted right until the rightmost character is display code numeric. Sign overpunch is converted to display code numeric. Source field characters are moved right-justified to the target field. Blanks are converted to display code zeros. The sign character is not counted in the target field length. If the source field contains no numeric characters, the target field is filled with display code zeros. If the sign overpunch is specified in the target picture, the sign is provided according to the overpunch sign or the explicit sign in the source field.

Error Condition: Any departure from the source field rules results in termination of the conversion with diagnostic 432 (660 octal) or 445 (675 octal). If significant digits are lost, conversion is terminated with diagnostic 432 (660 octal) or 445 (675 octal).

Source Data Class: 1

Target Data Class: 0

Source Field: Must contain display code blanks or alphabetic characters.

Operation: Same as 0-0.

Error Condition: Same as 0-1.

Source Data Class: 1

Target Data Class: 1

Source Field: Same as 1-0.

Operation: Same as 1-0.

Error Condition: Same as 1-0.

Source Data Class: 3

Target Data Class: X
(Applies to all 3-X conversions.)

Source Field: Must contain display code numeric characters, except for the sign overpunch if it is specified in the source item picture description.

Operation: Sign overpunch is interpreted and replaced by numeric equivalent.

Error Condition: If the source field contains illegal characters, conversion terminates with diagnostic 432 (660 octal) or 445 (675 octal).

Source Data Class: 3

Target Data Class: 0

Source Field: See 3-X.

Operation: Source field is moved and left-justified with blank fill to the right, unless right-justification is specified for the target, in which case the source field is moved and right-justified with blank fill to the left.

Error Condition: If all signficant digits cannot be accommodated in the target field, conversion terminates with diagnostic 432 (660 octal) or 445 (675 octal). See 3-X also.

Source Data Class: 3

Target Data Class: 3

Source Field: See 3-X.

Operation: Display code numeric characters are moved and right-justified with display code zero fill to the left. The sign overpunch is supplied in the appropriate position if specified in the target item picture.

Error Condition: If all signficant digits cannot be accommodated in the target field, conversion terminates with diagnostic 432 (660 octal) or 445 (675 octal). See 3-X also.

Source Data Class: 3

Target Data Class: 4

Source Field: See 3-X.

Operation: Same as 3-3, except that the explicit decimal is supplied if it is specified in the target picture. The target field receives display code zero fill to the right of the decimal position before execution of the move operation described in 3-3.

Error Condition: See 3-X and 3-3.

Source Data Class: 3

Target Data Class: 10

Source Field: See 3-X.

Operation: Display code characters are converted to a scaled binary integer. If the source item is negative, the entire 60-bit word is complemented.

Error Condition: If a binary integer of more than 48 bits is generated, conversion terminates with diagnostic 432 (660 octal) or 445 (675 octal).

Source Data Class: 3

Target Data Class: 13

Source Field: See 3-X.

Operation: Display code characters are converted to a binary integer, then to a normalized floating point representation, and complemented if negative.

Error Condition: See 3-X.

Source Data Class: 3

Target Data Class: 14

Source Field: See 3-X.

Operation: Display code characters are converted to a binary integer, then to a normalized floating point representation, complemented if negative, and stored in the most significant part. The least significant part is set to zero.

Error Condition: See 3-X.

Source Data Class: 4

Target Data Class: X
(Applies to all 4-X conversions.)

Source Field: Must contain display code numeric characters, except for the sign overpunch and/or explicit decimal if either or both are specified in the source item picture description.

Operation: Sign overpunch is interpreted and replaced by numeric equivalent.

Error Condition: See 3-X.

Source Data Class: 4

Target Data Class: 3

Source Field: See 4-X.

Operation: The integer part of the source field is moved right-justified, with display code zero fill to the left. Rounding occurs on the least significant digit according to the value of the fractional part of the source field. A sign overpunch is supplied if it is specified in the target item picture.

Error Condition: If all significant digits of the integer part of the source field cannot be accomodated in the target field, conversion terminates with diagnostic 432 (660 octal) or 445 (675 octal).

Source Data Class: 4

Target Data Class: 4

Source Field: See 4-X.

Operation: The explicit decimal is removed, moved, or supplied as required by the source and target item picture descriptions. Characters are moved and justified on the decimal position, with display code zero fill to the left and right. Rounding, if required, occurs on the least significant digit. A sign overpunch is supplied if it is specified in the target item picture.

Error Condition: If all significant digits of the integer part of the source field cannot be accomodated in the target field, conversion terminates with diagnostic 432 (660 octal) or 445 (675 octal).

Source Data Class: 4

Target Data Class: 10

Source Field: See 4-X.

Operation: Display code characters are converted to a binary integer; this binary integer is rounded, complemented if the source item is negative, and scaled.

Error Condition: If an integer of more than 48 digits is generated, conversion terminates with diagnostic 432 (660 octal) or 445 (675 octal).

Source Data Class: 4

Target Data Class: 13

Source Field: See 4-X.

Operation: Display code characters are converted to a binary integer, then to a normalized floating point representation scaled by the source decimal positions, and complemented if negative.

Error Condition: None.

Source Data Class: 4

Target Data Class: 14

Source Field: See 4-X.

Operation: Display code characters are converted to a binary integer scaled by the source decimal position, then to a normalized floating point representation scaled by the source decimal positions, complemented if negative, and stored in the first word of the target. The second word is set to zero.

Error Condition: None.

Source Data Class: 10

Target Data Class: X
(Applies to all 10-X conversions.)

Source Field: Must be a 48-bit binary integer, right-justified in a full word. The high order bits are all zero for a positive number or 7777 octal for a negative number.

Operation: If the source value is negative, it is complemented before conversion.

Source Data Class: 10

Target Data Class: 3

Source Field: See 10-X.

Operation: The coded arithmetic value is converted to display code numeric characters, which are stored in the target field right-justified with display code zero fill to the left. A sign overpunch is supplied if it is specified in the target item picture.

Error Condition: If all significant digits of the integer part of the source field cannot be accommodated in the target field, conversion terminates with diagnostic 432 (660 octal) or 445 (675 octal).

Source Data Class: 10

Target Data Class: 4

Source Field: See 10-X.

Operation: Same as 10-3, except the coded arithmetic value is converted to an integer and fractional parts are converted to display code numeric character strings. The value is stored in the target field justified on the decimal position. CDCS supplies a decimal point if specified in the target item picture and rounding or display code zero fill in the fractional part as required.

Error Condition: Same as 10-3.

Source Data Class: 10

Target Data Class: 10

Source Field: See 10-X.

Operation: The source field is scaled and packed, based on the decimal positions in the source and target items.

Error Condition: None.

Source Data Class: 10

Target Data Class: 13

Source Field: See 10-X.

Operation: The source field is scaled and normalized if the source is negative.

Error Condition: None.

Source Data Class: 10

Target Data Class: 14

Source Field: See 10-X.

Operation: The source field is scaled, normalized, complemented if the source is negative, and stored in the first word of the target. The second word is set to zero.

Error Condition: None.

Source Data Class: 10

Target Data Class: 15

Source Field: See 10-X.

Operation: The source field is scaled, normalized, complemented if the source is negative, and stored in the real part. The imaginary part is set to zero.

Error Condition: None.

Source Data Class: 13

Target Data Class: X
(Applies to all 13-X conversions.)

Source Field: Must be a full-word normalized floating point representation.

Operation: If sign bit is set, the source value is complemented before conversion.

Error Condition: None.

Source Data Class: 13

Target Data Class: 3

Source Field: See 13-X.

Operation: The coded arithmetic value is converted display code numeric characters, which are stored in the target field right-justified with display code zero fill to the left. A sign overpunch is supplied if it is specified in the target item picture.

Error Condition: None.

Source Data Class: 13

Target Data Class: 4

Source Field: See 13-X.

Operation: Same as 13-3, except the coded arithmetic value is converted to an integer and fractional parts are converted to display code numeric character strings. The value is stored in the target field justified on the decimal position. CDCS supplies a decimal point if specified in the target item picture and rounding or display code zero fill in the fractional part as required.

Error Condition: None.

Source Data Class: 13

Target Data Class: 10

Source Field: See 13-X.

Operation: The source field is unnormalized, scaled according to the target item decimal position, and complemented if the source is negative.

Error Condition: If any significant digits are lost, conversion is terminated with diagnostic 432 (660 octal) or 445 (675 octal).

Source Data Class: 13

Target Data Class: 13

Source Field: See 13-X.

Operation: No conversion is required.

Error Condition: None.

Source Data Class: 13

Target Data Class: 14

Source Field: See 13-X.

Operation: The source field is moved to the first word of the double precision target. The second word is supplied with a coefficient of zero and an exponent equal to the exponent of the first word minus 48. The value is complemented if the source is negative.

Error Condition: None.

Source Data Class: 13

Target Data Class: 15

Source Field: See 13-X.

Operation: The real part of the target field is set to the value of the source field. The imaginary part of the target field is set to zero.

Error Condition: None.

Source Data Class: 14

Target Data Class: X
(Applies to all 14-X conversions.)

Source Field: Must be a two full-word double precision floating point representation. Each word must contain a normalized floating point value, with the exponent of word 2 equal to the exponent of word 1 minus 48. Both words are complemented prior to conversion if the value of the source field is negative.

Error Condition: None apply to 14-X conversions.

Source Data Class: 14

Target Data Class: 3

Source Field: See 14-X.

Operation: Same as 13-3.

Error Condition: None.

Source Data Class: 14

Target Data Class: 4

Source Field: See 14-X.

Operation: Same as 13-4.

Error Condition: None.

Source Data Class: 14

Target Data Class: 10

Source Field: See 14-X.

Operation: If word 2 of the source field is not zero, word 1 is rounded. The source value in word 1 is then unnormalized, scaled according to the target item decimal position, and complemented if the source is negative.

Error Condition: If any significant digits are lost, conversion is terminated with diagnostic 432 (660 octal) or 445 (675).

Source Data Class: 14

Target Data Class: 13

Source Field: See 14-X.

Operation: If word 2 of the source field is not zero, word 1 is rounded. Word 1 is then moved to the target field and complemented if the source is negative.

Error Condition: None.

Source Data Class: 14

Target Data Class: 14

Source Field: See 14-X.

Operation: No conversion is required.

Error Condition: None.

Source Data Class: 14

Target Data Class: 15

Source Field: See 14-X.

Operation: The real part of the target field is set to the most significant part of the source field. The imaginary part of the target field is set to zero.

Error Condition: None.

Source Data Class: 15

Target Data Class: 10

Source Field: Must be two words of normalized floating point representation.

Operation: The real part of the source field is unnormalized, scaled according to the target item decimal position, and complemented if the source is negative. The imaginary part of the source field is discarded.

Error Condition: If any significant digits are lost, conversion is terminated with diagnostic 432 (660 octal) or 445 (675 octal).

Source Data Class: 15

Target Data Class: 13

Source Field: See 15-10.

Operation: No conversion is required for the real part of the source field. The imaginary part of the source field is discarded.

Error Condition: None.

Source Data Class: 15

Target Data Class: 14

Source Field: See 15-10.

Operation: The real part of the source field is moved to the first word of the double precision target. The second word is supplied with a coefficient of zero and an exponent equal to the exponent of the first word minus 48. The value is complemented if the source is negative. The imaginary part of the source field is discarded.

Error Condition: None.

Source Data Class: 15

Target Data Class: 15

Source Field: See 15-10.

Operation: No conversion is required.

Error Condition: None.

CONVERSION SEQUENCE NOTES

On a read operation followed by a rewrite operation involving schema to subschema to schema mapping, some conversion sequences do not produce conversion errors but can cause alteration of data in the data base file. These conversion sequences are noted in table H-1.

TABLE H-1. CONVERSION SEQUENCES

Conversion Sequence Data Classes			Conversion Notes
Schema	Subschema	Schema	
0	3	0	Position of digits and placement of sign is determined by the picture for the data class 3 item.
4	3	4	The fractional part of the number is lost; the integer part is rounded.
4	4	4	The fractional part of the number might be rounded.
13	3	13	Same as 4-3-4.
13	4	13	Same as 4-4-4.
3	13	3	Final target value has at most 14 significant digits.
3	13	4	
4	13	3	
4	13	4	
14	3	14	Double precision is lost.
14	4	14	
14	10	14	
14	13	14	
15	3	15	The imaginary part is lost.
15	4	15	
15	10	15	
15	13	15	

COLLATING SEQUENCES FOR DATA BASE FILES I

The collating sequence of a data base file is determined by the area control entry in the schema. The following collating sequences can be specified: ASCII, COBOL, and DISPLAY. If no collating sequence is specified in the schema area control entry, the COBOL collating sequence is the default for the file.

The collating sequence specified for a file affects the following:

- The order in which items are returned when a file is accessed sequentially.
- The order in which items are returned by alternate key retrieval for all file organizations if duplicate alternate keys are allowed.
- The order in which items are returned by primary key (or the major portion of a primary key) for files that have indexed sequential file organization.

The collating sequence of a Query Update catalog file must be the DISPLAY collating sequence.

Collating sequences are shown in table I-1.

TABLE I-1. DATA BASE COLLATING SEQUENCE

Collating Sequence		ASCII		COBOL		DISPLAY	
Decimal	Octal	Graphics	Display Code	Graphics	Display Code	Graphics	Display Code
00	00	blank	55	blank	55	:†	00†
01	01	!	66	≤	74	A	01
02	02	=	64	%†	63†	B	02
03	03	#	60	[	61	C	03
04	04	$	53	→	65	D	04
05	05	%†	63†	≡	60	E	05
06	06	&	67	∧	67	F	06
07	07	'	70	↑	70	G	07
08	10	(	51	↓	71	H	10
09	11	)	52	>	73	I	11
10	12	*	47	≥	75	J	12
11	13	+	45	¬	76	K	13
12	14	,	56	.	57	L	14
13	15	-	46	)	52	M	15
14	16	.	57	;	77	N	16
15	17	/	50	+	45	O	17
16	20	0	33	$	53	P	20
17	21	1	34	*	47	Q	21
18	22	2	35	-	46	R	22
19	23	3	36	/	50	S	23
20	24	4	37	,	56	T	24
21	25	5	40	(	51	U	25
22	26	6	41	=	54	V	26
23	27	7	42	≠	64	W	27
24	30	8	43	<	72	X	30
25	31	9	44	A	01	Y	31
26	32	:†	00†	B	02	Z	32
27	33	;	77	C	03	0	33
28	34	<	72	D	04	1	34
29	35	=	54	E	05	2	35
30	36	>	73	F	06	3	36
31	37	?	71	G	07	4	37
32	40	@	74	H	10	5	40
33	41	A	01	I	11	6	41
34	42	B	02	∨	66	7	42
35	43	C	03	J	12	8	43
36	44	D	04	K	13	9	44
37	45	E	05	L	14	+	45
38	46	F	06	M	15	-	46
39	47	G	07	N	16	*	47
40	50	H	10	O	17	/	50
41	51	I	11	P	20	(	51
42	52	J	12	Q	21	)	52
43	53	K	13	R	22	$	53
44	54	L	14	]	62	=	54
45	55	M	15	S	23	blank	55
46	56	N	16	T	24	,	56
47	57	O	17	U	25	.	57
48	60	P	20	V	26	≡	60
49	61	Q	21	W	27	[	61
50	62	R	22	X	30	]	62
51	63	S	23	Y	31	%†	63†
52	64	T	24	Z	32	≠	64
53	65	U	25	:†	00†	→	65
54	66	V	26	0	33	∨	66
55	67	W	27	1	34	∧	67
56	70	X	30	2	35	↑	70
57	71	Y	31	3	36	↓	71
58	72	Z	32	4	37	<	72
59	73	[	61	5	40	>	73
60	74	\	75	6	41	≤	74
61	75	]	62	7	42	≥	75
62	76	⌒	76	8	43	¬	76
63	77	-	65	9	44	;	77

†In installations using a 63-graphic set, the % graphic does not exist. The : graphic is display code 63.

SUMMARY OF DATA DEFINITION IN DMS-170 J

This summary indicates the correspondence of the clauses or statements that can be used in defining data items in DMS-170. Table J-1 shows the schema definition required for data items of each schema data class and the subschema definitions that correspond to each schema data class.

For Query Update access to schema-defined data base files in CYBER Record Manager (CRM) data base access mode, the data base must be defined in the Query Update subschema exactly as the data base is defined in the schema. Therefore, every data item must be defined to correspond in size and class to the schema definition of the item.

For COBOL, FORTRAN, and Query Update access to schema-defined files in CYBER Database Control System (CDCS) data base access mode, the definition of data items in the subschemas does not have to correspond exactly to the schema definition of data items. Through mapping, CDCS can generate a record image conforming to the subschema format from a record in schema format, or can perform the conversion of data from subschema format to schema format. Detailed information about the conversions allowed is included in sections 3 and 4 of this manual.

TABLE J-1. DATA DEFINITION IN DMS-170

SCHEMA					COBOL Subschema		FORTRAN 5 Subschema Type Statement	Query Update Subschema in CDCS Data Base Access Mode		Query Update Subschema in CRM Data Base Access Mode	
Data Class No.	Data Class Name	PICTURE Clause	TYPE Clause	Internal Representation	PICTURE Clause	USAGE Clause		PICTURE Clause	USAGE Clause	PICTURE Clause	USAGE Clause
0	Display alpha-numeric	Alpha-numeric (A X 9; not all As or 9s; mixed specification used as all Xs)	CHARACTER	Display code, alphanumeric	Alpha-numeric (A X 9; not all As or 9s; mixed specification used as all Xs)	DISPLAY (or none)	CHARACTER	Alpha-numeric (A X 9; not all As or 9s; mixed specification used as all Xs)	DISPLAY (or none)	Alpha-numeric (A X 9; not all As or 9s; mixed specification used as all Xs)	DISPLAY (or none)
1	Display alpha-betic	Alpha-betic (A)	None	Display code, alphabetic	Alpha-betic (A)	DISPLAY (or none)	CHARACTER	Alpha-betic (A)	DISPLAY (or none)	Alpha-betic (A) (or none)	DISPLAY (or none)
3	Display integer	Numeric (9 T)	None	Display code numeric, can have sign overpunch in last character position	Numeric (9 S)	DISPLAY COMP (or none)	None†	Numeric (9 S and insertion and re-placement characters)	DISPLAY COMP (or none)	Numeric (9 S and insertion and re-placement characters)	DISPLAY COMP (or none)
4	Display fixed	Numeric (9 P T V .)	None	Display code numeric plus implicit or explicit decimal or scaling position	Numeric (9 S V P)	DISPLAY COMP (or none)	None†	Numeric (9 S V P and insertion and replacement characters)	DISPLAY COMP (or none)	Numeric (9 S V P and insertion and replacement characters)	DISPLAY COMP (or none)
10	Coded binary integer	None	FIXED integer-1 integer-2 (where the integer value is 1 through 18)	Binary integer	Numeric (9 S V P)	COMP-1 INDEX††	INTEGER LOGICAL BOOLEAN	Numeric (9 S V P and insertion and replacement characters)	COMP-1 INDEX LOGICAL	Numeric (9 S V P and insertion and replacement characters)	COMP-1 INTEGER LOGICAL

TABLE J-1. DATA DEFINITION IN DMS-170 (Contd)

SCHEMA					COBOL Subschema		FORTRAN 5 Subschema Type Statement	Query Update Subschema in CDCS Data Base Access Mode		Query Update Subschema in CRM Data Base Access Mode	
Data Class No.	Data Class Name	PICTURE Clause	TYPE Clause	Internal Representation	PICTURE Clause	USAGE Clause		PICTURE Clause	USAGE Clause	PICTURE Clause	USAGE Clause
13	Coded floating point normalized	None	FLOAT integer-1 (where the integer value is 1 through 14)	Signed, normalized floating point (1 word)	Numeric (9 S V P)	COMP-2	REAL BOOLEAN	Numeric (9 S V P and insertion and replacement characters)	COMP-2	Numeric (9 S V P and insertion and replacement characters)	COMP-2
14	Coded double precision	None	FLOAT integer-1 (where the integer value is 15 through 29)	Signed, normalized floating point (2 words)	None†	None†	DOUBLE PRECISION	Numeric (9 S V P and insertion and replacement characters)	DOUBLE	Numeric (9 S V P and insertion and replacement characters)	DOUBLE
15	Coded complex	None	COMPLEX	Floating point with real part and imaginary part (2 words)	None†	None†	COMPLEX	Numeric (9 S V P and insertion and replacement characters)	COMPLEX	Numeric (9 S V P and insertion and replacement characters)	COMPLEX

†No corresponding subschema type. Valid conversion shown in sections 3 and 4 of this manual.

††No picture clause allowed.

INDEX