DB2 Universal Database for OS/390

Reference Summary

Version 6

SX26-3844-00

Note!

Before using this information and the product it supports, be sure to read the general information under Appendix B, “Notices” on page 411.

First Edition (June 1999)

This edition applies to Version 6 of DB2 Universal Database Server for OS/390, 5645-DB2, and to any subsequent releases until otherwise indicated in new editions. The information in this summary is compiled from the following publications:

DB2 SQL Reference, SC26-9014-00
DB2 Messages and Codes, GC26-9011-00
DB2 Command Reference, SC26-9006-00
DB2 Utility Guide and Reference, SC26-9015-00
DB2 Administration Guide, SC26-9003-00

Make sure you are using the correct edition for the level of the product.

Contents

Introduction to this book and the DB2® for OS/390® library

This reference summary contains a subset of the material in the following publications:

DB2 SQL Reference
DB2 Messages and Codes
DB2 Command Reference
DB2 Utility Guide and Reference
DB2 Administration Guide

It is intended to be used as a quick reference by experienced users of DB2.

About this book

This reference summary is divided into eight sections:

Section 1: "Definitions of SQL syntax elements" on page 7
This section contains syntax diagrams for SQL syntax elements: functions, expressions, predicates, search conditions, and queries.

Section 2: "SQL alphabetic reference" on page 27
This section contains syntax diagrams and examples of SQL statements organized alphabetically by name.

Section 3: "SQL return codes" on page 169
This section lists SQL return codes and explanations of their meanings.

Section 4: "DB2 catalog table reference" on page 197
This section lists DB2 catalog table names in alphabetic order along with the corresponding descriptions and column names.

Section 5: "Tables used by EXPLAIN and the resource limit facility" on page 209

This section lists the tables that are used by DB2's EXPLAIN function and the resource limit facility (governor). Table names are listed in alphabetic order along with the corresponding descriptions and column names.

Section 6: "Command alphabetic reference" on page 211
This section contains syntax diagrams and examples of DB2 commands organized alphabetically by name.

Section 7: "Online utility alphabetic reference" on page 323
This section contains syntax diagrams and examples of DB2 online utilities organized alphabetically by name.

Section 8: "Stand-alone utility alphabetic reference" on page 383
This section contains syntax diagrams and examples of DB2 stand-alone utilities organized alphabetically by name.

Appendix A: Appendix A, "Resource types for resource codes in DB2 messages" on page 407
This section contains -904 resource unavailable reason codes.

Introduction

What product terminology and citations mean

In this book, DB2 Universal Database Server for OS/390 is referred to as "DB2 for OS/390." In cases where the context makes the meaning clear, DB2 for OS/390 is referred to as "DB2." When this book refers to other books in this library, a short title is used. (For example, "See *DB2 SQL Reference*" is a citation to *IBM DATABASE 2 Universal Database Server for OS/390 SQL Reference*.)

References in this book to "DB2 UDB" relate to the DB2 Universal Database™ product that is available on the AIX®, OS/2®, and Windows NT™ operating systems. When this book refers to books about the DB2 UDB product, the citation includes the complete title and order number.

The following terms are used as indicated:

DB2® Represents either the DB2 licensed program or a particular DB2 subsystem.

C and C language Represent the C programming language.

CICS® Represents CICS/ESA® and CICS Transaction Server for OS/390 Release 1.

IMS™ Represents IMS/ESA®.

MVS Represents the MVS/Enterprise Systems Architecture (MVS/ESA™) element of OS/390.

How to use the DB2 library

Titles of books in the library begin with DB2 Universal Database for OS/390 Version 6. However, references from one book in the library to another are shortened and do not include the product name, version, and release. Instead, they point directly to the section that holds the information.

Throughout the library, the DB2 for OS/390 licensed program and a particular DB2 for OS/390 subsystem are each referred to as “DB2.” In each case, the context makes the meaning clear.

The most rewarding task associated with a database management system is asking questions of it and getting answers, the task we call *end use*. Other tasks are also necessary—defining the parameters of the system, putting the data in place, and so on. We group the tasks associated with DB2 into the following major categories (but supplemental information relating to all of the below tasks for new releases of DB2 can be found in *DB2 Release Guide*):

Installation: If you are involved with DB2 only to install the system, *DB2 Installation Guide* might be all you need.

If you will be using data sharing then you also need *DB2 Data Sharing: Planning and Administration*, which describes installation considerations for data sharing.

End use: End users issue SQL statements to retrieve data. They can also insert, update, or delete data, with SQL statements. They might need an introduction to SQL, detailed instructions for using SPUFI, and an alphabetized reference to the types of SQL statements. This information is found in *DB2 Application Programming and SQL Guide* and *DB2 SQL Reference*.

End users can also issue SQL statements through the Query Management Facility (QMF™) or some other program, and the library for that program might provide all the instruction or reference material they need. For a list of the titles in the QMF library, see the bibliography at the end of this book.

Application Programming: Some users access DB2 without knowing it, using programs that contain SQL statements. DB2 application programmers write those programs. Because they write SQL statements, they need *DB2 Application Programming and SQL Guide*, *DB2 SQL Reference*, and *DB2 ODBC Guide and Reference* just as end users do.

Application programmers also need instructions on many other topics:

- How to transfer data between DB2 and a host program—written in COBOL, C, or FORTRAN, for example
- How to prepare to compile a program that embeds SQL statements
- How to process data from two systems simultaneously, say DB2 and IMS™ or DB2 and CICS®
- How to write distributed applications across platforms
- How to write applications that use DB2 ODBC to access DB2 servers
- How to write applications that use Open Database Connectivity (ODBC) to access DB2 servers
- Write applications in the Java™ programming language to access DB2 servers

The material needed for writing a host program containing SQL is in *DB2 Application Programming and SQL Guide* and in *DB2 Application Programming Guide and Reference for Java™*. The material needed for writing applications that use DB2 ODBC or ODBC to access DB2 servers is in *DB2 ODBC Guide and Reference*. For handling errors, see *DB2 Messages and Codes*.

Information about writing applications across platforms can be found in *Distributed Relational Database Architecture™: Application Programming Guide*.

System and Database Administration: *Administration* covers almost everything else. *DB2 Administration Guide* divides those tasks among the following sections:

- Section 2 (Volume 1) of *DB2 Administration Guide* discusses the decisions that must be made when designing a database and tells how to bring the design into being by creating DB2 objects, loading data, and adjusting to changes.

Introduction

- Section 3 (Volume 1) of *DB2 Administration Guide* describes ways of controlling access to the DB2 system and to data within DB2, to audit aspects of DB2 usage, and to answer other security and auditing concerns.
- Section 4 (Volume 1) of *DB2 Administration Guide* describes the steps in normal day-to-day operation and discusses the steps one should take to prepare for recovery in the event of some failure.
- Section 5 (Volume 2) of *DB2 Administration Guide* explains how to monitor the performance of the DB2 system and its parts. It also lists things that can be done to make some parts run faster.

In addition, the appendixes in *DB2 Administration Guide* contain valuable information on DB2 sample tables, National Language Support (NLS), writing exit routines, interpreting DB2 trace output, and character conversion for distributed data.

If you are involved with DB2 only to design the database, or plan operational procedures, you need *DB2 Administration Guide*. If you also want to carry out your own plans by creating DB2 objects, granting privileges, running utility jobs, and so on, then you also need:

- *DB2 SQL Reference*, which describes the SQL statements you use to create, alter, and drop objects and grant and revoke privileges
- *DB2 Utility Guide and Reference*, which explains how to run utilities
- *DB2 Command Reference*, which explains how to run commands

If you will be using data sharing then you need *DB2 Data Sharing: Planning and Administration*, which describes how to plan for and implement data sharing.

Additional information about system and database administration can be found in *DB2 Messages and Codes*, which lists messages and codes issued by DB2, with explanations and suggested responses.

Diagnosis: Diagnosticians detect and describe errors in the DB2 program. They might also recommend or apply a remedy. The documentation for this task is in *DB2 Diagnosis Guide and Reference* and *DB2 Messages and Codes*.

How to obtain DB2 information

DB2 on the Web

Stay current with the latest information about DB2. View the DB2 home page on the World Wide Web. News items keep you informed about the latest enhancements to the product. Product announcements, press releases, fact sheets, and technical articles help you plan your database management strategy.

You can view and search DB2 publications on the Web, or you can download and print many of the most current DB2 books. Follow links to other Web sites with more

information about DB2 family and OS/390 solutions. Access DB2 on the Web at the following address:

http://www.software.ibm.com/data/db2/os390

DB2 publications

The DB2 publications for DB2 Universal Database Server for OS/390 are available in both hardcopy and softcopy format.

BookManager® format

Using online books on CD-ROM, you can read, search across books, print portions of the text, and make notes in these BookManager books. With the appropriate BookManager READ product or IBM Library Readers, you can view these books in the OS/390, VM, OS/2, DOS, AIX, and Windows™ environments. You can also view many of the DB2 BookManager books on the Web.

PDF format

Many of the DB2 books are available in Portable Document Format (PDF) for viewing or printing from CD-ROM or the Web. Download the PDF books to your intranet for distribution throughout your enterprise.

CD-ROMs

Books for Version 6 of DB2 Universal Database Server for OS/390 are available on CD-ROMs:

- *DB2 UDB for OS/390 Version 6 Licensed Online Book*, LK3T-3519, containing *DB2 UDB for OS/390 Version 6 Diagnosis Guide and Reference* in BookManager format, for ordering with the product.
- *DB2 UDB Server for OS/390 Version 6 Online and PDF Library*, SK3T-3518, a collection of books for the DB2 server in BookManager and PDF formats.

Periodically, the books will be refreshed on subsequent editions of these CD-ROMs.

The books for Version 6 of DB2 UDB Server for OS/390 are also available on the following collection kits that contain online books for many IBM products:

- *Online Library Omnibus Edition OS/390 Collection*, SK2T-6700, in English
- *IBM Online Library MVS Collection Kit*, SK88-8002, in Japanese, for viewing on DOS and Windows operating systems.

DB2 education

IBM Education and Training offers a wide variety of classroom courses to help you quickly and efficiently gain DB2 expertise. Classes are scheduled in cities all over the world. You can find class information, by country, at the IBM Learning Services Web site:

http://www.ibm.com/services/learning/

Introduction

For more information, including the current local schedule, please contact your IBM representative.

Classes can also be taught at your location, at a time that suits your needs. Courses can even be customized to meet your exact requirements. The *All-in-One Education and Training Catalog* describes the DB2 curriculum in the United States. You can inquire about or enroll in these courses by calling 1-800-IBM-TEACH (1-800-426-8322).

How to order the DB2 library

You can order DB2 publications and CD-ROMs through your IBM representative or the IBM branch office serving your locality. If you are located within the United States or Canada, you can place your order by calling one of the toll-free numbers :

- In the U.S., call 1-800-879-2755.
- In Canada, call 1-800-565-1234.

To order additional copies of licensed publications, specify the SOFTWARE option. To order additional publications or CD-ROMs, specify the PUBLICATIONS and SLSS option. Be prepared to give your customer number, the product number, and the feature code(s) or order numbers you want.

Definitions of SQL syntax elements

This section contains syntax diagrams for SQL syntax elements. For more information, see *DB2 SQL Reference*.

Expressions

An *expression* specifies a value. The form of an expression is as follows:

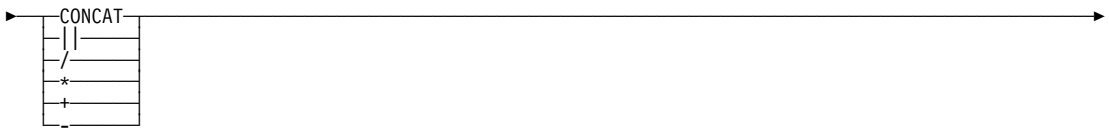
Notes:

1 Includes all functions except table functions.

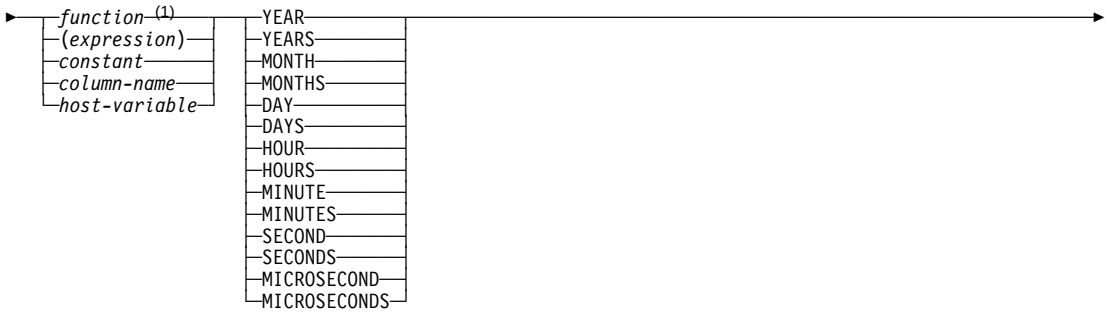
2 See “CASE expressions” on page 9 for more information.

3 See “CAST specification” on page 9 for more information.

operator:



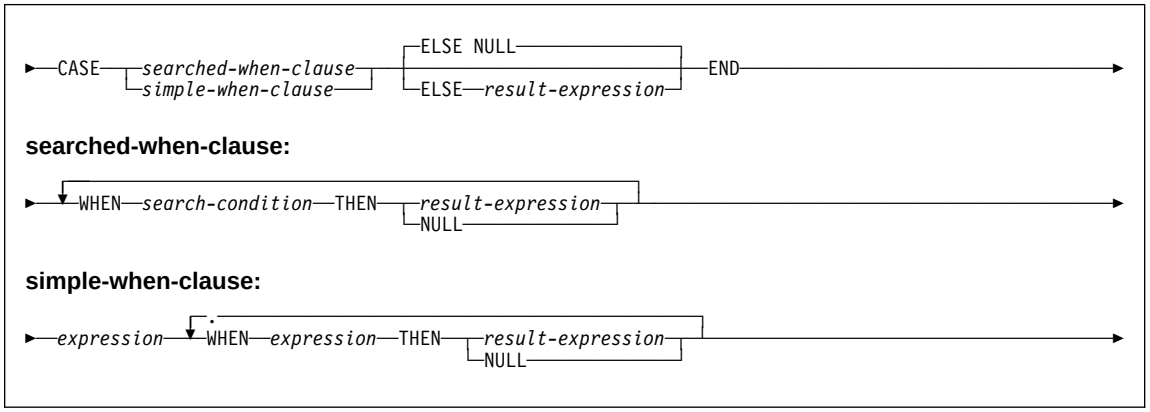
labeled-duration:



Note:

1 Includes all functions except table functions.

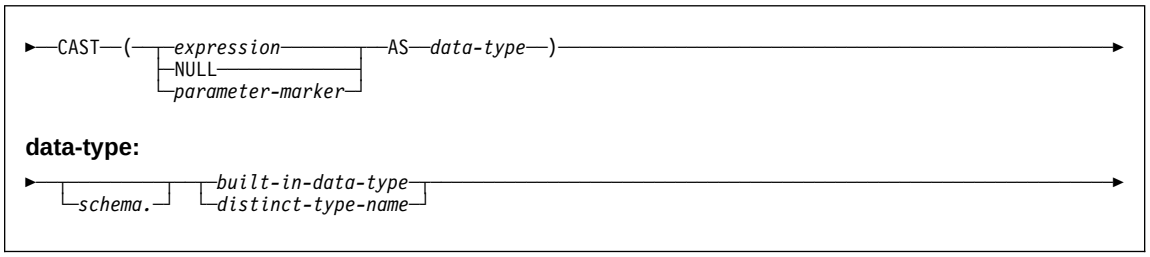
CASE expressions



Example 3: This example shows how to group the results of a query by a CASE expression without having to re-type the expression. Using the sample employee table, find the maximum, minimum, and average salary. Instead of finding these values for each department, assume that you want to combine some departments into the same group.

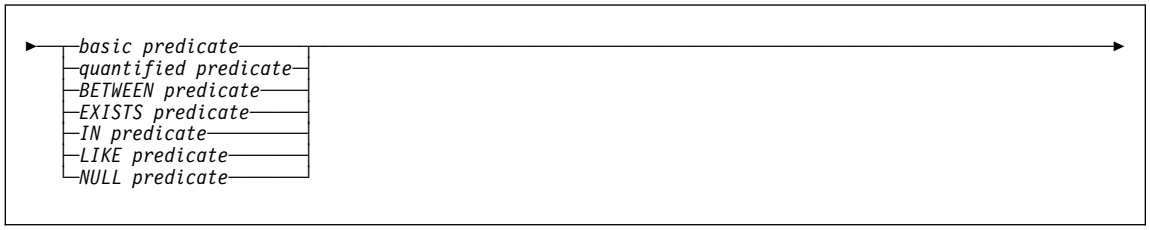
```
SELECT CASE_DEPT,MAX(SALARY),MIN(SALARY),AVG(SALARY)
FROM (SELECT SALARY,CASE WHEN WORKDEPT = 'A00' OR WORKDEPT = 'E21'
                           THEN 'A00_E21'
                         WHEN WORKDEPT = 'D11' OR WORKDEPT = 'E11'
                           THEN 'D11_E11'
                         ELSE WORKDEPT
                    END AS CASE_DEPT
      FROM DSN8610.EMP) X
      GROUP BY CASE_DEPT;
```

CAST specification

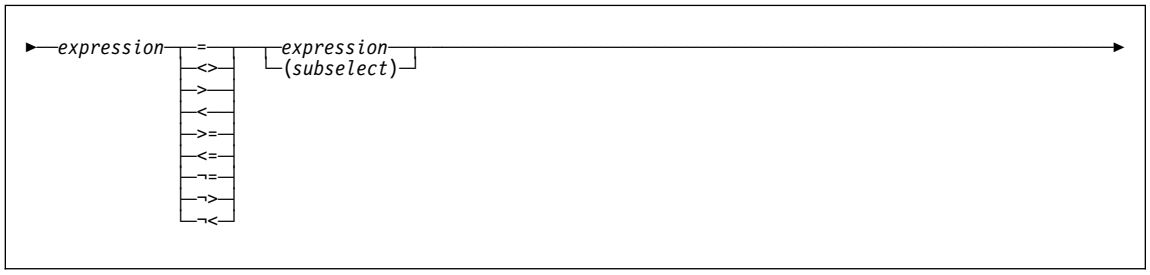


Predicates

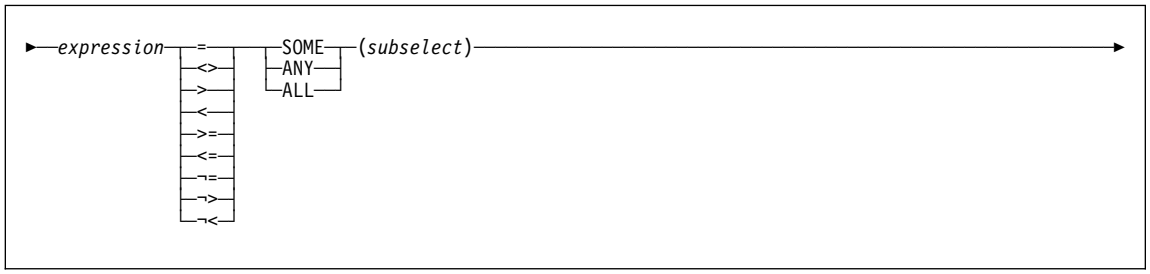
A *predicate* specifies a condition that is true, false, or unknown about a given row or group. The types of predicates are:



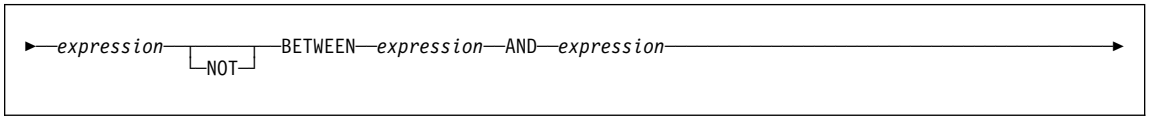
Basic predicate



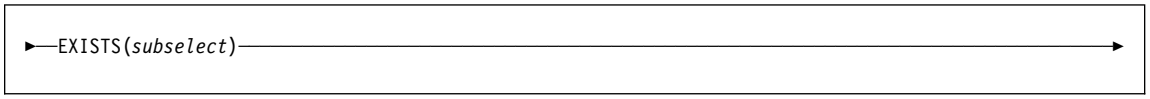
Quantified predicate



BETWEEN predicate

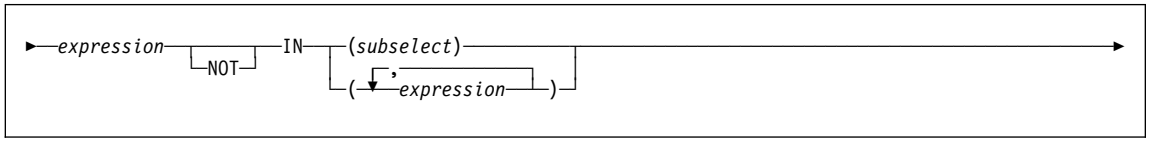


EXISTS predicate

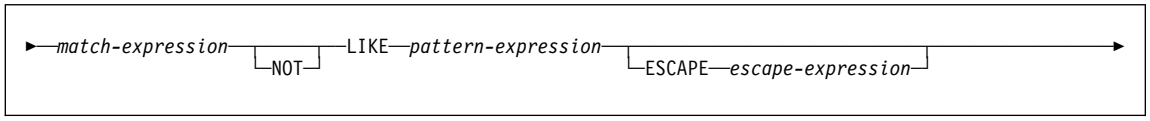


Predicates

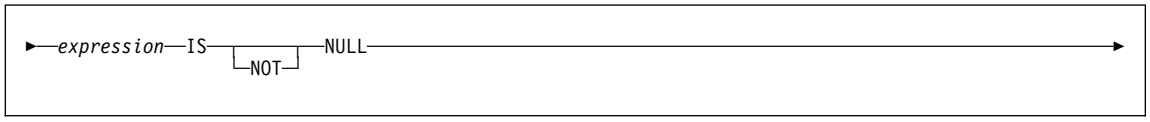
IN predicate



LIKE predicate

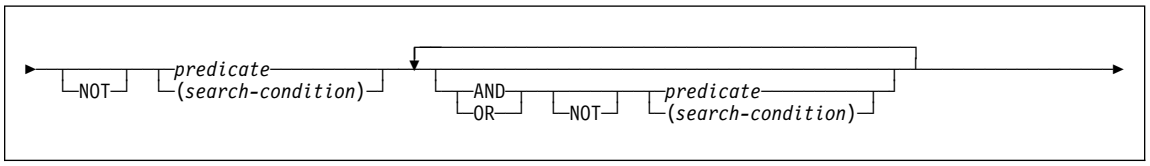


NULL predicate



Search conditions

A *search condition* specifies a condition that is true, false, or unknown about a given row or group. When the condition is true, the row or group qualifies for the results. When the condition is false or unknown, the row or group does not qualify.



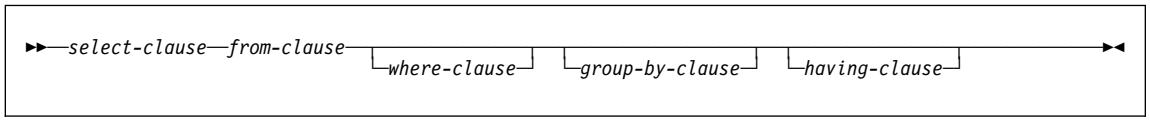
subselect

Queries

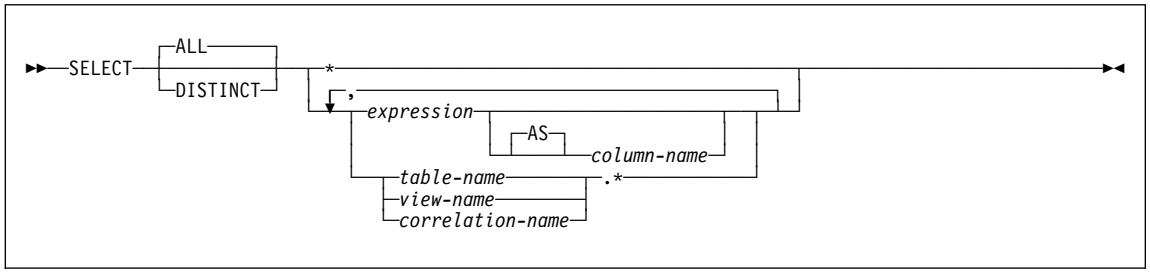
A *query* specifies a result table. A query is a component of certain SQL statements. There are three forms of a query:

A *subselect*
A *fullselect*
A *select-statement*

subselect



select-clause



from-clause

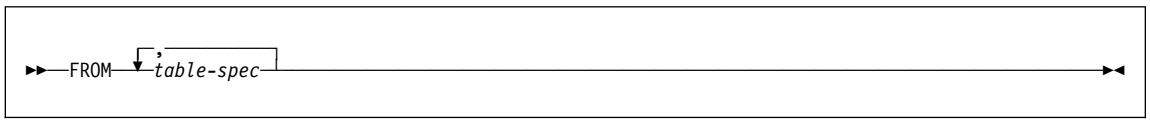
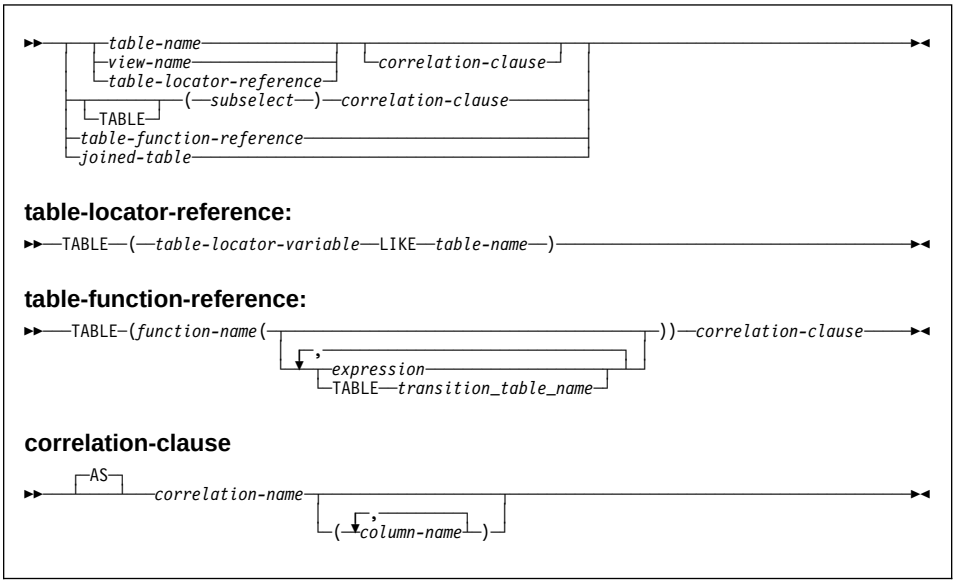
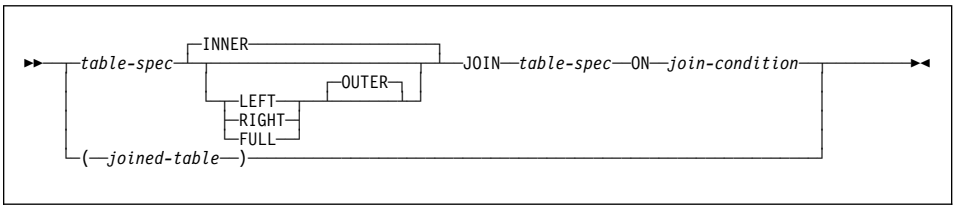


table-spec

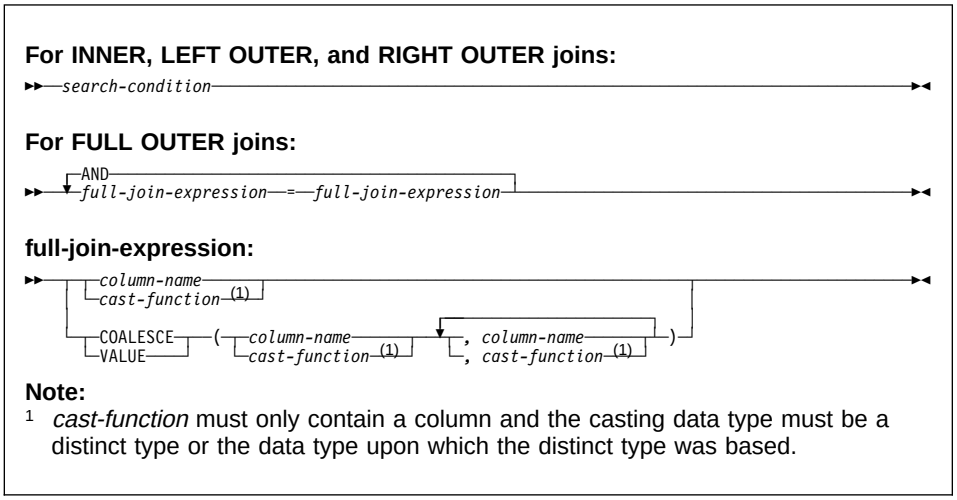


joined-table



subselect

join-condition

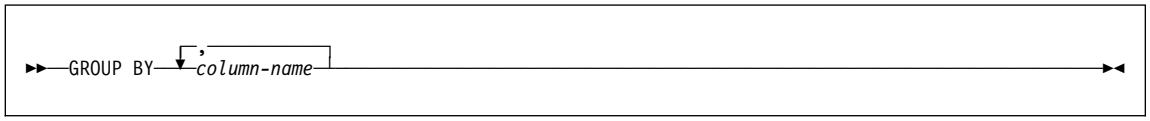


Note:

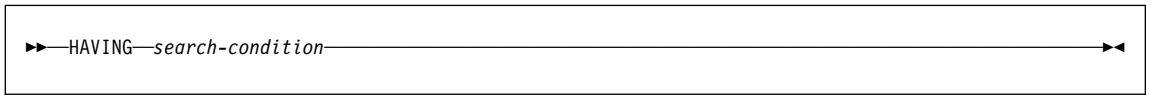
1 *cast-function* must only contain a column and the casting data type must be a distinct type or the data type upon which the distinct type was based.

where-clause

group-by-clause



having-clause



Examples of subselects

Example 1: Show all rows of the table DSN8610.EMP.

```
SELECT * FROM DSN8610.EMP;
```

Example 2: Show the job code, maximum salary, and minimum salary for each group of rows of DSN8610.EMP with the same job code, but only for groups with more than one row and with a maximum salary greater than 50000.

```
SELECT JOB, MAX(SALARY), MIN(SALARY)
  FROM DSN8610.EMP
  GROUP BY JOB
  HAVING COUNT(*) > 1 AND MAX(SALARY) > 50000;
```

Example 3: For each employee in department E11, get the following information from the table DSN8610.EMPPROJACT: employee number, activity number, activity start date, and activity end date. Using the CHAR function, convert the start and end dates to their USA formats. Get the needed department information from the table DSN8610.EMP:

```
SELECT EMPNO, ACTNO, CHAR(EMSTDATE,USA), CHAR(EMENDATE,USA)
  FROM DSN8610.EMPPROJACT
  WHERE EMPNO IN (SELECT EMPNO FROM DSN8610.EMP
                  WHERE WORKDEPT = 'E11');
```

Example 4: Show the department number and maximum departmental salary for all departments whose maximum salary is less than the average salary for all employees. (In this example, the subquery would be executed only once.)

```
SELECT WORKDEPT, MAX(SALARY)
  FROM DSN8610.EMP
  GROUP BY WORKDEPT
  HAVING MAX(SALARY) < (SELECT AVG(SALARY)
                        FROM DSN8610.EMP);
```

Example 5: Show the department number and maximum departmental salary for all departments whose maximum salary is less than the average salary for employees in all other departments. (In contrast to Example 4, the subquery in this statement, containing a correlated reference, would need to be executed for each group.)

```
SELECT WORKDEPT, MAX(SALARY)
  FROM DSN8610.EMP Q
  GROUP BY WORKDEPT
  HAVING MAX(SALARY) < (SELECT AVG(SALARY)
                        FROM DSN8610.EMP
                        WHERE NOT WORKDEPT = Q.WORKDEPT);
```

Example 6: For each group of employees hired during the same year, show the year-of-hire and current average salary. (This example demonstrates how to use the AS clause in a FROM clause to name a derived column that you want to refer to in a GROUP BY clause.)

```
SELECT HIREYEAR, AVG(SALARY)
  FROM (SELECT (YEAR(HIREDATE) AS HIREYEAR, SALARY
          FROM DSN8610.EMP) AS NEWEMP
  GROUP BY HIREYEAR;
```

subselect

Example 7: Get the employee number and employee name for all the employees in DSN8610.EMP. Order the results by the date of hire.

```
SELECT EMPNO, FIRSTNME, LASTNAME
  FROM DSN8610.EMP
  ORDER BY HIREDATE;
```

Example 8: Assume that an external function named ADDYEARS exists. For a given date, the function adds a given number of years and returns a new date. (The data types of the two input parameters to the function are DATE and INTEGER.) Get the employee number and employee name for all employees who have been hired within the last 5 years.

```
SELECT EMPNO, FIRSTNME, LASTNAME
  FROM DSN8610.EMP
  WHERE ADDYEARS(HIREDATE, 5) > CURRENT DATE;
```

To distinguish the different types of joins, to show nested table expressions, and to demonstrate how to combine join columns, the remaining examples use these two tables:

The PARTS table

PART	PROD#	SUPPLIER
WIRE	10	ACWF
OIL	160	WESTERN_CHEM
MAGNETS	10	BATEMAN
PLASTIC	30	PLASTIK_CORP
BLADES	205	ACE_STEEL

The PRODUCTS table

PROD#	PRODUCT	PRICE
505	SCREWDRIVER	3.70
30	RELAY	7.55
205	SAW	18.90
10	GENERATOR	45.75

Example 9: Join the tables on the PROD# column to get a table of parts with their suppliers and the products that use the parts:

```
SELECT PART, SUPPLIER, PARTS.PROD#, PRODUCT
   FROM PARTS, PRODUCTS
   WHERE PARTS.PROD# = PRODUCTS.PROD#;
```

or

```
SELECT PART, SUPPLIER, PARTS.PROD#, PRODUCT
  FROM PARTS INNER JOIN PRODUCTS
    ON PARTS.PROD# = PRODUCTS.PROD#;
```

Either one of these two statements give this result:

```
PART        SUPPLIER       PROD#     PRODUCT
=======     ============   =====     ==========
WIRE        ACWF           10        GENERATOR
MAGNETS     BATEMAN        10        GENERATOR
PLASTIC     PLASTIK_CORP   30        RELAY
BLADES      ACE_STEEL      205       SAW
```

You can specify more complicated join conditions to obtain different sets of results. For example, to eliminate the suppliers that begin with the letter A from the table of parts, suppliers, product numbers and products, write a query like this:

```
SELECT PART, SUPPLIER, PARTS.PROD#, PRODUCT
  FROM PARTS INNER JOIN PRODUCTS
    ON PARTS.PROD# = PRODUCTS.PROD#
    AND SUPPLIER NOT LIKE 'A%';
```

The result of the query is all rows that do not have a supplier that begins with A:

```
PART        SUPPLIER       PROD#     PRODUCT
=======     ============   =====     ==========
MAGNETS     BATEMAN        10        GENERATOR
PLASTIC     PLASTIK_CORP   30        RELAY
```

subselect

Example 10: Join the tables on the PROD# column to get a table of all parts and products, showing the supplier information, if any.

```
SELECT PART, SUPPLIER, PARTS.PROD#, PRODUCT
  FROM PARTS FULL OUTER JOIN PRODUCTS
    ON PARTS.PROD# = PRODUCTS.PROD#;
```

The result is:

```
PART        SUPPLIER       PROD#     PRODUCT
=======     ============   =====     ==========
WIRE        ACWF           10        GENERATOR
MAGNETS     BATEMAN        10        GENERATOR
PLASTIC     PLASTIK_CORP   30        RELAY
BLADES      ACE_STEEL      205       SAW
OIL         WESTERN_CHEM   160       (null)
(null)      (null)         (null)    SCREWDRIVER
```

Example 11: Join the tables on the PROD# column to get a table of all parts, showing what products, if any, the parts are used in:

```
SELECT PART, SUPPLIER, PARTS.PROD#, PRODUCT
  FROM PARTS LEFT OUTER JOIN PRODUCTS
    ON PARTS.PROD# = PRODUCTS.PROD#;
```

The result is:

```
PART        SUPPLIER       PROD#     PRODUCT
=======     ============   =====     ==========
WIRE        ACWF           10        GENERATOR
MAGNETS     BATEMAN        10        GENERATOR
PLASTIC     PLASTIK_CORP   30        RELAY
BLADES      ACE_STEEL      205       SAW
OIL         WESTERN_CHEM   160       (null)
```

Example 12: Join the tables on the PROD# column to get a table of all products, showing the parts used in that product, if any, and the supplier.

```
SELECT PART, SUPPLIER, PRODUCTS.PROD#, PRODUCT
  FROM PARTS RIGHT OUTER JOIN PRODUCTS
    ON PARTS.PROD# = PRODUCTS.PROD#;
```

The result is:

```
PART        SUPPLIER       PROD#     PRODUCT
=======     ============   =====     ===========
WIRE        ACWF           10        GENERATOR
MAGNETS     BATEMAN        10        GENERATOR
PLASTIC     PLASTIK_CORP   30        RELAY
BLADES      ACE_STEEL      205       SAW
(null)      (null)         505       SCREWDRIVER
```

Example 13: The result of “Example 10” (a full outer join) shows the product number for SCREWDRIVER as null, even though the PRODUCTS table contains a product number for it. This is because PRODUCTS.PROD# was not listed in the SELECT list of the query. Revise the query using COALESCE, a synonym for the VALUE function, so that all part numbers from both tables are shown.

```
SELECT PART, SUPPLIER,
  COALESCE(PARTS.PROD#, PRODUCTS.PROD#) AS PRODNUM, PRODUCT
   FROM PARTS FULL OUTER JOIN PRODUCTS
     ON PARTS.PROD# = PRODUCTS.PROD#;
```

In the result, notice that the AS clause (AS PRODNUM), provides a name for the result of the COALESCE function:

```
PART        SUPPLIER       PRODNUM   PRODUCT
=======     ============   =======   ===========
WIRE        ACWF           10        GENERATOR
MAGNETS     BATEMAN        10        GENERATOR
PLASTIC     PLASTIK_CORP   30        RELAY
BLADES      ACE_STEEL      205       SAW
OIL         WESTERN_CHEM   160       (null)
(null)      (null)         505       SCREWDRIVER
```

Example 14: For all parts that are used in product numbers less than 200, show the part, the part supplier, the product number, and the product name. Use a nested table expression.

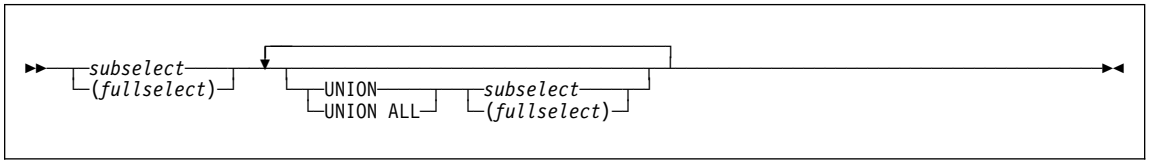
```
SELECT PART, SUPPLIER, PRODNUM, PRODUCT
  FROM (SELECT PART, PROD# AS PRODNUM, SUPPLIER
              FROM PARTS
                 WHERE PROD# < 200) AS PARTX
           LEFT OUTER JOIN PRODUCTS
               ON PRODNUM = PROD#;
```

The result is:

subselect

```
PART        SUPPLIER         PRODNUM     PRODUCT
=======     ============     =======     ==========
WIRE        ACWF             10          GENERATOR
MAGNETS     BATEMAN          10          GENERATOR
PLASTIC     PLASTIK_CORP     30          RELAY
OIL         WESTERN_CHEM     160         (null)
```

fullselect



Examples of fullselects

Example 1: A query specifies the union of result tables R1 and R2. A column in R1 has the data type CHAR(10) and the subtype BIT. The corresponding column in R2 has the data type CHAR(15) and the subtype SBCS. Hence, the column in the union has the data type CHAR(15) and the subtype BIT. Values from the first column are converted to CHAR(15) by adding five trailing blanks.

Example 2: Show all the rows from DSN8610.EMP.

```
SELECT * FROM DSN8610.EMP;
```

Example 3: Using sample tables DSN8610.EMP and DSN8610.EMPROJACT, list the employee numbers of all employees for which either of the following statements are true:

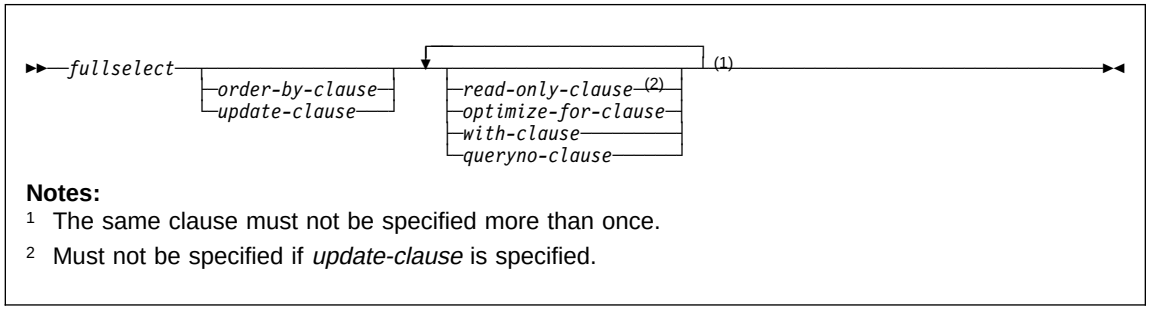
- Their department numbers begin with `'D'`.
- They are assigned to projects whose project numbers begin with `'AD'`.

```
SELECT EMPNO FROM DSN8610.EMP
  WHERE WORKDEPT LIKE 'D%'
  UNION
  SELECT EMPNO FROM DSN8610.EMPPROJACT
  WHERE PROJNO LIKE 'AD%';
```

The result is the union of two result tables, one formed from the sample table DSN8610.EMP, the other formed from the sample table DSN8610.EMPPROJACT. The result—a one-column table—is a list of employee numbers. Because UNION, rather than UNION ALL, was used, the entries in the list are distinct. If instead UNION ALL were used, certain employee numbers would appear in the list more than once. These would be the numbers for employees in departments that begin with `'D'` while their projects begin with `'AD'`.

select-statement

select-statement

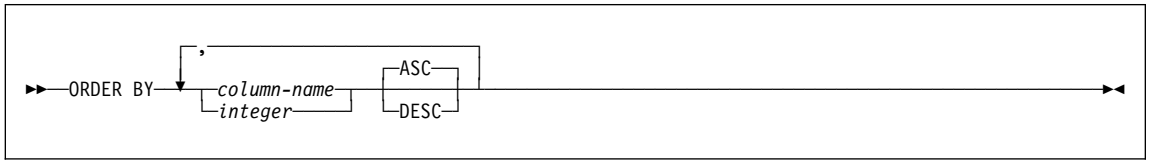


Notes:

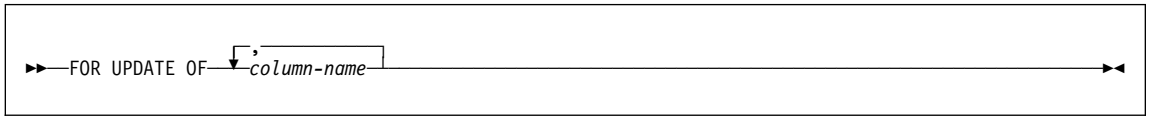
1 The same clause must not be specified more than once.

2 Must not be specified if *update-clause* is specified.

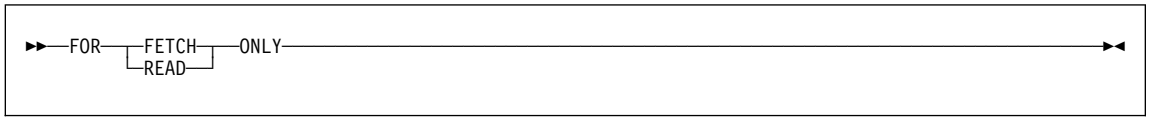
order-by-clause



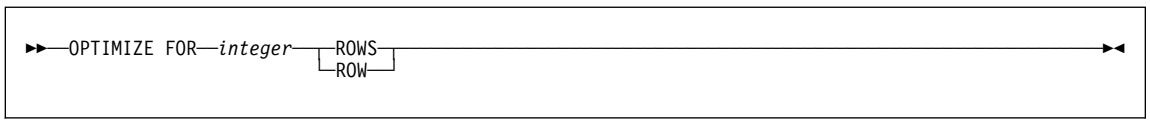
update-clause



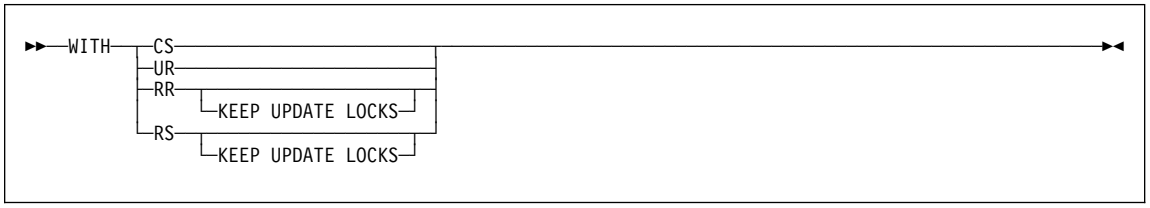
read-only-clause



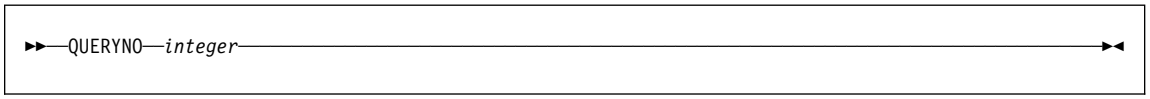
optimize-for-clause



with-clause



queryno-clause



Examples of select statements

Example 1: Select all the rows from DSN8610.EMP.

```
SELECT * FROM DSN8610.EMP;
```

Example 2: Select all the rows from DSN8610.EMP, arranging the result table in chronological order by date of hiring.

```
SELECT * FROM DSN8610.EMP ORDER BY HIREDATE;
```

Example 3: Select the department number (WORKDEPT) and average departmental salary (SALARY) for all departments in the table DSN8610.EMP. Arrange the result table in ascending order by average departmental salary.

```
SELECT WORKDEPT, AVG(SALARY)
  FROM DSN8610.EMP
  GROUP BY WORKDEPT
  ORDER BY 2;
```

Example 4: Change various salaries, bonuses, and commissions in the table DSN8610.EMP. Confine the changes to employees in departments D11 and D21. Use positioned updates to do this with a cursor named UP_CUR. Indicate the columns to be updated in a FOR UPDATE of clause in the cursor declaration. Below is the declaration for a PL/I program.

```
EXEC SQL DECLARE UP_CUR CURSOR FOR
  SELECT WORKDEPT, EMPNO, SALARY, BONUS, COMM
    FROM DSN8610.EMP
    WHERE WORKDEPT IN ('D11','D21')
    FOR UPDATE OF SALARY, BONUS, COMM;
```

Example 5: Find the maximum, minimum, and average bonus in the table DSN8610.EMP. Execute the statement with uncommitted read isolation, regardless of

the value of ISOLATION with which the plan or package containing the statement is bound. Assign 13 as the query number for the SELECT statement.

```
EXEC SQL
  SELECT MAX(BONUS), MIN(BONUS), AVG(BONUS)
    INTO :MAX, :MIN, :AVG
    FROM DSN8610.EMP
    WITH UR
    QUERYNO 13;
```

If bind option EXPLAIN(YES) is specified, rows are inserted into the plan table. The value used for the QUERYNO column for these rows is 13.

Example 6: The cursor declaration shown below is in a PL/I program. In the query within the declaration, X.RMT_TAB is an alias for a table at some other DB2. Hence, when the query is used, it is processed using DB2 private protocol access.

The declaration indicates that no positioned updates or deletes will be done with the query's cursor. It also specifies that the access path for the query be optimized for the retrieval of at most 50 rows. Even so, the program can retrieve more than 50 rows from the result table, which consists of the entire table identified by the alias. However, when more than 50 rows are retrieved, performance could possibly degrade.

```
EXEC SQL DECLARE C1 CURSOR FOR
  SELECT * FROM X.RMT_TAB
  OPTIMIZE FOR 50 ROWS
  FOR FETCH ONLY;
```

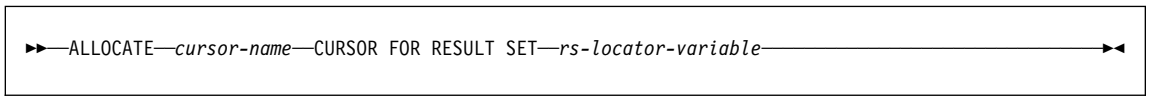
SQL alphabetic reference

This section contains syntax diagrams and examples of SQL statements organized alphabetically by name. For more information, see *DB2 SQL Reference*

ALLOCATE CURSOR

ALLOCATE CURSOR

Syntax



Example

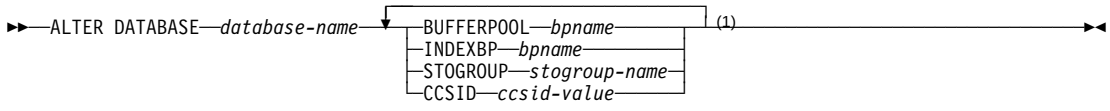
The statement in the following example is assumed to be in a PL/I program.

Define and associate cursor C1 with the result set locator variable LOC1 and the related result set returned by the stored procedure:

```
EXEC SQL ALLOCATE C1 CURSOR FOR RESULT SET :LOC1;
```

ALTER DATABASE

Syntax



Note:
1 The same clause must not be specified more than once.

Example

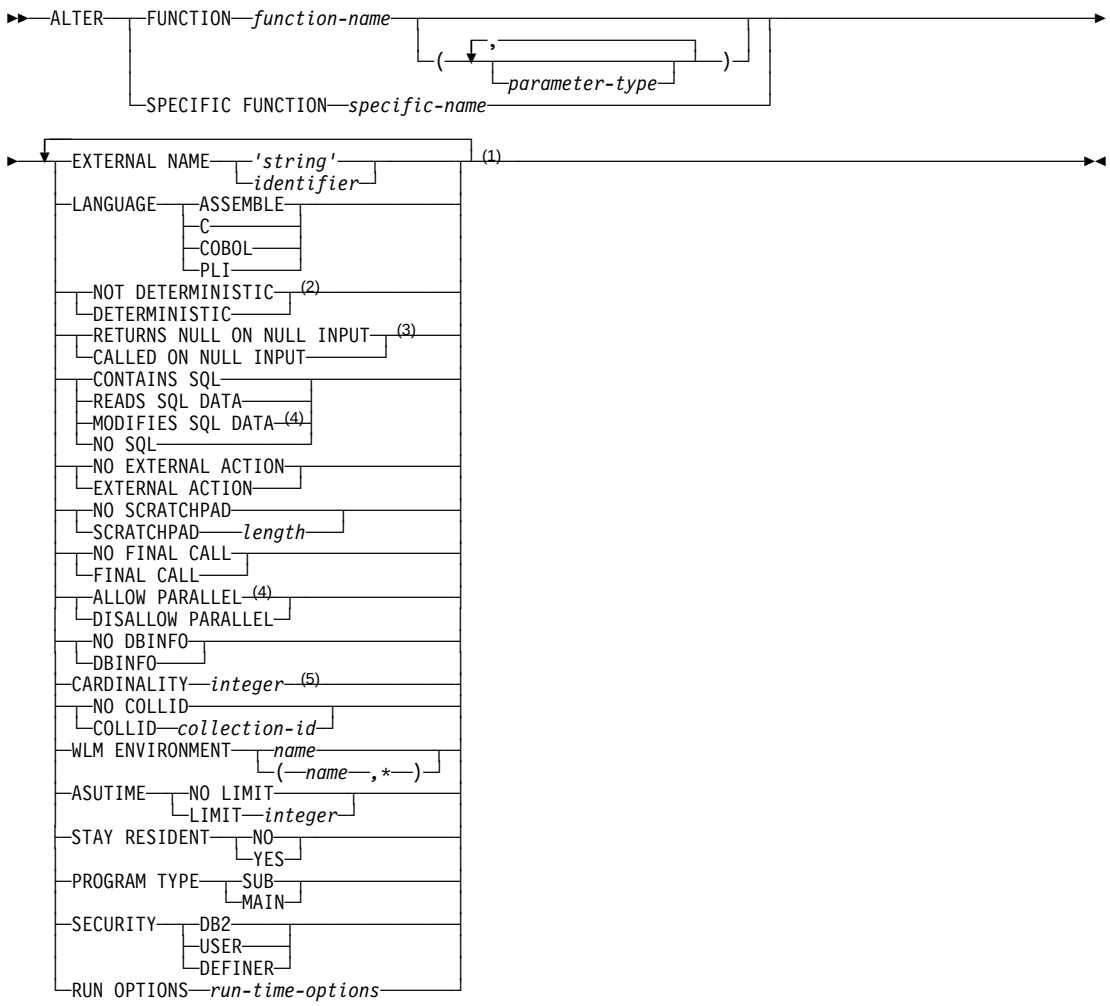
Change the default buffer pool for both table spaces and indexes within database ABCDE to BP2.

```
ALTER DATABASE ABCDE
  BUFFERPOOL BP2
  INDEXBP BP2;
```

ALTER FUNCTION

Syntax

ALTER FUNCTION



Notes:

1 The same clause must not be specified more than once.

2 Synonyms for this clause include VARIANT for NOT DETERMINISTIC, and NOT VARIANT for DETERMINISTIC.

3 Synonyms for this clause include NOT NULL CALL for RETURNS NULL ON NULL INPUT, and NULL CALL for CALLED ON NULL INPUT.

4 MODIFIES SQL DATA is not supported for *external table functions*.

5 CARDINALITY is not supported for *external scalar functions*.

ALTER FUNCTION

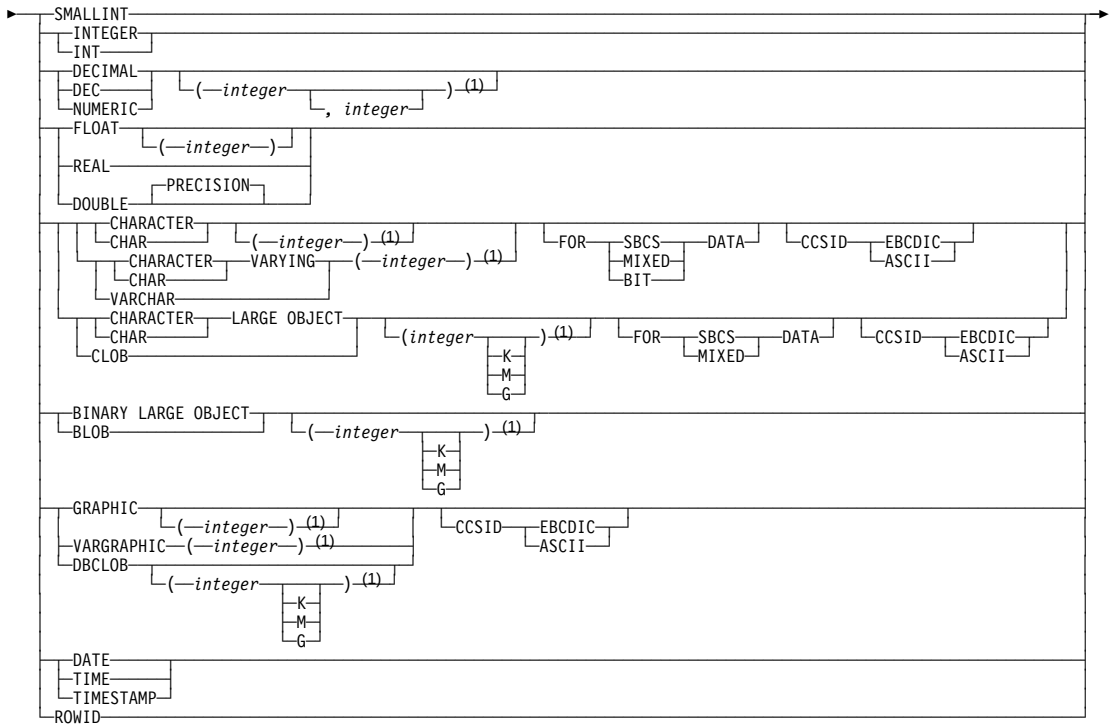
parameter-type:

Note:

1 AS LOCATOR can be specified only for a LOB data type or a distinct type based on a LOB data type.

data-type:

built-in-data-type:



Notes:

1 The values that are specified for length, precision, or scale attributes must match the values that were specified when the function was created. Coding specific values is optional. Empty parentheses, (), can be used instead to indicate that DB2 ignores the attributes when determining whether data types match.

2 The value that is specified does not have to match the value that was specified when the function was created because matching is based on data type (REAL or DOUBLE). 1<=*integer*<= 21 indicates REAL and 22<=*integer*<=53 indicates DOUBLE. Coding a specific value is optional. Empty parentheses cannot be used.

Examples

Example 1: Assume that there are two functions CENTER in the PELLOW schema. The first function has two input parameters with INTEGER and FLOAT data types, respectively. The specific name for the first function is FOCUS1. The second function has three parameters with CHAR(25), DEC(5,2), and INTEGER data types.

Using the specific name to identify the function, change the WLM environment in which the first function runs from WLMENVNAME1 to WLMENVNAME2.

```
ALTER SPECIFIC FUNCTION PELLOW.FOCUS1 WLM ENVIRONMENT WLMENVNAME2;
```

Example 2: Change the second function that is described in *Example 1* so that it is not invoked when any of the arguments are null. Use the function signature to identify the function,

```
ALTER FUNCTION PELLOW.CENTER (CHAR(25), DEC(5,2), INTEGER)
   RETURNS NULL ON NULL INPUT;
```

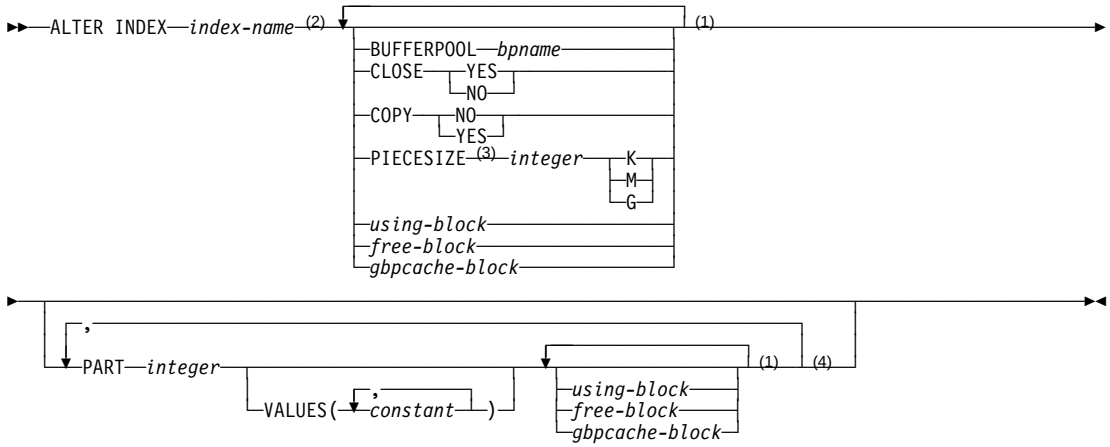
You can also code the ALTER FUNCTION statement without the exact values for the CHAR and DEC data types:

```
ALTER FUNCTION PELLOW.CENTER (CHAR(), DEC(), INTEGER)
   RETURNS NULL ON NULL INPUT;
```

If you use empty parentheses, DB2 ignores the length, precision, and scale attributes when looking for matching data types to find the function.

ALTER INDEX

Syntax



Notes:

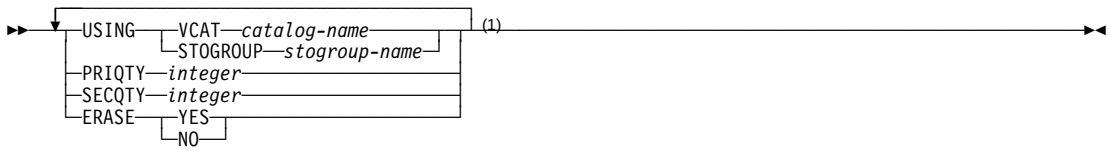
1 The same clause must not be specified more than once.

2 At least one clause must be specified after *index-name*.

3 The PIECESIZE clause can only be specified for nonpartitioning indexes.

4 The PART clause can only be specified for partitioning indexes. Any PART clauses that are specified must be specified last.

using-block:

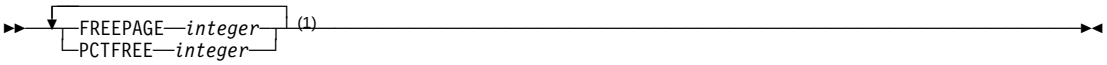


Note:

1 The same clause must not be specified more than once.

ALTER INDEX

free-block:



Note:
1 The same clause must not be specified more than once.

gbpcache-block:

Examples

Example 1: Alter the index DSN8610.XEMP1. Indicate that DB2 is not to close the data sets that support the index when there are no current users of the index.

```
ALTER INDEX DSN8610.XEMP1
  CLOSE NO;
```

Example 2: Alter the index DSN8610.XPROJ1. Use BP1 as the buffer pool that is to be associated with the index, indicate that full image or concurrent copies on the index are allowed, and change the maximum size of each data set to 8 megabytes.

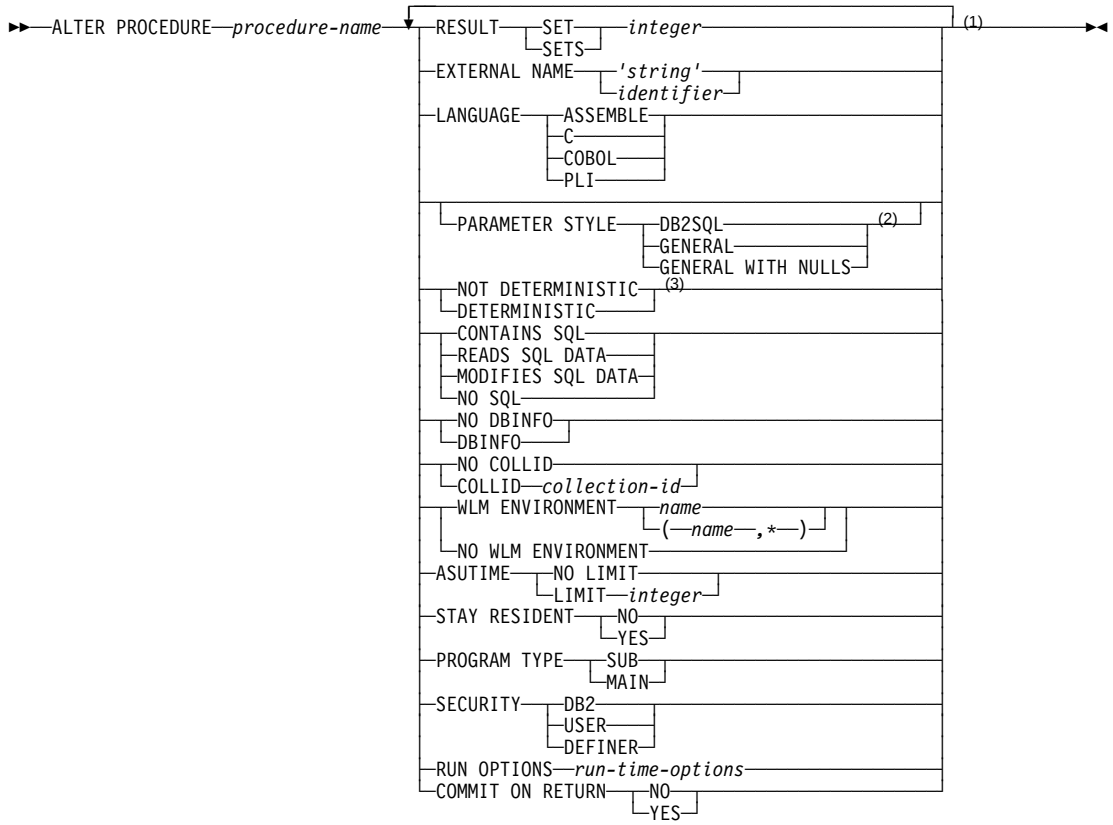
```
ALTER INDEX DSN8610.XPROJ1
  BUFFERPOOL BP1
  COPY YES
  PIECESIZE 8M;
```

Example 3: Alter partitioned index DSN8610.DEPT1. For partition 3, leave one page of free space for every 13 pages and 13 percent of free space per page. For partition 5, leave one page for every 25 pages and 25 percent of free space. For all the other partitions, leave one page of free space for every 6 pages and 11 percent of free space. Ensure that index pages are cached to the group buffer pool for all partitions except partition 4. For partition 4, write pages only when there is inter-DB2 R/W interest on the partition.

```
ALTER INDEX DSN8610.XDEPT1
   BUFFERPOOL BP1
   CLOSE YES
   COPY YES
   USING VCAT CATLGG
   FREEPAGE 6
   PCTFREE 11
   GBPCACHE ALL
   PART 3
      USING VCAT CATLGG
      FREEPAGE 13
      PCTFREE 13,
   PART 4
      USING VCAT CATLGG
      GBPCACHE CHANGED,
   PART 5
      USING VCAT CATLGG
      FREEPAGE 25
      PCTFREE 25;
```

ALTER PROCEDURE

Syntax



Notes:

1. The same clause must not be specified more than once.
2. Synonyms for the clause include STANDARD CALL for DB2SQL, SIMPLE CALL for GENERAL, and SIMPLE CALL WITH NULLS for GENERAL WITH NULLS.
3. Synonyms for the clause include VARIANT for NOT DETERMINISTIC, and NOT VARIANT is a synonym for DETERMINISTIC.

Example

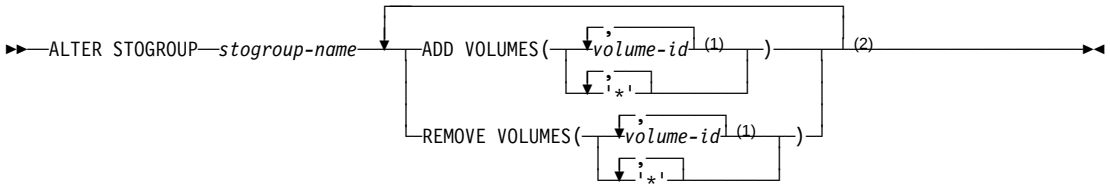
Assume that stored procedure SYSPROC.MYPROC is currently defined to run in WLM environment PARTSA and that you have appropriate authority on that WLM

environment and WLM environment PARTSEC. Change the definition of the stored procedure so that it runs in PARTSEC.

```
ALTER PROCEDURE SYSPROC.MYPROC WLM ENVIRONMENT PARTSEC;
```

ALTER STOGROUP

Syntax



Notes:

1. The same *volume-id* must not be specified more than once in the same clause.
2. The same clause must not be specified more than once.

Examples

Example 1: Alter storage group DSN8G610. Add volumes DSNV04 and DSNV05.

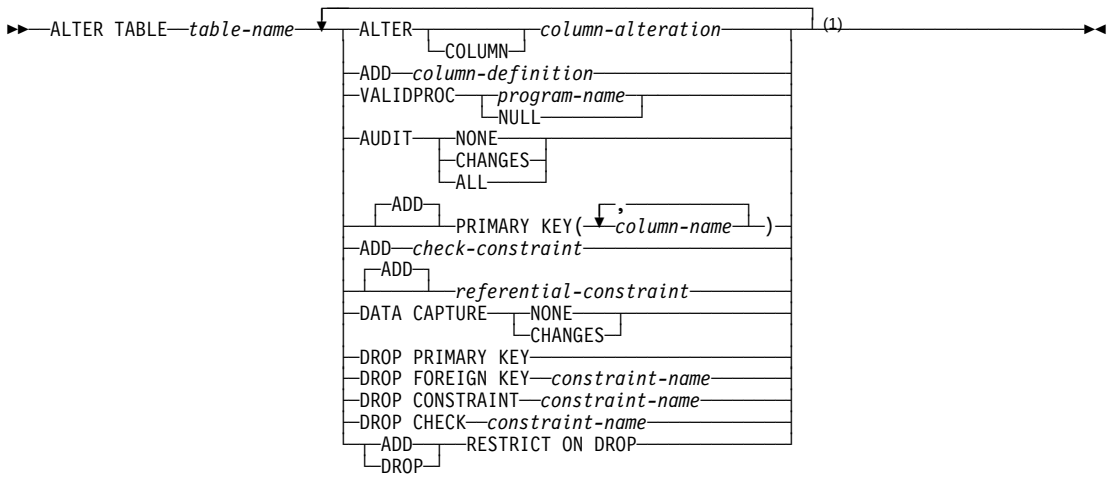
```
ALTER STOGROUP DSN8G610
    ADD VOLUMES (DSNV04,DSNV05);
```

Example 2: Alter storage group DSN8G610. Remove volumes DSNV04 and DSNV05.

```
ALTER STOGROUP DSN8G610
  REMOVE VOLUMES (DSNV04,DSNV05);
```

ALTER TABLE

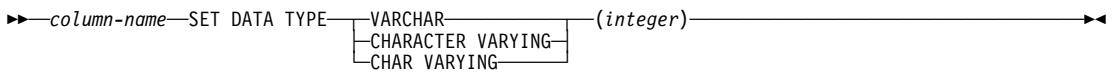
Syntax



Note:

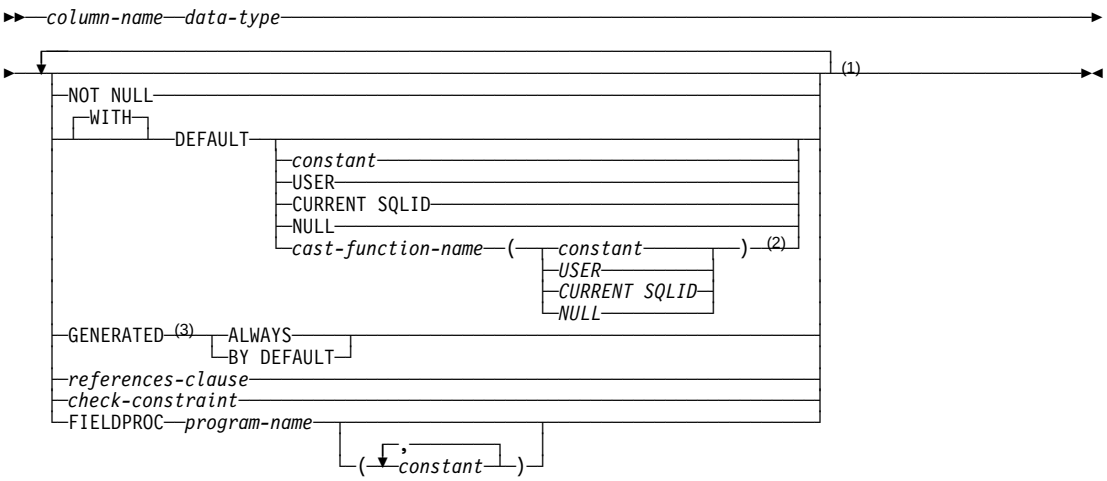
1 The same clause must not be specified more than once, except for the ALTER COLUMN clause, which can be specified more than once. Do not specify DROP CONSTRAINT if DROP FOREIGN KEY or DROP CHECK is specified.

column-alteration:



ALTER TABLE

column-definition:



Notes:

1 The same clause must not be specified more than once.

2 This form of the DEFAULT value can only be used with columns that are defined as a distinct type.

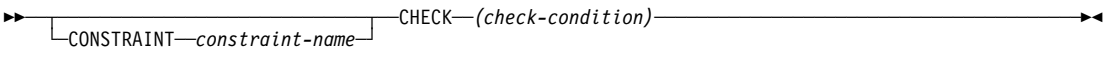
3 GENERATED can be specified only for a column that is defined as a ROWID data type (or a distinct type that is based on a ROWID data type).

data-type:

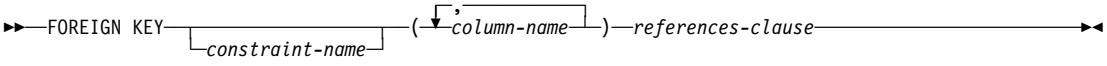
Note:

1 For the syntax, see “built-in-data-type” on page 86.

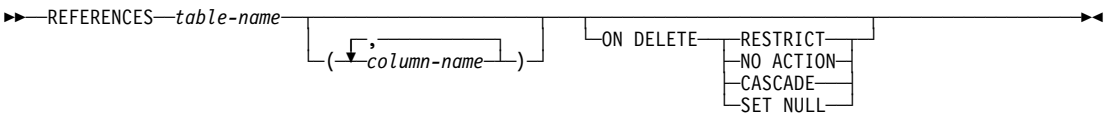
check-constraint:



referential-constraint:



references-clause:



Examples

Example 1: Column DEPTNAME in table DSN8610.DEPT was created as a VARCHAR(36). Increase its length to 50 bytes. Also, add the column BLDG to the table DSN8610.DEPT. Describe the new column as a character string column that holds SBCS data.

```
ALTER TABLE DSN8610.DEPT
  ALTER COLUMN DEPTNAME SET DATA TYPE VARCHAR(50)
  ADD BLDG CHAR(3) FOR SBCS DATA;
```

Example 2: Assign a validation procedure named DSN8EAEM to the table DSN8610.EMP.

```
ALTER TABLE DSN8610.EMP
  VALIDPROC DSN8EAEM;
```

Example 3: Disassociate the current validation procedure from the table DSN8610.EMP. After the statement is executed, the table no longer has a validation procedure.

```
ALTER TABLE DSN8610.EMP
  VALIDPROC NULL;
```

Example 4: Define ADMRDEPT as the foreign key of a self-referencing constraint on DSN8610.DEPT.

```
ALTER TABLE DSN8610.DEPT
  FOREIGN KEY(ADMRDEPT) REFERENCES DSN8610.DEPT ON DELETE CASCADE;
```

Example 5: Add a check constraint to the table DSN8610.EMP which checks that the minimum salary an employee can have is $10,000.

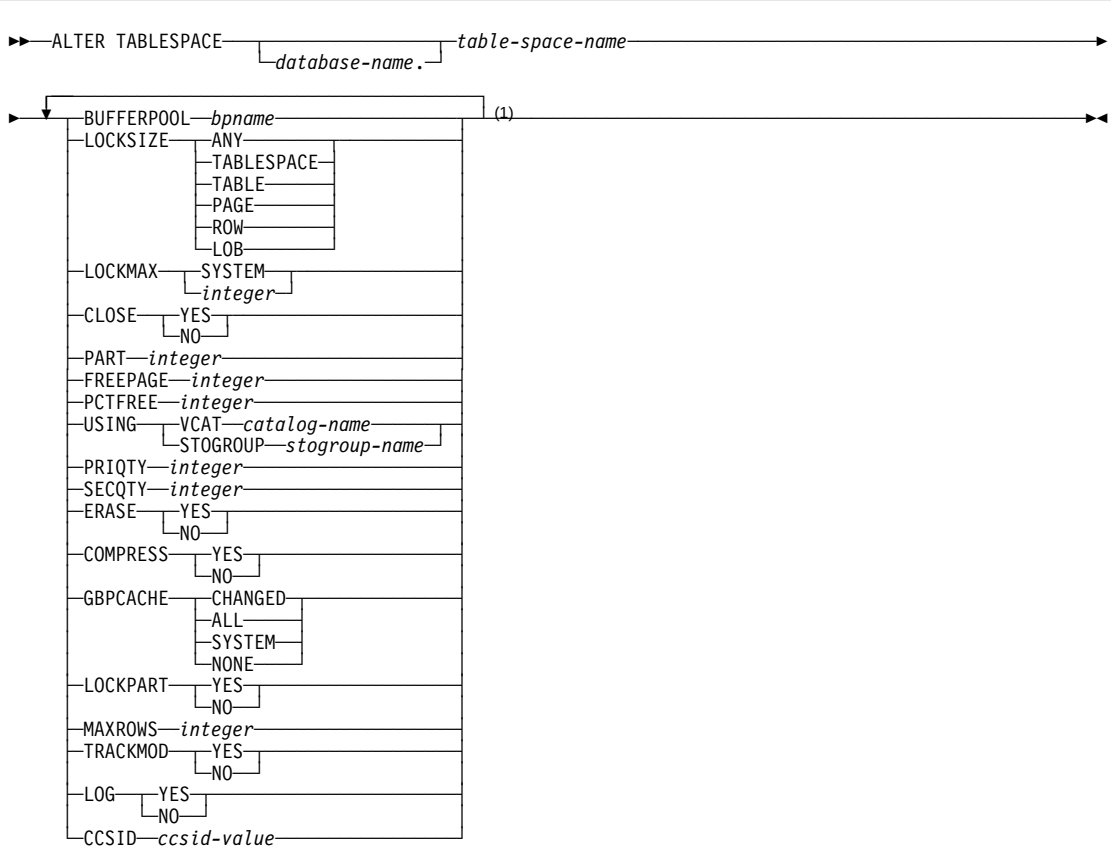
```
ALTER TABLE DSN8610.EMP
  ADD CHECK (SALARY >= 10000);
```

Example 6: Alter the PRODINFO table to define a foreign key that references a non-primary unique key in the product version table (PRODVER_1). The columns of the unique key are VERNAME, RELNO.

```
ALTER TABLE PRODINFO
  FOREIGN KEY (PRODNAME,PRODVERNO)
    REFERENCES PRODVER_1 (VERNAME,RELNO) ON DELETE RESTRICT;
```

ALTER TABLESPACE

Syntax



Note:
1 The same clause must not be specified more than once.

Examples

Example 1: Alter table space DSN8S61E in database DSN8D61A. CLOSE NO means that the data sets of the table space are not to be closed when there are no current users of the table space.

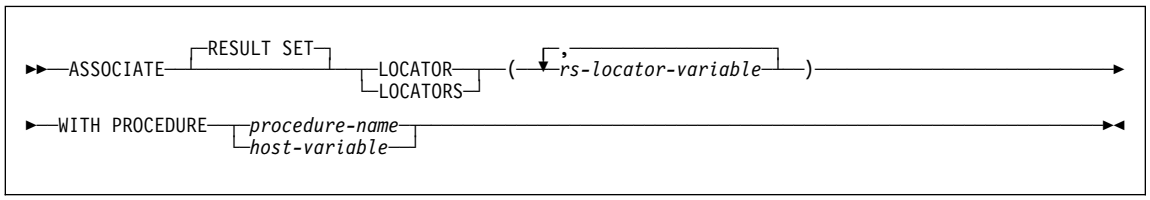
```
ALTER TABLESPACE DSN8D61A.DSN8S61E
  CLOSE NO;
```

Example 2: Alter table space DSN8S61D in database DSN8D61A. BP2 is the buffer pool associated with the table space. PAGE is the level at which locking is to take place.

```
ALTER TABLESPACE DSN8D61A.DSN8S61D
  BUFFERPOOL BP2
  LOCKSIZE PAGE;
```

ASSOCIATE LOCATORS

Syntax



Examples

The statements in the following examples are assumed to be in PL/I programs.

Example 1: Use result set locator variables LOC1 and LOC2 to get the result set locator values for the two result sets returned by stored procedure P1. Assume that the stored procedure is called with a one-part name from current server SITE2.

```
EXEC SQL CONNECT TO SITE2;
EXEC SQL CALL P1;
EXEC SQL ASSOCIATE RESULT SET LOCATORS (:LOC1, :LOC2)
         WITH PROCEDURE P1;
```

Example 2: Repeat the scenario in Example 1, but use a two-part name to specify an explicit schema name for the stored procedure to ensure that stored procedure P1 in schema MYSCHEMA is used.

```
EXEC SQL CONNECT TO SITE2;
EXEC SQL CALL MYSCHEMA.P1;
EXEC SQL ASSOCIATE RESULT SET LOCATORS (:LOC1, :LOC2)
         WITH PROCEDURE MYSCHEMA.P1;
```

Example 3: Use result set locator variables LOC1 and LOC2 to get the result set locator values for the two result sets that are returned by the stored procedure named by host variable HV1. Assume that host variable HV1 contains the value SITE2.MYSCHEMA.P1 and the stored procedure is called with a three-part name.

```
EXEC SQL CALL SITE2.MYSCHEMA.P1;
EXEC SQL ASSOCIATE LOCATORS (:LOC1, :LOC2)
         WITH PROCEDURE :HV1;
```

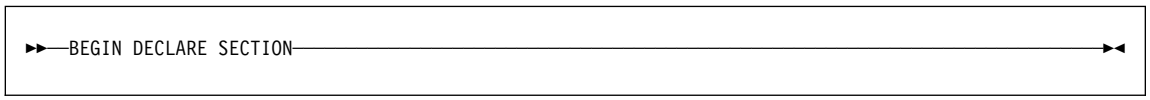
The preceding example would be invalid if host variable HV1 had contained the value MYSCHEMA.P1, a two-part name. For the example to be valid with that two-part name in host variable HV1, the current server must be the same as the location name that is specified on the CALL statement as the following statements demonstrate. This is the only condition under which the names do not have to be specified the same way and a three-part name on the CALL statement can be used with a two-part name on the ASSOCIATE LOCATORS statement.

```
EXEC SQL CONNECT TO SITE2;
EXEC SQL CALL SITE2.MYSCHEMA.P1;
EXEC SQL ASSOCIATE LOCATORS (:LOC1, :LOC2)
         WITH PROCEDURE :HV1;
```

BEGIN DECLARE SECTION

BEGIN DECLARE SECTION

Syntax

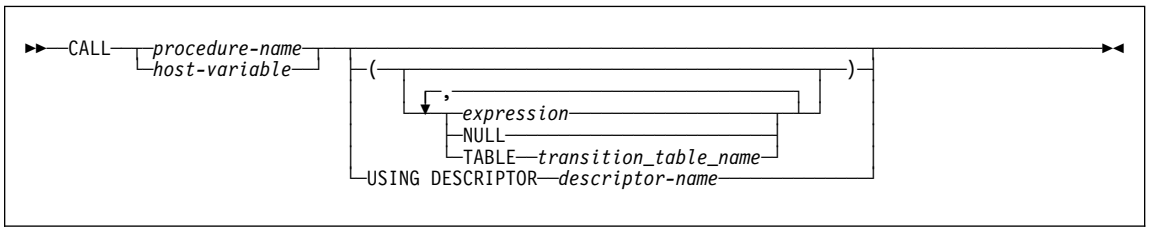


Example

```
EXEC SQL BEGIN DECLARE SECTION;

 (host variable declarations)

EXEC SQL END DECLARE SECTION;
```

CALL

Syntax



Example

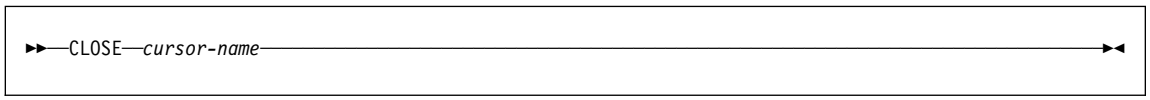
A PL/I application has been precompiled on DB2 ALPHA and a package was created at DB2 BETA with the BIND subcommand. A CREATE PROCEDURE statement was issued at BETA to define the procedure SUMARIZE, which allows nulls and has two parameters. The first parameter is defined as IN and the second parameter is defined as OUT. Some of the statements that the application that runs at DB2 ALPHA might use to call stored procedure SUMARIZE include:

```
EXEC SQL CONNECT TO BETA;
V1 = 528671;
IV = -1;
EXEC SQL CALL SUMARIZE(:V1,:V2 INDICATOR :IV);
```

CLOSE

CLOSE

Syntax



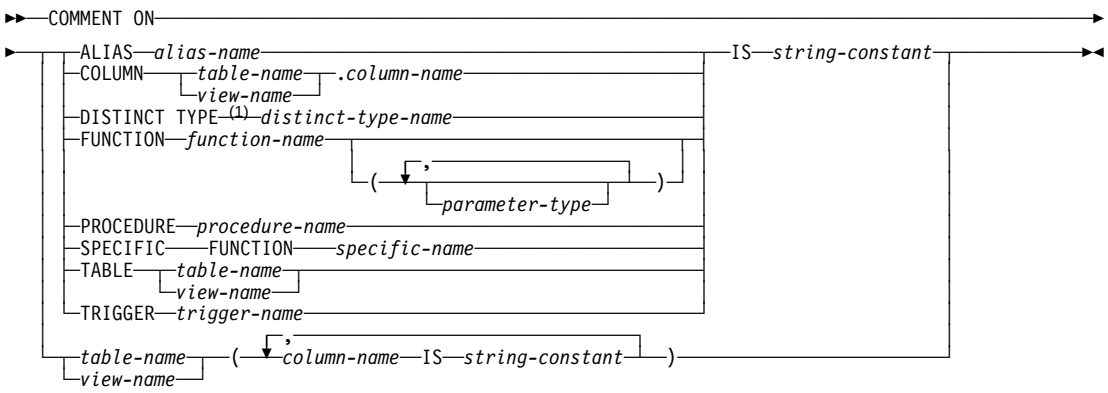
Example

A cursor is used to fetch one row at a time into the application program variables DNUM, DNAME, and MNUM. Finally, the cursor is closed. If the cursor is reopened, it is again located at the beginning of the rows to be fetched.

```
EXEC SQL DECLARE C1 CURSOR FOR
   SELECT DEPTNO, DEPTNAME, MGRNO
   FROM DSN8610.DEPT
   WHERE ADMRDEPT = 'A00'
   END-EXEC.

EXEC SQL OPEN C1 END-EXEC.

EXEC SQL FETCH C1 INTO :DNUM, :DNAME, :MNUM END-EXEC.

IF SQLCODE = 100
   PERFORM DATA-NOT-FOUND
ELSE
   PERFORM GET-REST-OF-DEPT
   UNTIL SQLCODE IS NOT EQUAL TO ZERO.

EXEC SQL CLOSE C1 END-EXEC.

GET-REST-OF-DEPT.
   EXEC SQL FETCH C1 INTO :DNUM, :DNAME, :MNUM END-EXEC.
```

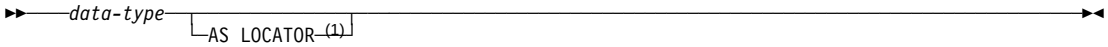
COMMENT ON

Syntax



Note:
1 DATA can be used as a synonym for DISTINCT.

parameter-type

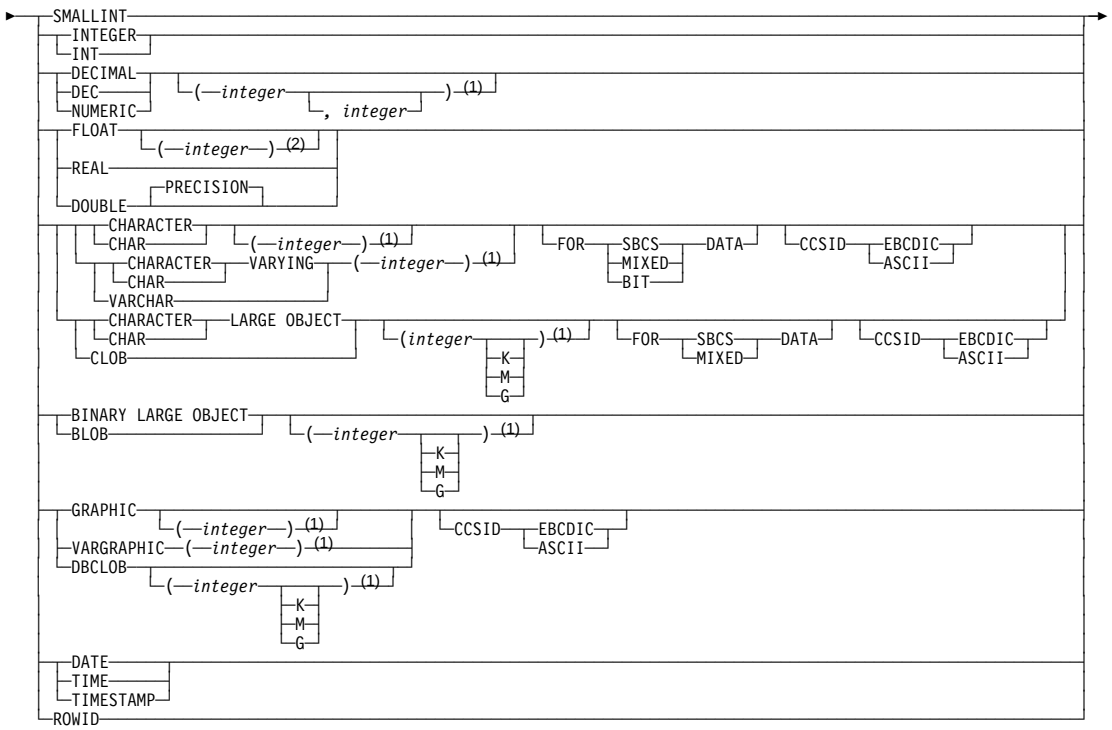


Note:
1 AS LOCATOR can be specified only for a LOB data type or a distinct type that is based on a LOB data type.

data-type

COMMENT ON

built-in-data-type



Notes:

1. The values that are specified for length, precision, or scale attributes must match the values that were specified when the function was created. Coding specific values is optional. Empty parentheses, (), can be used instead to indicate that DB2 ignores the attributes when determining whether data types match.
2. The value that is specified does not have to match the value that was specified when the function was created because matching is based on data type (REAL or DOUBLE). 1<=*integer*<= 21 indicates REAL and 22<=*integer*<=53 indicates DOUBLE. Coding a specific value is optional. Empty parentheses cannot be used.

Examples

Example 1: Enter a comment on table DSN8610.EMP.

```
COMMENT ON TABLE DSN8610.EMP
  IS 'REFLECTS 1ST QTR 81 REORG';
```

Example 2: Enter a comment on view DSN8610.VDEPT.

```
COMMENT ON TABLE DSN8610.VDEPT
  IS 'VIEW OF TABLE DSN8610.DEPT';
```

Example 3: Enter a comment on the DEPTNO column of table DSN8610.DEPT.

```
COMMENT ON COLUMN DSN8610.DEPT.DEPTNO
  IS 'DEPARTMENT ID - UNIQUE';
```

Example 4: Enter comments on the two columns in table DSN8610.DEPT.

```
COMMENT ON DSN8610.DEPT
  (MGRNO IS 'EMPLOYEE NUMBER OF DEPARTMENT MANAGER',
   ADMRDEPT IS 'DEPARTMENT NUMBER OF ADMINISTERING DEPARTMENT');
```

Example 5: Assume that you are SMITH and that you created the distinct type DOCUMENT in schema SMITH. Enter comments on DOCUMENT.

```
COMMENT ON DISTINCT TYPE DOCUMENT
  IS 'CONTAINS DATE, TABLE OF CONTENTS, BODY, INDEX, and GLOSSARY';
```

Example 6: Assume that you are SMITH and you know that ATOMIC_WEIGHT is the only function with that name in schema CHEM. Enter comments on ATOMIC_WEIGHT.

```
COMMENT ON FUNCTION CHEM.ATOMIC_WEIGHT
  IS 'TAKES ATOMIC NUMBER AND GIVES ATOMIC WEIGHT';
```

Example 7: Assume that you are SMITH and that you created the function CENTER in schema SMITH. Enter comments on CENTER, using the signature to uniquely identify the function instance.

```
COMMENT ON FUNCTION CENTER (INTEGER, FLOAT)
  IS 'USES THE CHEBYCHEV METHOD';
```

Example 8: Assume that you are SMITH and that you created another function named CENTER in schema JOHNSON. You gave the function the specific name FOCUS97. Enter comments on CENTER, using the specific name to identify the function instance.

```
COMMENT ON SPECIFIC FUNCTION JOHNSON.FOCUS97
  IS 'USES THE SQUARING TECHNIQUE';
```

Example 9: Assume that you are SMITH and that stored procedure OSMOSIS is in schema BIOLOGY. Enter comments on OSMOSIS.

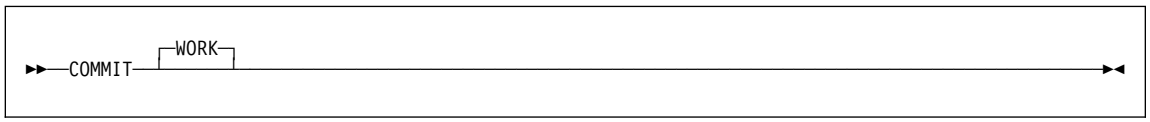
```
COMMENT ON PROCEDURE BIOLOGY.OSMOSIS
  IS 'CALCULATIONS THAT MODEL OSMOSIS';
```

Example 11: Assume that you are SMITH and that trigger BONUS is in your schema. Enter comments on BONUS.

```
COMMENT ON TRIGGER BONUS
  IS 'LIMITS BONUSES TO 10% OF SALARY;
```

COMMIT

Syntax



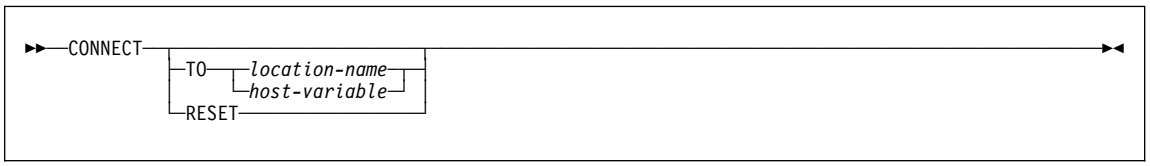
Example

Commit all DB2 database changes made since the unit of recovery was started.

```
COMMIT WORK;
```

CONNECT (Type 1)

Syntax



Examples

Example 1: Connect the application to a DBMS whose location identifier is in the character-string variable LOCNAME.

```
EXEC SQL CONNECT TO :LOCNAME;
```

Example 2: Use the CONNECT statement to obtain information about the current server. The information is then stored in the SQLERRP field of the SQLCA.

```
EXEC SQL CONNECT;
```

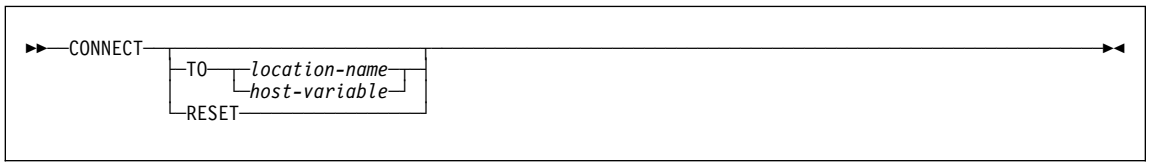
Example 3: An application has connected to a DB2 server that is not the local DBMS. During the connection, the application has opened a cursor and fetched rows from the cursor's result table. To connect to the local DBMS, the application executes the following statements:

```
EXEC SQL COMMIT WORK;
EXEC SQL CONNECT RESET;
```

The commit operation is required because the OPEN statement for the cursor has caused the application to enter the unconnectable and connected state. If the cursor had been declared with WITH HOLD and had not been closed with a CLOSE statement, it would still be open after the execution of the COMMIT, but would be closed with the execution of the CONNECT.

CONNECT (Type 2)

Syntax



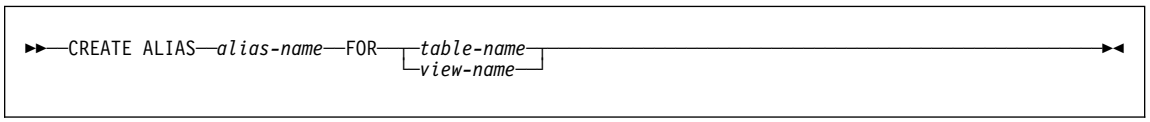
Example

Execute SQL statements at TOROLAB1 and TOROLAB2. The first CONNECT statement creates the TOROLAB1 connection. The second CONNECT statement creates the TOROLAB2 connection and places the TOROLAB1 connection in the dormant state.

```
EXEC SQL CONNECT TO TOROLAB1;

  (execute statements referencing objects at TOROLAB1)

EXEC SQL CONNECT TO TOROLAB2;

  (execute statements referencing objects at TOROLAB2)
```

CREATE ALIAS

Syntax



Example

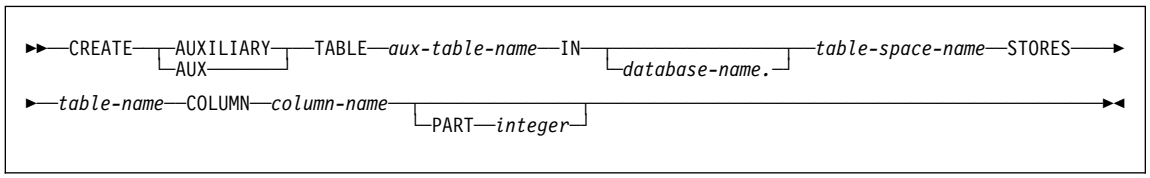
Create an alias for a catalog table at a DB2 with location name DB2USCALABOA5281.

```
CREATE ALIAS LATABLES FOR DB2USCALABOA5281.SYSIBM.SYSTABLES;
```

CREATE AUXILIARY TABLE

CREATE AUXILIARY TABLE

Syntax



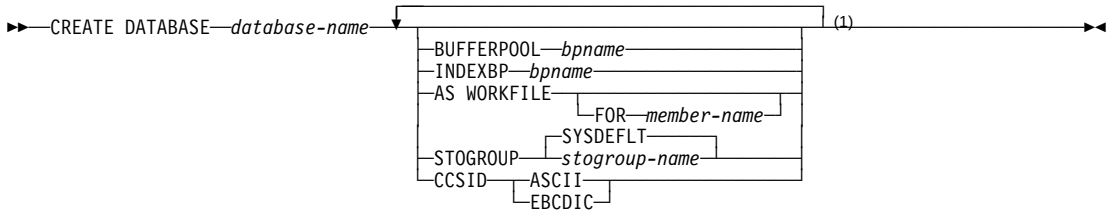
Example

Assume that a column named EMP_PHOTO with a data type of BLOB(110K) has been added to sample employee table DSN8610.EMP for each employee's photo. Create auxiliary table EMP_PHOTO_ATAB to store the BLOB data for the BLOB column in LOB table space DSN8D61A.PHOTOLTS.

```
CREATE AUX TABLE EMP_PHOTO_ATAB
   IN DSN8D61A.PHOTOLTS
   STORES DSN8610.EMP
   COLUMN EMP_PHOTO;
```

CREATE DATABASE

Syntax



Note:
1 The same clause must not be specified more than once.

Examples

Example 1: Create database DSN8D61P. Specify DSN8G610 as the default storage group to be used for the table spaces and indexes in the database. Specify 8KB buffer pool BP8K1 as the default buffer pool to be used for table spaces in the database, and BP2 as the default buffer pool to be used for indexes in the database.

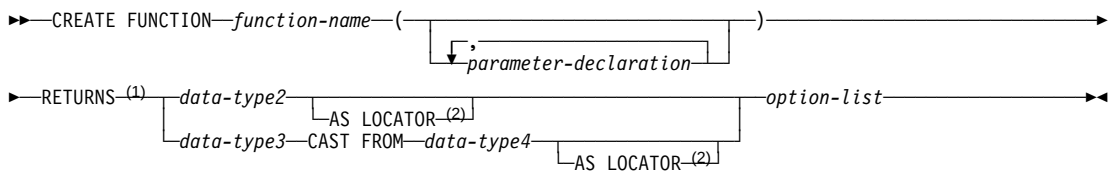
```
CREATE DATABASE DSN8D61P
  STOGROUP DSN8G610
  BUFFERPOOL BP8K1
  INDEXBP BP2;
```

Example 2: Create database DSN8TEMP. Use the defaults for the default storage group and default buffer pool names. Specify ASCII as the default encoding scheme for data stored in the database.

```
CREATE DATABASE DSN8TEMP
  CCSID ASCII;
```

CREATE FUNCTION (external scalar)

Syntax

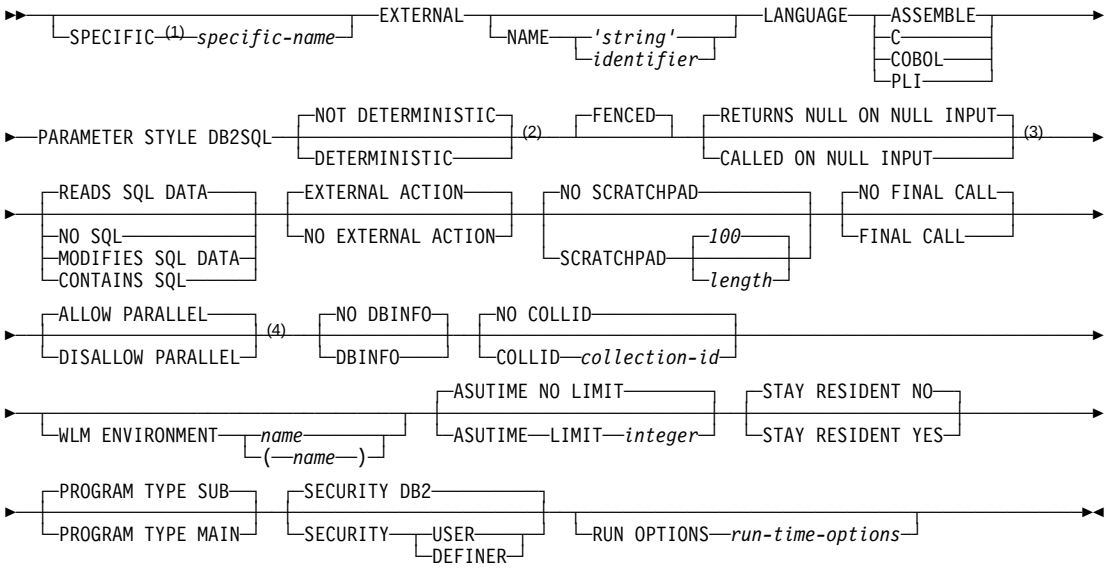


Notes:

1 This clause and the clauses that follow in the *option-list* can be specified in any order.

2 AS LOCATOR can be specified only for a LOB data type or a distinct type based on a LOB data type.

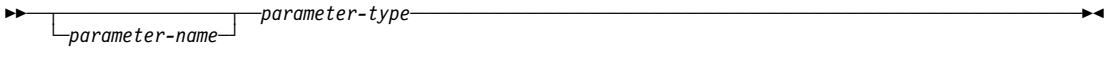
option-list:



Notes:

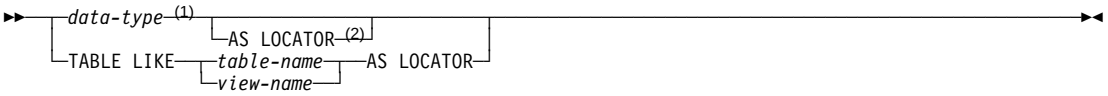
1. This clause and the other clauses in the *option-list* can be specified in any order.
2. Synonyms for this clause include VARIANT for NOT DETERMINISTIC, and NOT VARIANT for DETERMINISTIC.
3. Synonyms for this clause include NOT NULL CALL for RETURNS NULL ON NULL INPUT, and NULL CALL for CALLED ON NULL INPUT.
4. If NOT DETERMINISTIC, EXTERNAL ACTION, SCRATCHPAD, or FINAL CALL is specified, DISALLOW PARALLEL is the default.

parameter-declaration:



CREATE FUNCTION (external scalar)

parameter-type:



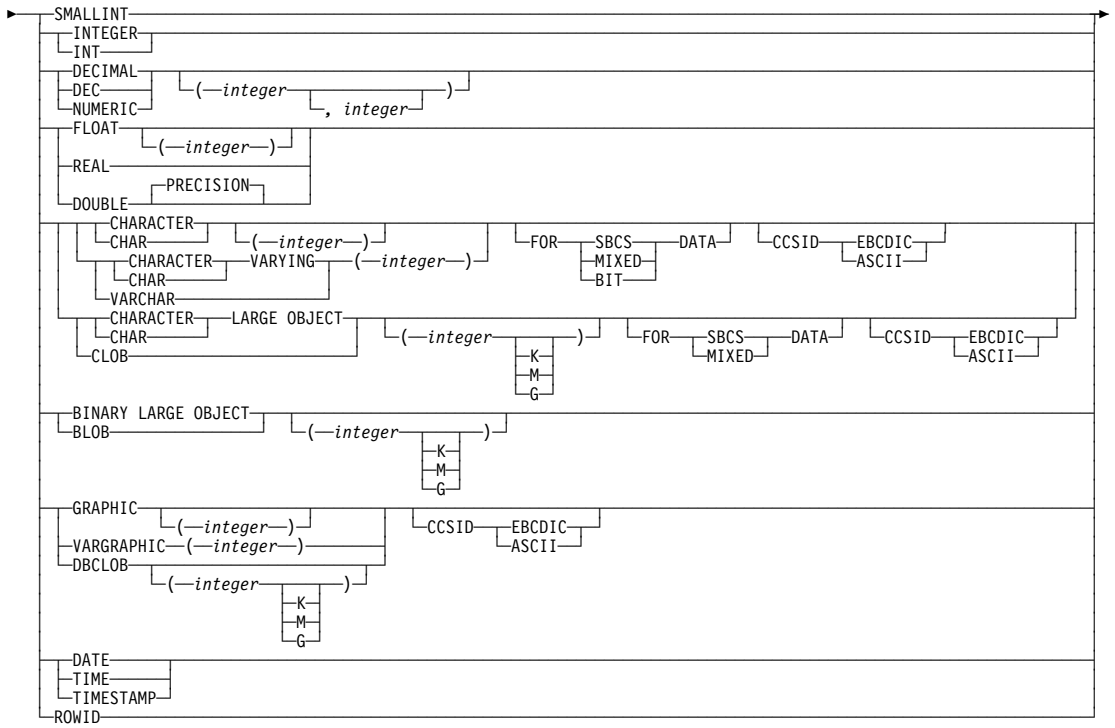
Notes:

1 A LOB data type or distinct type based on a LOB data type must be no greater than 1M unless a locator is passed.

2 AS LOCATOR can be specified only for a LOB data type or a distinct type based on a LOB data type.

data-type:

built-in-data-type:



Examples

Example 1: Assume that you want to write an external function program in C that implements the following logic:

```
output = 2 * input - 4
```

The function should return a null value if and only if one of the input arguments is null. The simplest way to avoid a function call and get a null result when an input value is null is to specify RETURNS NULL ON NULL INPUT on the CREATE FUNCTION statement or allow it to be the default. Write the statement needed to register the function, using the specific name MINENULL1.

```
CREATE FUNCTION NTEST1 (SMALLINT)
   RETURNS SMALLINT
   EXTERNAL NAME 'NTESTMOD'
   SPECIFIC MINENULL1
   LANGUAGE C
   DETERMINISTIC
   NO SQL
   FENCED
   PARAMETER STYLE DB2SQL
   RETURNS NULL ON NULL INPUT
   NO EXTERNAL ACTION;
```

Example 2: Assume that user Smith wants to register an external function named CENTER in schema SMITH. The function program will be written in C and will be reentrant. Write the statement that Smith needs to register the function, letting DB2 generate a specific name for the function.

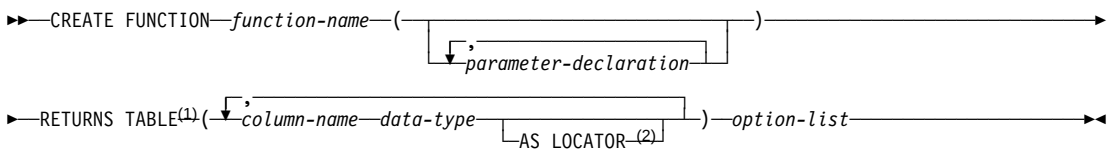
```
CREATE FUNCTION CENTER (INTEGER, FLOAT)
   RETURNS FLOAT
   EXTERNAL NAME 'MIDDLE'
   LANGUAGE C
   DETERMINISTIC
   NO SQL
   FENCED
   PARAMETER STYLE DB2SQL
   NO EXTERNAL ACTION
   STAY RESIDENT YES;
```

Example 3: Assume that user McBride (who has administrative authority) wants to register an external function named CENTER in the SMITH schema. McBride plans to give the function specific name FOCUS98. The function program uses a scratchpad to perform some one-time only initialization and save the results. The function program returns a value with a FLOAT data type. Write the statement McBride needs to register the function and ensure that when the function is invoked, it returns a value with a data type of DECIMAL(8,4).

```
CREATE FUNCTION SMITH.CENTER (FLOAT, FLOAT, FLOAT)
   RETURNS DECIMAL(8,4) CAST FROM FLOAT
   EXTERNAL NAME 'CMOD'
   SPECIFIC FOCUS98
   LANGUAGE C
   DETERMINISTIC
   NO SQL
   FENCED
   PARAMETER STYLE DB2SQL
   NO EXTERNAL ACTION
   SCRATCHPAD
   NO FINAL CALL;
```

CREATE FUNCTION (external table)

Syntax

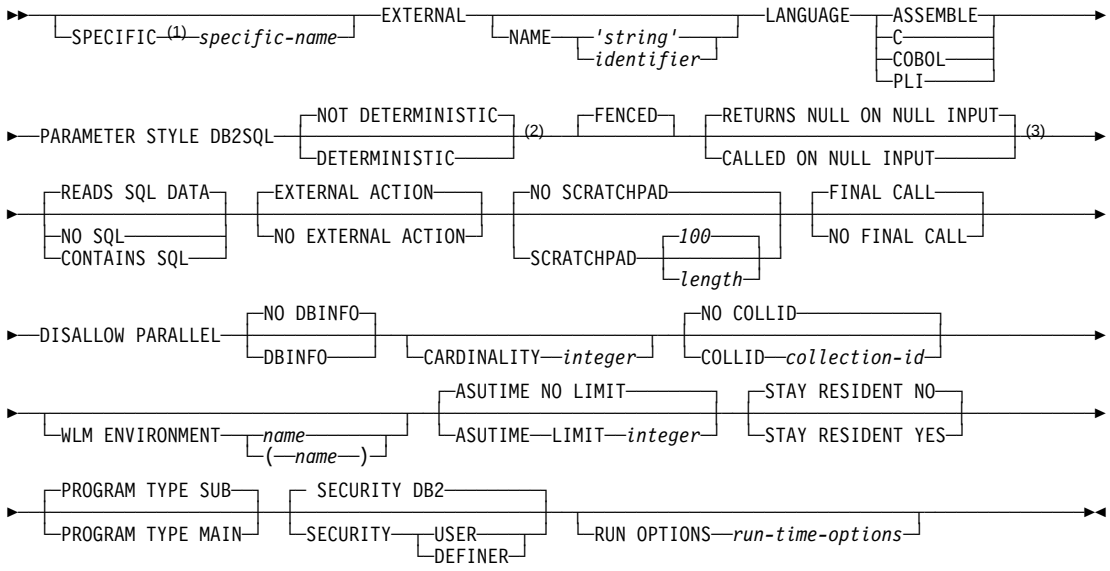


Notes:

1 This clause and the clauses that follow in the *option-list* can be specified in any order.

2 AS LOCATOR can be specified only for a LOB data type or a distinct type based on a LOB data type.

option-list:



Notes:

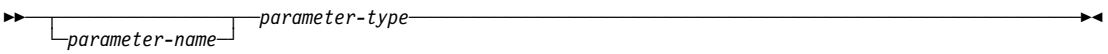
1 This clause and the other clauses in the *option-list* can be specified in any order.

2 Synonyms include VARIANT for NOT DETERMINISTIC, and NOT VARIANT for DETERMINISTIC.

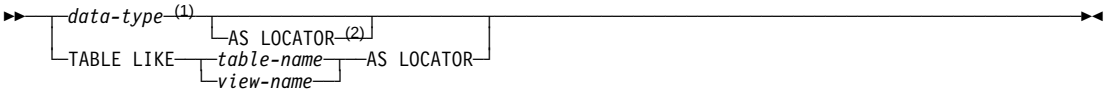
3 Synonyms include NOT NULL CALL for RETURNS NULL ON NULL INPUT, and NULL CALL for CALLED ON NULL INPUT.

CREATE FUNCTION (external table)

parameter-declaration:



parameter-type:

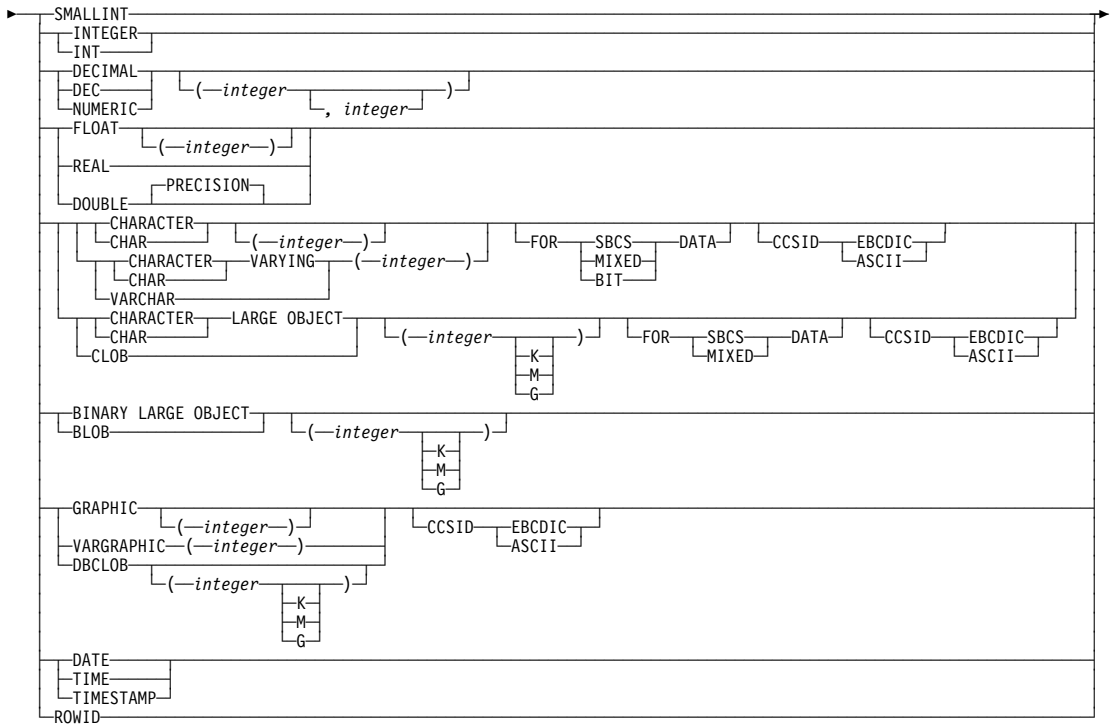


Notes:

1. A LOB data type or distinct type based on a LOB data type must be no greater than 1M unless a locator is passed.
2. AS LOCATOR can be specified only for a LOB data type or a distinct type based on a LOB data type.

data-type:

built-in-data-type:



Examples

The following registers a table function written to return a row consisting of a single document identifier column for each known document in a text management system. The first parameter matches a given subject area and the second parameter contains a given string.

Within the context of a single session, the table function will always return the same table; therefore, it is defined as DETERMINISTIC. In addition, the DISALLOW PARALLEL keyword is added because table functions cannot operate in parallel.

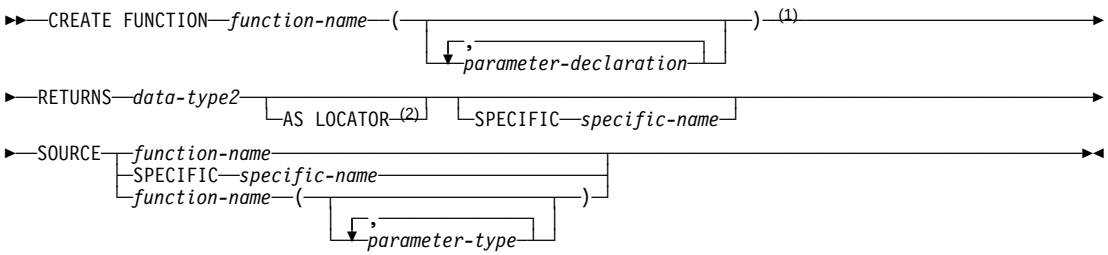
Although the size of the output for DOCMATCH is highly variable, CARDINALITY 20 is a representative value, and is specified to help DB2.

CREATE FUNCTION (external table)

```
CREATE FUNCTION DOCMATCH (VARCHAR(30), VARCHAR(255))
                  RETURNS TABLE (DOC_ID CHAR(16))
  EXTERNAL NAME ABC
  LANGUAGE C
  PARAMETER STYLE DB2SQL
  NO SQL
  DETERMINISTIC
  NO EXTERNAL ACTION
  FENCED
  SCRATCHPAD
  FINAL CALL
  DISALLOW PARALLEL
  CARDINALITY 20;
```

CREATE FUNCTION (sourced)

Syntax

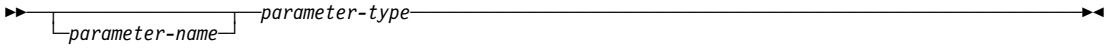


Notes:

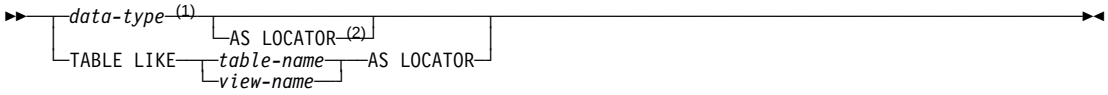
1 RETURNS, SPECIFIC, and SOURCE can be specified in any order.

2 AS LOCATOR can be specified only for a LOB data type or a distinct type based on a LOB data type.

parameter-declaration:



parameter-type:



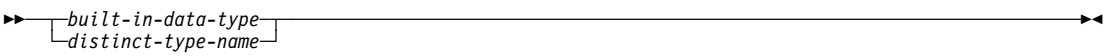
Notes:

1 A LOB data type or distinct type based on a LOB data type must be no greater than 1M unless a locator is passed.

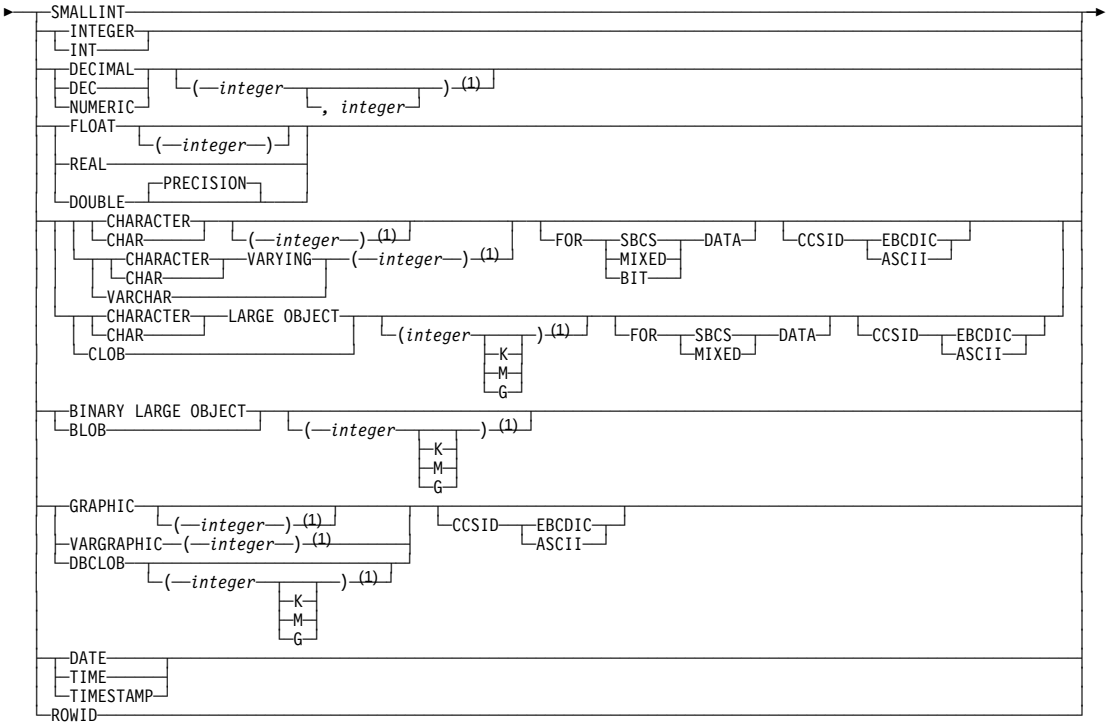
2 AS LOCATOR can be specified only for a LOB data type or a distinct type based on a LOB data type.

CREATE FUNCTION (sourced)

data-type:



built-in-data-type:



Notes:

1 Coding specific values for the length, precision, or scale attributes of a data type for a parameter in the SOURCE clause is optional. Empty parentheses, (), can be used instead to indicate that DB2 ignores the attributes when determining whether data types match. However, if the length, precision, or scale attributes is specified, the value must exactly match the value that was specified when the source function was created.

2 Coding a specific value is optional. If a value is specified, it does not have to match the value that was specified when the source function was created because matching is based on data type (REAL or DOUBLE). 1<=*integer*<= 21 indicates REAL and 22<=*integer*<=53 indicates DOUBLE. Empty parentheses cannot be used.

Examples

Example 1: Assume that you created a distinct type HATSIZE, which you based on the built-in data type INTEGER. You want to have an AVG function to compute the average hat size of different departments. Create a sourced function that is based on built-in function AVG.

```
CREATE FUNCTION AVE (HATSIZE) RETURNS HATSIZE
   SOURCE SYSIBM.AVG (INTEGER);
```

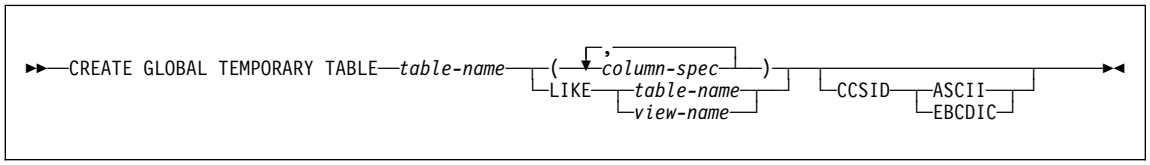
When you created distinct type HATSIZE, two cast functions were generated, which allow HATSIZE to be cast to INTEGER for the argument and INTEGER to be cast to HATSIZE for the result of the function.

Example 2: After Smith registered the external scalar function CENTER in his schema, you decide that you want to use this function, but you want it to accept two INTEGER arguments instead of one INTEGER argument and one FLOAT argument. Create a sourced function that is based on CENTER.

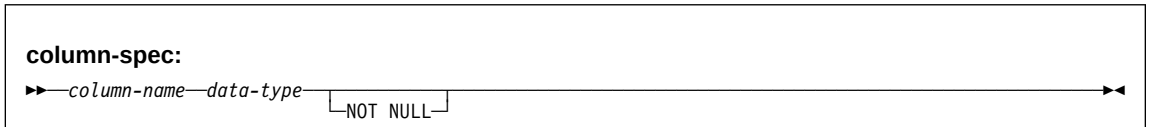
```
CREATE FUNCTION MYCENTER (INTEGER, INTEGER)
   RETURNS FLOAT
   SOURCE SMITH.CENTER (INTEGER, FLOAT);
```

CREATE GLOBAL TEMPORARY TABLE

Syntax

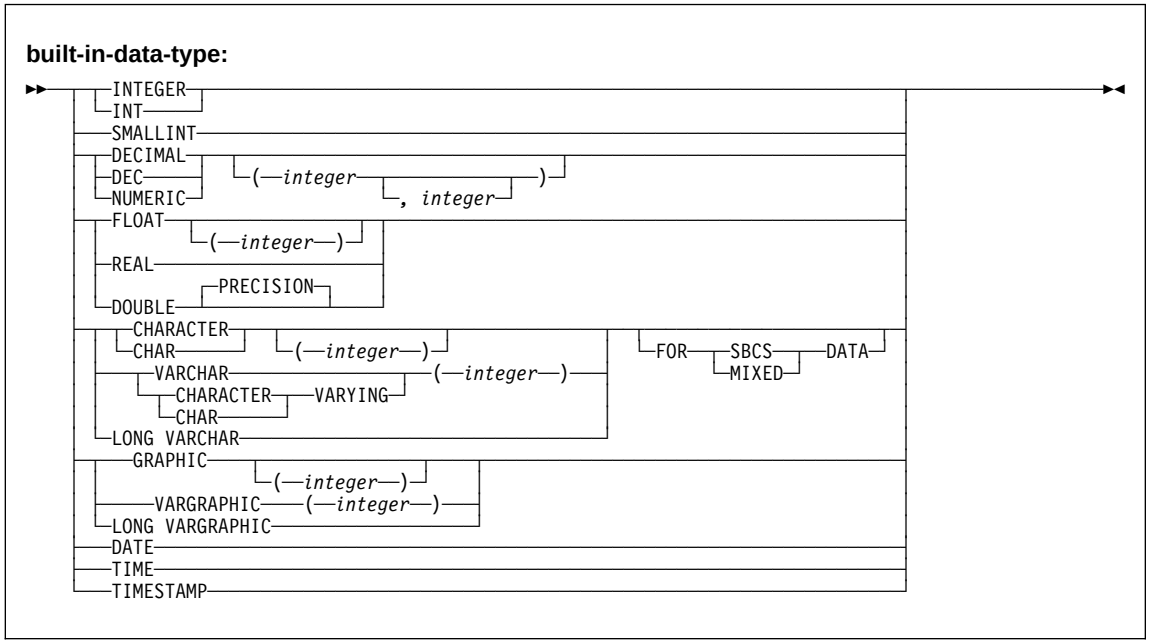


column-spec:



data-type:

built-in-data-type:



Examples

Example 1: Create a temporary table, CURRENTMAP. Name two columns, CODE and MEANING, both of which cannot contain nulls. CODE contains numeric data and MEANING has character data. Assuming a value of NO for the field MIXED DATA on installation panel DSNTIPF, column MEANING has a subtype of SBCS:

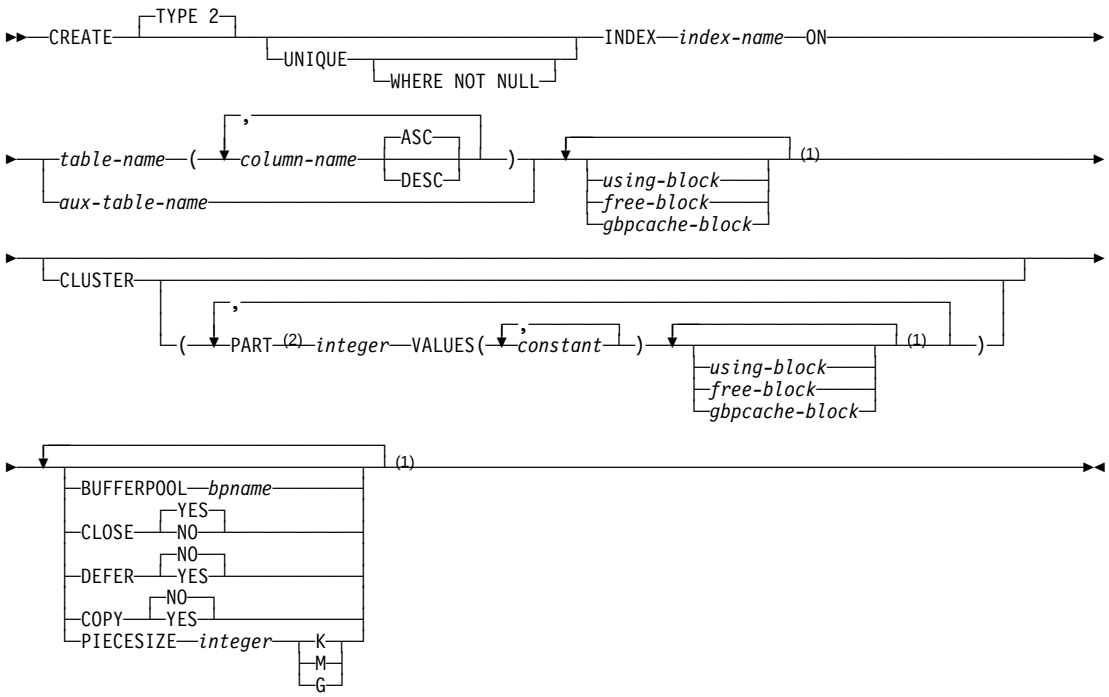
```
CREATE GLOBAL TEMPORARY TABLE CURRENTMAP
     (CODE INTEGER NOT NULL, MEANING VARCHAR(254) NOT NULL);
```

Example 2: Create a temporary table, EMP:

```
CREATE GLOBAL TEMPORARY TABLE EMP
     (TMPDEPTNO   CHAR(3)     NOT NULL,
      TMPDEPTNAME VARCHAR(36) NOT NULL,
      TMPMGRNO    CHAR(6)             ,
      TMPLOCATION CHAR(16)            );
```

CREATE INDEX

Syntax

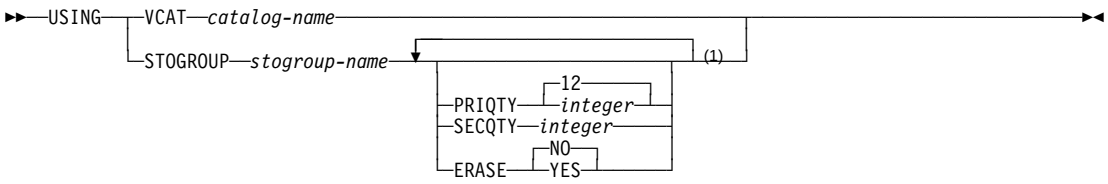


Notes:

1 The same clause must not be specified more than once.

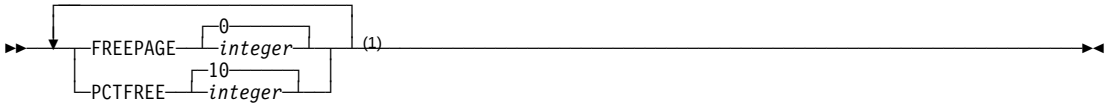
2 The presence of one or more PART clauses indicates that the index is a partitioning index; otherwise, it is a nonpartitioning index.

using-block:



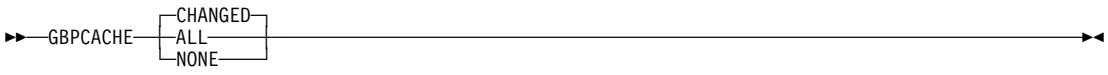
Note:
1 The same clause must not be specified more than once.

free-block:



Note:
1 The same clause must not be specified more than once.

gbpcache-block:



Examples

Example 1: Create a unique index, named DSN8610.XDEPT1, on table DSN8610.DEPT. Index entries are to be in ascending order by the single column DEPTNO. DB2 is to define the data sets for the index, using storage group DSN8G610. Each data set (piece) should hold 1 megabyte of data at most. Use 512 kilobytes as the primary space allocation for each data set and 64 kilobytes as the secondary space allocation. These specifications enable each data set to be extended up to 8 times before a new data set is used—512KB + (8*64KB)= 1024KB.

The data sets can be closed when no one is using the index and do not need to be erased if the index is dropped.

CREATE INDEX

```
CREATE UNIQUE INDEX DSN8610.XDEPT1
  ON DSN8610.DEPT
    (DEPTNO ASC)
  USING STOGROUP DSN8G610
    PRIQTY 512
    SECQTY 64
    ERASE NO
  BUFFERPOOL BP1
  CLOSE YES
  PIECESIZE 1 M;
```

Example 2: Create a cluster index, named XEMP2, on table EMP in database DSN8610. Put the entries in ascending order by column EMPNO. Let DB2 define the data sets for each partition using storage group DSN8G610. Make the primary space allocation be 36 kilobytes, and allow DB2 to use the default value for SECQTY, which for this example is 12 kilobytes (3 times 4KB). If the index is dropped, the data sets need not be erased.

There are to be 4 partitions, with index entries divided among them as follows:

Partition 1: entries up to H99
Partition 2: entries above H99 up to P99
Partition 3: entries above P99 up to Z99
Partition 4: entries above Z99

Associate the index with buffer pool BP1 and allow the data sets to be closed when no one is using the index. Enable the use of the COPY utility for full image or concurrent copies and the RECOVER utility.

```
CREATE INDEX DSN8610.XEMP2
  ON DSN8610.EMP
    (EMPNO ASC)
  USING STOGROUP DSN8G610
    PRIQTY 36
    ERASE NO
   CLUSTER
    (PART 1 VALUES('H99'),
     PART 2 VALUES('P99'),
     PART 3 VALUES('Z99'),
     PART 4 VALUES('999'))
  BUFFERPOOL BP1
  CLOSE YES
  COPY YES;
```

Example 3: Create a nonpartitioning index, named DSN8610.XDEPT1, on table DSN8610.DEPT. Put the entries in ascending order by column DEPTNO. Assume that the data sets are managed by the user with catalog name DSNCAT and each data set (piece) is to hold 1 gigabyte of data at most before the next data set is used.

```
CREATE UNIQUE INDEX DSN8610.XDEPT1
  ON DSN8610.DEPT
    (DEPTNO ASC)
  USING VCAT DSNCAT
  PIECESIZE 1048576 K;
```

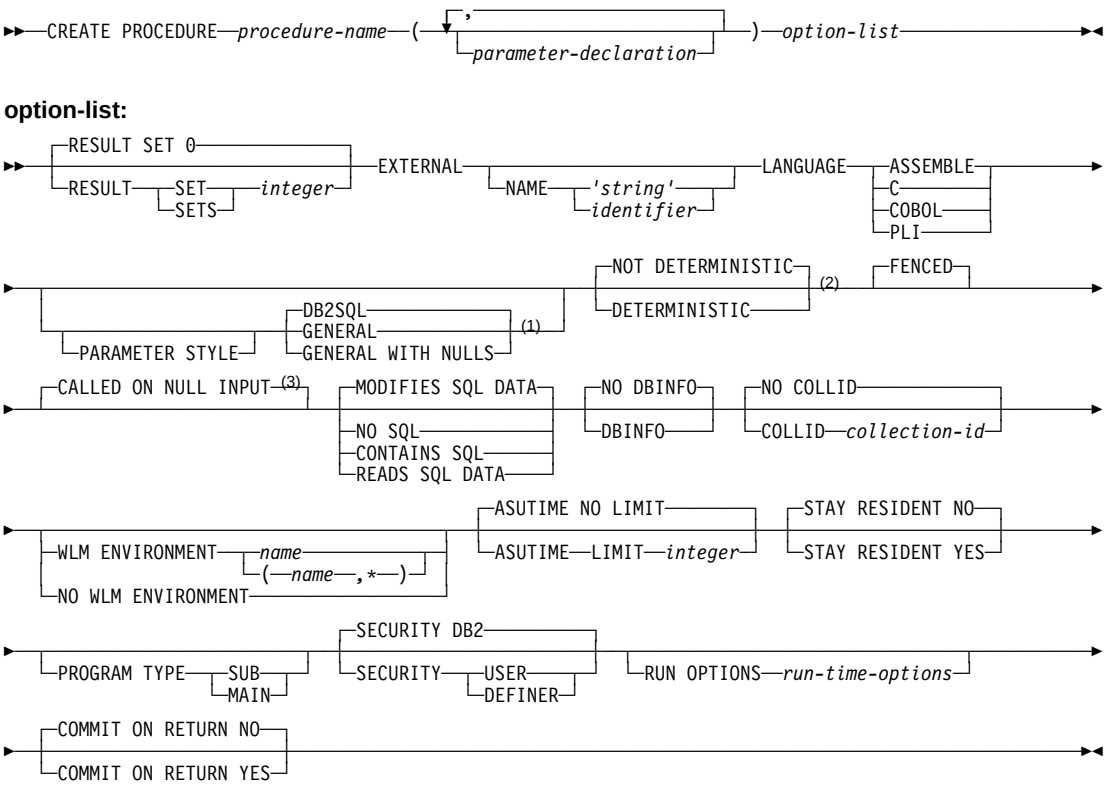
Example 4: Assume that a column named EMP_PHOTO with a data type of BLOB(110K) was added to the sample employee table for each employee's photo and auxiliary table EMP_PHOTO_ATAB was created in LOB table space DSN8D61A.PHOTOLTS to store the BLOB data for the column. Create an index named XPHOTO on the auxiliary table. The data sets are to be user-managed with catalog name DSNCAT.

```
CREATE UNIQUE INDEX DSN8610.XPHOTO
   ON DSN8610.EMP_PHOTO_ATAB
   USING VCAT DSNCAT
   COPY YES;
```

In this example, no columns are specified for the key because auxiliary indexes have implicitly generated keys.

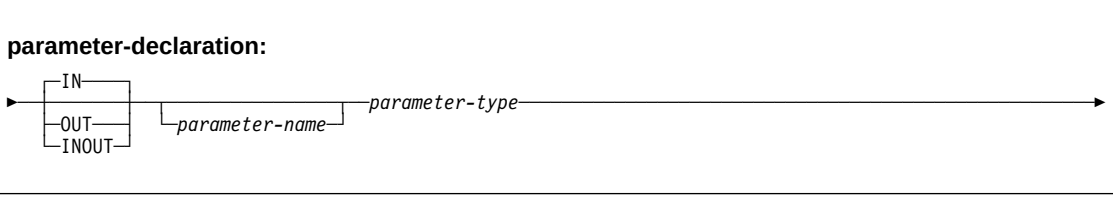
CREATE PROCEDURE

Syntax

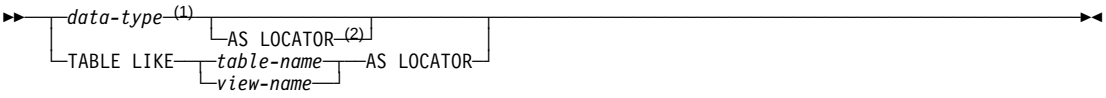


Notes:

1. Synonyms include STANDARD CALL for DB2SQL, SIMPLE CALL for GENERAL, and SIMPLE CALL WITH NULLS for GENERAL WITH NULLS.
2. Synonyms include VARIANT for NOT DETERMINISTIC, and NOT VARIANT for DETERMINISTIC.
3. NULL CALL is a synonym for CALLED ON NULL INPUT.



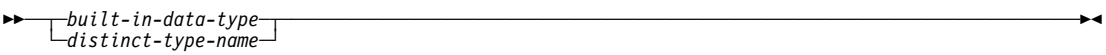
parameter-type:



Notes:

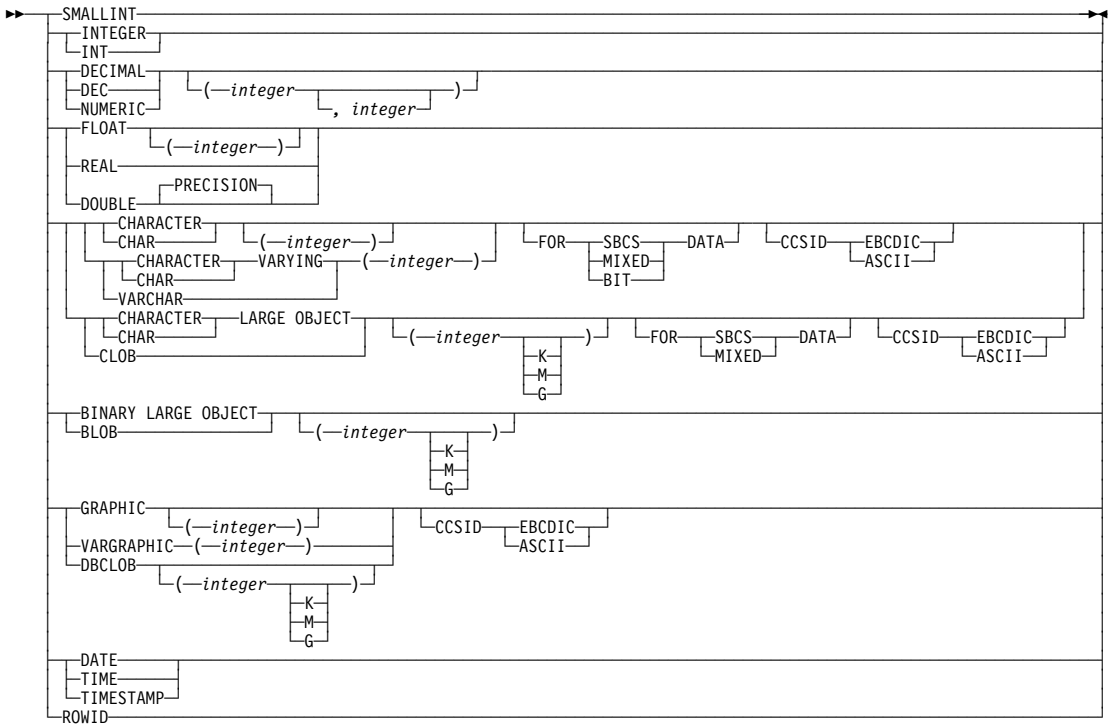
1. A LOB data type or distinct type based on a LOB data type must be no greater than 1M unless a locator is passed.
2. AS LOCATOR can be specified only for a LOB data type or a distinct type based on a LOB data type.

data-type:



CREATE PROCEDURE

built-in-data-type:



Examples

Example 1: Create the definition for a stored procedure that is written in COBOL. The procedure accepts an assembly part number and returns the number of parts that make up the assembly, the total part cost, and a result set. The result set lists the part numbers, quantity, and unit cost of each part. Assume that the input parameter cannot contain a null value and that the procedure is to run in a WLM environment called PARTSA.

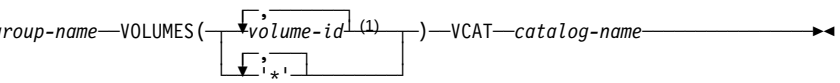
```
CREATE PROCEDURE SYSPROC.MYPROC(IN INT, OUT INT, OUT DECIMAL(7,2))
       LANGUAGE COBOL
       EXTERNAL NAME MYMODULE
       PARAMETER STYLE GENERAL
       WLM ENVIRONMENT PARTSA
       RESULT SETS 1;
```

Example 2: Create the definition for the stored procedure described in Example 1, except use the linkage convention that passes more information than the parameter specified on the CALL statement. Specify Language Environment® run-time options HEAP, BELOW, ALL31, and STACK.

```
CREATE PROCEDURE SYSPROC.MYPROC(IN INT, OUT INT, OUT DECIMAL(7,2))
       LANGUAGE COBOL
       EXTERNAL NAME MYMODULE
       PARAMETER STYLE DB2SQL
       WLM ENVIRONMENT PARTSA
       RESULT SETS 1
       RUN OPTIONS 'HEAP(,,ANY),BELOW(4K,,),ALL31(ON),STACK(,,ANY,)';
```

CREATE STOGROUP

Syntax



Note:
1 The same *volume-id* must not be specified more than once.

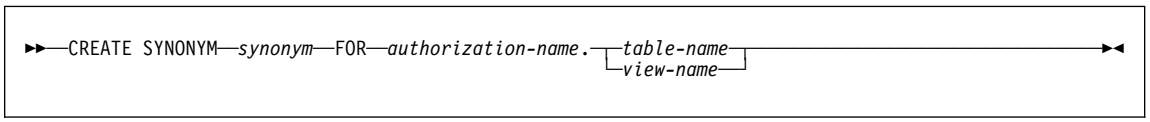
Example

Create storage group, DSN8G610, of volumes ABC005 and DEF008. DSNCAT is the integrated catalog facility catalog name.

```
CREATE STOGROUP DSN8G610
  VOLUMES (ABC005,DEF008)
  VCAT DSNCAT;
```

CREATE SYNONYM

Syntax



Example

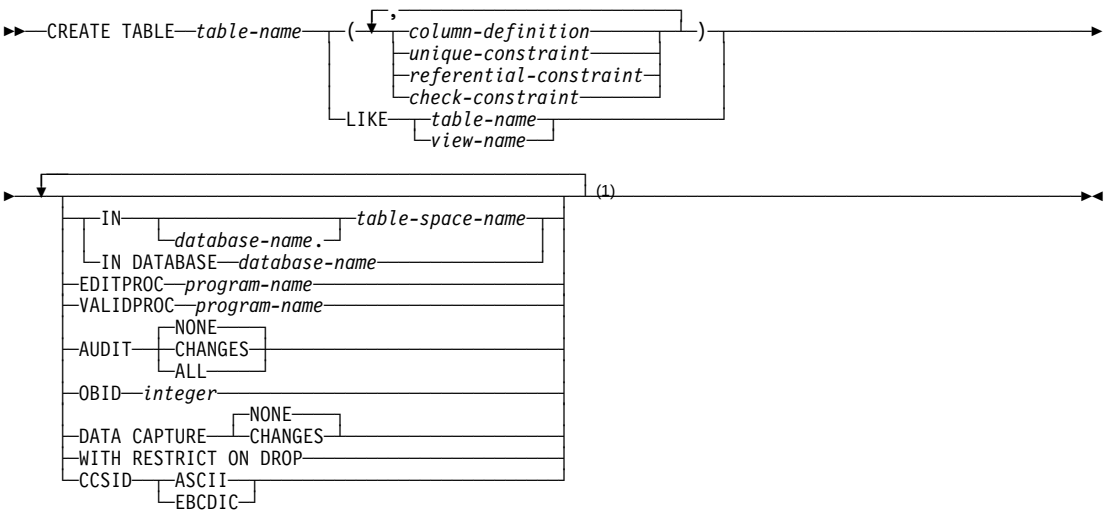
Define DEPT as a synonym for the table DSN8610.DEPT.

```
CREATE SYNONYM DEPT
  FOR DSN8610.DEPT;
```

This example does not work if the current SQL authorization ID is DSN8610.

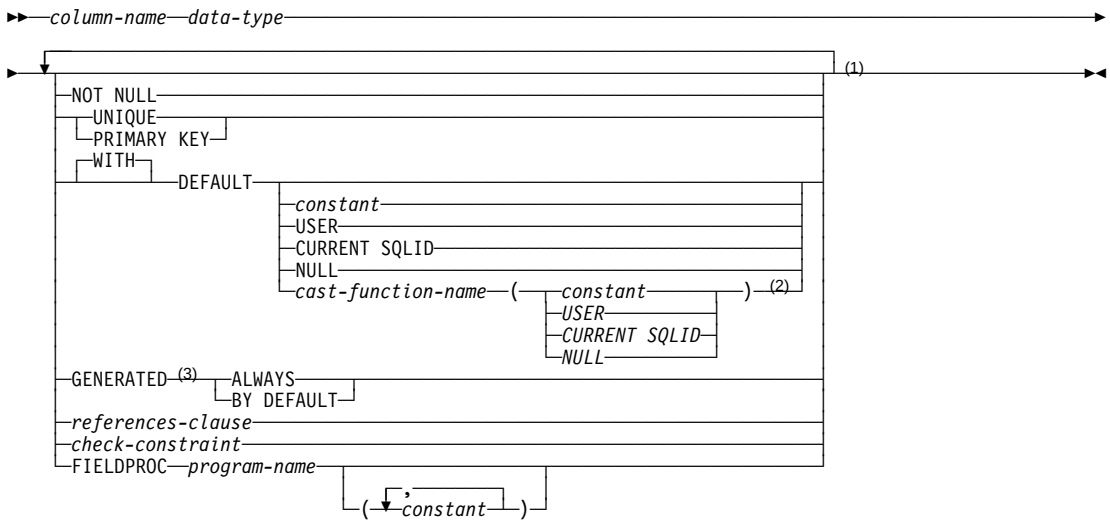
CREATE TABLE

Syntax



Note:
1 The same clause must not be specified more than once.

column-definition:



Notes:

1 The same clause must not be specified more than once.

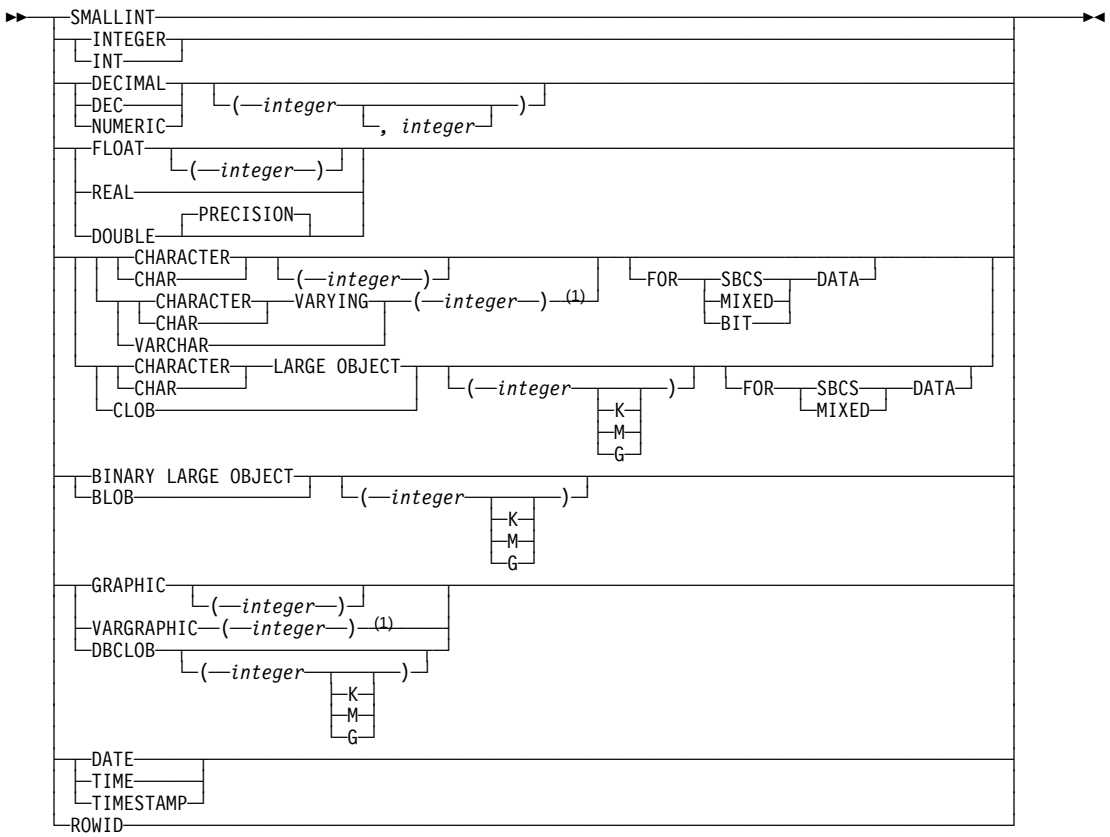
2 This form of the DEFAULT value can only be used with columns that are defined as a distinct type.

3 GENERATED can be specified only for a column that is defined as a ROWID data type (or a distinct type that is based on a ROWID data type).

data-type:

CREATE TABLE

built-in-data-type:



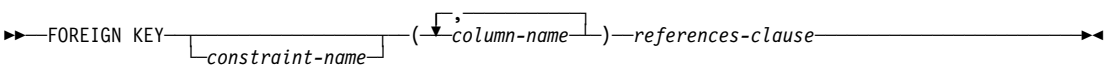
Note:

1 Although the syntax of LONG VARCHAR and LONG VARGRAHPIC is supported, the alternative syntax of VARCHAR(*integer*) and VARGRAPHIC(*integer*), is preferred. VARCHAR(*integer*) and VARGRAPHIC(*integer*) are recommended because after the CREATE TABLE statement is processed, DB2 considers a LONG VARCHAR column to be VARCHAR and a LONG VARGRAPHIC column to be VARGRAPHIC.

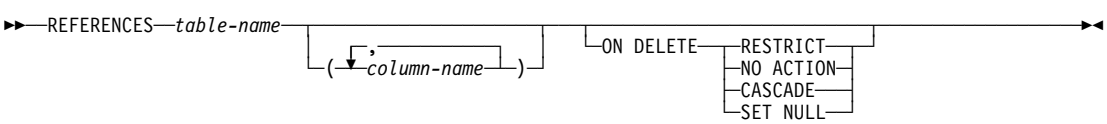
To determine the maximum length of a column defined as LONG VARCHAR or LONG VARGRAPHIC, see the CREATE TABLE statement in *DB2 SQL Reference*.

unique-constraint:

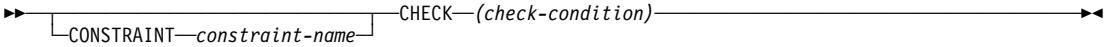
referential-constraint:



references-clause:



check-constraint:



Examples

Example 1: Create a table named DSN8610.DEPT in the table space DSN8S61D of the database DSN8D61A. Name the table's five columns DEPTNO, DEPTNAME, MGRNO, ADMRDEPT, and LOCATION, allowing only MGRNO to contain nulls, and designating DEPTNO as the only column in the table's primary key. All five columns hold character string data. Assuming a value of NO for the field MIXED DATA on installation panel DSNTIPF, all five columns have the subtype SBCS.

```
CREATE TABLE DSN8610.DEPT
  (DEPTNO   CHAR(3)     NOT NULL,
   DEPTNAME VARCHAR(36) NOT NULL,
   MGRNO    CHAR(6)             ,
   ADMRDEPT CHAR(3)     NOT NULL,
   LOCATION CHAR(16)            ,
   PRIMARY KEY(DEPTNO)          )
  IN DSN8D61A.DSN8S61D;
```

Example 2: Create a table named DSN8610.PROJ in an implicitly created table space of the database DSN8D61A. Assign the table a validation procedure named DSN8EAPR.

```
CREATE TABLE DSN8610.PROJ
  (PROJNO   CHAR(6)      NOT NULL,
   PROJNAME VARCHAR(24)  NOT NULL,
   DEPTNO   CHAR(3)      NOT NULL,
   RESPEMP  CHAR(6)      NOT NULL,
   PRSTAFF  DECIMAL(5,2)         ,
   PRSTDATE DATE                 ,
   PRENDATE DATE                 ,
   MAJPROJ  CHAR(6)      NOT NULL)
  IN DATABASE DSN8D61A
  VALIDPROC DSN8EAPR;
```

Example 3: Assume that table PROJECT has a non-primary unique key that consists of columns DEPTNO and RESPEMP (the department number and employee responsible for a project). Create a project activity table named ACTIVITY with a foreign key on that on that unique key.

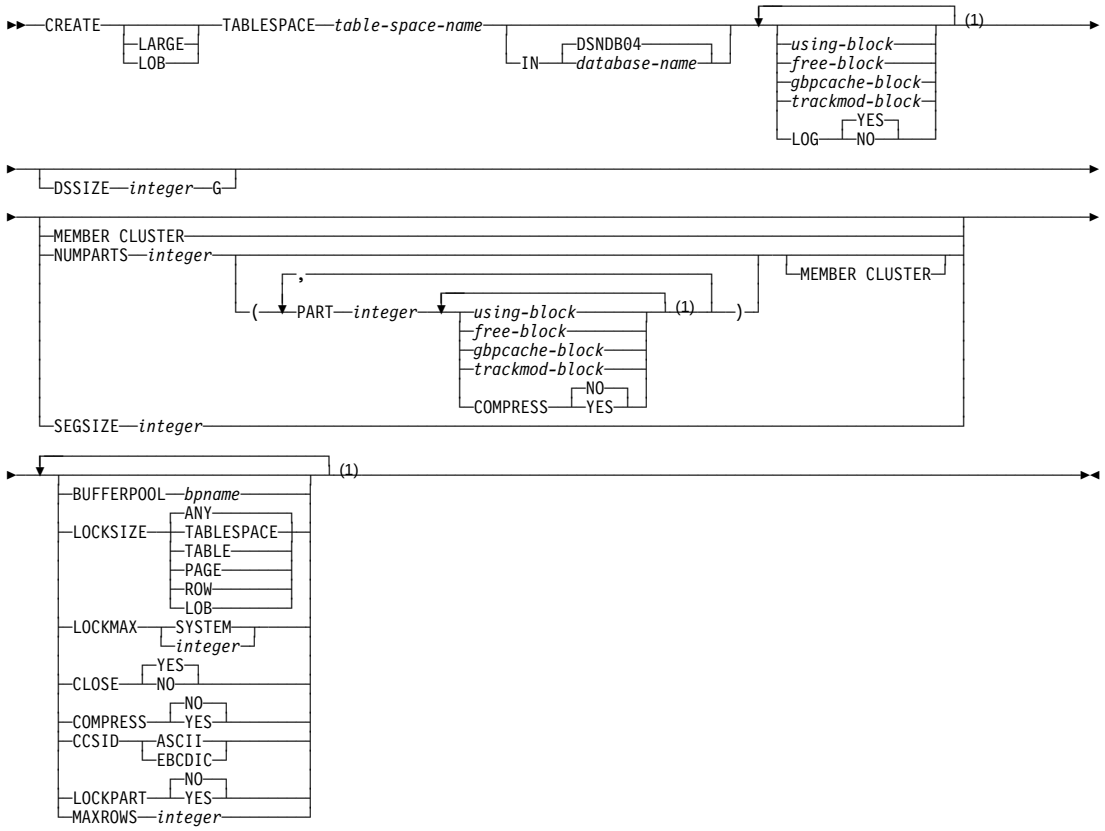
```
CREATE TABLE ACTIVITY
  (PROJNO   CHAR(6)      NOT NULL,
   ACTNO    SMALLINT     NOT NULL,
   ACTDEPT  CHAR(3)      NOT NULL,
   ACTOWNER CHAR(6)      NOT NULL,
   ACSTAFF  DECIMAL(5,2)         ,
   ACSTDATE DATE         NOT NULL,
   ACENDATE DATE                 ,
   FOREIGN KEY (ACTDEPT,ACTOWNER)
      REFERENCES PROJECT (DEPTNO,RESPEMP) ON DELETE RESTRICT)
  IN DSN8D61A.DSN8S61D;
```

Example 4: Create an employee photo and resume table EMP_PHOTO_RESUME that complements the sample employee table. The table contains a photo and resume for each employee. Put the table in table space DSN8D61A.DSN8S61E. Let DB2 always generate the values for the ROWID column.

```
CREATE TABLE DSN8610.EMP_PHOTO_RESUME
  (EMPNO      CHAR(6)     NOT NULL,
   EMP_ROWID  ROWID GENERATED ALWAYS,
   EMP_PHOTO  BLOBL(110K),
   RESUME     CLOB(5K)),
   PRIMARY KEY EMPNO
  IN DSN8D61A.DSN8S61E
  CCSID EBCDIC;
```

CREATE TABLESPACE

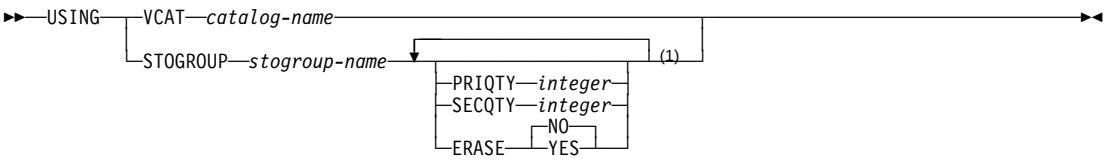
Syntax



Note:
1 The same clause must not be specified more than once.

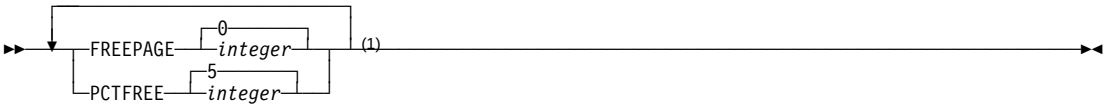
CREATE TABLESPACE

using-block:



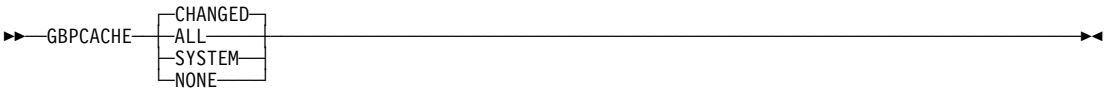
Note:
1 The same clause must not be specified more than once.

free-block:

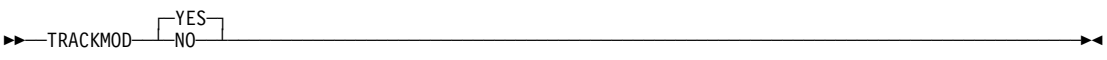


Note:
1 The same clause must not be specified more than once.

gbpcache-block:



trackmod-block:



Examples

Example 1: Create table space DSN8S61D in database DSN8D61A. Let DB2 define the data sets, using storage group DSN8G610. The primary space allocation is 52 kilobytes; the secondary, 20 kilobytes. The data sets need not be erased before they are deleted.

Locking on tables in the space is to take place at the page level. Associate the table space with buffer pool BP1. The data sets can be closed when no one is using the table space.

```
CREATE TABLESPACE DSN8S61D
  IN DSN8D61A
  USING STOGROUP DSN8G610
    PRIQTY 52
    SECQTY 20
    ERASE NO
  LOCKSIZE PAGE
  BUFFERPOOL BP1
  CLOSE YES;
```

Example 2: Assume that a large query database application uses a table space to record historical sales data for marketing statistics. Create large table space SALESHX in database DSN8D61A for the application. Create it with 82 partitions, specifying that the data in partitions 80 through 82 is to be compressed.

Let DB2 define the data sets for all the partitions in the table space, using storage group DSN8G610. For each data set, the primary space allocation is 4000 kilobytes, and the secondary space allocation is 130 kilobytes. Except for the data set for partition 82, the data sets do not need to be erased before they are deleted.

Locking on the table is to take place at the page level. There can only be one table in a partitioned table space. Associate the table space with buffer pool BP1. The data sets cannot be closed when no one is using the table space. If there are no CLOSE YES data sets to close, the buffer manager may close the CLOSE NO data sets when the DSMAX is reached.

```
CREATE TABLESPACE SALESHX
  IN DSN8D61A
  USING STOGROUP DSN8G610
    PRIQTY 4000
    SECQTY 130
    ERASE NO
  NUMPARTS 82
  (PART 80
    COMPRESS YES,
   PART 81
    COMPRESS YES,
   PART 82
    ERASE YES
    COMPRESS YES)
  LOCKSIZE PAGE
  BUFFERPOOL BP1
  CLOSE NO;
```

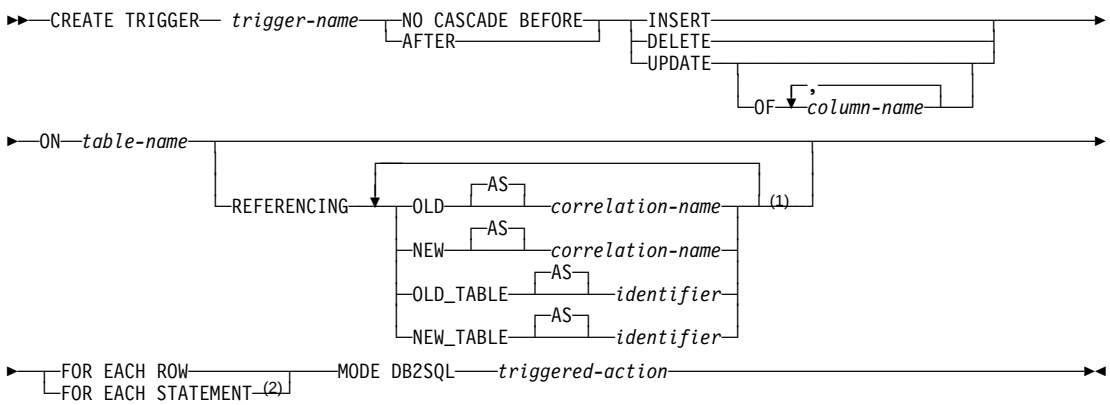
Example 3: Assume that a column named EMP_PHOTO with a data type of BLOB(110K) has been added to the sample employee table for each employee's photo. Create LOB table space PHOTOLTS in database DSN8D61A for the auxiliary table that will hold the BLOB data.

Let DB2 define the data sets for the table space, using storage group DSN8G610. For each data set, the primary space allocation is 3200 kilobytes, and the secondary space allocation is 1600 kilobytes. The data sets do not need to be erased before they are deleted.

```
CREATE LOB TABLESPACE PHOTOLTS
  IN DSN8D61A
  USING STOGROUP DSN8G610
    PRIQTY 3200
    SECQTY 1600
  LOCKSIZE LOB
  BUFFERPOOL BP16K0
  GBPCACHE SYSTEM
  LOG NO
  CLOSE NO;
```

CREATE TRIGGER

Syntax

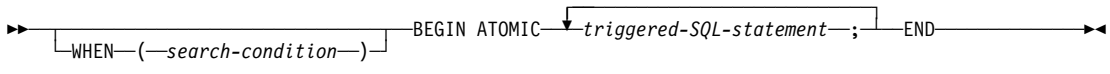


Notes:

1 The same clause must not be specified more than once. OLD_TABLE and NEW_TABLE must be specified only for AFTER triggers.

2 FOR EACH STATEMENT must not be specified for BEFORE triggers.

triggered-action



Examples

Example 1: Create two triggers that track the number of employees that a company manages. The triggering table is the EMPLOYEE table, and the triggers increment and decrement a column with the total number of employees in the COMPANY_STATS table. The tables have these columns:

EMPLOYEE table: ID, NAME, ADDRESS, and POSITION
COMPANY_STATS table: NBEMP, NBPRODUCT, and REVENUE

This example shows the use of transition variables in a row trigger to maintain summary data in another table.

Create the first trigger, NEW_HIRE, so that it increments the number of employees each time a new person is hired; that is, each time a new row is inserted into the

EMPLOYEE table, increase the value of column NBEMP in table COMPANY_STATS by 1.

```
CREATE TRIGGER NEW_HIRE
   AFTER INSERT ON EMPLOYEE
   FOR EACH ROW MODE DB2SQL
   BEGIN ATOMIC
     UPDATE COMPANY_STATS SET NBEMP = NBEMP + 1;
   END
```

Create the second trigger, FORM_EMP, so that it decrements the number of employees each time an employee leaves the company; that is, each time a row is deleted from the table EMPLOYEE, decrease the value of column NBEMP in table COMPANY_STATS by 1.

```
CREATE TRIGGER FORM_EMP
   AFTER DELETE ON EMPLOYEE
   FOR EACH ROW MODE DB2SQL
   BEGIN ATOMIC
     UPDATE COMPANY_STATS SET NBEMP = NBEMP - 1;
   END
```

Example 2: Create a trigger, REORDER, that invokes user-defined function ISSUE_SHIP_REQUEST to issue a shipping request whenever a parts record is updated and the on-hand quantity for the affected part is less than 10% of its maximum stocked quantity. User-defined function ISSUE_SHIP_REQUEST orders a quantity of the part that is equal to the part's maximum stocked quantity minus its on-hand quantity; the function also ensures that the request is sent to the appropriate supplier.

The parts records are in the PARTS table. Although the table has more columns, the trigger is activated only when columns PARTNO and MAX_STOCKED are updated.

```
 CREATE TRIGGER REORDER
AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
REFERENCING NEW AS NROW
FOR EACH ROW MODE DB2SQL
WHEN (NROW.ON_HAND < 0.10 * NROW.MAX_STOCKED)
BEGIN ATOMIC
  VALUES(ISSUE_SHIP_REQUEST(NROW.MAX_STOCKED - NROW.ON_HAND, NROW.PARTNO));
END
```

Example 3: Repeat the scenario in *Example 2* except use a fullselect instead of a VALUES statement to invoke the user-defined function. This example also shows how to define the trigger as a statement trigger instead of a row trigger. For each row in the transition table that evaluates to true for the WHERE clause, a shipping request is issued for the part.

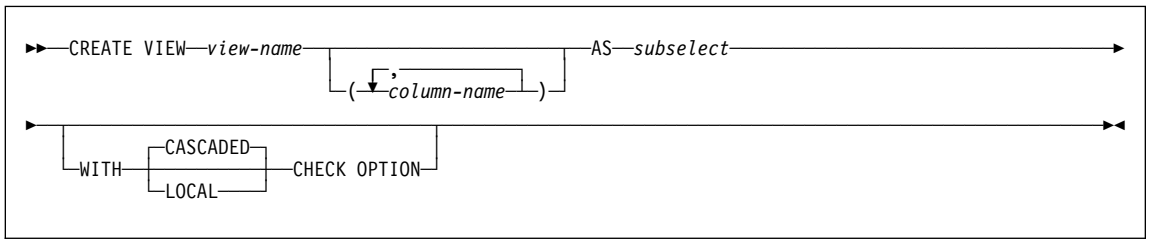
```
 CREATE TRIGGER REORDER
AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
REFERENCING NEW_TABLE AS NTABLE
FOR EACH STATEMENT MODE DB2SQL
  BEGIN ATOMIC
    SELECT ISSUE_SHIP_REQUEST(MAX_STOCKED - ON_HAND, PARTNO)
      FROM NTABLE
    WHERE (ON_HAND < 0.10 * MAX_STOCKED);
END
```

Example 4: Assume that table EMPLOYEE contains column SALARY. Create a trigger, SAL_ADJ, that prevents an update to an employee's salary that exceeds 20% and signals such an error. Have the error that is returned with an SQLSTATE of '75001' and a description. This example shows that the SIGNAL SQLSTATE statement is useful for restricting changes that violate business rules.

```
CREATE TRIGGER SAL_ADJ
  AFTER UPDATE OF SALARY ON EMPLOYEE
  REFERENCING OLD AS OLD_EMP
              NEW AS NEW_EMP
  FOR EACH ROW MODE DB2SQL
  WHEN (NEW_EMP.SALARY > (OLD_EMP.SALARY * 1.20))
    BEGIN ATOMIC
      SIGNAL SQLSTATE '75001' ('Invalid Salary Increase - Exceeds 20%');
    END
```

CREATE VIEW

Syntax



Examples

Example 1: Create the view DSN8610.VPROJRE1. PROJNO, PROJNAME, PROJDEP, RESPEMP, FIRSTNME, MIDINIT, and LASTNAME are column names. The view is a join of tables and is therefore read-only.

```
CREATE VIEW DSN8610.VPROJRE1
  (PROJNO,PROJNAME,PROJDEP,RESPEMP,
   FIRSTNME,MIDINIT,LASTNAME)
  AS SELECT ALL
  PROJNO,PROJNAME,DEPTNO,EMPNO,
  FIRSTNME,MIDINIT,LASTNAME
  FROM DSN8610.PROJ, DSN8610.EMP
  WHERE RESPEMP = EMPNO;
```

In the example, the WHERE clause refers to the column EMPNO, which is contained in one of the base tables but is not part of the view. In general, a column named in the WHERE, GROUP BY, or HAVING clause need not be part of the view.

Example 2: When a view that is defined WITH LOCAL CHECK OPTION is defined on a view that was defined without a check option. You can update or insert rows that do not conform to the definition of the view. Consider the following views:

```
CREATE VIEW UNDER AS SELECT * FROM DSN8610.EMP
                     WHERE SALARY < 35000;

CREATE VIEW OVER AS SELECT * FROM UNDER
                     WHERE SALARY > 30000 WITH LOCAL CHECK OPTION;
```

The following UPDATE statement that uses OVER is successful because the updated rows only need to conform to the definition of OVER (SALARY > 30000):

```
UPDATE OVER SET SALARY = SALARY + 5000;
```

However, not all of the rows that you can retrieve through view OVER (over 35,000 rows) are accessible using view UNDER. For example, issuing:

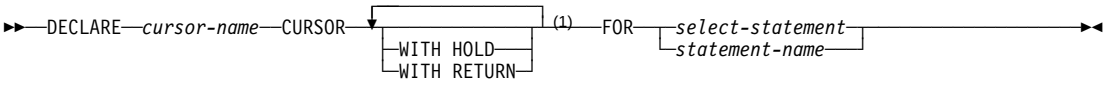
```
SELECT * FROM UNDER
```

returns no rows because no rows conform to the definition of UNDER (SALARY < 35000).

With the CASCADED CHECK OPTION, this situation cannot occur. If OVER had been defined with the WITH CASCADED CHECK OPTION, the UPDATE statement would have failed because the updated rows would not conform to the conjunction of the search conditions OVER and UNDER (SALARY > 3000 and SALARY < 35000).

DECLARE CURSOR

Syntax



Note:
1 The same clause must not be specified more than once.

Examples

The statements in the following examples are assumed to be in PL/I programs.

Example 1: Declare C1 as the cursor of a query to retrieve data from the table DSN8610.DEPT. The query itself appears in the DECLARE CURSOR statement.

```
EXEC SQL DECLARE C1 CURSOR FOR
   SELECT DEPTNO, DEPTNAME, MGRNO
   FROM DSN8610.DEPT
   WHERE ADMRDEPT = 'A00';
```

Example 2: Declare C2 as the cursor for a statement named STMT2.

```
EXEC SQL DECLARE C2 CURSOR FOR STMT2;
```

Example 3: Declare C3 as the cursor for a query to be used in positioned updates of the table DSN8610.EMP. Allow the completed updates to be committed from time to time without closing the cursor.

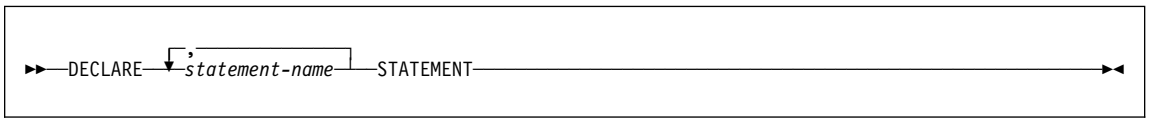
```
EXEC SQL DECLARE C3 CURSOR WITH HOLD FOR
  SELECT * FROM DSN8610.EMP
    FOR UPDATE OF WORKDEPT, PHONENO, JOB, EDLEVEL, SALARY;
```

Example 4: In stored procedure SP1, declare C4 as the cursor for a query of the table DSN8610.PROJ. Enable the cursor to return a result set to the caller of SP1, which performs a commit on return.

```
EXEC SQL DECLARE C4 CURSOR WITH HOLD WITH RETURN FOR
  SELECT PROJNO, PROJNAME
  FROM DSN8610.PROJ
  WHERE DEPTNO = 'A01';
```

DECLARE STATEMENT

Syntax



Example

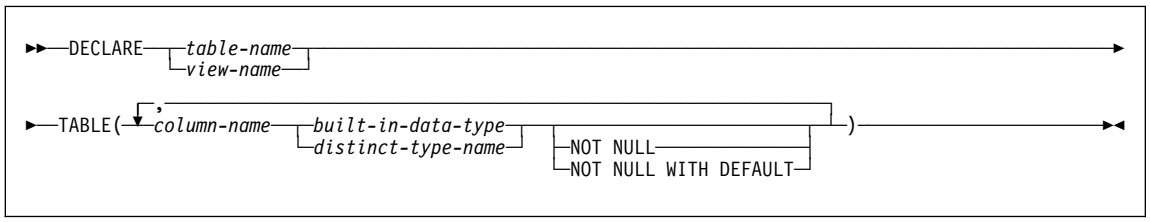
This example shows the use of the DECLARE STATEMENT statement in a PL/I program.

```
EXEC SQL DECLARE OBJECT_STATEMENT STATEMENT;

EXEC SQL INCLUDE SQLDA;
EXEC SQL DECLARE C1 CURSOR FOR OBJECT_STATEMENT;

( SOURCE_STATEMENT IS "SELECT DEPTNO, DEPTNAME,
  MGRNO FROM DSN8610.DEPT WHERE ADMRDEPT = 'A00'"  )

EXEC SQL PREPARE OBJECT_STATEMENT FROM SOURCE_STATEMENT;
EXEC SQL DESCRIBE OBJECT_STATEMENT INTO SQLDA;

(Examine SQLDA)

EXEC SQL OPEN C1;

DO WHILE (SQLCODE = 0);
  EXEC SQL FETCH C1 USING DESCRIPTOR SQLDA;

(Print results)

END;

EXEC SQL CLOSE C1;
```

DECLARE TABLE

DECLARE TABLE

Syntax



Examples

Example 1: Declare the sample employee table, DSN8610.EMP.

```
EXEC SQL DECLARE DSN8610.EMP TABLE
  (EMPNO     CHAR(6)     NOT NULL,
   FIRSTNME  VARCHAR(12) NOT NULL,
   MIDINIT   CHAR(1)     NOT NULL,
   LASTNAME  VARCHAR(15) NOT NULL,
   WORKDEPT  CHAR(3)             ,
   PHONENO   CHAR(4)             ,
   HIREDATE  DATE                ,
   JOB       CHAR(8)             ,
   EDLEVEL   SMALLINT            ,
   SEX       CHAR(1)             ,
   BIRTHDATE DATE                ,
   SALARY    DECIMAL(9,2)        ,
   BONUS     DECIMAL(9,2)        ,
   COMM      DECIMAL(9,2)        );
```

Example 2: Assume that table CANADIAN_SALES keeps information for your company's sales in Canada. The table was created with the following definition:

```
CREATE TABLE CANADIAN_SALES
  (PRODUCT_ITEM   INTEGER,
   MONTH          INTEGER,
   YEAR           INTEGER,
   TOTAL          CANADIAN_DOLLAR);
```

CANADIAN_DOLLAR is a distinct type that was created with the following statement:

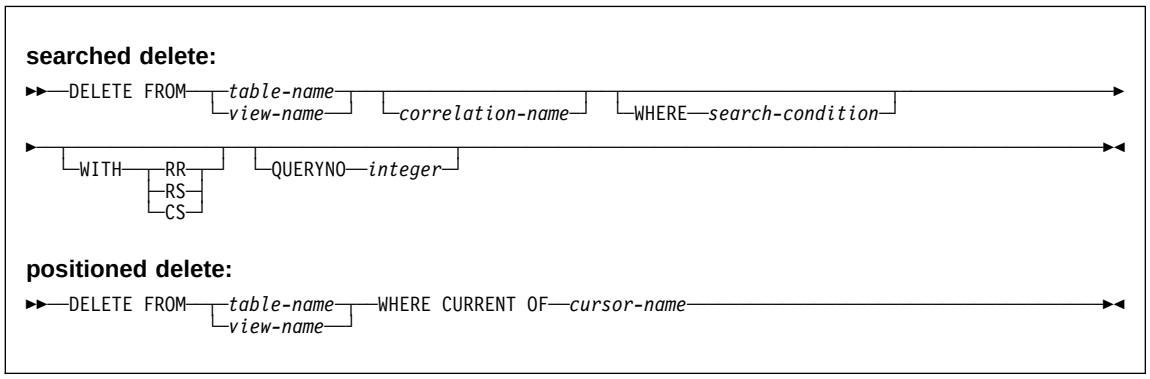
```
CREATE DISTINCT TYPE CANADIAN_DOLLAR
   AS DECIMAL(9,2) WITH COMPARISONS;
```

Declare the CANADIAN_SALES table, using the source type for CANADIAN_DOLLAR instead of the distinct type name.

```
DECLARE TABLE CANADIAN_SALES
  (PRODUCT_ITEM   INTEGER,
   MONTH          INTEGER,
   YEAR           INTEGER,
   TOTAL          DECIMAL(9,2);
```

DELETE

Syntax



Examples

Assume that the statements in these examples are embedded in PL/I programs.

Example 1: From the table DSN8610.EMP delete the row on which the cursor C1 is currently positioned.

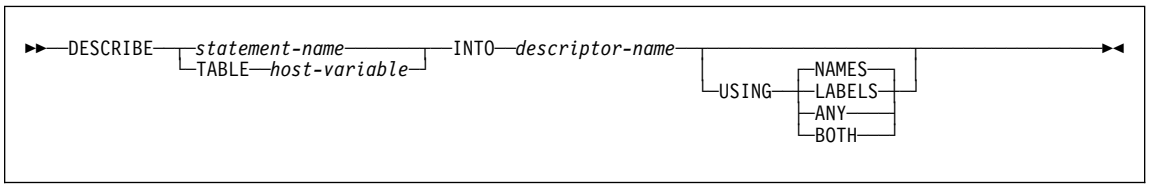
```
EXEC SQL DELETE FROM DSN8610.EMP WHERE CURRENT OF C1;
```

Example 2: From the table DSN8610.EMP, delete all rows for departments E11 and D21.

```
EXEC SQL DELETE FROM DSN8610.EMP
  WHERE WORKDEPT = 'E11' OR WORKDEPT = 'D21';
```

DESCRIBE

Syntax



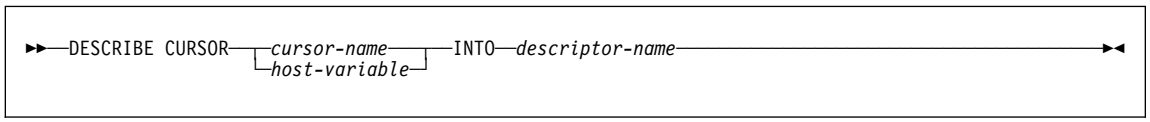
Example

In a PL/I program, execute a DESCRIBE statement with an SQLDA that has no occurrences of SQLVAR. If SQLD is greater than zero, use the value to allocate an SQLDA with the necessary number of occurrences of SQLVAR and then execute a DESCRIBE statement using that SQLDA.

```
EXEC SQL  BEGIN DECLARE SECTION;
  DCL  STMT1_STR   CHAR(200)  VARYING;
EXEC SQL  END DECLARE SECTION;
EXEC SQL  INCLUDE SQLDA;
EXEC SQL  DECLARE DYN_CURSOR CURSOR FOR STMT1_NAME;

... /* code to prompt user for a query, then to generate */
    /* a select-statement in the STMT1_STR           */
EXEC SQL  PREPARE STMT1_NAME FROM :STMT1_STR;

... /* code to set SQLN to zero and to allocate the SQLDA */
EXEC SQL  DESCRIBE STMT1_NAME INTO :SQLDA;

... /* code to check that SQLD is greater than zero, to set */
    /* SQLN to SQLD, then to re-allocate the SQLDA          */
EXEC SQL  DESCRIBE STMT1_NAME INTO :SQLDA;

... /* code to prepare for the use of the SQLDA            */
EXEC SQL  OPEN DYN_CURSOR;

... /* loop to fetch rows from result table                */
EXEC SQL  FETCH DYN_CURSOR USING DESCRIPTOR :SQLDA;
.
.
.
```

DESCRIBE CURSOR

Syntax



Examples

The statements in the following examples are assumed to be in PL/I programs.

Example 1: Place information about the result set associated with cursor C1 into the descriptor named by :sqlda1.

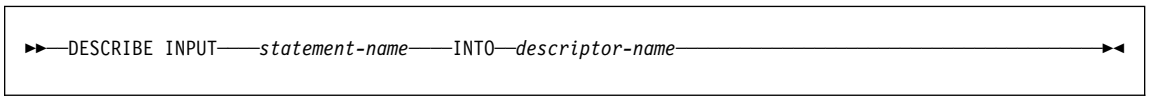
```
EXEC SQL DESCRIBE CURSOR C1 INTO :sqlda1
```

Example 2: Place information about the result set associated with the cursor named by :hv1 into the descriptor named by :sqlda2.

```
EXEC SQL DESCRIBE CURSOR :hv1 INTO :sqlda2
```

DESCRIBE INPUT

Syntax



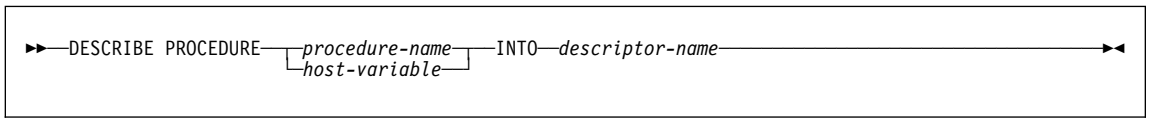
Example

Execute a DESCRIBE INPUT statement with an SQLDA that has enough SQLVAR occurrences to describe any number of input parameters a prepared statement might have. Assume that five parameter markers at most will need to be described and that the input data does not contain LOBs.

```
     /* STMT1_STR contains INSERT statement with VALUES clause  */
   EXEC SQL  PREPARE STMT1_NAME FROM :STMT1_STR;

  ... /* code to set SQLN to 5 and to allocate the SQLDA         */
   EXEC SQL  DESCRIBE INPUT STMT1_NAME INTO :SQLDA;
  .
  .
  .
```

DESCRIBE PROCEDURE

Syntax



Examples

The statements in the following examples are assumed to be in PL/I programs.

Example 1: Place information about the result sets returned by stored procedure P1 into the descriptor named by SQLDA1. Assume that the stored procedure is called with a one-part name from current server SITE2.

```
EXEC SQL CONNECT TO SITE2;
EXEC SQL CALL P1;
EXEC SQL DESCRIBE PROCEDURE P1 INTO :SQLDA1;
```

Example 2: Repeat the scenario in Example 1, but use a two-part name to specify an explicit schema name for the stored procedure to ensure that stored procedure P1 in schema MYSCHEMA is used.

```
EXEC SQL CONNECT TO SITE2;
EXEC SQL CALL MYSCHEMA.P1;
EXEC SQL DESCRIBE PROCEDURE MYSCHEMA.P1 INTO :SQLDA1;
```

Example 3: Place information about the result sets returned by the stored procedure identified by host variable HV1 into the descriptor named by SQLDA2. Assume that host variable HV1 contains the value SITE2.MYSCHEMA.P1 and the stored procedure is called with a three-part name.

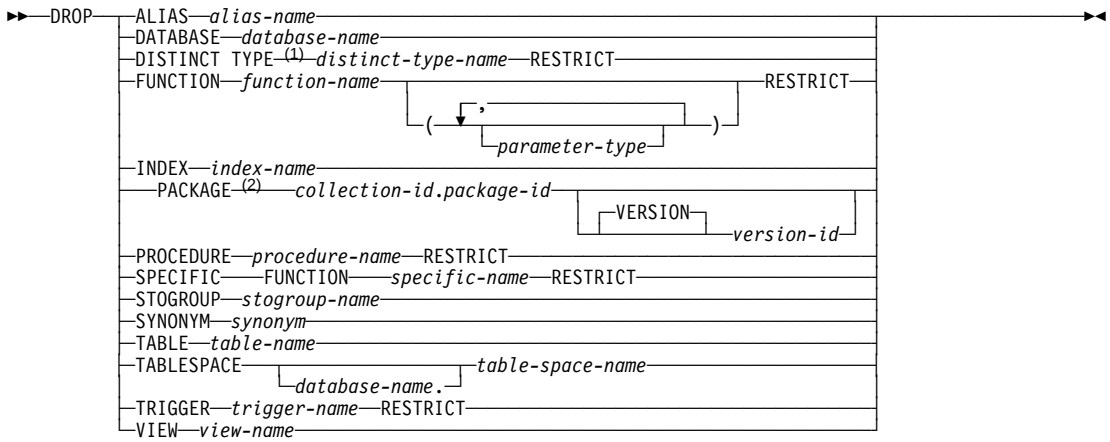
```
EXEC SQL CALL SITE2.MYSCHEMA.P1;
EXEC SQL DESCRIBE PROCEDURE :HV1 INTO :SQLDA2;
```

The preceding example would be invalid if host variable HV1 had contained the value MYSCHEMA.P1, a two-part name. For the example to be valid with that two-part name in host variable HV1, the current server must be the same as the location name that is specified on the CALL statement as the following statements demonstrate. This is the only condition under which the names do not have to be specified the same way and a three-part name on the CALL statement can be used with a two-part name on the DESCRIBE PROCEDURES statement.

```
EXEC SQL CONNECT TO SITE2;
EXEC SQL CALL SITE2.MYSCHEMA.P1;
EXEC SQL ASSOCIATE LOCATORS (:LOC1, :LOC2)
         WITH PROCEDURE :HV1;
```

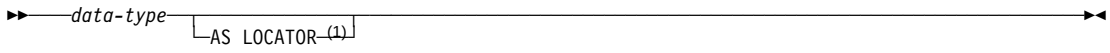
DROP

Syntax



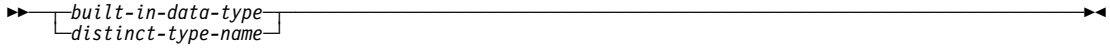
Notes:
1 DATA can be used as a synonym for DISTINCT.
2 PROGRAM can be used as a synonym for PACKAGE.

parameter type:



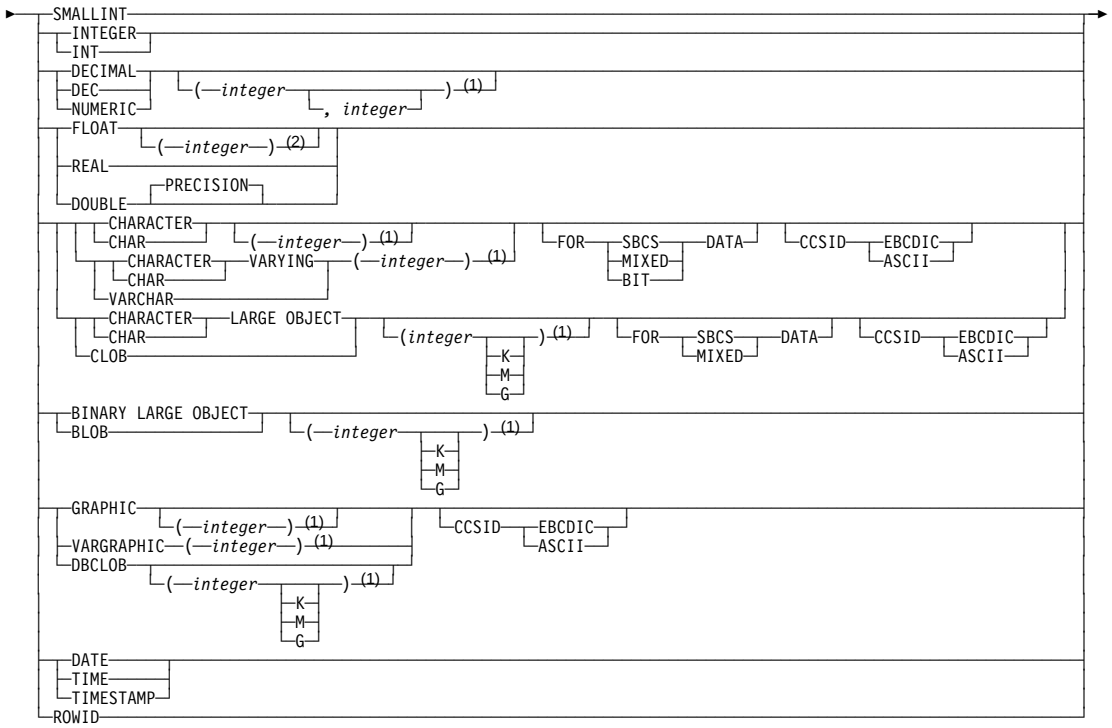
Note:
1 AS LOCATOR can be specified only for a LOB data type or a distinct type based on a LOB data type.

data type:



DROP

built-in-data-type:



Notes:

1 The values that are specified for length, precision, or scale attributes must match the values that were specified when the function was created. Coding specific values is optional. Empty parentheses, (), can be used instead to indicate that DB2 ignores the attributes when determining whether data types match.

2 The value that is specified does not have to match the value that was specified when the function was created because matching is based on data type (REAL or DOUBLE). 1<=*integer*<= 21 indicates REAL and 22<=*integer*<=53 indicates DOUBLE. Coding a specific value is optional. Empty parentheses cannot be used.

Examples

Example 1: Drop table DSN8610.DEPT.

```
DROP TABLE DSN8610.DEPT;
```

Example 2: Drop table space DSN8S61D in database DSN8D61A.

```
DROP TABLESPACE DSN8D61A.DSN8S61D;
```

Example 3: Drop the view DSN8610.VPROJRE1:

```
DROP VIEW DSN8610.VPROJRE1;
```

Example 4: Drop the package DSN8CC0 with the version identifier VERSZZZZ. The package is in the collection DSN8CC61. Use the version identifier to distinguish the package to be dropped from another package with the same name in the same collection.

```
DROP PACKAGE DSN8CC61.DSN8CC0 VERSION VERSZZZZ;
```

Example 5: Drop the package DSN8CC0 with the version identifier "1994-07-14-09.56.30.196952." When a version identifier is generated by the VERSION(AUTO) precompiler option, delimit the version identifier.

```
DROP PACKAGE DSN8CC61.DSN8CC0 VERSION "1994-07-14-09.56.30.196952";
```

Example 6: Drop the distinct type DOCUMENT, if it is not currently in use:

```
DROP DISTINCT TYPE DOCUMENT RESTRICT;
```

Example 7: Assume that you are SMITH and that ATOMIC_WEIGHT is the only function with that name in schema CHEM. Drop ATOMIC_WEIGHT.

```
DROP FUNCTION CHEM.ATOMIC_WEIGHT RESTRICT;
```

Example 8: Assume that you are SMITH and that you created the function CENTER in schema SMITH. Drop CENTER, using the function signature to identify the function instance to be dropped.

```
DROP FUNCTION CENTER(INTEGER, FLOAT) RESTRICT;
```

Example 9: Assume that you are SMITH and that you created another function named CENTER, which you gave the specific name FOCUS97, in schema JOHNSON. Drop CENTER, using the specific name to identify the function instance to be dropped.

```
DROP SPECIFIC FUNCTION JOHNSON.FOCUS97 RESTRICT;
```

Example 10: Assume that you are SMITH and that stored procedure OSMOSIS is in schema BIOLOGY. Drop OSMOSIS.

```
DROP PROCEDURE BIOLOGY.OSMOSIS RESTRICT;
```

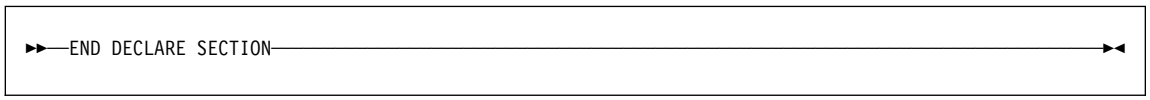
Example 11: Assume that you are SMITH and that trigger BONUS is in your schema. Drop BONUS.

```
DROP TRIGGER BONUS RESTRICT;
```

END DECLARE SECTION

END DECLARE SECTION

Syntax

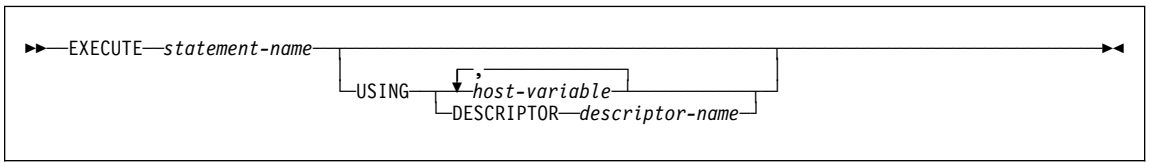


Example

```
EXEC SQL BEGIN DECLARE SECTION;

 (host variable declarations)

EXEC SQL END DECLARE SECTION;
```

EXECUTE

Syntax



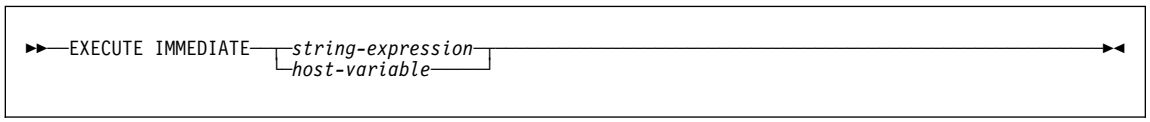
Example

In this example, an INSERT statement with parameter markers is prepared and executed. S1 is a structure that corresponds to the format of DSN8610.DEPT.

```
EXEC SQL PREPARE DEPT_INSERT FROM
  'INSERT INTO DSN8610.DEPT VALUES(?,?,?,?)';

(Check for successful execution and read values into S1)

EXEC SQL EXECUTE DEPT_INSERT USING :S1;
```

EXECUTE IMMEDIATE

Syntax



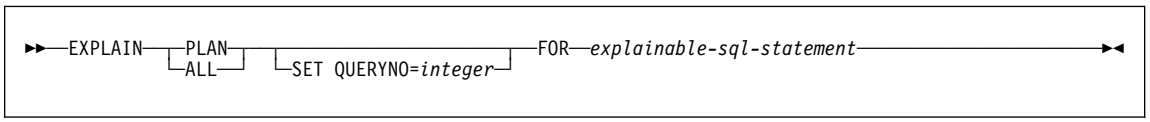
Example

In this PL/I example, the EXECUTE IMMEDIATE statement is used to execute a DELETE statement in which the rows to be deleted are determined by a search-condition specified by the value of PREDS.

```
EXEC SQL EXECUTE IMMEDIATE 'DELETE FROM DSN8610.DEPT
  WHERE' || PREDS;
```

EXPLAIN

Syntax



Examples

Example 1: Determine the steps required to execute the query `'SELECT X.ACTNO...'`. Assume that no set of rows in the PLAN_TABLE has the value 13 for the QUERYNO column.

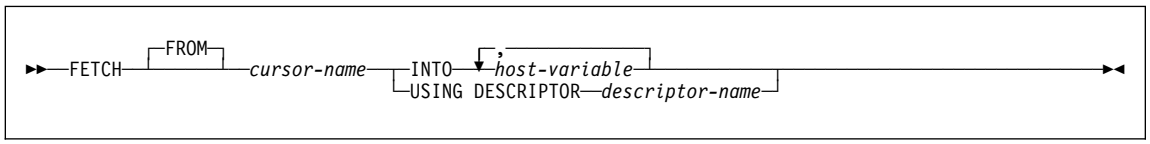
```
EXPLAIN PLAN SET QUERYNO = 13
FOR SELECT X.ACTNO, X.PROJNO, X.EMPNO, Y.JOB, Y.EDLEVEL
    FROM DSN8610.EMPPROJACT X, DSN8610.EMP Y
       WHERE X.EMPNO = Y.EMPNO
          AND X.EMPTIME > 0.5
          AND (Y.JOB = 'DESIGNER' OR Y.EDLEVEL >= 12)
       ORDER BY X.ACTNO, X.PROJNO;
```

Example 2: Retrieve the information returned in Example 1. Assume that a statement table exists, so also retrieve the estimated cost of processing the query. Use the following query, which joins the plan table and the statement table.

```
SELECT * FROM PLAN_TABLE A, DSN_STATEMNT_TABLE B
  WHERE A.QUERYNO = 13 and B.QUERYNO = 13
  ORDER BY A.QBLOCKNO, A.PLANNO, A.MIXOPSEQ;
```

FETCH

Syntax



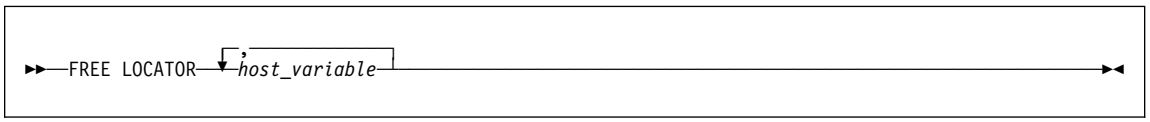
Example

The FETCH statement fetches the results of the SELECT statement into the application program variables DNUM, DNAME, and MNUM. When no more rows remain to be fetched, the not found condition is returned.

```
EXEC SQL DECLARE C1 CURSOR FOR
  SELECT DEPTNO, DEPTNAME, MGRNO FROM DSN8610.DEPT
  WHERE ADMRDEPT = 'A00';

EXEC SQL OPEN C1;

DO WHILE (SQLCODE = 0);
  EXEC SQL FETCH C1 INTO :DNUM, :DNAME, :MNUM;

END;

EXEC SQL CLOSE C1;
```

FREE LOCATOR

Syntax



Example

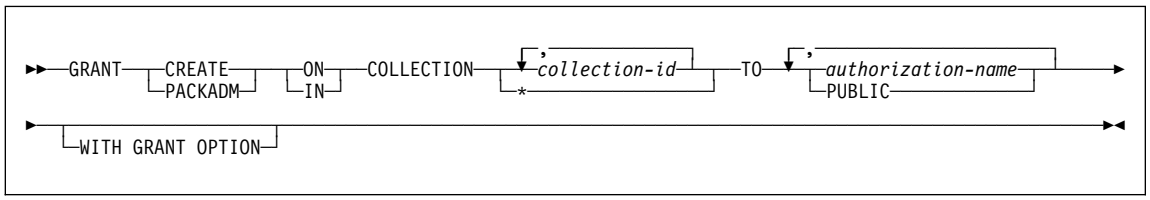
Assume that the employee table contains columns RESUME, HISTORY, and PICTURE and that locators have been established in a program to represent the column values. Free the CLOB locator variables LOCRES and LOCHIST, and the BLOB locator variable LOCPIC.

```
EXEC SQL FREE LOCATOR :LOCRES, :LOCHIST, :LOCPIC
```

GRANT (collection privileges)

GRANT (collection privileges)

Syntax



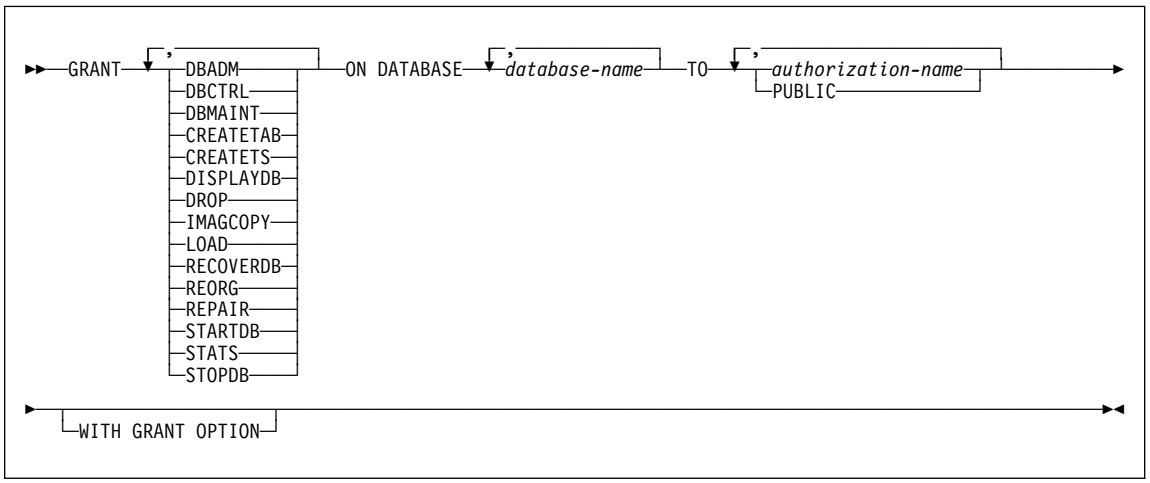
Example

Grant the privilege to create new packages in collections QAACLONE and DSN8CC61 to CLARK.

```
GRANT CREATE IN COLLECTION QAACLONE, DSN8CC61 TO CLARK;
```

GRANT (database privileges)

Syntax



Examples

Example 1: Grant drop privileges on database DSN8D61A to user PEREZ.

```
GRANT DROP
  ON DATABASE DSN8D61A
  TO PEREZ;
```

Example 2: Grant repair privileges on database DSN8D61A to all local users.

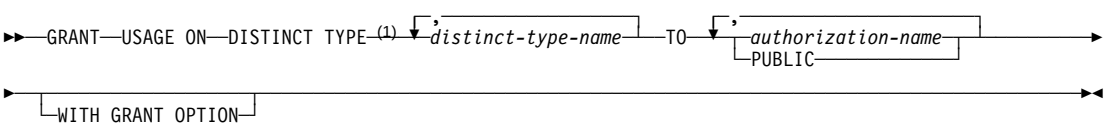
```
GRANT REPAIR
  ON DATABASE DSN8D61A
  TO PUBLIC;
```

Example 3: Grant authority to create new tables and load tables in database DSN8D61A to users WALKER, PIANKA, and FUJIMOTO, and give them grant privileges.

```
GRANT CREATETAB,LOAD
  ON DATABASE DSN8D61A
  TO WALKER,PIANKA,FUJIMOTO
  WITH GRANT OPTION;
```

GRANT (distinct type privileges)

Syntax



Note:
1 DATA can be used as a synonym for DISTINCT. DISTINCT is the keyword that is used on CREATE.

Examples

Example 1: Grant the USAGE privilege on distinct type SHOE_SIZE to user JONES. This GRANT statement does not give JONES the privilege to execute the cast functions that are associated with the distinct type SHOE_SIZE.

```
GRANT USAGE ON DISTINCT TYPE SHOE_SIZE TO JONES;
```

Example 2: Grant the USAGE privilege on distinct type US_DOLLAR to all users at the current server.

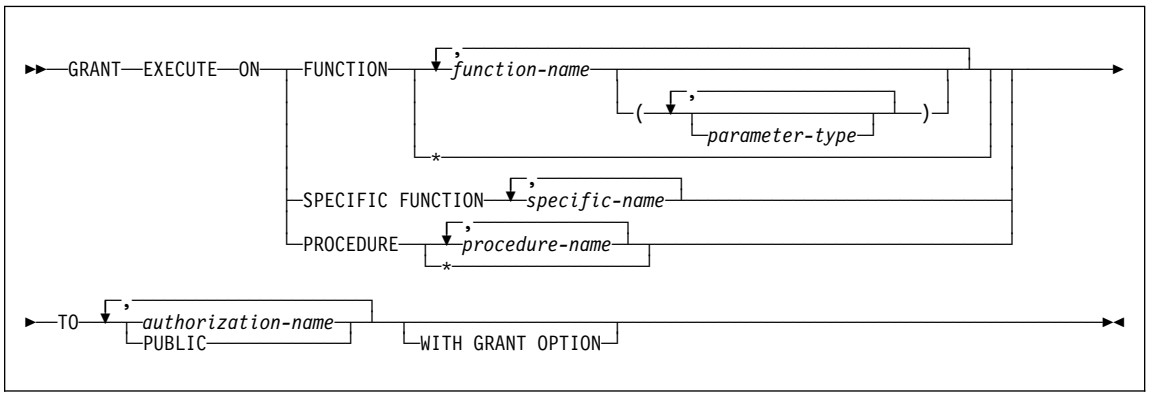
```
GRANT USAGE ON DISTINCT TYPE US_DOLLAR TO PUBLIC;
```

Example 3: Grant the USAGE privilege on distinct type CANADIAN_DOLLAR to the administrative assistant (ADMIN_A) , and give this user the ability to grant the USAGE privilege on the distinct type to others. The administrative assistant cannot grant the privilege to execute the cast functions that are associated with the distinct type CANADIAN_DOLLAR because WITH GRANT OPTION does not give the administrative assistant the EXECUTE authority on these cast functions.

```
GRANT USAGE ON DISTINCT TYPE CANADIAN_DOLLAR TO ADMIN_A
      WITH GRANT OPTION;
```

GRANT (function or procedure privileges)

Syntax



parameter type:

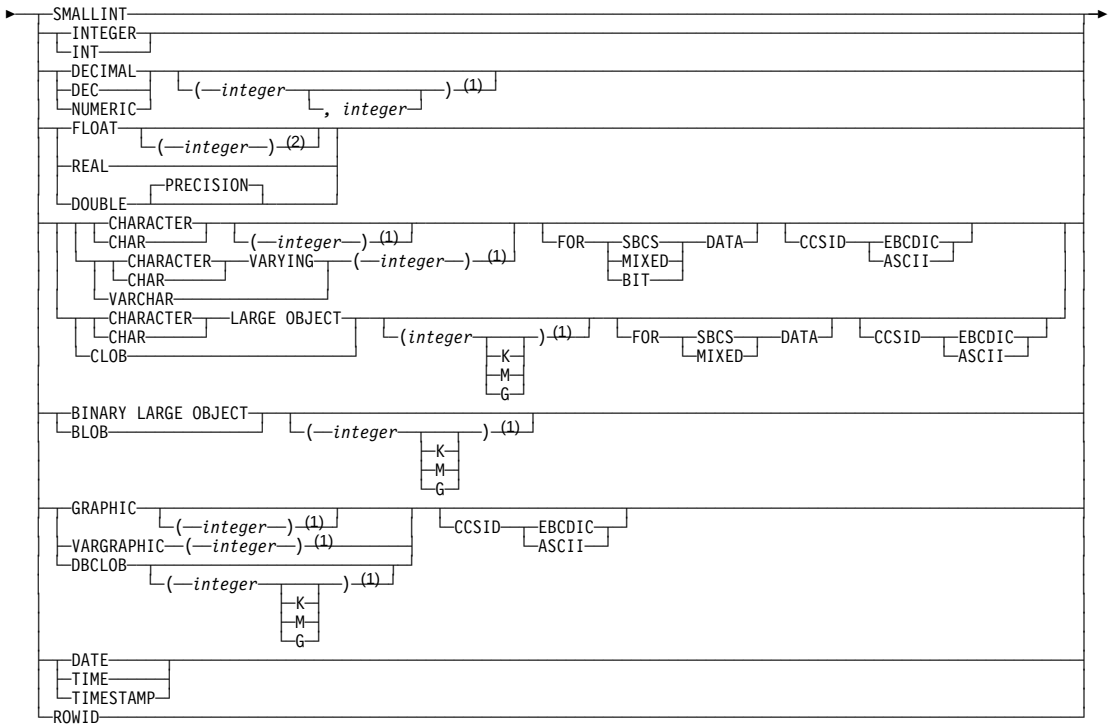
Note:

1 AS LOCATOR can be specified only for a LOB data type or a distinct type based on a LOB data type.

data type:

GRANT (function or procedure privileges)

built-in data type:



Notes:

1. The values that are specified for length, precision, or scale attributes must match the values that were specified when the function was created. Coding specific values is optional. Empty parentheses, (), can be used instead to indicate that DB2 ignores the attributes when determining whether data types match.
2. The value that is specified does not have to match the value that was specified when the function was created because matching is based on data type (REAL or DOUBLE). 1<=*integer*<= 21 indicates REAL and 22<=*integer*<=53 indicates DOUBLE. Coding a specific value is optional. Empty parentheses cannot be used.

Examples

Example 1: Grant the EXECUTE privilege on function CALC_SALARY to user JONES. Assume that there is only one function in the schema with function name CALC_SALARY.

```
GRANT EXECUTE ON FUNCTION CALC_SALARY TO JONES;
```

Example 2: Grant the EXECUTE privilege on procedure VACATION_ACCR to all users at the current server.

```
GRANT EXECUTE ON PROCEDURE VACATION_ACCR TO PUBLIC;
```

Example 3: Grant the EXECUTE privilege on function DEPT_TOTALS to the administrative assistant and give the assistant the ability to grant the EXECUTE privilege on this function to others. The function has the specific name DEPT85_TOT. Assume that the schema has more than one function that is named DEPT_TOTALS.

```
GRANT EXECUTE ON SPECIFIC FUNCTION DEPT85_TOT TO ADMIN_A
      WITH GRANT OPTION;
```

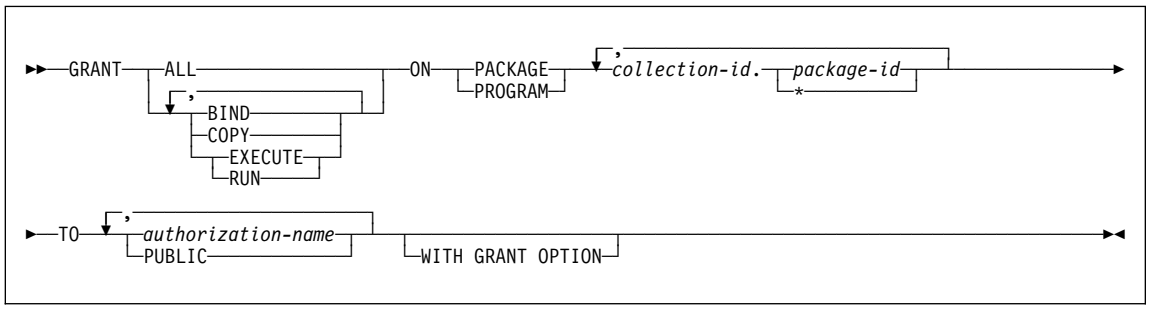
Example 4: Grant the EXECUTE privilege on function NEW_DEPT_HIRES to HR (Human Resources). The function has two input parameters with data types of INTEGER and CHAR(10), respectively. Assume that the schema has more than one function that is named NEW_DEPT_HIRES.

```
GRANT EXECUTE ON FUNCTION NEW_DEPT_HIRES (INTEGER, CHAR(10))
      TO HR;
```

You can also code the CHAR(10) data type as CHAR().

GRANT (package privileges)

Syntax



Examples

Example 1: Grant the privilege to copy all packages in collection DSN8CC61 to LEWIS.

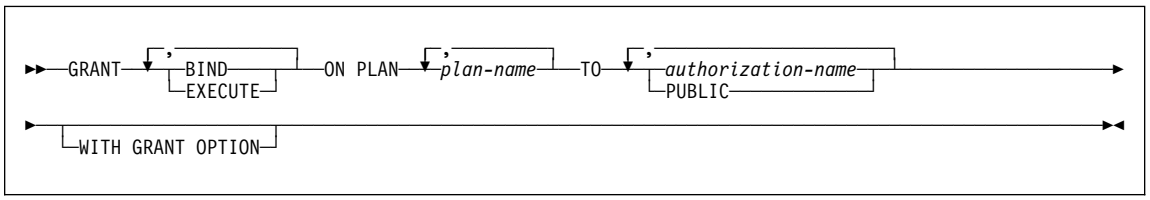
```
GRANT COPY ON PACKAGE DSN8CC61.* TO LEWIS;
```

Example 2: You have the BIND privilege with GRANT authority over the package CLCT1.PKG1. You have the EXECUTE privilege with GRANT authority over the package CLCT2.PKG2. You have no other privileges with GRANT authority over any package in the collections CLCT1 AND CLCT2. Hence, the following statement, when executed by you, grants LEWIS the BIND privilege on CLCT1.PKG1 and the EXECUTE privilege on CLCT2.PKG2, and makes no other grant. The privileges granted include no GRANT authority.

```
GRANT ALL ON PACKAGE CLCT1.PKG1, CLCT2.PKG2 TO JONES;
```

GRANT (plan privileges)

Syntax



Examples

Example 1: Grant the privilege to bind plan DSN8IP61 to user JONES.

```
GRANT BIND ON PLAN DSN8IP61 TO JONES;
```

Example 2: Grant privileges to bind and execute plan DSN8CP61 to all users at the current server.

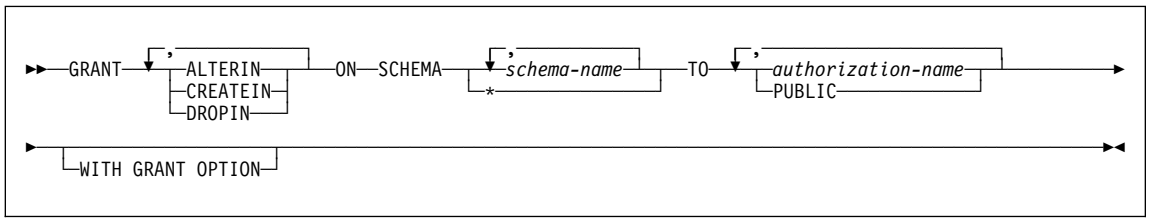
```
GRANT BIND,EXECUTE ON PLAN DSN8CP61 TO PUBLIC;
```

Example 3: Grant the privilege to execute plan DSN8CP61 to users ADAMSON and BROWN with grant option.

```
GRANT EXECUTE ON PLAN DSN8CP61 TO ADAMSON,BROWN WITH GRANT OPTION;
```

GRANT (schema privileges)

Syntax



Examples

Example 1: Grant the CREATEIN privilege on schema T_SCORES to user JONES.

```
GRANT CREATEIN ON SCHEMA T_SCORES TO JONES;
```

Example 2: Grant the CREATEIN privilege on schema VAC to all users at the current server.

```
GRANT CREATEIN ON SCHEMA VAC TO PUBLIC;
```

Example 3: Grant the ALTERIN privilege on schema DEPT to the administrative assistant and give the grantee the ability to grant ALTERIN privileges on this schema to others.

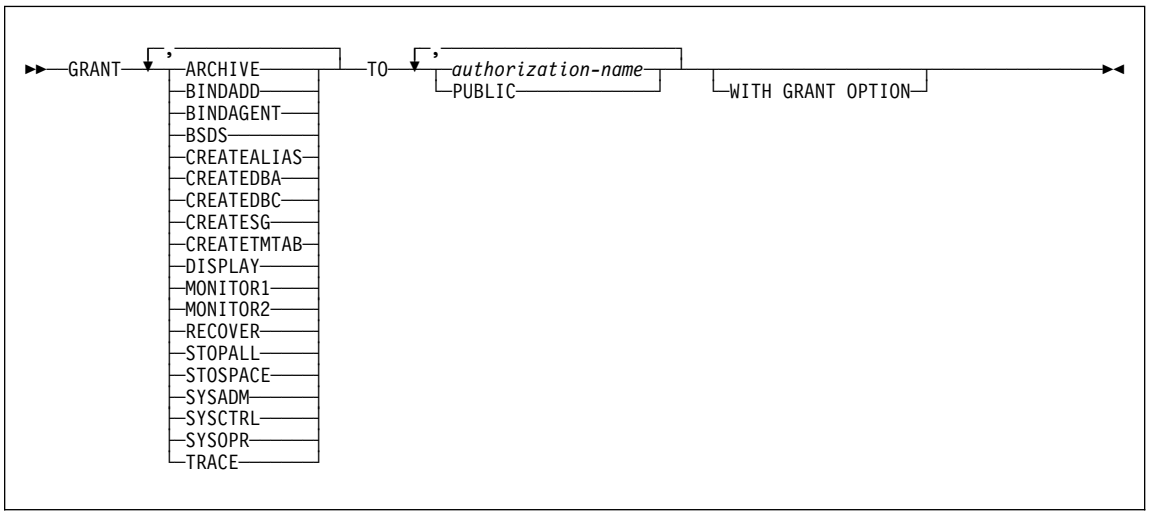
```
GRANT ALTERIN ON SCHEMA DEPT TO ADMIN_A
      WITH GRANT OPTION;
```

Example 4: Grant the CREATEIN, ALTERIN, and DROPIN privileges on schemas NEW_HIRE, PROMO, and RESIGN to HR (Human Resources).

```
GRANT CREATEIN, ALTERIN, DROPIN ON SCHEMA NEW_HIRE, PROMO, RESIGN TO HR;
```

GRANT (system privileges)

Syntax



Examples

Example 1: Grant DISPLAY privileges to user LUTZ.

```
GRANT DISPLAY
  TO LUTZ;
```

Example 2: Grant BSDS and RECOVER privileges to users PARKER and SETRIGHT, with the WITH GRANT OPTION.

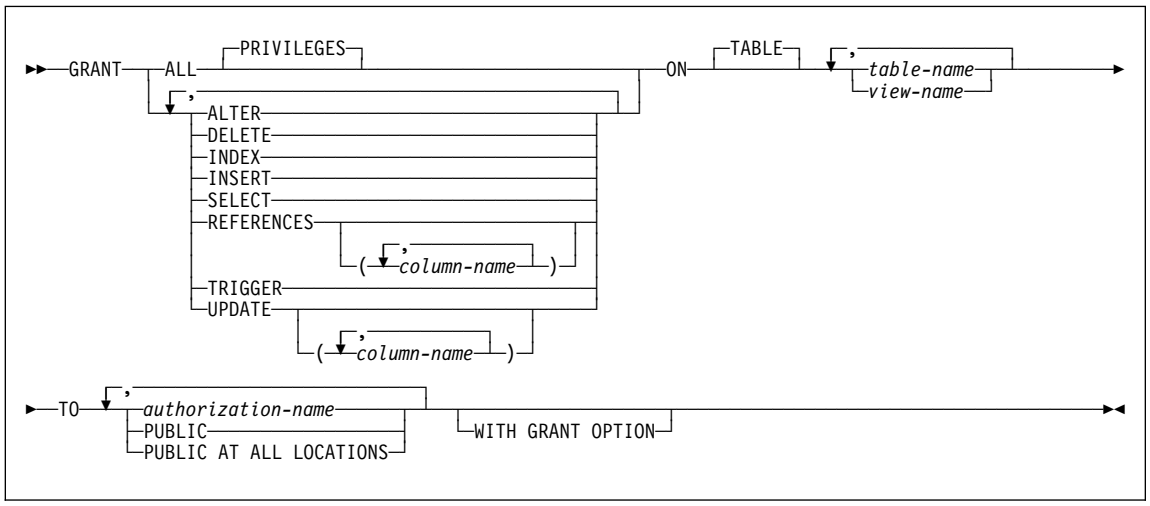
```
GRANT BSDS,RECOVER
  TO PARKER,SETRIGHT
  WITH GRANT OPTION;
```

Example 3: Grant TRACE privileges to all local users.

```
GRANT TRACE
  TO PUBLIC;
```

GRANT (table or view privileges)

Syntax



Examples

Example 1: Grant SELECT privileges on table DSN8610.EMP to user PULASKI.

```
GRANT SELECT ON DSN8610.EMP TO PULASKI;
```

Example 2: Grant UPDATE privileges on columns EMPNO and WORKDEPT in table DSN8610.EMP to all users at the current server.

```
GRANT UPDATE (EMPNO,WORKDEPT) ON TABLE DSN8610.EMP TO PUBLIC;
```

Example 3: Grant all privileges on table DSN8610.EMP to users KWAN and THOMPSON, with the WITH GRANT OPTION.

```
GRANT ALL ON TABLE DSN8610.EMP TO KWAN,THOMPSON WITH GRANT OPTION;
```

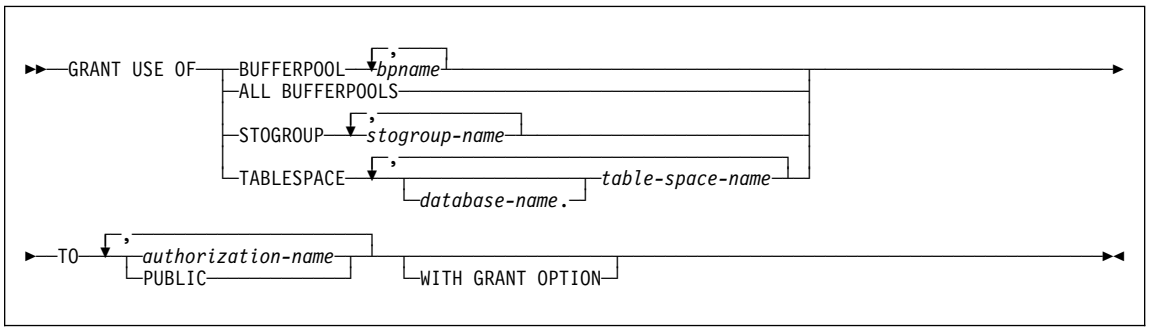
Example 4: Grant the SELECT and UPDATE privileges on the table DSN8610.DEPT to every user in the network.

```
GRANT SELECT, UPDATE ON TABLE DSN8610.DEPT
  TO PUBLIC AT ALL LOCATIONS;
```

Even with this grant, it is possible that some network users do not have access to the table at all, or to any other object at the table's subsystem. Controlling access to the subsystem involves the communications databases at the subsystems in the network.

GRANT (use privileges)

Syntax



Examples

Example 1: Grant authority to use buffer pools BP1 and BP2 to user MARINO.

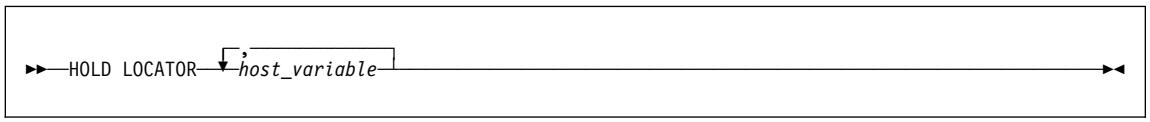
```
GRANT USE OF BUFFERPOOL BP1,BP2
  TO MARINO;
```

Example 2: Grant to all local users the authority to use table space DSN8S61D in database DSN8D61A.

```
GRANT USE OF TABLESPACE
  DSN8D61A.DSN8S61D
  TO PUBLIC;
```

HOLD LOCATOR

Syntax



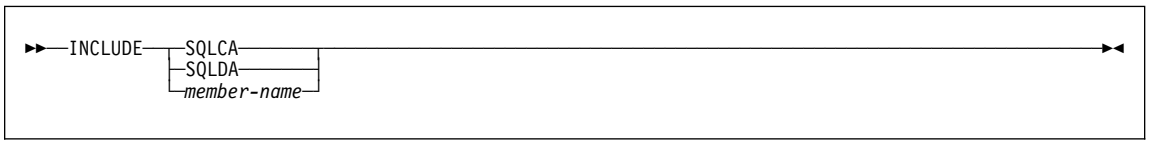
Example

Assume that the employee table contains columns RESUME, HISTORY, and PICTURE and that locators have been established in a program to represent the values represented by the columns. Give the CLOB locator variables LOCRES and LOCHIST, and the BLOB locator variable LOCPIC the hold property.

```
EXEC SQL HOLD LOCATOR :LOCRES, :LOCHIST, :LOCPIC
```

INCLUDE

Syntax



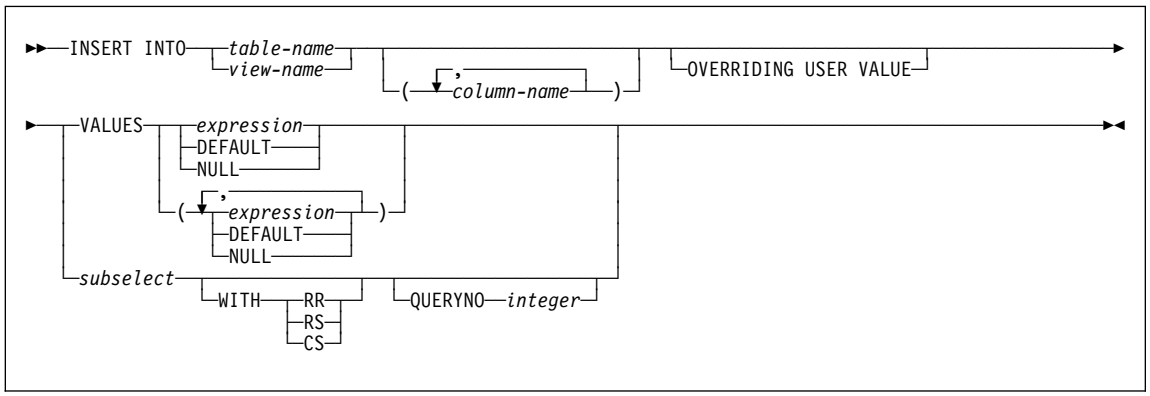
Example

Include an SQL communications area in a PL/I program.

```
EXEC SQL INCLUDE SQLCA;
```

INSERT

Syntax



Examples

Example 1: Insert values into sample table DSN8610.EMP.

```
INSERT INTO DSN8610.EMP
  VALUES ('000205','MARY','T','SMITH','D11','2866',
           '1981-08-10','ANALYST',16,'F','1956-05-22',
          16345,500,2300);
```

Example 2: Populate the temporary table SMITH.TEMPEMPL with data from sample table DSN8610.EMP.

```
INSERT INTO SMITH.TEMPEMPL
  SELECT *
  FROM DSN8610.EMP;
```

Example 3: Populate the temporary table SMITH.TEMPEMPL with data from department D11 in sample table DSN8610.EMP.

```
INSERT INTO SMITH.TEMPEMPL
  SELECT *
  FROM DSN8610.EMP
  WHERE WORKDEPT='D11';
```

Example 4: Insert a row into sample table DSN8610.EMP_PHOTO_RESUME. Set the value for column EMPNO to the value in host variable HV_ENUM. Let the value for column EMP_ROWID be generated because it was defined with a row ID data type and with clause GENERATED ALWAYS.

```
INSERT INTO DSN8610.EMP_PHOTO_RESUME(EMPNO, EMP_ROWID)
  VALUES (:HV_ENUM, DEFAULT);
```

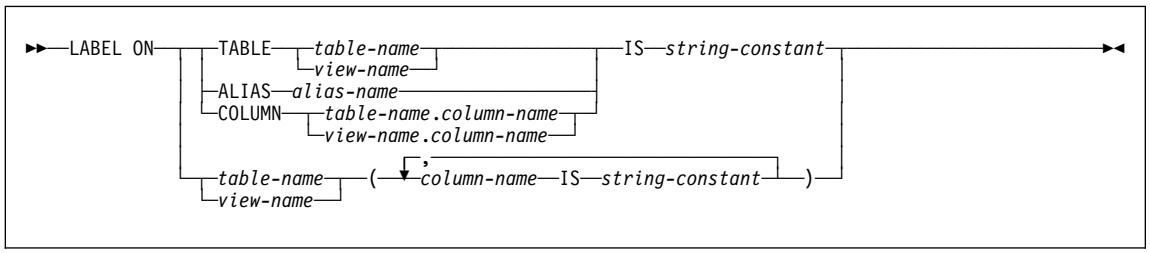
Unlike columns defined as GENERATED BY DEFAULT for which you can insert a value, you cannot insert data into a column that is defined as GENERATED ALWAYS. Therefore, if you were to try to insert a value into EMP_ROWID, instead of specifying DEFAULT as above, the statement would fail unless you specify OVERRIDING USER VALUE. For columns that are defined as GENERATED ALWAYS, the OVERRIDING USER VALUE clause causes DB2 to ignore any user-specified value and generate a value instead.

For example, assume that you want to copy the rows in DSN8610.EMP_PHOTO_RESUME to another table that has a similar definition (both tables have a ROWID columns defined as GENERATED ALWAYS). For following INSERT statement, the OVERRIDING USER VALUE clause causes DB2 to ignore the EMP_ROWID column values from DSN8610.EMP_PHOTO_RESUME and generate values for the corresponding ROWID column in B.EMP_PHOTO_RESUME.

```
INSERT INTO B.EMP_PHOTO_RESUME
  OVERRIDING USER VALUE
  SELECT * FROM DSN8610.EMP_PHOTO_RESUME;
```

LABEL ON

Syntax



Examples

Example 1: Enter a label on the DEPTNO column of table DSN8610.DEPT.

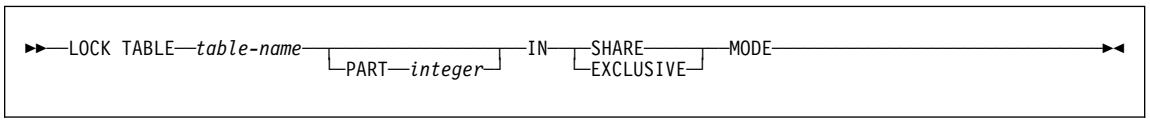
```
LABEL ON COLUMN DSN8610.DEPT.DEPTNO
  IS 'DEPARTMENT NUMBER';
```

Example 2: Enter labels on two columns in table DSN8610.DEPT.

```
LABEL ON DSN8610.DEPT
 (MGRNO IS 'MANAGER'S EMPLOYEE NUMBER',
  ADMRDEPT IS 'ADMINISTERING DEPARTMENT');
```

LOCK TABLE

Syntax



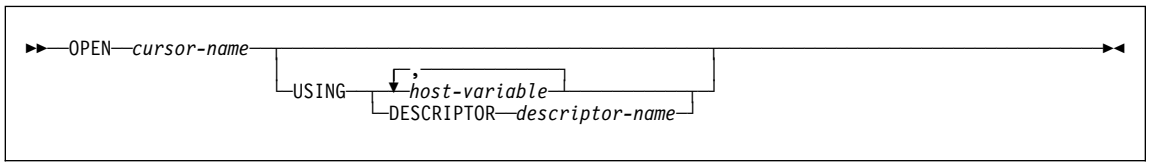
Example

Obtain a lock on the sample table named DSN8610.EMP, which resides in a partitioned table space. The lock obtained applies to every partition and prevents other application programs from either reading or updating the table.

```
LOCK TABLE DSN8610.EMP IN EXCLUSIVE MODE;
```

OPEN

Syntax



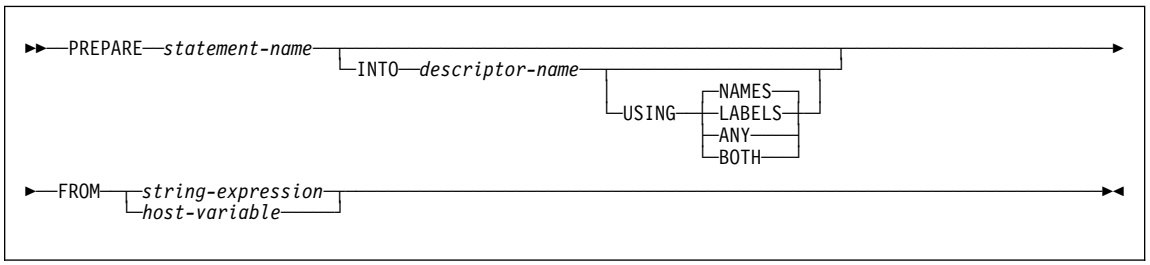
Example

The OPEN statement in the following example places the cursor at the beginning of the rows to be fetched.

```
EXEC SQL DECLARE C1 CURSOR FOR
  SELECT DEPTNO, DEPTNAME, MGRNO FROM DSN8610.DEPT
  WHERE ADMRDEPT = 'A00';

EXEC SQL OPEN C1;

DO WHILE (SQLCODE = 0);
  EXEC SQL FETCH C1 INTO :DNUM, :DNAME, :MNUM;

END;

EXEC SQL CLOSE C1;
```

PREPARE

Syntax



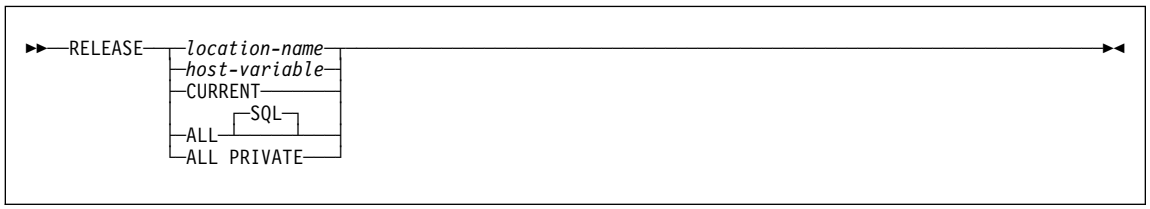
Example

In this PL/I example, an INSERT statement with parameter markers is prepared and executed. Before execution, values for the parameter markers are read into the host variables S1, S2, S3, S4, and S5.

```
EXEC SQL PREPARE DEPT_INSERT FROM
  'INSERT INTO DSN8610.DEPT VALUES(?,?,?,?,?)';

(Check for successful execution and read values into host variables)

EXEC SQL EXECUTE DEPT_INSERT USING :S1, :S2, :S3, :S4, :S5;
```

RELEASE

Syntax



Examples

Example 1: The SQL connection to TOROLAB1 is not needed in the next unit of work. The following statement causes it to be ended during the next commit operation:

```
EXEC SQL RELEASE TOROLAB1;
```

Example 2: The current SQL connection is not needed in the next unit of work. The following statement causes it to be ended during the next commit operation:

```
EXEC SQL RELEASE CURRENT;
```

Example 3: The first phase of an application involves explicit CONNECTs to remote servers and the second phase involves the use of DB2 private protocol access with the local DB2 subsystem as the application server. None of the existing connections are needed in the second phase and their existence could prevent the allocation of DB2 private connections. Accordingly, the following statement is executed before the commit operation that separates the two phases:

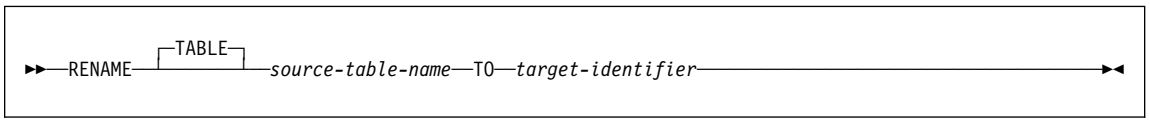
```
EXEC SQL RELEASE ALL SQL;
```

Example 4: The first phase of an application involves the use of DB2 private protocol access with the local DB2 subsystem as the application server and the second phase involves explicit CONNECTs to remote servers. The existence of the DB2 private connections allocated during the first phase could cause a CONNECT operation to fail. Accordingly, the following statement is executed before the commit operation that separates the two phases:

```
EXEC SQL RELEASE ALL PRIVATE;
```

RENAME

Syntax



Example

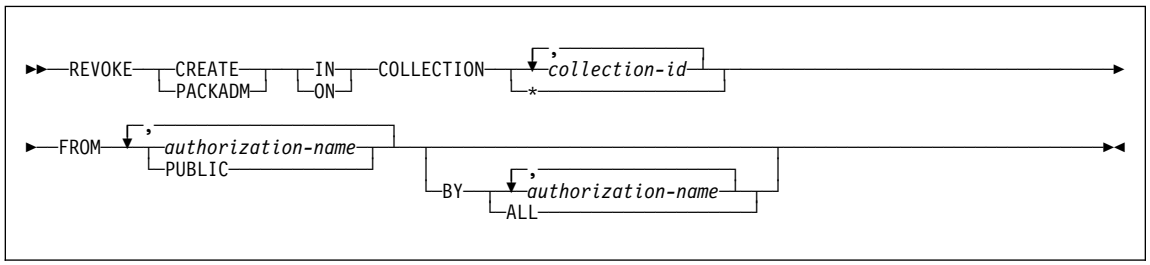
Change the name of the EMP table to EMPLOYEE:

```
RENAME TABLE EMP TO EMPLOYEE;
```

REVOKE (collection privileges)

REVOKE (collection privileges)

Syntax



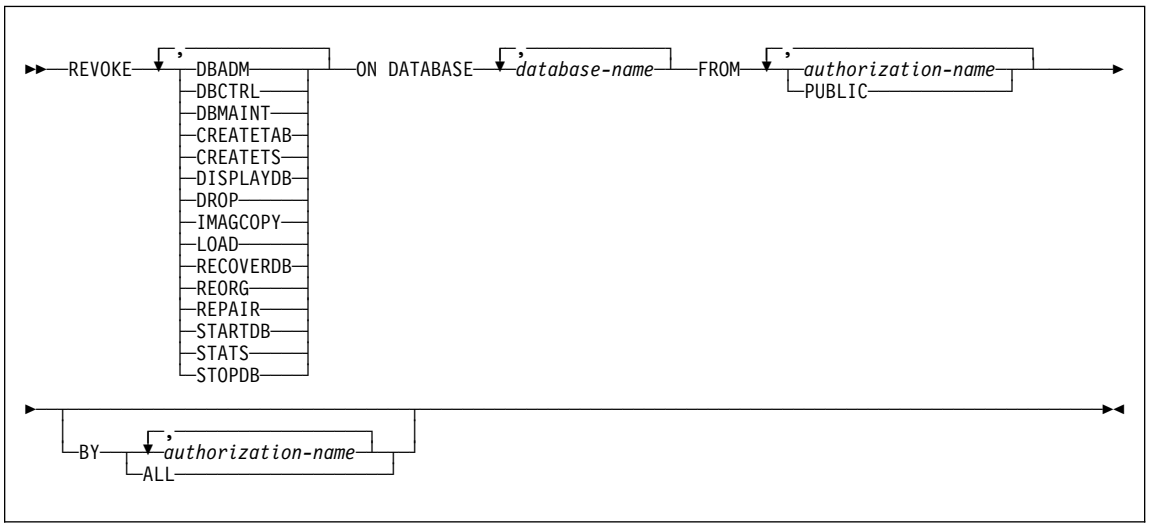
Example

Revoke the privilege to create new packages in collections QAACLONE and DSN8CC61 from CLARK.

```
REVOKE CREATE IN COLLECTION QAACLONE, DSN8CC61 FROM CLARK;
```

REVOKE (database privileges)

Syntax



Examples

Example 1: Revoke drop privileges on database DSN8D61A from user PEREZ.

```
REVOKE DROP
  ON DATABASE DSN8D61A
  FROM PEREZ;
```

Example 2: Revoke repair privileges on database DSN8D61A from all local users. (Grants to specific users will not be affected.)

```
REVOKE REPAIR
  ON DATABASE DSN8D61A
  FROM PUBLIC;
```

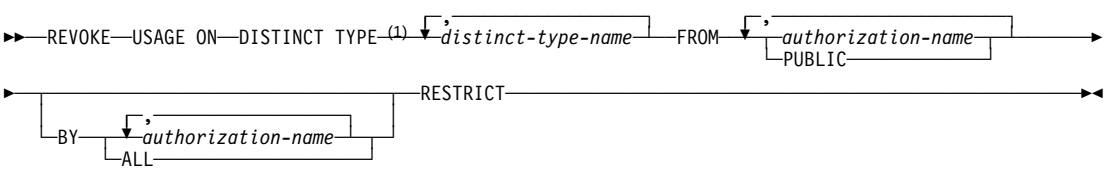
Example 3: Revoke authority to create new tables and load tables in database DSN8D61A from users WALKER, PIANKA, and FUJIMOTO.

```
REVOKE CREATETAB,LOAD
  ON DATABASE DSN8D61A
  FROM WALKER,PIANKA,FUJIMOTO;
```

REVOKE (distinct type privileges)

REVOKE (distinct type privileges)

Syntax



Note:
1 DATA is a synonym for DISTINCT. DISTINCT is the keyword that is used on CREATE.

Examples

Example 1: Revoke the USAGE privilege on distinct type SHOESIZE from user JONES.

```
REVOKE USAGE ON DISTINCT TYPE SHOESIZE FROM JONES RESTRICT;
```

Example 2: Revoke the USAGE privilege on distinct type US_DOLLAR from all users at the current server except for those who have been specifically granted USAGE and not through PUBLIC.

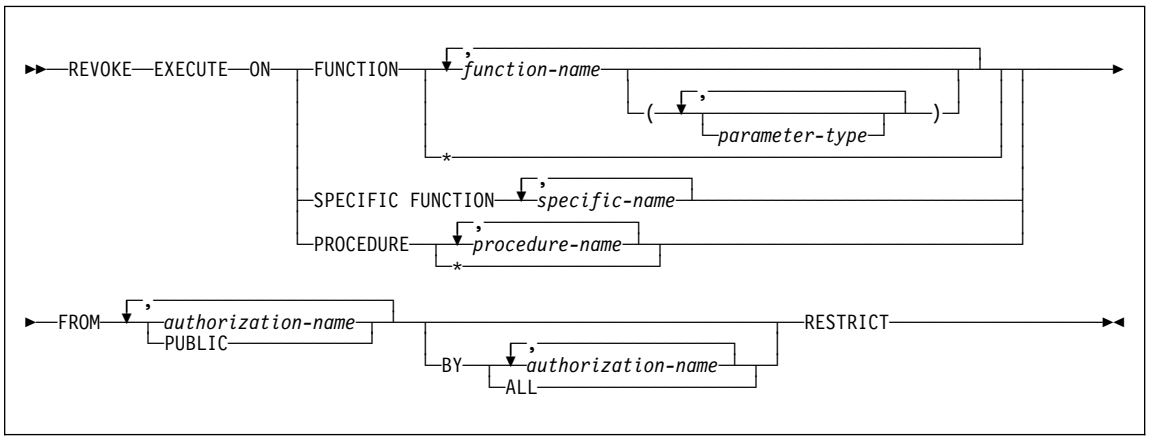
```
REVOKE USAGE ON DISTINCT TYPE US_DOLLAR FROM PUBLIC RESTRICT;
```

Example 3: Revoke the USAGE privilege on distinct type CANADIAN_DOLLARS from the administrative assistant (ADMIN_A) .

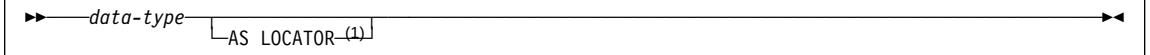
```
REVOKE USAGE ON DISTINCT TYPE CANADIAN_DOLLARS
      FROM ADMIN_A RESTRICT;
```

REVOKE (function or procedure privileges)

Syntax



parameter type:



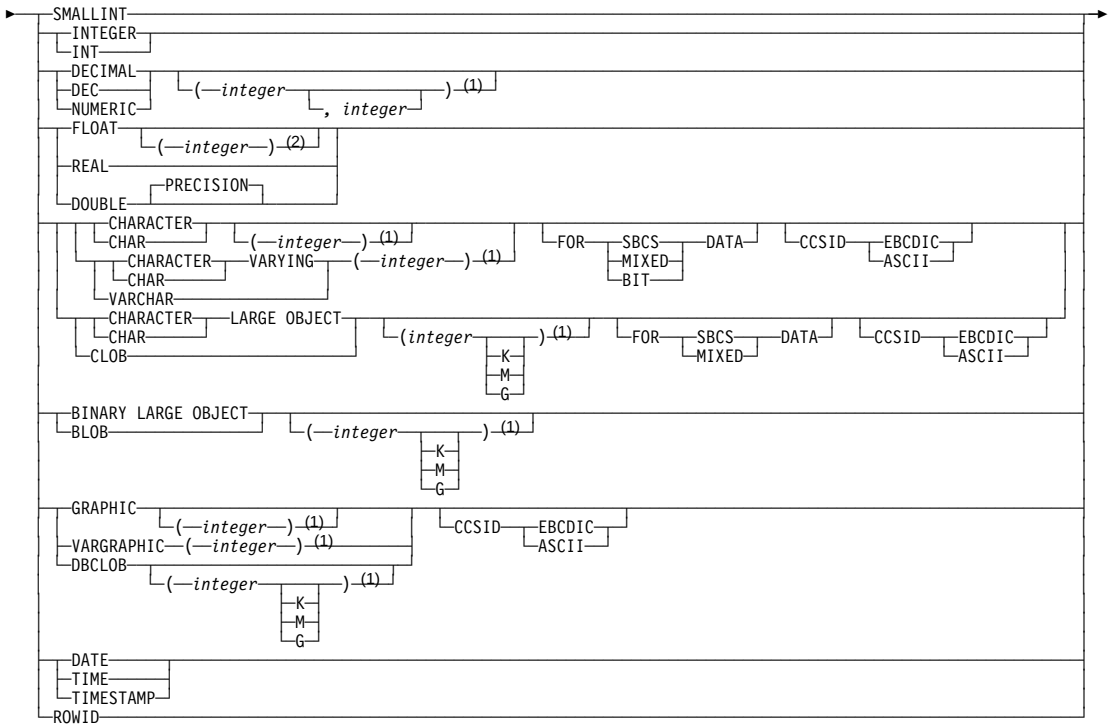
Note:

1 AS LOCATOR can be specified only for a LOB data type or a distinct type that is based on a LOB data type.

data type:

REVOKE (function or procedure privileges)

built-in data type:



Notes:

1. The values that are specified for length, precision, or scale attributes must match the values that were specified when the function was created. Coding specific values is optional. Empty parentheses, (), can be used instead to indicate that DB2 ignores the attributes when determining whether data types match.
2. The value that is specified does not have to match the value that was specified when the function was created because matching is based on data type (REAL or DOUBLE). 1<=*integer*<= 21 indicates REAL and 22<=*integer*<=53 indicates DOUBLE. Coding a specific value is optional. Empty parentheses cannot be used.

Examples

Example 1: Revoke the EXECUTE privilege on function CALC_SALARY for user JONES. Assume that there is only one function in the schema with function CALC_SALARY.

```
REVOKE EXECUTE ON FUNCTION CALC_SALARY FROM JONES RESTRICT;
```

Example 2: Revoke the EXECUTE privilege on procedure VACATION_ACCR from all users at the current server.

```
REVOKE EXECUTE ON PROCEDURE VACATION_ACCR FROM PUBLIC RESTRICT;
```

Example 3: Revoke the privilege of the administrative assistant to grant EXECUTE privileges on function DEPT_TOTAL to other users. The administrative assistant will still have the EXECUTE privilege on function *DEPT_TOTALS*.

```
REVOKE EXECUTE ON FUNCTION DEPT_TOTALS
      FROM ADMIN_A RESTRICT;
```

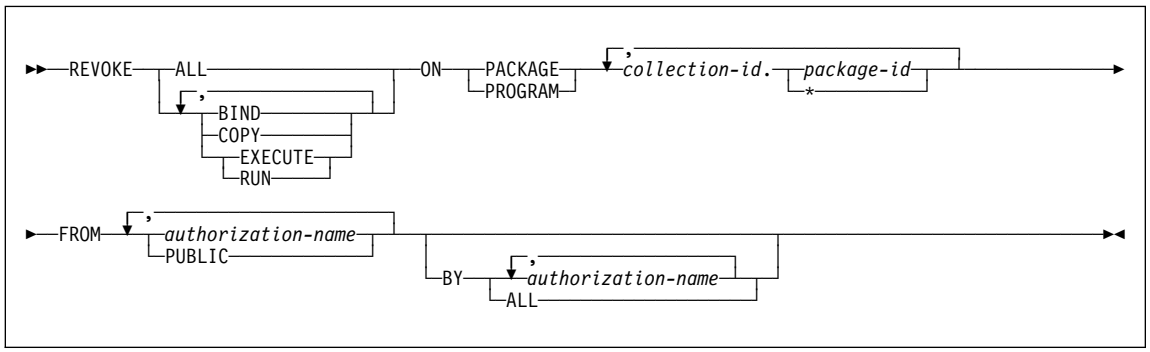
Example 4: Revoke the EXECUTE privilege on function NEW_DEPT_HIRES for HR (Human Resources). The function has two input parameters with data types of INTEGER and CHAR(10), respectively. Assume that the schema has more than one function that is named NEW_DEPT_HIRES.

```
REVOKE EXECUTE ON FUNCTION NEW_DEPT_HIRES (INTEGER, CHAR(10))
      FROM HR RESTRICT;
```

You can also code the CHAR(10) data type as CHAR().

REVOKE (package privileges)

Syntax



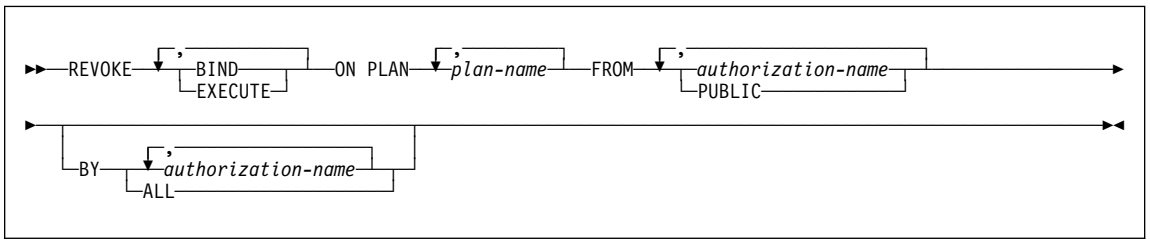
Example

Revoke the privilege to copy all packages in collection DSN8CC61 from LEWIS.

```
REVOKE COPY ON PACKAGE DSN8CC61.* FROM LEWIS;
```

REVOKE (plan privileges)

Syntax



Examples

Example 1: Revoke authority to bind plan DSN8IP61 from user JONES.

```
REVOKE BIND ON PLAN DSN8IP61 FROM JONES;
```

Example 2: Revoke authority previously granted to all users at the current server to bind and execute plan DSN8CP61. (Grants to specific users will not be affected.)

```
REVOKE BIND,EXECUTE ON PLAN DSN8CP61 FROM PUBLIC;
```

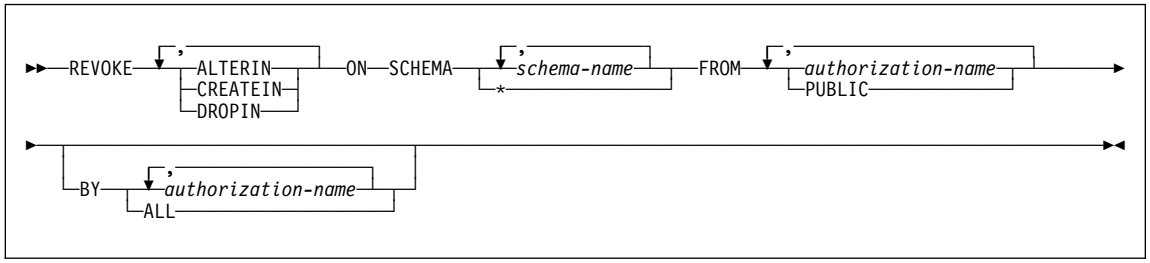
Example 3: Revoke authority to execute plan DSN8CP61 from users ADAMSON and BROWN.

```
REVOKE EXECUTE ON PLAN DSN8CP61 FROM ADAMSON,BROWN;
```

REVOKE (schema privileges)

This form of the REVOKE statement revokes privileges on schemas.

Syntax



Examples

Example 1: Revoke the CREATEIN privilege on schema T_SCORES from user JONES.

```
REVOKE CREATEIN ON SCHEMA T_SCORES FROM JONES;
```

Example 2: Revoke the CREATEIN privilege on schema VAC from all users at the current server.

```
REVOKE CREATEIN ON SCHEMA VAC FROM PUBLIC;
```

Example 3: Revoke the ALTERIN privilege on schema DEPT from the administrative assistant.

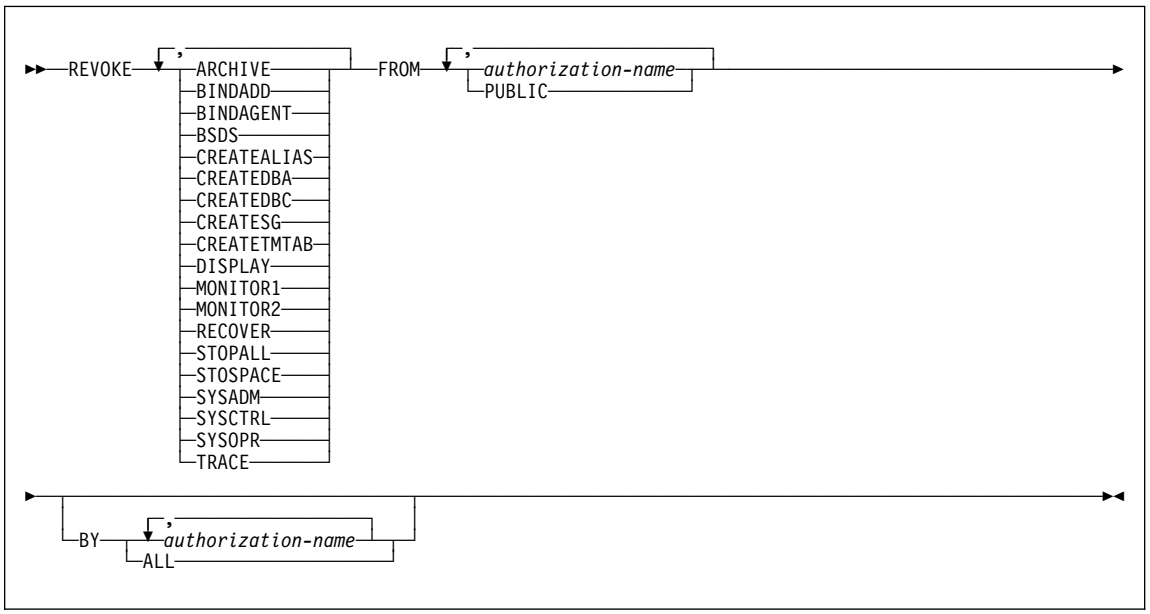
```
REVOKE ALTERIN ON SCHEMA DEPT FROM ADMIN_A;
```

Example 4: Revoke the ALTERIN and DROPIN privileges on schemas NEW_HIRE, PROMO, and RESIGN from HR (Human Resources).

```
REVOKE ALTERIN, DROPIN ON SCHEMA NEW_HIRE, PROMO, RESIGN FROM HR;
```

REVOKE (system privileges)

Syntax



Examples

Example 1: Revoke DISPLAY privileges from user LUTZ.

```
REVOKE DISPLAY
  FROM LUTZ;
```

Example 2: Revoke BSDS and RECOVER privileges from users PARKER and SETRIGHT.

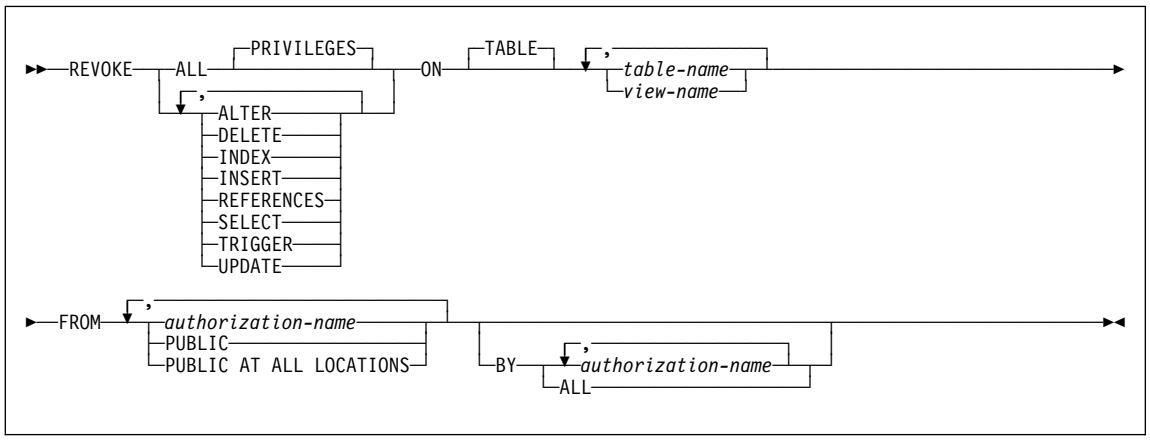
```
REVOKE BSDS,RECOVER
  FROM PARKER,SETRIGHT;
```

Example 3: Revoke TRACE privileges previously granted to all local users. (Grants to specific users will not be affected.)

```
REVOKE TRACE
  FROM PUBLIC;
```

REVOKE (table or view privileges)

Syntax



Examples

Example 1: Revoke SELECT privileges on table DSN8610.EMP from user PULASKI.

```
REVOKE SELECT ON TABLE DSN8610.EMP FROM PULASKI;
```

Example 2: Revoke update privileges on table DSN8610.EMP previously granted to all local DB2 users. (Grants to specific users are not affected.)

```
REVOKE UPDATE ON TABLE DSN8610.EMP FROM PUBLIC;
```

Example 3: Revoke all privileges on table DSN8610.EMP from users KWAN and THOMPSON.

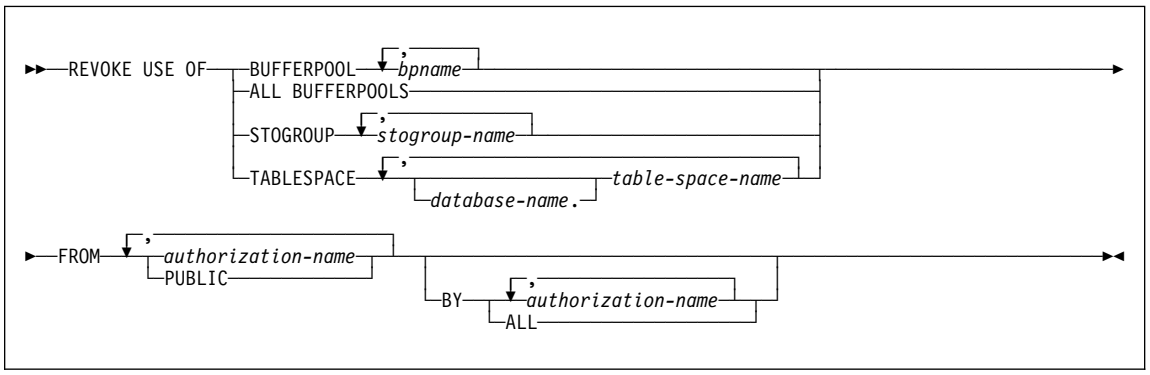
```
REVOKE ALL ON TABLE DSN8610.EMP FROM KWAN,THOMPSON;
```

Example 4: Revoke the grant of SELECT and UPDATE privileges on the table DSN8610.DEPT to every user in the network. Doing so does not affect users who obtained these privileges from some other grant.

```
REVOKE SELECT, UPDATE ON TABLE DSN8610.DEPT
  FROM PUBLIC AT ALL LOCATIONS;
```

REVOKE (use privileges)

Syntax



Examples

Example 1: Revoke authority to use buffer pool BP2 from user MARINO.

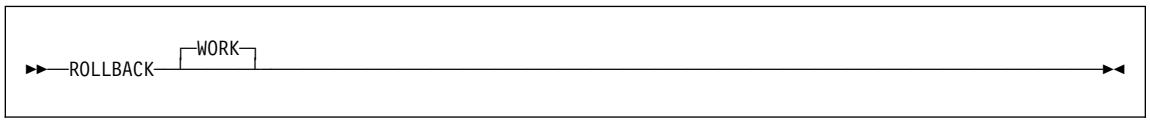
```
REVOKE USE OF BUFFERPOOL BP2
  FROM MARINO;
```

Example 2: Revoke a grant of the USE privilege on the table space DSN8S61D in the database DSN8D61A. The grant is to PUBLIC, that is, to everyone at the local DB2 subsystem. (Grants to specific users are not affected.)

```
REVOKE USE OF TABLESPACE DSN8D61A.DSN8S61D
  FROM PUBLIC;
```

ROLLBACK

Syntax



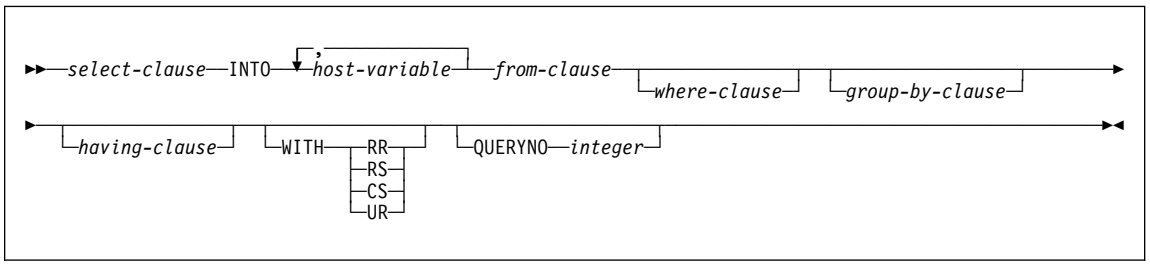
Example

Roll back all DB2 database changes made since the unit of recovery was started.

```
ROLLBACK WORK;
```

SELECT INTO

Syntax



Examples

Example 1: Put the maximum salary in DSN8610.EMP into the host variable MAXSALRY.

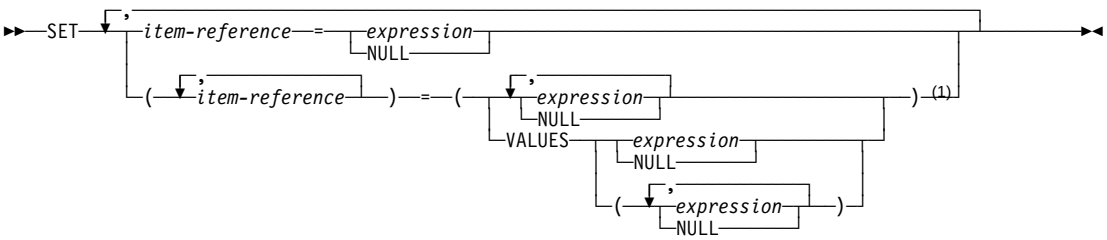
```
EXEC SQL SELECT MAX(SALARY)
  INTO :MAXSALRY
  FROM DSN8610.EMP;
```

Example 2: Put the row for employee 528671, from DSN8610.EMP, into the host structure EMPREC.

```
EXEC SQL SELECT * INTO :EMPREC
  FROM DSN8610.EMP
  WHERE EMPNO = '528671'
END-EXEC.
```

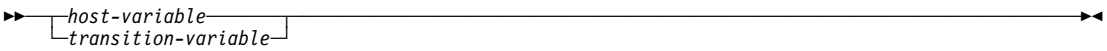
SET Assignment

Syntax



Note:
1 The number of *expression*s and NULL keywords must match the number of *item-reference*s.

item-reference:



transition-variable:

Examples

Example 1: Set the host variable HVL to the value of the CURRENT PATH special register.

```
SET :HVL = CURRENT PATH;
```

Example 2: Set the host variable SERVER to the name of the current server and the host variable XTIME to the local time at the current server.

```
SET :SERVER = CURRENT SERVER,
    :XTIME = CURRENT TIME;
```

Example 3: Set the host variable DETAILS to a portion of a LOB value, using a LOB expression with a LOB locator to refer the extracted portion of the value.

```
SET :DETAILS = SUBSTR(:LOCATOR,1,35);
```

Example 4: Set host variable HV1 to the results of external function CALC_SALARY and host variable HV2 to the value of special register CURRENT PATH. Use an indicator value with HV1 in case CALC_SALARY returns a null value.

```
SET (:HV1:IND1, :HV2) =
    (CALC_SALARY(:HV3, :HF4), CURRENT PATH);
```

Example 5: Assume that you want to create a before trigger that sets the salary and commission columns to default values for newly inserted rows in the EMPLOYEE table and that you will define the trigger only with NEW in the REFERENCING clause. Write the SET statement that assigns the default values to the SALARY and COMMISSION columns.

```
SET (SALARY, COMMISSION) = (50000, 8000);
```

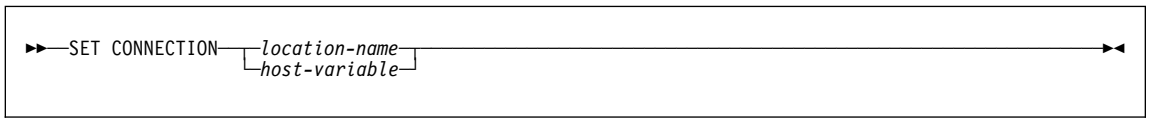
Example 6: Assume that you want to create a before trigger that detects any commission increases greater than 10% for updated rows in the EMPLOYEE table and limits the commission increase to 10%. You will define the trigger with both OLD and NEW in the REFERENCING clause. Write the SET statement that limits an increase to the COMMISSION column to 10% .

```
SET NEWROW.COMMISSION = 1.1 * OLDROW.COMMISSION;
```

SET CONNECTION

SET CONNECTION

Syntax



Example

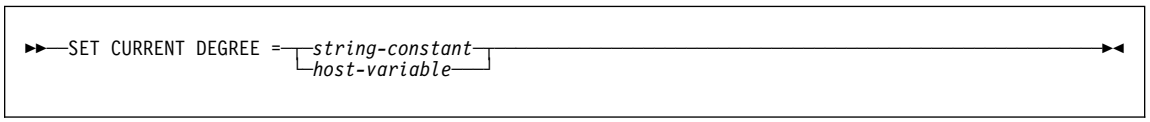
Execute SQL statements at TOROLAB1, execute SQL statements at TOROLAB2, and then execute more SQL statements at TOROLAB1.

```
EXEC SQL CONNECT TO TOROLAB1;

  (execute statements referencing objects at TOROLAB1)

EXEC SQL CONNECT TO TOROLAB2;

  (execute statements referencing objects at TOROLAB2)

EXEC SQL SET CONNECTION TOROLAB1;

  (execute statements referencing objects at TOROLAB1)
```

The first CONNECT statement creates the TOROLAB1 connection, the second CONNECT statement places it in the dormant state, and the SET CONNECTION statement returns it to the current state.

SET CURRENT DEGREE

Syntax



Examples

Example 1: The following statement inhibits parallel operations:

```
SET CURRENT DEGREE = '1';
```

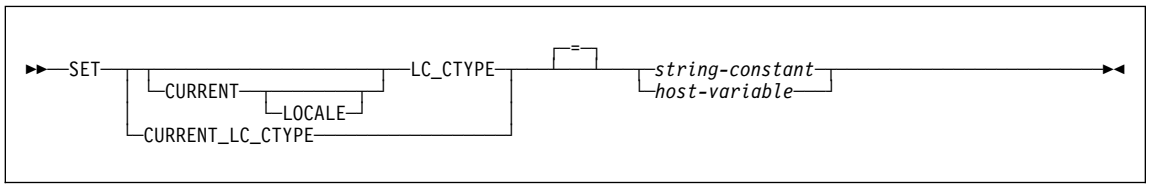
Example 2: The following statement allows parallel operations:

```
SET CURRENT DEGREE = 'ANY';
```

SET CURRENT LOCALE LC_CTYPE

SET CURRENT LOCALE LC_CTYPE

Syntax



Examples

Example 1: Set the CURRENT LOCALE LC_CTYPE special register to the locale 'En_US'.

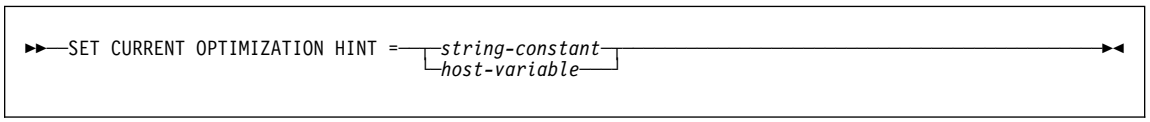
```
EXEC SQL SET CURRENT LOCALE LC_CTYPE = 'En_US';
```

Example 2: Set the CURRENT LOCALE LC_CTYPE special register to the value of host variable HV1, which contains 'Fr_FR@EURO'.

```
EXEC SQL SET CURRENT LOCALE LC_CTYPE = :HV1;
```

SET CURRENT OPTIMIZATION HINT

Syntax



Example

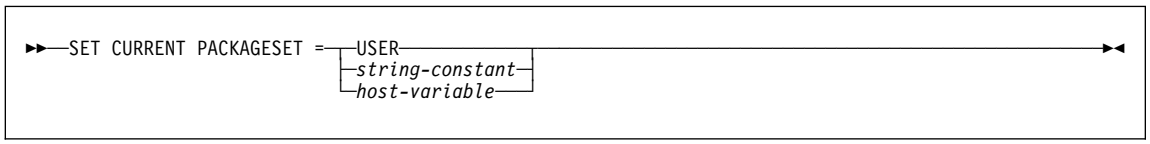
Assume that delimited identifier 'NOHYB' identifies a user-defined optimization hint in authid.PLAN_TABLE. Set the CURRENT OPTIMIZATION HINT special register so that DB2 uses this optimization hint to generate the access path for dynamic statements.

```
SET CURRENT OPTIMIZATION HINT = 'NOHYB'
```

SET CURRENT PACKAGESET

SET CURRENT PACKAGESET

Syntax



Examples

Example 1: Limit the plan element selection to packages in the PERSONNEL collection at the current server.

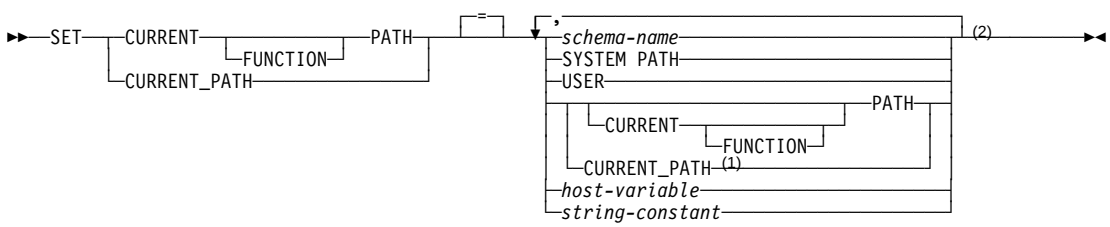
```
EXEC SQL SET CURRENT PACKAGESET = 'PERSONNEL';
```

Example 2: Eliminate collections as a factor in plan element selection.

```
EXEC SQL SET CURRENT PACKAGESET = '';
```

SET CURRENT PATH

Syntax



Notes:

1 This form is consistent with SQL standard name of the special register.

2 SYSTEM PATH, USER and CURRENT PATH can be specified only once each.

Examples

Example 1: Set the CURRENT PATH special register to the list of schemas: "SCHEMA1", "SCHEMA#2", "SYSIBM".

```
SET CURRENT PATH = SCHEMA1,"SCHEMA#2", SYSIBM;
```

When the special register provides the SQL path, SYSPROC which was not explicitly specified in the special register, is implicitly assumed at the front of the SQL path, making the effective value of the path:

```
SYSPROC, SCHEMA1, SCHEMA#2, SYSIBM
```

Example 2: Add schema SMITH and SYSPROC to the value of the CURRENT PATH special register that was set in Example 1.

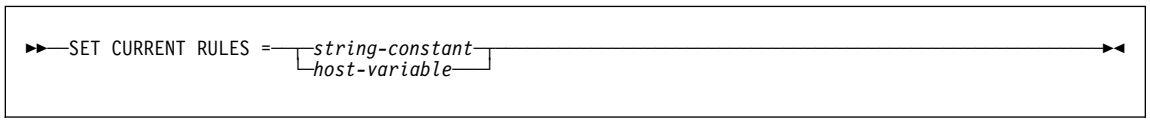
```
SET CURRENT PATH = CURRENT PATH, SMITH, SYSPROC;
```

The value of the special register becomes:

```
SCHEMA1, SCHEMA#2, SYSIBM, SMITH, SYSPROC
```

SET CURRENT RULES

Syntax



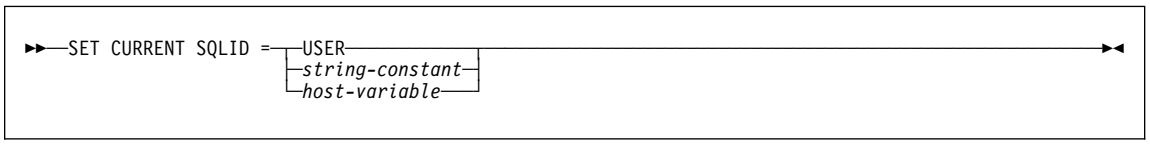
Example

Set the SQL rules to be followed to DB2.

```
EXEC SQL SET CURRENT RULES = 'DB2';
```

SET CURRENT SQLID

Syntax



Example

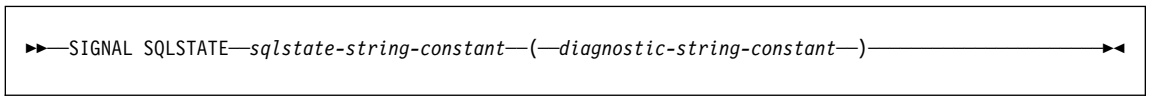
Set the CURRENT SQLID to the primary authorization ID.

```
SET CURRENT SQLID=USER;
```

SIGNAL SQLSTATE

SIGNAL SQLSTATE

Syntax



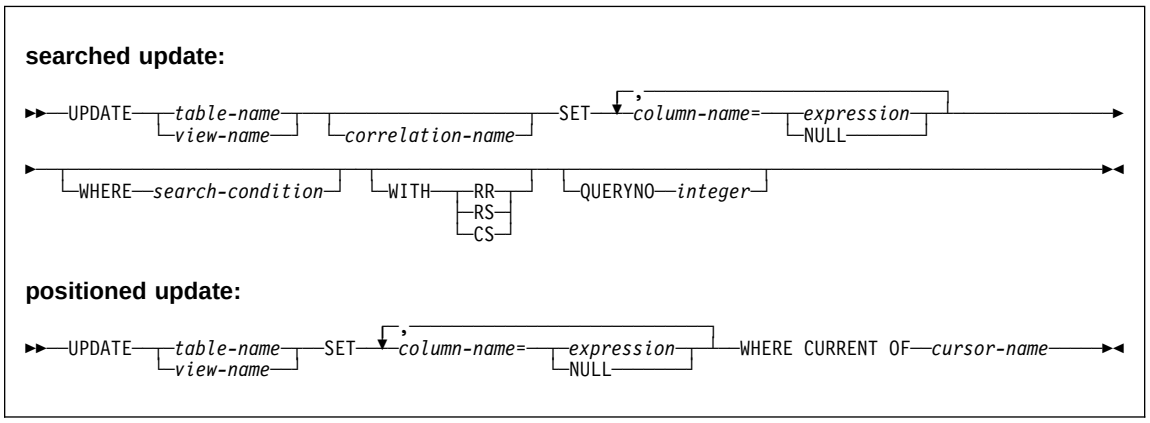
Example

Consider a trigger for an order system that allows orders to be recorded in an ORDERS table (ORDERNO, CUSTNO, PARTNO, QUANTITY) only if there is sufficient stock in the PARTS tables. When there is insufficient stock for an order, SQLSTATE '75001' is returned along with an appropriate error description.

```
CREATE TRIGGER CK_AVAIL
      NO CASCADE BEFORE INSERT ON ORDERS
      REFERENCING NEW AS NEW_ORDER
      FOR EACH ROW MODE DB2SQL
      WHEN (NEW_ORDER.QUANTITY > (SELECT ON_HAND FROM PARTS
                                  WHERE NEW_ORDER.PARTNO = PARTS.PARTNO))
        BEGIN ATOMIC
          SIGNAL SQLSTATE '75001' ('Insufficient stock for order');
        END
```

UPDATE

Syntax



Examples

Example 1: Change employee 000190's telephone number to 3565 in DSN8610.EMP.

```
EXEC UPDATE DSN8610.EMP
  SET PHONENO='3565'
  WHERE EMPNO='000190';
```

Example 2: Give each member of department D11 a 100-dollar raise.

```
EXEC UPDATE DSN8610.EMP
  SET SALARY = SALARY + 100
  WHERE WORKDEPT = 'D11';
```

Example 3: Employee 000250 is going on a leave of absence. Set the salary to null.

```
EXEC UPDATE DSN8610.EMP
  SET SALARY = NULL
  WHERE EMPNO='000250';
```

Example 4: Double the salary of the employee represented by the row on which the cursor C1 is positioned.

```
EXEC SQL UPDATE DSN8610.EMP
  SET SALARY = 2 * SALARY
  WHERE CURRENT OF C1;
```

Example 5: Assume that employee table EMP1 was created with the following statement:

UPDATE

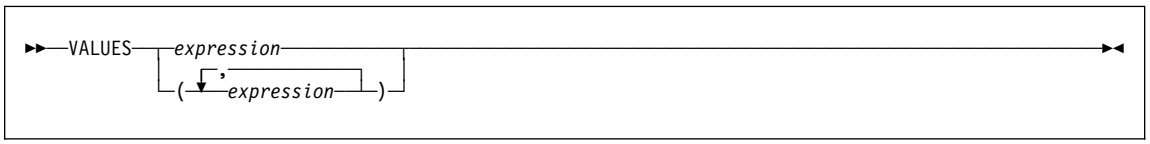
```
CREATE TABLE EMP1
  (EMP_ROWID    ROWID GENERATED ALWAYS,
   EMPNO        CHAR(6),
   NAME         CHAR(30),
   SALARY       DECIMAL(9,2),
   PICTURE      BLOB(250K),
   RESUME       CLOB(32K));
```

Assume that host variable HV_EMP_ROWID contains the value of the ROWID column for employee with employee number '350000'. Using that ROWID value to identify the employee and user-defined function UPDATE_RESUME, increase the employee's salary by $1000 and update that employee's resume.

```
EXEC SQL UPDATE EMP1
  SET SALARY = SALARY + 1000,
      RESUME = UPDATE_RESUME(:HV_RESUME)
  WHERE EMP_ROWID = :HV_EMP_ROWID;
```

VALUES

Syntax



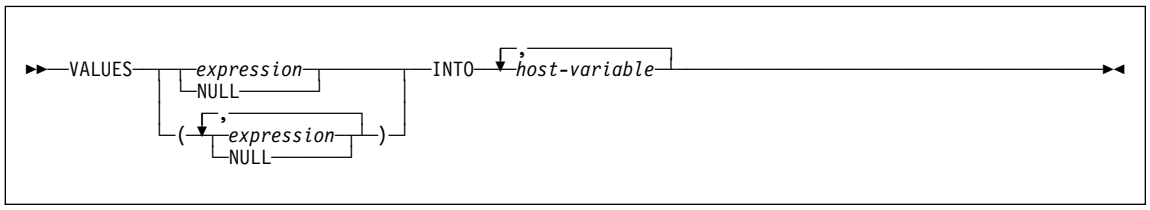
Example

Example: Create an after trigger EMPISRT1 that invokes user-defined function NEWEMP when the trigger is activated. An insert operation on table EMP activates the trigger. Pass transition variables for the new employee number, last name, and first name to the user-defined function.

```
CREATE TRIGGER EMPISRT1
   AFTER INSERT ON EMP
   REFERENCING NEW AS N
   FOR EACH ROW
   MODE DB2SQL
   BEGIN ATOMIC
      VALUES(NEWEMP(N.EMPNO, N.LASTNAME, N.FIRSTNAME));
   END
```

VALUES INTO

Syntax



Examples

Example 1: Assign the value of the CURRENT PATH special register to host variable HV1.

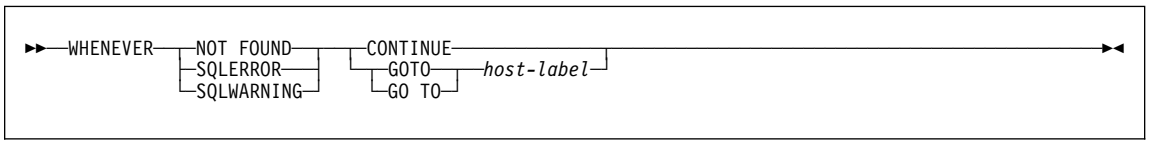
```
EXEC SQL VALUES(CURRENT PATH)
         INTO :HV1;
```

Example 2: Assume that LOB locator LOB1 is associated with a CLOB value. Assign a portion of the CLOB value to host variable DETAILS using the LOB locator.

```
EXEC SQL VALUES (SUBSTR(:LOB1,1,35))
         INTO :DETAILS;
```

WHENEVER

Syntax



Examples

The following statements can be embedded in a COBOL program.

Example 1: Go to the label HANDLER for any statement that produces an error.

```
EXEC SQL WHENEVER SQLERROR GOTO HANDLER END-EXEC.
```

Example 2: Continue processing for any statement that produces a warning.

```
EXEC SQL WHENEVER SQLWARNING CONTINUE END-EXEC.
```

Example 3: Go to the label ENDDATA for any statement that does not return.

```
EXEC SQL WHENEVER NOT FOUND GO TO ENDDATA END-EXEC.
```

WHENEVER

SQL return codes

This section lists possible SQL return codes generated by attempting to execute SQL statements. For more information on the meaning of these codes, see *DB2 Messages and Codes*.

SQL return codes

Successful execution SQL code

000	SUCCESSFUL EXECUTION

Warning SQL codes

+012	THE UNQUALIFIED COLUMN NAME *column-name* WAS INTERPRETED AS A CORRELATED REFERENCE
+098	A DYNAMIC SQL STATEMENT ENDS WITH A SEMICOLON.
+100	ROW NOT FOUND FOR FETCH, UPDATE OR DELETE, OR THE RESULT OF A QUERY IS AN EMPTY TABLE
+110	SQL UPDATE TO A DATA CAPTURE TABLE NOT SIGNALED TO ORIGINATING SUBSYSTEM
+111	THE SUBPAGES OPTION IS NOT SUPPORTED FOR TYPE 2 INDEXES
+117	THE NUMBER OF INSERT VALUES IS NOT THE SAME AS THE NUMBER OF OBJECT COLUMNS
+162	TABLESPACE *database-name.tablespace-name* HAS BEEN PLACED IN CHECK PENDING
+203	THE QUALIFIED COLUMN NAME *column-name* WAS RESOLVED USING A NON-UNIQUE OR UNEXPOSED NAME
+204	*name* IS AN UNDEFINED NAME
+206	*column-name* IS NOT A COLUMN OF AN INSERTED TABLE, UPDATED TABLE, OR ANY TABLE IDENTIFIED IN A FROM CLAUSE
+218	THE SQL STATEMENT REFERENCING A REMOTE OBJECT CANNOT BE EXPLAINED
+219	THE REQUIRED EXPLANATION TABLE *table-name* DOES NOT EXIST
+220	THE COLUMN *column-name* IN EXPLANATION TABLE *table-name* IS NOT DEFINED PROPERLY
+236	SQLDA INCLUDES *integer1* SQLVAR ENTRIES, BUT *integer2* ARE REQUIRED FOR *integer3* COLUMNS
+237	SQLDA INCLUDES *integer1* SQLVAR ENTRIES, BUT *integer2* ARE REQUIRED BECAUSE AT LEAST ONE OF THE COLUMNS BEING DESCRIBED IS A DISTINCT TYPE
+238	SQLDA INCLUDES *integer1* SQLVAR ENTRIES, BUT *integer2* SQLVAR ENTRIES ARE NEEDED FOR *integer3* COLUMNS BECAUSE AT LEAST ONE OF THE COLUMNS BEING DESCRIBED IS A LOB
+239	SQLDA INCLUDES *integer1* SQLVAR ENTRIES, BUT *integer2* ARE REQUIRED FOR *integer3* COLUMNS BECAUSE AT LEAST ONE OF THE COLUMNS BEING DESCRIBED IS A DISTINCT TYPE

+304 A VALUE WITH DATA TYPE *data-type1* CANNOT BE ASSIGNED TO A HOST VARIABLE BECAUSE THE VALUE IS NOT WITHIN THE RANGE OF THE HOST VARIABLE IN POSITION *position-number* WITH DATA TYPE *data-type2*

+331 THE NULL VALUE HAS BEEN ASSIGNED TO A HOST VARIABLE BECAUSE THE STRING CANNOT BE TRANSLATED. REASON *reason-code*, CHARACTER

+339 THE SQL STATEMENT HAS BEEN SUCCESSFULLY EXECUTED, BUT THERE MAY BE SOME CHARACTER CONVERSION INCONSISTENCIES

+394 USER SPECIFIED OPTIMIZATION HINTS USED DURING ACCESS PATH SELECTION

+395 USER SPECIFIED OPTIMIZATION HINTS ARE INVALID (REASON CODE = 'reason-code'). THE OPTIMIZATION HINTS ARE IGNORED.

+402 LOCATION *location* IS UNKNOWN

+403 THE LOCAL OBJECT REFERENCED BY THE CREATE ALIAS STATEMENT DOES NOT EXIST

+434 OPTION *keyword* IS A DEPRECATED FEATURE

+445 VALUE *value* HAS BEEN TRUNCATED

+462 EXTERNAL FUNCTION OR PROCEDURE *name* (SPECIFIC NAME

+464 PROCEDURE *proc* RETURNED *num* QUERY RESULT SETS, WHICH EXCEEDS THE DEFINED LIMIT *integer*

+466 CREATE PROCEDURE *proc* RETURNED *num* QUERY RESULTS SETS

+494 NUMBER OF RESULT SETS IS GREATER THAN NUMBER OF LOCATORS

+495 ESTIMATED PROCESSOR COST OF *estimate-amount1* PROCESSOR SECONDS (*estimate-amount2* SERVICE UNITS) IN COST CATEGORY *cost-category* EXCEEDS A RESOURCE LIMIT WARNING THRESHOLD OF *limit- amount* SERVICE UNITS

+535 THE RESULT OF THE POSITIONED UPDATE OR DELETE MAY DEPEND ON THE ORDER OF THE ROWS

+541 THE REFERENTIAL OR UNIQUE CONSTRAINT *name* HAS BEEN IGNORED BECAUSE IT IS A DUPLICATE

+551 *auth-id* DOES NOT HAVE THE PRIVILEGE TO PERFORM OPERATION

+552 *auth-id* DOES NOT HAVE THE PRIVILEGE TO PERFORM OPERATION

+558 THE WITH GRANT OPTION IS IGNORED

+561 THE ALTER, INDEX, REFERENCES, AND TRIGGER PRIVILEGES CANNOT BE GRANTED PUBLIC AT ALL LOCATIONS

+562 A GRANT OF A PRIVILEGE WAS IGNORED BECAUSE THE GRANTEE ALREADY HAS THE PRIVILEGE FROM THE GRANTOR

SQL return codes

+585	THE SCHEMA NAME *schema-name* APPEARS MORE THAN ONCE IN THE CURRENT PATH
+599	COMPARISON FUNCTIONS ARE NOT CREATED FOR A DISTINCT TYPE BASED ON A LONG STRING DATA TYPE
+610	A CREATE/ALTER ON OBJECT *object-name* HAS PLACED OBJECT IN
+625	THE DEFINITION OF TABLE *table-name* HAS BEEN CHANGED TO INCOMPLETE
+626	DROPPING THE INDEX TERMINATES ENFORCEMENT OF THE UNIQUENESS OF A KEY THAT WAS DEFINED WHEN THE TABLE WAS CREATED
+645	WHERE NOT NULL IS IGNORED BECAUSE THE INDEX KEY CANNOT CONTAIN NULL VALUES
+650	THE TABLE BEING CREATED OR ALTERED CANNOT BECOME A DEPENDENT TABLE
+653	TABLE *table-name* IN PARTITIONED TABLESPACE *tspace-name* IS NOT AVAILABLE BECAUSE ITS PARTITIONED INDEX HAS NOT BEEN CREATED
+655	STOGROUP stogroup_name HAS BOTH SPECIFIC AND NON-SPECIFIC VOLUME IDS. IT WILL NOT BE ALLOWED IN FUTURE RELEASES
+658	THE SUBPAGES VALUE IS IGNORED FOR THE CATALOG INDEX *index-name*
+664	THE INTERNAL LENGTH OF THE LIMIT-KEY FIELDS FOR THE PARTITIONED INDEX
+738	DEFINITION CHANGE OF *object object_name* MAY REQUIRE SIMILAR CHANGE ON READ-ONLY SYSTEMS
+799	A SET STATEMENT REFERENCES A SPECIAL REGISTER THAT DOES NOT EXIST AT THE SERVER SITE
+802	EXCEPTION ERROR *exception-type* HAS OCCURRED DURING
+806	BIND ISOLATION LEVEL RR CONFLICTS WITH TABLESPACE LOCKSIZE PAGE OR LOCKSIZE ROW AND LOCKMAX 0
+807	THE RESULT OF DECIMAL MULTIPLICATION MAY CAUSE OVERFLOW
+863	THE CONNECTION WAS SUCCESSFUL BUT ONLY SBCS WILL BE SUPPORTED
+2000	TYPE 1 INDEXES WITH SUBPAGES GREATER THAN 1 CANNOT BECOME GROUP BUFFER POOL DEPENDENT IN A DATA SHARING ENVIRONMENT
+20002	THE GBPCACHE SPECIFICATION IS IGNORED, *bpname* DOES NOT ALLOW CACHING

+20007 USE OF OPTIMIZATION HINTS IS DISALLOWED BY A DB2 SUBSYSTEM PARAMETER. THE SPECIAL REGISTER 'OPTIMIZATION HINT' IS SET TO THE DEFAULT VALUE OF BLANKS.

+30100 OPERATION COMPLETED SUCCESSFULLY BUT A DISTRIBUTION PROTOCOL VIOLATION HAS BEEN DETECTED. ORIGINAL SQLCODE=*original-sqlcode* AND ORIGINAL SQLSTATE=*original-sqlstate*

Error SQL codes

-007 STATEMENT CONTAINS THE ILLEGAL CHARACTER *character*

-010 THE STRING CONSTANT BEGINNING *string* IS NOT TERMINATED

-029 INTO CLAUSE REQUIRED

-060 INVALID *type* SPECIFICATION : *spec*

-084 UNACCEPTABLE SQL STATEMENT

-097 THE USE OF LONG VARCHAR OR LONG VARGRAPHIC IS NOT ALLOWED IN THIS CONTEXT

-101 THE STATEMENT IS TOO LONG OR TOO COMPLEX

-102 LITERAL STRING IS TOO LONG. STRING BEGINS *string*

-103 *literal* IS AN INVALID NUMERIC LITERAL

-104 ILLEGAL SYMBOL "*token*". SOME SYMBOLS THAT MIGHT BE LEGAL ARE: *token-list*

-105 INVALID STRING

-107 THE NAME *name* IS TOO LONG. MAXIMUM ALLOWABLE SIZE IS

-108 THE NAME *name* IS QUALIFIED INCORRECTLY

-109 *clause* CLAUSE IS NOT PERMITTED

-110 INVALID HEXADECIMAL LITERAL BEGINNING *string*

-111 A COLUMN FUNCTION DOES NOT INCLUDE A COLUMN NAME

-112 THE OPERAND OF A COLUMN FUNCTION IS ANOTHER COLUMN FUNCTION

-113 INVALID CHARACTER FOUND IN *string*, REASON CODE *nnn*

-114 THE LOCATION NAME *location* DOES NOT MATCH THE CURRENT SERVER

-115 A PREDICATE IS INVALID BECAUSE THE COMPARISON OPERATOR operator IS FOLLOWED BY A PARENTHESIZED LIST OR BY ANY OR ALL WITHOUT A SUBQUERY

-117 THE NUMBER OF INSERT VALUES IS NOT THE SAME AS THE NUMBER OF OBJECT COLUMNS

SQL return codes

-118 THE OBJECT TABLE OR VIEW OF THE DELETE OR UPDATE STATEMENT IS ALSO IDENTIFIED IN A FROM CLAUSE

-119 A COLUMN IDENTIFIED IN A HAVING CLAUSE IS NOT INCLUDED IN THE GROUP BY CLAUSE

-120 A WHERE CLAUSE, SET CLAUSE, VALUES CLAUSE, OR A SET ASSIGNMENT STATEMENT INCLUDES A COLUMN FUNCTION

-121 THE COLUMN *name* IS IDENTIFIED MORE THAN ONCE IN THE INSERT OR UPDATE OR SET TRANSITION VARIABLE STATEMENT

-122 A SELECT STATEMENT WITH NO GROUP BY CLAUSE CONTAINS A COLUMN NAME AND A COLUMN FUNCTION IN THE SELECT CLAUSE OR A COLUMN NAME IS CONTAINED IN THE SELECT CLAUSE BUT NOT IN THE GROUP BY CLAUSE

-123 THE PARAMETER IN POSITION *n* IN THE FUNCTION *name* MUST BE A CONSTANT OR KEYWORD

-125 AN INTEGER IN THE ORDER BY CLAUSE DOES NOT IDENTIFY A COLUMN OF THE RESULT

-126 THE SELECT STATEMENT CONTAINS BOTH AN UPDATE CLAUSE AND AN ORDER BY CLAUSE

-127 DISTINCT IS SPECIFIED MORE THAN ONCE IN A SUBSELECT

-128 INVALID USE OF NULL IN A PREDICATE

-129 THE STATEMENT CONTAINS TOO MANY TABLE NAMES

-130 THE ESCAPE CLAUSE CONSISTS OF MORE THAN ONE CHARACTER, OR THE STRING PATTERN CONTAINS AN INVALID OCCURRENCE OF THE ESCAPE CHARACTER

-131 STATEMENT WITH LIKE PREDICATE HAS INCOMPATIBLE DATA TYPES

-132 AN OPERAND OF *value* IS NOT VALID

-133 A COLUMN FUNCTION IN A SUBQUERY OF A HAVING CLAUSE IS INVALID BECAUSE ALL COLUMN REFERENCES IN ITS ARGUMENT ARE NOT CORRELATED TO THE GROUP BY RESULT THAT THE HAVING CLAUSE IS APPLIED TO

-134 IMPROPER USE OF LONG STRING COLUMN *column-name* OR AN EXPRESSION OF MAXIMUM LENGTH GREATER THAN 255

-136 SORT CANNOT BE EXECUTED BECAUSE THE SORT KEY LENGTH IS GREATER THAN 4000 BYTES

-137 THE LENGTH RESULTING FROM *operation* IS GREATER THAN *maximum-length*

-138 THE SECOND OR THIRD ARGUMENT OF THE SUBSTR FUNCTION IS OUT OF RANGE

-142 THE SQL STATEMENT IS NOT SUPPORTED

-144 INVALID SECTION NUMBER *number*

-147 ALTER FUNCTION *function-name* FAILED BECAUSE SOURCE FUNCTIONS CANNOT BE ALTERED

-148 THE SOURCE TABLE *source-name* CANNOT BE RENAMED OR ALTERED

-150 THE OBJECT OF THE INSERT, DELETE, OR UPDATE STATEMENT IS A VIEW OR TRANSITION TABLE FOR WHICH THE REQUESTED OPERATION IS NOT PERMITTED

-151 THE UPDATE STATEMENT IS INVALID BECAUSE THE CATALOG DESCRIPTION OF COLUMN *column-name* INDICATES THAT IT CANNOT BE UPDATED

-152 THE DROP *clause* CLAUSE IN THE ALTER STATEMENT IS INVALID BECAUSE *constraint-name* IS A *constraint-type*

-153 THE CREATE VIEW STATEMENT DOES NOT INCLUDE A REQUIRED COLUMN LIST

-154 THE CREATE VIEW FAILED BECAUSE THE VIEW DEFINITION CONTAINS A UNION, A UNION ALL, OR A REMOTE OBJECT

-156 THE STATEMENT DOES NOT IDENTIFY A TABLE

-157 ONLY A TABLE NAME CAN BE SPECIFIED IN A FOREIGN KEY CLAUSE.

-158 THE NUMBER OF COLUMNS SPECIFIED FOR THE VIEW IS NOT THE SAME AS THE NUMBER OF COLUMNS SPECIFIED BY THE SELECT CLAUSE, OR THE NUMBER OF COLUMNS SPECIFIED IN THE CORRELATION CLAUSE IN A FROM CLAUSE IS NOT THE SAME AS THE NUMBER OF COLUMNS IN THE CORRESPONDING TABLE, VIEW, TABLE EXPRESSION, OR TABLE FUNCTION

-159 DROP OR COMMENT ON *object* IDENTIFIES A(N) *object-type1* RATHER THAN A(N) *object-type2*

-160 THE WITH CHECK OPTION CANNOT BE USED FOR THE SPECIFIED VIEW

-161 THE INSERT OR UPDATE IS NOT ALLOWED BECAUSE A RESULTING ROW DOES NOT SATISFY THE VIEW DEFINITION

-164 *auth-id1* DOES NOT HAVE THE PRIVILEGE TO CREATE A VIEW WITH QUALIFICATION *authorization* ID

-170 THE NUMBER OF ARGUMENTS SPECIFIED FOR *function-name* IS INVALID

-171 THE DATA TYPE, LENGTH, OR VALUE OF ARGUMENT *nn* OF

-173 UR IS SPECIFIED ON THE WITH CLAUSE BUT THE CURSOR IS NOT READ-ONLY

SQL return codes

-180	THE DATE, TIME, OR TIMESTAMP VALUE *value* IS INVALID
-181	THE STRING REPRESENTATION OF A DATETIME VALUE IS NOT A VALID DATETIME VALUE
-182	AN ARITHMETIC EXPRESSION WITH A DATETIME VALUE IS INVALID
-183	AN ARITHMETIC OPERATION ON A DATE OR TIMESTAMP HAS A RESULT THAT IS NOT WITHIN THE VALID RANGE OF DATES
-184	AN ARITHMETIC EXPRESSION WITH A DATETIME VALUE CONTAINS A PARAMETER MARKER
-185	THE LOCAL FORMAT OPTION HAS BEEN USED WITH A DATE OR TIME AND NO LOCAL EXIT HAS BEEN INSTALLED
-186	THE LOCAL DATE LENGTH OR LOCAL TIME LENGTH HAS BEEN INCREASED AND EXECUTING PROGRAM RELIES ON THE OLD LENGTH
-187	A REFERENCE TO A CURRENT DATE/TIME SPECIAL REGISTER IS INVALID BECAUSE THE MVS TOD CLOCK IS BAD OR THE MVS PARMTZ IS OUT OF RANGE
-188	THE STRING REPRESENTATION OF A NAME IS INVALID
-189	CCSID *ccsid* IS UNKNOWN OR INVALID FOR THE DATA TYPE OR SUBTYPE
-190	ATTRIBUTES OF COLUMN *column-name* IN TABLE *table-name* ARE NOT COMPATIBLE WITH THE EXISTING COLUMN
-191	A STRING CANNOT BE USED BECAUSE IT IS INVALID MIXED DATA
-197	QUALIFIED COLUMN NAMES IN ORDER BY CLAUSE NOT PERMITTED WHEN UNION OR UNION ALL SPECIFIED
-198	THE OPERAND OF THE PREPARE OR EXECUTE IMMEDIATE STATEMENT IS BLANK OR EMPTY
-199	ILLEGAL USE OF KEYWORD *keyword*. TOKEN *token-list* WAS EXPECTED
-203	A REFERENCE TO COLUMN *column-name* IS AMBIGUOUS
-204	*name* IS AN UNDEFINED NAME
-205	*column-name* IS NOT A COLUMN OF TABLE *table-name*
-206	*column-name* IS NOT A COLUMN OF AN INSERTED TABLE, UPDATED TABLE, OR ANY TABLE IDENTIFIED IN A FROM CLAUSE, OR IS NOT A COLUMN OF THE TRIGGERING TABLE OF A TRIGGER
-208	THE ORDER BY CLAUSE IS INVALID BECAUSE COLUMN *name* IS NOT PART OF THE RESULT TABLE
-212	*name* IS SPECIFIED MORE THAN ONCE IN THE REFERENCING CLAUSE OF A TRIGGER DEFINITION

-214 An expression starting with *expression-start* in the

-219 THE REQUIRED EXPLANATION TABLE *table-name* DOES NOT EXIST

-220 THE COLUMN *column-name* IN EXPLANATION TABLE *table-name* IS NOT DEFINED PROPERLY

-221 “SET OF OPTIONAL COLUMNS” IN EXPLANATION TABLE*table-name* IS INCOMPLETE. OPTIONAL COLUMN *column-name* IS MISSING

-229 THE LOCALE *locale* SPECIFIED IN A SET LOCALE OR OTHER STATEMENT THAT IS LOCALE SENSITIVE WAS NOT FOUND

-240 THE PART CLAUSE OF A LOCK TABLE STATEMENT IS INVALID

-250 THE LOCAL LOCATION NAME IS NOT DEFINED WHEN PROCESSING A THREE-PART OBJECT NAME

-251 TOKEN *name* IS NOT VALID

-300 THE STRING CONTAINED IN HOST VARIABLE OR PARAMETER

-301 THE VALUE OF INPUT HOST VARIABLE OR PARAMETER NUMBER

-302 THE VALUE OF INPUT VARIABLE OR PARAMETER NUMBER *position-number* IS INVALID OR TOO LARGE FOR THE TARGET COLUMN OR THE TARGET VALUE

-303 A VALUE CANNOT BE ASSIGNED TO OUTPUT HOST VARIABLE NUMBER

-304 A VALUE WITH DATA TYPE *data-type1* CANNOT BE ASSIGNED TO A HOST VARIABLE BECAUSE THE VALUE IS NOT WITHIN THE RANGE OF THE HOST VARIABLE IN POSITION *position-number* WITH DATA TYPE *data-type2*

-305 THE NULL VALUE CANNOT BE ASSIGNED TO OUTPUT HOST VARIABLE NUMBER

-309 A PREDICATE IS INVALID BECAUSE A REFERENCED HOST VARIABLE HAS THE NULL VALUE

-310 DECIMAL HOST VARIABLE OR PARAMETER *number* CONTAINS NON-DECIMAL DATA

-311 THE LENGTH OF INPUT HOST VARIABLE NUMBER *position-number* IS NEGATIVE OR GREATER THAN THE MAXIMUM

-312 *variable-name* IS AN UNDEFINED OR UNUSABLE HOST VARIABLE OR IS USED IN A DYNAMIC SQL STATEMENT OR A TRIGGER DEFINITION

-313 THE NUMBER OF HOST VARIABLES SPECIFIED IS NOT EQUAL TO THE NUMBER OF PARAMETER MARKERS

-314 THE STATEMENT CONTAINS AN AMBIGUOUS HOST VARIABLE REFERENCE

-327 THE ROW CANNOT BE INSERTED BECAUSE IT IS OUTSIDE THE BOUND OF THE PARTITION RANGE FOR THE LAST PARTITION

SQL return codes

-330 A STRING CANNOT BE USED BECAUSE IT CANNOT BE TRANSLATED. REASON

-331 A STRING CANNOT BE ASSIGNED TO A HOST VARIABLE BECAUSE IT CANNOT BE TRANSLATED. REASON *reason-code*, CHARACTER *code-point*, POSITION *position-number*

-332 SYSSTRINGS DOES NOT DEFINE A TRANSLATION FROM CCSID *ccsid* TO *ccsid*

-333 THE SUBTYPE OF A STRING VARIABLE IS NOT THE SAME AS THE SUBTYPE KNOWN AT BIND TIME AND THE DIFFERENCE CANNOT BE RESOLVED BY TRANSLATION

-338 AN ON CLAUSE IS INVALID

-339 THE SQL STATEMENT CANNOT BE EXECUTED FROM AN ASCII BASED DRDA APPLICATION REQUESTOR TO A V2R2 DB2 SUBSYSTEM

-350 INVALID SPECIFICATION OF A LARGE OBJECT COLUMN

-351 AN UNSUPPORTED SQLTYPE WAS ENCOUNTERED IN POSITION

-352 AN UNSUPPORTED SQLTYPE WAS ENCOUNTERED IN POSITION

-355 A LOB COLUMN IS TOO LARGE TO BE LOGGED

-372 ONLY ONE COLUMN DEFINED AS ROWID IS ALLOWED IN A TABLE

-390 THE FUNCTION *function-name*, SPECIFIC NAME

-392 SQLDA PROVIDED FOR CURSOR *cursor* HAS BEEN CHANGED FROM THE PREVIOUS FETCH

-396 *object-type object-name* ATTEMPTED TO EXECUTE AN SQL STATEMENT DURING FINAL CALL PROCESSING

-397 THE OPTION GENERATED IS SPECIFIED WITH A COLUMN THAT IS NOT A ROW ID OR DISTINCT TYPE BASED ON A ROW ID

-397 THE OPTION GENERATED IS SPECIFIED WITH A COLUMN THAT IS NOT A ROW ID OR DISTINCT TYPE BASED ON A ROW ID

-398 A LOCATOR WAS REQUESTED FOR HOST VARIABLE NUMBER *position-number* BUT THE VARIABLE IS NOT A LOB

-399 ATTEMPTED TO INSERT AN INVALID VALUE INTO A ROWID COLUMN

-400 THE CATALOG HAS THE MAXIMUM NUMBER OF USER DEFINED INDEXES

-401 THE OPERANDS OF AN ARITHMETIC OR COMPARISON OPERATION ARE NOT COMPARABLE

-402 AN ARITHMETIC FUNCTION OR OPERATOR *arith-fop* IS APPLIED TO CHARACTER OR DATETIME DATA

-404 THE SQL STATEMENT SPECIFIES A STRING THAT IS TOO LONG

-405 THE NUMERIC LITERAL literal CANNOT BE USED AS SPECIFIED BECAUSE IT IS OUT OF RANGE

-406 A CALCULATED OR DERIVED NUMERIC VALUE IS NOT WITHIN THE RANGE OF ITS OBJECT COLUMN

-407 AN UPDATE, INSERT, OR SET VALUE IS NULL, BUT THE OBJECT COLUMN *column-name* CANNOT CONTAIN NULL VALUES

-408 THE VALUE IS NOT COMPATIBLE WITH THE DATA TYPE OF ITS TARGET

-409 INVALID OPERAND OF A COUNT FUNCTION

-410 THE FLOATING POINT LITERAL *literal* CONTAINS MORE THAN 30 CHARACTERS

-411 CURRENT SQLID CANNOT BE USED IN A STATEMENT THAT REFERENCES REMOTE OBJECTS

-412 THE SELECT CLAUSE OF A SUBQUERY SPECIFIES MULTIPLE COLUMNS

-413 OVERFLOW OCCURRED DURING NUMERIC DATA TYPE CONVERSION

-414 A LIKE PREDICATE IS INVALID BECAUSE THE FIRST OPERAND IS NOT A STRING

-415 THE CORRESPONDING COLUMNS, *column-number*, OF THE OPERANDS OF A UNION OR A UNION ALL DO NOT HAVE COMPARABLE COLUMN DESCRIPTIONS

-416 AN OPERAND OF A UNION CONTAINS A LONG STRING COLUMN

-417 A STATEMENT STRING TO BE PREPARED INCLUDES PARAMETER MARKERS AS THE OPERANDS OF THE SAME OPERATOR

-418 A STATEMENT STRING TO BE PREPARED CONTAINS AN INVALID USE OF PARAMETER MARKERS

-419 THE DECIMAL DIVIDE OPERATION IS INVALID BECAUSE THE RESULT WOULD HAVE A NEGATIVE SCALE

-420 THE VALUE OF A CHARACTER STRING ARGUMENT WAS NOT ACCEPTABLE TO THE

-421 THE OPERANDS OF A UNION OR UNION ALL DO NOT HAVE THE SAME NUMBER OF COLUMNS

-423 INVALID VALUE FOR LOCATOR IN POSITION *position-#*

-426 DYNAMIC COMMIT NOT VALID AT AN APPLICATION SERVER WHERE UPDATES ARE NOT ALLOWED

-427 DYNAMIC ROLLBACK NOT VALID AT AN APPLICATION SERVER WHERE UPDATES ARE NOT ALLOWED

-430 *routine-type routine-name* (SPECIFIC NAME

SQL return codes

-433	VALUE *value* IS TOO LONG
-435	AN INVALID SQLSTATE *sqlstate* IS SPECIFIED IN THE FUNCTION RAISE_ERROR OR IN A SIGNAL SQLSTATE STATEMENT
-438	APPLICATION RAISED ERROR WITH DIAGNOSTIC TEXT: *text*
-440	NO *routine-type* BY THE NAME *routine-name* HAVING COMPATIBLE ARGUMENTS WAS FOUND
-441	INVALID USE OF 'DISTINCT' OR 'ALL' WITH SCALAR FUNCTION *function-name*
-443	EXTERNAL FUNCTION *function-name* (SPECIFIC NAME
-444	USER PROGRAM *name* COULD NOT BE FOUND
-449	CREATE OR ALTER STATEMENT FOR FUNCTION OR PROCEDURE *routine-name* CONTAINS AN INVALID FORMAT OF THE EXTERNAL NAME CLAUSE OR IS MISSING THE EXTERNAL NAME CLAUSE
-450	USER-DEFINED FUNCTION OR STORED PROCEDURE *name*, PARAMETER NUMBER
-451	THE *data-item* DEFINITION, IN THE CREATE FUNCTION FOR
-453	THERE IS A PROBLEM WITH THE RETURNS CLAUSE IN THE CREATE FUNCTION STATEMENT FOR *function-name*
-454	THE SIGNATURE PROVIDED IN THE CREATE FUNCTION STATEMENT FOR
-455	IN CREATE FUNCTION FOR *function-name*, THE SCHEMA NAME
-456	IN CREATE FUNCTION FOR *function-name*, THE SPECIFIC NAME
-457	A FUNCTION OR DISTINCT TYPE CANNOT BE CALLED *name* SINCE IT IS RESERVED FOR SYSTEM USE
-458	IN A REFERENCE TO FUNCTION *function-name* BY SIGNATURE, A MATCHING FUNCTION COULD NOT BE FOUND
-461	A VALUE WITH DATA TYPE *source-data-type* CANNOT BE CAST TO TYPE *target-data-type*
-463	EXTERNAL ROUTINE *routine-name* (SPECIFIC NAME
-469	SQL CALL STATEMENT MUST SPECIFY AN OUTPUT HOST VARIABLE FOR PARAMETER *number*
-470	SQL CALL STATEMENT SPECIFIED A NULL VALUE FOR INPUT PARAMETER
-471	INVOCATION OF FUNCTION OR PROCEDURE *name* FAILED DUE TO REASON *rc*
-472	CURSOR *cursor-name* WAS LEFT OPEN BY EXTERNAL FUNCTION
-473	A USER DEFINED DATA TYPE CANNOT BE CALLED THE SAME NAME AS A SYSTEM PREDEFINED TYPE (BUILT-IN TYPE)

-475 THE RESULT TYPE *type-1* OF THE SOURCE FUNCTION CANNOT BE CAST TO THE RETURNS TYPE *type-2* OF THE USER-DEFINED FUNCTION

-476 REFERENCE TO FUNCTION *function-name* WAS NAMED WITHOUT A SIGNATURE, BUT THE FUNCTION IS NOT UNIQUE WITHIN ITS SCHEMA

-478 DROP OR REVOKE ON OBJECT TYPE *type1* CANNOT BE PROCESSED BECAUSE OBJECT *name* OF TYPE *type2* IS DEPENDENT ON IT

-480 THE PROCEDURE *procedure-name* HAS NOT YET BEEN CALLED

-482 THE PROCEDURE *procedure-name* RETURNED NO LOCATORS

-483 IN CREATE FUNCTION FOR *function-name* STATEMENT, THE NUMBER OF PARAMETERS DOES NOT MATCH THE NUMBER OF PARAMETERS OF THE SOURCE FUNCTION

-487 *object-type object-name* ATTEMPTED TO EXECUTE AN SQL STATEMENT WHEN THE DEFINITION OF THE FUNCTION OR PROCEDURE DID NOT SPECIFY THIS ACTION

-491 CREATE STATEMENT FOR USER-DEFINED FUNCTION *function-name* MUST HAVE A RETURNS CLAUSE, AND EITHER THE EXTERNAL CLAUSE (WITH OTHER REQUIRED KEYWORDS) OR THE SOURCE CLAUSE

-492 THE CREATE FUNCTION FOR *function-name* HAS A PROBLEM WITH PARAMETER NUMBER *number*. IT MAY INVOLVE A MISMATCH WITH A SOURCE FUNCTION

-495 ESTIMATED PROCESSOR COST OF *estimate-amount1* PROCESSOR SECONDS (*estimate-amount2* SERVICE UNITS) IN COST CATEGORY *cost-category* EXCEEDS A RESOURCE LIMIT ERROR THRESHOLD OF *limit- amount* SERVICE UNITS

-496 THE SQL STATEMENT CANNOT BE EXECUTED BECAUSE IT REFERENCES A RESULT SET THAT WAS NOT CREATED BY THE CURRENT SERVER

-497 THE MAXIMUM LIMIT OF INTERNAL IDENTIFIERS HAS BEEN EXCEEDED FOR DATABASE

-499 CURSOR *cursor-name* HAS ALREADY BEEN ASSIGNED TO THIS OR ANOTHER RESULT SET FROM PROCEDURE *procedure-name*.

-500 THE IDENTIFIED CURSOR WAS CLOSED WHEN THE CONNECTION WAS DESTROYED

-501 THE CURSOR IDENTIFIED IN A FETCH OR CLOSE STATEMENT IS NOT OPEN

-502 THE CURSOR IDENTIFIED IN AN OPEN STATEMENT IS ALREADY OPEN

SQL return codes

-503 A COLUMN CANNOT BE UPDATED BECAUSE IT IS NOT IDENTIFIED IN THE UPDATE CLAUSE OF THE SELECT STATEMENT OF THE CURSOR

-504 THE CURSOR NAME *cursor-name* IS NOT DEFINED

-507 THE CURSOR IDENTIFIED IN THE UPDATE OR DELETE STATEMENT IS NOT OPEN

-508 THE CURSOR IDENTIFIED IN THE UPDATE OR DELETE STATEMENT IS NOT POSITIONED ON A ROW

-509 THE TABLE IDENTIFIED IN THE UPDATE OR DELETE STATEMENT IS NOT THE SAME TABLE DESIGNATED BY THE CURSOR

-510 THE TABLE DESIGNATED BY THE CURSOR OF THE UPDATE OR DELETE STATEMENT CANNOT BE MODIFIED

-511 THE FOR UPDATE CLAUSE CANNOT BE SPECIFIED BECAUSE THE TABLE DESIGNATED BY THE CURSOR CANNOT BE MODIFIED

-512 STATEMENT REFERENCE TO REMOTE OBJECT IS INVALID

-513 THE ALIAS *alias-name* MUST NOT BE DEFINED ON ANOTHER LOCAL OR REMOTE ALIAS

-514 THE CURSOR *cursor-name* IS NOT IN A PREPARED STATE

-516 THE DESCRIBE FOR STATIC STATEMENT DOES NOT IDENTIFY A PREPARED STATEMENT

-517 CURSOR *cursor-name* CANNOT BE USED BECAUSE ITS STATEMENT NAME DOES NOT IDENTIFY A PREPARED SELECT STATEMENT

-518 THE EXECUTE STATEMENT DOES NOT IDENTIFY A VALID PREPARED STATEMENT

-519 THE PREPARE STATEMENT IDENTIFIES THE SELECT STATEMENT OF THE OPENED CURSOR *cursor-name*

-525 THE SQL STATEMENT CANNOT BE EXECUTED BECAUSE IT WAS IN ERROR AT BIND TIME FOR SECTION = *sectno* PACKAGE = *pkgname* CONSISTENCY TOKEN = X'contoken'

-526 THE REQUESTED FUNCTION DOES NOT APPLY TO GLOBAL TEMPORARY TABLES

-530 THE INSERT OR UPDATE VALUE OF FOREIGN KEY *constraint-name* IS INVALID

-531 PARENT KEY IN A PARENT ROW CANNOT BE UPDATED BECAUSE IT HAS ONE OR MORE DEPENDENT ROWS IN RELATIONSHIP *constraint-name*

-532 THE RELATIONSHIP *constraint-name* RESTRICTS THE DELETION OF ROW WITH RID X'rid-number'

-533 INVALID MULTIPLE-ROW INSERT

-534 THE PRIMARY KEY CANNOT BE UPDATED BECAUSE OF MULTIPLE-ROW UPDATE

-536 THE DELETE STATEMENT IS INVALID BECAUSE TABLE table-name CAN BE AFFECTED BY THE OPERATION

-537 THE PRIMARY KEY CLAUSE, A FOREIGN KEY CLAUSE, OR A UNIQUE CLAUSE IDENTIFIES COLUMN *column-name* MORE THAN ONCE

-538 FOREIGN KEY *name* DOES NOT CONFORM TO THE DESCRIPTION OF A PARENT KEY OF TABLE *table-name*

-539 TABLE *table-name* DOES NOT HAVE A PRIMARY KEY

-540 THE DEFINITION OF TABLE *table-name* IS INCOMPLETE BECAUSE IT LACKS A PRIMARY INDEX OR A REQUIRED UNIQUE INDEX

-542 *column-name* CANNOT BE A COLUMN OF A PRIMARY KEY, A UNIQUE CONSTRAINT, OR A PARENT KEY BECAUSE IT CAN CONTAIN NULL VALUES

-543 A ROW IN A PARENT TABLE CANNOT BE DELETED BECAUSE THE CHECK CONSTRAINT

-544 THE CHECK CONSTRAINT SPECIFIED IN THE ALTER TABLE STATEMENT CANNOT BE ADDED BECAUSE AN EXISTING ROW VIOLATES THE CHECK CONSTRAINT

-545 THE REQUESTED OPERATION IS NOT ALLOWED BECAUSE A ROW DOES NOT SATISFY THE CHECK CONSTRAINT *check-constraint*

-546 THE CHECK CONSTRAINT *constraint-name* IS INVALID

-548 A CHECK CONSTRAINT THAT IS DEFINED WITH *column-name* IS INVALID

-549 THE *statement* STATEMENT IS NOT ALLOWED FOR *object_type1 object_name* BECAUSE THE BIND OPTION DYNAMICRULES(RUN) IS NOT IN EFFECT FOR *object_type2*

-551 *auth-id* DOES NOT HAVE THE PRIVILEGE TO PERFORM OPERATION

-552 *auth-id* DOES NOT HAVE THE PRIVILEGE TO PERFORM OPERATION

-553 *auth-id* SPECIFIED IS NOT ONE OF THE VALID AUTHORIZATION IDS

-554 AN AUTHORIZATION ID CANNOT GRANT A PRIVILEGE TO ITSELF

-555 AN AUTHORIZATION ID CANNOT REVOKE A PRIVILEGE FROM ITSELF

-556 *authid2* CANNOT HAVE THE *privilege* PRIVILEGE

-557 INCONSISTENT GRANT/REVOKE KEYWORD *keyword*. PERMITTED KEYWORDS ARE *keyword-list*

-558 INVALID CLAUSE OR COMBINATION OF CLAUSES ON A GRANT OR REVOKE

-559 ALL AUTHORIZATION FUNCTIONS HAVE BEEN DISABLED

SQL return codes

-567 *bind-type* AUTHORIZATION ERROR USING *auth-id* AUTHORITY PACKAGE = *package-name* PRIVILEGE = *privilege*

-571 THE STATEMENT WOULD RESULT IN A MULTIPLE SITE UPDATE

-573 TABLE *table-name* DOES NOT HAVE A UNIQUE KEY WITH THE SPECIFIED COLUMN NAMES

-574 THE SPECIFIED DEFAULT VALUE CONFLICTS WITH THE DEFINITION OF COLUMN *column-name*

-577 *object-type object-name* ATTEMPTED TO MODIFY DATA WHEN THE DEFINITION OF THE FUNCTION OR PROCEDURE DID NOT SPECIFY THIS ACTION

-579 *object-type object-name* ATTEMPTED TO READ OR MODIFY DATA WHEN THE DEFINITION OF THE FUNCTION OR PROCEDURE DID NOT SPECIFY THIS ACTION

-580 THE RESULT-EXPRESSIONS OF A CASE EXPRESSION CANNOT ALL BE NULL

-581 THE DATA TYPES OF THE RESULT-EXPRESSIONS OF A CASE EXPRESSION ARE NOT COMPATIBLE

-582 THE SEARCH-CONDITION IN A SEARCHED-WHEN-CLAUSE CANNOT BE A QUANTIFIED PREDICATE, IN PREDICATE, OR AN EXISTS PREDICATE.

-583 The use of function *function-name* is invalid because it is not deterministic or may have an external action.

-585 THE SCHEMA NAME *schema-name* CANNOT APPEAR MORE THAN ONCE IN THE CURRENT PATH

-586 THE TOTAL LENGTH OF THE CURRENT PATH SPECIAL REGISTER CANNOT EXCEED 254 CHARACTERS

-587 A list of item-references are not in the same family.

-590 PARAMETER NAME *parameter-name* IS NOT UNIQUE IN THE CREATE FOR ROUTINE *routine-name*

-592 NOT AUTHORIZED TO CREATE FUNCTIONS OR PROCEDURES IN WLM ENVIRONMENT

-601 THE NAME OF THE OBJECT TO BE CREATED OR THE TARGET OF A RENAME STATEMENT IS IDENTICAL TO THE EXISTING NAME *name* OF THE OBJECT TYPE

-602 TOO MANY COLUMNS SPECIFIED IN A CREATE INDEX

-603 A UNIQUE INDEX CANNOT BE CREATED BECAUSE THE TABLE CONTAINS ROWS WHICH ARE DUPLICATES WITH RESPECT TO THE VALUES OF THE IDENTIFIED COLUMNS

-604 A DATA TYPE DEFINITION SPECIFIES AN INVALID LENGTH, PRECISION, OR SCALE ATTRIBUTE

-607 OPERATION OR OPTION *operation* IS NOT DEFINED FOR THIS OBJECT

-611 ONLY LOCKMAX 0 CAN BE SPECIFIED WHEN THE LOCK SIZE OF THE TABLESPACE IS TABLESPACE OR TABLE

-612 *column-name* IS A DUPLICATE COLUMN NAME

-613 THE PRIMARY KEY OR A UNIQUE CONSTRAINT IS TOO LONG OR HAS TOO MANY COLUMNS

-614 THE INDEX CANNOT BE CREATED OR THE LENGTH OF A COLUMN CANNOT BE CHANGED BECAUSE THE SUM OF THE INTERNAL LENGTHS OF THE IDENTIFIED COLUMNS IS GREATER THAN THE ALLOWABLE MAXIMUM

-615 *operation-type* IS NOT ALLOWED ON A PACKAGE IN USE

-616 *obj-type1 obj-name1* CANNOT BE DROPPED BECAUSE IT IS REFERENCED BY *obj-type2 obj-name2*

-617 A TYPE 1 INDEX IS NOT VALID FOR TABLE *table-name*

-618 OPERATION *operation* IS NOT ALLOWED ON SYSTEM DATABASES

-619 OPERATION DISALLOWED BECAUSE THE WORK FILE DATABASE IS NOT STOPPED

-620 KEYWORD *keyword* IN *stmt type* STATEMENT IS NOT PERMITTED FOR A TABLE SPACE IN THE WORK FILE DATABASE

-621 DUPLICATE DBID *dbid* WAS DETECTED AND PREVIOUSLY ASSIGNED TO *database-name*

-622 *FOR MIXED DATA* IS INVALID BECAUSE THE MIXED DATA INSTALL OPTION IS NO

-623 A CLUSTERING INDEX ALREADY EXISTS ON TABLE *table-name*

-624 TABLE *table-name* ALREADY HAS A PRIMARY KEY

-625 TABLE *table-name* DOES NOT HAVE AN INDEX TO ENFORCE THE UNIQUENESS OF THE PARENT KEY

-626 THE ALTER STATEMENT IS NOT EXECUTABLE BECAUSE THE PAGE SET IS NOT STOPPED

-627 THE ALTER STATEMENT IS INVALID BECAUSE THE PAGESET HAS USER-MANAGED DATA SETS

-628 THE CLAUSES ARE MUTUALLY EXCLUSIVE.

-629 SET NULL CANNOT BE SPECIFIED BECAUSE FOREIGN KEY *name* CANNOT CONTAIN NULL VALUES

-630 THE WHERE NOT NULL SPECIFICATION IS INVALID FOR TYPE 1 INDEXES

-631 FOREIGN KEY *name* IS TOO LONG OR HAS TOO MANY COLUMNS

SQL return codes

-632 THE TABLE CANNOT BE DEFINED AS A DEPENDENT OF *table-name* BECAUSE OF DELETE RULE RESTRICTIONS

-633 THE DELETE RULE MUST BE *delete-rule*

-634 THE DELETE RULE MUST NOT BE CASCADE

-635 THE DELETE RULES CANNOT BE DIFFERENT OR CANNOT BE SET NULL

-636 THE PARTITIONING KEYS FOR PARTITION *part-num* ARE NOT SPECIFIED IN ASCENDING OR DESCENDING ORDER

-637 DUPLICATE *keyword* KEYWORD

-638 TABLE *table-name* CANNOT BE CREATED BECAUSE COLUMN DEFINITION IS MISSING

-639 A NULLABLE COLUMN OF A FOREIGN KEY WITH A DELETE RULE OF SET NULL CANNOT BE A COLUMN OF THE KEY OF A PARTITIONED INDEX

-640 LOCKSIZE ROW CANNOT BE SPECIFIED BECAUSE TABLE IN THIS TABLESPACE HAS TYPE 1 INDEX

-642 TOO MANY COLUMNS IN UNIQUE CONSTRAINTS

-643 CHECK CONSTRAINT EXCEEDS MAXIMUM ALLOWABLE LENGTH

-644 INVALID VALUE SPECIFIED FOR KEYWORD *keyword* IN

-646 TABLE *table-name* CANNOT BE CREATED IN SPECIFIED TABLE SPACE *table-space-name* BECAUSE IT ALREADY CONTAINS A TABLE

-647 BUFFERPOOL *bp-name* CANNOT BE SPECIFIED BECAUSE IT HAS NOT BEEN ACTIVATED

-650 THE ALTER INDEX CANNOT BE EXECUTED, REASON *reason*

-651 TABLE DESCRIPTION EXCEEDS MAXIMUM SIZE OF OBJECT DESCRIPTOR.

-652 VIOLATION OF INSTALLATION DEFINED EDIT OR VALIDATION PROCEDURE

-653 TABLE *table-name* IN PARTITIONED TABLE SPACE *tspace-name* IS NOT AVAILABLE BECAUSE ITS PARTITIONED INDEX HAS NOT BEEN CREATED

-655 THE CREATE OR ALTER STOGROUP IS INVALID BECAUSE THE STORAGE GROUP WOULD HAVE BOTH SPECIFIC AND NON-SPECIFIC VOLUME IDS

-658 A *object-type* CANNOT BE DROPPED USING THE

-660 INDEX *index-name* CANNOT BE CREATED OR ALTERED ON PARTITIONED TABLE SPACE *tspace-name* BECAUSE KEY LIMITS ARE NOT SPECIFIED

-661 INDEX *index-name* CANNOT BE CREATED ON PARTITIONED TABLE SPACE *tspace-name* BECAUSE THE NUMBER OF PART SPECIFICATIONS IS NOT EQUAL TO THE NUMBER OF PARTITIONS OF THE TABLE SPACE

-662 A PARTITIONED INDEX CANNOT BE CREATED ON A NON-PARTITIONED TABLE SPACE *tspace-name*

-663 THE NUMBER OF KEY LIMIT VALUES IS EITHER ZERO, OR GREATER THAN THE NUMBER OF COLUMNS IN THE KEY OF INDEX *index-name*

-665 THE PART CLAUSE OF AN ALTER STATEMENT IS OMITTED OR INVALID

-666 *stmt-verb object* CANNOT BE EXECUTED BECAUSE *function* IS IN PROGRESS

-667 THE CLUSTERING INDEX FOR A PARTITIONED TABLE SPACE CANNOT BE EXPLICITLY DROPPED

-668 THE COLUMN CANNOT BE ADDED TO THE TABLE BECAUSE THE TABLE HAS AN EDIT PROCEDURE

-669 A TABLE IN A PARTITIONED TABLE SPACE CANNOT BE EXPLICITLY DROPPED

-670 THE RECORD LENGTH OF THE TABLE EXCEEDS THE PAGE SIZE LIMIT

-671 THE BUFFERPOOL ATTRIBUTE OF THE TABLE SPACE CANNOT BE ALTERED AS SPECIFIED BECAUSE IT WOULD CHANGE THE PAGE SIZE OF THE TABLE SPACE

-672 OPERATION DROP NOT ALLOWED ON TABLE *table_name*

-676 ONLY A 4K PAGE BUFFERPOOL CAN BE USED FOR AN INDEX

-677 INSUFFICIENT VIRTUAL STORAGE FOR BUFFERPOOL EXPANSION

-678 THE LITERAL literal SPECIFIED FOR THE INDEX LIMIT KEY MUST CONFORM TO THE DATA TYPE *data-type* OF THE CORRESPONDING COLUMN

-679 THE OBJECT *name* CANNOT BE CREATED BECAUSE A DROP IS PENDING ON THE OBJECT

-680 TOO MANY COLUMNS SPECIFIED FOR A TABLE

-681 COLUMN *column-name* IN VIOLATION OF INSTALLATION DEFINED FIELD PROCEDURE. RT: *return-code*, RS:

-682 FIELD PROCEDURE *procedure-name* COULD NOT BE LOADED

-683 THE SPECIFICATION FOR COLUMN, DISTINCT TYPE, FUNCTION, OR PROCEDURE *data-item* CONTAINS INCOMPATIBLE CLAUSES

-684 THE LENGTH OF LITERAL LIST BEGINNING *string* IS TOO LONG

-685 INVALID FIELD TYPE, *column-name*

SQL return codes

-686 COLUMN DEFINED WITH A FIELD PROCEDURE CAN NOT COMPARE WITH ANOTHER COLUMN WITH DIFFERENT FIELD PROCEDURE

-687 FIELD TYPES INCOMPARABLE

-688 INCORRECT DATA RETURNED FROM FIELD PROCEDURE, *column-name*,

-689 TOO MANY COLUMNS DEFINED FOR A DEPENDENT TABLE

-690 THE STATEMENT IS REJECTED BY DATA DEFINITION CONTROL SUPPORT. REASON

-691 THE REQUIRED REGISTRATION TABLE *table-name* DOES NOT EXIST

-692 THE REQUIRED UNIQUE INDEX *index-name* FOR DDL REGISTRATION TABLE *table-name* DOES NOT EXIST

-693 THE COLUMN *column-name* IN DDL REGISTRATION TABLE OR INDEX

-694 THE DDL STATEMENT CANNOT BE EXECUTED BECAUSE A DROP IS PENDING ON THE DDL REGISTRATION TABLE *table-name*

-696 THE DEFINITION OF TRIGGER *trigger-name* INCLUDES AN INVALID USE OF CORRELATION NAME OR TRANSITION TABLE NAME *name*. REASON CODE=*reason-code*

-697 OLD OR NEW CORRELATION NAMES ARE NOT ALLOWED IN A TRIGGER DEFINED WITH THE FOR EACH STATEMENT CLAUSE. OLD_TABLE OR NEW_TABLE NAMES ARE NOT ALLOWED IN A TRIGGER WITH THE BEFORE CLAUSE.

-713 THE REPLACEMENT VALUE *value* FOR *special-register* IS INVALID

-715 PROGRAM *program-name* WITH MARK *release-dependency-mark* FAILED BECAUSE IT DEPENDS ON FUNCTIONS OF THE RELEASE FROM WHICH FALLBACK HAS OCCURRED

-716 PROGRAM *program-name* PRECOMPILED WITH INCORRECT LEVEL FOR THIS RELEASE

-717 *bind-type* FOR *object-type object-name* WITH MARK

-718 REBIND OF PACKAGE *package-name* FAILED BECAUSE IBMREQD OF

-719 BIND ADD ERROR USING *auth-id* AUTHORITY PACKAGE

-720 BIND ERROR, ATTEMPTING TO REPLACE PACKAGE = *package_name* WITH VERSION = *version2* BUT THIS VERSION ALREADY EXISTS

-721 BIND ERROR FOR PACKAGE = *pkg-id* CONTOKEN = 'contoken'X IS NOT UNIQUE SO IT CANNOT BE CREATED

-722 *bind-type* ERROR USING *auth-id* AUTHORITY PACKAGE

-723 AN ERROR OCCURRED IN A TRIGGERED SQL STATEMENT IN TRIGGER

-724 THE ACTIVATION OF THE *object-type* OBJECT *object-name* WOULD EXCEED THE MAXIMUM LEVEL OF INDIRECT SQL CASCADING

-725 THE SPECIAL REGISTER *register* AT LOCATION *location* WAS SUPPLIED AN INVALID VALUE

-726 BIND ERROR ATTEMPTING TO REPLACE PACKAGE = <package_name>. THERE ARE ENABLE OR DISABLE ENTRIES CURRENTLY ASSOCIATED WITH THE PACKAGE

-728 DATA TYPE *data-type* IS NOT ALLOWED IN DB2 PRIVATE PROTOCOL PROCESSING

-729 A STORED PROCEDURE SPECIFYING COMMIT ON RETURN CANNOT BE THE TARGET OF A NESTED CALL STATEMENT

-730 THE PARENT OF A TABLE IN A READ-ONLY SHARED DATABASE MUST ALSO BE A TABLE IN A READ-ONLY SHARED DATABASE

-731 USER-DEFINED DATASET *dsname* MUST BE DEFINED WITH SHAREOPTIONS(1,3)

-732 THE DATABASE IS DEFINED ON THIS SUBSYSTEM WITH THE ROSHARE READ ATTRIBUTE BUT THE TABLE SPACE OR INDEX SPACE HAS NOT BEEN DEFINED ON THE OWNING SUBSYSTEM

-733 THE DESCRIPTION OF A TABLE SPACE, INDEX SPACE, OR TABLE IN A ROSHARE READ DATABASE MUST BE CONSISTENT WITH ITS DESCRIPTION IN THE OWNER SYSTEM

-734 THE ROSHARE ATTRIBUTE OF A DATABASE CANNOT BE ALTERED FROM ROSHARE READ

-735 DATABASE dbid CANNOT BE ACCESSED BECAUSE IT IS NO LONGER A SHARED DATABASE

-736 INVALID OBID *obid* SPECIFIED

-737 IMPLICIT TABLE SPACE NOT ALLOWED

-739 ALTER FUNCTION *function-name* FAILED BECAUSE FUNCTIONS CANNOT MODIFY DATA WHEN THEY ARE PROCESSED IN PARALLEL

-740 FUNCTION *name* IS DEFINED WITH THE OPTION MODIFIES SQL DATA WHICH IS NOT VALID IN THE CONTEXT IN WHICH IT WAS INVOKED

-741 A WORK FILE DATABASE IS ALREADY DEFINED FOR MEMBER *member-name*

-742 DSNDB07 IS THE IMPLICIT WORK FILE DATABASE

-746 THE SQL STATEMENT IN AN EXTERNAL FUNCTION, TRIGGER, OR IN STORED PROCEDURE *name* VIOLATES THE NESTING SQL RESTRICTION

-747 TABLE *table-name* IS NOT AVAILABLE UNTIL THE AUXILIARY TABLES AND INDEXES FOR ITS EXTERNALLY STORED COLUMNS HAVE BEEN CREATED

SQL return codes

-748 AN INDEX ALREADY EXISTS ON AUXILIARY TABLE

-750 THE SOURCE TABLE *source-name* CANNOT BE RENAMED BECAUSE IT IS REFERENCED IN EXISTING VIEW DEFINITIONS OR TRIGGER DEFINITIONS

-751 *object-type object-name* (SPECIFIC NAME *specific name*) ATTEMPTED TO EXECUTE AN SQL STATEMENT *statement* THAT IS NOT ALLOWED

-752 THE CONNECT STATEMENT IS INVALID BECAUSE THE PROCESS IS NOT IN THE CONNECTABLE STATE

-763 INVALID TABLE SPACE NAME *table-space-name*

-764 A LOB TABLE SPACE AND ITS ASSOCIATED BASE TABLE SPACE MUST BE IN THE SAME DATABASE

-765 TABLE IS NOT COMPATIBLE WITH DATABASE

-766 THE OBJECT OF A STATEMENT IS AN AUXILIARY TABLE FOR WHICH THE REQUESTED OPERATION IS NOT PERMITTED

-767 MISSING OR INVALID COLUMN SPECIFICATION FOR INDEX

-768 AN AUXILIARY TABLE ALREADY EXISTS FOR THE SPECIFIED COLUMN OR PARTITION

-769 SPECIFICATION OF CREATE AUX TABLE DOES NOT MATCH THE CHARACTERISTICS OF THE BASE TABLE

-770 TABLE *table-name* CANNOT HAVE A LOB COLUMN UNLESS IT ALSO HAS A ROWID COLUMN

-771 INVALID SPECIFICATION OF A ROWID COLUMN

-797 ATTEMPT to CREATE TRIGGER *trigger-name* WITH AN UNSUPPORTED TRIGGERED SQL STATEMENT

-798 ATTEMPTED TO INSERT A VALUE INTO A ROWID GENERATED ALWAYS COLUMN*column-name*

-802 EXCEPTION ERROR 'exception-type' HAS OCCURRED DURING 'operation-type' OPERATION ON 'data-type' DATA, POSITION 'position-number'

-803 AN INSERTED OR UPDATED VALUE IS INVALID BECAUSE THE INDEX IN INDEX SPACE *indexspace-name* CONSTRAINS COLUMNS OF THE TABLE SO NO TWO ROWS CAN CONTAIN DUPLICATE VALUES IN THOSE COLUMNS. RID OF EXISTING ROW IS X'rid'

-804 AN ERROR WAS FOUND IN THE APPLICATION PROGRAM INPUT PARAMETERS FOR THE SQL STATEMENT, REASON *reason*

-805 DBRM OR PACKAGE NAME

-807 ACCESS DENIED: PACKAGE *package-name* IS NOT ENABLED FOR ACCESS FROM *connection-type connection-name*

-808 THE CONNECT STATEMENT IS NOT CONSISTENT WITH THE FIRST CONNECT STATEMENT

-811 THE RESULT OF AN EMBEDDED SELECT STATEMENT IS A TABLE OF MORE THAN ONE ROW, OR THE RESULT OF THE SUBQUERY OF A BASIC PREDICATE IS MORE THAN ONE VALUE

-812 THE SQL STATEMENT CANNOT BE PROCESSED BECAUSE A BLANK COLLECTION-ID WAS FOUND IN THE CURRENT PACKAGESET SPECIAL REGISTER WHILE TRYING TO FORM A QUALIFIED PACKAGE NAME FOR PROGRAM *program-name.consistency-token* USING PLAN *plan-name*

-815 A GROUP BY OR HAVING CLAUSE IS IMPLICITLY OR EXPLICITLY SPECIFIED IN AN EMBEDDED SELECT STATEMENT OR A SUBQUERY OF A BASIC PREDICATE

-817 THE SQL STATEMENT CANNOT BE EXECUTED BECAUSE THE STATEMENT WILL RESULT IN A PROHIBITED UPDATE OPERATION.

-818 THE PRECOMPILER-GENERATED TIMESTAMP *x* IN THE LOAD MODULE IS DIFFERENT FROM THE BIND TIMESTAMP *y* BUILT FROM THE DBRM *z*

-819 THE VIEW CANNOT BE PROCESSED BECAUSE THE LENGTH OF ITS PARSE TREE IN THE CATALOG IS ZERO

-820 THE SQL STATEMENT CANNOT BE PROCESSED BECAUSE *catalog-table* CONTAINS A VALUE THAT IS NOT VALID IN THIS RELEASE

-822 THE SQLDA CONTAINS AN INVALID DATA ADDRESS OR INDICATOR VARIABLE ADDRESS

-840 TOO MANY ITEMS RETURNED IN A SELECT OR INSERT LIST

-842 A CONNECTION TO *location-name* ALREADY EXISTS

-843 THE SET CONNECTION OR RELEASE STATEMENT MUST SPECIFY AN EXISTING CONNECTION

-870 THE NUMBER OF HOST VARIABLES IN THE STATEMENT IS NOT EQUAL TO THE NUMBER OF DESCRIPTORS

-872 A VALID CCSID HAS NOT YET BEEN SPECIFIED FOR THIS SUBSYSTEM

-873 DATA ENCODED WITH DIFFERENT CCSIDS CANNOT BE REFERENCED IN THE SAME SQL STATEMENT

-874 THE ENCODING SCHEME SPECIFIED FOR THE *object-type* MUST BE THE SAME AS THE CONTAINING TABLE SPACE OR OTHER PARAMETERS

-875 *operand* CANNOT BE USED WITH THE ASCII DATA REFERENCED

-876 '*object*' CANNOT BE CREATED, REASON '*reason*'

SQL return codes

-877 CCSID ASCII IS NOT ALLOWED FOR THIS DATABASE OR TABLE SPACE

-878 THE PLAN_TABLE USED FOR EXPLAIN CANNOT BE ASCII

-879 CREATE or ALTER STATEMENT FOR *obj-name* CANNOT DEFINE A COLUMN, DISTINCT TYPE, FUNCTION OR STORED PROCEDURE PARAMETER AS MIXED OR GRAPHIC WITH ENCODING SCHEME *encoding-scheme*

-900 THE SQL STATEMENT CANNOT BE EXECUTED BECAUSE THE APPLICATION PROCESS IS NOT CONNECTED TO AN APPLICATION SERVER

-901 UNSUCCESSFUL EXECUTION CAUSED BY A SYSTEM ERROR THAT DOES NOT PRECLUDE THE SUCCESSFUL EXECUTION OF SUBSEQUENT SQL STATEMENTS

-902 POINTER TO THE ESSENTIAL CONTROL BLOCK (CT/RDA) HAS VALUE 0, REBIND REQUIRED

-904 UNSUCCESSFUL EXECUTION CAUSED BY AN UNAVAILABLE RESOURCE. REASON

-905 UNSUCCESSFUL EXECUTION DUE TO RESOURCE LIMIT BEING EXCEEDED, RESOURCE NAME = *resource-name* LIMIT = *limit-amount1* CPU SECONDS (*limit-amount2* SERVICE UNITS) DERIVED FROM *limit-source*

-906 THE SQL STATEMENT CANNOT BE EXECUTED BECAUSE THIS FUNCTION IS DISABLED DUE TO A PRIOR ERROR

-908 *bind-type* ERROR USING *auth-id* AUTHORITY. BIND, REBIND OR AUTO-REBIND OPERATION IS NOT ALLOWED

-909 THE OBJECT HAS BEEN DELETED

-910 THE SQL STATEMENT CANNOT ACCESS AN OBJECT ON WHICH A DROP OR ALTER IS PENDING

-911 THE CURRENT UNIT OF WORK HAS BEEN ROLLED BACK DUE TO DEADLOCK OR TIMEOUT. REASON *reason-code*, TYPE OF RESOURCE *resource-type*, AND RESOURCE NAME *resource-name*

-913 UNSUCCESSFUL EXECUTION CAUSED BY DEADLOCK OR TIMEOUT. REASON CODE

-917 BIND PACKAGE FAILED

-918 THE SQL STATEMENT CANNOT BE EXECUTED BECAUSE A CONNECTION HAS BEEN LOST

-919 A ROLLBACK OPERATION IS REQUIRED

-922 AUTHORIZATION FAILURE: *error-type* ERROR. REASON

-923 CONNECTION NOT ESTABLISHED: DB2 *condition* REASON

-924 DB2 CONNECTION INTERNAL ERROR, *function-code, return-code, reason-code*

-925 COMMIT NOT VALID IN IMS OR CICS ENVIRONMENT

-926 ROLLBACK NOT VALID IN IMS OR CICS ENVIRONMENT

-927 THE LANGUAGE INTERFACE (LI) WAS CALLED WHEN THE CONNECTING ENVIRONMENT WAS NOT ESTABLISHED. THE PROGRAM SHOULD BE INVOKED UNDER THE DSN COMMAND

-929 FAILURE IN A DATA CAPTURE EXIT: *token*

-939 ROLLBACK REQUIRED DUE TO UNREQUESTED ROLLBACK OF A REMOTE SERVER

-947 THE SQL STATEMENT FAILED BECAUSE IT WILL CHANGE A TABLE DEFINED WITH DATA CAPTURE CHANGES, BUT THE DATA CANNOT BE PROPAGATED

-948 DISTRIBUTED OPERATION IS INVALID

-950 THE LOCATION NAME SPECIFIED IN THE CONNECT STATEMENT IS INVALID OR NOT LISTED IN THE COMMUNICATIONS DATABASE

-981 THE SQL STATEMENT FAILED BECAUSE THE RRSAF CONNECTION IS NOT IN A STATE THAT ALLOWS SQL OPERATIONS, REASON *reason-code*.

-991 CALL ATTACH WAS UNABLE TO ESTABLISH AN IMPLICIT CONNECT OR OPEN TO DB2. RC1= *rc1* RC2= *rc2*

-2001 THE NUMBER OF HOST VARIABLE PARAMETERS FOR A STORED PROCEDURE IS NOT EQUAL TO THE NUMBER OF EXPECTED HOST VARIABLE PARAMETERS. ACTUAL NUMBER

-20003 GBPCACHE NONE CANNOT BE SPECIFIED FOR TABLESPACE OR INDEX IN GRECP

-20004 8K or 16K BUFFERPOOL PAGESIZE INVALID FOR A WORKFILE OBJECT

-20005 THE INTERNAL ID LIMIT OF *limit* HAS BEEN EXCEEDED FOR OBJECT TYPE *object-type*

-20006 LOBS CANNOT BE SPECIFIED AS PARAMETERS WHEN NO WLM ENVIRONMENT IS SPECIFIED

-20008 UNSUPPORTED OPTION *keyword* SPECIFIED

-20070 AUXILIARY TABLE *table-name* CANNOT BE CREATED BECAUSE COLUMN

-20071 WLM ENVIRONMENT NAME MUST BE SPECIFIED

-20072 *csect-name bind-type bind-subtype* ERROR USING

-20073 THE FUNCTION *function-name* CANNOT BE ALTERED BECAUSE IT IS REFERENCED IN EXISTING VIEW DEFINITIONS

SQL return codes

-20074 THE OBJECT *object-name* CANNOT BE CREATED BECAUSE THE FIRST THREE CHARACTERS ARE RESERVED FOR SYSTEM OBJECTS

-20100 AN ERROR OCCURRED WHEN BINDING A TRIGGERED SQL STATEMENT. INFORMATION RETURNED: SECTION NUMBER : *section-number* SQLCODE *sqlerror*, SQLSTATE *sqlstate*, AND MESSAGE TOKENS

-20101 THE FUNCTION *function* FAILED WITH REASON *rc*

-20102 CREATE OR ALTER STATEMENT FOR USER-DEFINED FUNCTION *function-name* SPECIFIED THE *option* OPTION WHICH IS NOT ALLOWED FOR THE TYPE OF FUNCTION

-20104 AN ATTEMPT TO ALTER A CCSID FROM

-20106 The CCSID for table space or database cannot be changed because the table space or database already contains a table that is referenced in existing view definitions.

-30000 EXECUTION FAILED DUE TO A DISTRIBUTION PROTOCOL ERROR THAT WILL NOT AFFECT THE SUCCESSFUL EXECUTION OF SUBSEQUENT COMMANDS OR SQL STATEMENTS: REASON *reason-code (sub-code)*

-30002 THE SQL STATEMENT CANNOT BE EXECUTED DUE TO A PRIOR CONDITION IN A CHAIN OF STATEMENTS

-30020 EXECUTION FAILED DUE TO A DISTRIBUTION PROTOCOL ERROR THAT CAUSED DEALLOCATION OF THE CONVERSATION: REASON <*reason-code (sub-code)*>

-30021 EXECUTION FAILED DUE TO A DISTRIBUTION PROTOCOL ERROR THAT WILL AFFECT THE SUCCESSFUL EXECUTION OF SUBSEQUENT COMMANDS OR SQL STATEMENTS: MANAGER *manager* AT LEVEL *level* NOT SUPPORTED ERROR

-30030 COMMIT REQUEST WAS UNSUCCESSFUL, A DISTRIBUTION PROTOCOL VIOLATION HAS BEEN DETECTED, THE CONVERSATION HAS BEEN DEALLOCATED. ORIGINAL SQLCODE=*original-sqlcode* AND ORIGINAL SQLSTATE=*original-sqlstate*

-30040 EXECUTION FAILED DUE TO UNAVAILABLE RESOURCES THAT WILL NOT AFFECT THE SUCCESSFUL EXECUTION OF SUBSEQUENT COMMANDS OR SQL STATEMENTS. REASON <*reason-code*> TYPE OF RESOURCE <*resource-type*> RESOURCE NAME <*resource-name*> PRODUCT ID <*pppvvrrm*> RDBNAME <*rdbname*>

-30041 EXECUTION FAILED DUE TO UNAVAILABLE RESOURCES THAT WILL AFFECT THE SUCCESSFUL EXECUTION OF SUBSEQUENT COMMANDS AND SQL STATEMENTS. REASON <*reason-code*> TYPE OF RESOURCE <*resource-type*> RESOURCE NAME <*resource-name*> PRODUCT ID <*pppvvrrm*> RDBNAME <*rdbname*>

-30050 <*command-or-SQL-statement-type*> COMMAND OR SQL STATEMENT INVALID WHILE BIND PROCESS IN PROGRESS

-30051 BIND PROCESS WITH SPECIFIED PACKAGE NAME AND CONSISTENCY TOKEN NOT ACTIVE

-30052 PROGRAM PREPARATION ASSUMPTIONS ARE INCORRECT

-30053 OWNER AUTHORIZATION FAILURE

-30060 RDB AUTHORIZATION FAILURE

-30061 RDB NOT FOUND

-30070 <*command*> COMMAND NOT SUPPORTED ERROR

-30071 <*object-type*> OBJECT NOT SUPPORTED ERROR

-30072 <*parameter*>:<*subcode*> PARAMETER NOT SUPPORTED ERROR

-30073 <*parameter*>:<*subcode*> PARAMETER VALUE NOT SUPPORTED ERROR

-30074 REPLY MESSAGE WITH *codepoint (svrcod)* NOT SUPPORTED ERROR

-30080 COMMUNICATION ERROR *code (subcode)*

-30081 *prot* COMMUNICATION ERROR DETECTED. API=*api*, LOCATION=*loc*, FUNCTION=*func*, ERROR CODES=*rc1 rc2 rc3*

-30082 CONNECTION FAILED FOR SECURITY REASON *reason-code* (*reason-string*)

-30090 REMOTE OPERATION INVALID FOR APPLICATION EXECUTION ENVIRONMENT

-30104 ERROR IN BIND OPTION *option* AND BIND VALUE *value*.

-30105 BIND OPTION *option1* IS NOT ALLOWED WITH BIND OPTION

SQL return codes

DB2 catalog table reference

DB2 maintains a set of tables called DB2 catalog tables that store information about the DB2 system. The catalog tables describe tables, columns, indexes, programs, authorizations, and other DB2 objects. The catalog tables are automatically updated by DB2 during normal operation and in response to certain SQL statements. Data in the tables (with the exception of password data) is available to authorized DB2 users using SQL.

This section lists the unqualified names of the DB2 catalog tables in alphabetic order. Each table name is qualified by SYSIBM (as in 'SYSIBM.SYSTABLES'), but the qualifiers are omitted from the list. Along with the name of the table is a description of its purpose and a list of the names of its columns (to be read from left to right, by rows). In addition, for tables that have indexes defined, the name of the index and the columns in the index key are also listed.

For more information on the DB2 catalog tables, see Appendix D of *DB2 SQL Reference*.

Table 1 (Page 1 of 12). List of Catalog Tables

Catalog Table Name	Column Names and Indexes

IPNAMES	Defines the remote DRDA® servers DB2 can access using TCP/IP.
	`LINKNAME SECURITY_OUT USERNAMES IBMREQD IPADDR`
	`DSNFPX01: LINKNAME`

LOCATIONS	Contains a row for every accessible remote server.
	`LOCATION LINKNAME IBMREQD PORT TPN`
	`DSNFCX01: LOCATION`

LULIST	Allows multiple LU names to be specified for a given LOCATION.
	`LINKNAME LUNAME IBMREQD`
	`DSNFLX01: LINKNAME.LUNAME`
	`DSNFLX02: LUNAME`

LUMODES	Each row of the table provides VTAM® with conversation limits for a specific combination of LUNAME and MODENAME.
	`LUNAME MODENAME CONVLIMIT IBMREQD`
	`DSNFMX01: LUNAME.MODENAME`

Catalog tables

Table 1 (Page 2 of 12). List of Catalog Tables

Catalog Table Name	Column Names and Indexes
LUNAMES	The table must contain a row for each remote SNA client or server that communicates with DB2. LUNAME SYSMODENAME SECURITY_IN SECURITY_OUT ENCRYPTPSWDS MODESELECT USERNAMES GENERIC IBMREQD DSNFNX01: LUNAME
MODESELECT	Associates a mode name with any conversation created to support an outgoing SQL request. AUTHID PLANNAME LUNAME MODENAME IBMREQD DSNFDX01: LUNAME.AUTHID.PLANNAME
SYSAUXRELS	Contains one row for each auxiliary table created for a LOB column. A base table space that is partitioned must have one auxiliary table for each partition of each LOB column. TBOWNER TBNAME COLNAME PARTITION AUXTBOWNER AUXTBNAME AUXRELOBID IBMREQD DSNOXX01: TBOWNER.TBNAME DSNOXX02: AUXTBOWNER.AUXTBNAME
SYSCHECKDEP	Contains one row for each reference to a column in a table check constraint. TBOWNER TBNAME CHECKNAME COLNAME IBMREQD DSNSDX01: TBOWNER.TBNAME.CHECKNAME.COLNAME
SYSCHECKS	Contains one row for each table check constraint. TBOWNER CREATOR DBID OBID TIMESTAMP RBA IBMREQD TBNAME CHECKNAME CHECKCONDITION DSNSCX01: TBOWNER.TBNAME.CHECKNAME
SYSCOLAUTH	Records the UPDATE privileges held by users on individual columns of a table or view. GRANTOR GRANTEE GRANTEETYPE CREATOR TNAME DATEGRANTED TIMEGRANTED COLNAME IBMREQD COLLID CONTOKEN PRIVILEGE GRANTEDTS
SYSCOLDIST	Contains one or more rows for the first key column of an index key. STATSTIME IBMREQD TBOWNER TBNAME NAME COLVALUE TYPE CARDF COLGROUPCOLNO NUMCOLUMNS FREQUENCYF DSNTNX01: TBOWNER.TBNAME.NAME

Table 1 (Page 3 of 12). List of Catalog Tables

Catalog Table Name	Column Names and Indexes
SYSCOLDISTSTATS	Contains zero or more rows per partition for the first key column of a partitioned index. STATSTIME IBMREQD PARTITION TBOWNER TBNAME NAME COLVALUE TYPE CARDF COLGROUPCOLNO NUMCOLUMNS FREQUENCYF DSNTPX01: TBOWNER.TBNAME.NAME.PARTITION
SYSCOLSTATS	Contains partition statistics for selected columns. HIGHKEY HIGH2KEY LOWKEY LOW2KEY STATSTIME IBMREQD PARTITION TBOWNER TBNAME NAME COLCARDDATA DSNTCX01: TBOWNER.TBNAME.NAME.PARTITION
SYSCOLUMNS	Contains one row for every column of each table and view. NAME TBNAME TBCREATOR COLNO COLTYPE LENGTH SCALE NULLS HIGH2KEY LOW2KEY UPDATES IBMREQD REMARKS DEFAULT KEYSEQ FOREIGNKEY FLDPROC LABEL STATSTIME DEFAULTVALUE COLCARDF COLSTATUS LENGTH2 DATATYPEID SOURCETYPEID TYPESCHEMA TYPENAME CREATEDTS DSNDCX01: TBCREATOR.TBNAME.NAME DSNDCX02: TYPESCHEMA.TYPENAME
SYSCONSTDEP	Records dependencies on check contraints or user-defined defaults for a column. BNAME BSCHEMA BTYPE DTBNAME DTBCREATOR DCONSTNAME DTYPE IBMREQD DSNCCX01: BSCHEMA.BNAME.BTYPE DSNCCX02: DTBCREATOR.DTBNAME
SYSCOPY	Contains information needed for recovery. DBNAME TSNAME DSNUM ICTYPE ICDATE START_RBA FILESEQNO DEVTYPE IBMREQD DSNAME ICTIME SHRLEVEL DSVOLSER TIMESTAMP ICBACKUP ICUNIT STYPE PIT_RBA GROUP_MEMBER OTYPE LOWDSNUM HIGHDSNUM DSNUCH01: DBNAME.TSNAME.START_RBA.TIMESTAMP DSNUCX01: DSNAME
SYSDATABASE	Contains one row for each database, except for database DSNDB01. NAME CREATOR STGROUP BPOOL DBID IBMREQD CREATEDBY TIMESTAMP TYPE GROUP_MEMBER CREATEDTS ALTEREDTS ENCODING_SCHEME SBCS_CCSID DBCS_CCSID MIXED_CCSID INDEXBP DSNDDH01: NAME DSNDDX02: GROUP_MEMBER

Catalog tables

Table 1 (Page 4 of 12). List of Catalog Tables

Catalog Table Name	Column Names and Indexes
SYSDATATYPES	Contains one row for each distinct type defined to the system. SCHEMA OWNER NAME CREATEDBY SOURCESCHEMA SOURCETYPE METATYPE DATATYPEID SOURCETYPEID LENGTH SCALE SUBTYPE CREATEDTS ENCODING_SCHEME IBMREQD REMARKS DSN0DX01: SCHEMA.NAME DSNODX02: DATATYPEID
SYSDBAUTH	Records the privileges held by users over databases. GRANTOR GRANTEE NAME DATEGRANTED TIMEGRANTED AUTHHOWGOT CREATETABAUTH CREATETSAUTH DBADMAUTH DBCTRLAUTH DBMAINTAUTH DISPLAYDBAUTH DROPAUTH IMAGCOPYAUTH LOADAUTH REORGAUTH RECOVERDBAUTH REPAIRAUTH STARTDBAUTH STATSAUTH STOPAUTH IBMREQD GRANTEDTS DSNADH01: GRANTEE.NAME DSNADX01: GRANTOR.NAME
SYSDBRM	Contains one row for each DBRM of each application plan. NAME TIMESTAMP PDSNAME PLNAME PLCREATOR PRECOMPTIME PRECOMPDATE QUOTE COMMA HOSTLANG IBMREQD CHARSET MIXED DEC31 VERSION PRECOMPTS
SYSDUMMY1	Contains one row. The table is used for SQL statements in which a table reference is required, but the contents of the table are not important. IBMREQD
SYSFIELDS	Contains one row for every column that has a field procedure. TBCREATOR TBNAME COLNO NAME FLDTYPE LENGTH SCALE FLDPROC WORKAREA IBMREQD EXITPARML PARMLIST EXITPARM
SYSFOREIGNKEYS	Contains one row for every column of every foreign key. CREATOR TBNAME RELNAME COLNAME COLNO COLSEQ IBMREQD

Table 1 (Page 5 of 12). List of Catalog Tables

Catalog Table Name	Column Names and Indexes
SYSINDEXES	Contains one row for every index. NAME CREATOR TBNAME TBCREATOR UNIQUERULE COLCOUNT CLUSTERING CLUSTERED DBID OBID ISOBID DBNAME INDEXSPACE NLEAF NLEVELS BPOOL PGSIZE ERASERULE CLOSERULE SPACE IBMREQD CLUSTERRATIO CREATEDBY STATSTIME INDEXTYPE FIRSTKEYCARDF FULLKEYCARDF CREATEDTS ALTEREDTS PIECESIZE COPY COPYLRSN CLUSTERRATIOF DSNDXX01: CREATOR.NAME DSNDXX02: DBNAME.INDEXSPACE DSNDXX03: TBCREATOR.TBNAME.CREATOR.NAME
SYSINDEXPART	Contains one row for each nonpartitioned index and one row for each partition of a partitioned index. PARTITION IXNAME IXCREATOR PQTY SQTY STORTYPE STORNAME VCATNAME LEAFDIST IBMREQD LIMITKEY FREEPAGE PCTFREE SPACE STATSTIME GBPCACHE FAROFFPOSF NEAROFFPOSF CARDF SECQTYI IPREFIX ALTEREDTS DSNDRX01: IXCREATOR.IXNAME.PARTITION DSNDRX01: STORNAME
SYSINDEXSTATS	Contains one row for each partition of a partitioned index. FIRSTKEYCARD FULLKEYCARD NLEAF NLEVELS CLUSTERRATIO STATSTIME IBMREQD PARTITION OWNER NAME KEYCOUNT FIRSTKEYCARDF FULLKEYCARDF KEYCOUNTF CLUSTERRATIOF DSNTXX01: OWNER.NAME.PARTITION
SYSKEYS	Contains one row for each column of an index key. IXNAME IXCREATOR COLNAME COLNO COLSEQ ORDERING IBMREQD DSNDKX01: IXCREATOR.IXNAME.COLNAME
SYSLOBSTATS	Contains one row for each LOB table space. STATSTIME AVGSIZE FREESPACE ORGRATIO DBNAME NAME IBMREQD DSNLNX01: DBNAME.NAME

Catalog tables

Table 1 (Page 6 of 12). List of Catalog Tables

Catalog Table Name	Column Names and Indexes
SYSPACKAGE	Contains a row for every package. LOCATION COLLID NAME CONTOKEN OWNER CREATOR TIMESTAMP BINDTIME QUALIFIER PKSIZE AVGSIZE SYSENTRIES VALID OPERATIVE VALIDATE ISOLATION RELEASE EXPLAIN QUOTE COMMA HOSTLANG CHARSET MIXED DEC31 DEFERPREP SQLERROR REMOTE PCTIMESTAMP IBMREQD VERSION PDSNAME DEGREE GROUP_MEMBER DYNAMICRULES REOPTVAR DEFERPREPARE KEEPDYNAMIC PATHSCHEMAS TYPE DBPROTOCOL FUNCTIONTS OPTHINT DSNKKX01: LOCATION.COLLID.NAME.VERSION DSNKKX02: LOCATION.COLLID.NAME.CONTOKEN
SYSPACKAUTH	Records the privileges held by users over packages. GRANTOR GRANTEE LOCATION COLLID NAME TIMESTAMP GRANTEETYPE AUTHHOWGOT BINDAUTH COPYAUTH EXECUTEAUTH IBMREQD DSNKAX01: GRANTOR.LOCATION.COLLID.NAME DSNKAX02: GRANTEE.LOCATION.COLLID.NAME.BINDAUTH. COPYAUTH.EXECUTEAUTH DSNKAX03: LOCATION.COLLID.NAME
SYSPACKDEP	Records the dependencies of packages on local tables, views, synonyms, table spaces, indexes, and aliases. BNAME BQUALIFIER BTYPE DLOCATION DCOLLID DNAME DCONTOKEN IBMREQD DOWNER DTYPE DSNKDX01: DLOCATION.DCOLLID.DNAME.DCONTOKEN DSNKDX02: BQUALIFIER.BNAME.BTYPE DSNKDX02: BQUALIFIER.BNAME.BTYPE.DTYPE
SYSPACKLIST	Contains one or more rows for every local application plan bound with a package list. PLANNAME SEQNO LOCATION COLLID NAME TIMESTAMP IBMREQD DSNKLX01: LOCATION.COLLID.NAME DSNKLX02: PLANNAME.SEQNO.LOCATION.COLLID.NAME
SYSPACKSTMT	Contains one or more rows for each statement in a package. LOCATION COLLID NAME CONTOKEN SEQNO STMTNO SECTNO BINDERROR IBMREQD VERSION STMT ISOLATION STATUS ACCESSPATH STMTNOI SECTNOI DSNKSX01: LOCATION.COLLID.NAME.CONTOKEN.SEQNO

Table 1 (Page 7 of 12). List of Catalog Tables

Catalog Table Name	Column Names and Indexes
SYSPARMS	Contains one row for each parameter of a routine or multiple rows for table parameters (one for each column of the table)..
	SCHEMA OWNER NAME SPECIFICNAME ROUTINETYPE CAST_FUNCTION PARMNAME ROUTINEID ROWTYPE ORDINAL TYPESCHEMA TYPENAME DATATYPEID SOURCETYPEID LOCATOR TABLE TABLE_COLNO LENGTH SCALE SUBTYPE CCSID CAST_FUNCTION_ID ENCODING_SCHEME IBMREQD DSNOPX01: SCHEMA.SPECIFICNAME.ROUTINETYPE.ROWTYPE.ORDINAL DSNOPX02: TYPESCHEMA.TYPENAME.ROUTINETYPE.CAST_FUNCTION. OWNER.SCHEMA.SPECIFICNAME DSNOPX03: TYPESCHEMA.TYPENAME
SYSPKSYSTEM	Contains zero or more rows for every package.
	LOCATION COLLID NAME CONTOKEN SYSTEM ENABLE CNAME IBMREQD DSNKYX01: LOCATION.COLLID.NAME.CONTOKEN.SYSTEM.ENABLE
SYSPLAN	Contains one row for each application plan.
	NAME CREATOR BINDDATE VALIDATE ISOLATION VALID OPERATIVE BINDTIME PLSIZE IBMREQD AVGSIZE ACQUIRE RELEASE EXPLAN EXPREDICATE BOUNDBY QUALIFIER CACHESIZE PLENTRIES DEFERPREP CURRENTSERVER SYSENTRIES DEGREE SQLRULES DISCONNECT GROUP_MEMBER DYNAMICRULES BOUNDTS REOPTVAR KEEPDYNAMIC PATHSCHEMAS DBPROTOCOL FUNCTIONTS OPTHINT DSNPPH01: NAME
SYSPLANAUTH	Records the privileges held by users over application plans.
	GRANTOR GRANTEE NAME DATEGRANTED TIMEGRANTED AUTHHOWGOT BINDAUTH EXECUTEAUTH IBMREQD GRANTEDTS DSNAPH01: GRANTEE.NAME.EXECUTEAUTH DSNAPX01: GRANTOR
SYSPLANDEP	Records the dependencies of plans on tables, views, aliases, synonyms, table spaces, and indexes.
	BNAME BCREATOR BTYPE DNAME IBMREQD DSNGGX01: BCREATOR.BNAME.BTYPE
SYSPLSYSTEM	Contains zero or more rows for every plan.
	NAME SYSTEM ENABLE CNAME IBMREQD DSNKPX01: NAME.SYSTEM.ENABLE

Catalog tables

Table 1 (Page 8 of 12). List of Catalog Tables

Catalog Table Name	Column Names and Indexes
SYSPROCEDURES	Contains one row for every stored procedure. PROCEDURE AUTHID LUNAME LOADMOD LINKAGE COLLID LANGUAGE ASUTIME STAYRESIDENT IBMREQD RUNOPTS PARMLIST RESULT_SETS WLM_ENV PGM_TYPE EXTERNAL_SECURITY COMMIT_ON_RETURN DSNKCX01: PROCEDURE.AUTHID.LUNAME
SYSRELS	Contains one row for every referential constraint. CREATOR TBNAME RELNAME REFTBNAME REFTBCREATOR COLCOUNT DELETERULE IBMREQD RELOBID1 RELOBID2 TIMESTAMP IXOWNER IXNAME DSNDLX01: REFTBCREATOR.REFTBNAME
SYSRESAUTH	Records USE privileges for buffer pools, storage groups, and table spaces, and CREATE IN and PACKADM ON privileges for collections. GRANTOR GRANTEE QUALIFIER NAME AUTHHOWGOT OBTYPE DATEGRANTED TIMEGRANTED USEAUTH IBMREQD GRANTEDTS DSNAGH01: GRANTEE.QUALIFIER.NAME.OBTYPE DSNAGX01: GRANTOR.QUALIFIER.NAME.OBTYPE
SYSROUTINEAUTH	Records the privileges that are held by users on routines. (A routine can be a user-defined function, cast function generated for a distinct type, or stored procedure.) GRANTOR GRANTEE SCHEMA SPECIFICNAME GRANTEDTS ROUTINETYPE GRANTEETYPE AUTHHOWGOT EXECUTEAUTH COLLID CONTOKEN IBMREQD DSNOAX01: GRANTOR.SCHEMA.SPECIFICNAME.ROUTINETYPE.GRANTEETYPE. EXECUTEAUTH DSNOAX02: GRANTEE.SCHEMA.SPECIFICNAME.ROUTINETYPE.GRANTEETYPE. EXECUTEAUTH DSNOAX03: SCHEMA.SPECIFICNAME.ROUTINETYPE

Table 1 (Page 9 of 12). List of Catalog Tables

Catalog Table Name	Column Names and Indexes
SYSROUTINES	Contains one row for each routine. (A routine can be a user-defined function, cast function generated for a distinct type, or stored procedure.) SCHEMA OWNER NAME ROUTINETYPE CREATEDBY SPECIFICNAME ROUTINEID RETURN_TYPE ORIGIN FUNCTION_TYPE PARM_COUNT LANGUAGE COLLID SOURCESCHEMA SOURCESPECIFIC DETERMINISTIC EXTERAL_ACTION NULL_CALL CAST_FUNCTION SCRATCHPAD SCRATCHPAD_LENGTH FINAL_CALL PARALLEL PARAMETER_STYLE FENCED SQL_DATA_ACCESS DBINFO STAYRESIDENT ASUTIME WLM_ENVIRONMENT WLM_ENV_FOR_NESTED PROGRAM_TYPE EXTERNAL_SECURITY COMMIT_ON_RETURN RESULT_SETS LOBCOLUMNS CREEATEDTS ALTEREDTS IBMREQD PARM1 PARM2 PARM3 ... PARM30 IOS_PER_INVOC INSTS_PER_INVOC INITIAL_IOS INITIAL_INSTS CARDINALITY RESULT_COLS EXTERNAL_NAME PARM_SIGNATURE RUNOPTS REMARKS DSNOFX01: NAME.PARM_COUNT.PARM_SIGNATURE.ROUTINETYPE. SCHEMA.PARM1.PARM2.PARM3.PARM4.PARM5.PARM6.PARM7.PARM8. PARM9.PARM10.PARM11.PARM12.PARM14.PARM15.PARM16.PARM17. PARM18.PARM19.PARM20.PARM21.PARM22.PARM23.PARM24.PARM25. PARM26.PARM27.PARM28.PARM29.PARM30 DSNOFX02: SCHEMA.SPECIFICNAME.ROUTINETYPE DSNOFX03: NAME.SCHEMA.CAST_FUNCTION.PARM_COUNT.PARM_SIGNATURE. PARM1 DSNOFX04: ROUTINEID DSNOFX05: SOURCESCHEMA.SOURCESPECIFIC.ROUTINETYPE DSNOFX06: SCHEMA.NAME.ROUTINETYPE.PARM_COUNT
SYSSCHEMAUTH	Contains one or more rows for each user that is granted a privilege on a particular schema in the database. GRANTOR GRANTEE SCHEMANAME AUTHHOWGOT CREATINAUTH ALTERINAUTH DROPINAUTH GRANTEDTS IBMREQD DSNSKX01: GRANTEE.SCHEMANAME DSNSKX02: GRANTOR
SYSSTMT	Contains one or more rows for each SQL statement of each DBRM. NAME PLNAME PLCREATOR SEQNO STMTNO SECTNO IBMREQD TEXT ISOLATION STATUS
SYSSTOGROUP	Contains one row for each storage group. NAME CREATOR VCATNAME SPACE PCDATE IBMREQD CREATEDBY STATSTIME CREATEDTS ALTEREDTS DSNSSH01: NAME

Catalog tables

Table 1 (Page 10 of 12). List of Catalog Tables

Catalog Table Name	Column Names and Indexes
SYSSTRINGS	Contains information about character conversion. INCCSID OUTCCSID TRANSTYPE ERRORBYTE SUBBYTE TRANSPROC IBMREQD TRANSTAB DSNSSX01: OUTCCSID.INCCSID.IBMREQD
SYSSYNONYMS	Contains one row for each synonym of a table or view. NAME CREATOR TBNAME TBCREATOR IBMREQD CREATEDBY CREATEDTS DSNDYX01: CREATOR.NAME
SYSTABAUTH	Records the privileges held by users on tables and views. GRANTOR GRANTEE GRANTEETYPE DBNAME SCREATOR STNAME TCREATOR TTNAME AUTHHOWGOT DATEGRANTED TIMEGRANTED UPDATECOLS ALTERAUTH DELETEAUTH INDEXAUTH INSERTAUTH SELECTAUTH UPDATEAUTH IBMREQD COLLID CONTOKEN REFERENCESAUTH REFCOLS GRANTEDTS TRIGGERAUTH DSNATX01: GRANTOR DSNATX02: GRANTEE.TCREATOR.TTNAME.GRANTEETYPE. UPDATECOLS.ALTERAUTH.DELETEAUTH.INDEXAUTH. INSERTAUTH.SELECTAUTH.UPDATEAUTH.CAPTUREAUTH. REFERENCEAUTH.REFCOLS.TRIGGERAUTH
SYSTABLEPART	Contains one row for each nonpartitioned table space and one row for each partition of a partitioned table space. PARTITION TSNAME DBNAME IXNAME IXCREATOR PQTY SQTY STORTYPE STORNAME VCATNAME CARD FARINDREF NEARINDREF PERCACTIVE PERCDROP IBMREQD LIMITKEY FREEPAGE PCTFREE CHECKFLAG CHECKRID SPACE COMPRESS PAGESAVE STATSTIME GBPCACHE CHECKRID5B TRACKMOD EPOCH SECQTYI CARDF IPREFIX ALTEREDTS DSNDOX01: DBNAME.TSNAME.PARTITION DSNDOX01: STORNAME

Table 1 (Page 11 of 12). List of Catalog Tables

Catalog Table Name	Column Names and Indexes
SYSTABLES	Contains one row for each table, view, or alias. NAME CREATOR TYPE DBNAME TSNAME DBID OBID COLCOUNT EDPROC VALPROC CLUSTERTYPE CARD NPAGES PCTPAGES IBMREQD REMARKS PARENTS CHILDREN KEYCOLUMNS RECLENGTH STATUS KEYOBID LABEL CHECKFLAG CHECKRID AUDITING CREATEDBY LOCATION TBCREATOR TBNAME CREATEDTS ALTEREDTS DATACAPTURE RBA1 RBA2 PCTROWCOMP STATSTIME CHECKS CARDF CHECKRID5B ENCODING_SCHEME TABLESTATUS DSNDTX01: CREATOR.NAME DSNDTX02: DBID.OBID.CREATOR.NAME
SYSTABLESPACE	Contains one row for each table space. NAME CREATOR DBNAME DBID OBID PSID BPOOL PARTITIONS LOCKRULE PGSIZE ERASERULE STATUS IMPLICIT NTABLES NACTIVE CLOSERULE SPACE IBMREQD SEGSIZE CREATEDBY STATSTIME LOCKMAX TYPE CREATEDTS ALTEREDTS ENCODING_SCHEME SBCS_CCSID DBCS_CCSID MIXED_CCSID MAXROWS LOCKPART LOG NACTIVEF DSSIZE DSNDSX01: DBNAME.NAME
SYSTABLESTATS	Contains one row for partition of a partitioned table space. CARD NPAGES PCTPAGES NACTIVE PCTROWCOMP STATSTIME IBMREQD DBNAME TSNAME PARTITION OWNER NAME CARDF DSNTTX01: OWNER.NAME.PARITION
SYSTRIGGERS	Contains one row for each trigger.. NAME SCHEMA SEQNO DBID OBID OWNER CREATEDBY TBNAME TBOWNER TRIGTIME TRIGEVENT GRANULARITY CREATEDTS IBMREQD TEXT REMARKS DSNOTX01: SCHEMA.NAME.SEQNO DSNOTX02: TBOWNER.TBNAME

Catalog tables

Table 1 (Page 12 of 12). List of Catalog Tables

Catalog Table Name	Column Names and Indexes
SYSUSERAUTH	Records the system privileges held by users. GRANTOR GRANTEE DATEGRANTED TIMEGRANTED AUTHHOWGOT BINDADDAUTH BSDSAUTH CREATEDBAAUTH CREATEDBCAUTH CREATESGAUTH DISPLAYAUTH RECOVERAUTH STOPALLAUTH STOSPACEAUTH SYSADMAUTH SYSOPRAUTH TRACEAUTH IBMREQD MON1AUTH MON2AUTH CREATEALIASAUTH SYSCTRLAUTH BINDAGENTAUTH ARCHIVEAUTH GRANTEDTS CREATETMTABAUTH DSNAUH01: GRANTEE.GRANTEDTS DSNAUX01: GRANTOR
SYSVIEWDEP	Records the dependencies of views on tables and other views. BNAME BCREATOR BTYPE DNAME DCREATOR IBMREQD BSCHEMA DSNGGX02: BCREATOR.BNAME.BTYPE DSNGGX03: BSCHEMA.BNAME.BTYPE
SYSVIEWS	Contains one or more rows for each view. NAME CREATOR SEQNO CHECK IBMREQD TEXT PATHSCHEMAS DSNVVX01: CREATOR.NAME.SEQNO
SYSVOLUMES	Contains one row for each volume of each storage group. SGNAME SGCREATOR VOLID IBMREQD
USERNAMES	Each row in the table is used to carry out outbound ID translation, or inbound ID translation and “come from” checking TYPE AUTHID LINKNAME NEWAUTHID PASSWORD IBMREQD DSNFEX01: TYPE.AUTHID.LINKNAME

Tables used by EXPLAIN and the resource limit facility

This section lists the tables that are used by DB2's EXPLAIN function and the resource limit facility (governor). Along with the name of the table is a description of its purpose and a list of the names of its columns (to be read from left to right, by rows).

Create the DSN_FUNCTION_TABLE, DSN_STATEMNT_TABLE, and PLAN_TABLE tables using SQL statements similar to those contained in sample job DSNTESC. For a description of the columns in the PLAN_TABLE or DSN_STATEMNT_TABLE, see "Improving Query Performance Using EXPLAIN" in *DB2 Administration Guide*. For a description of the columns in the DSN_FUNCTION_TABLE, see Section 4 of *DB2 Application Programming and SQL Guide*.

Create resource limit specification tables (RLSTs) using an SQL statement similar to the one contained in sample job DSNTIJSG. For more information on the resource limit specification table (RLST), see "Improving Resource Utilization" in *DB2 Administration Guide*.

Table 2 (Page 1 of 2). List of Other DB2 Tables

Table Name	Column Names
DSN_FUNCTION_TABLE	The SQL EXPLAIN statement can obtain and place information in a user-created *function table.* EXPLAIN populates this table with information about how DB2 resolves the user-defined functions that are referred to in the explainable statement. QUERYNO APPLNAME PROGNAME COLLID GROUP_MEMBER EXPLAIN_TIME SCHEMA_NAME FUNCTION_NAME SPEC_FUNC_ID FUNCTION_TYPE VIEW_CREATOR VIEW_NAME PATH FUNCTION_TEXT
DSN_STATEMNT_TABLE	The SQL EXPLAIN statement can obtain and place information in a user-created *statement table.* EXPLAIN populates this table with information about the estimated cost of executing the explainable statement. Each row in the table provides a cost estimate, in service units and milliseconds, of processing an explainable statement. QUERYNO APPLNAME PROGNAME COLLID GROUP_MEMBER EXPLAIN_TIME STMT_TYPE COST_CATEGORY PROCMS PROCSU REASON

EXPLAIN and resource limit facility table reference

Table 2 (Page 2 of 2). List of Other DB2 Tables

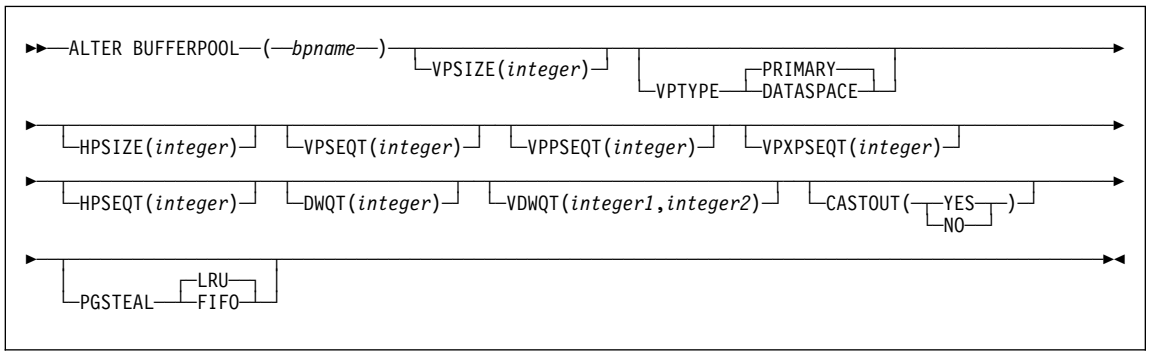
Table Name	Column Names
PLAN_TABLE	The SQL EXPLAIN statement can obtain and place information in a user-created *plan table.* EXPLAIN populates this table with information about access path selection for an *explainable statement.* Each row in a plan table describes a step in the execution of a query or subquery in an explainable statement. The column values for the row identify, among other things, the query or subquery, the tables involved, and the method used to carry out the step. QUERYNO QBLOCKNO APPLNAME PROGNAME PLANNO METHOD CREATOR TNAME TABNO ACCESSTYPE MATCHCOLS ACCESSCREATOR ACCESSNAME INDEXONLY SORTN_UNIQ SORTN_JOIN SORTN_ORDERBY SORTN_GROUPBY SORTC_UNIQ SORTC_JOIN SORTC_ORDERBY SORTC_GROUPBY TSLOCKMODE TIMESTAMP REMARKS PREFETCH COLUMN_FN_EVAL MIXOPSEQ VERSION COLLID **Note:** The following nine columns, from ACCESS_DEGREE through CORRELATION_NAME, contain the null value if the plan or package was bound using a plan table with fewer than 43 columns. Otherwise, each of them contains a null value if the method it refers to does not apply. ACCESS_DEGREE ACCESS_PGROUP_ID JOIN_DEGREE JOIN_PGROUP_ID SORTC_PGROUP_ID SORTN_PGROUP_IID PARALLELISM_MODE MERGE_JOIN_COLS CORRELATION_NAME PAGE_RANGE JOIN_TYPE GROUP_MEMBER IBM_SERVICE_DATA WHEN_OPTIMIZE QBLOCK_TYPE BIND_TIME OPTHINT HINT_USED PRIMARY_ACCESSTYPE
Resource Limit Specification Table (RLST)	Provides governing information to DB2. The name of the table is *authid*.DSNRLST*xx*, where *xx* is any 2-character alphanumeric value, and *authid* is specified when DB2 is installed. Because the two characters *xx* must be entered as part of the START command, they must be alphanumeric—no special or DBCS characters. AUTHID PLANNAME ASUTIME LUNAME RLFFUNC RLFBIND RLFCOLLN RLFPKG RLFASUERR RLFASUWARN RLF_CATEGORY_B

Command alphabetic reference

This section contains syntax diagrams and examples of DB2 commands organized alphabetically by name. For more information, see *DB2 Command Reference*.

-ALTER BUFFERPOOL (DB2)

Syntax



Examples

Example 1: Set the virtual buffer pool and hiperpool for BP0 to 1000 and 10000 buffers, respectively.

```
-ALTER BUFFERPOOL(BP0) VPSIZE(1000) HPSIZE(10000)
```

Example 2: Set the sequential steal threshold of the virtual buffer pool for BP0 to 75 percent of the virtual pool size, while disabling caching of sequentially accessed pages in the hiperpool.

```
-ALTER BUFFERPOOL(BP0) VPSEQT(75) HPSEQT(0)
```

Example 3: Set the hiperpool size for BP4 to 10000 buffers and explicitly specify that cached data in the hiperpool for BP4 can be discarded.

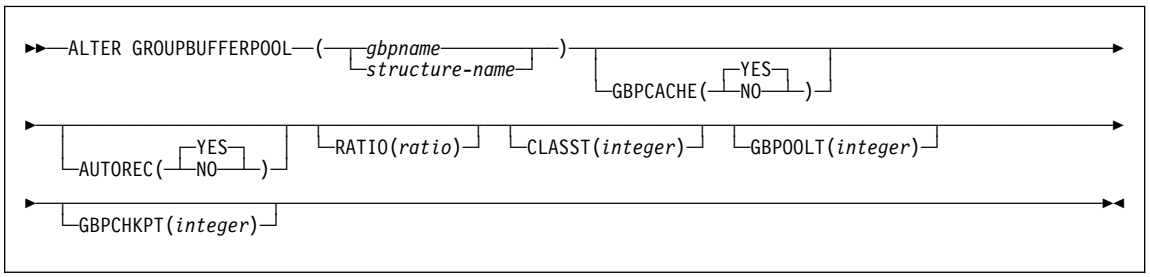
```
-ALTER BUFFERPOOL(BP4) HPSIZE(10000) CASTOUT(YES)
```

Example 4: Delete BP1. Be very careful using this option because when you specify a 0 size for an active buffer pool, DB2 quiesces all current database access and fails all subsequent page set open requests.

```
-ALTER BUFFERPOOL(BP1) VPSIZE(0)
```

-ALTER GROUPBUFFERPOOL (DB2)

Syntax



Examples

Example 1: For group buffer pool 0, change the ratio of directory entries to data pages to 1 directory entry for every data page. The RATIO specification becomes effective at the next allocation of the group buffer pool.

```
-DB1G ALTER GROUPBUFFERPOOL (GBP0) RATIO(1)
```

Example 2: For group buffer pool 2, change the class castout threshold to 5% and the group buffer pool castout threshold to 30%. The new values take effect immediately.

```
-DB1G ALTER GROUPBUFFERPOOL (GBP2) CLASST(5) GBPOOLT(30)
```

Example 3: Assume that the DB2 group name is DSNCAT. For group buffer pool 3, change the class castout threshold to 5%. The new value takes effect immediately. Because the group name is DSNCAT, the coupling facility structure name is DSNCAT_GBP3. Also, in the event of a structure failure, the AUTOREC(YES) option enables DB2 to automatically recover the page sets and partitions that are in a GRECP status or that have pages on the logical page list.

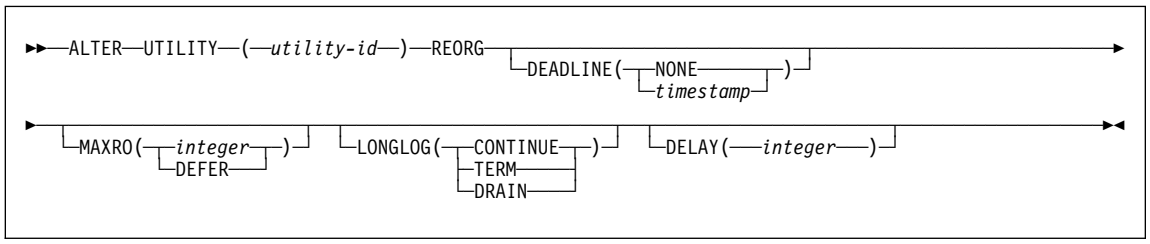
```
-DB1G ALTER GROUPBUFFERPOOL (DSNCAT_GBP3) CLASST(5) AUTOREC(YES)
```

Example 4: For group buffer pool 32K, change the GBP checkpoint frequency to 5 minutes. The new value takes effect immediately. Here, with AUTOREC(NO) specified, you are in effect taking control of the recovery process rather than DB2 in the event of a structure failure. You might choose to do this to determine what pagesets or partitions are in a GRECP status or that have pages on the logical page list and before entering the START DATABASE command to enable DB2 to recover the data with the options you specify.

```
-DB1G ALTER GROUPBUFFERPOOL (GBP32K) GBPCHKPT(5) AUTOREC(NO)
```

-ALTER UTILITY (DB2)

Syntax



Example

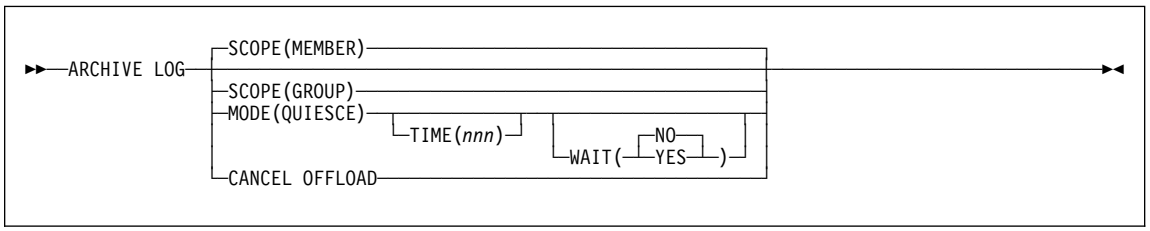
Example: The following example alters the execution of the REORG utility for the utility job step whose utility identifier is REORGEMP:

- MAXRO(240) changes the maximum tolerable time for the last iteration of log processing to 240 seconds (4 minutes).
- LONGLOG DRAIN changes the action that DB2 performs (if reorganization's reading of the log is not catching up to applications' writing of the log quickly enough) to draining of the write claim class.
- DELAY was not specified and therefore, the example does not change the existing delay between sending of the LONGLOG message to the console and performing the action specified by LONGLOG.
- DEADLINE was not specified and the example does not change the existing deadline (if any) of the last iteration of log processing.

```
-ALTER UTILITY (REORGEMP) REORG MAXRO(240) LONGLOG DRAIN
```

-ARCHIVE LOG (DB2)

Syntax



Examples

Example 1: Truncate the current active log data sets and initiate an asynchronous job to off-load the truncated data sets. No quiesce processing occurs.

```
-ARCHIVE LOG
```

Example 2: Initiate a quiesce period. If all DB2 update activity is stopped within this period, truncate the current active log data set and switch to the next available active log data set. Let the value in the field QUIESCE PERIOD of installation panel DSNTIPA determine the length of the quiesce period. The MODE(QUIESCE) processing is asynchronous.

If the DB2 subsystem can successfully block all update activity before the quiesce period ends, it proceeds to the next processing step. If the quiesce time period is insufficient to successfully quiesce the DB2 subsystem, the active log data sets are not truncated and the archive does not occur.

```
-ARCHIVE LOG MODE(QUIESCE)
```

Example 3: Initiate a quiesce period. If all DB2 update activity is stopped within this period, truncate the current active log data set and switch to the next available active log data set. The maximum length of the quiesce processing period is seven minutes (420 seconds) and the processing is synchronous for the entire seven minutes.

If the DB2 subsystem can successfully block all update activity before the quiesce period ends, it proceeds to the next processing step. If the quiesce time period is insufficient to successfully quiesce the DB2 subsystem, the active log data sets are not truncated and the archive does not occur.

```
-ARCHIVE LOG MODE(QUIESCE) WAIT(YES) TIME(420)
```

Example 4: In a data sharing environment, initiate a quiesce period for all members of the data sharing group. If all DB2 update activity is stopped within this period, truncate the current active log data set and switch to the next available active log data set. Specify a quiesce time period of 10 minutes (600 seconds) to override the value in the field QUIESCE PERIOD of installation panel DSNTIPA for member DB1G. If the update

activity has not quiesced after the 10 minute quiesce period, the command fails and new update activity is allowed to proceed.

```
-DB1G ARCHIVE LOG MODE(QUIESCE) TIME(600)
```

Example 5: In a data sharing environment, truncate the active log data sets for group member DB2G and initiate an asynchronous job to off-load the truncated data sets, without any quiesce processing. In this example, SCOPE(MEMBER) is used by default.

```
-DB2G ARCHIVE LOG
```

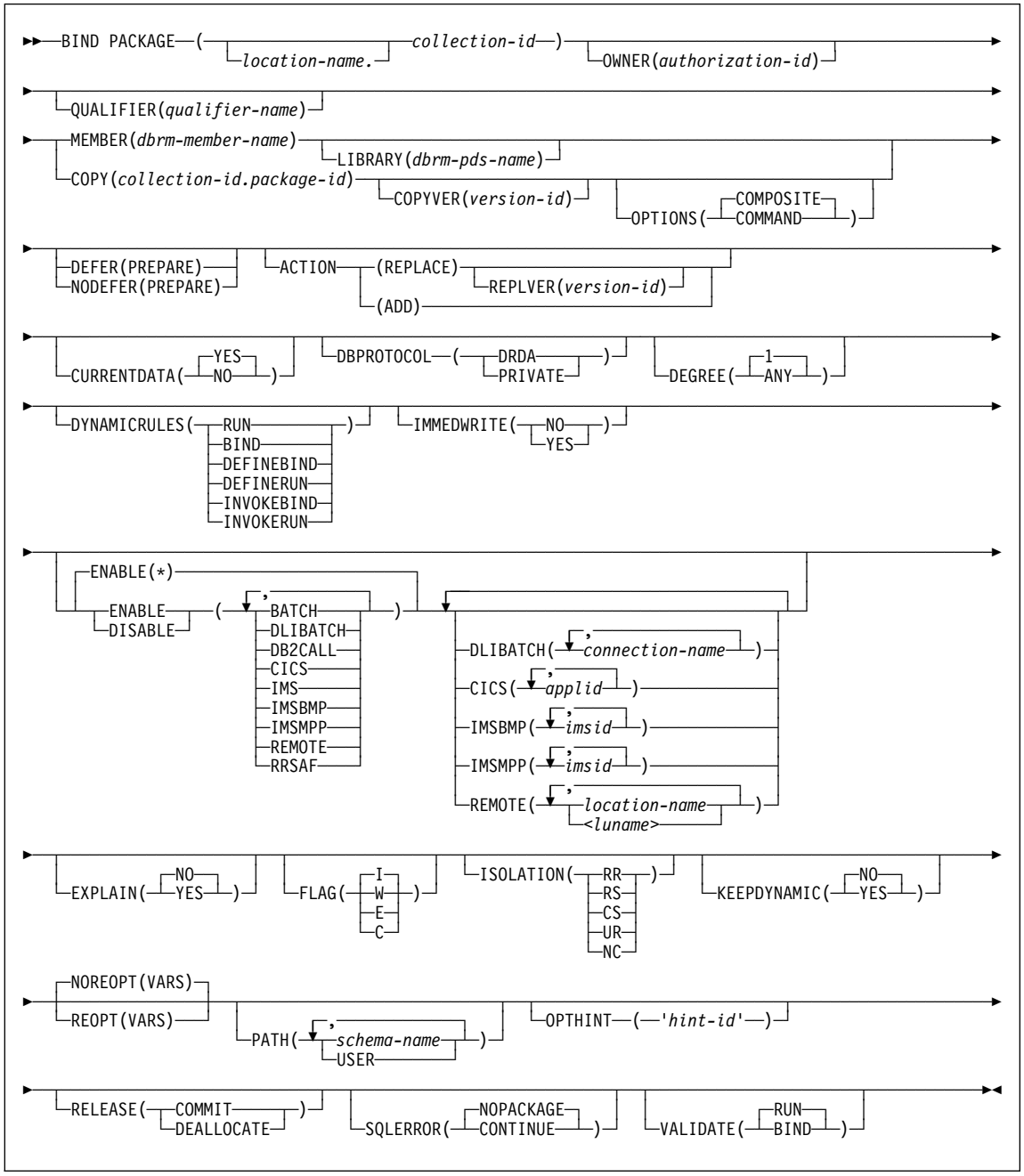
Example 6: In a data sharing environment, truncate the data sets for all members of the data sharing group and initiate an asynchronous job to off-load the truncated data sets, without any quiesce processing.

```
-DB2G ARCHIVE LOG SCOPE(GROUP)
```

BIND PACKAGE (DSN)

Syntax

BIND PACKAGE (DSN)



Examples

Example 1: Replace version APRIL_VERSION of package TEST.DSN8BC61 at local location USIBMSTODB22 with another version of the package. The new version (or it could be the same) is in the DBRM DSN8BC61. If the DBRM contains no version ID, the version ID of the package defaults to the empty string. The package runs only from the TSO BATCH environment, and from the CICS environment if the connection ID is CON1. The name PRODUCTN qualifies all unqualified table, view, alias and index names.

```
BIND PACKAGE (USIBMSTODB22.TEST) -
  MEMBER (DSN8BC61) -
  ACTION (REPLACE) REPLVER (APRIL_VERSION) -
  QUALIFIER (PRODUCTN) -
  ENABLE (BATCH, CICS) CICS (CON1)
```

Example 2: UR isolation acquires almost no locks. It is fast and causes little contention, but it reads uncommitted data. Do not use ISOLATION(UR) unless you are sure that your applications and end users can accept the logically inconsistent data that can occur, such as in the case of this example.

Assume that a supervisor routinely executes SQL statements using SPUFI to check the status of parts as they go through the assembly process and to update a table with the results of her inspection. She does not need to know the exact status of the parts; a small margin of error is acceptable.

The supervisor queries the status of the parts from a production table called ASSEMBLY-STATUS and makes the updates in a non-production table called REPORTS. She uses the SPUFI option AUTOCOMMIT NO and has the habit of leaving data on the screen while she performs other tasks.

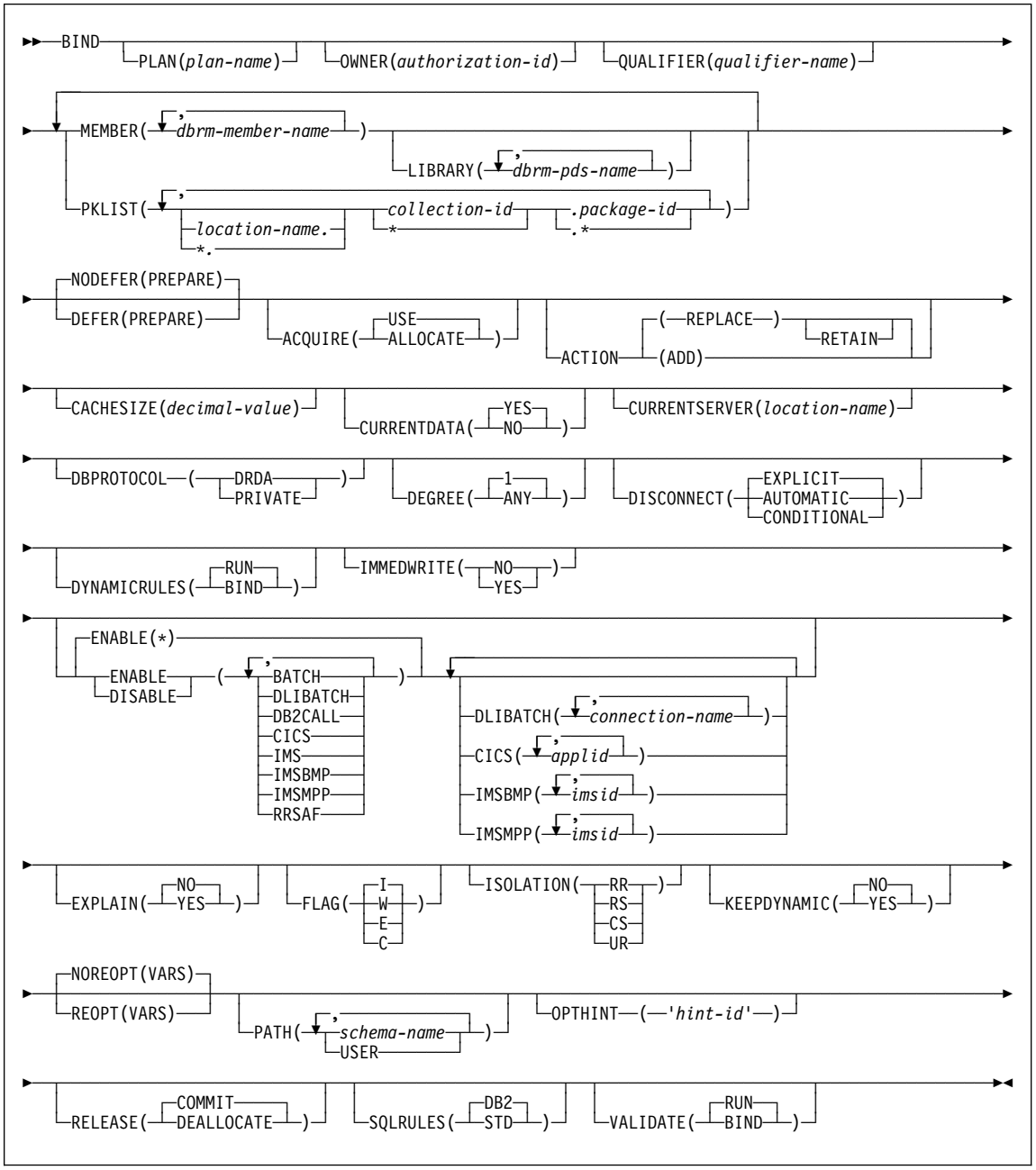
If the supervisor executes a version of SPUFI that is bound with ISOLATION(UR), the query for the status of the parts executes without acquiring locks using UR isolation level and the update executes using CS isolation level. Thus, the query does not inadvertently hold locks in the production table interfering with the production jobs, and the supervisor has data good enough for her purposes.

The SPUFI application is bound as follows:

```
BIND PACKAGE(DSNESPUR) -
  COPY(DSNESPCS.DSNESM68) -
  ACTION(ADD) -
  ISOLATION(UR)
```

BIND PLAN (DSN)

Syntax


BIND
PLAN(plan-name)
OWNER(authorization-id)
QUALIFIER(qualifier-name)
MEMBER(, dbrm-member-name)
LIBRARY(, dbrm-pds-name)
PKLIST(, location-name. *. collection-id * .package-id .*)
NODEFER(PREPARE)
DEFER(PREPARE)
ACQUIRE(USE ALLOCATE)
ACTION (REPLACE) RETAIN (ADD)
CACHESIZE(decimal-value)
CURRENTDATA(YES NO)
CURRENTSERVER(location-name)
DBPROTOCOL (DRDA PRIVATE)
DEGREE(1 ANY)
DISCONNECT(EXPLICIT AUTOMATIC CONDITIONAL)
DYNAMICRULES(RUN BIND)
IMMEDWRITE(NO YES)
ENABLE(*)
ENABLE DISABLE (, BATCH DLIBATCH DB2CALL CICS IMS IMSBMP IMSMPP RRSAF)
DLIBATCH(, connection-name)
CICS(, applid)
IMSBMP(, imsid)
IMSMPP(, imsid)
EXPLAIN(NO YES)
FLAG(I W E C)
ISOLATION(RR RS CS UR)
KEEPDYNAMIC(NO YES)
NOREOPT(VARS)
REOPT(VARS)
PATH(, schema-name USER)
OPTHINT ('hint-id')
RELEASE(COMMIT DEALLOCATE)
SQLRULES(DB2 STD)
VALIDATE(RUN BIND)

Examples

Example 1: This subcommand creates a new plan called IMSONLY. The SQL statements for the plan are in the DBRM member DSN8BC61. An ISOLATION level of cursor stability (CS) provides maximum concurrency when you run the plan, and protects database values only while the program uses them. DEPTM92 owns the plan, but PRODUCTN qualifies any unqualified table, view, index, and alias names referenced in the DBRM.

A cache size of 0 indicates that users will not run the plan repeatedly. Caching the names of users authorized to run the plan helps only when the same user runs the plan repeatedly while it is in the EDM pool. Since this is not the case with this plan, there is no need to reserve space in the EDM pool for a cache that the plan does not use.

The option ENABLE(IMS) runs the plan only from an IMS environment (DLI Batch, BMP and MPP). If you attempt to run the plan from another environment, such as TSO Batch, the plan allocation fails.

```
BIND PLAN(IMSONLY) -
  MEMBER(DSN8BC61) -
  ACTION(ADD) -
  ISOLATION(CS) -
  OWNER(DEPTM92) -
  QUALIFIER(PRODUCTN) -
  CACHESIZE -
  ENABLE(IMS)
```

Example 2: If the DBRM of plan IMSONLY in example 1 contains both embedded and dynamic SQL statements and you want to allow other users to run the plan, you must grant the EXECUTE privilege on plan IMSONLY to those users' authorization IDs. However, because the EXECUTE privilege on a plan is sufficient authority to run embedded SQL statements in a DBRM but is not sufficient authority to run dynamic SQL statements, you must also do one of the following:

- Use the SQL GRANT statement to grant the necessary privileges on the objects (tables, views, aliases, and indexes) referenced in the dynamic SQL statements to the users' authorization IDs, or
- BIND the plan IMSONLY with the option DYNAMICRULES(BIND) as follows:

```
BIND PLAN(IMSONLY) -
  MEMBER(DSN8BC61) -
  ACTION(ADD) -
  ISOLATION(CS) -
  OWNER(DEPTM92) -
  QUALIFIER(PRODUCTN) -
  CACHESIZE(0) -
  ENABLE(IMS) -
  DYNAMICRULES(BIND)
```

To allow other users having only the EXECUTE privilege on a plan to run both the embedded and dynamic SQL statements, you must bind that plan with the option DYNAMICRULES(BIND). When DYNAMICRULES(BIND) is in effect for plan IMSONLY:

- A single authorization ID, the authorization ID for DEPTM92, is used for authorization checking of both the embedded and dynamic SQL statements in the DBRM.
- PRODUCTN is the implicit qualifier of unqualified object names referenced in both the embedded and dynamic SQL statements in the DBRM.

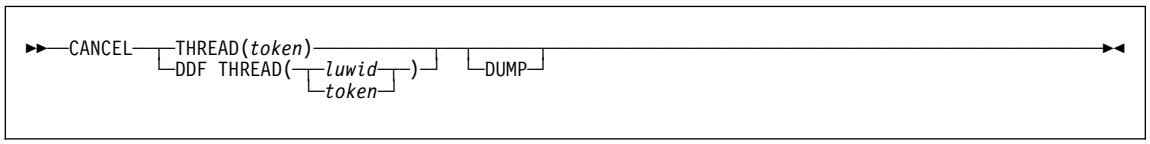
Example 3: This subcommand creates a new plan called CICSONLY. The plan specifies an ISOLATION level of cursor stability (CS). DEPTM12 owns the plan, but TESTSYS qualifies any unqualified table, view, index, and alias names referenced in the DBRM. A cache size of 0 indicates that users will not run the plan repeatedly.

The option ENABLE(CICS) CICS(CON1) runs the plan only from CICS VTAM® node CON1 which is specified in the APPLID parameter of the CICS SIT table. If you attempt to run the plan from another environment or from another CICS VTAM note, the run attempt fails.

```
BIND PLAN(CICSONLY) -
  MEMBER(DSN8BC61) -
  ACTION(ADD) -
  ISOLATION(CS) -
  OWNER(DEPTM12) -
  QUALIFIER(TESTSYS) -
  CACHESIZE(0) -
  ENABLE(CICS) CICS(CON1)
```

-CANCEL THREAD (DB2)

Syntax



Examples

Example 1: To cancel a non-distributed thread whose token you found through -DISPLAY THREAD and to produce a diagnostic dump, issue:

```
-CANCEL THREAD (123) DUMP
```

Example 2: To cancel a distributed thread whose LUWID you found through -DISPLAY THREAD, issue:

```
-CANCEL DDF THREAD (LUDALLAS.DB2SQL1.3042512B6425)
```

Assume that the output from -DISPLAY THREAD shows that the thread-ID and token associated with this LUWID is 45162. You can also cancel this thread by issuing:

```
-CANCEL DDF THREAD (45162)
```

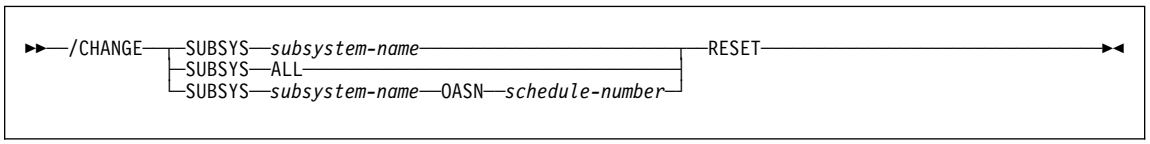
or

```
-CANCEL THREAD (45162)
```

As in the first example, specifying DUMP with any of the commands shown in this example would cause a diagnostic dump to be produced.

/CHANGE (IMS)

Syntax



Examples

Example 1: Reset all indoubt recovery units for subsystem DB2.

```
/CHA SUBSYS DB2 RESET
```

Example 2: Reset all indoubt recovery units for all subsystems.

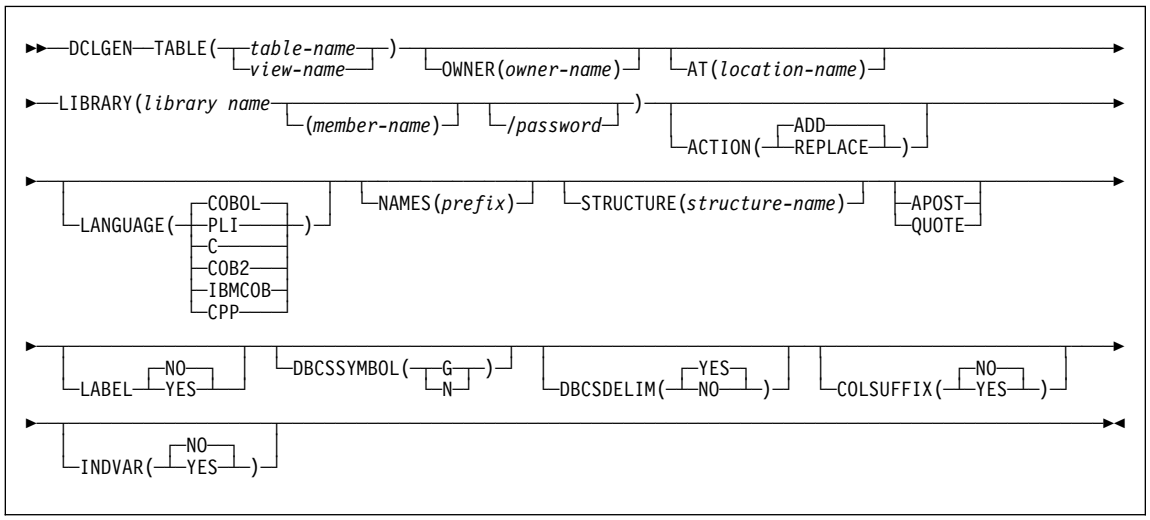
```
/CHA SUBSYS ALL RESET
```

Example 3: Reset indoubt recovery units identified by OASN numbers 99, 685, and 2920 for subsystem DB2.

```
/CHA SUBSYS DB2 OASN 99 685 2920 RESET
```

DCLGEN (DECLARATIONS GENERATOR) (DSN)

Syntax



Examples

Example 1: This example shows the use of the DCLGEN. The statement

```
DCLGEN TABLE(VEMPL) -
       LIBRARY('prefix.SRCLIB.DATA(DSN8MPEM)') -
       LANGUAGE(PLI) -
       APOST
```

produces the following statements in *prefix*.SRCLIB.DATA(DSN8MPEM):

```
/************************************************************************/
/* DCLGEN TABLE(VEMPL) -                                                */
/*        LIBRARY('prefix.SRCLIB.DATA(DSN8MPEM)') -                     */
/*        LANGUAGE(PLI) -                                               */
/*        APOST                                                         */
/* ... IS THE DCLGEN COMMAND THAT MADE THE FOLLOWING STATEMENTS         */
/************************************************************************/
EXEC SQL DECLARE VEMPL TABLE
         ( EMPNO                         CHAR(6) NOT NULL,
           FIRSTNME                      VARCHAR(12) NOT NULL,
           MIDINIT                       CHAR(1) NOT NULL,
           LASTNAME                      VARCHAR(15) NOT NULL,
           WORKDEPT                      CHAR(3) NOT NULL
         ) ;
/************************************************************************/
/* PLI DECLARATION FOR TABLE VEMPL                                      */
/************************************************************************/
DCL 1 DCLVEMPL,
     5 EMPNO    CHAR(6),
     5 FIRSTNME CHAR(12) VAR,
     5 MIDINIT  CHAR(1),
     5 LASTNAME CHAR(15) VAR,
     5 WORKDEPT CHAR(3);
/************************************************************************/
/* THE NUMBER OF COLUMNS DESCRIBED BY THIS DECLARATION IS 5             */
/************************************************************************/
```

Example 2: This example shows the use of NAMES and STRUCTURE. The statement

```
DCLGEN TABLE(VEMPL) -
       LIBRARY('prefix.SRCLIB.DATA(DSN8MPEM)') -
       LANGUAGE(PLI) -
       NAMES(FIELD) -
       STRUCTURE(EMPRECORD) -
       APOST
```

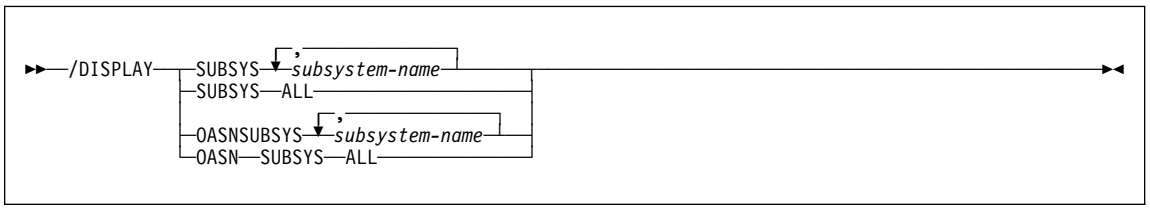
produces the following statements in *prefix*.SRCLIB.DATA(DSN8MPEM):

DCLGEN (DSN)

```
/************************************************************************/
/* DCLGEN TABLE(VEMPL) -                                                */
/*        LIBRARY('prefix.SRCLIB.DATA(DSN8MPEM)') -                     */
/*        LANGUAGE(PLI) -                                               */
/*        NAMES(FIELD) -                                                */
/*        STRUCTURE(EMPRECORD) -                                        */
/*        APOST                                                         */
/* ... IS THE DCLGEN COMMAND THAT MADE THE FOLLOWING STATEMENTS         */
/************************************************************************/
EXEC SQL DECLARE VEMPL TABLE
          ( EMPNO                         CHAR(6) NOT NULL,
            FIRSTNME                      VARCHAR(12) NOT NULL,
            MIDINIT                       CHAR(1) NOT NULL,
            LASTNAME                      VARCHAR(15) NOT NULL,
            WORKDEPT                      CHAR(3) NOT NULL
          ) ;
/************************************************************************/
/* PLI DECLARATION FOR TABLE VEMPL                                      */
/************************************************************************/
DCL 1 EMPRECORD,
     5 FIELD1   CHAR(6),
     5 FIELD2   CHAR(12) VAR,
     5 FIELD3   CHAR(1),
     5 FIELD4   CHAR(15) VAR,
     5 FIELD5   CHAR(3);
/************************************************************************/
/* THE NUMBER OF COLUMNS DESCRIBED BY THIS DECLARATION IS 5             */
/************************************************************************/
```

/DISPLAY (IMS)

Syntax



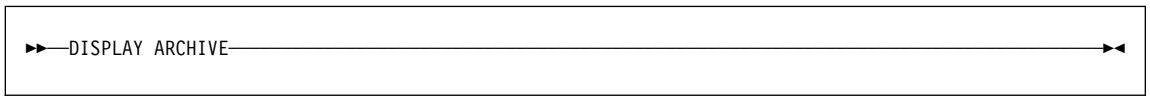
Example

Example: Display the status of connections between IMS and all subsystems.

```
/DISPLAY SUBSYSTEM ALL
```

-DISPLAY ARCHIVE (DB2)

Syntax



Example

Example: Display tape unit information about input archive logs.

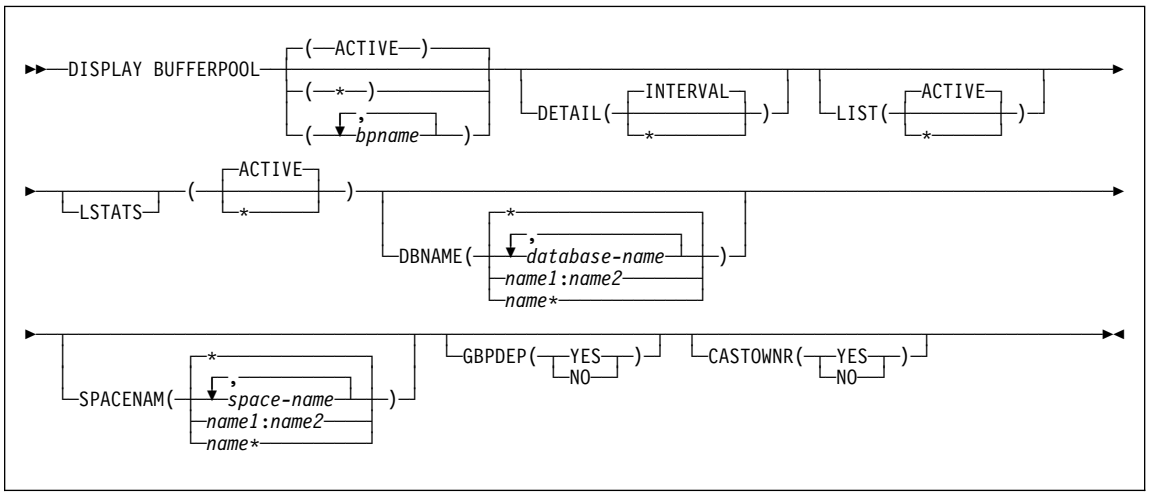
```
-DISPLAY ARCHIVE

DSNJ322I - DISPLAY ARCHIVE REPORT FOLLOWS-
               COUNT                   TIME
             (TAPE UNITS)            (MIN,SEC)
DSNZPARM           2                     0,00
CURRENT            2                     5,30
==============================
ADDR STATUS CORR-ID  VOLSER DATASET_NAME
 290  AVAIL  *****   TAPE1  DSNCAT.ARCHLOG1.A0000033
 294  PREM   *****   TAPE3  DSNCAT.ARCHLOG1.A0000035
 293  BUSY  RECOVER2 TAPE2  DSNCAT.ARCHLOG1.A0000034
END OF DISPLAY ARCHIVE REPORT.
DSN9022I - DSNJC001 '-DISPLAY ARCHIVE' NORMAL COMPLETION
```

This example report shows:

- The subsystem parameter values for MAX RTU (COUNT) and DEALLC PERIOD TIME as recorded in the DSNZPxxx load module
- Current specifications for the COUNT and TIME parameters
- Availability status of allocated dedicated tape units
- Volume and data set names associated with all busy tape units

-DISPLAY BUFFERPOOL (DB2)

Syntax



Examples

Example 1: A summary report is the default report if the DETAIL option is not specified. The following is an example of a summary report which could be produced by the command:

```
-DIS BUFFERPOOL(BP0) LIST(*) DBNAME(DSN8*)

   DSNB401I - BUFFERPOOL NAME BP0, BUFFERPOOL ID 0, USE COUNT 20
   DSNB402I - VIRTUAL BUFFERPOOL SIZE = 500 BUFFERS 736
                ALLOCATED       =     500   TO BE DELETED   =        0
                IN-USE/UPDATED  =      0
   DSNB403I - HIPERPOOL SIZE = 10000 BUFFERS, CASTOUT = YES
                ALLOCATED       =   0   TO BE DELETED   =        0
                BACKED BY ES    =    0
   DSNB404I - THRESHOLDS - 739
                VP SEQUENTIAL         = 80   HP SEQUENTIAL         = 75
                DEFERRED WRITE        = 85   VERTICAL DEFERRED WRT = 80,0
                PARALLEL SEQUENTIAL   = 50 ASSISTING PARALLEL SEQT  = 0
   DSNB406I - VIRTUAL BUFFERPOOL TYPE = -737
                CURRENT              = PRIMARY
                PENDING               = PRIMARY
              PAGE STEALING METHOD = LRU
   DSNB460I -   740
```

Example 2: A detail report can be generated that includes all summary report information and additional buffer pool related statistics. The following is an example of a detail report that could be produced by the command:

```
-DISPLAY BUFFERPOOL(BP0) DETAIL
```

```
DSNB401I - BUFFERPOOL NAME BP0, BUFFERPOOL ID 0, USE COUNT 10
DSNB402I - VIRTUAL BUFFERPOOL SIZE = 1000 BUFFERS
             ALLOCATED       =     1000   TO BE DELETED   =        0
             IN-USE/UPDATED  =      200
DSNB403I - HIPERPOOL SIZE = 600000 BUFFERS, CASTOUT = YES
             ALLOCATED       =   600000   TO BE DELETED   =        0
             BACKED BY ES    =   483651
DSNB404I - THRESHOLDS -
             VP SEQUENTIAL         = 80   HP SEQUENTIAL         = 80
             DEFERRED WRITE        = 50   VERTICAL DEFERRED WRT = 10
             PARALLEL SEQUENTIAL   = 50
DSNB405I - HIPERSPACE NAMES - @001SSOP @002SSOP

DSNB409I - INCREMENTAL STATISTICS SINCE 10:32:48 OCT 23, 1993

DSNB411I - RANDOM GETPAGE    =      230 SYNC READ I/O (R) =      180
           SEQ.   GETPAGE    =      610 SYNC READ I/O (S) =       20
           DMTH HIT          =        0

DSNB412I - SEQUENTIAL PREFETCH -
             REQUESTS        =        0   PREFETCH  I/O   =        0
             PAGES READ      =        0

DSNB413I - LIST PREFETCH -
             REQUESTS        =        0   PREFETCH  I/O   =        0
             PAGES READ      =        0

DSNB414I - DYNAMIC PREFETCH -
             REQUESTS        =        0   PREFETCH  I/O   =        0
             PAGES READ      =        0

DSNB415I - PREFETCH DISABLED -
             NO BUFFER       =        0   NO READ ENGINE  =           0
DSNB420I - SYS PAGE UPDATES  =        0 SYS PAGES WRITTEN =           0
           ASYNC WRITE I/O   =        0 SYNC WRITE I/O    =           0
DSNB421I - DWT HIT           =        0 VERTICAL DWT HIT  =           0
           NO WRITE ENGINE   =        0

DSNB430I - HIPERPOOL ACTIVITY (NOT USING ASYNCHRONOUS
           DATA MOVER FACILITY) -
             SYNC HP READS   =      100   SYNC HP WRITES  =           0
             ASYNC HP READS  =        0   ASYNC HP WRITES =           0
             READ FAILURES   =        0   WRITE FAILURES  =           0

DSNB431I - HIPERPOOL ACTIVITY (USING ASYNCHRONOUS
           DATA MOVER FACILITY) -
             HP READS        =      244   HP WRITES       =           3
             READ FAILURES   =        0   WRITE FAILURES  =           0

DSNB440I - PARALLEL ACTIVITY -
             PARALL REQUEST  =        0   DEGRADED PARALL =       0

DSN9022I - DSNB1CMD '-DISPLAY BUFFERPOOL' NORMAL COMPLETION
```

Example 3: With the summary or detail report, you can list open table spaces and index spaces associated with the buffer pool. You can also request a display of

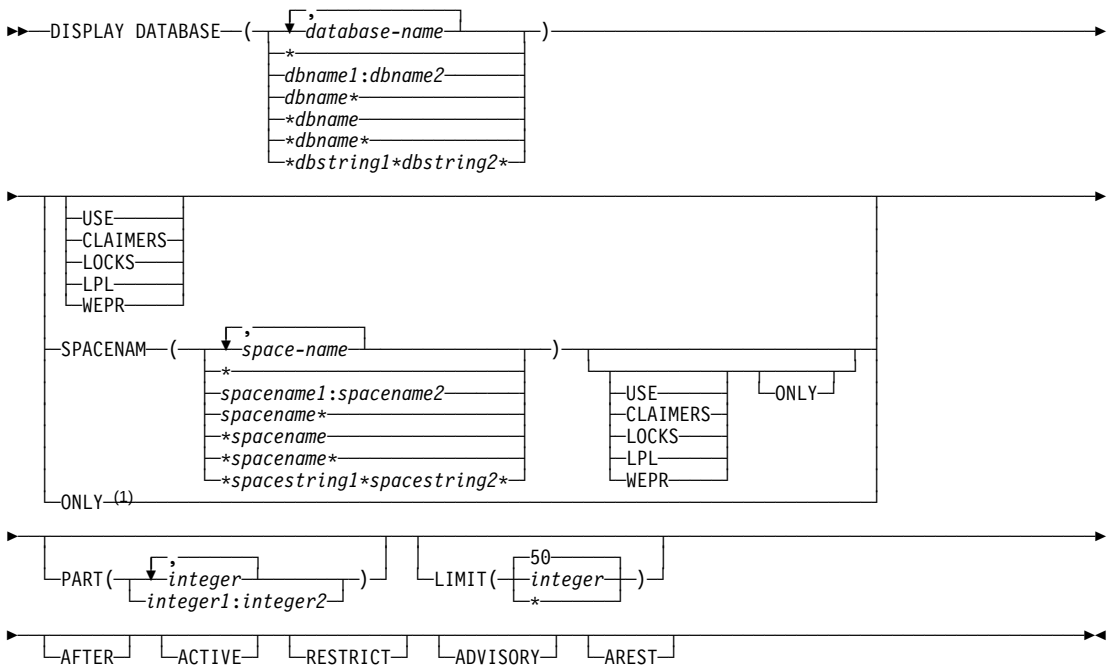
statistics for each listed table space and index space. An example of a report generating this information could be produced by the command:

```
-DISPLAY BUFFERPOOL(BP0) LIST LSTATS
```

```
   DSNB401I - BUFFERPOOL NAME BP0, BUFFERPOOL ID 0, USE COUNT 3
   DSNB402I - VIRTUAL BUFFERPOOL SIZE = 1000 BUFFERS
                ALLOCATED       =     1000   TO BE DELETED   =        0
                IN-USE/UPDATED  =      200
   DSNB403I - HIPERPOOL SIZE = 100000 BUFFERS, CASTOUT = YES
                ALLOCATED       =   100000   TO BE DELETED   =        0
                BACKED BY ES    =    89152
   DSNB404I - THRESHOLDS -
                VP SEQUENTIAL         = 80   HP SEQUENTIAL         = 80
                DEFERRED WRITE        = 50   VERTICAL DEFERRED WRT = 10
                PARALLEL SEQUENTIAL   = 50
   DSNB405I - HIPERSPACE NAMES - @001SSOP
   DSNB455I -   SYNCHRONOUS I/O DELAYS -
                  AVERAGE DELAY  =     22   MAXIMUM DELAY      =    35
                  TOTAL PAGES    =     23
   DSN9022I - DSNB1CMD '-DISPLAY BUFFERPOOL' NORMAL COMPLETION
```

-DISPLAY DATABASE (DB2)

Syntax



Note:
1 If you specify the ONLY option without the SPACENAM() keyword, only the LIMIT, AFTER, and RESTRICT keywords apply.

Examples

Example 1: Display information about table space TBS33 in database CB3. USE causes *connection-name*(CONNID), *correlation-id*(CORRID), and *authorization ID* (USERID) information to be displayed.

```
-DISPLAY DATABASE(CB3) SPACENAM(TBS33) USE
```

Example 2: Display information about table space TBS33 in database CB3 when the table space is defined with LOCKPART YES. LOCKS displays lock information for table spaces and tables specified; LUWIDs and locations of any remote threads; and *connection-name*, *correlation-id*, and *authorization ID* information.

```
-DISPLAY DATABASE(CB3) SPACENAM(TBS33) LOCKS
```

Example 3: Display information about table space TBS33 in database CB3. CLAIMERS displays claim types and durations; LUWIDs and locations of any remote threads; and *connection-name*, *correlation-id*, and *authorization ID* information.

```
-DISPLAY DATABASE(CB3) SPACENAM(TBS33) CLAIMERS
```

Example 4: In a data sharing environment, display information about locks held when the table space is defined with LOCKPART YES.

```
-DISPLAY DATABASE(DSN8D51A) SPACENAM(TSPART) LOCKS
```

Example 5: Display information about page sets in database DSNDB01 that have entries in the logical page list. Limit the number of messages displayed to the space available.

```
-DB1G DISPLAY DATABASE(DSNDB01) SPACENAM(*) LIMIT(*) LPL
```

Example 6: Suppose that table space TSPART, which is in database DSN8D61A, is defined with the keyword LOCKPART NO, which means that DB2 does not do selective partition locking on TSPART. When you specify this command:

```
-DB1G DISPLAY DATABASE(DSN8D61A) SPACE(TSPART) PART(1,4) LOCKS
```

two applications are accessing TSPART, and the partitions have different statuses. In the output, DB2 displays the locks as table space locks, as shown here:

```
NAME     TYPE PART STATUS             CONNID   CORRID       LOCKINFO
-------- ---- ---- ------------------ -------- ------------ ---------
TSPART   TS                          LSS001   DSN2SQL      H-IS,S,C
TSPART   TS                          LSS002   DSN2SQL      H-IS,S,C
TSPART   TS    01  RO
TSPART   TS    04  RW
```

Example 7: Suppose that you have executed the ALTER TABLESPACE statement on table space TSPART so that TSPART is now defined with LOCKPART YES. LOCKPART YES causes DB2 to do selective partition locking on TSPART. When you specify this command:

```
-DB1G DISPLAY DATABASE(DSN8D61A) SPACE(TSPART) PART(1:4) LOCKS
```

two applications are accessing TSPART. The application identified by connection ID LSS001 has locked partitions 1 and 2. The application identified by connection ID LSS002 has locked partitions 1 and 3. In the output, DB2 displays the locks as partition locks, as shown here:

```
NAME     TYPE PART STATUS              CONNID   CORRID       LOCKINFO
-------- ---- ---- ------------------ -------- ------------ ---------
TSPART   TS    01  RO                 LSS001   DSN2SQL      H-IS,P,C
TSPART   TS    01  RO                 LSS002   DSN2SQL      H-IS,P,C
TSPART   TS    02  RW                 LSS001   DSN2SQL      H-IS,P,C
TSPART   TS    03  RW                 LSS002   DSN2SQL      H-IS,P,C
TSPART   TS    04  RW
```

Example 8: Display information about all table spaces and index spaces in the range of databases from DBKD0101 to DBKD0106 that are in a restrictive status. Limit the number of messages that are displayed to the available space.

```
-DISPLAY DATABASE(DBKD0101,DBKD0103) SPACENAM(*) RESTRICT LIMIT(*)
```

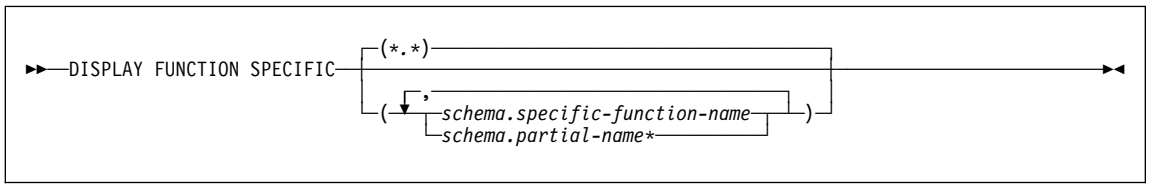
Example 9: Display information about all table spaces that are in the auxiliary warning advisory status (AUXW), and all index spaces that are in informational COPY pending status (ICOPY) in database DBIQUQ01. Limit the number of messages that are displayed to the available space.

```
-DISPLAY DATABASE(DBIQUQ01) SPACENAM(*) LIMIT(*) ADVISORY
```

DISPLAY FUNCTION SPECIFIC (DB2)

-DISPLAY FUNCTION SPECIFIC (DB2)

Syntax



Examples

Example 1: Display information about functions in the PAYROLL schema and the HRPROD schema.

```
-DISPLAY FUNCTION SPECIFIC(PAYROLL.*, HRPROD.*)
```

This command produces output similar to the following output:

```
DSNX975I csect - DISPLAY FUNCTION SPECIFIC REPORT FOLLOWS-

------ SCHEMA=PAYROLL
FUNCTION           STATUS   ACTIVE  QUEUED  MAXQUE  TIMEOUT   WLM_ENV
PAYRFNC1           STARTED       0       0       1       0    PAYROLL
PAYRFNC2           STOPQUE       0       5       5       3    PAYROLL
PAYRFNC3           STARTED       2       0       6       0    PAYROLL
USERFNC4           STOPREJ       0       0       1       0    SANDBOX

------ SCHEMA=HRPROD
FUNCTION           STATUS   ACTIVE  QUEUED  MAXQUE  TIMEOUT   WLM_ENV
HRFNC1             STARTED       0       0       1       0    HRFUNCS
HRFNC2             STOPREJ       0       0       1       0    HRFUNCS
DSNX9DIS DISPLAY FUNCTION SPECIFIC REPORT COMPLETE
DSN9022I - DSNX9COM '-DISPLAY FUNC' NORMAL COMPLETION
```

Example 2: Display information about specific functions in the PAYROLL schema.

```
-DISPLAY FUNCTION SPECIFIC(PAYROLL.USERFNC2,PAYROLL.USERFNC4)
```

This command produces output similar to the following output:

```
DSNX975I csect - DISPLAY FUNCTION SPECIFIC REPORT FOLLOWS-

------ SCHEMA=PAYROLL
FUNCTION           STATUS   ACTIVE  QUEUED  MAXQUE  TIMEOUT   WLM_ENV
USERFNC2           STOPQUE       0       5       5       3    SANDBOX
USERFNC4           STOPREJ       0       0       1       0    SANDBOX
DSNX9DIS DISPLAY FUNCTION SPECIFIC REPORT COMPLETE
DSN9022I - DSNX9COM '-DISPLAY FUNC' NORMAL COMPLETION
```

Example 3: Display information about all functions in the PAYROLL schema that DB2 applications have accessed. This example assumes that the STOP FUNCTION SPECIFIC(PAYROLL.*) ACTION(QUEUE) command is in effect at the time you issue the DISPLAY FUNCTION SPECIFIC command:

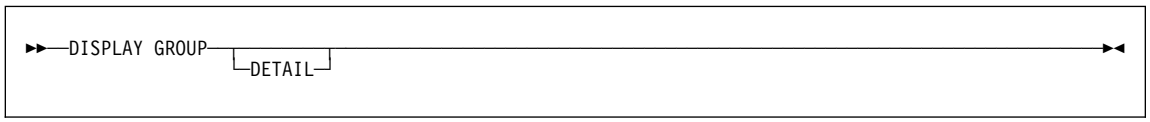
```
-DISPLAY FUNCTION SPECIFIC(PAYROLL.*)
```

This command produces output similar to the following output:

```
DSNX975I csect - DISPLAY FUNCTION SPECIFIC REPORT FOLLOWS-

------ SCHEMA=PAYROLL
FUNCTION           STATUS   ACTIVE  QUEUED  MAXQUE  TIMEOUT   WLM_ENV
USERFNC2           STOPQUE       0       5       5       3    SANDBOX
USERFNC4           STOPQUE       0       0       1       0    SANDBOX
FUNCTIONS USERFNC2            - USERFNC29999999999 STOP QUEUE
FUNCTIONS USERFNC4            - USERFNC49999999999 STOP QUEUE
DSNX9DIS DISPLAY FUNCTION SPECIFIC REPORT COMPLETE
DSN9022I - DSNX9COM '-DISPLAY FUNC' NORMAL COMPLETION
```

-DISPLAY GROUP (DB2)

Syntax



Examples

Example 1: The following sample output for a data sharing group can be generated by the command:

```
-DB1A DIS GROUP
```

```
DSN7100I -DB1A DSN7GCMD
*** BEGIN DISPLAY OF GROUP(DSNDB10 ) GROUP LEVEL(610)
                                    GROUP ATTACH NAME(DB10)
------------------------------------------------------------------------
DB2                                    DB2 SYSTEM    IRLM
MEMBER   ID  SUBSYS CMDPREF   STATUS   LVL NAME      SUBSYS IRLMPROC
-------- --- ----   --------  -------- --- --------  ----   --------
DB1A       1 DB1A   -DB1A     ACTIVE   610 MVSA      DJ1A   DB1AIRLM
DB1B       2 DB1B   -DB1B     ACTIVE   610 MVSB      DJ1B   DB1BIRLM
DB1C       3 DB1C   -DB1C     ACTIVE   610 MVSC      DJ1C   DB1CIRLM
DB1D       4 DB1D   -DB1D     FAILED   610 MVSD      DJ1D   DB1DIRLM
DB1E       5 DB1E   -DB1E     QUIESCED 610 MVSE      DJ1E   DB1EIRLM
DB1F       6 DB1F   -DB1F     ACTIVE   610 MVSF      DJ1F   DB1FIRLM
DB1G       7 DB1G   -DB1G     ACTIVE   610 MVSG      DJ1G   DB1GIRLM
------------------------------------------------------------------------
SCA   STRUCTURE SIZE:     1024 KB, STATUS= AC,   SCA IN USE:    11 %
LOCK1 STRUCTURE SIZE:     1536 KB
NUMBER  LOCK ENTRIES:      262144
NUMBER  LIST ENTRIES:        7353, LIST ENTRIES  IN USE:           0
*** END DISPLAY OF GROUP(DSNDB10 )
DSN9022I -DB1A DSN7GCMD 'DISPLAY GROUP ' NORMAL COMPLETION
```

Example 2: In a non-data-sharing environment, the following sample output is generated by the command:

```
-DB1A DISPLAY GROUP
```

```
DSN7100I -DB1A DSN7GCMD
*** BEGIN DISPLAY OF GROUP(.......) GROUP LEVEL(...)
                                    GROUP ATTACH NAME(....)
-------------------------------------------------------------------
DB2                                    DB2 SYSTEM    IRLM
MEMBER   ID  SUBSYS CMDPREF   STATUS   LVL NAME      SUBSYS IRLMPROC
-------- --- ----   --------  -------- --- --------  ----   --------
........   0 DB1A   -DB1A     ACTIVE   610 MVSA      DJ1A   DB1AIRLM
-------------------------------------------------------------------
*** END DISPLAY OF GROUP(DSNDB10)
DSN9022I -DB1A DSN7GCMD 'DISPLAY GROUP ' NORMAL COMPLETION
```

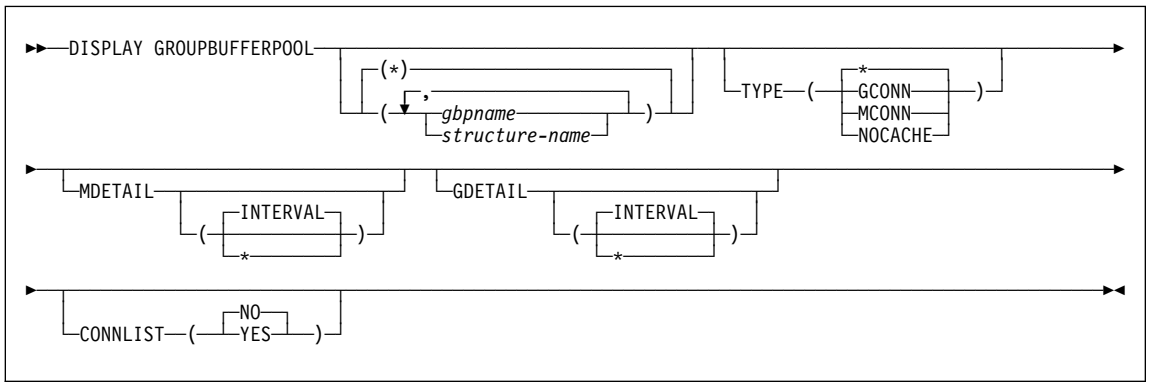
Example 3: Using the DETAIL option, you can obtain more information about the data sharing group as shown in the following example using the command:

```
-DB1A DIS GROUP DETAIL
```

```
DSN7100I -DB1A DSN7GCMD
*** BEGIN DISPLAY OF GROUP(DSNCAT ) GROUPLEVEL(610)
-------------------------------------------------------------------
DB2                                    SYSTEM        IRLM
MEMBER   ID  SUBSYS CMDPREF   STATUS   NAME     LVL  SUBSYS IRLMPROC
-------- --- ----   --------  -------- -------- ---  ----   --------
DB1A       1 DB1A   -DB1A     ACTIVE   MVSA     610  AR21   ARLM21
DB1B       2 DB1B   -DB1B     ACTIVE   MVSB     610  BR21   BRLM21
DB1C       3 DB1C   -DB1C     ACTIVE   MVSC     510  CRLM   CRLM21
DB2D       4 DB2D   -DB2D     FAILED   MVSD     610  DR21   DRLM21
DB2E       5 DB2E   -DB2E     QUIESCED MVSE     610  ER21   ERLM21
DB2F       6 DB2F   -DB2F     ACTIVE   MVSF     610  FR21   FRLM21
DB2G       7 DB2G   -DB2G     ACTIVE   MVSG     610  GR21   GRLM21
-------------------------------------------------------------------
DB2      PARALLEL    PARALLEL
MEMBER   COORDINATOR ASSISTANT
-------- ----------- ---------
DB2A             YES        NO
DB2B             YES       YES
DB2B             YES       YES
DB1C            ****      ****
DB2D            ****      ****
DB2E            ****      ****
DB2F               NO       YES
DB2G               NO        NO
-------------------------------------------------------------------
SCA   STRUCTURE SIZE:     1024 KB, STATUS= AC,   SCA IN USE:    11 %
LOCK1 STRUCTURE SIZE:     1536 KB,             LOCK1 IN USE: <   1 %
NUMBER  LOCK ENTRIES:      262144, LOCK ENTRIES  IN USE:          33
NUMBER  LIST ENTRIES:        7353, LIST ENTRIES  IN USE:           0
*** END DISPLAY OF GROUP(DSNCAT )
DSN9022I -DB1A DSN7GCMD 'DISPLAY GROUP ' NORMAL COMPLETION
```

-DISPLAY GROUPBUFFERPOOL (DB2)

Syntax



Examples

Example 1: This is an example of a summary report that can be produced by the following command:

```
-DISPLAY GROUPBUFFERPOOL(GBP29)
```

Message DSNB799I is displayed if the group buffer pool is duplexed and the secondary group buffer pool is currently allocated. If a secondary group buffer pool is not allocated, message DSNB799I is not included in the output.

```
DSNB750I - DISPLAY FOR GROUP BUFFER POOL GBP29 FOLLOWS
DSNB755I - DB2 GROUP BUFFER POOL STATUS
             CONNECTED                                = YES
             CURRENT DIRECTORY TO DATA RATIO          = 5
             PENDING DIRECTORY TO DATA RATIO          = 5
             CURRENT GBPCACHE ATTRIBUTE               = YES
             PENDING GBPCACHE ATTRIBUTE               = YES
DSNB756I -   CLASS CASTOUT THRESHOLD                  = 10%
             GROUP BUFFER POOL CASTOUT THRESHOLD       = 50%
             GROUP BUFFER POOL CHECKPOINT INTERVAL     = 8 MINUTES
             RECOVERY STATUS                          = NORMAL
             AUTOMATIC RECOVERY                       = Y
DSNB757I - MVS CFRM POLICY STATUS FOR DSNCAT_GBP29     = NORMAL
             MAX SIZE INDICATED IN POLICY             = 2048 KB
             DUPLEX INDICATOR IN POLICY               = ENABLED
             CURRENT DUPLEXING MODE                   = DUPLEX
             ALLOCATED                                = YES
DSNB758I -     ALLOCATED SIZE                         = 2048 KB
               VOLATILITY STATUS                      = VOLATILE
               REBUILD STATUS                         = DUPLEXED
               CFNAME                                 = CACHE01
               CFLEVEL                                = 5
DSNB759I -     NUMBER OF DIRECTORY ENTRIES            = 1950
               NUMBER OF DATA PAGES                   = 389
               NUMBER OF CONNECTIONS                  = 2
DSNB798I - LAST GROUP BUFFER POOL CHECKPOINT
                                      17:08:41 OCT 16, 1997
             GBP CHECKPOINT RECOVERY LRSN             = AF6BBAEF3307
             STRUCTURE OWNER                          = V61B
DSNB799I - SECONDARY GBP ATTRIBUTES
             ALLOCATED SIZE                           = 2048 KB
             VOLATILITY STATUS                        = VOLATILE
             CFNAME                                   = LF01
             CFLEVEL                                  = 5
             NUMBER OF DIRECTORY ENTRIES              = 1950
             NUMBER OF DATA PAGES                     = 389
DSNB790I - DISPLAY FOR GROUP BUFFER POOL GBP29 IS COMPLETE
DSN9022I - DSNB1CMD '-DISPLAY GBPOOL' NORMAL COMPLETION
```

Example 2: Assume you want a summary report about group buffer pool twenty-nine, including all connections to that group buffer pool. Enter the following command:

```
-DISPLAY GROUPBUFFERPOOL(GBP29) CONNLIST(YES)
```

-DISPLAY GROUPBUFFERPOOL (DB2)

Here is what the display might look like:

```
DSNB750I - DISPLAY FOR GROUP BUFFER POOL GBP29 FOLLOWS
DSNB755I - DB2 GROUP BUFFER POOL STATUS
             CONNECTED                                = YES
             CURRENT DIRECTORY TO DATA RATIO          = 5
             PENDING DIRECTORY TO DATA RATIO          = 5
             CURRENT GBPCACHE ATTRIBUTE               = YES
             PENDING GBPCACHE ATTRIBUTE               = YES
DSNB756I -   CLASS CASTOUT THRESHOLD                  = 10%
             GROUP BUFFER POOL CASTOUT THRESHOLD       = 50%
             GROUP BUFFER POOL CHECKPOINT INTERVAL     = 8 MINUTES
             RECOVERY STATUS                          = NORMAL
             AUTOMATIC RECOVERY                       = Y
DSNB757I - MVS CFRM POLICY STATUS FOR DSNCAT_GBP29    = NORMAL
             MAX SIZE INDICATED IN POLICY             = 2048 KB
             DUPLEX INDICATOR IN POLICY               = ENABLED
             CURRENT DUPLEXING MODE                   = SIMPLEX
             ALLOCATED                                = YES
DSNB758I -     ALLOCATED SIZE                         = 2048 KB
               VOLATILITY STATUS                      = VOLATILE
               REBUILD STATUS                         = DUPLEXED
               CFNAME                                 = CACHE01
               CFLEVEL                                = 5
DSNB759I -     NUMBER OF DIRECTORY ENTRIES            = 1950
               NUMBER OF DATA PAGES                   = 389
               NUMBER OF CONNECTIONS                  = 2
DSNB798I - LAST GROUP BUFFER POOL CHECKPOINT
                                      17:08:41 OCT 16, 1997
             GBP CHECKPOINT RECOVERY LRSN             = AF6BBAEF3307
             STRUCTURE OWNER                          = V61B
DSNB799I - SECONDARY GBP ATTRIBUTES
             ALLOCATED SIZE                           = 2048 KB
             VOLATILITY STATUS                        = VOLATILE
             CFNAME                                   = LF01
             CFLEVEL                                  = 5
             NUMBER OF DIRECTORY ENTRIES              = 1950
             NUMBER OF DATA PAGES                     = 389
DSNB766I - THE CONNLIST REPORT FOLLOWS
DSNB767I - CONNECTION NAME = DB2_V61B        , CONNECTION STATUS = D
               CONNECTOR'S RELEASE                  =     6100
DSNB767I - CONNECTION NAME = DB2_V61A        , CONNECTION STATUS = D
               CONNECTOR'S RELEASE                  =     6100
DSNB769I - THE CONNLIST REPORT IS COMPLETE
DSNB790I - DISPLAY FOR GROUP BUFFER POOL GBP29 IS COMPLETE
DSN9022I - DSNB1CMD '-DISPLAY GBPOOL' NORMAL COMPLETION
```

Example 3: This example shows a group detail report that is produced by the command:

```
-DISPLAY GROUPBUFFERPOOL(GBP29) GDETAIL(*)
```

Message DSNB762I is displayed in the output only if the secondary group buffer pool is allocated.

```
DSNB750I - DISPLAY FOR GROUP BUFFER POOL GBP29 FOLLOWS
DSNB755I - DB2 GROUP BUFFER POOL STATUS
             CONNECTED                                = YES
             CURRENT DIRECTORY TO DATA RATIO          = 5
             PENDING DIRECTORY TO DATA RATIO          = 5
             CURRENT GBPCACHE ATTRIBUTE               = YES
             PENDING GBPCACHE ATTRIBUTE               = YES
DSNB756I -   CLASS CASTOUT THRESHOLD                  = 10%
             GROUP BUFFER POOL CASTOUT THRESHOLD       = 50%
             GROUP BUFFER POOL CHECKPOINT INTERVAL     = 8 MINUTES
             RECOVERY STATUS                          = NORMAL
             AUTOMATIC RECOVERY                       = Y
DSNB757I - MVS CFRM POLICY STATUS FOR DSNCAT_GBP29     = NORMAL
             MAX SIZE INDICATED IN POLICY             = 2048 KB
             DUPLEX INDICATOR IN POLICY               = ENABLED
             CURRENT DUPLEXING MODE                   = DUPLEX
             ALLOCATED                                = YES
DSNB758I -     ALLOCATED SIZE                         = 2048 KB
               VOLATILITY STATUS                      = VOLATILE
               REBUILD STATUS                         = DUPLEXED
               CFNAME                                 = CACHE01
               CFLEVEL                                = 5
DSNB759I -     NUMBER OF DIRECTORY ENTRIES            = 1950
               NUMBER OF DATA PAGES                   = 389
               NUMBER OF CONNECTIONS                  = 2
DSNB798I - LAST GROUP BUFFER POOL CHECKPOINT
                                       17:08:41 OCT 16, 1997
             GBP CHECKPOINT RECOVERY LRSN             = AF6BBAEF3307
             STRUCTURE OWNER                          = V61B
DSNB799I - SECONDARY GBP ATTRIBUTES
             ALLOCATED SIZE                           = 2048 KB
             VOLATILITY STATUS                        = VOLATILE
             CFNAME                                   = LF01
             CFLEVEL                                  = 5
             NUMBER OF DIRECTORY ENTRIES              = 1950
             NUMBER OF DATA PAGES                     = 389
DSNB783I - CUMULATIVE GROUP DETAIL STATISTICS SINCE 17:08:35 OCT 16,
1997
DSNB784I - GROUP DETAIL STATISTICS
             READS
               DATA RETURNED                          = 4
DSNB785I -     DATA NOT RETURNED
                 DIRECTORY ENTRY EXISTED              = 0
                 DIRECTORY ENTRY CREATED              = 45
                 DIRECTORY ENTRY NOT CREATED          = 0, 0
DSNB786I -   WRITES
               CHANGED PAGES                          = 5
               CLEAN PAGES                            = 0
               FAILED DUE TO LACK OF STORAGE          = 0
             CHANGED PAGES SNAPSHOT VALUE             = 5
DSNB787I -   RECLAIMS
               FOR DIRECTORY ENTRIES                  = 0
               FOR DATA ENTRIES                       = 0
             CASTOUTS                                 = 0
```

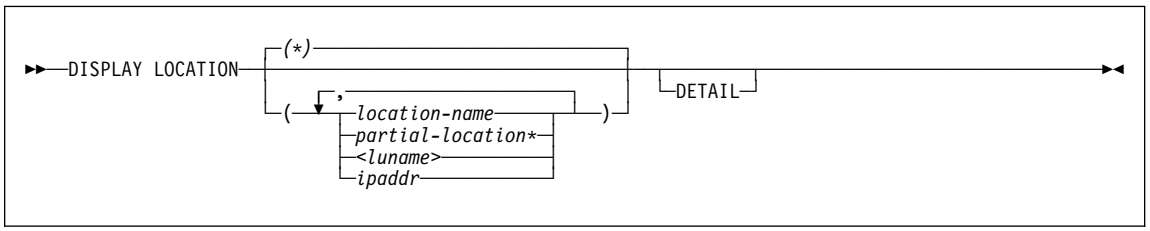
-DISPLAY GROUPBUFFERPOOL (DB2)

```
DSNB788I -   CROSS INVALIDATIONS
               DUE TO DIRECTORY RECLAIMS               = 0
               DUE TO WRITES                           = 0
               EXPLICIT                                = 0
DSNB762I - DUPLEXING STATISTICS FOR GBP29-SEC
             WRITES
               CHANGED PAGES                           = 5
               FAILED DUE TO LACK OF STORAGE           = 0
             CHANGED PAGES SNAPSHOT VALUE              = 5
DSNB790I - DISPLAY FOR GROUP BUFFER POOL GBP29 IS COMPLETE
DSN9022I - DSNB1CMD '-DISPLAY GBPOOL' NORMAL COMPLETION
```

Messages DSNB764I and DSNB793I are displayed in the output only if the secondary group buffer pool is allocated.

-DISPLAY LOCATION (DB2)

Syntax



Examples

Example 1: Display information about threads and conversations with specific remote locations, using the following command:

```
-DISPLAY LOCATION(SAN_JOSE,SAN_FRANCISCO)

DSNL200I - DISPLAY LOCATION REPORT FOLLOWS-
LOCATION       PRDID        LINKNAME           REQUESTERS SERVERS CONVS
SAN_JOSE       DSN05010     LUND1                   1         0      1
SAN_FRANCISCO  DSN05010     LUND3                   1         0      1
DISPLAY LOCATION REPORT COMPLETE
```

Example 2: Display information about threads and conversations with all remote locations. Additionally, display detail conversation information about DB2 system threads that communicate with other locations. This is an example of the output generated by the following command:

-DISPLAY LOCATION (DB2)

```
-DISPLAY LOCATION DETAIL

DSNL200I - DISPLAY LOCATION REPORT FOLLOWS-
LOCATION          PRDID     LINKNAME           REQUESTERS SERVERS CONVS
SAN_JOSE          DSN05010  LUND1                  1         0      3
 -SYSTASK   SESSID           A ST TIME
 -SYSCON-O  00D359691359EE80   S  9128009214880
 -SYSCON-I  00D359691359EE81 W R  9128009214881
MENLO_PARK        DSN05010  LUND2                  1         0      4
 -SYSTASK   SESSID           A ST TIME
 -SYSCON-O  00D359691359EE82   S  9128009214882
 -SYSCON-I  00D359691359EE83 W R  9128009214883
 -RESYNC    00D359691359EE84 V R  9128009214884
SAN_FRANCISCO     DSN05010  LUND3                  1         0      6
 -SYSTASK   SESSID           A ST TIME
 -SYSCON-O  0000000000000000   C  9128009214885
 -SYSCON-I  00D359691359EE86 W R  9128009214886
 -RESYNC    00D359691359EE87 W R  9128009214887
 -RESYNC    00D359691359EE88 W R  9128009214888
 -RESYNC    00D359691359EE89 W R  9128009214889
DISPLAY LOCATION REPORT COMPLETE
```

Example 3: Display information for a DB2 that is connected to the following DRDA partners:

- A non-MVS server named DRDALOC via TCP/IP.
- Several TCP/IP clients from the same TCP/IP host as the DRDALOC server.
- A DB2 for MVS server named DB2SERV via SNA.

```
DISPLAY LOCATION(*)

DSNL200I - DISPLAY LOCATION REPORT FOLLOWS -
 LOCATION      PRDID     LINKNAME        REQUESTERS  SERVERS   CONVS
 DRDALOC       SQL03030  124.63.51.17    3           0         3
 124.63.51.17  SQL03030  124.63.51.17    0           15        15
 DB2SERV       DSN05010  LULA            1           0         1
 DISPLAY LOCATION REPORT COMPLETE
```

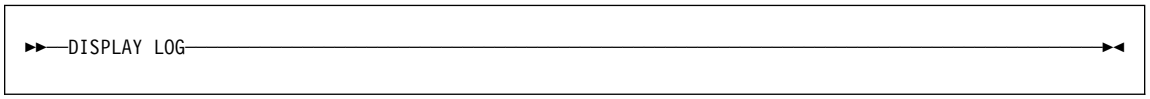
Example 4: The following example assumes DB2 is connected to the following DRDA partners:

- DB2A is connected to this DB2, using TCP/IP for DRDA connections and SNA for DB2 private protocol connections.
- DB2SERV is connected to this DB2 using only SNA.

```
DISPLAY LOCATION(*)

DSNL200I - DISPLAY LOCATION REPORT FOLLOWS -
 LOCATION      PRDID     LINKNAME        REQUESTERS  SERVERS   CONVS
 DB2A          DSN05010  LUDB2A          3           4         9
 DB2A          DSN05010  124.38.54.16    2           1         3
 DB2SERV       DSN04010  LULA            1           1         3
 DISPLAY LOCATION REPORT COMPLETE
```

DISPLAY LOG (DB2)

Syntax



Examples

Example 1: Display log information and status of the offload task.

```
DISPLAY LOG
```

This command produces output similar to the following output:

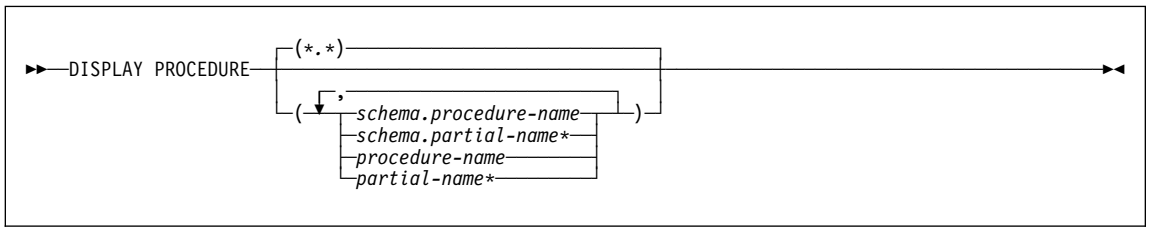
```
DSNJ370I - DSNJCOOA LOG DISPLAY
CURRENT COPY1 LOG = DSNC610.LOGCOPY1.DS03 IS 22% FULL
CURRENT COPY2 LOG = DSNC610.LOGCOPY2.DS03 IS 22% FULL
H/W RBA = 0000039A9F24, LOGLOAD = 150000
FULL LOGS TO OFFLOAD = 2 OF 6, OFFLOAD TASK IS (BUSY,ALLC)
DSNJ371I - DB2 RESTARTED 14:06:23 MAY 22, 1998
RESTART RBA 0000039A8000
DSN9002I - DSNJC001 'DIS LOG' NORMAL COMPLETION
```

This example shows the following information:

- The active log data sets are 22% full. If you are running dual logs and the percentages are different, the log data sets are of different sizes. DB2 switches both active logs when one reaches the end of the file. This can result in unused active log space if one log data set is larger than the other.
- The current LOGLOAD setting is 150000 log records between system checkpoints. You can modify this value using the SET LOG command.
- Two of the six active log data sets require archiving. The status of the offload task includes the indicator that it is busy, allocating an archive log data set. This might be an indication of an outstanding tape mount on the system console. If the status remains busy and no longer seems to be functioning, you can terminate the task, and then restart it using the ARCHIVE LOG CANCEL OFFLOAD command.
- DB2 was started at 14:06:23 on MAY 22, 1998, and began logging at RBA 0000039A8000.

-DISPLAY PROCEDURE (DB2)

Syntax



Examples

Example 1: Display information about all stored procedures that have been accessed by DB2 applications.

```
-DISPLAY PROCEDURE
```

This command produces output similar to the following output:

```
DSNX940I DSNX9DIS - DISPLAY PROCEDURE REPORT FOLLOWS-
PROCEDURE          STATUS    ACTIVE    QUEUED  MAXQUE  TIMEOUT
USERPRC1           STARTED        0         0       1        0
USERPRC2           STOPQUE        0         5       5        3
USERPRC3           STARTED        2         0       6        0
USERPRC4           STOPREJ        0         0       1        0
DSNX9DIS DISPLAY PROCEDURE REPORT COMPLETE
DSN9022I - DSNX9COM '-DISPLAY PROC' NORMAL COMPLETION
```

Example 2: Display information about specific stored procedures in the SYSPROC schema.

```
-DISPLAY PROCEDURE(SYSPROC.USERPRC2,USERPRC4)
```

This command produces output similar to the following output:

```
DSNX940I DSNX9DIS - DISPLAY PROCEDURE REPORT FOLLOWS-

------ SCHEMA=SYSPROC
PROCEDURE          STATUS    ACTIVE    QUEUED  MAXQUE  TIMEOUT  WLM_ENV
USERPRC2           STOPQUE        0         5       5        3  SANDBOX
USERPRC4           STOPREJ        0         0       1        0  SANDBOX
DSNX9DIS DISPLAY PROCEDURE REPORT COMPLETE
DSN9022I - DSNX9COM '-DISPLAY PROC' NORMAL COMPLETION
```

Example 3: Display information about stored procedures in the PAYROLL and HRPROD schemas.

```
-DISPLAY PROCEDURE(PAYROLL.*,HRPROD.*)
```

This command produces output similar to the following output:

```
DSNX940I DSNX9DIS - DISPLAY PROCEDURE REPORT FOLLOWS-

------ SCHEMA=PAYROLL
PROCEDURE          STATUS    ACTIVE    QUEUED  MAXQUE  TIMEOUT  WLM_ENV
PAYPRC1            STARTED        0         0       1        0  PAYROLL
PAYPRC2            STOPQUE        0         5       5        3  PAYROLL
PAYPRC3            STARTED        2         0       6        0  PAYROLL
USERPRC4           STOPREJ        0         0       1        0  SANDBOX

------ SCHEMA=HRPROD
PROCEDURE          STATUS    ACTIVE    QUEUED  MAXQUE  TIMEOUT  WLM_ENV
HRPRC1             STARTED        0         0       1        0  HRPROCS
HRPRC2             STOPREJ        0         0       1        0  HRPROCS
DSNX9DIS DISPLAY PROCEDURE REPORT COMPLETE
DSN9022I - DSNX9COM '-DISPLAY PROC' NORMAL COMPLETION
```

Example 4: Display information about all stored procedures in the SYSPROC schema that have been accessed by DB2 applications. Assume the -STOP PROCEDURE(SYSPROC.*) ACTION(QUEUE) command is in effect at the time this command is issued.

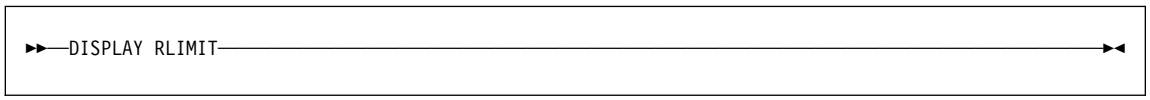
```
-DISPLAY PROCEDURE(SYSPROC.*)
```

This command produces output similar to the following output:

```
DSNX940I DSNX9DIS - DISPLAY PROCEDURE REPORT FOLLOWS-

------ SCHEMA=SYSPROC
PROCEDURE          STATUS    ACTIVE    QUEUED  MAXQUE  TIMEOUT  WLM_ENV
USERPRC2           STOPQUE        0         5       5        3  SANDBOX
USERPRC4           STOPQUE        0         0       1        0  SANDBOX
PROCEDURES USERFNC2           - USERFNC29999999999 STOP QUEUE
PROCEDURES USERFNC4           - USERFNC49999999999 STOP QUEUE
DSNX9DIS DISPLAY PROCEDURE REPORT COMPLETE
DSN9022I - DSNX9COM '-DISPLAY PROC' NORMAL COMPLETION
```

-DISPLAY RLIMIT (DB2)

Syntax



Example

Example: Display the current status of the resource limit facility.

```
-DISPLAY RLIMIT
```

If the resource limit facility (RLF) is inactive, the following output is generated:

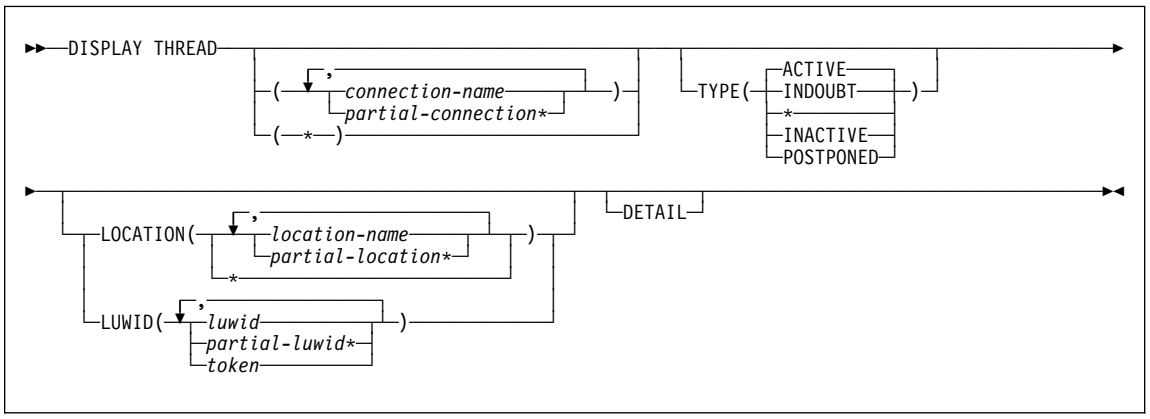
```
DSNT701I - RESOURCE LIMIT FACILITY IS INACTIVE
DSN9022I - DSNTCDIS 'DISPLAY RLIMIT' NORMAL COMPLETION
```

If the RLF is active, the value of field RESOURCE AUTHID on panel DSNTIPP is SYSADM, and the resource limit specification table with RLST NAME SUFFIX = 03 was started, the following output is generated:

```
DSNT700I = SYSADM.DSNRLST03 IS THE ACTIVE RESOURCE LIMIT
SPECIFICATION TABLE
DSN9022I = DSNTCDIS 'DISPLAY RLIMIT' NORMAL COMPLETION
```

-DISPLAY THREAD (DB2)

Syntax



Examples

Example 1: The output of the command DISPLAY THREAD shows a token for every thread, distributed or not. This example shows the token for an allied thread that is not distributed. The token is 123. You can use the thread's token as the parameter in the command CANCEL THREAD.

```
-DIS THD(*) DETAIL
```

This command produces output similar to the following output:

```
DSNV401I - DISPLAY THREAD REPORT FOLLOWS -
DSNV402I - ACTIVE THREADS -
NAME     ST A   REQ ID               AUTHID    PLAN      ASID TOKEN
BATCH    T  *     5 BKH2C            SYSADM    BKH2      000D   123
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I - DSNVDT '-DIS THD' NORMAL COMPLETION
```

Example 2: This example shows information about conversation activity when distribution information is displayed for active threads. DB2 returns the following message, indicating that the local site application is waiting for a conversation to be allocated in DB2, and a DB2 server that is accessed by a DRDA client using TCP/IP.

```
-DIS THD(*) LOCATION(*) DETAIL
```

This command produces output similar to the following output:

```
-DIS THD(*) LOC(*) DET
  DSNV401I - DISPLAY THREAD REPORT FOLLOWS -
  DSNV402I - ACTIVE THREADS -
     NAME     ST A   REQ ID           AUTHID   PLAN     ASID   TOKEN
     TSO      TR *     3 SYSADM       SYSADM   DSNESPRR 002E       2
      V436-PGM=DSNESPRR.DSNESM68, SEC=1, STMNT=116
      V444-DB2NET.LUND0.A238216C2FAE=2 ACCESSING DATA AT
      V446-USIBMSTODB22:LUND1
      V447--LOCATION        SESSID        A ST    TIME
      V448--USIBMSTODB22    0000000000000000 V A1 9015816504776
     TSO      RA *    11 SYSADM       SYSADM   DSNESPRR 001A      15
      V445-STLDRIV.SSLU.A23555366A29=15 ACCESSING DATA FOR 123.34.101.98
      V447--LOCATION        SESSID           A ST    TIME
      V448--123.34.101.98   446:3171        S2    9015611253108
     DISPLAY ACTIVE REPORT COMPLETE
  DSN9022I - DSNVDT '-DIS THD' NORMAL COMPLETION
```

Example 3: In this example, a system at STL has a TSO application and an IMS application. The system at STL fails after DB2 commits the TSO application, but before the commit decision has been communicated to the participant subsystems at SJ and LA. The failure occurs before IMS has communicated the commit or rollback decision to STL's DB2. The DISPLAY THREAD commands that are issued after the STL DB2 restarts but before reconnect with IMS. DISPLAY THREAD commands that are issued at each location show output similar to the following output:

At STL:

```
-DIS THD(*) TYPE(INDOUBT)
```

This command produces output similar to the following output:

```
DSNV401I - DISPLAY THREAD REPORT FOLLOWS -
DSNV406I - INDOUBT THREADS -
 COORDINATOR               STATUS     RESET URID         AUTHID
 STLIMS01                  INDOUBT          0F201050A010 SM09H
  V467-HAS LUWID IBM.STLDB21.15A86A876789.0010=1
  V449-HAS NID=A5 AND ID=STLIMS01
  V450-HAS PARTICIPANT INDOUBT AT
  V446--IBMSJ0DB20001:STLDB22
 IBMSTLDB20001             COMMITTED        0F20105B0000 JO78S
  V467-HAS LUWID IBM.STLDB21.16B57B954427.0003=2
  V450-HAS PARTICIPANT INDOUBT AT
  V446--IBMSJ0DB20001:STLDB22  IBMLA0DB20001:STLDB23
 DISPLAY INDOUBT REPORT COMPLETE -
 DSN9022I - DSNVDT '-DIS THD' NORMAL COMPLETION
```

At San Jose:

```
-DIS THD(*) TYPE(INDOUBT)
```

This command produces output similar to the following output:

```
 DSNV401I - DISPLAY THREAD REPORT FOLLOWS -
 DSNV406I - INDOUBT THREADS -
 COORDINATOR               STATUS     RESET URID         AUTHID
 IBMSTLDB20001:STLDB21     INDOUBT          03201050A010 HEU4443
  V467-HAS LUWID IBM.STLDB21.15A86A876789.0010=8
  V466-THREAD HAS BEEN INDOUBT FOR 00:05:20
 IBMSTLDB20001:STDB21      INDOUBT          03201050B000 PP433MM
  V467-HAS LUWID IBM.STLDB21.16B57B954427.0003=6
 DISPLAY INDOUBT REPORT COMPLETE
 DSN9022I - DSNVDT '-DIS THD' NORMAL COMPLETION
```

At Los Angeles (both ACTIVE and INDOUBT threads are displayed):

```
-DIS THD(*) TYPE(*) DETAIL
```

This command produces output similar to the following output:

-DISPLAY THREAD (DB2)

```
 DSNV401I - DISPLAY THREAD REPORT FOLLOWS -
 DSNV402I - ACTIVE THREADS -
 NAME     ST A   REQ ID           AUTHID   PLAN     ASID TOKEN
 SERVER   RA *     0 RUW2STAT     JONES    DISTSERV 0005     4
  V465-THREAD HAS BEEN PREPARED FOR 00:05:20
  V445-IBM.STLDB21.15A86A876789=4 ACCESSING DATA FOR
       IBMSJ0DB20001:STLDB21
  V447--LOCATION         SESSID           A ST TIME
  V448--IBMSJ0DB20001    0000000400000004 W R4 9034817015032
 DISPLAY ACTIVE REPORT COMPLETE
 DSNV406I - INDOUBT THREADS -
 COORDINATOR               STATUS     RESET URID         AUTHID
 IBMSTLDB20001:STLDB21     INDOUBT          03201050B000 SM43YY33
  V467-HAS LUWID IBM.STLDB21.16B57B954427.0003=5
  V466-THREAD HAS BEEN INDOUBT FOR 00:05:20
 DISPLAY INDOUBT REPORT COMPLETE
 DSN9022I - DSNVDT '-DIS THD' NORMAL COMPLETION
```

Example 9: This example shows the token for a thread that is executing a user-defined function. The token is 18.

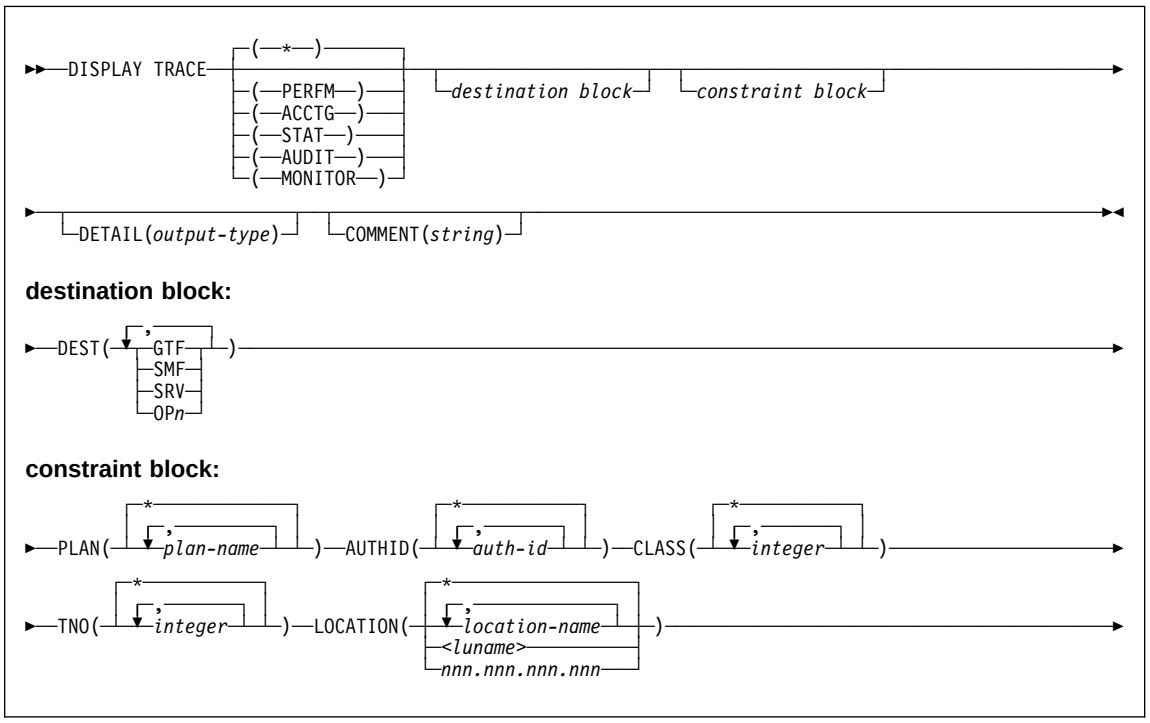
```
-DISPLAY THREAD(*) DETAIL
```

This command produces output similar to the following output:

```
DSNV401I - DISPLAY THREAD REPORT FOLLOWS -
DSNV402I - ACTIVE THREADS -
NAME     ST A   REQ ID           AUTHID   PLAN     ASID TOKEN
BATCH    T  *   231 DISTHD       ADMF001           0021    95
BATCH    SW *    38 INSERT       ADMF001  DSNTEP3  0025    18
 V436-PGM=CLIP74C1.UFIP74C1, SEC=0, STMNT=0
 V429 CALLING FUNCTION =SCIP7401.SP_UFIP74C1       ,
     PROC=V61AWLM3, ASID=0030, WLM_ENV=WLMENV3
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I - DSNVDT '-DISPLAY THREAD' NORMAL COMPLETION
```

-DISPLAY TRACE (DB2)

Syntax



Examples

Example 1: List all traces that have the generalized trace facility as their only destination.

```
-DISPLAY TRACE (*) DEST (GTF)
```

Example 2: List the trace started for Example 2 of -START TRACE.

```
-DISPLAY TRACE (ACCTG) PLAN (DSN8BC61)
  COMMENT ('ACCTG TRACE FOR DSN8BC61')
```

Example 3: List all active performance traces.

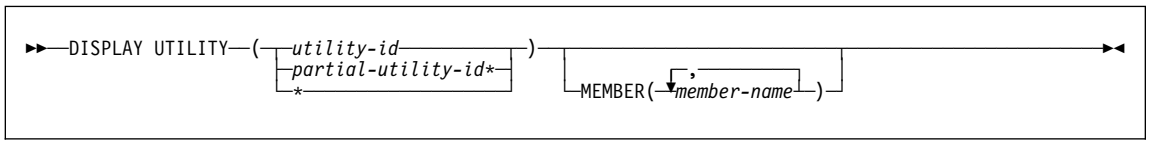
```
-DISPLAY TRACE=P
```

Example 4: List all active audit traces for threads that are connected to the DB2 subsystem with location name USIBMSTODB23.

```
-DISPLAY TRACE (AUDIT) LOCATION (USIBMSTODB23)
```

-DISPLAY UTILITY (DB2)

Syntax



Examples

Example 1: Display status information for all utility jobs currently known to DB2.

```
-DISPLAY UTILITY (*)
```

Example 2: Display the status of utilities on all members of the data sharing group.

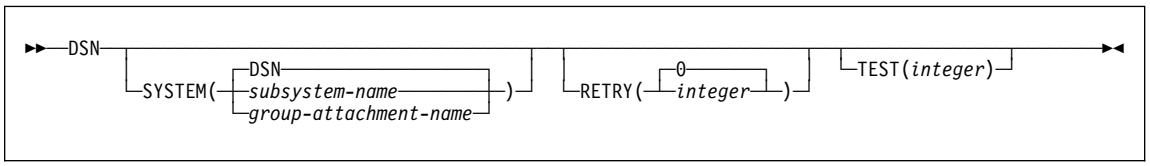
```
-DB1G DISPLAY UTILITY (*)
```

Example 3: In a data sharing environment, display the status of utilities on member DB1G.

```
-DB1G DISPLAY UTILITY (*) MEMBER (DB1G)
```

DSN (TSO)

Syntax



Examples

Example 1: Start a DSN session. If the attempt to connect to DB2 fails, five retries (at 30 second intervals) are to be made.

```
DSN SYSTEM (DB2) RETRY (5)
```

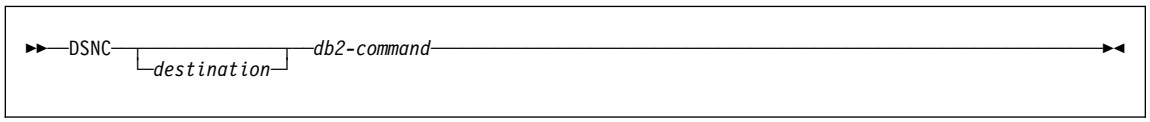
Example 2: Start a DSN session, run a program, and then end the session and return to TSO.

```
TSO prompt :  READY
USER enters:  DSN SYS (SSTR)
DSN prompt :  DSN
USER enters:  RUN PROGRAM (MYPROG)
DSN prompt :  DSN
USER enters:  END
TSO prompt :  READY
```

DSNC (CICS)

DSNC (CICS Attachment Facility)

Syntax



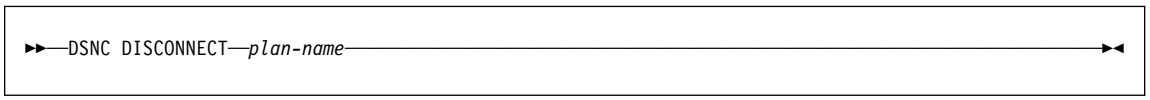
Example

Example: Issue the DB2 command -DISPLAY THREAD from a CICS terminal.

```
DSNC -DISPLAY THREAD
```

DSNC DISCONNECT (CICS Attachment Facility)

Syntax



Example

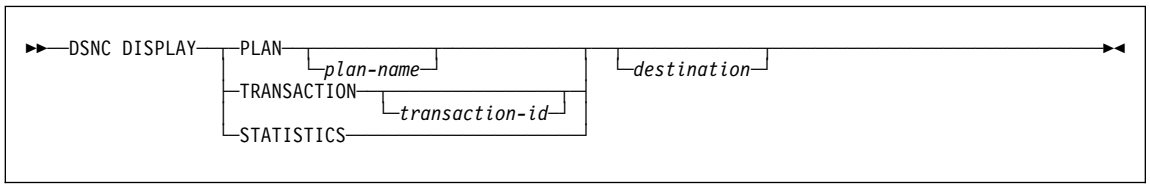
Example: Disconnect active threads for PLAN1.

```
DSNC DISC PLAN1
```

DSNC DISPLAY (CICS)

DSNC DISPLAY (CICS Attachment Facility)

Syntax



Examples

Example 1: Display information on all active plan IDs listed in the resource control table. The display information is to be sent to another terminal designated as MTO2.

```
DSNC DISP PLAN * MTO2
```

Example 2: Display information about all active transactions listed in the resource control table.

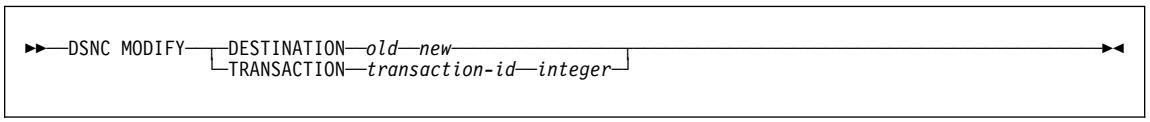
```
DSNC DISP TRANSACTION
```

Example 3: Display statistical counters associated with each entry in the resource control table.

```
DSNC DISP STAT
```

DSNC MODIFY (CICS Attachment Facility)

Syntax



Example

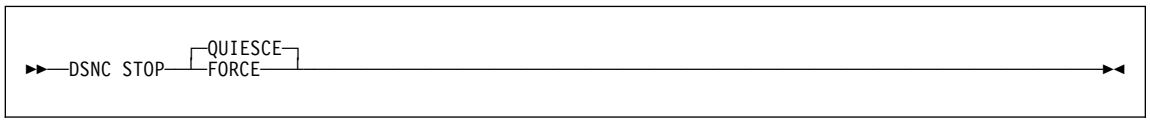
Example: Change the specification of the ERRDEST parameter in the resource control table from MTO1 to MTO2.

```
DSNC MODI DEST MTO1 MTO2
```

DSNC STOP (CICS)

DSNC STOP (CICS Attachment Facility)

Syntax



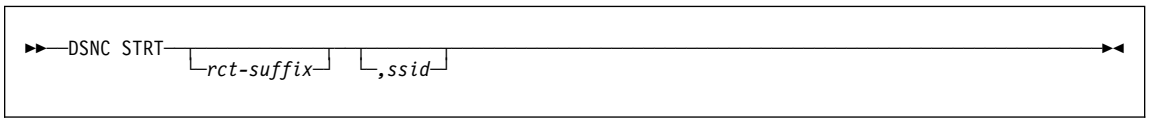
Example

Example: Stop the CICS attachment facility.

```
DSNC STOP FORCE
```

DSNC STRT (CICS attachment facility)

Syntax



Examples

Example 1: Start the CICS Version 4 Release 1 attachment facility. Use DSN2CT33 to connect to SSID DB2P.

```
DSNC STRT 33,DB2P
```

Example 2: Start the CICS Version 4 Release 1 attachment facility. Use the default resource control table with SSID DBA1.

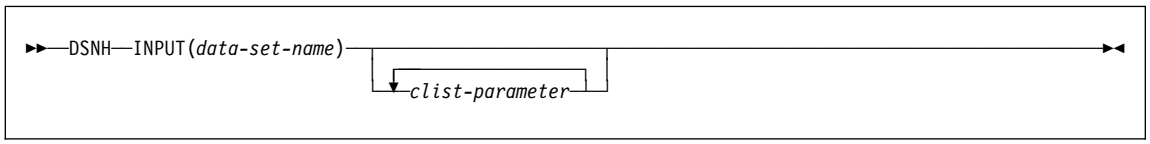
```
DSNC STRT ,DBA1
```

Example 3: Start the CICS Version 4 Release 1 attachment facility. Use DSN2CTA to connect to the SSID that is specified in DSN2CTA.

```
DSNC STRT A
```

DSNH (TSO CLIST)

Syntax



Examples

Example 1: Precompile, bind, compile, link-edit, and run the COBOL program in data set *prefix*.SDSNSAMP(DSN8BC4).

- The compiler load module is in SYS1.LINKLIB (IKFCBL00).
- Additional load modules to be included are in *prefix*.RUNLIB.LOAD and *prefix*.SDSNSAMP.
- The load module is be put into the data set *prefix*.RUNLIB.LOAD(DSN8BC4).
- The plan name is DSN8BC61 for the bind and run.
- DCLGEN data from *prefix*.SRCLIB.DATA is required for the precompile.

This example assumes that the DSNH CLIST is in your SYSPROC concatenation.

```
DSNH INPUT('prefix.SDSNSAMP(DSN8BC4)'') -
  COBLOAD('SYS1.LINKLIB(IKFCBL00)'') -
  LLIB('prefix.RUNLIB.LOAD'') -
  L2LIB('prefix.SDSNSAMP'') -
  LOAD('prefix.RUNLIB.LOAD'') -
  PLAN(DSN8BC61) -
  PLIB('prefix.SRCLIB.DATA'')
```

Example 2: Precompile, bind, compile, and link-edit the program in data set *prefix*.SDSNSAMP.PLI(DSN8BP4).

- The program is written in PL/I; the macro pass is not needed.
- The PL/I compiler options MAP and LIST are to be used.
- Additional load modules to be included are in *prefix*.RUNLIB.LOAD and *prefix*.SDSNSAMP.
- The PL/I optimizing compiler load module is in library SYS2.LINKLIB(IEL0AA).
- The DB2 subsystem identifier is SSTR.
- The load module is put into the data set *prefix*.RUNLIB.LOAD(DSN8BC4).
- Printed output is sent to the following data sets:

Output	Data Set
Precompiler listings	*prefix*.PROG.PCLIST
Compiler listings	*prefix*.PROG.LIST
Link edit listings	*prefix*.PROG.LINKLIST

- The plan name is DSN8BC61 for the bind and run.
- The DCLGEN data from *prefix*.SRCLIB.DATA is required for the precompile.

```
DSNH INPUT('prefix.SDSNSAMP(DSN8BP4)'') -
  HOST(PLI) MACRO(NO) -
  COPTION ('MAP LIST') -
  LLIB('prefix.RUNLIB.LOAD'') -
  L2LIB('prefix.SDSNSAMP'') -
  PLILOAD('SYS2.LINKLIB(IEL0AA)'') -
  SYSTEM(SSTR) -
  LOAD('prefix.RUNLIB.LOAD'') -
  PRINT(PROG) -
  PLAN(DSN8BC61) -
  PLIB('prefix.SRCLIB.DATA'')
```

The COPTION parameters are enclosed between single apostrophes so that they are passed by TSO as a single parameter. If a single token is being passed as a parameter, no apostrophes are needed. That same rule applies to the PARMS and CICSOPT parameters.

If a data set name is being passed as a parameter, and you want TSO to add your user prefix, no apostrophes are needed. If the usual TSO prefixing and suffixing must not be performed, the data set name must be enclosed between sets of three apostrophes if the CLIST is executed implicitly, and sets of six apostrophes if the CLIST is executed explicitly.

The user prefix for that example is *prefix*; if it had been SMITH, the listing data set names would be as shown above, except that SMITH would be used as the first level qualifier. For example, the compiler listings would have gone to SMITH.PROG.LIST.

Example 3: Invocation of the DB2-C sample application program *prefix*.SDSNSAMP(DSN8BD3).

- The C linkage editor include library is EDC.V1R1M1.SEDCBASE
- The C compiler load module is EDC.V1R1M1.SEDCCOMP(EDCCOMP)
- Printed output is sent to the following data sets:

Output	Data Set
Precompiler listings	*user_id*.TEMP.PCLIST
Compiler listings	*user_id*.TEMP.SYSCPRT.LIST
Prelink utility listings	*user_id*.TEMP.SYSOUT.PRELLIST
Link-edit listings	*user_id*.TEMP.LINKLIST

- The following C DD names are allocated based on the PRINT keyword value:

DD Name	Allocation
SYSCPRT	Used in the compile step
SYSUT10	Used in the compile step
SYSOUT	Used in the prelink step.

  SYSUT10 and SYSCPRT are always allocated to the same data set or destination.

- SYSTERM is used in the compile step. It is based on the TERM keyword.
- CEEDUMP is used in the run step. It is based on the RUNOUT keyword.

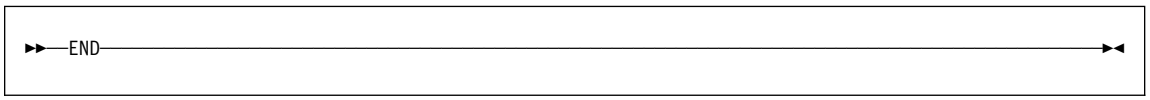
DSNH (TSO CLIST)

- The LOPTION keyword values of AMODE(31) and RMODE(ANY) are required when link editing the C sample program to insure 31-bit addressability during execution.

```
ALLOC      DD(SYSPROC) DSN('prefix.SDSNCLST ') SHR
%DSNH BIND(YES) ACQUIRE(USE) ACTION(REPLACE)-
 EXPLAIN(NO) -
 CICSXLAT(NO) -
 COMPILE(YES) -
 CCLLIB('EDC.V1R1M1.SEDCBASE'')-
 CCLOAD('EDC.V1R1M1.SEDCCOMP(EDCCOMP)'')-
 DBRM('prefix.DBRMLIB.DATA(DSN8BD3)'')-
 DECIMAL(PERIOD) DELIMIT(DEFAULT) FLAG(I)-
 HOST(C) ISOLATION(RR)-
 INPUT('prefix.SDSNSAMP(DSN8BD3)'')-
 LINK(YES)-
 LLIB('prefix.RUNLIB.LOAD'')-
 L2LIB('prefix.SDSNLOAD'')-
 LOAD('prefix.RUNLIB.LOAD'')-
 LOPTION('AMODE(31) RMODE(ANY)')-
  MACRO(NO)-
 OUTNAME(TEMP)-
 PLAN(DSN8BD31) PRECOMP(YES)-
 PLIB('prefix.SDSNSAMP'')-
 PRELINK(NO)-
 POPTION(NONE)-
 PRINT(TEMP) RCTERM(8)-
 RELEASE(COMMIT) RETAIN(YES)-
 RUN(NO) RUNIN(TERM)-
 RUNOUT(TERM) SOURCE(YES)-
 SYSTEM(DSN) SQLDELIM(DEFAULT)-
 VALIDATE(RUN)
```

END (DSN)

Syntax



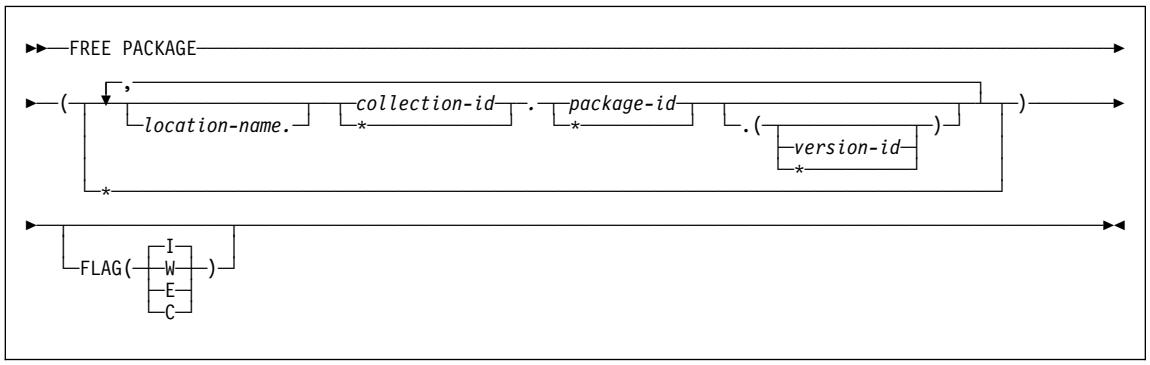
Example

Example: End the DSN session and return to TSO.

```
TSO prompt :  READY
USER enters:  DSN SYS (SSTR)
DSN prompt :  DSN
USER enters:  RUN PROGRAM (MYPROG)
DSN prompt :  DSN
USER enters:  END
TSO prompt :  READY
```

FREE PACKAGE (DSN)

Syntax



Examples

Example 1: Free version *newver* of the package TEST.DSN8BC61 located at USIBMSTODB22. Generate only warning, error, and completion messages (not informational messages).

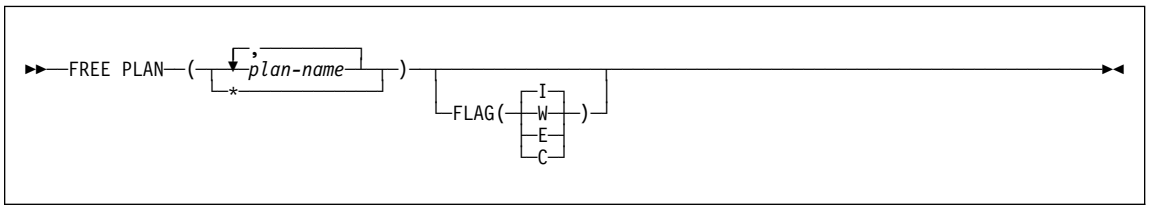
```
FREE PACKAGE (USIBMSTODB22.TEST.DSN8BC61.(newver)) FLAG(W)
```

Example 2: Free all packages at the local server in the collection named TESTCOLLECTION.

```
FREE PACKAGE (TESTCOLLECTION.*)
```

FREE PLAN (DSN)

Syntax



Example

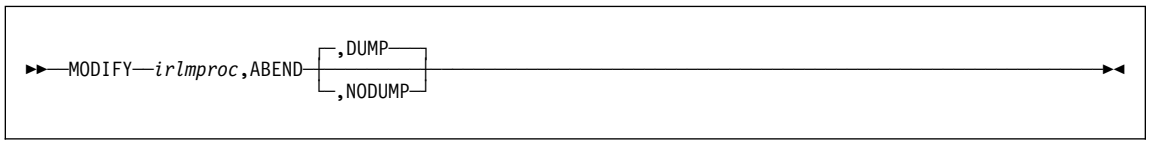
Example: Free plan DSN8BC61 from DB2. Generate only warning, error, and completion messages (not informational messages).

```
FREE PLAN (DSN8BC61) -
  FLAG (W)
```

MODIFY ...,ABEND (MVS IRLM)

MODIFY irlmproc,ABEND (MVS IRLM)

Syntax



Example

Example: Enter on an MVS system console:

```
F KRLM1,ABEND
```

Response on the MVS system console:

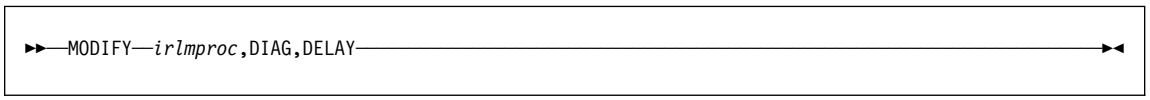
```
 DXR124E IR21 ABENDED VIA MODIFY COMMAND
*IEA911E COMPLETE DUMP ON SYS1.DUMP00
  FOR ASID(0004)
   ERROR ID = SEQ00001 CPU00 ASID0004 TIME08.34.59.9
 DXR121I IR21 END-OF-TASK CLEANUP SUCCESSFUL
 IEF450I IR210 IR210 - ABEND=S000 U2020 REASON=00000000
```

The default is dump. If you do not want a dump, you must specify:

```
F KRLM1,ABEND,NODUMP
```

MODIFY irlmproc,DIAG,DELAY (MVS IRLM)

Syntax



Example

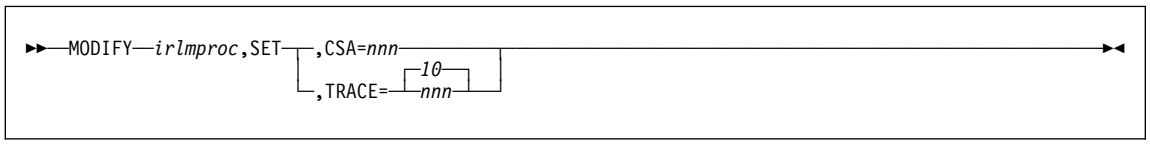
Example 1: Issue this command to initiate one diagnostic dump for the IR21PROC IRLM subsystem. The dump occurs once, after the propagation of child locks takes longer than 45 seconds.

```
MODIFY IR21PROC,DIAG,DELAY
```

MODIFY ...,SET (MVS IRLM)

MODIFY irlmproc,SET (MVS IRLM)

Syntax



Examples

Example 1: Enter on an MVS system console:

```
F IR21PROC,SET,CSA=10
```

Response on the MVS system console:

```
DXR178I IR21033 MAXIMUM CSA IS SET TO 10MB
```

Explanation: IR21033 is the IRLM subsystem name concatenated with the IRLM system ID.

Example 2: Enter on an MVS system console:

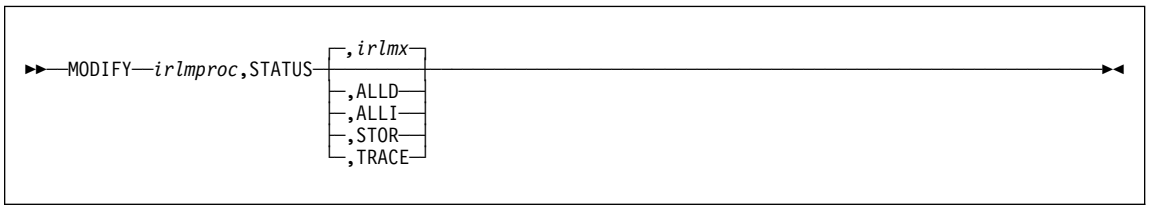
```
F IR21PROC,SET,TRACE=20
```

Response on the MVS system console:

```
DXR177I IR21033 THE MAXIMUM NUMBER OF TRACE BUFFERS
                FOR EACH TRACE TYPE IS SET TO 20
```

MODIFY irlmproc,STATUS (MVS IRLM)

Syntax



Examples

Example 1: Enter on the MVS1 system console:

```
F IRTPROC,STATUS
```

Response on MVS1 system console:

```
DXR101I IR2T STATUS SCOPE=LOCAL
        SUBSYSTEMS IDENTIFIED                PT01
        NAME      STATUS    UNITS     HELD      WAITING     RET_LKS
        DSNT1     UP-NS     0005      0010      0002        0
```

Explanation: The operator on system 1 has requested information about the DB2 systems connected to the IRLM identified by the IRLM procedure named IRTPROC.

If the IRLM is SCOPE=GLOBAL on the irlmproc and is not connected to any group, the status message shows:

```
DXR101I IR21 STATUS SCOPE=DISCON
```

Example 2: Assume you have a data sharing group. Enter on a system console:

```
F DB1GIRLM,STATUS,ALLD
```

Response on system console:

```
14.02.10 STC00086  DXR102I DJ1G STATUS IRLMID=001
        SUBSYSTEMS IDENTIFIED                PT01
        NAME      STATUS    RET_LKS   IRLMID  IRLM_NAME
        DB4G      UP              0    004       DJ4G
        DB3G      UP              0    003       DJ3G
        DB2G      UP              0    002       DJ2G
        DB1G      UP              0    001       DJ1G
```

Explanation: The output shows all the DB2s that are connected to IRLMs in this data sharing group (the group to which the IRLM processing the request belongs). The value “UP” in the STATUS field indicates that the DB2 is active. Other possible values for STATUS include:

MODIFY ...,STATUS (MVS IRLM)

DOWN — The DB2 is failed. All “modify” type locks held by this DB2 have been retained by IRLM. The DB2 is known to be down only if it has retained locks.

SYSFAIL — The IRLM that DB2 is identified to has been disconnected from the data sharing group. All “modify” type locks held by this DB2 have been retained by IRLM. The DB2 is known to be SYSFAIL only if it has retained locks.

Example 3: Again, assume data sharing is in effect. Enter the following on the system console:

```
F DB1GIRLM,STATUS,ALLI
```

The response on the console is:

```
17.17.03 STC00092  DXR103I LRLM STATUS IRLMID=007
        IRLMS PARTICIPATING IN DATA SHARING GROUP FUNCTION LEVEL=006
        IRLM_NAME IRLMID STATUS LEVEL SERVICE MIN_LEVEL MIN_SERVICE
          JRLM      005    UP    013  PN92893    006      IRLM2.1
          KRLM      006    UP    006  IRLM2.1    006      IRLM2.1
          LRLM      007    UP    013  PN92893    006      IRLM2.1
```

Explanation: The output shows the IRLMs that are participating in this data sharing group (the group which includes the IRLM processing the request). Other information includes:

STATUS The value “UP” in the STATUS field indicates that the IRLM is active. STATUS shows “DOWN” if the IRLM is failed. An IRLM is known to be “DOWN” only if the DB2 that was identified to it has retained locks. This connection between a failed DB2 and IRLM is lost after a rebuild or a group restart.

LEVEL The current IRLM function level.

SERVICE The IRLM service or release that corresponds to the function level given in "LEVEL".

MIN_LEVEL The minimum IRLM function level this IRLM can coexist with.

MIN_SERVICE The IRLM service or release that corresponds to the function level given in "MIN-LEVEL".

Group Function Level The IRLM function level in use by all the IRLMs in the data sharing group.

Example 4: Assume that this command is issued in a non-data sharing environment. Enter the following on the system console:

```
F DB1GIRLM,STATUS,ALLI
```

The response on the console is:

```
15.12.01 STC00092  DXR103I VRLM STATUS IRLMID=007
        IRLMS PARTICIPATING IN DATA SHARING GROUP FUNCTION LEVEL=016
        IRLM_NAME IRLMID STATUS LEVEL SERVICE MIN_LEVEL MIN_SERVICE
          VRLM      007    UP    016  PQ15854    012      PN90337
```

Explanation: The output shows information only for the IRLM specified. The group function level shown is the function level for the specified IRLM. Refer to Example 3 on page 276 for additional information on interpreting output.

Example 5: Enter the following command on the system console:

```
F IR21PROC,STATUS,STOR
```

The response on the console is:

```
DXR100I IR21 STOR STATS
        PC: NO    MAXCSA:    6M
CSA USE: ACNT:    132K  AHWM:    132K  CUR:  4048K  HWM:  4086K
        ABOVE 16M:     72  4033K    BELOW 16M:      6     15K
CLASS   TYPE  SEGS     MEM   TYPE  SEGS     MEM   TYPE  SEGS     MEM
ACCNT   T-1      1     64K   T-2      1     64K   T-3      1      4K
PROC    WRK     11     58K   SRB      3      3K   OTH      2      2K
MISC    VAR     60   4081K   N-V      6     22K   FIX      1     24K
```

Explanation: The example shows that current storage allocated for IRLM is 4048 KB, and the greatest amount that has been allocated since the last time IRLM was started is 4086KB. The storage for the locking structures (RHB and RLB) is contained within ECSA, because this IRLM is defined with PC=NO. Use the following information to interpret the display output:

PC — Displays the current value for the PC option of the IRLM startup procedure.

MAXCSA — Displays the current value for the MAXCSA option of the IRLM startup procedure. The MAXCSA value is 6MB in this example.

CSA USE — Shows storage use that is accountable toward the MAXCSA value of the IRLM procedure. In this output, the current use accountable storage (ACNT) is 132KB. The high water mark since the last time IRLM was started (AHWM) is also 132KB.

CUR — Shows the total current CSA and ECSA usage. In this case, the current usage (CUR) is 4048KB, and the high water mark (HWM) is 4086KB. The accountable storage is a subset of this total storage.

ACCNT — The ACCNT row of the report is a breakdown of lock control block structures and their storage use.

T-1 Type one structures are for resources. In this case, it shows that one storage segment is held for a total of 64KB.

T-2 Type two structures are for all resource requests after the first request for a specific resource. This example shows that one storage segment is held for a total of 64KB.

T-3 Type three structures are for requesters or work units that are waiting for or hold resources. This example shows that one storage segment is held for a total of 4KB.

PROC and MISC rows These rows contain usage information for CSA, ECSA, and private storage used to process DBMS requests. Use this information under the guidance of IBM® Support Center for diagnosing problems.

For more information, see the explanation of message DXR100I in *DB2 Messages and Codes*.

Example 6: Assume the IRLM was started with PC=YES. Enter the following command on the system console:

```
F IR21PROC,STATUS,STOR
```

The response on the console is:

```
DXR100I JR21 STOR STATS
        PC: YES   MAXCSA:    N/A
CSA USE: ACNT:      OK  AHWM:      OK  CUR:  4362K  HWM:  5830K
        ABOVE 16M:     78  4376K    BELOW 16M:     23     32K
CLASS   TYPE  SEGS     MEM   TYPE  SEGS     MEM   TYPE  SEGS     MEM
ACCNT   T-1      1     64K   T-2      1     64K   T-3      1      4K
PROC    WRK     11     58K   SRB     20     20K   OTH      2      2K
MISC    VAR     68   4497K   N-V      6     22K   FIX      1     24K
```

Explanation: This example was created using a poorly-tuned application, and shows how important a well-tuned system is for predicting system storage needs. This example illustrates what can happen when an application generates a high IRLM lock contention rate; the high value of 20 segments with 20KB each for type SRB storage, and the high value of 23 segments with 23KB each for storage below the 16MB line are some of the results.

For more information about reducing lock contention, see Section 5 (Volume 2) of *DB2 Administration Guide*. For more information about tuning your system, see Chapter 7 of *DB2 Data Sharing: Planning and Administration*.

Example 7: Enter the following command on the system console:

```
F PR21PROC,STATUS,TRACE
```

The response on the console is:

```
DXR179I PR21034 TRACE USAGE
TRACE BUFFER STORAGE IN USE:   256KB
MAXIMUM NUMBER OF TRACE BUFFERS ALLOWED PER TRACE TYPE:   10
TRACE TYPE   ACTIVE   BUFFERS IN USE   CTRACE WRITER
----------   ------   --------------   -------------
   SLM          N            0               N
   XIT          Y            2               N
   XCF          N            0               N
   DBM          N            0               N
   EXP          Y            1               N
   INT          Y            1               N
```

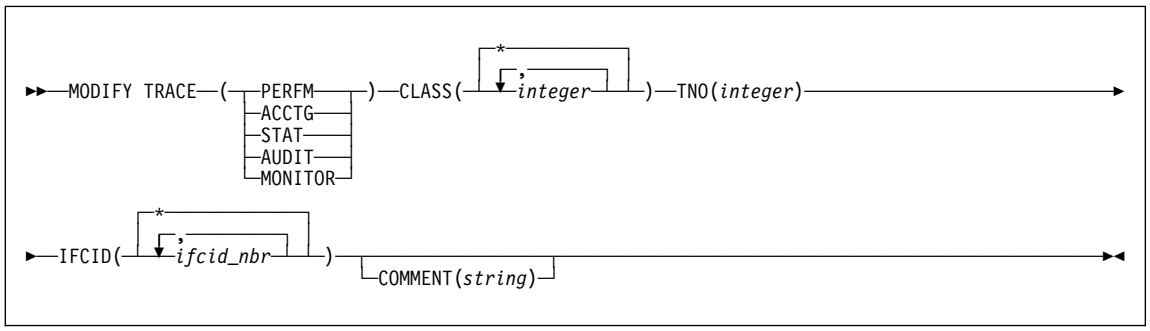
Explanation: This example shows that the storage currently allocated for IRLM tracing is 256KB, the maximum number of trace buffers allowed per trace type is set to 10, and the external CTRACE writer is not active. For more information about the trace types, see “TRACE CT (MVS IRLM)” on page 321.

Use the TRACE CT command of MVS on page 321 to activate or deactivate traces. You cannot turn off the EXP and INT traces. The XIT (for data sharing), EXP, and INT traces are automatically activated when you start IRLM. All traces are automatically activated with IRLMPROC TRACE=YES.

The trace size for each buffer is 64KB. Use the MODIFY irlmproc,SET,TRACE=*nnn* command to change the maximum number of trace buffers.

-MODIFY TRACE (DB2)

Syntax



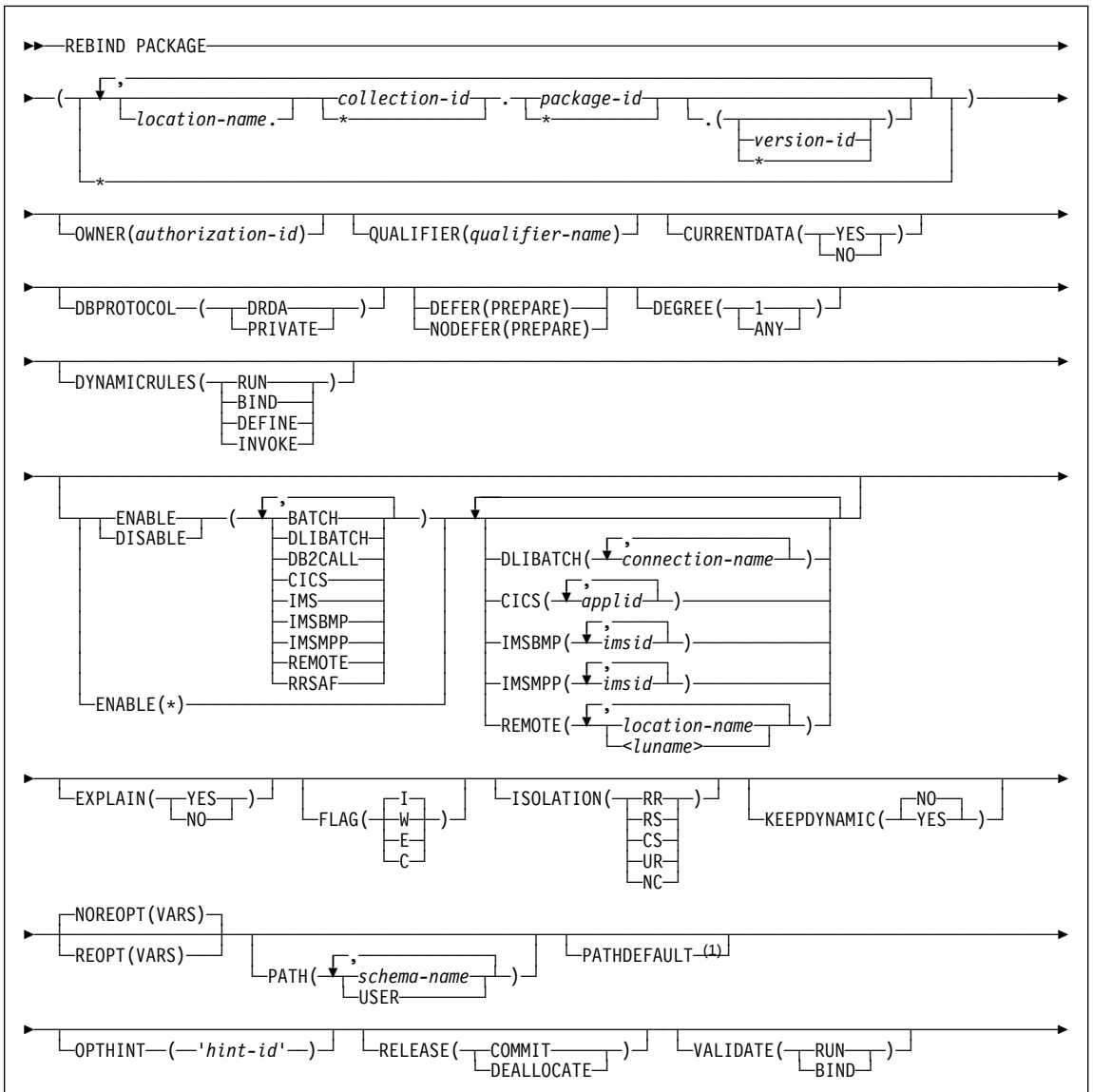
Example

Example: Change trace number 6 so that it collects only statistics and accounting data. You can define CLASS(30) at your site.

```
-MODIFY TRACE(S) IFCID(1,2,3) TNO(6) CLASS(30)
  COMMENT ('STATS AND ACCOUNTING ON')
```

REBIND PACKAGE (DSN)

Syntax



Note:

1 The **PATHDEFAULT** keyword is mutually exclusive with the **PATH** keyword. Do not specify both keywords in the same REBIND command.

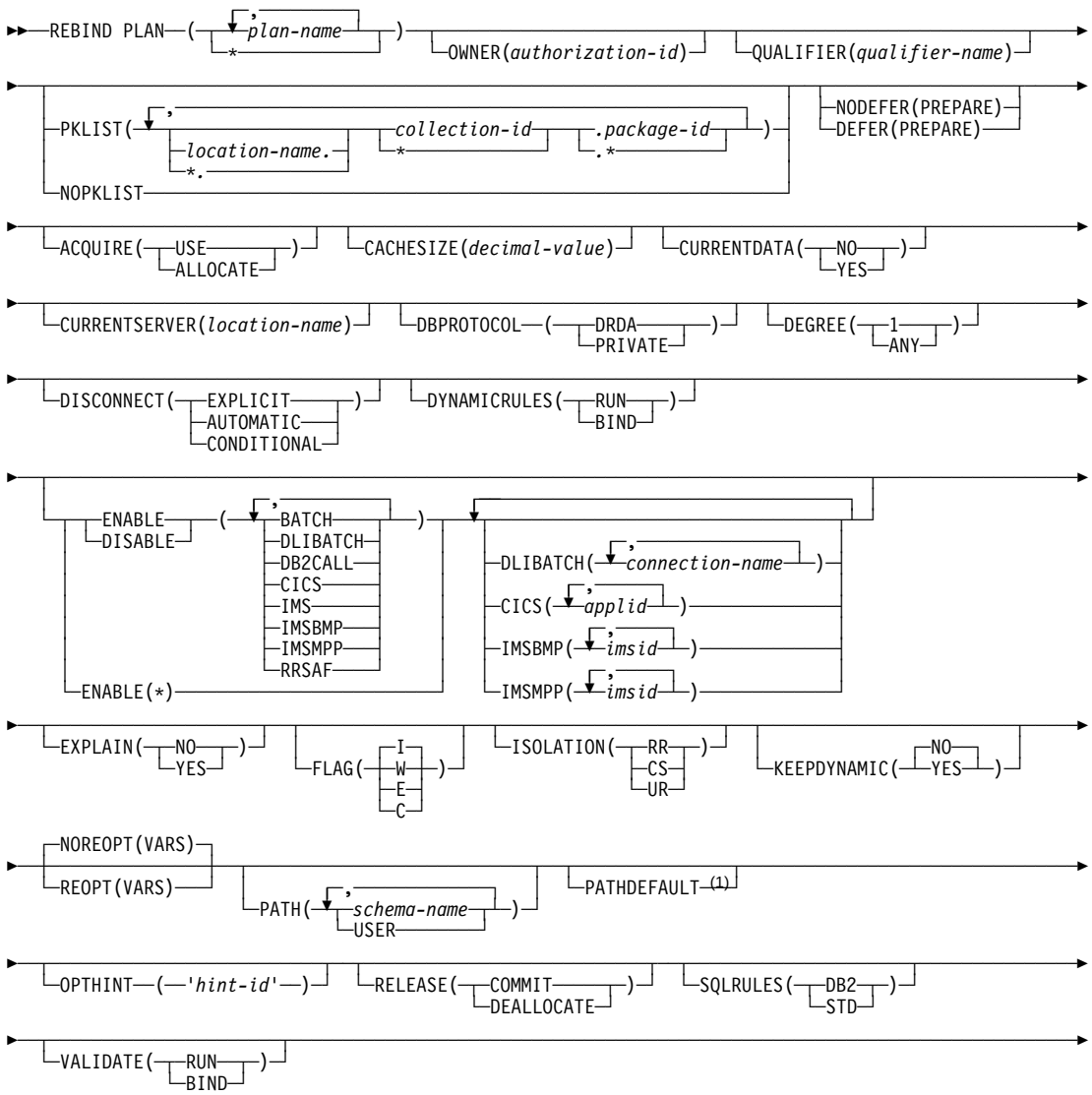
Example

Example: Rebind packages TEST.DSN8BC61.(MAY_VERSION) and PRODUCTION.DSN8BC61.(DEC_VERSION), both located at the local location USIBMSTODB22. The packages can run only from the CICS or the DLIBATCH environments if the connection ID is CON2. This replaces the CON1 specified on the BIND PACKAGE command.

```
REBIND PACKAGE (USIBMSTODB22.TEST.DSN8BC61.(MAY_VERSION),
                USIBMSTODB22.PRODUCTION.DSN8BC61.(DEC_VERSION)) -
  ENABLE (CICS,DLIBATCH) CICS (CON2)
```

REBIND PLAN (DSN)

Syntax



Note:

1 The **PATHDEFAULT** keyword is mutually exclusive with the **PATH** keyword. Do not specify both keywords in the same REBIND command.

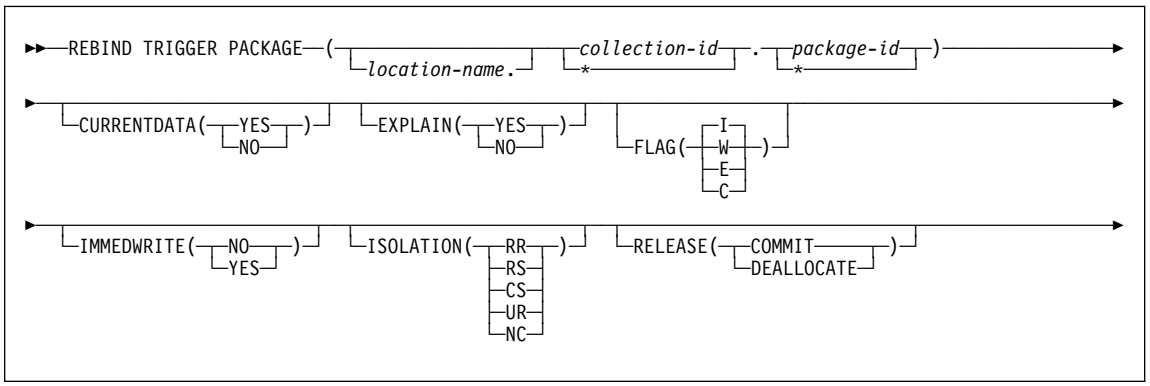
Example

Example: Rebind plan DSN8BC61 to enable DB2 to take advantage of a newly created index. Use FLAG(W) to issue warning, error, and completion messages, but not informational messages. Use VALIDATE(BIND) to point out any error conditions during the bind process. Use ISOLATION(CS) to prevent other applications from changing the database values that this application uses only while the application is using them. This isolation level protects changed values until the application commits or terminates. Omit the OWNER keyword to leave the plan's owner authorization ID the same. Omit the ENABLE or DISABLE keywords to use the connections previously defined for the plan.

```
REBIND PLAN (DSN8BC61) -
  FLAG (W) -
  VALIDATE (BIND) -
  ISOLATION (CS)
```

REBIND TRIGGER PACKAGE (DSN)

Syntax



Output

REBIND TRIGGER PACKAGE updates the COLLID and NAME columns in the SYSPACKAGE catalog table.

Example

Example: Issue the following command to rebind trigger package TRIG1 in the ADMF001 collection of packages:

```
REBIND TRIGGER PACKAGE (ADMF001.TRIG1);
```

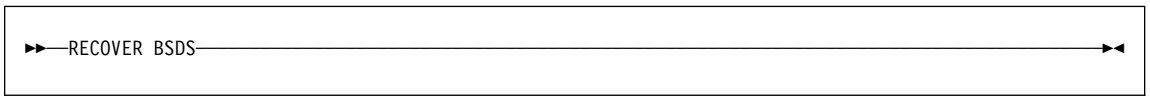
This command produces output similar to the following output:

REBIND TRIGGER PACKAGE (DSN)

```
DSNT254I - DSNTBRB2 REBIND OPTIONS FOR
           PACKAGE = STLEC1.ADMF001.TRIG1.()
           ACTION
           OWNER         ADMF001
           QUALIFIER     ADMF001
           VALIDATE      BIND
           EXPLAIN       NO
           ISOLATION     CS
           RELEASE       COMMIT
           COPY
DSNT255I - DSNTBRB2 REBIND OPTIONS FOR
           PACKAGE = STLEC1.ADMF001.TRIG1.()
           SQLERROR      NOPACKAGE
           CURRENTDATA   YES
           DEGREE        1
           DYNAMICRULES  BIND
           NODEFER       PREPARE
           NOREOPT       VARS
           KEEPDYNAMIC   NO
           DBPROTOCOL    DRDA
           QUERYOPT      1
           PATH
"SYSIBM","SYSFUN","SYSPROC","SYSADM","ADMF001"
DSNT232I - SUCCESSFUL REBIND FOR
           PACKAGE = STLEC1.ADMF001.TRIG1.()
```

-RECOVER BSDS (DB2)

Syntax



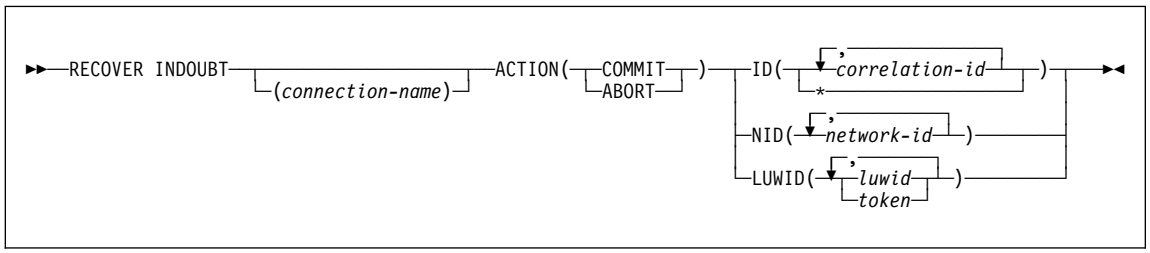
Example

Example: Reestablish dual BSDS mode.

```
-RECOVER BSDS
```

-RECOVER INDOUBT (DB2)

Syntax



Examples

Example 1: Recover indoubt allied threads. Schedule a commit for all threads associated with the connection name from which the command is entered.

```
-RECOVER INDOUBT ACTION(COMMIT) ID(*)
```

Example 2: Recover an indoubt thread from a remote requester. Schedule a commit for the indoubt thread whose token is 1332.

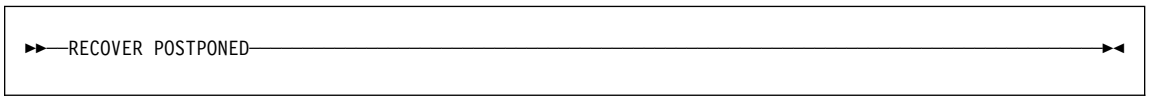
```
-RECOVER INDOUBT ACTION(COMMIT) LUWID(1332)
```

Example 3: Recover indoubt threads from remote requesters. Schedule an abort for two indoubt threads. The first has an LUWID = DB2NET.LUNSITE0.A11A7D7B2057.0002 (the '0002' in the last segment of the LUWID represents the commit sequence number). The second has a token = 442.

```
-RECOVER INDOUBT ACTION(ABORT)
         LUWID (DB2NET.LUNSITE0.A11A7D7B2057.0002, 442)
```

-RECOVER POSTPONED (DB2)

Syntax



Example

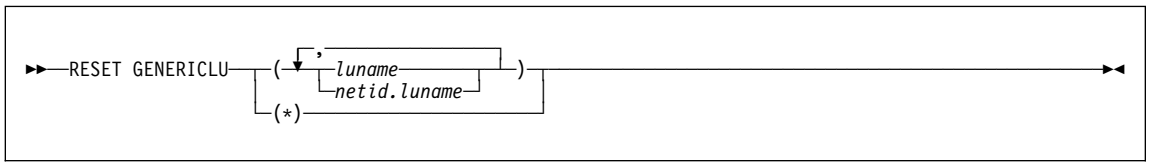
Example 1: Issue the following command to recover postponed-abort units of recovery.

```
-RECOVER POSTPONED
```

-RESET GENERICLU (DB2)

-RESET GENERICLU (DB2)

Syntax



Examples

Example 1: Purge the VTAM generic name mapping associated with partner NET1.USER5LU.

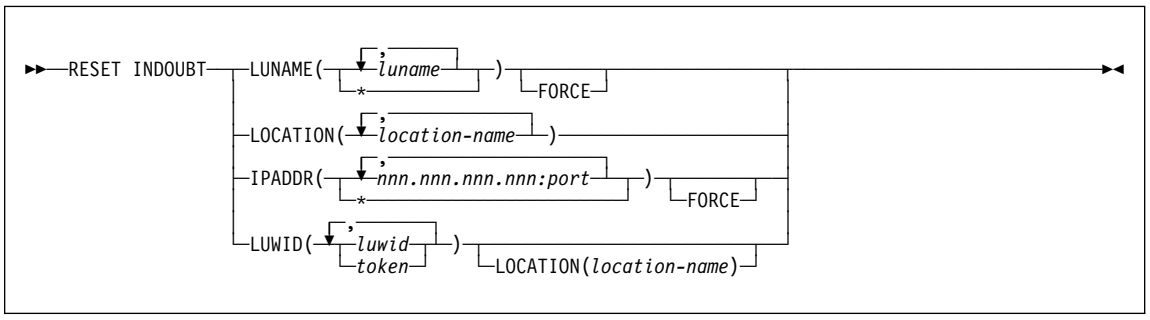
```
-DB2A RESET GENERICLU(NET1.USER5LU)
```

Example 2: Purge the VTAM generic name mappings for all LUs that are partners of this DB2 subsystem. Use this version of the command only when removing this DB2 from the data sharing group.

```
-DB2A RESET GENERICLU(*)
```

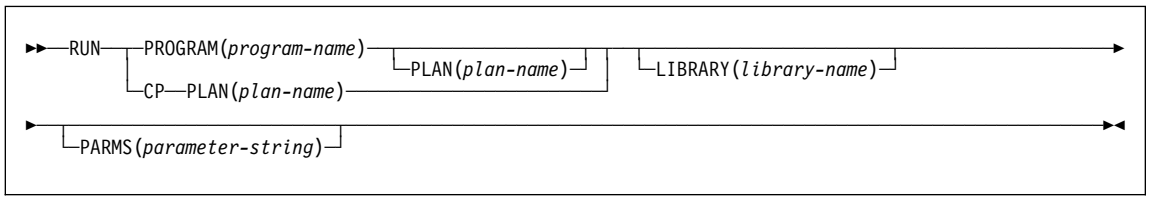
-RESET INDOUBT (DB2)

Syntax



RUN (DSN)

Syntax



Examples

Example 1: Run application program DSN8BC4. The application plan has the same name. The program is in library '*prefix*.RUNLIB.LOAD'.

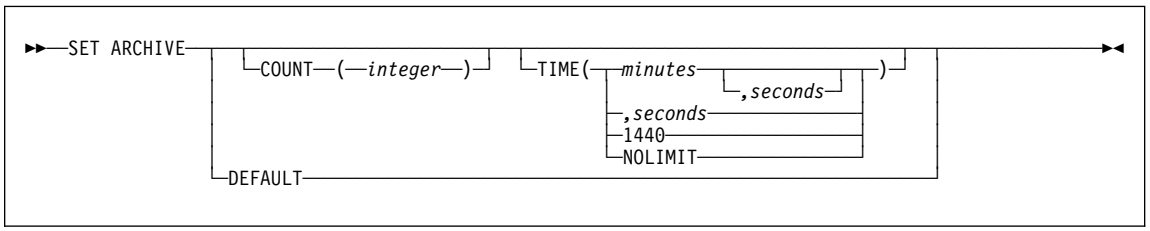
```
DSN SYSTEM (DSN)
RUN PROGRAM (DSN8BC4) LIB ('prefix.RUNLIB.LOAD')
```

Example 2: Run application program DSN8BP4. The application plan is DSN8BE61. The program is in library '*prefix*.RUNLIB.LOAD'. Pass the parameter O'TOOLE to the PL/I application program with no PL/I run-time options.

```
DSN SYSTEM (DSN)
RUN PROGRAM (DSN8BP4) PLAN (DSN8BE61) -
    LIB ('prefix.RUNLIB.LOAD') PARMS ('/O'TOOLE')
```

-SET ARCHIVE (DB2)

Syntax



Examples

Example 1: Allocate 2 tape units that can remain unused for 30 seconds before they are deallocated.

```
-SET ARCHIVE COUNT(2) TIME(,30)
```

Example 2: Allocate 4 tape units that can remain unused for 2 minutes before they are deallocated.

```
-SET ARCHIVE COUNT(4) TIME(2)
```

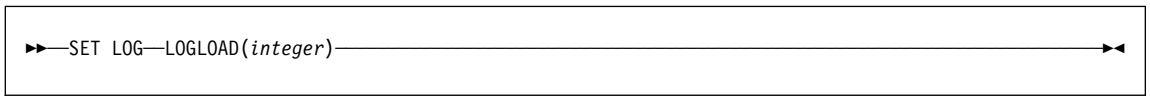
Example 3: Allocate 1 tape unit that is never deallocated.

```
-SET ARCHIVE COUNT(1) TIME(1440)
```

SET LOG (DB2)

SET LOG (DB2)

Syntax



Examples

Example 1: Initiate a system checkpoint without modifying the current LOGLOAD value.

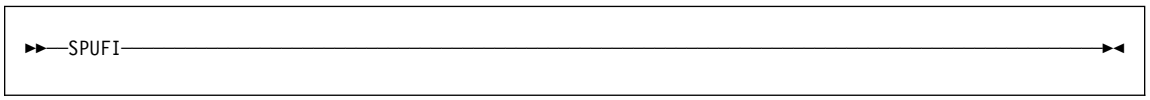
```
SET LOG LOGLOAD(0)
```

Example 2: Modify the system checkpoint interval to every 150000 log records.

```
SET LOG LOGLOAD(150000)
```

SPUFI (DSN)

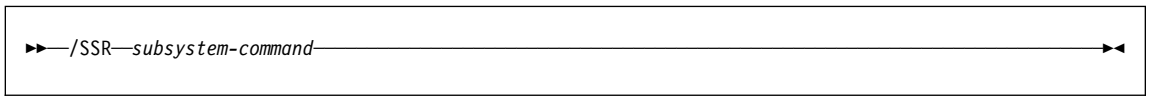
Syntax



/SSR (IMS)

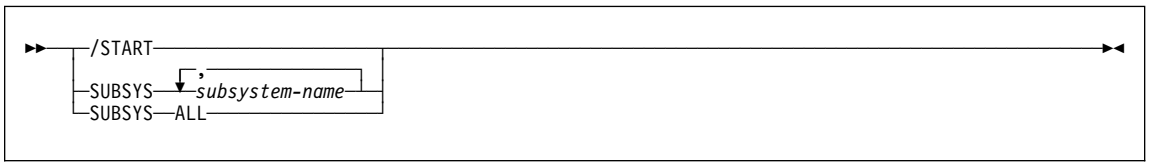
/SSR (IMS)

Syntax



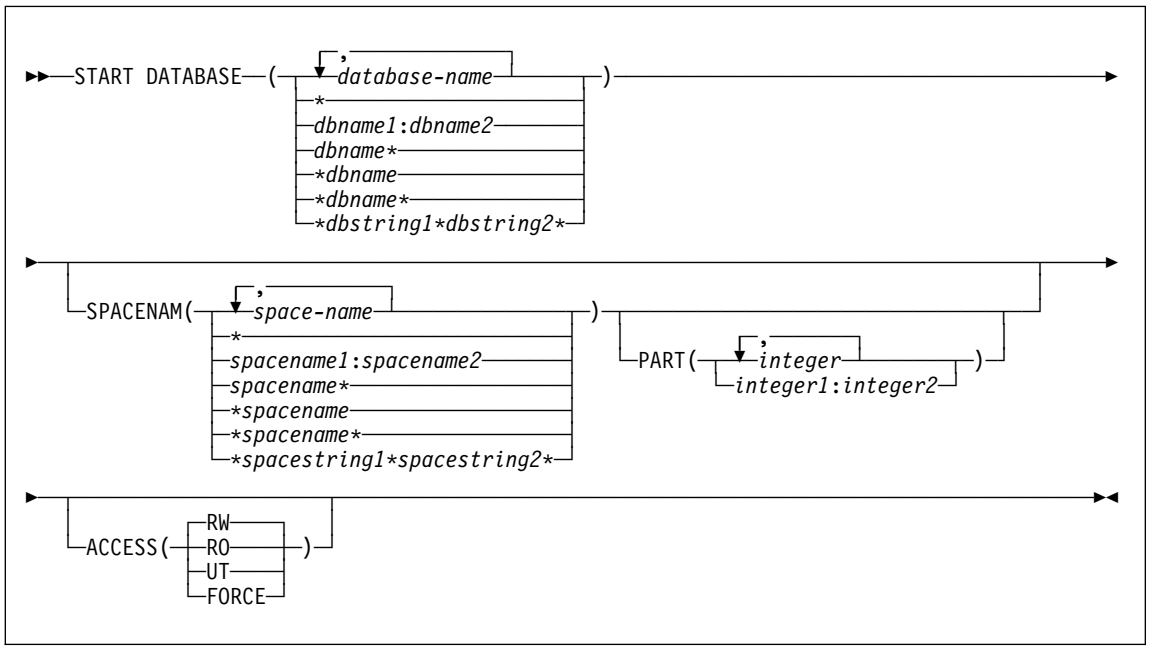
/START (IMS)

Syntax



-START DATABASE (DB2)

Syntax



Examples

Example 1: Start table space DSN8S61E in database DSN8D61A. Recover the table space if it is in GRECP status or recover the pages on the LPL if one exists.

```
-START DATABASE (DSN8D61A) SPACENAM (DSN8S61E)
```

Example 2: Start all databases (except DSNDB01, DSNDB06, and work file databases) for which you have authority. Recovery for any objects with GRECP or LPL status is not performed.

```
-START DATABASE (*)
```

Example 3: Start the third and fourth partitions of table space DSN8S61E in database DSN8D61A for read-only access. Recover the partitions if they are in GRECP status or recover the pages on the LPL if one exists.

```
-START DATABASE (DSN8D61A) SPACENAM (DSN8S61E) PART (3,4) ACCESS (RO)
```

Example 4: Start all table spaces that begin with "T" and end with the string "IQUA03" in database DBIQUA01 for read and write access.

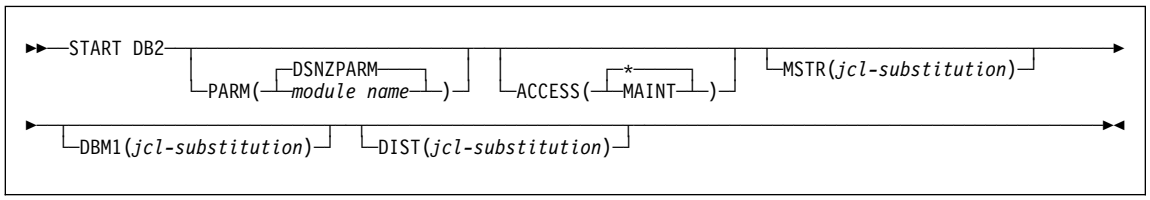
```
-START DATABASE (DBIQUA01) SPACENAM (T*IQUA03) ACCESS (RW)
```

This command produces output similar to the following output:

```
DSN9022I - DSNTDDIS 'START DATABASE' NORMAL COMPLETION
```

-START DB2 (DB2)

Syntax



Examples

Example 1: Start the DB2 subsystem.

```
-START DB2
```

Example 2: Start the DB2 subsystem and provide a new value for the REGION parameter in the startup procedure for the system services address space.

```
-START DB2 MSTR('REGION=6000K')
```

Example 3: Start the DB2 subsystem. Assuming that the EXEC statement of the JCL that executes the startup procedure for the system services address space uses the symbol RGN, provide a value for that symbol.

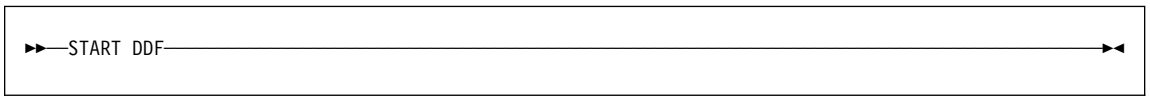
```
-START DB2 MSTR('RGN=6000K')
```

Example 4: DB2 subsystems DB1G and DB2G are members of a data sharing group. Both were installed with a command prefix scope of STARTED. Start DB1G and DB2G by routing the appropriate commands to the MVS system on which they are to be started, MVS1 and MVS2.

```
ROUTE MVS1,-DB1G START DB2
ROUTE MVS2,-DB2G START DB2
```

-START DDF (DB2)

Syntax



Example

Example: Start the distributed data facility.

```
-START DDF
```

-START FUNCTION SPECIFIC

-START FUNCTION SPECIFIC (DB2)

The DB2 command START FUNCTION SPECIFIC activates an external function that is stopped. Built-in functions or user-defined functions that are sourced on another function cannot be started with this command.

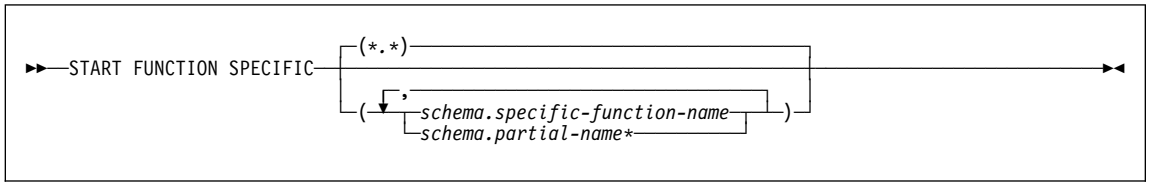
On successful completion of the command, queued requests for the specified functions begin executing. The abend counts for those functions are set to zero.

You do not need to issue the START FUNCTION SPECIFIC command when defining a new function to DB2. DB2 automatically activates the new function on the first SQL statement that invokes the new function.

Historical statistics in the DISPLAY FUNCTION SPECIFIC report (MAXQUE, TIMEOUT) are reset each time a START FUNCTION SPECIFIC command is issued for a given function.

Abbreviation: -STA FUNC SPEC

Syntax



Examples

Example 1: Start all functions.

```
-START FUNCTION SPECIFIC
```

Output similar to the following output is generated:

```
DSN9022I - DSNX9COM '-START FUNC' NORMAL COMPLETION
```

Example 2: Start functions USERFN1 and USERFN2. If any requests are queued for these functions, the functions are executed.

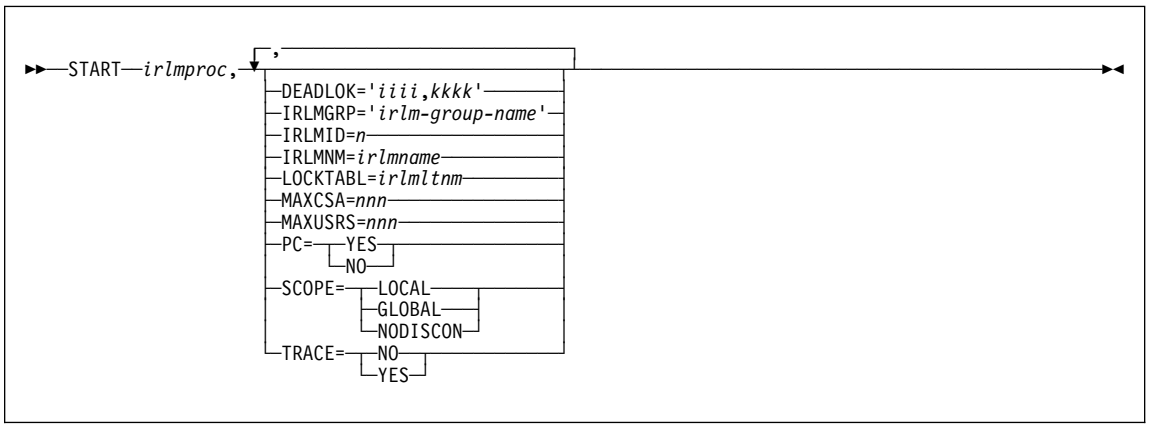
```
-START FUNCTION SPECIFIC(PAYROLL.USERFN1,PAYROLL.USERFN2)
```

Output similar to the following output is generated:

```
DSN9022I - DSNX9COM '-START FUNC' NORMAL COMPLETION
```

START irlmproc (MVS IRLM)

Syntax



Example

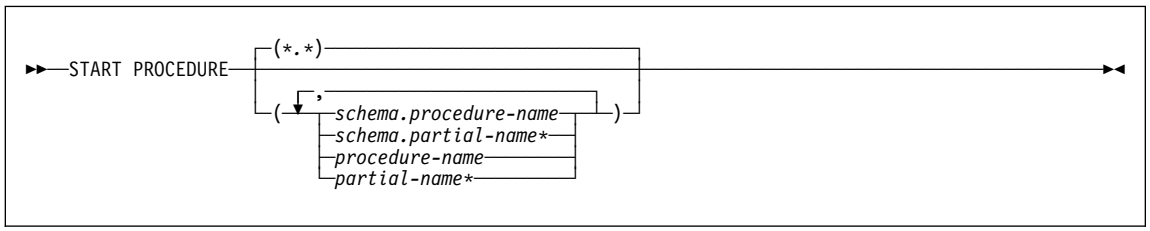
Example: Enter the following command on the MVS system console:

```
S irlmproc,MAXCSA=8
```

This command starts the IRLM with 8MB for MAXCSA which controls the CSA (including ECSA) usage for locks when PC=NO.

-START PROCEDURE (DB2)

Syntax



Examples

Example 1: Start all stored procedures.

```
-START PROCEDURE
```

This command produces output similar to the following output:

```
DSNX946I - DSNX9ST2 START PROCEDURE SUCCESSFUL FOR *.*
DSN9022I - DSNX9COM '-START PROC' NORMAL COMPLETION
```

Example 2: Make specific stored procedures available to be called, and start any requests waiting for those procedures.

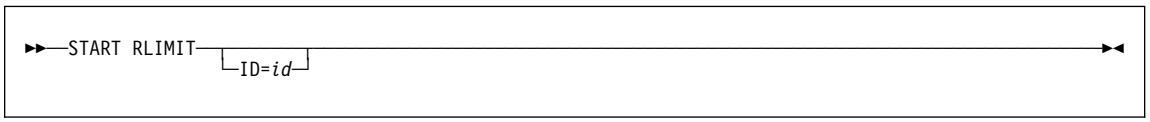
```
-START PROCEDURE(USERPRC1,USERPRC2)
```

This command produces output similar to the following output:

```
DSNX946I - DSNX9ST2 START PROCEDURE SUCCESSFUL FOR USERPRC1
DSNX946I - DSNX9ST2 START PROCEDURE SUCCESSFUL FOR USERPRC2
DSN9022I - DSNX9COM '-START PROC' NORMAL COMPLETION
```

-START RLIMIT (DB2)

Syntax



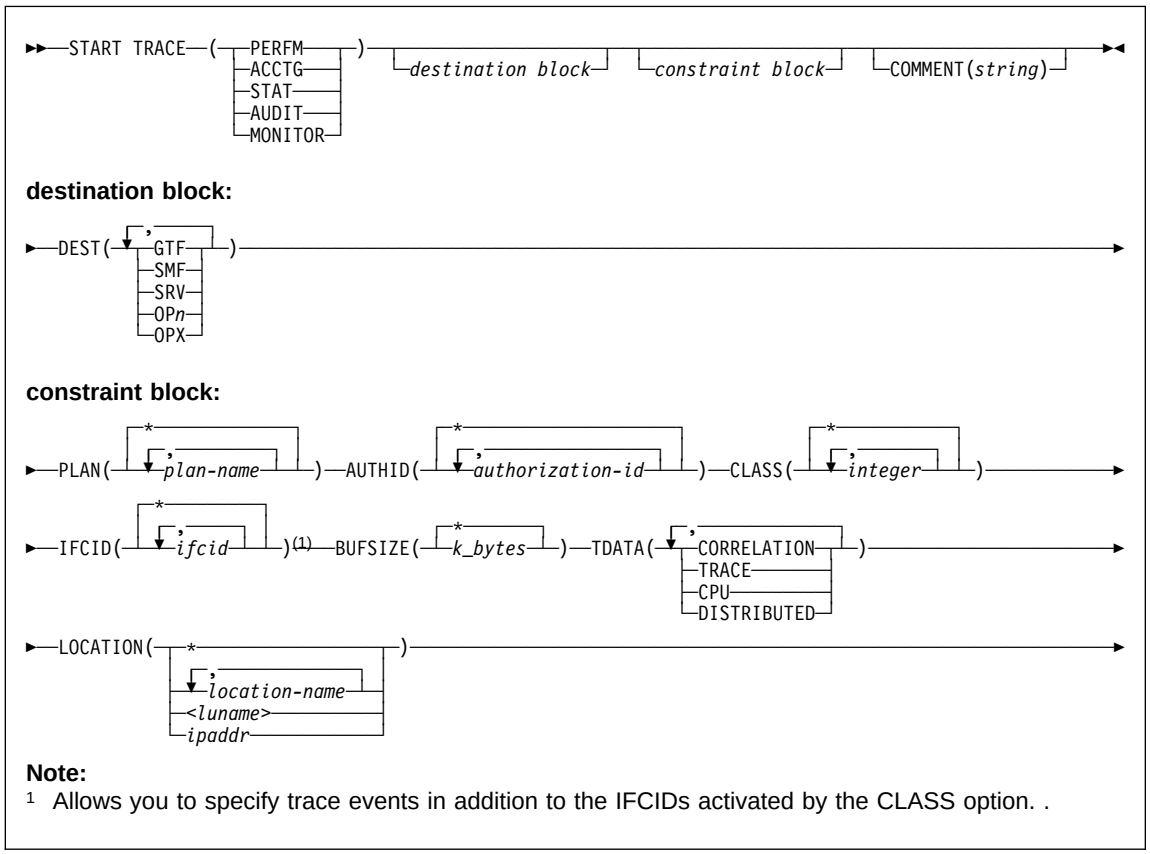
Example

Example: Start the resource limit facility.

```
-START RLIMIT ID=01
```

-START TRACE (DB2)

Syntax



Note:
1 Allows you to specify trace events in addition to the IFCIDs activated by the CLASS option. .

Examples

Example 1: Start a performance trace for threads with remote activity to location USIBMSTODB21. Only activate IFCIDs 44 (lock suspends) and 54 (lock contention). Trace class 30 is available for installation use.

```
-START TRACE (PERFM)
  DEST(GTF)
  LOCATION(USIBMSTODB21)
  CLASS(30)
  IFCID(44)
```

Example 2: Start an accounting trace for plan DSN8BC61. Write records to SMF (that will happen by default). Include a comment to identify the trace.

```
-START TRACE (ACCTG)
  PLAN (DSN8BC61)
  COMMENT ('ACCTG TRACE FOR DSN8BC61')
```

Example 3: Start the statistics trace. Write records to SMF (by default).

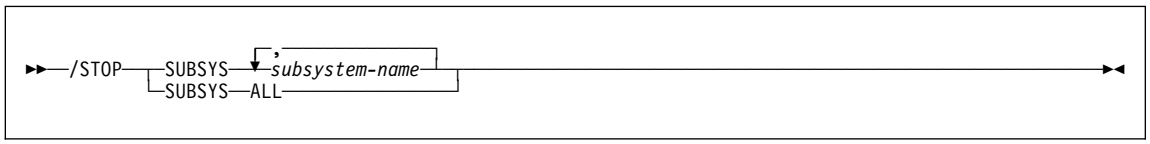
```
-START TRACE=S
```

Example 4: Start monitor tracing (usually done by an application program). Write records to OPX (by default).

```
-START TRACE(MON)
```

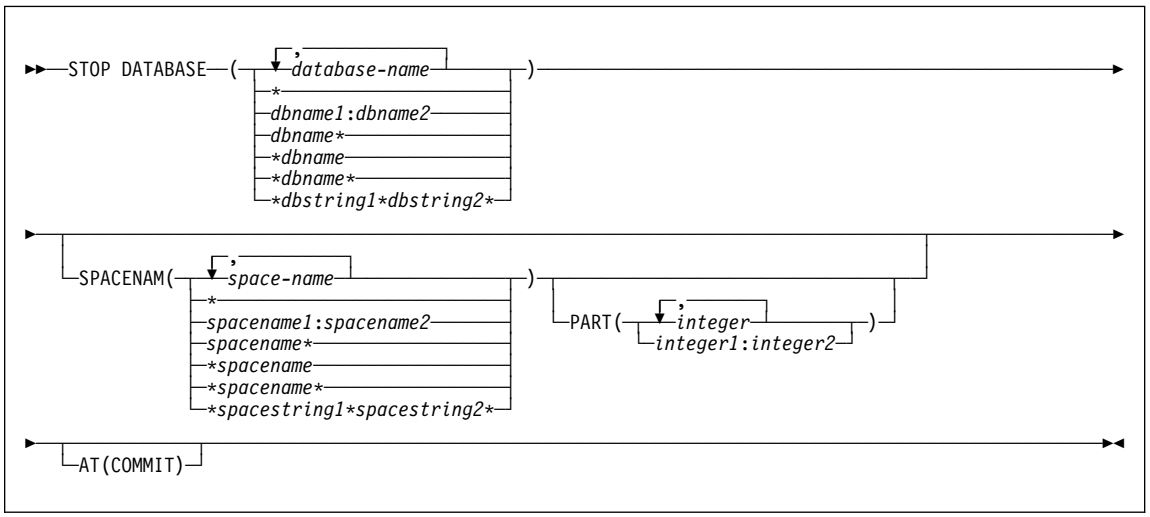
/STOP (IMS)

Syntax



-STOP DATABASE (DB2)

Syntax



Examples

Example 1: Stop table space DSN8S61E in database DSN8D61A and close the data sets that belong to that table space.

```
-STOP DATABASE(DSN8D61A) SPACENAM(DSN8S61E)
```

Example 2: Stop all databases (except DSNDB01, DSNDB06, and work file databases)

```
-STOP DATABASE(*)
```

Example 3: Stop all databases (except DSNDB01, DSNDB06, and work file databases) when all jobs release their claims and all utilities release their drain locks.

```
-STOP DATABASE(*) AT(COMMIT)
```

Example 4: Stop the first partition of XEMP2, a nonpartitioned index of a partitioned table space in database DSN8D61A. Partition 1 is logically stopped and cannot be accessed by applications; however, no data sets are closed because parts of a nonpartitioned index are not associated with separate physical data sets.

```
-STOP DATABASE(DSN8D61A) SPACENAM(XEMP2) PART(1)
```

Example 5: Stop all table spaces with names that begin with "T" and end with the "IQUA03" string in database DSN8D61A.

```
-STOP DATABASE(DSN8D61A) SPACENAM(T*IQUA03)
```

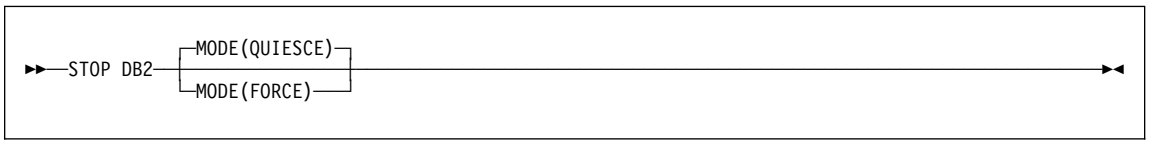
-STOP DATABASE (DB2)

Output similar to the following output indicates that the command completed successfully:

```
DSN9022I - DSNTDDIS 'STOP DATABASE' NORMAL COMPLETION
DSNT736I - ASYNCHRONOUS STOP DATABASE COMMAND HAS
COMPLETED FOR COMMAND: STOP DB(DSN8D61A) SPACE(T*IQUA03)
```

-STOP DB2 (DB2)

Syntax



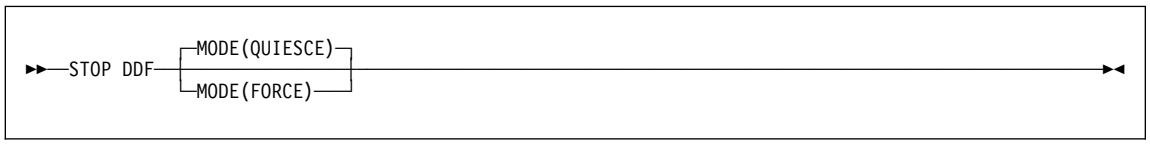
Example

Example: Stop the DB2 subsystem. Allow currently active programs to complete. Do not allow new programs to identify to DB2.

```
-STOP DB2 MODE (QUIESCE)
```

-STOP DDF (DB2)

Syntax



Examples

Example 1: Stop the distributed data facility (MODE QUIESCE).

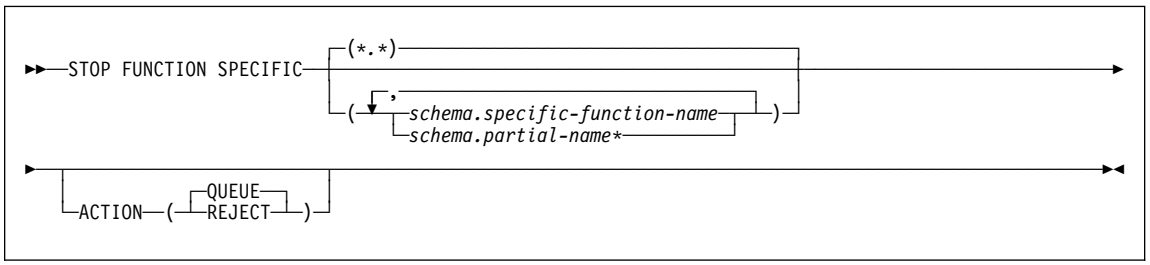
```
-STOP DDF
```

Example 2: Stop the distributed data facility (MODE FORCE).

```
-STOP DDF MODE(FORCE)
```

-STOP FUNCTION SPECIFIC

Syntax



Examples

Example 1: Stop access to all functions. While the -STOP FUNCTION SPECIFIC command is in effect, DB2 queues all attempts to execute functions.

```
-STOP FUNCTION SPECIFIC ACTION(QUEUE)
```

This command produces output similar to the following output:

```
DSN9022I - DSNX9COM '-STOP FUNC' NORMAL COMPLETION
```

Example 2: Stop access to all functions. While the -STOP FUNCTION SPECIFIC command is in effect, DB2 rejects attempts to execute functions.

```
-STOP FUNCTION SPECIFIC ACTION(REJECT)
```

This command produces output similar to the following output:

```
DSN9022I - DSNX9COM '-STOP FUNC' NORMAL COMPLETION
```

Example 3: Stop functions PAYROLL.USERFN1 and PAYROLL.USERFN3. While the -STOP FUNCTION SPECIFIC command is in effect, DB2 queues all attempts to execute functions.

```
-STOP FUNCTION SPECIFIC(PAYROLL.USERFN1,PAYROLL.USERFN3)
```

This command produces output similar to the following output:

```
DSN9022I - DSNX9COM '-STOP FUNC' NORMAL COMPLETION
```

Example 4: Stop functions PAYROLL.USERFN1 and PAYROLL.USERFN3. While the -STOP FUNCTION SPECIFIC command is in effect, DB2 rejects attempts to execute either of these functions.

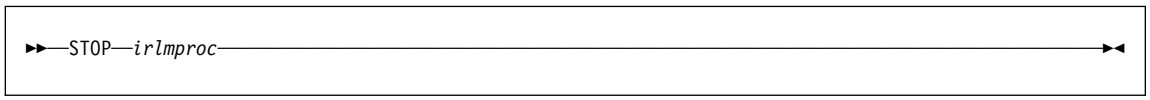
```
-STOP FUNCTION SPECIFIC(PAYROLL.USERFN1,PAYROLL.USERFN3) ACTION(REJECT)
```

This command produces output similar to the following output:

```
DSN9022I - DSNX9COM '-STOP FUNC' NORMAL COMPLETION
```

STOP irlmproc (MVS IRLM)

Syntax



Example

Example: Enter on the MVS1 system console:

```
P KRLM1
```

IRLM responses on MVS1 system console:

```
DXR165I IR21 TERMINATED VIA IRLM MODIFY COMMAND
DXR121I IR21 END-OF-TASK CLEANUP SUCCESSFUL - HI-CSA     325K
```

Response on MVS2 system console:

```
DXR025I JRLM SESSION LOST, SHARING STATE IS IRLM FAILED
```

Explanation:

The operator on system 1 has terminated the IRLM procedure named KRLM1. The operator on system 2 is informed that the IRLM in system 1 has terminated, but no operator action on system 2 is required.

Note when in a data sharing environment: You cannot issue the P command to IRLM in a data sharing group until there are no DB2s identified and the IRLM has issued the following messages:

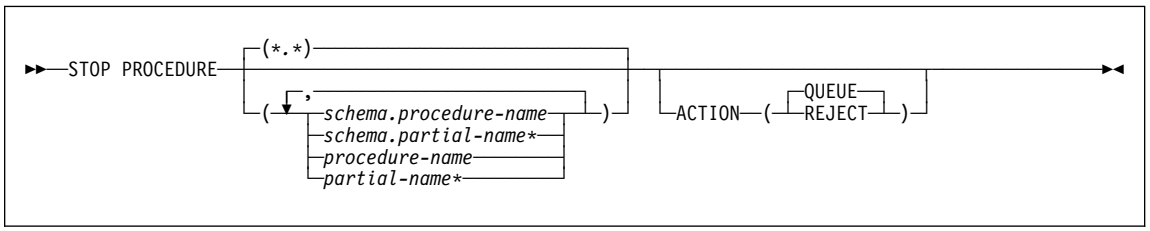
```
DXR136I IR21 HAS DISCONNECTED FROM THE DATA SHARING GROUP
```

Any members still active in the group issue:

```
DXR137I JR21 GROUP STATUS CHANGED. IR21 233 HAS BEEN DISCONNECTED
FROM THE DATA SHARING GROUP
```

-STOP PROCEDURE (DB2)

Syntax



Examples

Example 1: Stop access to all stored procedures, and terminate the DB2 stored procedures address space. While the -STOP PROCEDURE command is in effect, attempts to execute stored procedures are queued.

```
-STOP PROCEDURE ACTION(QUEUE)

DSNX947I - DSNX9SP2 STOP PROCEDURE SUCCESSFUL FOR *.*
DSN9022I - DSNX9COM '-STOP PROC' NORMAL COMPLETION
```

Example 2: Stop access to all stored procedures, and terminate the DB2 stored procedures address space. While the -STOP PROCEDURE command is in effect, attempts to execute stored procedures are rejected.

```
-STOP PROCEDURE ACTION(REJECT)

DSNX947I - DSNX9SP2 STOP PROCEDURE SUCCESSFUL FOR *.*
DSN9022I - DSNX9COM '-STOP PROC' NORMAL COMPLETION
```

Example 3: Stop stored procedures USERPRC1 and USERPRC3. While the -STOP PROCEDURE command is in effect, attempts to execute these stored procedure are queued.

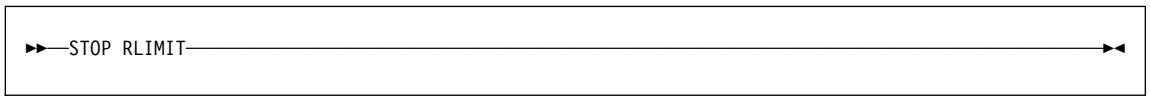
```
-STOP PROCEDURE(USERPRC1,USERPRC3)

DSNX947I - DSNX9SP2 STOP PROCEDURE SUCCESSFUL FOR USERPRC1
DSNX947I - DSNX9SP2 STOP PROCEDURE SUCCESSFUL FOR USERPRC3
DSN9022I - DSNX9COM '-STOP PROC' NORMAL COMPLETION
```

Example 4: Stop stored procedures USERPRC1 and USERPRC3. While the -STOP PROCEDURE command is in effect, attempts to execute these stored procedure are rejected.

```
-STOP PROCEDURE(USERPRC1,USERPRC3) ACTION(REJECT)

DSNX947I - DSNX9SP2 STOP PROCEDURE SUCCESSFUL FOR USERPRC1
DSNX947I - DSNX9SP2 STOP PROCEDURE SUCCESSFUL FOR USERPRC3
DSN9022I - DSNX9COM '-STOP PROC' NORMAL COMPLETION
```

-STOP RLIMIT (DB2)

Syntax



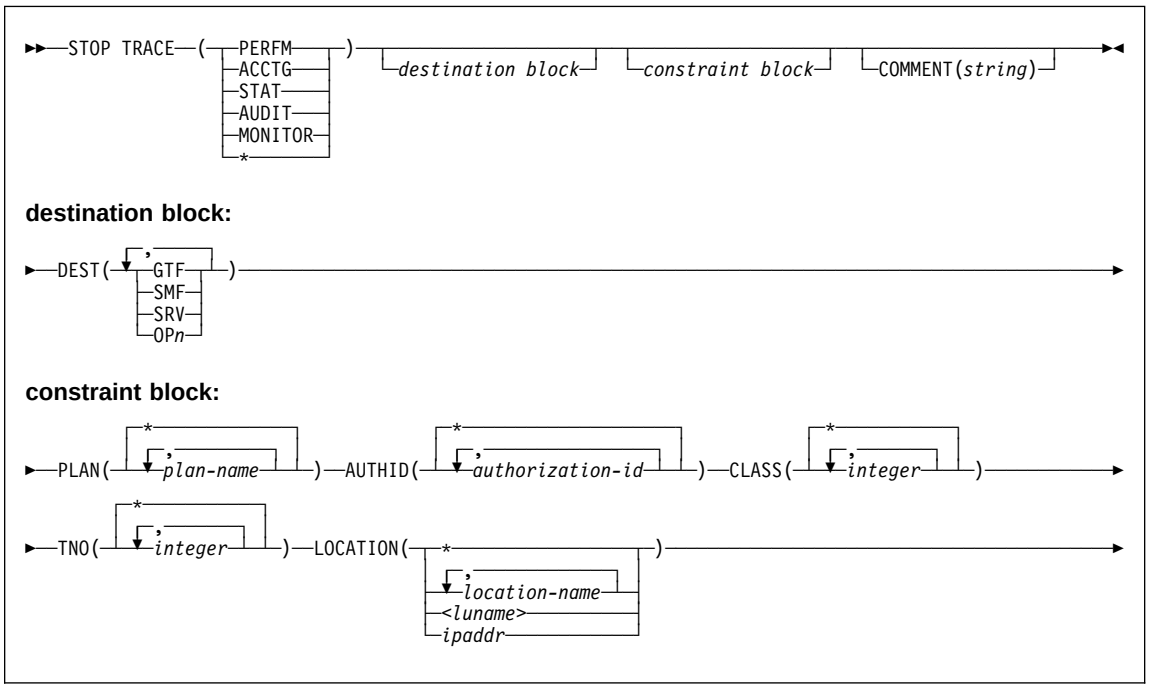
Example

Example: Stop the resource limit facility.

```
-STOP RLIMIT
```

-STOP TRACE (DB2)

Syntax



Examples

Example 1: Stop all traces that have the generalized trace facility as their only destination.

```
-STOP TRACE (*) DEST (GTF)
```

Example 2: Stop an accounting trace of all threads between the local and USIBMSTODB21 DB2 subsystems for plan DSN8BC61. Include a comment.

```
-STOP TRACE (ACCTG)
  PLAN (DSN8BC61)
  LOCATION (USIBMSTODB21)
  COMMENT('ACCTG TRACE FOR DSN8BC61')
```

Example 3: Stop trace number 4.

```
-STOP TRACE (P) TNO(4)
```

Example 4: Stop all active traces of any type for USIBMSTODB22.

```
-STOP TRACE (*) LOCATION (USIBMSTODB22)
```

-STOP TRACE (DB2)

Example 5: Stop all performance traces.

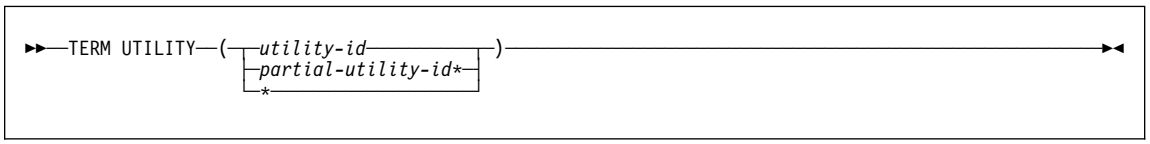
```
-STOP TRACE=P
```

Example 6: Stop all monitor tracing.

```
-STOP TRACE(MON)
```

-TERM UTILITY (DB2)

Syntax



Examples

Example 1: Terminate all utility jobs for which you are authorized.

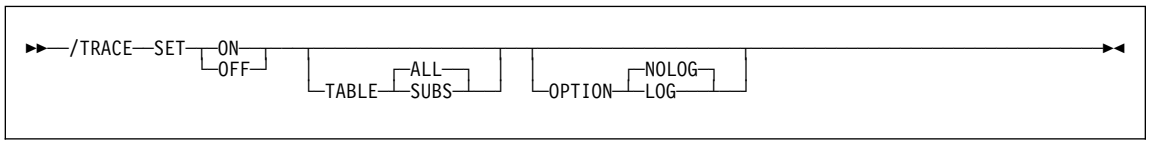
```
-TERM UTILITY (*)
```

Example 2: Terminate all utility jobs whose utility ID begins with SMITH.

```
-TERM UTILITY
  (SMITH*)
```

/TRACE (IMS)

Syntax



Examples

Example 1: This command starts IMS tracing and:

- Enables the DB2 trace
- Writes IMS trace tables to the IMS log before they wrap.

```
/TRACE SET ON TABLE SUBS OPTION LOG
```

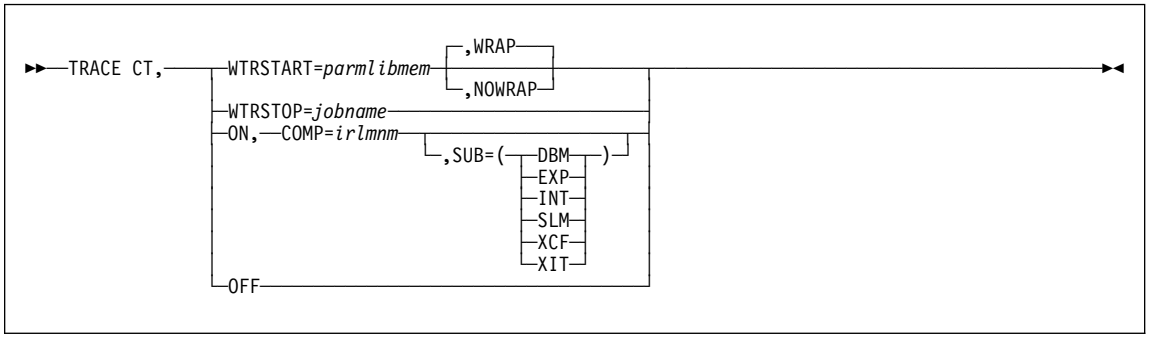
Example 2: This command starts IMS tracing and:

- Enables all trace tables (including DB2's); (ALL is the default parameter for the TABLE keyword)
- Writes IMS trace tables to the IMS log before they wrap.

```
/TRACE SET ON TABLE ALL OPTION LOG
```

TRACE CT (MVS IRLM)

Syntax



TRACE CT (MVS IRLM)

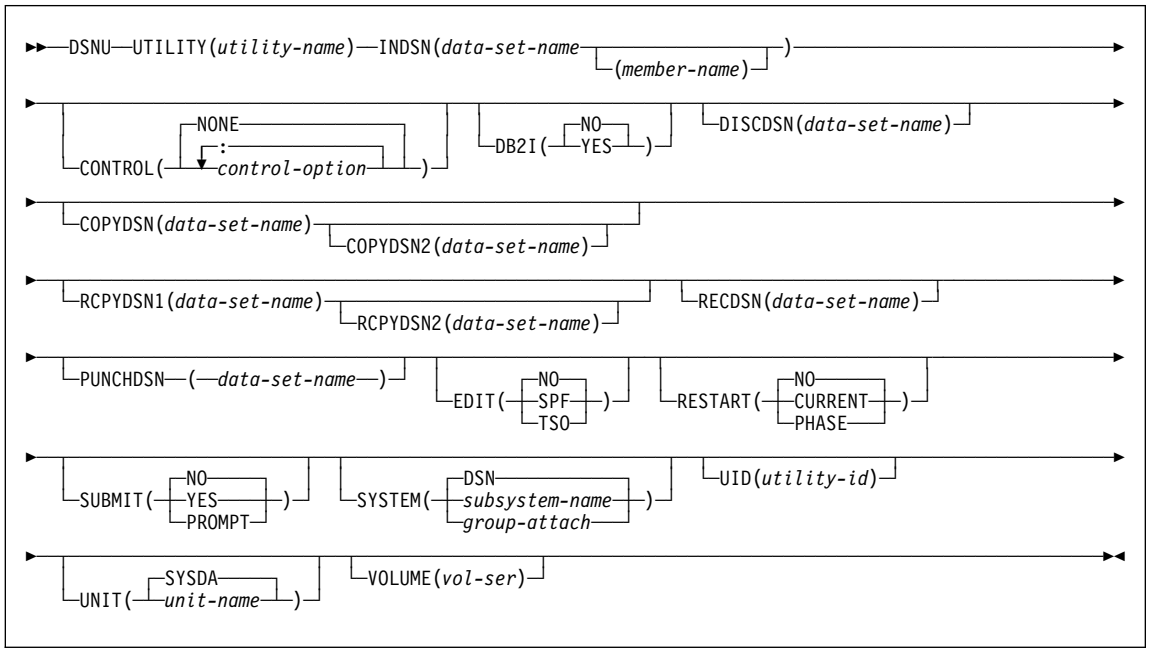
Online utility alphabetic reference

This section contains syntax diagrams and examples of DB2 online utilities organized alphabetically by name. For more information, see Section 2 of *DB2 Utility Guide and Reference*.

DSNU (TSO CLIST)

Using the DSNU CLIST command in TSO

Syntax



Examples

Example 1: The following CLIST command statement generates a data set called *authorization-id*.DSNURGT.CNTL that contains JCL statements that invoke the DSNUPROC procedure.

```
%DSNU UTILITY(REORG TABLESPACE) INDSN(MYREOR.DATA)
  RECDSN(MYREOR.WORK) RESTART(NO)
  EDIT(TSO) SUBMIT(YES)
```

The DSNUPROC procedure invokes the REORG TABLESPACE utility. The MYREOR.DATA data set is merged into the JCL data set as SYSIN input. MYREOR.WORK is a temporary data set required by REORG TABLESPACE. The TSO editor is invoked to allow editing of the JCL data set, *authorization-id*.DSNURGT.CNTL. The TSO editor then submits the JCL data set as a batch job and will not be modified by this CLIST command statement until a new request is made to execute the REORG TABLESPACE utility.

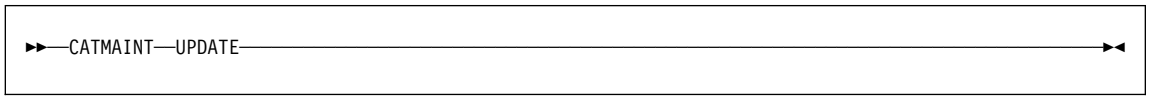
Example 2: The following example shows how to invoke the CLIST command for the COPY utility.

DSNU (TSO CLIST)

```
%DSNU
  UTILITY (COPY)
  INDSN ('MYCOPY(STATEMNT)')
  COPYDSN ('MYCOPIES.DSN8D61A.JAN1')
  EDIT (TSO)
  SUBMIT (YES)
  UID (TEMP)
  RESTART (NO)
```

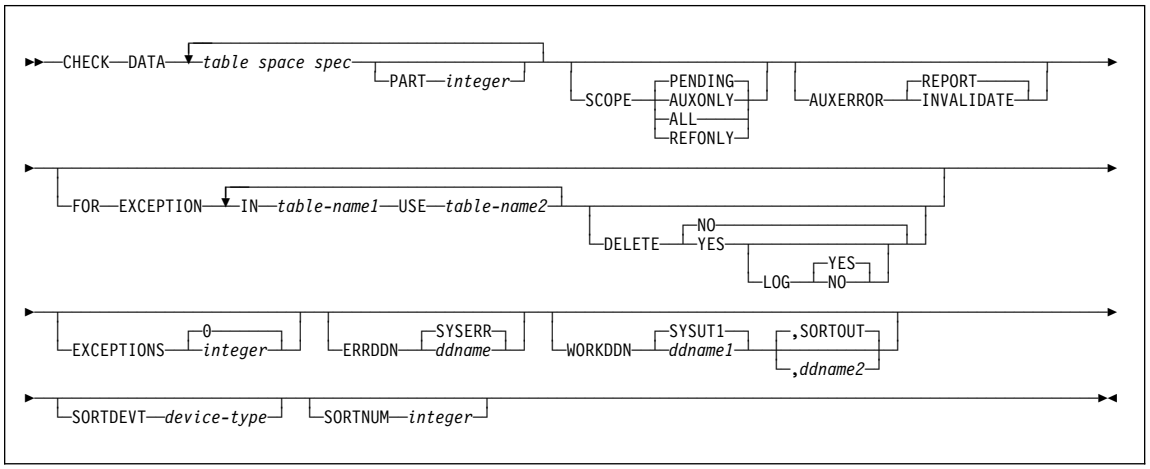
CATMAINT

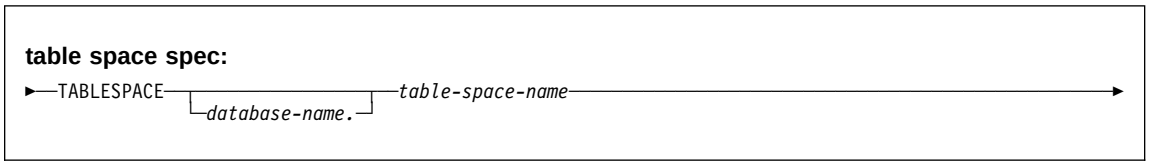
Syntax



CHECK DATA

Syntax





Examples

Example 1: CHECK DATA with DELETE. The following shows CHECK DATA JCL for checking and deleting.

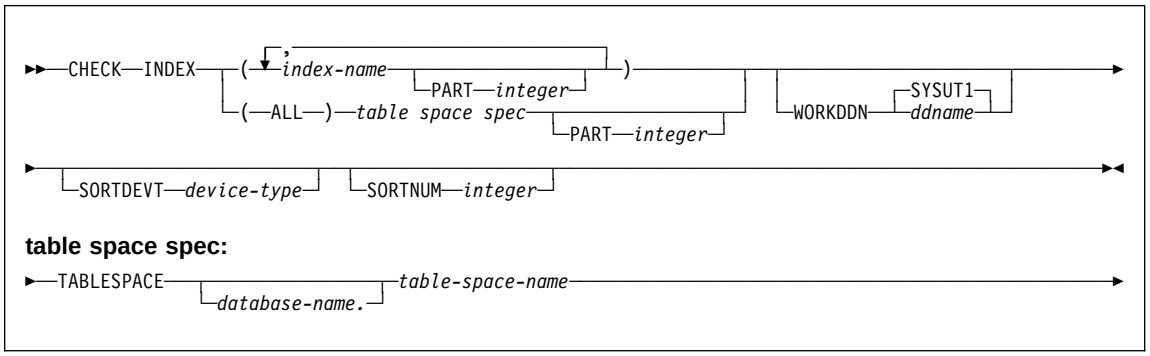
```
//STEP1    EXEC DSNUPROC,UID='IUIQU1UQ.CHK1',
//         UTPROC='',
//         SYSTEM='V61A'
//SYSUT1   DD DSN=IUIQU1UQ.CHK3.STEP1.SYSUT1,DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(8000,(200,20),,,ROUND)
//SYSERR   DD DSN=IUIQU1UQ.CHK3.SYSERR,DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(6000,(20,20),,,ROUND)
//SORTOUT  DD DSN=IUIQU1UQ.CHK3.STEP1.SORTOUT,DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(6000,(20,20),,,ROUND)
//SYSIN    DD *
CHECK DATA TABLESPACE DSN8D61A.DSN8S61D
           TABLESPACE DSN8D61A.DSN8S61E
     FOR EXCEPTION IN DSN8610.DEPT        USE DSN8610.EDEPT
                   IN DSN8610.EMP         USE DSN8610.EEMP
                   IN DSN8610.PROJ        USE DSN8610.EPROJ
                   IN DSN8610.PROJACT     USE DSN8610.EPROJACT
                   IN DSN8610.EMPPROJACT  USE DSN8610.EEPA
     DELETE YES
//*
```

Example 2: Control statement for deleting error rows. Check for and delete all constraint violations in table spaces DSN8D61A.DSN8S61D and DSN8D61A.DSN8S61E.

```
CHECK DATA TABLESPACE DSN8D61A.DSN8S61D
 TABLESPACE DSN8D61A.DSN8S61E
 FOR EXCEPTION IN DSN8610.DEPT USE DSN8610.EDEPT
               IN DSN8610.EMP USE DSN8610.EEMP
               IN DSN8610.PROJ USE DSN8610.EPROJ
               IN DSN8610.PROJECT USE DSN8610.EPROJECT
               IN DSN8610.EMPPROJECT USE DSN8610.EEMPPROJECT
 DELETE YES
```

CHECK INDEX

Syntax



Examples

Example 1: Check all indexes in a sample table space.

```
//STEP1   EXEC DSNUPROC,UID='IUIQU1UQ.CHK1',
//         UTPROC='',
//         SYSTEM='V61A'
//SYSUT1   DD DSN=IUIQU1UQ.CHK3.STEP1.SYSUT1,DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(8000,(200,20),,,ROUND)
//SYSERR   DD DSN=IUIQU1UQ.CHK3.SYSERR,DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(6000,(20,20),,,ROUND)
//SORTOUT  DD DSN=IUIQU1UQ.CHK3.STEP1.SORTOUT,DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(6000,(20,20),,,ROUND)
//SYSIN    DD *
CHECK INDEX (ALL) TABLESPACE DSN8D61A.DSN8S61E
//*
```

Example 2: Check one index. Check the project-number index (DSN8610.XPROJ1) on the sample project table.

```
CHECK INDEX (DSN8610.XPROJ1)
  SORTDEVT SYSDA
```

Example 3: Check more than one index. Check the indexes DSN8610.XEMPRAC1 and DSN8610.XEMPRAC2 on the employee to project activity sample table.

```
CHECK INDEX NAME (DSN8610.XEMPRAC1, DSN8610.XEMPRAC2)
```

Example 4: Check all indexes on a table space. Check all indexes on the employee-table table space (DSN8S61E).

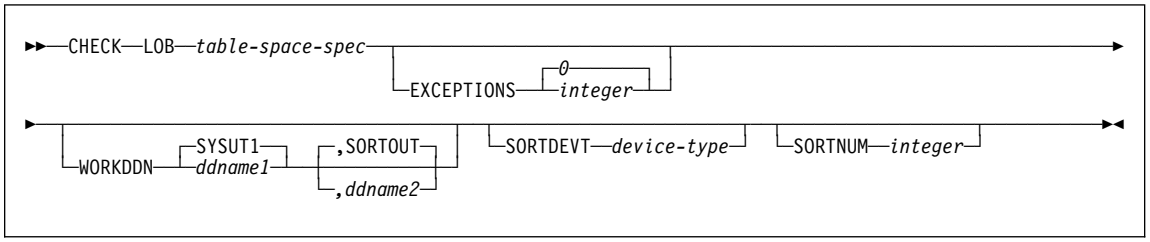
CHECK INDEX

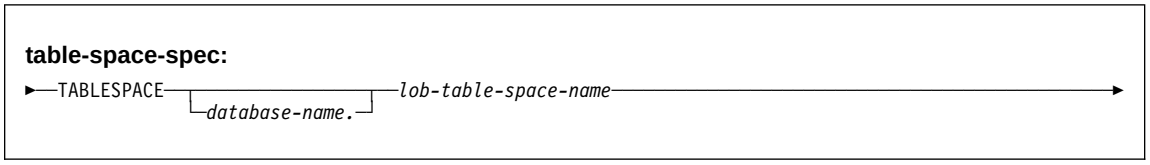
```
CHECK INDEX (ALL) TABLESPACE DSN8S61E
  SORTDEVT 3380
```

CHECK LOB

- STATS privilege for the database
- DBADM, DBCTRL, or DBMAINT authority for the database
- SYSCTRL or SYSADM authority

Syntax





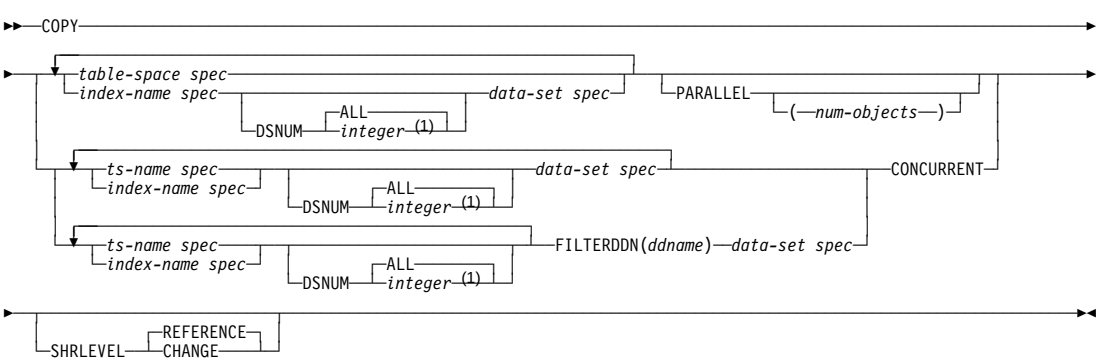
Examples

Example: Checking a LOB table space. Check the table space TLIQUG02 in database DBIQUG01 for structural defects or invalid LOB values.

```
//STEP1    EXEC DSNUPROC,UID='IUIQU2UG.CHECKL',
//         UTPROC='',
//         SYSTEM='V61A'
//SYSERR   DD DSN=IUIQU2UG.STEP1.SYSERR,DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSUT1   DD DSN=IUIQU2UG.CHECKL.SYSUT1,UNIT=SYSDA,
//         SPACE=(4000,(20,20),,,ROUND),
//         DISP=(MOD,DELETE,CATLG)
//SYSREC   DD DSN=CUST.FM.CSFT320.DATA,DISP=SHR,UNIT=SYSDA,
//         VOL=SER=123456
//SORTOUT  DD DSN=IUIQU2UG.STEP1.SORTOUT,DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSIN    DD *
CHECK LOB TABLESPACE DBIQUG01.TLIQUG02
      EXCEPTIONS 3 WORKDDN SYSUT1,SORTOUT SORTDEVT SYSDA
      SORTNUM 4
```

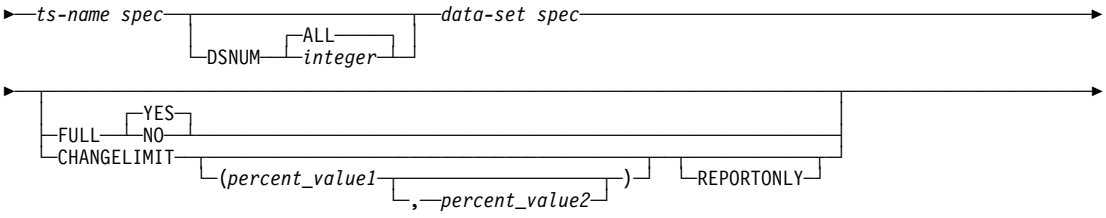
COPY

Syntax

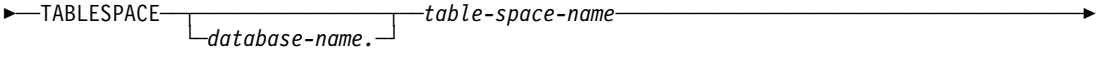


Note:
1 Not valid for nonpartitioning indexes.

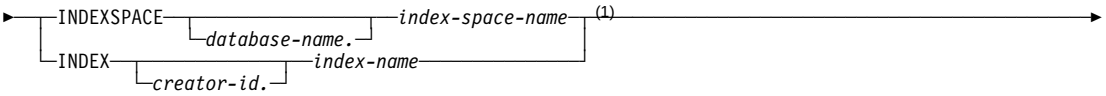
table-space spec:



ts-name spec:

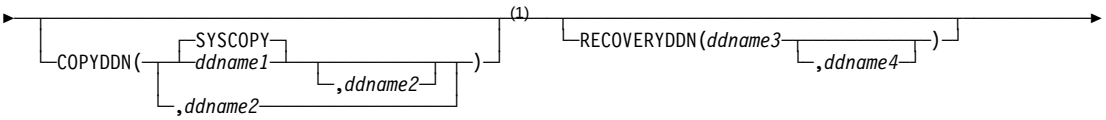


index-name spec:



Note:
1 INDEXSPACE is the preferred specification.

data-set spec:



Note:
1 If you specified a list of objects, only one object in the list can use the default (SYSCOPY); you must specify the DD names for the rest of the objects listed.

Examples

Example 1: Full image copy. Make a full image copy of table space DSN8S61E in database DSN8D61A.

```
//STEP1    EXEC DSNUPROC,UID='IUJMU111.COPYTS',
//         UTPROC='',
//         SYSTEM='V61A',DB2LEV=DB2A
//SYSIN    DD *
//SYSCOPY  DD DSN=COPY001F.IFDY01,UNIT=SYSDA,VOL=SER=CPY01I,
//         SPACE=(CYL,(15,1)),DISP=(NEW,CATLG,CATLG)
//SYSIN DD *
COPY TABLESPACE DSN8D61A.DSN8S61E
```

Example 2: Full image copies of a list of objects. Make full image copies at the local site and recovery site, along with backups, of the following objects:

- table space DSN8D61A.DSN8S61D, and its indexes:
  - DSN8610.XDEPT1
  - DSN8610.XDEPT2
  - DSN8610.XDEPT3
- table space DSN8D61A.DSN8S61E, and its indexes

Do not allow updates during the copy, and process four objects in parallel. As the copy of an object completes, the next object in the list begins processing in parallel until all the objects have been processed.

This COPY job creates a point of consistency for the table spaces and their indexes. You can subsequently use the RECOVER utility with the TOCOPY option to recover all of these objects; see page 360 for an example.

```
COPY
   TABLESPACE DSN8D61A.DSN8S61D
        COPYDDN (COPY1,COPY2)
        RECOVERYDDN (COPY3,COPY4)
  INDEX DSN8610.XDEPT1
        COPYDDN (COPY5,COPY6)
        RECOVERYDDN (COPY7,COPY8)
  INDEX DSN8610.XDEPT2
        COPYDDN (COPY9,COPY10)
        RECOVERYDDN (COPY11,COPY12)
  INDEX DSN8610.XDEPT3
        COPYDDN (COPY13,COPY14)
        RECOVERYDDN (COPY15,COPY16)
  TABLESPACE DSN8D61A.DSN8S61E
        COPYDDN (COPY17,COPY18)
        RECOVERYDDN (COPY19,COPY20)
  INDEX DSN8610.XEMP1
        COPYDDN (COPY21,COPY22)
        RECOVERYDDN (COPY23,COPY24)
  INDEX DSN8610.XEMP2
        COPYDDN (COPY25,COPY26)
        RECOVERYDDN (COPY27,COPY28)
 PARALLEL(4)
 SHRLEVEL REFERENCE
```

Example 3: Copies for local site and recovery site. Make full image copies of table space DSN8S61C in database DSN8D61P at the local site and the recovery site.

```
//STEP1    EXEC DSNUPROC,UID='IUJMU111.COPYLST',
//         UTPROC='',
//         SYSTEM='V61A',DB2LEV=DB2A
//COPY1    DD DSN=IUJMU111.COPYLST.STEP1.COPY1,
//         DISP=(MOD,CATLG,CATLG),UNIT=SYSDA,
//         SPACE=(4000,(20,20),,,ROUND)
//COPY2    DD DSN=IUJMU111.COPYLST.STEP1.COPY2,
//         DISP=(MOD,CATLG,CATLG),UNIT=SYSDA,
//         SPACE=(2000,(20,20),,,ROUND)
//COPY3    DD DSN=IUJMU111.COPYLST.STEP1.COPY3,
//         DISP=(MOD,CATLG,CATLG),UNIT=SYSDA,
//         SPACE=(2000,(20,20),,,ROUND)
//COPY4    DD DSN=IUJMU111.COPYLST.STEP1.COPY4,
//         DISP=(MOD,CATLG,CATLG),UNIT=SYSDA,
//         SPACE=(2000,(20,20),,,ROUND)
//SYSIN    DD *
 COPY TABLESPACE DSN8D61P.DSN8S61C
       COPYDDN     (COPY1,COPY2)
       RECOVERYDDN (COPY3,COPY4)
```

Example 4: Incremental copy with updates allowed. Make incremental image copies of table space DSN8S61D in database DSN8D61A, allowing update activity to occur during the copy process.

```
//STEP1    EXEC DSNUPROC,UID='IUJMU111.COPYLSTI',
//         UTPROC='',
//         SYSTEM='V61A',DB2LEV=DB2A
//SYSCOPY  DD DSN=IUJMU111.COPYLSTI.STEP1.CPY01I,
//         DISP=(MOD,CATLG,CATLG),UNIT=SYSDA,
//         SPACE=(4000,(20,20),,,ROUND)
//COPY2    DD DSN=IUJMU111.COPYLSTI.STEP1.CPY02I,
//         DISP=(MOD,CATLG,CATLG),UNIT=SYSDA,
//         SPACE=(2000,(20,20),,,ROUND)
//COPY3    DD DSN=IUJMU111.COPYLSTI.STEP1.CPY03I,
//         DISP=(MOD,CATLG,CATLG),UNIT=SYSDA,
//         SPACE=(2000,(20,20),,,ROUND)
//COPY4    DD DSN=IUJMU111.COPYLSTI.STEP1.CPY04I,
//         DISP=(MOD,CATLG,CATLG),UNIT=SYSDA,
//         SPACE=(2000,(20,20),,,ROUND)
//SYSIN    DD *
COPY TABLESPACE DSN8D61A.DSN8S61D
     COPYDDN      (SYSCOPY,COPY2)
     RECOVERYDDN  (COPY3,COPY4)
     FULL NO
     SHRLEVEL CHANGE
```

Example 5: Invoking DFSMS concurrent copy with the COPY utility. Copy a table space, using the CONCURRENT option to execute DFSMS™ concurrent copy. Use a DSSPRINT DD card for message output.

```
//COPY     EXEC DSNUPROC,SYSTEM=V61A
//SYSCOPY1 DD DSN=COPY1,DISP=(NEW,CATLG,CATLG),
//         SPACE=(4000,(20,20),,,ROUND),UNIT=SYSDA,VOL=SER=DB2CC5
//SYSPRINT DD DSN=COPY1.PRINT1,DISP=(NEW,CATLG,CATLG),
//         SPACE=(4000,(20,20),,,ROUND),UNIT=SYSDA,VOL=SER=DB2CC5
//DSSPRINT DD DSN=COPY1.PRINT2,DISP=(NEW,CATLG,CATLG),
//         SPACE=(4000,(20,20),,,ROUND),UNIT=SYSDA,VOL=SER=DB2CC5
//SYSIN    DD *
COPY TABLESPACE DBASE1AA.TABLESPC
  COPYDDN     (SYSCOPY1)
  CONCURRENT
```

Example 6: Invoking DFSMS concurrent copy with the COPY utility using FILTER. Copy a list of table spaces, using the CONCURRENT and FILTERDDN options to create a single "DUMP" statement for DFSMS concurrent copy, allowing maximum availability.

```
//SYSCOPY  DD DSN=CONCOPY.WFILT,DISP=(MOD,CATLG,DELETE),
//            UNIT=SYSDA,SPACE=(CYL,(42,5),RLSE)
//FILT     DD DSN=FILT.TEST1,DISP=(MOD,CATLG,DELETE),
//            UNIT=SYSDA,SPACE=(CYL,(1,1),RLSE)
//SYSIN    DD *
COPY TABLESPACE TS1
     TABLESPACE TS2
     TABLESPACE TS3
FILTERDDN(FILT)
COPYDDN(SYSCOPY)
CONCURRENT
    SHRLEVEL REFERENCE
```

Example 7: Invoking DFSMS concurrent copy with a list. Copy a list of table spaces, using the CONCURRENT option to execute DFSMS concurrent copy. Allow update activity during the COPY operation.

```
//STEP1    EXEC DSNUPROC,UID='IUJMU111.COPYLST',
//         UTPROC='',
//         SYSTEM='V61A',DB2LEV=DB2A
//COPY1    DD DSN=IUJMU111.COPYLST.STEP1.TS1,
//         DISP=(MOD,CATLG,CATLG),UNIT=SYSDA,
//         SPACE=(4000,(20,20),,,ROUND)
//COPY2    DD DSN=IUJMU111.COPYLST.STEP1.TS2,
//         DISP=(MOD,CATLG,CATLG),UNIT=SYSDA,
//         SPACE=(2000,(20,20),,,ROUND)
//COPY3    DD DSN=IUJMU111.COPYLST.STEP1.TS3,
//         DISP=(MOD,CATLG,CATLG),UNIT=SYSDA,
//         SPACE=(2000,(20,20),,,ROUND)
//SYSIN    DD *
COPY TABLESPACE DBAU2901.TPAU2901
     COPYDDN(COPY1)
     TABLESPACE DBAU2901.TLAU2902
     COPYDDN(COPY2)
     TABLESPACE DBAU2901.TSAU2903
     COPYDDN(COPY3)
     CONCURRENT SHRLEVEL CHANGE
```

Example 8: Report image copy information for a table space. Recommend a full image copy if the percent of changed pages is equal to or greater than 40 percent. Recommend an incremental image copy if the percent of changed pages is greater than 10 and less than 40 percent. Recommend no image copy if the percent of changed pages is 10 percent or less.

```
COPY TABLESPACE DSN8D61P.DSN8S61C CHANGELIMIT(10,40) REPORTONLY
```

Example 9: Make a conditional image copy. Take a full image copy of a table space if the number of changed pages is equal to or greater than 5 percent. Take an incremental image copy if the percent of changed pages is greater than 0 and less than 5 percent. If no pages have changed, do not take an image copy.

```
COPY TABLESPACE DSN8D61P.DSN8S61C CHANGELIMIT(5)
```

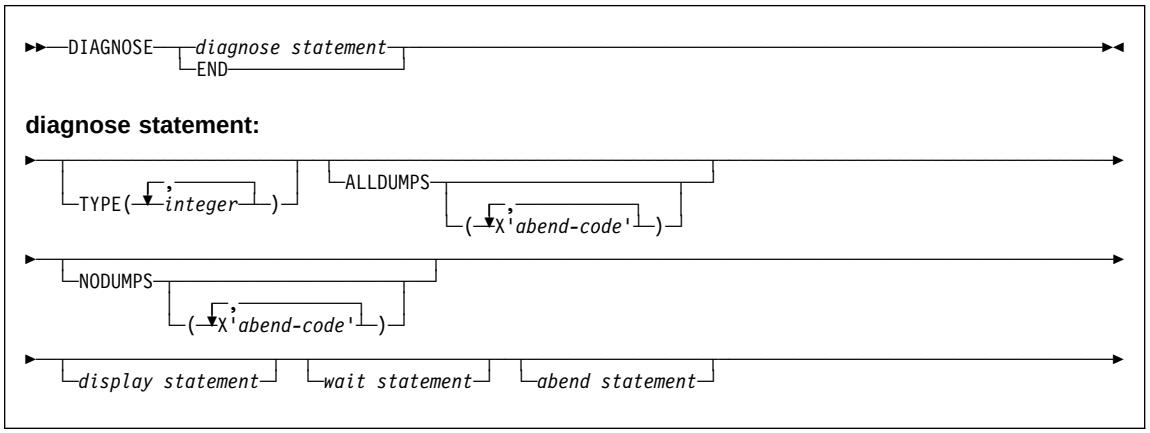
Example 10: Copying LOB table spaces together with related objects. Take a full image copy of base table space TPIQUD01 and LOB table spaces TLIQUDA1, TLIQUDA2, TLIQUDA3, and TLIQUDA4 in database DBIQUD01 if the number of changed pages is equal to or greater than the decimal percentage values specified for each object. Take an incremental image copy if the percent of changed pages falls in the range between the specified decimal percentage values. If no pages have changed, do not take an image copy. Also take full image copies of index spaces IPIQUD01, IXIQUD02, IUIQUD03, IXIQUDA1, IXIQUDA2, IXIQUDA3, and IXIQUDA4.

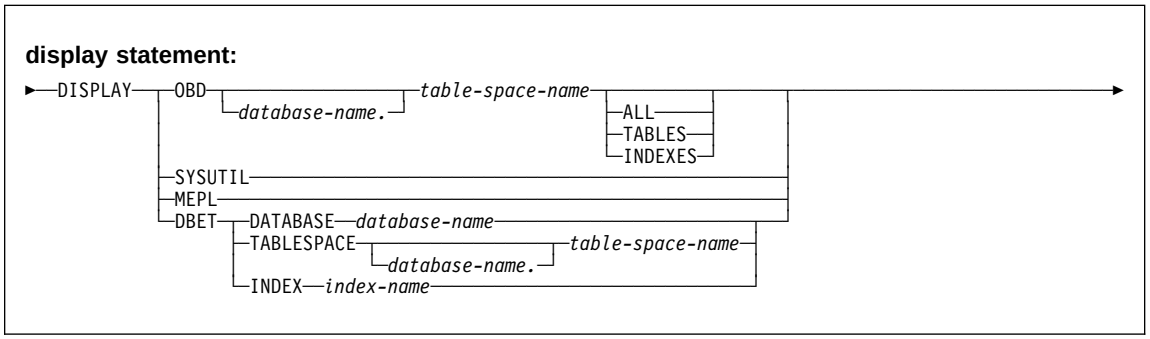
COPY

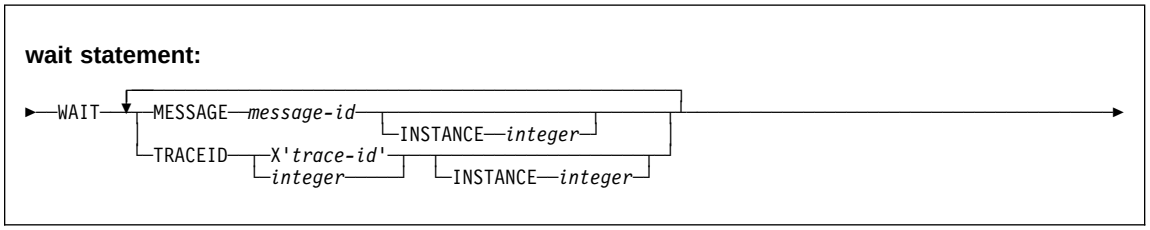
```
COPY
     TABLESPACE DBIQUD01.TPIQUD01 DSNUM ALL CHANGELIMIT(3.3,6.7)
       COPYDDN(COPYTB1)
     TABLESPACE DBIQUD01.TLIQUDA1 DSNUM ALL CHANGELIMIT(7.9,25.3)
       COPYDDN(COPYTA1)
     TABLESPACE DBIQUD01.TLIQUDA2 DSNUM ALL CHANGELIMIT(2.2,4.3)
       COPYDDN(COPYTA2)
     TABLESPACE DBIQUD01.TLIQUDA3 DSNUM ALL CHANGELIMIT(1.2,9.3)
       COPYDDN(COPYTA3)
     TABLESPACE DBIQUD01.TLIQUDA4 DSNUM ALL CHANGELIMIT(2.2,4.0)
       COPYDDN(COPYTA4)
     INDEXSPACE DBIQUD01.IPIQUD01 DSNUM ALL
       COPYDDN(COPYIX1)
     INDEXSPACE DBIQUD01.IXIQUD02 DSNUM ALL
       COPYDDN(COPYIX2)
     INDEXSPACE DBIQUD01.IUIQUD03 DSNUM ALL
       COPYDDN(COPYIX3)
     INDEXSPACE DBIQUD01.IXIQUDA1 DSNUM ALL
       COPYDDN(COPYIXA1)
     INDEXSPACE DBIQUD01.IXIQUDA2 DSNUM ALL
       COPYDDN(COPYIXA2)
     INDEXSPACE DBIQUD01.IXIQUDA3 DSNUM ALL
       COPYDDN(COPYIXA3)
     INDEXSPACE DBIQUD01.IXIQUDA4 DSNUM ALL
       COPYDDN(COPYIXA4)
 SHRLEVEL REFERENCE
```

DIAGNOSE

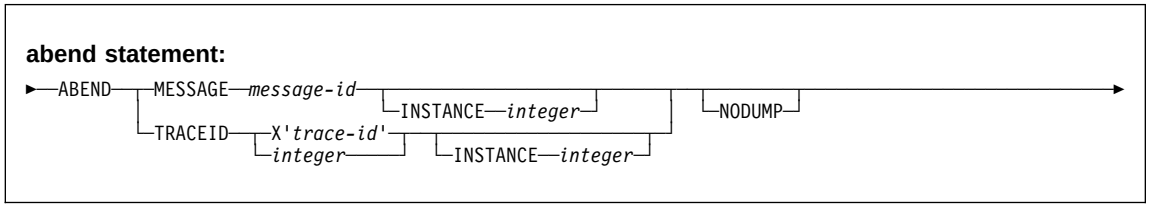
Syntax







DIAGNOSE



Examples

Example 1: Sample JCL for DIAGNOSE.

```
//STEP1    EXEC DSNUPROC,UID='IUJMU116.COPY1',
//         UTPROC='',
//         SYSTEM='V61A'
//SYSCOPY1  DD DSN=IUJMU116.COPY.STEP1.SYSCOPY1,DISP=(NEW,CATLG,CATLG),
//         UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSIN    DD *
DIAGNOSE  ABEND MESSAGE U400
                INSTANCE 1
                NODUMP
                   COPY TABLESPACE DSN8D61A.DSN8S61E
                     COPYDDN SYSCOPY1
DIAGNOSE
           END
//*
```

Example 2: Force dump for utility abend. Force a dump for any utility abend that occurs during the execution of the COPY utility.

```
DIAGNOSE
 ALLDUMPS
 COPY TABLESPACE DSNDB06.SYSDBASE
DIAGNOSE END
```

Example 3: Force utility abend if message is issued. Abend the LOAD utility the fifth time message DSNU311 is issued. Do not generate a dump.

```
DIAGNOSE
 ABEND MESSAGE U311 INSTANCE 5 NODUMP
LOAD DATA RESUME NO
  INTO TABLE TABLE1
   (NAME POSITION(1) CHAR(20))
DIAGNOSE END
```

Example 4: Display SYSUTIL table. Display all rows in the SYSUTIL table, and the DB2 and utility MEPLs.

```
DIAGNOSE
 DISPLAY SYSUTIL
DIAGNOSE
 DISPLAY MEPL
```

Example 5: Abend LOAD utility if message is issued. Abend the LOAD utility when unique index key violations occur.

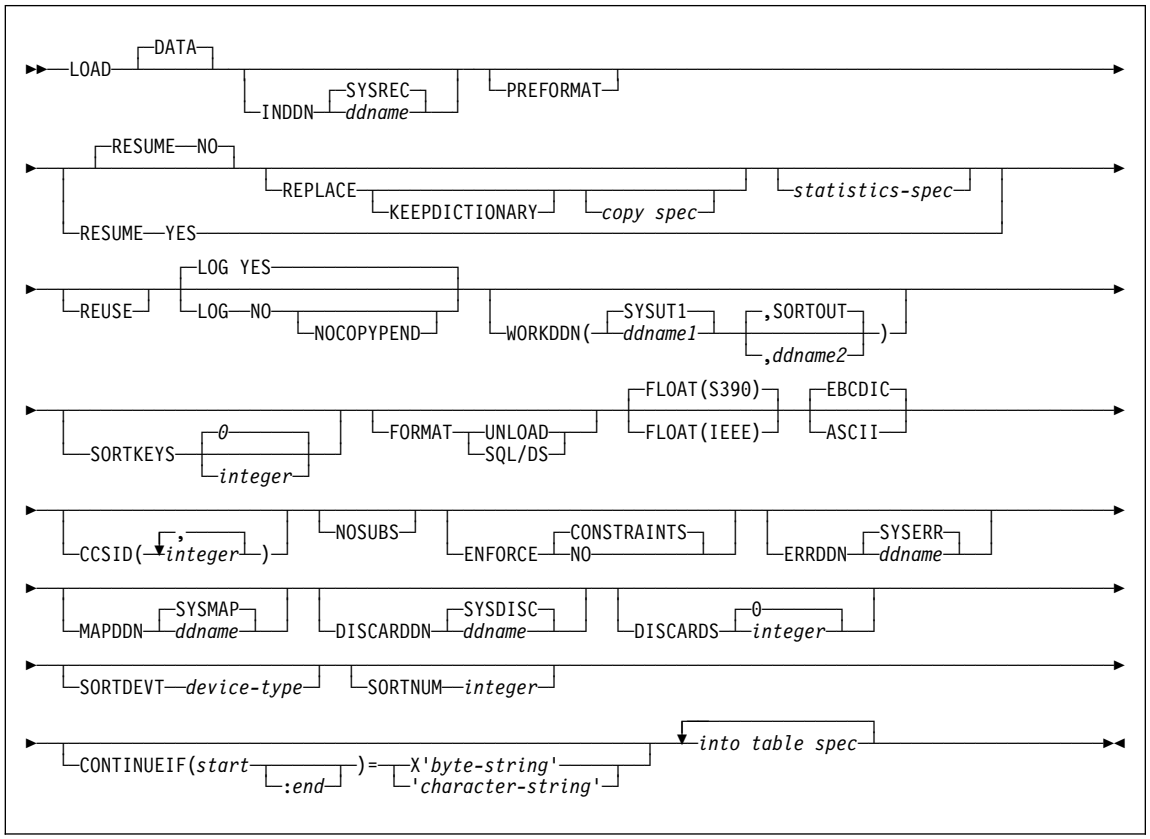
```
DIAGNOSE
 ABEND MESSAGE U344
```

Example 6: Force dump if abend with specified reason code. Cause a dump to be taken if an abend occurs with either of the specified reason codes.

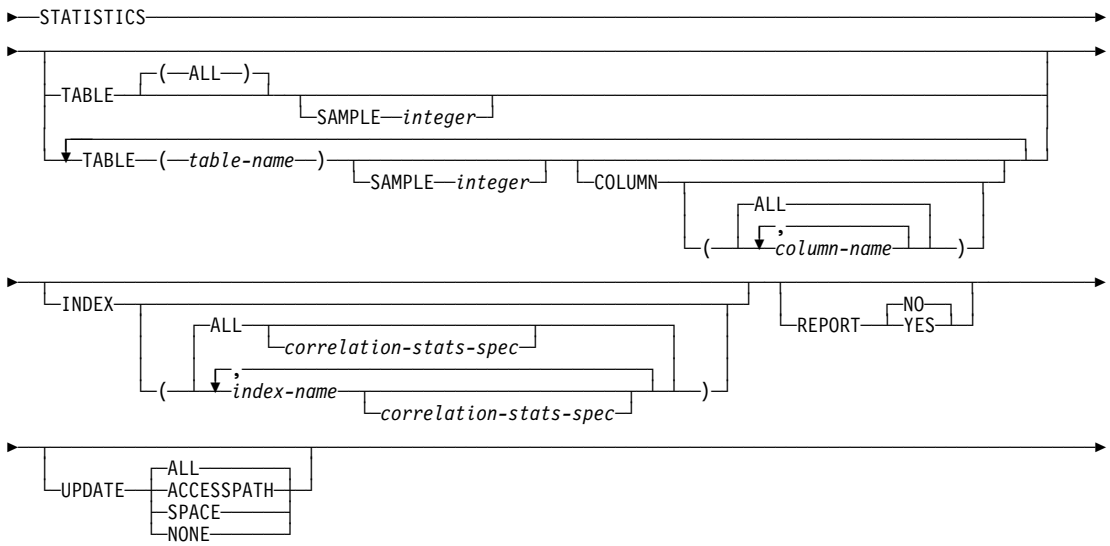
```
DIAGNOSE
 ALLDUMPS(X'00E40322',X'00E40323')
```

LOAD

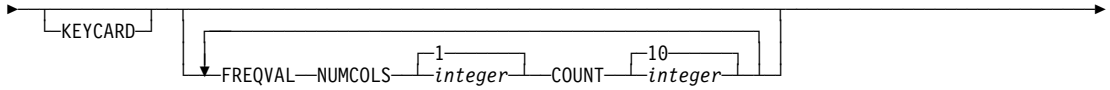
Syntax


LOAD
DATA
INDDN
SYSREC
ddname
PREFORMAT
RESUME
NO
YES
REPLACE
KEEPDICTIONARY
copy spec
statistics-spec
REUSE
LOG
YES
NO
NOCOPYPEND
WORKDDN(
SYSUT1
ddname1
,SORTOUT
,ddname2
)
SORTKEYS
0
integer
FORMAT
UNLOAD
SQL/DS
FLOAT(S390)
FLOAT(IEEE)
EBCDIC
ASCII
CCSID(
integer
,
)
NOSUBS
ENFORCE
CONSTRAINTS
NO
ERRDDN
SYSERR
ddname
MAPDDN
SYSMAP
DISCARDDN
SYSDISC
DISCARDS
0
SORTDEVT
device-type
SORTNUM
integer
CONTINUEIF(start
:end
)=
X'byte-string'
'character-string'
into table spec

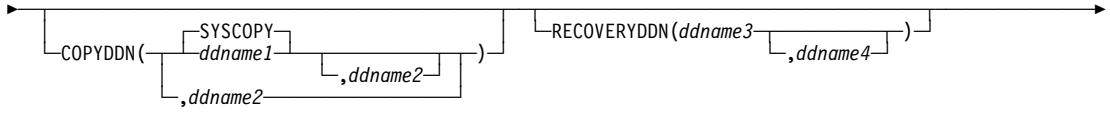
statistics-spec:



correlation-stats-spec:



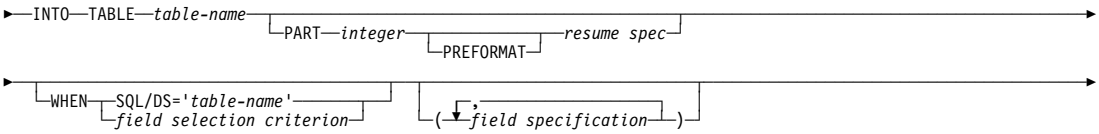
copy spec:



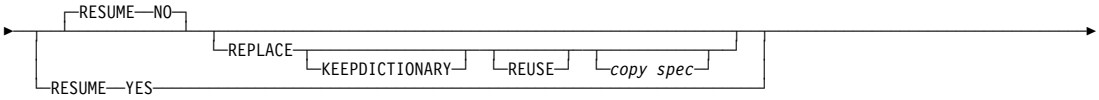
LOAD

INTO TABLE spec

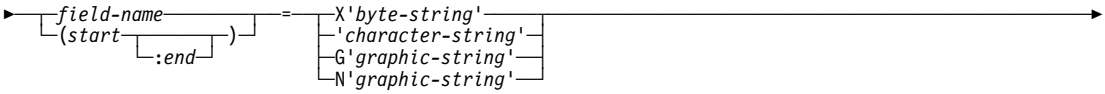
INTO TABLE spec:



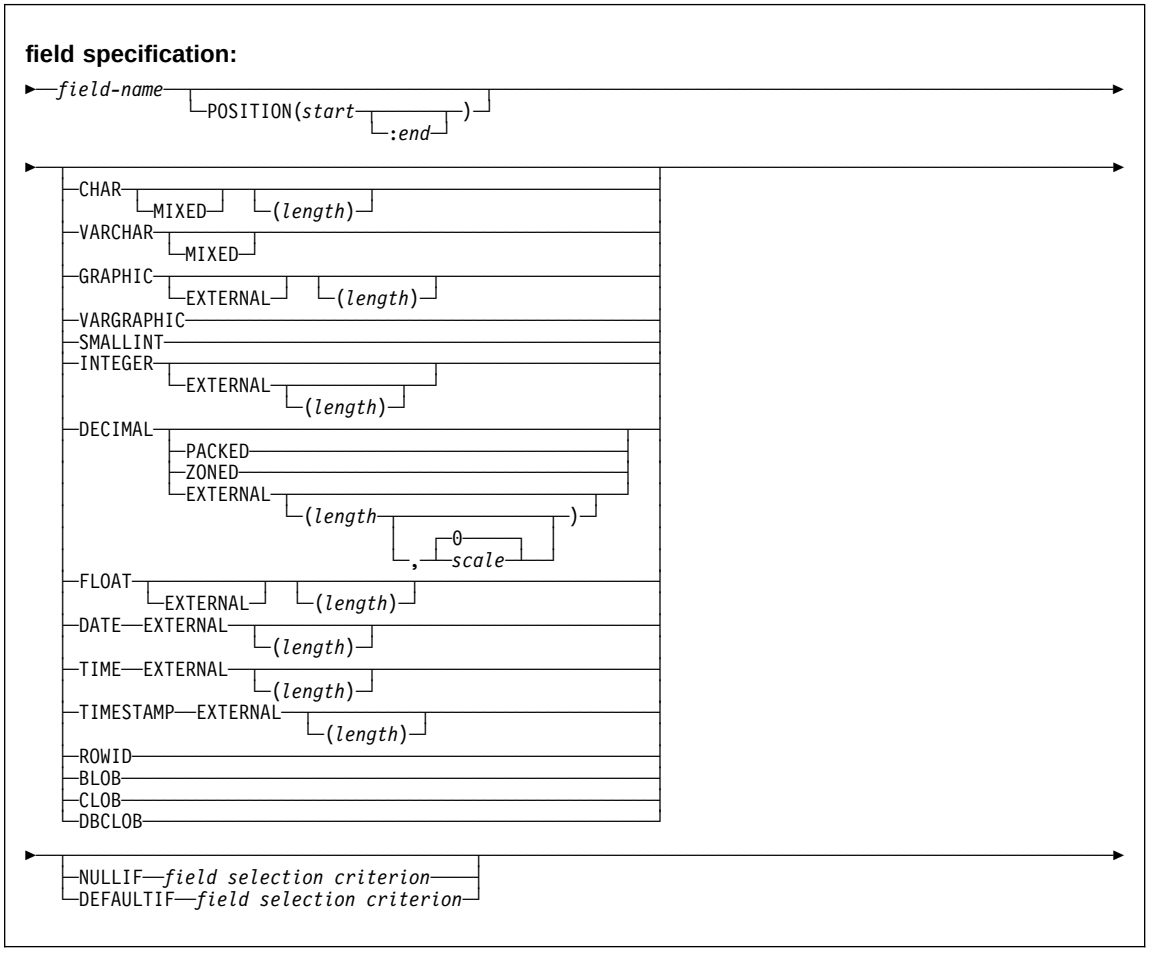
resume spec:



field selection criterion:



field specification:



Examples

Example 1: LOAD JCL with RESUME YES and ENFORCE NO.

LOAD

```
//STEP1    EXEC DSNUPROC,UID='IUIQU2UB.LOAD',
//         UTPROC='',
//         SYSTEM='V61A'
//SYSRECAC   DD DSN=IUIQU2UB.LOAD.DATA,DISP=SHR,VOL=SER=SCR03,
//         UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSUT1   DD DSN=IUIQU2UB.LOAD.STEP1.SYSUT1,
//         DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SORTOUT  DD DSN=IUIQU2UB.LOAD.STEP1.SORTOUT,
//         DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSIN    DD *
LOAD DATA INDDN(SYSRECAC) RESUME YES
     INTO TABLE DSN8610.ACT
          (ACTNO    POSITION( 1) INTEGER EXTERNAL(3),
           ACTKWD   POSITION( 5) CHAR(6),
           ACTDESC  POSITION(13) VARCHAR)
     ENFORCE NO
//*
```

Example 2: Control statement with RESUME YES option.

```
LOAD DATA
  RESUME YES
  INTO TABLE DSN8610.DEPT
  ( DEPTNO    POSITION (1:3)     CHAR(3),
    DEPTNAME  POSITION (4:39)    CHAR(36),
    MGRNO     POSITION (40:45)   CHAR(6),
    ADMRDEPT  POSITION (46:48)   CHAR(3),
    LOCATION  POSITION (49:64)   CHAR(16) )
```

Figure 1. Example of a LOAD utility statement

Example 3: Load data into a table. Load data from the data set specified by the EMPLDS DD statement into the EMP table.

```
LOAD DATA INDDN EMPLDS
  INTO TABLE DSN8610.EMP
```

Example 4: Load data into two tables. Load data from the data set specified by the EMPLDS DD statement into the DSN8610.EMP and SMITH.EMPEMPL tables.

```
LOAD DATA INDDN EMPLDS
  INTO TABLE DSN8610.EMP
  INTO TABLE SMITH.EMPEMPL
```

Example 5: Load selected records into a table. Load data from the data set specified by the EMPLDS DD statement into the EMP table. Load only from source input records that begin with LKA.

```
LOAD DATA INDDN EMPLDS
  INTO TABLE DSN8610.EMP
  WHEN (1:3)='LKA'
```

Example 6: Load selected records into a non-empty table space. The data from the sequential data set identified by the SYSREC DD statement is selectively loaded into the DSN8610.DEPT table whenever positions 1 through 3 contain the value LKA. The table space need not be empty for loading to proceed.

For each source record that has LKA in its first three positions:

- The characters in positions 7 through 9 are loaded into the DEPTNO column
- The characters in positions 10 through 35 are loaded into the DEPTNAME VARCHAR column
- The characters in positions 36 through 41 are loaded into the MGRNO column
- Characters in positions 42 through 44 are loaded into the ADMRDEPT column.

```
LOAD DATA
  RESUME YES
  INTO TABLE DSN8610.DEPT WHEN (1:3)='LKA'
  (DEPTNO POSITION (7:9) CHAR,
   DEPTNAME POSITION (10:35) CHAR,
   MGRNO POSITION (36:41) CHAR,
   ADMRDEPT POSITION (42:44) CHAR)
```

Example 7: Load selected records into an empty table space. Data from the sequential data set identified by the SYSRECPJ DD statement is selectively loaded into the DSN8610.PROJ table. The table space containing the DSN8610.PROJ table is currently empty, because the RESUME YES option was not specified.

For each source input record, data is loaded into the specified columns (that is, PROJNO, PROJNAME, DEPTNO ..., and so on) to form a table row. Any other columns in a DSN8610.PROJ row are set to NULL.

Starting positions of the fields in the sequential data set are defined by the field specification POSITION options. The ending position of the fields in the sequential data set are implicitly defined either by the length specification of the data type options (CHAR length) or by the length specification of the external numeric data type (LENGTH).

The numeric data represented in SQL constant format (EXTERNAL format) is converted to the correct internal format by the LOAD process and placed in the indicated column names. The two dates are assumed to be represented by eight digits and two separator characters, as in the USA format (for example, 11/15/1987). The length of the date fields is given as 10 explicitly, though in many cases it defaults to the same value.

```
LOAD DATA INDDN(SYSRECPJ)
  INTO TABLE DSN8610.PROJ
  (PROJNO    POSITION   (1) CHAR(6),
  PROJNAME   POSITION   (8) CHAR(22),
  DEPTNO     POSITION  (31) CHAR(3),
  RESPEMP    POSITION  (35) CHAR(6),
  PRSTAFF    POSITION  (42) DECIMAL EXTERNAL(5),
  PRSTDATE   POSITION  (48) DATE EXTERNAL(10),
  PRENDATE   POSITION  (59) DATE EXTERNAL(10),
  MAJPROJ    POSITION  (70) CHAR(6))
```

Example 8: Load data selectively using the CONTINUEIF option. Data from the sequential data set specified by the SYSRECOV DD statement is assembled and selectively loaded into the DSN8610.TOPTVAL table. The table space that contains DSN8610.TOPTVAL is currently empty because the RESUME YES option is not specified.

Fields destined for columns in the same table row can span more than one source record. Source records having fields containing columns that belong to the same row as the next source record all have an X in column 72 (that is, `CONTINUEIF(72:72)='X'`).

For each assembled source record, fields are loaded into the DSN8610.TOPTVAL table columns (that is, MAJSYS, ACTION, OBJECT ..., DSPINDEX) to form a table row. Any columns not mentioned are set to NULL.

The starting positions of the fields in the assembled source record input are given in the POSITION option. Starting positions are numbered from the first column of the internally assembled input record, not from the start of the source records in the sequential data set. The ending positions are defined by the character string lengths given with the input data type.

No conversions are required to load the source character strings into their designated columns, which are also defined to be fixed character strings. However, because columns INFOTXT, HELPTXT, and PFKTXT are defined as 79 characters in length and the strings being loaded are 71 characters in length, those strings are padded with blanks as they are loaded.

```
LOAD DATA INDDN(SYSRECOV) CONTINUEIF(72:72)='X'
  INTO TABLE DSN8610.TOPTVAL
  (MAJSYS   POSITION   (2) CHAR(1),
   ACTION   POSITION   (4) CHAR(1),
   OBJECT   POSITION   (6) CHAR(2),
   SRCHCRIT POSITION   (9) CHAR(2),
   SCRTYPE  POSITION  (12) CHAR(1),
   HEADTXT  POSITION  (80) CHAR(50),
   SELTXT   POSITION (159) CHAR(50),
   INFOTXT  POSITION (238) CHAR(71),
   HELPTXT  POSITION (317) CHAR(71),
   PFKTXT   POSITION (396) CHAR(71),
   DSPINDEX POSITION (475) CHAR(2))
```

Example 9: Load data with referential constraints. Data from the sequential data set identified by the SYSREC DD statement is loaded into the DSN8610.PROJ. table. Referential constraints are enforced on data added. Output consists of a summary report of violations of referential constraints, and all records causing these violations are placed in the SYSDISC discard data set.

```
LOAD DATA INDDN(SYSREC) CONTINUEIF(72:72)='X'
 RESUME YES
 ENFORCE CONSTRAINTS
 INTO TABLE DSN8610.PROJ
  (PROJNO POSITION (1) CHAR (6),
   PROJNAME POSITION (8) VARCHAR,
   DEPTNO POSITION (33) CHAR (3),
   RESPEMP POSITION (37) CHAR (6),
   PRSTAFF POSITION (44) DECIMAL EXTERNAL (5),
   PRSTDATE POSITION (50) DATE EXTERNAL,
   PRENDATE POSITION (61) DATE EXTERNAL,
   MAJPROJ POSITION (80) CHAR (6) NULLIF(MAJPROJ='      '))
```

Example 10: Load data using SORTKEYS. Use the SORTKEYS keyword to improve performance of the index key sort as shown in the following example. Assume there are 22,000 rows to load into the DSN8610.DEPT table. This table has 3 indexes.

This example specifies dynamic allocation of the required data sets by DFSORT™, using the SORTDEVT and SORTNUM keywords. If sufficient virtual storage resources are available, one utility subtask pair will be started to build each index. This example does not require UTPRIN*nn* DD statements, because it uses DSNUPROC to invoke utility processing, which includes a DD statement that allocates UTPRINT to SYSOUT.

LOAD statement:

```
//SAMPJOB  JOB  ...
//STEP1    EXEC DSNUPROC,UID='SAMPJOB.LOAD',UTPROC='',SYSTEM='V61A'
//SORTOUT  DD DSN=SAMPJOB.LOAD.STEP1.SORTOUT,DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//SYSUT1   DD DSN=SAMPJOB.LOAD.STEP1.SYSUT1,DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//SYSERR   DD DSN=SAMPJOB.LOAD.STEP1.SYSERR,DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(2000,(20,20),,,ROUND)
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=2400)
//SYSMAP   DD DSN=SAMPJOB.LOAD.STEP1.SYSMAP,DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(2000,(20,20),,,ROUND),
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=2400)
//SYSREC   DSN=SAMPJOB.TEMP.DATA,DISP=SHR,UNIT=SYSDA
//SYSIN    DD *
LOAD DATA REPLACE INDDN SYSREC CONTINUEIF(79:80)='++'
 SORTKEYS 66000 SORTDEVT SYSDA SORTNUM 3
 INTO TABLE DSN8610.DEPT
/*
```

Example 11: LOAD with inline copy. Use the REPLACE option with COPYDDN and RECOVERYDDN to create copies during LOAD.

```
LOAD DATA REPLACE INDDN INPUT
 SORTKEYS 66000
 COPYDDN SYSCOPY RECOVERYDDN REMCOPY
 CONTINUEIF(79:80)='++'
 INTO TABLE DSN8610.DEPT
```

Example 12: Load ASCII input data. Use the ASCII option to load ASCII input data into a table named MYASCIIT that was created with the CCSID ASCII clause.

```
LOAD REPLACE LOG NO ASCII INTO TABLE MYASCIIT
 (NAME POSITION(1) CHAR(40),
  ADDRESS POSITON(41) CHAR(40),
  ZIP POSITION(81) DECIMAL EXTERNAL(9),
  DEPARTMENT POSITION(90) CHAR(3),
  TITLE POSITION(93) GRAPHIC(20))
```

The CCSID keyword is not specified in this example; therefore, the CCSIDs of the ASCII input data are assumed to be the ASCII CCSIDs specified at installation. Conversions are done only if the CCSIDs of the target table differ from the ASCII CCSIDs specified at installation.

Example 13: Load data using statistics collection. Use the STATISTICS keyword to gather catalog statistics for the table space. This eliminates the need to run the RUNSTATS utility after completing the load operation. Specify REUSE so that all partitions are logically reset rather than deleted and redefined.

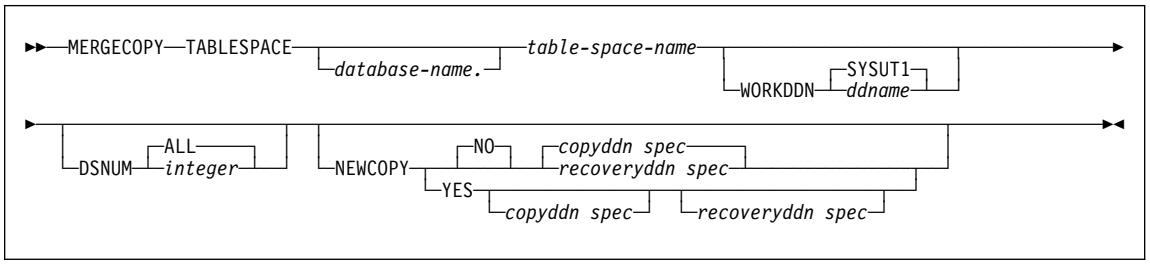
```
LOAD DATA
     INDDN SYSREC
     REPLACE
     STATISTICS TABLE(ALL)
     INDEX(ALL)
     REPORT YES UPDATE ALL
     REUSE
     CONTINUEIF(79:80)='++'
     INTO TABLE
     DSN8610.DEPT
```

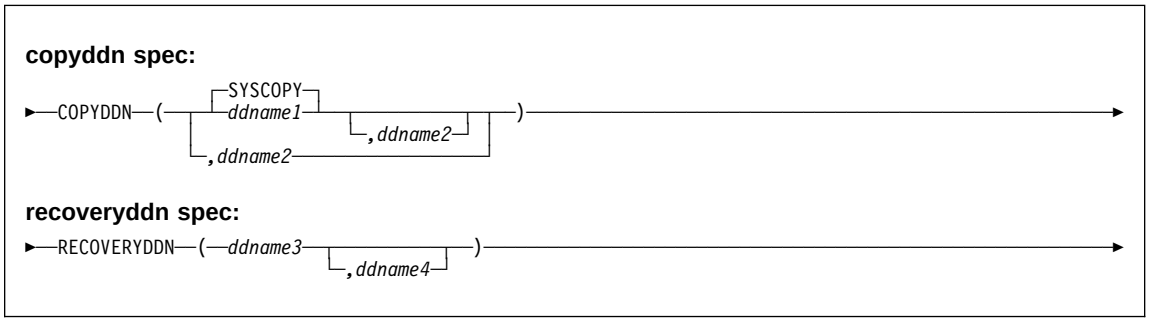
Example 14: Load data for a partitioned table space using statistics collection. Load data for a specified partition, using the STATISTICS keyword to gather catalog statistics of the partitioned table space.

```
LOAD STATISTICS
  INTO TABLE DSN8610.DEPT PART 1 REPLACE
```

MERGECOPY

Syntax





Examples

Example 1: Creating a merged incremental copy. Create a merged incremental image copy of table space DSN8S61C.

```
//STEP1    EXEC DSNUPROC,UID='IUJMU107.MERGE1',
//          UTPROC='',SYSTEM='V61A'
//COPY1  DD DSN=IUJMU107.MERGE1.STEP1.COPY1,DISP=(MOD,CATLG,CATLG),
//          UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//COPY2  DD DSN=IUJMU107.MERGE1.STEP1.COPY2,DISP=(MOD,CATLG,CATLG),
//          UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSUT1   DD DSN=IUJMU107.MERGE1.STEP1.SYSUT1,DISP=(MOD,DELETE,CATLG),
//          UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSIN    DD *
 MERGECOPY TABLESPACE DSN8D61P.DSN8S61C
  COPYDDN      (COPY1,COPY2)
  NEWCOPY  NO
```

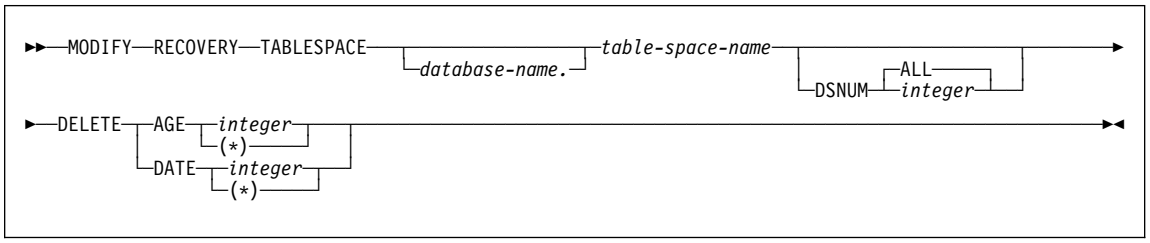
Example 2: Creating a merged full image copy. Create a merged full image copy of table space DSN8S61C.

MERGECOPY

```
//STEP1    EXEC DSNUPROC,UID='IUJMU107.MERGE2',
//          UTPROC='',SYSTEM='V61A'
//COPY1  DD DSN=IUJMU107.MERGE2.STEP1.COPY1,DISP=(MOD,CATLG,CATLG),
//          UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//COPY2  DD DSN=IUJMU107.MERGE2.STEP1.COPY2,DISP=(MOD,CATLG,CATLG),
//          UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSUT1   DD DSN=IUJMU107.MERGE2.STEP1.SYSUT1,DISP=(MOD,DELETE,CATLG),
//          UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSIN    DD *
MERGECOPY TABLESPACE DSN8D61P.DSN8S61C
  COPYDDN      (COPY1,COPY2)
  NEWCOPY  YES
```

MODIFY

Syntax



Examples

Example 1: Delete SYSCOPY records by age. For the table space containing the employee table, delete all SYSCOPY records older than 90 days.

```
//STEP1    EXEC DSNUPROC,UID='IUIQU2UD.MODRCV1',
//            UTPROC='',SYSTEM='V61A'
//SYSIN    DD *
MODIFY RECOVERY TABLESPACE DSN8D61A.DSN8S61E DELETE AGE(90)
/*
```

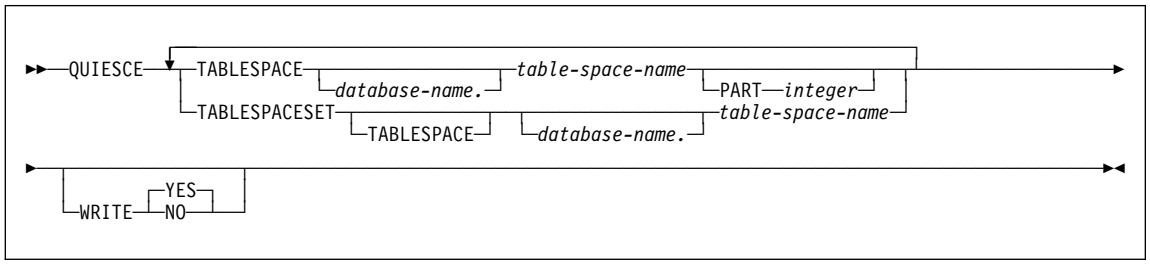
Example 2: Delete SYSCOPY records by date. For the table space containing the department table, delete all SYSCOPY records written before 10 September 1998.

```
MODIFY RECOVERY TABLESPACE DSN8D61A.DSN8S61D
  DELETE DATE(19980910)
```

QUIESCE

QUIESCE

Syntax



Examples

Example 1: Sample JCL for QUIESCE. Establish a quiesce point for three table spaces.

```
//STEP1    EXEC DSNUPROC,UID='IUIQU2UD.QUIESC2',
//            UTPROC='',SYSTEM='V61A'
//SYSIN    DD *
QUIESCE TABLESPACE DSN8D61A.DSN8S61D
        TABLESPACE DSN8D61A.DSN8S61E
        TABLESPACE DSN8D61A.DSN8S61P
//*
```

Example 2: Sample control statement for QUIESCE. Establish a quiesce point for the DSN8D61A.DSN8S61E and DSN8D61A.DSN8S61D table spaces.

```
QUIESCE TABLESPACE DSN8D61A.DSN8S61E TABLESPACE DSN8D61A.DSN8S61D
```

The following is output of the preceding command:

```
DSNU000I   DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = TSLQ.STEP1
DSNU050I   DSNUGUTC - QUIESCE TABLESPACE DSN8D61A.DSN8S61E
                              TABLESPACE DSN8D61A.DSN8S61D
DSNU477I - DSNUQUIA - QUIESCE SUCCESSFUL FOR TABLESPACE DSN8D61A.DSN8S61E
DSNU477I - DSNUQUIA - QUIESCE SUCCESSFUL FOR TABLESPACE DSN8D61A.DSN8S61D
DSNU474I - DSNUQUIA - QUIESCE AT RBA 000000052708
DSNU475I   DSNUQUIB - QUIESCE UTILITY COMPLETE, ELAPSED TIME= 00:00:25
DSNU010I   DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```

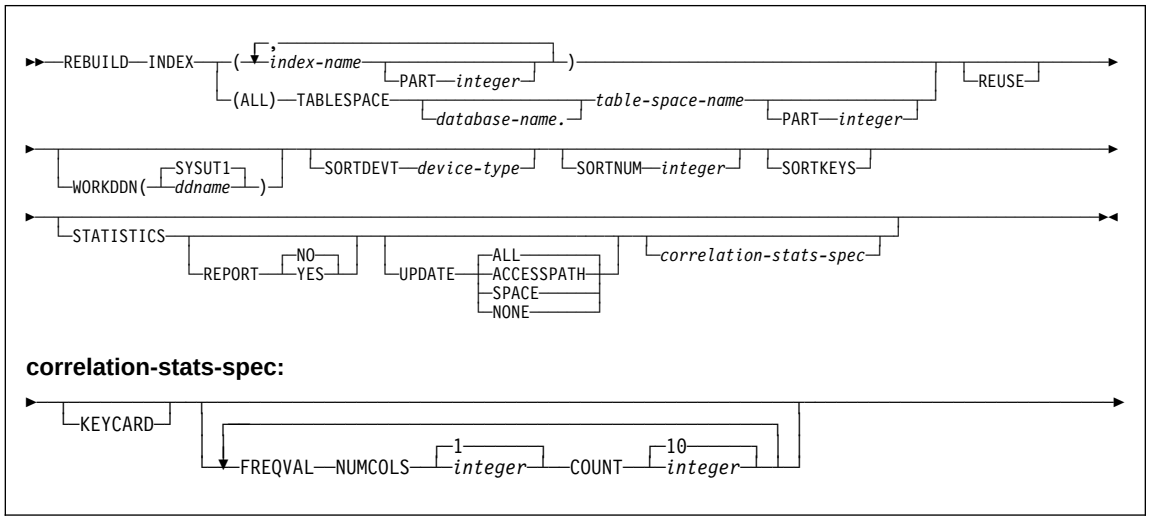
Example 3: QUIESCE point for a table space set. Establish a quiesce point for the table space set of the sample application.

```
QUIESCE TABLESPACESET TABLESPACE DSN8D61A.DSN8S61D
```

```
DSNU000I   DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = TSLQ.STEP1
DSNU050I   DSNUGUTC - QUIESCE TABLESPACESET TABLESPACE DSN8D61A.DSN8S61D
DSNU477I - DSNUQUIA - QUIESCE SUCCESSFUL FOR TABLESPACESET DSN8D61A.DSN8S61D
DSNU477I - DSNUQUIA -    QUIESCE SUCCESSFUL FOR TABLESPACE DSN8D61A.DSN8S61D
DSNU477I - DSNUQUIA -    QUIESCE SUCCESSFUL FOR TABLESPACE DSN8D61A.DSN8S61E
DSNU477I - DSNUQUIA -    QUIESCE SUCCESSFUL FOR TABLESPACE DSN8D61A.PROJ
DSNU477I - DSNUQUIA -    QUIESCE SUCCESSFUL FOR TABLESPACE DSN8D61A.ACT
DSNU477I - DSNUQUIA -    QUIESCE SUCCESSFUL FOR TABLESPACE DSN8D61A.PROJACT
DSNU477I - DSNUQUIA -    QUIESCE SUCCESSFUL FOR TABLESPACE DSN8D61A.EMPPROJA
DSNU477I - DSNUQUIA -    QUIESCE SUCCESSFUL FOR TABLESPACE DSN8D61A.DSN8S1D
DSNU474I - DSNUQUIA - QUIESCE AT RBA 000000052708 AND AT LRSN 000000052708
DSNU475I   DSNUQUIB - QUIESCE UTILITY COMPLETE, ELAPSED TIME= 00:00:25
DSNU010I   DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```

REBUILD INDEX

Syntax



Examples

Example 1: Rebuild an index. Rebuild the DSN8610.XDEPT1 index, which indexes the DSN8610.TDEPT table in the DSN8D61A database.

```
//STEP1    EXEC DSNUPROC,UID='IUIQU2UT.RBLD1',TIME=1440,
//         UTPROC='',
//         SYSTEM='V61A',DB2LEV=DB2A
//SYSREC   DD DSN=IUIQU2UT.RBLD1.STEP1.SYSREC,DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(8000,(20,20),,,ROUND)
//SYSUT1   DD DSN=IUIQU2UT.RBLD1.STEP1.SYSUT1,DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(8000,(20,20),,,ROUND)
//SYSIN    DD *
REBUILD INDEX (DSN8610.XDEPT1)
//*
```

Example 2: Rebuild index partitions. Rebuild partitions 2 and 3 of the DSN8610.XEMP1 index.

```
REBUILD INDEX (DSN8610.XEMP1 PART 2, DSN8610.XEMP1 PART 3)
```

Example 3: Rebuild a single index on a segmented table space. Rebuild the DSN8610.XDEPT1 index. This example specifies the SORTKEYS keyword to use parallelism and uses dynamic data set and message set allocation with the SORTDEVT and SORTNUM keywords.

DB2 starts one utility sort subtask pair to build the index. This example does not require UTPRIN*nn* DD statements, because it uses DSNUPROC to invoke utility processing, which includes a DD statement that allocates UTPRINT to SYSOUT.

```
//SAMPJOB  JOB  ...
//STEP1    EXEC DSNUPROC,UID='SAMPJOB.RCVINDEX',UTPROC='',SYSTEM='V61A'
//SYSIN    DD *
REBUILD INDEX (DSN8610.XDEPT1)
 SORTDEVT SYSWK
 SORTNUM  4
 SORTKEYS
/*
```

Example 4: Rebuild multiple partitions of a partitioning index. Rebuild partitions 2 and 3 of the DSN8610.XDEPT1 index, using parallel index build processing. This example specifies the SORTKEYS keyword to use parallelism and uses dynamic data set and message set allocation with the SORTDEVT and SORTNUM keywords.

If sufficient virtual storage resources are available, DB2 starts one utility sort subtask pair for each partition. This example does not require UTPRIN*nn* DD statements, because it uses DSNUPROC to invoke utility processing, which includes a DD statement allocating UTPRINT to SYSOUT.

```
//SAMPJOB  JOB  ...
//STEP1    EXEC DSNUPROC,UID='SAMPJOB.RBINDEX',UTPROC='',SYSTEM='V61A'
//SYSIN    DD *
REBUILD INDEX (DSN8610.XEMP1 PART 2, DSN8610.XEMP1 PART 3)
 SORTDEVT SYSWK
 SORTNUM  4
 SORTKEYS
/*
```

Example 5: Rebuild all partitions of a partitioning index. Rebuilds all index partitions of the DSN8610.XEMP1 partitioning index, using parallel index build processing. This example specifies the SORTKEYS keyword and allocates sort work data sets in two groups, which limits the number of utility subtask pairs to two. This example does not require UTPRIN*nn* DD statements, because it uses DSNUPROC to invoke utility processing, which includes a DD statement allocating UTPRINT to SYSOUT.

```
//SAMPJOB  JOB  ...
//STEP1    EXEC DSNUPROC,UID='SAMPJOB.RCVINDEX',UTPROC='',SYSTEM='V61A'
//* First group of sort work data sets for parallel index rebuild
//SW01WK01 DD UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//SW01WK02 DD UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//SW01WK03 DD UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//* Second group of sort work data sets for parallel index rebuild
//SW02WK01 DD UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//SW02WK02 DD UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//SW02WK03 DD UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//SYSIN    DD *
  REBUILD INDEX (DSN8610.XEMP1)
   SORTKEYS
/*
```

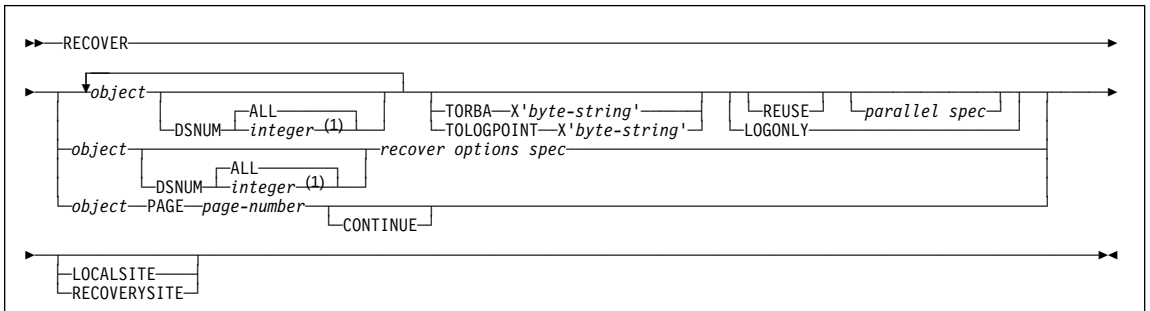
Example 6: Rebuild all indexes of a partitioned table space. Rebuild all indexes for table space DSN8S61E in database DSN8D61A, using parallel index build processing. This example specifies the SORTKEYS keyword and uses dynamic data set and message set allocation with the SORTDEVT and SORTNUM keywords.

If sufficient virtual storage resources are available, DB2 starts one utility sort subtask to build the partitioning index and another utility sort subtask to build the non-partitioning index. This example does not require UTPRIN*nn* DD statements, because it uses DSNUPROC to invoke utility processing, which includes a DD statement allocating UTPRINT to SYSOUT.

```
//SAMPJOB  JOB  ...
//STEP1    EXEC DSNUPROC,UID='SAMPJOB.RCVINDEX',UTPROC='',SYSTEM='V61A'
//SYSIN    DD *
REBUILD INDEX (ALL) TABLESPACE DSN8D61A.DSN8S61E
 SORTKEYS
 SORTDEVT SYSWK
 SORTNUM  4
/*
```

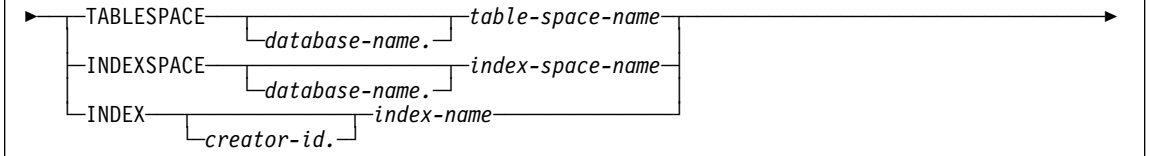
RECOVER

Syntax

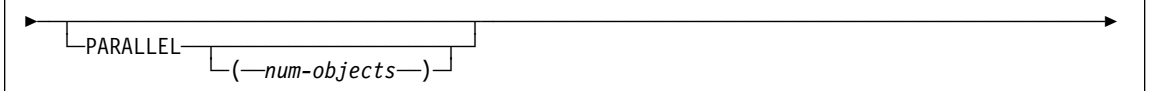


Note:
1 Not valid for nonpartitioning indexes.

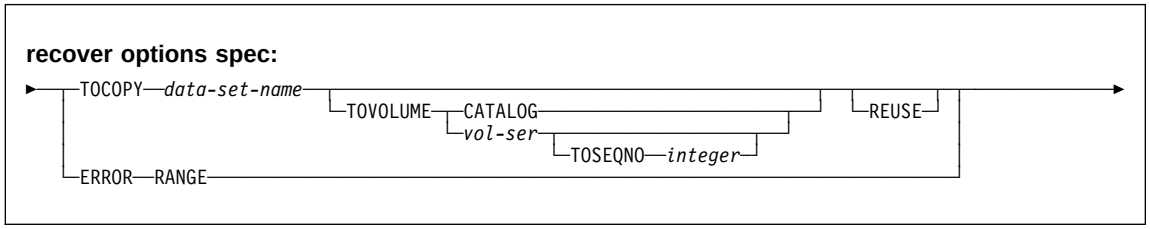
object:



parallel spec:



RECOVER



Examples

Example 1: Recover an error range. Recover from reported media failure in partition 2 of table space DSN8D61A.DSN8S61D.

```
//STEP5    EXEC DSNUPROC,UID='HUIAU326.RESTORE',TIME=1440,
//         UTPROC='',
//         SYSTEM='V61A',DB2LEV=DB2A
//SYSIN    DD *
RECOVER TABLESPACE DSN8D61A.DSN8S61D DSNUM 2 ERROR RANGE
/*
```

Example 2: Recover a table space. Recover table space DSN8S61D, in database DSN8D61A.

```
RECOVER TABLESPACE DSN8D61A.DSN8S61D
```

Example 3: Recover a table space partition. Recover the second partition of table space DSN8S61D.

```
RECOVER TABLESPACE DSN8D61A.DSN8S61D DSNUM 2
```

Example 4: Recover a table space to a specific RBA. Recover table spaces DSN8D61A.DSN8S61E and DSN8D61A.DSN8S61D to their quiesce point (RBA X'000007425468').

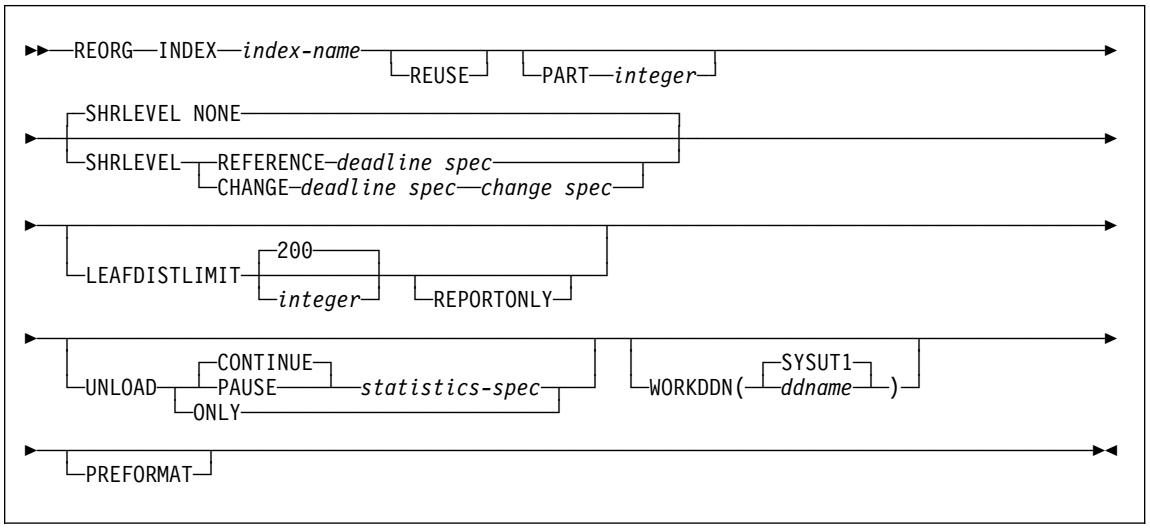
```
RECOVER TABLESPACE DSN8D61A.DSN8S61E DSNUM 2
        TABLESPACE DSN8D61A.DSN8S61D
        TORBA X'000007425468'
```

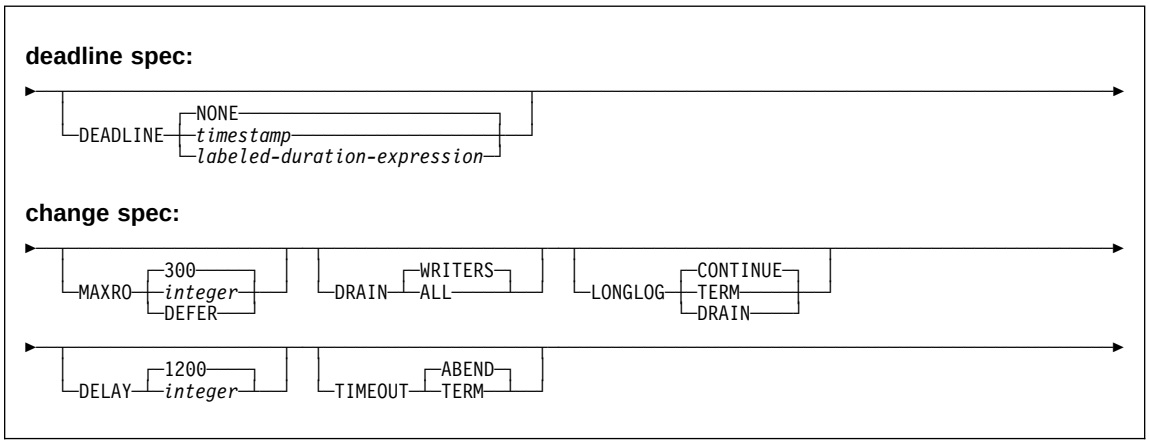
Example 5: Recover a list of objects to a point in time. The point in time is the common LRSN value from the SYSIBM.SYSCOPY records for the list of objects in the COPY SHRLEVEL REFERENCE job on page 334. The objects in the list are synchronized after successful completion of this RECOVER utility statement. This example restores four objects in parallel.

```
RECOVER TOLOGPOINT X'1234567890AB' PARALLEL(4)
      TABLESPACE DSN8D61A.DSN8S61D
      INDEX DSN8610.XDEPT1
      INDEX DSN8610.XDEPT2
      INDEX DSN8610.XDEPT3
      TABLESPACE DSN8D61A.DSN8S61E
      INDEX DSN8610.XEMP1
      INDEX DSN8610.XEMP2
```

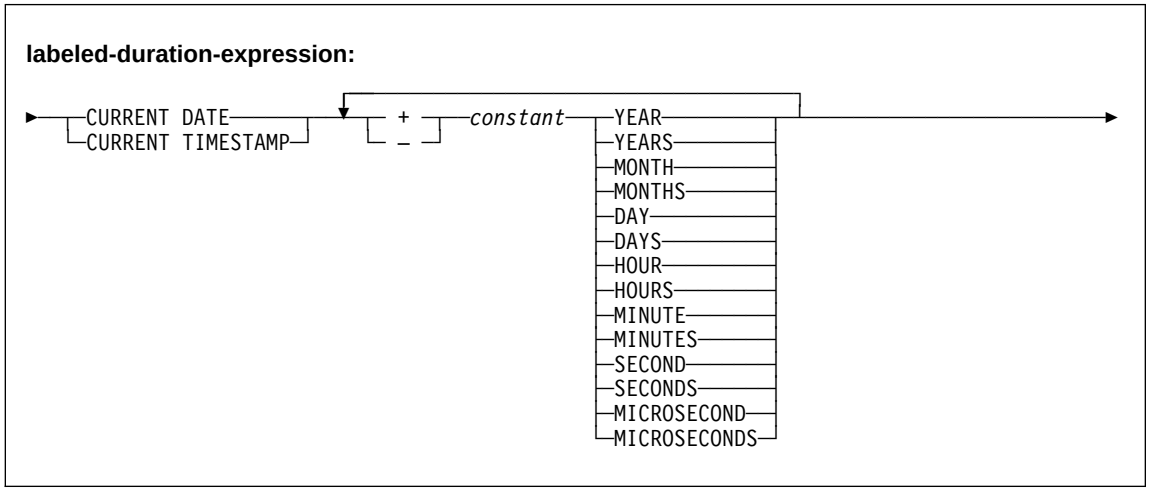
REORG INDEX

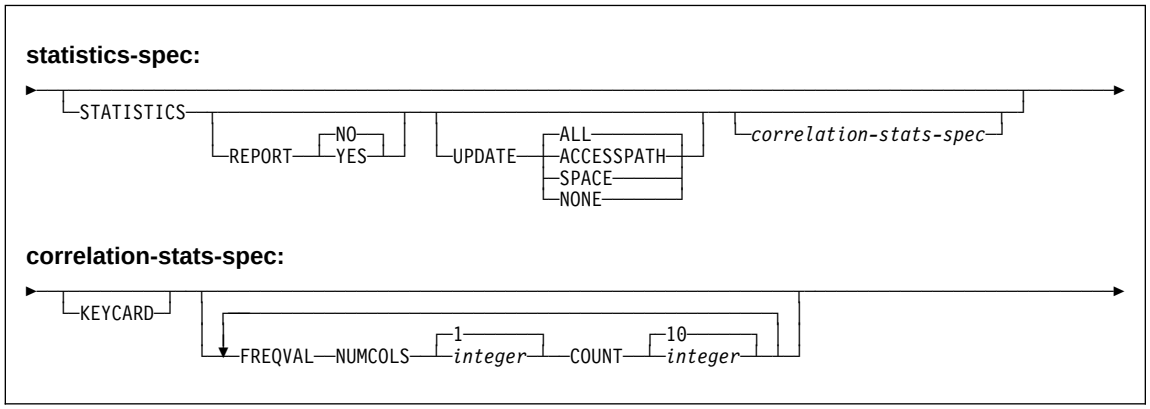
Syntax





REORG INDEX





Examples

Example 1: Reorganizing an index. Reorganize index XMSGTXT1. Stop the utility after the index keys have been unloaded, but allow for subsequent restart.

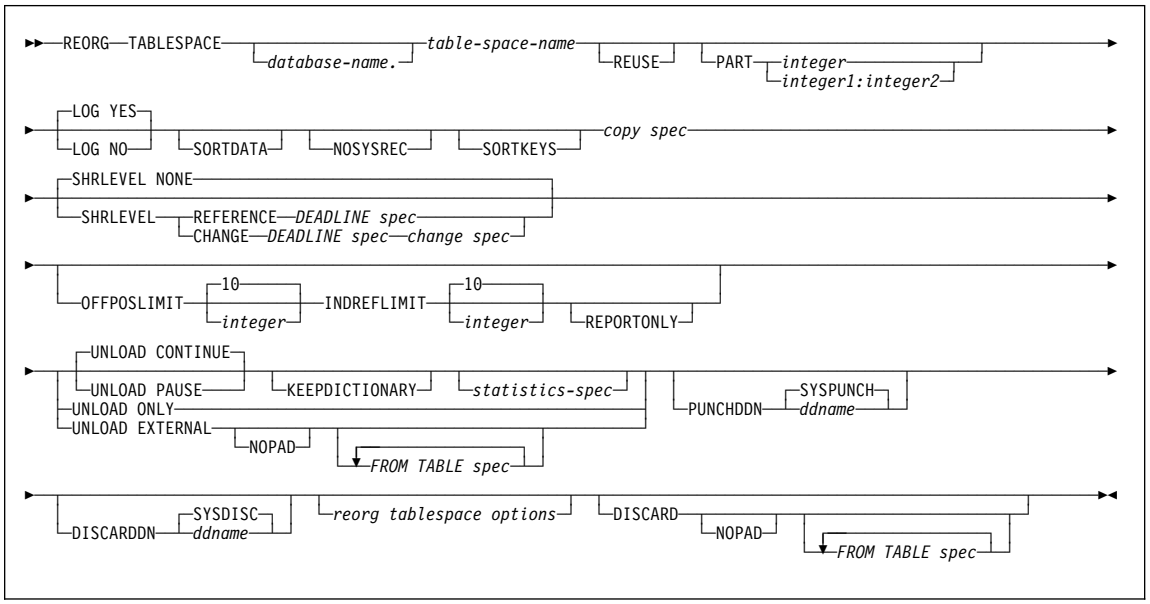
```
REORG INDEX DSN8610.XMSGTXT1
  UNLOAD PAUSE
```

Example 2: REORG INDEX using STATISTICS. Reorganize the index XEMPL1, using the STATISTICS option to update the catalog table statistics for this index.

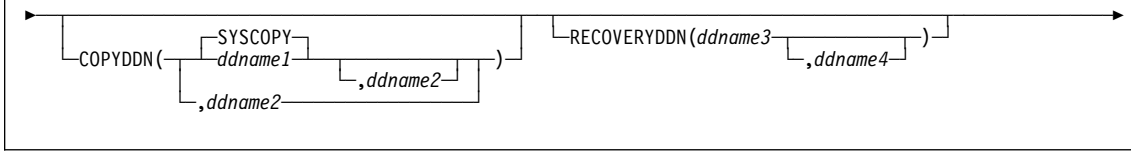
```
REORG INDEX DSN8610.XEMPL1
   SHRLEVEL REFERENCE STATISTICS
```

REORG TABLESPACE

Syntax

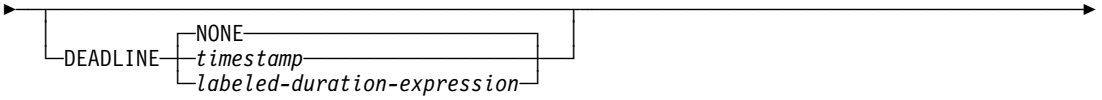


copy spec:

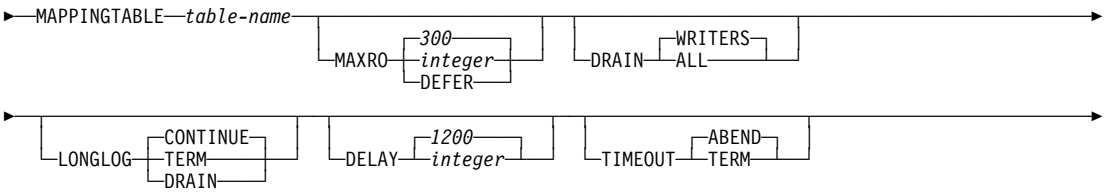


REORG TABLESPACE

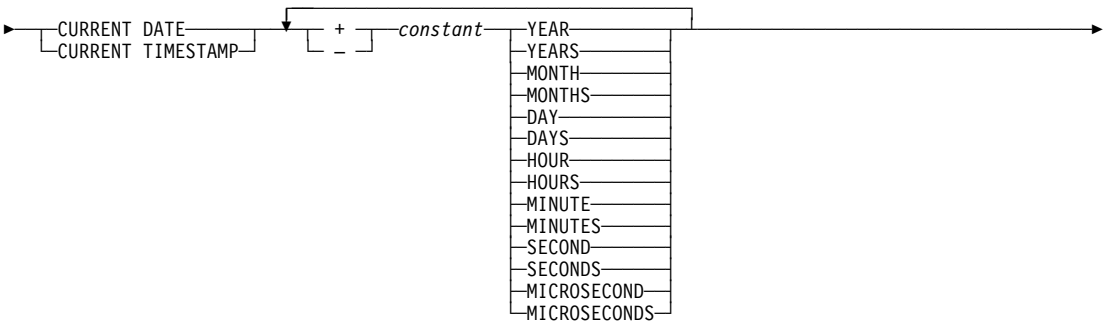
DEADLINE spec:



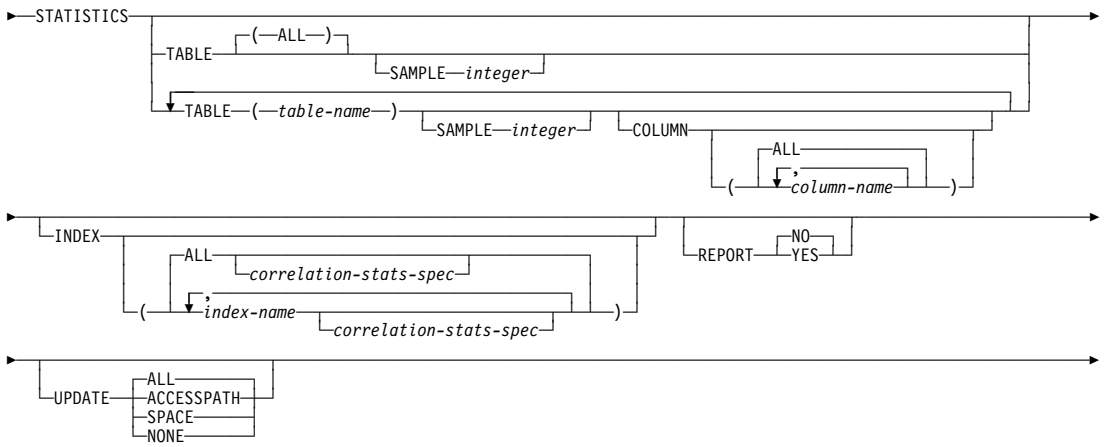
change spec:



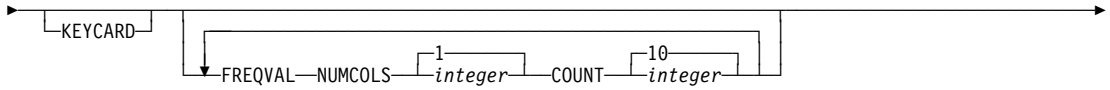
labeled-duration-expression:



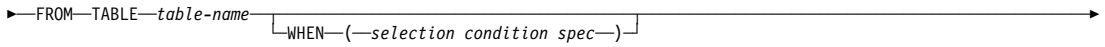
statistics-spec:



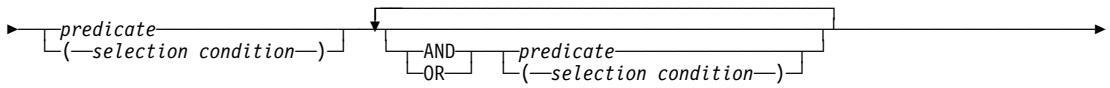
correlation-stats-spec:



FROM TABLE spec:



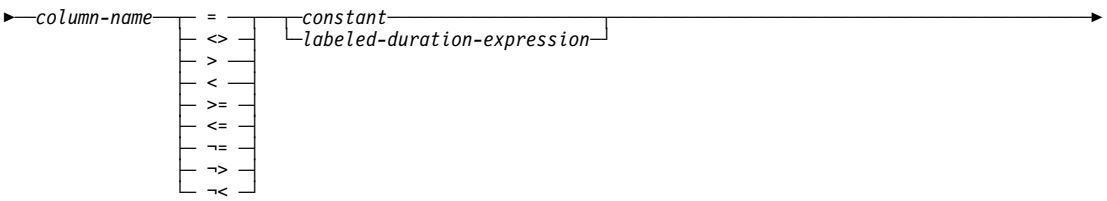
selection condition spec:



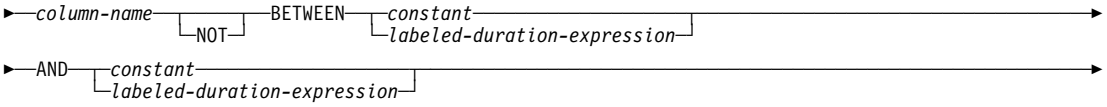
predicate:

REORG TABLESPACE

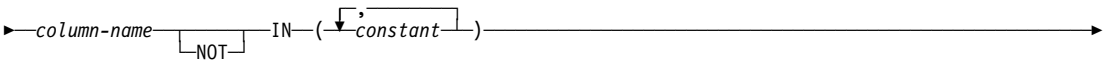
basic predicate:



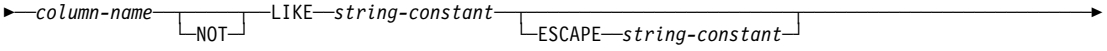
BETWEEN predicate:



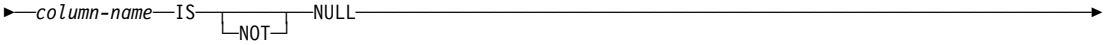
IN predicate:



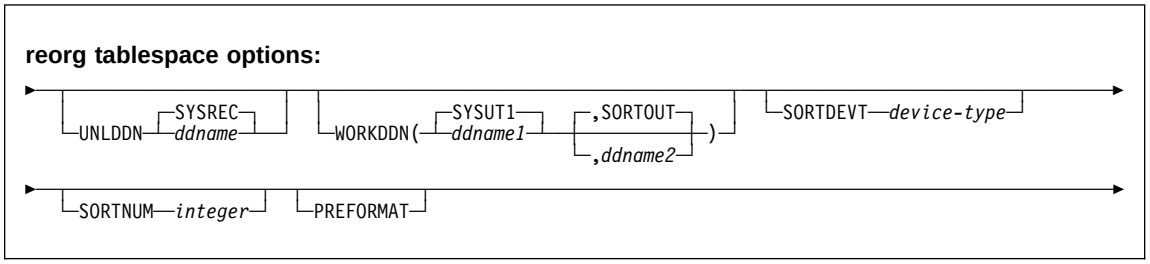
LIKE predicate:



NULL predicate:



REORG TABLESPACE options syntax



Examples

Example 1: REORG using default sort output data set. This example shows the DDNAME for the unload data set is UNLD, the DDNAME for the sort input data set is WORK, and the DDNAME for the sort output data set is defaulted to SORTOUT.

```
//STEP1    EXEC DSNUPROC,UID='IUJLU101.REORG',
//         UTPROC='',
//         SYSTEM='V61A'
//UTPRINT  DD  SYSOUT=*
//UNLD     DD DSN=IUJLU101.REORG.STEP1.UNLD,DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//DATAWK01  DD DSN=IUJLU101.REORG.STEP1.DATAWK01,DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SORTWK01  DD DSN=IUJLU101.REORG.STEP1.SORTWK01,DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SORTWK02  DD DSN=IUJLU101.REORG.STEP1.SORTWK02,DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSUT1   DD DSN=IUJLU101.REORG.STEP1.SYSUT1,DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SORTOUT  DD DSN=IUJLU101.REORG.STEP1.SORTOUT,DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSIN    DD *
REORG TABLESPACE (DSN8D61A.DSN8S61D)
      SORTDATA
      UNLDDN (UNLD)
      WORKDDN (WORK)
//*
```

Example 2: Reorganizing a table space. Reorganize table space DSN8S61D in database DSN8D61A.

```
REORG TABLESPACE DSN8D61A.DSN8S61D
         SORTDATA
```

Example 3: Reorganizing a table space partition. Reorganize partition 3 of table space DSN8S61E in database DSN8D61A.

REORG TABLESPACE

```
REORG TABLESPACE DSN8D61A.DSN8S61E
  PART 3
  SORTDATA
  SORTDEVT SYSDA
```

Example 4: REORG with DFSORT unloading by table space scan. Reorganize table space DSN8S61E in database DSN8D61A. Specify that DFSORT unloads the data by table space scan.

```
REORG TABLESPACE DSN8D61A.DSN8S61E SORTDATA
```

Example 5: REORG Using SORTKEYS. Use the SORTKEYS option to invoke parallel index build for a reorganization of the table space DSN8S61D in database DSNDB04. This example does not specify that dynamic allocation is to be used by DFSORT. Instead, it allocates sort work data sets in two groups, which limits the number of utility subtask pairs to two. This example does not require UTPRIN*nn* DD statements, because it uses DSNUPROC to invoke utility processing, which includes a DD statement that allocates UTPRINT to SYSOUT.

```
//SAMPJOB  JOB  ...
//STEP1    EXEC DSNUPROC,UID='SAMPJOB.REORG',UTPROC='',SYSTEM='V61A'
//SYSREC   DD DSN=SAMPJOB.REORG.STEP1.SYSREC,DISP=(NEW,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//* First group of sort work data sets for parallel index build
//SW01WK01 DD UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//SW01WK02 DD UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//SW01WK03 DD UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//* Second group of sort work data sets for parallel index build
//SW02WK01 DD UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//SW02WK02 DD UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//SW02WK03 DD UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//* Sort work data sets for use by SORTDATA
//SORTWK01 DD  UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//SORTWK02 DD  UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//SORTWK03 DD  UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND)
//SYSIN    DD *
REORG TABLESPACE DSNDB04.DSN8S61D LOG NO SORTDATA SORTKEYS
/*
```

Example 6: REORG Using SORTKEYS while allowing read-write access. Use the SORTKEYS option to invoke parallel index build for reorganization of the table space DSN8S61E in database DSNDB04 in database DSNDB04. The name of the mapping table is DSN8MAP. DFSORT dynamically allocates sort work data sets. This example does not require UTPRIN*nn* DD statements, because it uses DSNUPROC to invoke utility processing, which includes a DD statement that allocates UTPRINT to SYSOUT.

```
//SAMPJOB  JOB  ...
//STEP1    EXEC DSNUPROC,UID='SAMPJOB.REORG',UTPROC='',SYSTEM='V61A'
//SYSCOPY  DD UNIT=SYSDA,SPACE=(CYL,(10,20),,,ROUND),
//            DSN=SAMPJOB,COPY,DISP=(NEW,CATLG,CATLG)
//SYSIN    DD *
REORG TABLESPACE DSNDB04.DSN8S61E LOG NO SORTDEVT SYSDA SORTNUM 4
  SHRLEVEL CHANGE MAPPINGTABLE DSN8MAP
/*
```

Example 7: Reorganizing a table while allowing read-only access. Reorganize table space DSN8S61D in database DSN8D61A. The deadline for start of the SWITCH phase is *3:15* on *February 4, 1997.*

```
REORG TABLESPACE DSN8D61A.DSN8S61D COPYDDN(MYCOPY1)
  RECOVERYDDN(MYCOPY2) SHRLEVEL REFERENCE
  DEADLINE 1997-2-4-03.15.00
```

Example 8: Reorganizing a table while allowing read-write access. Reorganize table space DSN8S61D in database DSN8D61A. The deadline for start of the SWITCH phase is *3:15* on February 4, 1997. The name of the mapping table is MYMAPTABLE. The maximum desired amount of time for the log processing in the read-only (last) iteration of log processing is *240* seconds. If reorganization's reading of the log is not catching up to applications' writing of the log quickly enough, DB2 will drain the write claim class after sending the LONGLOG message to the operator. That draining will take place at least 900 seconds after the LONGLOG message is sent.

```
REORG TABLESPACE DSN8D61A.DSN8S61D COPYDDN(MYCOPY1)
  RECOVERYDDN(MYCOPY2) SHRLEVEL CHANGE
  DEADLINE 1997-2-4-03.15.00
  MAPPINGTABLE
DSN8610.MAP_TBL MAXRO 240 LONGLOG DRAIN DELAY 900
```

Example 9: Reorganizing a range of table space partitions. Reorganize partitions 3 through 5 of table space DSN8S61E in database DSN8D61A.

```
REORG TABLESPACE DSN8D61A.DSN8S61E
  PART 3:5
  STATISTICS
  SORTDEVT SYSDA
    SHRLEVEL NONE
    COPYDDN SYSCOPY
    SORTDATA
```

Example 10: REORG a partition using STATISTICS. Reorganize partition 3 of table space DSN8S61E in database DSN8D61A, using the STATISTICS option to update catalog table statistics for that table.

```
REORG TABLESPACE DSN8D61A.DSN8S61E
   SORTDATA STATISTICS PART 3
```

Example 11: REORG using STATISTICS to update table space and index statistics. Reorganize table space DSN8S61E in database DSN8D61A, using the

STATISTICS option to update catalog statistics for the table space and all indexes defined on that table.

```
REORG TABLESPACE DSN8D61A.DSN8S61E SORTDATA STATISTICS
   TABLE
   INDEX(ALL) KEYCARD FREQVAL NUMCOLS 1
   COUNT 10 REPORT YES UPDATE NONE
```

Example 12: Checking if a table should be reorganized. Report if the OFFPOSLIMIT or INDREFLIMIT values are exceeded for the TPHR5201 table space in database DBHR5201.

```
//STEP1    EXEC DSNUPROC,UID='HUHRU252.REORG2',TIME=1440,
//         UTPROC='',
//         SYSTEM='V61A',DB2LEV=DB2A
//SYSREC   DD DSN=HUHRU252.REORG2.STEP1.SYSREC,DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSCOPY  DD DSN=HUHRU252.REORG2.STEP1.SYSCOPY,DISP=(MOD,CATLG,CATLG),
//         UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSUT1   DD DSN=HUHRU252.REORG2.STEP1.SYSUT1,DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SORTOUT  DD DSN=HUHRU252.REORG2.STEP1.SORTOUT,
//         DISP=(MOD,DELETE,CATLG),UNIT=SYSDA,
//         SPACE=(4000,(20,20),,,ROUND)
//SYSIN    DD *
REORG TABLESPACE DBHR5201.TPHR5201 PART 11
      SORTDATA SORTKEYS NOSYSREC
      REPORTONLY
      SHRLEVEL CHANGE MAPPINGTABLE ADMF001.MAP1
      COPYDDN (SYSCOPY)
          OFFPOSLIMIT 11 INDREFLIMIT 15
          STATISTICS UPDATE SPACE
/*
```

On successful completion, DB2 returns output similar to the following output:

```
DSNU050I   DSNUGUTC - REORG TABLESPACE DBHR5201.TPHR5201 PART 11 SORTDATA SORTKEYS NOSYSREC REPORTONLY SHRLEVEL CHA
NGE MAPPINGTABLE ADMF001.MAP1 COPYDDN(SYSCOPY) OFFPOSLIMIT 11 INDREFLIMIT 15 STATISTICS UPDATE SPACE
DSNU286I ( DSNURLIM - REORG TABLESPACE DBHR5201.TPHR5201 OFFPOSLIMIT SYSINDEXPART ROWS
  CREATOR .IXNAME             CREATOR .TBNAME             PART CARDF       FAROFFPOSF  NEAROFFPOSF      STATSTIME
* ADMF001 .IPHR5201           ADMF001 .TBHR5201             11  6.758E+03   2.972E+03   7.38E+02   1999-02-05-08.27.04
DSNU287I ( DSNURLIM - REORG TABLESPACE DBHR5201.TPHR5201 INDREFLIMIT SYSTABLEPART ROWS
  DBNAME  .TSNAME   PART         CARD  FARINDREF NEARINDREF      STATSTIME
  DBHR5201.TPHR5201   11         6758          0          0 1999-02-05-08.27.04
DSNU289I ( DSNURLIM -  REORG LIMITS HAVE BEEN MET
DSNU010I   DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=2
DSNU000I   DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = HUHRU252.REORG2
DSNU050I   DSNUGUTC - REORG INDEX ADMF001.IPHR5201 PART 11 LEAFDISTLIMIT 2 REPORTONLY
DSNU288I ( DSNURLIM - REORG INDEX ADMF001.IPHR5201 LEAFDISTLIMIT SYSINDEXPART ROWS
  CREATOR .IXNAME             PART   LEAFDIST      STATSTIME
* ADMF001 .IPHR5201             11          3 1999-02-05-08.27.04
DSNU289I ( DSNURLIM -  REORG LIMITS HAVE BEEN MET
DSNU050I   DSNUGUTC - CHECK INDEX(ADMF001.IPHR5201 PART 11)
DSNU700I ( DSNUKGET - 6761 INDEX ENTRIES UNLOADED FROM INDEX='ADMF001.IPHR5201' PARTITION= 11
DSNU705I   DSNUK001 - UNLOAD PHASE COMPLETE - ELAPSED TIME=00:00:00
DSNU042I   DSNUGSOR - SORT PHASE STATISTICS -
                      NUMBER OF RECORDS=6761
                      ELAPSED TIME=00:00:01
DSNU717I ( DSNUKTER - 6761 ENTRIES CHECKED FOR INDEX 'ADMF001.IPHR5201' PARTITION= 11
DSNU720I   DSNUK001 - CHECKIDX PHASE COMPLETE, ELAPSED TIME=00:00:02
DSNU010I   DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=2
```

Figure 2. Sample output showing REORG limits have been met

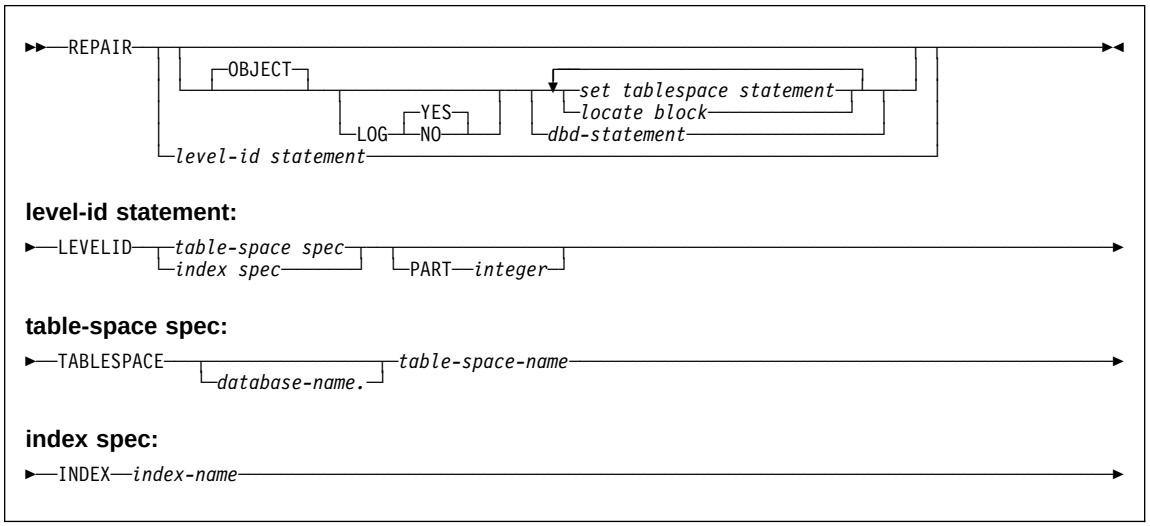
Example 13: Conditionally reorganizing a table. To ensure recent statistics for the table space, execute the RUNSTATS utility for the TPHR5201 table space. Then, reorganize the TPHR5201 table space in database DBHR5201 if the OFFPOSLIMIT or INDREFLIMIT value is exceeded.

REORG TABLESPACE

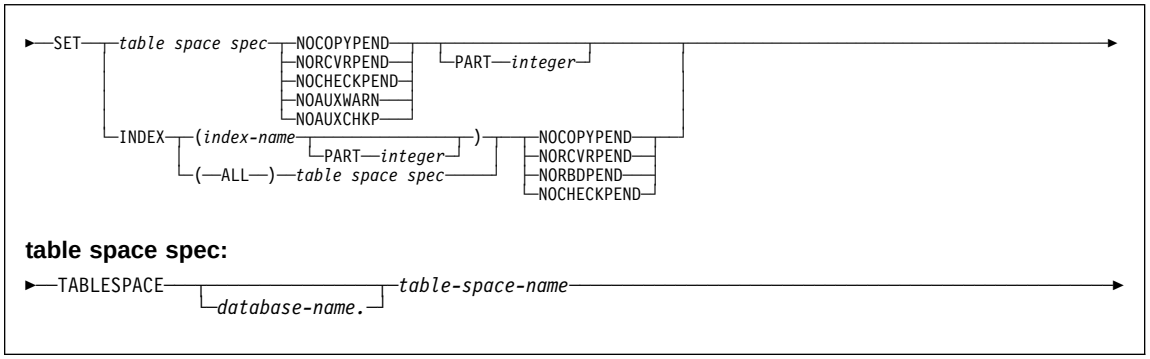
```
//*****************************************************************
//* COMMENT: UPDATE STATISTICS
//*****************************************************************
//STEP1    EXEC DSNUPROC,UID='HUHRU252.REORG1',TIME=1440,
//         UTPROC='',
//         SYSTEM=V61AR',DB2LEV=DB2A
//SYSREC   DD DSN=HUHRU252.REORG1.STEP1.SYSREC,DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSUT1   DD DSN=HUHRU252.REORG1.STEP1.SYSUT1,DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SORTOUT  DD DSN=HUHRU252.REORG1.STEP1.SORTOUT,
//         DISP=(MOD,DELETE,CATLG),UNIT=SYSDA,
//         SPACE=(4000,(20,20),,,ROUND)
//SYSIN    DD *
RUNSTATS TABLESPACE DBHR5201.TPHR5201
      UPDATE SPACE
/*
//*****************************************************************
//* COMMENT: REORG THE TABLESPACE
//*****************************************************************
//STEP2    EXEC DSNUPROC,UID='HUHRU252.REORG1',TIME=1440,
//         UTPROC='',
//         SYSTEM='V61A',DB2LEV=DB2A
//SYSREC   DD DSN=HUHRU252.REORG1.STEP1.SYSREC,DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SYSCOPY1 DD DSN=HUHRU252.REORG1.STEP1.SYSCOPY1,
//         DISP=(MOD,CATLG,CATLG),UNIT=SYSDA,
//         SPACE=(4000,(20,20),,,ROUND)
//SYSUT1   DD DSN=HUHRU252.REORG1.STEP1.SYSUT1,DISP=(MOD,DELETE,CATLG),
//         UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SORTOUT  DD DSN=HUHRU252.REORG1.STEP1.SORTOUT,
//         DISP=(MOD,DELETE,CATLG),UNIT=SYSDA,
//         SPACE=(4000,(20,20),,,ROUND)
//SYSIN    DD *
REORG TABLESPACE DBHR5201.TPHR5201
      SORTDATA NOSYSREC SORTKEYS
      COPYDDN SYSCOPY1
      OFFPOSLIMIT
      INDREFLIMIT
      STATISTICS TABLE(ALL) INDEX(ALL)
/*
```

REPAIR

Syntax

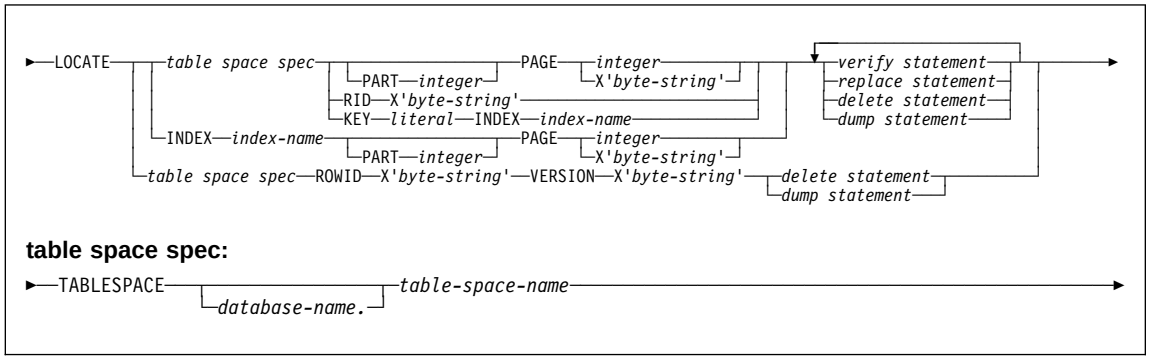


SET TABLESPACE and SET INDEX statement syntax

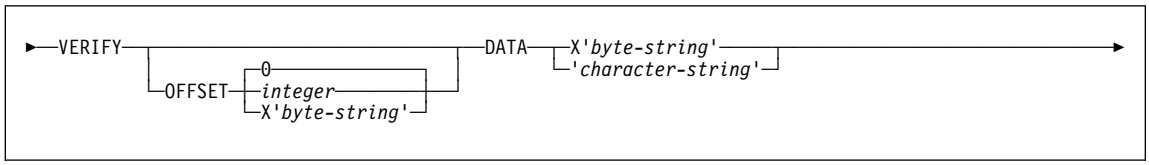


REPAIR

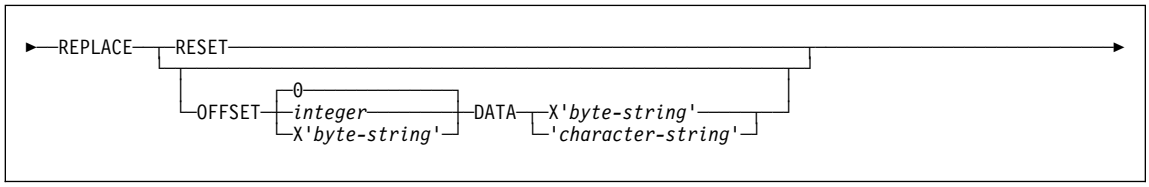
LOCATE block syntax



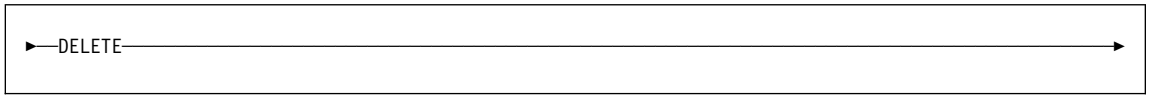
VERIFY statement syntax



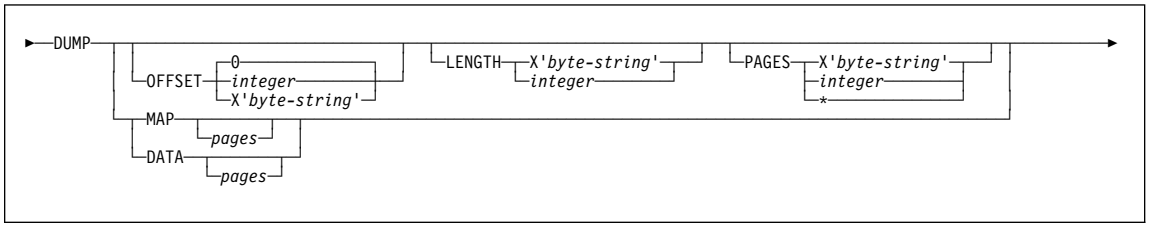
REPLACE statement syntax



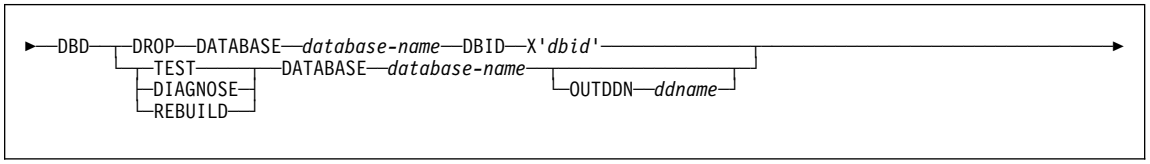
DELETE statement syntax and description



DUMP statement syntax



DBD statement syntax



Examples

Example 1: Replacing damaged data and verifying replacement. Repair the specified page of table space DSN8S61E. Verify that, at the specified offset (50), the damaged data (A00) is found. Replace it with the desired data (D11). Initiate a dump beginning at offset 50, for 4 bytes, to verify the replacement.

```
//STEP1    EXEC  DSNUPROC,UID='IUIQU1UH',UTPROC='',SYSTEM='V61A'
//SYSIN DD *
REPAIR OBJECT
  LOCATE TABLESPACE DSN8D61A.DSN8S61D PAGE X'02'
    VERIFY OFFSET 50 DATA X'A00'
    REPLACE OFFSET 50 DATA X'D11'
    DUMP OFFSET 50 LENGTH 4
```

Example 2: Removing a nonindexed row found by REORG. When reorganizing table space DSNDB04.TS1, you received the following message:

```
DSNU3401 DSNURBXA - ERROR LOADING INDEX, DUPLICATE KEY
                    INDEX = EMPINDEX
                    TABLE = EMP
                    RID OF INDEXED ROW  = X'00000201'
                    RID OF NONINDEXED ROW = X'00000503'
```

To resolve this error message, delete the nonindexed row and log the change. (The LOG keyword is not required; it is logged by default.)

```
REPAIR
  LOCATE TABLESPACE DSNDB04.TS1 RID (X'00000503')
    DELETE
```

Example 3: Report whether catalog and directory DBDs differ. Determine if the DBDs in the DB2 catalog and the DB2 directory are consistent for database DSN8D2AP.

```
REPAIR DBD TEST DATABASE DSN8D2AP
```

Example 4: Report differences between catalog and directory DBDs. After running the TEST option on database DSN8D2AP, and determining that the DBDs are inconsistent, determine the differences between the DBDs.

```
REPAIR DBD DIAGNOSE DATABASE DSN8D2AP OUTDDN SYSREC
```

Example 5: REPAIR table space with orphan row. After running DSN1CHKR on table space SYSDBASE, you received the following message:

```
DSN1812I ORPHAN ID = 20 ID ENTRY = 0190 FOUND IN
         PAGE = 000024
```

From a DSN1PRNT of page X'000024' and X'00002541', you identify that RID X'00002420' has a forward pointer of X'00002521'.

Repair the table space as follows:

1. Set the orphan's backward pointer to zeros.

   ```
   REPAIR OBJECT LOG YES
    LOCATE TABLESPACE DSNDB06.SYSDBASE RID X'00002420
     VERIFY OFFSET X'0A' DATA X'00002422'
     REPLACE OFFSET X'0A' DATA X'00000000'
   ```

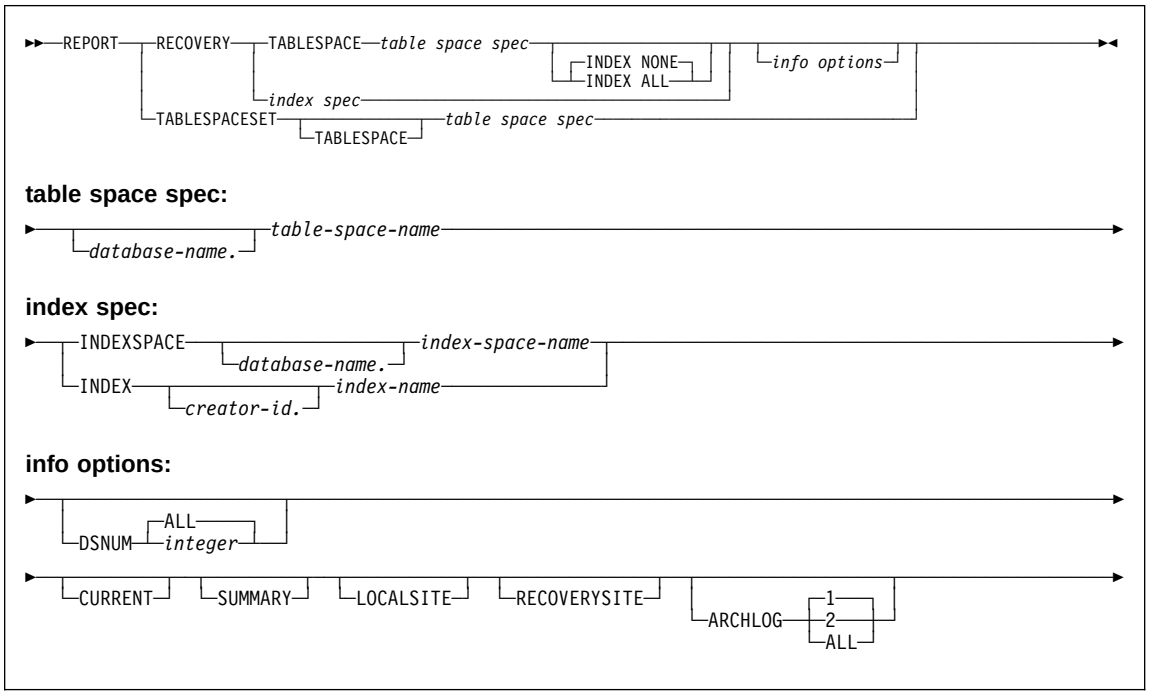
   Setting the pointer to zeros prevents the next step from updating link pointers while deleting, which can cause DB2 to abend if the orphan's pointers are incorrect.

2. Delete the orphan.

   ```
   REPAIR OBJECT LOG YES
    LOCATE TABLESPACE DSNDB06.SYSDBASE RID X'00002420'
     VERIFY OFFSET X'06' DATA X'00002521'
     DELETE
   ```

REPORT

Syntax



Examples

Example 1: Sample JCL for REPORT RECOVERY.

```
//STEP1    EXEC DSNUPROC,UID='IUKUU206.REPORT2',
//            UTPROC='',SYSTEM='V61A'
//SYSIN    DD *
REPORT RECOVERY
       TABLESPACE DSN8D61A.DSN8S61E
//*
```

Example 2: Sample control statement for REPORT TABLESPACESET.

```
REPORT TABLESPACESET
       TABLESPACE UTQPD22A.UTQPS22E
```

Example 3: REPORT referentially related table spaces. The following statement reports the names of all table spaces in the table space set containing table space DSN8D61A.DSN8S61E.

```
REPORT TABLESPACESET TABLESPACE DSN8D61A.DSN8S61E
```

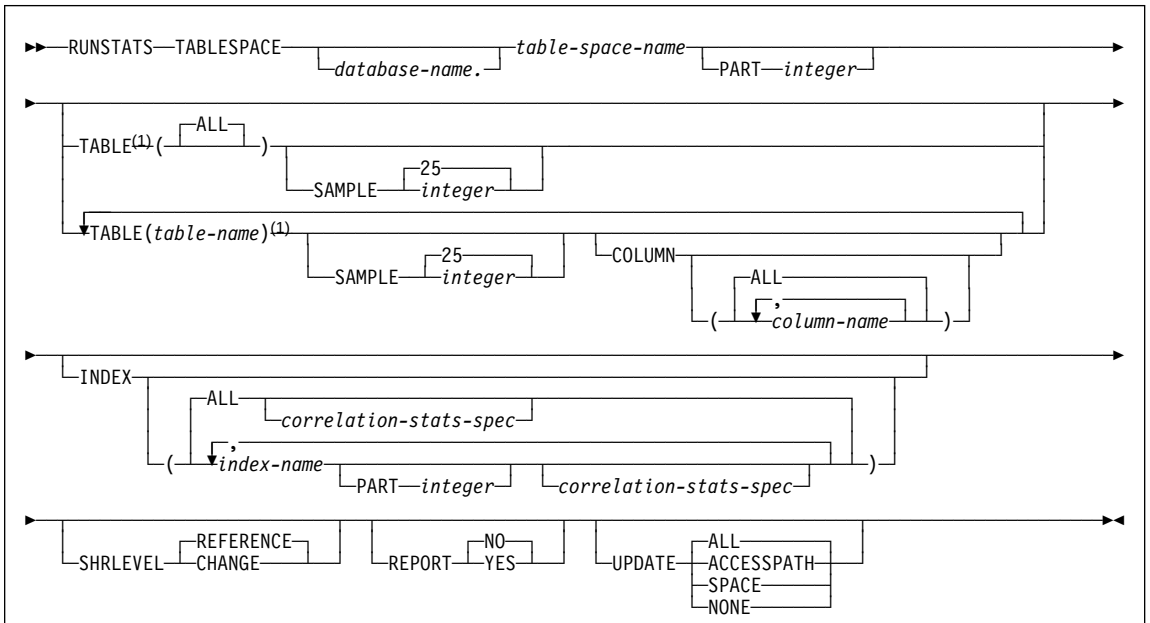
Example 4: REPORT RECOVERY information for a table space. This statement reports recovery information for table space DSN8D61A.DSN8S61D.

```
REPORT RECOVERY TABLESPACE DSN8D61A.DSN8S61D DSNUM ALL
```

RUNSTATS

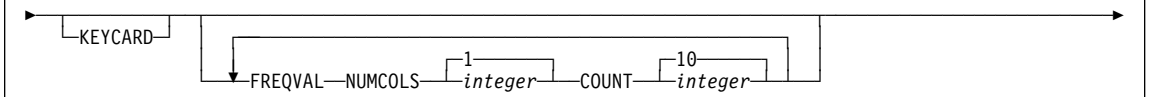
Syntax

RUNSTATS TABLESPACE syntax



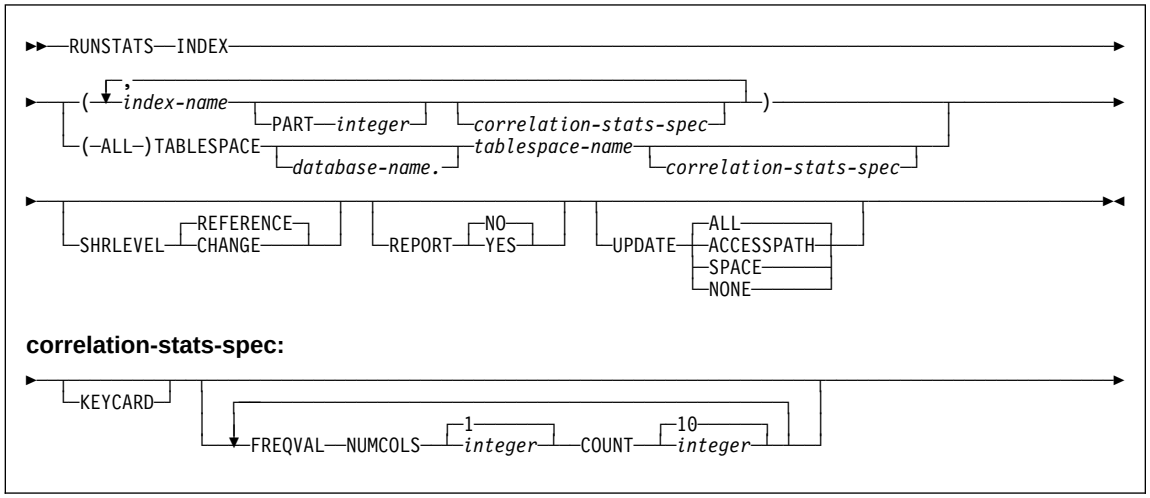
Note:
1 The TABLE keyword is not valid for a LOB table space.

correlation-stats-spec



RUNSTATS

RUNSTATS INDEX syntax



Examples

Example 1: Update catalog statistics while allowing changes. Update the catalog statistic columns for table space DSN8S61E and all its associated indexes, sampling 25 percent of the rows. Permit other processes to make changes while this utility is executing.

```
//STEP1    EXEC DSNUPROC,UID='IUJQU225.RUNSTA',TIME=1440,
//         UTPROC='',
//         SYSTEM='V61A',DB2LEV=DB2A
//UTPRINT  DD  SYSOUT=*
//SYSIN    DD *
RUNSTATS TABLESPACE DSN8D61A.DSN8S61E
   TABLE(ALL) SAMPLE 25
    INDEX(ALL)
   SHRLEVEL CHANGE
```

Example 2: Update catalog statistics, do not allow updates. Update the catalog statistics for indexes XEMPL1 and XEMPL2. Do not permit other processes to change the table space associated with XEMPL1 and XEMPL2 (table space DSN8S61E) while this utility is executing.

```
RUNSTATS INDEX (DSN8610.XEMPL1,DSN8610.XEMPL2)
```

Example 3: Update index statistics. Obtain statistics on the index XEMPL1.

```
RUNSTATS INDEX (DSN8610.XEMPL1)
```

Example 4: Update statistics for several tables. Update the catalog statistics for all columns in the TCONA and TOPTVAL tables in table space DSN8D61P.DSN8S61C.

Update the column statistics for the LINENO and DSPLINE columns in the TDSPTXT table in table space DSN8D61P.DSN8S61C.

```
RUNSTATS TABLESPACE(DSN8D61P.DSN8S61C) TABLE (TCONA)
                                  TABLE (TOPTVAL) COLUMN(ALL)
                                  TABLE (TDSPTXT) COLUMN(LINENO,DSPLINE)
```

Example 5: Update all statistics for a table space. Update all catalog statistics (table space, tables, columns, and indexes) for a table space.

```
RUNSTATS TABLESPACE(DSN8D61P.DSN8S61C) TABLE INDEX
```

Example 6: Update statistics used for access path selection. Update the catalog with *only* the statistics that are collected for access path selection. Report the collected statistics and route the statistics to SYSPRINT.

```
RUNSTATS TABLESPACE DSN8D61A.DSN8S61E
  REPORT YES
  UPDATE ACCESSPATH
```

Example 7: Update all statistics and generate report. Update the catalog with *all* the statistics (access path and space). Report the collected statistics and route the statistics to SYSPRINT.

```
RUNSTATS TABLESPACE DSN8D61A.DSN8S61E
  REPORT YES
  UPDATE ALL
```

Example 8: Report statistics without updating catalog. Do not update the catalog with the collected statistics. Report the collected statistics and route the statistics to SYSPRINT.

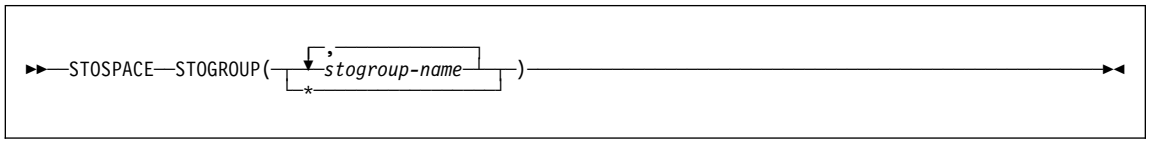
```
RUNSTATS TABLESPACE DSN8D61A.DSN8S61E
  REPORT YES
  UPDATE NONE
```

Example 9: Update statistics for a partition. Update the statistics for the table space and the partitioning index after a change to partition 1.

```
RUNSTATS TABLESPACE DSN8D61A.DSN8S61E PART 1 INDEX(DSN8610.XEMP1 PART 1)
```

STOSPACE

Syntax



Example

Example: Update catalog SPACE columns. Update the DB2 catalog SPACE columns for storage group DSN8G610.

```
//STEP1    EXEC DSNUPROC,UID='FUAUU330.STOSPCE',
//         UTPROC='',
//         SYSTEM='V61A'
//SYSIN    DD *
STOSPACE STOGROUP DSN8G610
//*
```

Stand-alone utility alphabetic reference

This section contains syntax diagrams and examples of DB2 stand-alone utilities organized alphabetically by name. For more information, see Section 3 of *DB2 Utility Guide and Reference*.

DSNJLOGF (Preformat Active Log)

Run DSNJLOGF as an MVS job. This is a sample of the JCL to invoke DSNJLOGF:

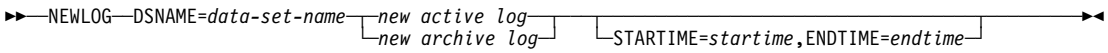
Sample control statement

```
//JOBLIB DD  DSN=DSN610.SDSNLOAD,DISP=SHR
//STEP1    EXEC PGM=DSNJLOGF
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//SYSUT1   DD DSN=DSNC610.LOGCOPY1.DS01,DISP=SHR
//STEP2    EXEC PGM=DSNJLOGF
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//SYSUT1   DD DSN=DSNC610.LOGCOPY1.DS02,DISP=SHR
//STEP3    EXEC PGM=DSNJLOGF
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//SYSUT1   DD DSN=DSNC610.LOGCOPY2.DS01,DISP=SHR
//STEP4    EXEC PGM=DSNJLOGF
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//SYSUT1   DD DSN=DSNC610.LOGCOPY2.DS02,DISP=SHR
```

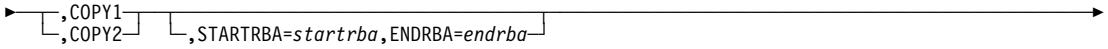
DSNJU003 (Change Log Inventory)

Syntax

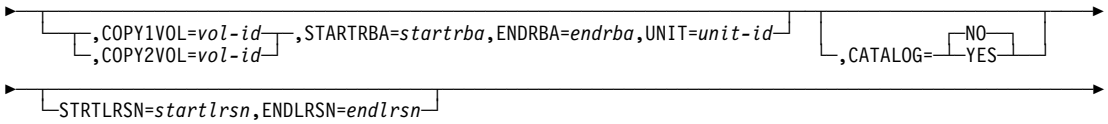
NEWLOG statement



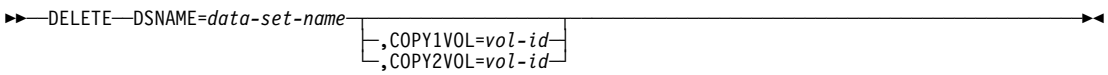
new active log:



new archive log:

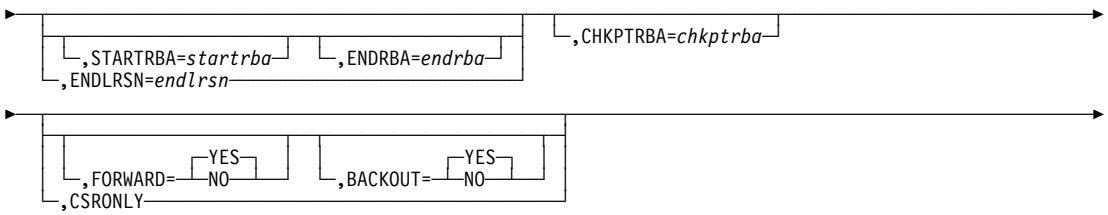


DELETE statement



CRESTART statement

create spec:

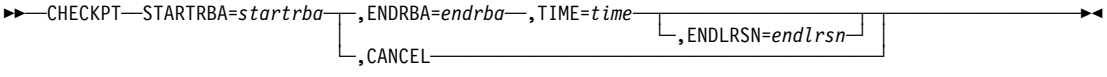


DSNJU003 (Change Log Inventory)

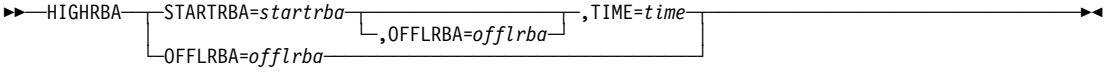
NEWCAT statement

DDF statement

CHECKPT statement



HIGHRBA statement



Examples

Example 1: Adding a new archive log data set

```
NEWLOG DSNAME=DSNREPAL.A0001187,COPY1VOL=DSNV04,UNIT=SYSDA,
STARTRBA=3A190000,ENDRBA=3A1F0000,CATALOG=NO
```

Example 2: Deleting a data set

```
DELETE DSNAME=DSNREPAL.A0001187,COPY1VOL=DSNV04
```

Example 3: Creating a new conditional restart record. The following statement creates a new conditional restart control record, specifying no backward-log recovery and log truncation (a new relative byte address for the end of the log).

```
CRESTART CREATE,BACKOUT=NO,ENDRBA=000000010000
```

Example 4: Adding a communication record to the BSDS

```
DDF LOCATION=USIBMSTODB22,LUNAME=STL#M08,PASSWORD=$STL@290
```

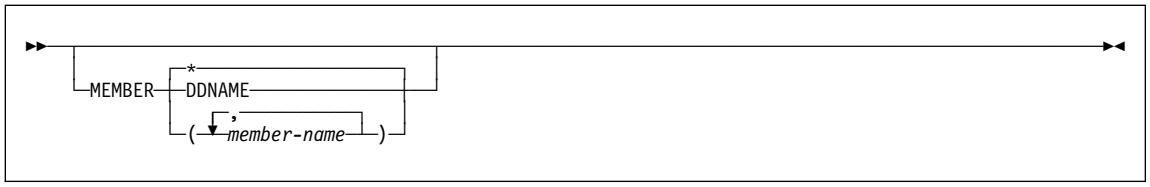
DSNJU004 (Print Log Map)

DSNJU004 (Print Log Map)

The following EXEC statement is used to invoke this utility:

```
//   EXEC   PGM=DSNJU004
```

Syntax



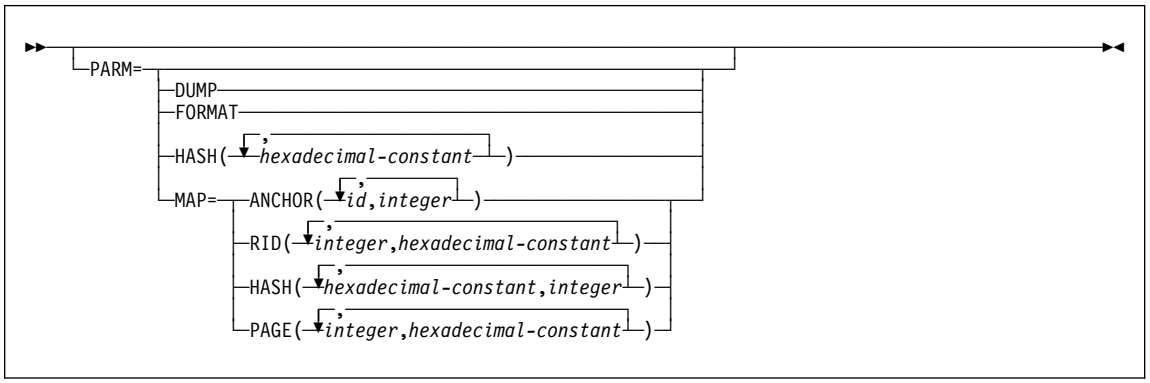
Example

The following statement prints information from the BSDS of each member in the data sharing group:

```
//PLM      EXEC PGM=DSNJU004
//SYSUT1   DD DSN=DBD1.BSDS01,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
           MEMBER *
```

DSN1CHKR

Syntax



Examples

Example 1: Running DSN1CHKR on a temporary data set. STEP1 allocates a temporary data set. STEP2 stops database DSNDB06 with the STOP DATABASE command. STEP3 copies the target table space into the temporary data set with DSN1COPY. The CHECK option is used to check the table space for page integrity errors. After DSN1COPY with the check option has ensured that no errors exist, STEP4 restarts the table space for access to DB2 again. STEP5 runs DSN1CHKR on the temporary data set.

DSN1CHKR prints the chains beginning at the pointers specified on the RID option of the MAP parameter. In this example, the first pointer is located on page 2, at an offset of 6 bytes from record 1, and the second pointer is located on page B, at an offset of 6 bytes from record 1.

The RIDs in STEP5 of the example are for example purposes only. Using them results in a error message. Change them to the actual RIDs to be checked.

DSN1CHKR

```
//YOUR JOBCARD
//*
//JOBCAT  DD DSNAME=DSNCAT1.USER.CATALOG,DISP=SHR
//STEP1    EXEC PGM=IDCAMS
//*******************************************************************
//*         ALLOCATE A TEMPORARY DATA SET FOR SYSDBASE               *
//*******************************************************************
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//SYSIN    DD *
   DELETE                                                 -
        (TESTCAT.DSNDBC.TEMPDB.TMPDBASE.I0001.A001)       -
          CATALOG(DSNCAT)
   DEFINE CLUSTER                                         -
        ( NAME(TESTCAT.DSNDBC.TEMPDB.TMPDBASE.I0001.A001) -
          NONINDEXED                                      -
          REUSE                                           -
          CONTROLINTERVALSIZE(4096)                       -
          VOLUMES(XTRA02)                                 -
          RECORDS(783 783)                                -
          RECORDSIZE(4089 4089)                           -
          SHAREOPTIONS(3 3) )                             -
   DATA                                                   -
        ( NAME(TESTCAT.DSNDBD.TEMPDB.TMPDBASE.I0001.A001))-
          CATALOG(DSNCAT)
/*
//STEP2    EXEC PGM=IKJEFT01,DYNAMNBR=20
//*******************************************************************
//*              STOP DSNDB06.SYSDBASE                              *
//*******************************************************************
//STEPLIB  DD DSN=prefix.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//SYSTSIN  DD *
 DSN SYSTEM(V61A)
     -STOP DB(DSNDB06) SPACENAM(SYSDBASE)
 END
/*
```

Figure 3 (Part 1 of 2). Sample JCL for running DSN1CHKR on a temporary data set

```
//STEP3    EXEC PGM=DSN1COPY,PARM=(CHECK)
//*******************************************************************
//*              CHECK SYSDBASE AND RUN DSN1COPY                    *
//*******************************************************************
//STEPLIB  DD DSN=prefix.SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSUT1   DD DSN=DSNCAT.DSNDBC.DSNDB06.SYSDBASE.I0001.A001,DISP=SHR
//SYSUT2   DD DSN=TESTCAT.DSNDBC.TEMPDB.TMPDBASE.I0001.A001,DISP=SHR
/*

//STEP4    EXEC PGM=IKJEFT01,DYNAMNBR=20
//*******************************************************************
//*              START DSNDB06.SYSDBASE                             *
//*******************************************************************
//STEPLIB  DD DSN=prefix.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//SYSTSIN  DD *
 DSN SYSTEM(V61A)
     -START DB(DSNDB06) SPACENAM(SYSDBASE)
 END
/*//STEP5    EXEC  PGM=DSN1CHKR,PARM='MAP=RID(00000201,06,00000B01,06)',
//         COND=(4,LT)
//*******************************************************************
//*              CHECK LINKS OF SYSDBASE                            *
//*******************************************************************
//STEPLIB  DD DSN=prefix.SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSUT1   DD DSN=TESTCAT.DSNDBC.TEMPDB.TMPDBASE.I0001.A001,DISP=SHR
/*
```

Figure 3 (Part 2 of 2). Sample JCL for running DSN1CHKR on a temporary data set

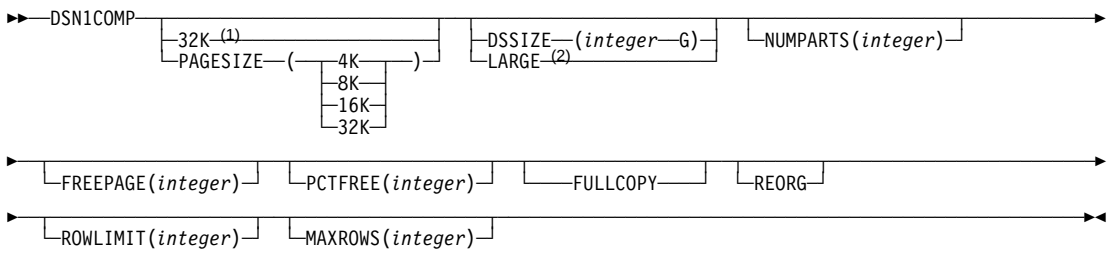
Example 2: Running DSN1CHKR on an actual table space. STEP1 stops database DSNDB06 with the STOP DATABASE command. STEP2 runs DSN1CHKR on the target table space; its output is identical to the output in Example 1. STEP3 restarts the database with the START DATABASE command.

```
//YOUR JOBCARD
//*
//STEP1    EXEC PGM=IKJEFT01,DYNAMNBR=20
//*********************************************************************
//*                        EXAMPLE 2                                  *
//*                                                                   *
//*                 STOP DSNDB06.SYSDBASE                             *
//*********************************************************************
//STEPLIB  DD DSN=prefix.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//SYSTSIN  DD *
 DSN SYSTEM(V61A)
     -STOP DB(DSNDB06) SPACENAM(SYSDBASE)
 END
/*

//STEP2    EXEC  PGM=DSN1CHKR,PARM='MAP=RID(00000201,06,00000B01,06)',
//         COND=(4,LT)
//*********************************************************************
//*              CHECK LINKS OF SYSDBASE                              *
//*********************************************************************
//STEPLIB  DD DSN=prefix.SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSUT1   DD DSN=DSNCAT.DSNDBD.DSNDB06.SYSDBASE.I0001.A001,DISP=SHR
/*
//STEP3    EXEC PGM=IKJEFT01,DYNAMNBR=20
//*********************************************************************
//*              RESTART DSNDB06.SYSDBASE                             *
//*********************************************************************
//STEPLIB  DD DSN=prefix.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//SYSTSIN  DD *
 DSN SYSTEM(V61A)
     -START DB(DSNDB06) SPACENAM(SYSDBASE)
 END
/*
```

Figure 4. Sample JCL for running DSN1CHKR on a stopped table space.

DSN1COMP

Syntax



Notes:

1 The preferred option is PAGESIZE(32K).

2 The preferred method of specifying LARGE is DSSIZE(4G).

Important: If you specify more than one parameter:

- Separate them by commas (no blanks).
- Specify them in any order.

Examples

Example 1: Running DSN1COMP

```
//jobname JOB acct info
//COMPEST  EXEC PGM=DSN1COMP,PARM='FULLCOPY'
//STEPLIB  DD DSN=prefix.SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSABEND DD SYSOUT=A
//SYSUT1   DD DSN=DSNCAT.DSNDBC.DB254A.TS254A.I0001.A001,DISP=SHR
```

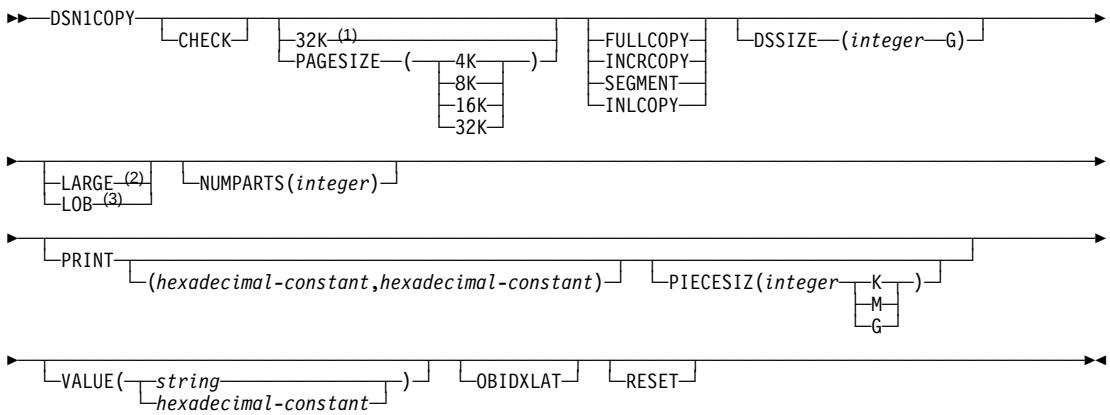
DSN1COMP

Example 2: Running DSN1COMP using the PCTFREE and FREEPAGE options

```
//DSN1COMP JOB  MSGLEVEL=(1,1),CLASS=A,MSGCLASS=A,REGION=3000K,
//         USER=SYSADM,PASSWORD=SYSADM
/*ROUTE PRINT STLXXXX.USERID
//STEP1    EXEC PGM=DSN1COMP,PARM='PCTFREE(20),FREEPAGE(5)'
//STEPLIB  DD DSN=prefix.SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSDUMP  DD SYSOUT=A
//SYSABEND DD SYSOUT=A
//SYSUT1   DD DSN=DSNC610.DSNDBD.DB254SP4.TS254SP4.I0001.A001,DISP=SHR
/*
//STEP2    EXEC PGM=DSN1COMP,PARM='ROWLIMIT(20000)'
//STEPLIB  DD DSN=prefix.SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSDUMP  DD SYSOUT=A
//SYSABEND DD SYSOUT=A
//SYSUT1   DD DSN=DSNC610.DSNDBD.DB254SP4.TS254SP4.I0001.A001,DISP=SHR
/*
//
```

DSN1COPY

Syntax



Notes:

1 The preferred option is PAGESIZE(32K).

2 The preferred method of specifying LARGE is DSSIZE(4G).

3 You cannot specify the SEGMENT and INLCOPY keywords with the LOB keyword.

Important: If you specify more than one parameter:

- Separate them by commas (no blanks).
- Specify them in any order.

Examples

Example 1: Running DSN1COPY with the CHECK option

```
//RUNCOPY EXEC PGM=DSN1COPY,PARM='CHECK'
//* COPY VSAM TO SEQUENTIAL AND CHECK PAGES
//STEPLIB  DD DSN=PDS CONTAINING DSN1COPY
//SYSPRINT DD SYSOUT=A
//SYSUT1   DD DSN=DSNCAT.DSNDBC.DSNDB01.SYSUTILX.I0001.A001,DISP=OLD
//SYSUT2   DD DSN=TAPE.DS,UNIT=TAPE,DISP=(NEW,KEEP),VOL=SER=UTLBAK
```

Example 2: Translating DB2 internal identifiers using the OBIDXLAT parameter

```
//EXECUTE EXEC PGM=DSN1COPY,PARM='OBIDXLAT'
//STEPLIB  DD  DSN=PDS CONTAINING DSN1COPY
//SYSPRINT DD  SYSOUT=A
//SYSUT1   DD  DSN=DSNC610.DSNDBC.DSN8D61P.DSN8S61C.I0001.A001,
//             DISP=OLD
//SYSUT2   DD  DSN=DSNC618.DSNDBC.DSN8D61P.DSN8S61C.I0001.A001,
//             DISP=OLD
//SYSXLAT  DD  *
260,280
2,10
3,55
6,56
7,57
/*
```

Example 3: Printing a single page of a partitioned table space

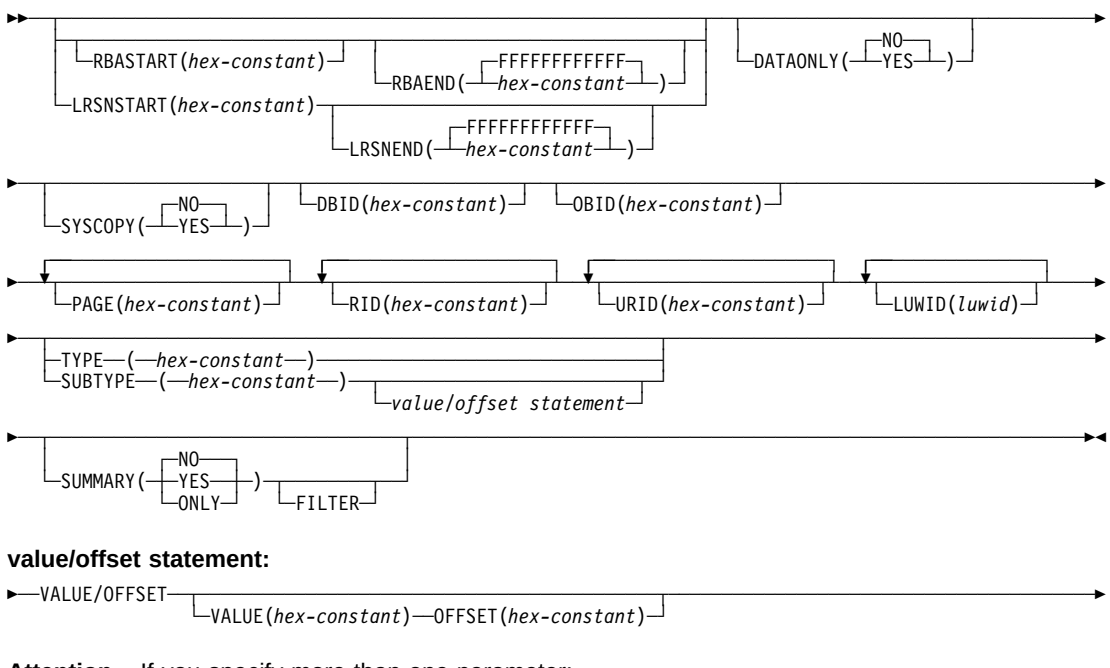
```
//PRINT EXEC PGM=DSN1COPY,PARM='PRINT(2002A1),NUMPARTS(8)'
//* PRINT A PAGE IN THE THIRD PARTITION OF A TABLE SPACE CONSISTING
//* OF 8 PARTITIONS.
//SYSUDUMP DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//SYSUT2   DD DUMMY
//SYSUT1   DD DSN=DSNCAT.DSNDBD.MMRDB.PARTEMP1.I0001.A003,DISP=OLD
```

Example 4: Printing 16 pages of a nonpartitioned index

```
//PRINT2 EXEC PGM=DSN1COPY,PARM=(PRINT(F0000,F000F),PIECESIZ(64M))
//* PRINT 16 PAGES IN THE 61ST PIECE OF AN NPI WITH PIECE SIZE OF 64M
//SYSUDUMP DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//SYSUT2    DD DUMMY
//SYSUT1    DD DISP=OLD,DSN=DSNCAT.DSTDBD.MMRDB.NPI1.I0001.A061
```

DSN1LOGP

Syntax



value/offset statement:

Attention: If you specify more than one parameter:

- Separate them by commas (no blanks)
- Specify them in any order.

Examples

Example 1: Using DSN1LOGP with an available BSDS. This example shows how to extract the information from the recovery log when you have the BSDS available. The extraction starts at the log RBA of X'AF000' and ends at the log RBA of X'B3000', for the table space or index space identified by the DBID of X'10A' (266 decimal) and the OBID of X'1F' (31 decimal).

DSN1LOGP

```
//STEP1 EXEC PGM=DSN1LOGP
//STEPLIB  DD DSN=PDS containing DSN1LOGP
//SYSPRINT DD SYSOUT=A
//SYSABEND DD SYSOUT=A
//BSDS     DD DSN=DSNCAT.BSDS01,DISP=SHR
//SYSIN    DD *
  RBASTART (AF000) RBAEND (B3000)
  DBID (10A) OBID(1F)
/*
```

You can think of the DB2 recovery log as a large sequential file. Whenever recovery log records are written, they are written to the end of the log. A log RBA is the address of a byte on the log. Because the recovery log is larger than a single data set, the recovery log is physically stored on many data sets. DB2 records the RBA ranges and their corresponding data sets in the BSDS. To determine which data set contains a specific RBA, read the information about the DSNJU004 utility and see Section 4 (Volume 1) of *DB2 Administration Guide*. During normal DB2 operation, messages are issued that include information about log RBAs.

Example 2: Using DSN1LOGP on the active log (no BSDS available). This example shows how to extract the information from the active log when the BSDS is not available. The extraction includes log records that apply to the table space or index space identified by the DBID of X'10A' and the OBID of X'1F'. The only information that is extracted is information relating to page numbers X'3B' and X'8C'. You can omit beginning and ending RBA values for ACTIVE*n* or ARCHIVE DD statements, because the DSN1LOGP search includes all specified ACTIVE*n* DD statements. The DD statements ACTIVE1, ACTIVE2, and ACTIVE3 specify the log data sets in ascending log RBA range. Use the DSNJU004 utility to determine what the log RBA range is for each active log data set. If the BSDS is not available and you cannot determine the ascending log RBA order of the data sets, you must run each log data set individually.

```
//STEP1 EXEC PGM=DSN1LOGP
//STEPLIB  DD DSN=PDS containing DSN1LOGP
//SYSPRINT DD SYSOUT=A
//SYSABEND DD SYSOUT=A
//ACTIVE1  DD DSN=DSNCAT.LOGCOPY1.DS02,DISP=SHR      RBA X'A000' - X'BFFF'
//ACTIVE2  DD DSN=DSNCAT.LOGCOPY1.DS03,DISP=SHR      RBA X'C000' - X'EFFF'
//ACTIVE3  DD DSN=DSNCAT.LOGCOPY1.DS01,DISP=SHR      RBA X'F000' - X'12FFF'
//SYSIN    DD *
  DBID (10A) OBID(1F) PAGE(3B) PAGE(8C)
/*
```

Example 3: Using DSN1LOGP on archive log data (no BSDS available). This example shows how to extract the information from archive logs when the BSDS is not available. The extraction includes log records that apply to a single unit of recovery (whose URID is X'61F321'). Because the BEGIN UR is the first record for the unit of recovery and is at X'61F321', the beginning RBA is specified to indicate that it is the first RBA in the range from which to extract recovery log records. Also, because no ending RBA value is specified, all specified archive logs are scanned for qualifying log records. The specification of DBID(4) limits the scan to changes that the specified unit

of recovery made to all table spaces and index spaces in the database whose DBID is X'4'.

```
//STEP1 EXEC PGM=DSN1LOGP
//STEPLIB  DD DSN=PDS containing DSN1LOGP
//SYSPRINT DD SYSOUT=A
//SYSABEND DD SYSOUT=A
//ARCHIVE  DD DSN=DSNCAT.ARCHLOG1.A0000037,UNIT=TAPE,VOL=SER=T10067,
//            DISP=(OLD,KEEP),LABEL=(2,SL)
//         DD DSN=DSNCAT.ARCHLOG1.A0000039,UNIT=TAPE,VOL=SER=T30897,
//            DISP=(OLD,KEEP),LABEL=(2,SL)
//         DD DSN=DSNCAT.ARCHLOG1.A0000041,UNIT=TAPE,VOL=SER=T06573,
//            DISP=(OLD,KEEP),LABEL=(2,SL)
//SYSIN    DD *
  RBASTART (61F321)
  URID (61F321) DBID(4)
/*
```

Example 4: Using DSN1LOGP with the SUMMARY option. The DSN1LOGP SUMMARY option allows you to scan the recovery log to determine what work is incomplete at restart time. You can specify this option either by itself or when you use DSN1LOGP to produce a detail report of log data. Summary log results appear in SYSSUMRY; therefore, you must include a SYSSUMRY DD statement as part of the JCL with which you execute DSN1LOGP.

This example produces both a detail and a summary report using the BSDS to identify the log data sets. The summary report summarizes all recovery log information within the RBASTART and RBAEND specifications. You cannot limit the output of the summary report with any of the other options, except by using the FILTER option with a URID or LUWID specification. RBASTART and RBAEND specification use depends on whether a BSDS is used.

This example is similar to Example 1, in that it shows how to extract the information from the recovery log when you have the BSDS available. However, this example also shows you how to specify a summary report of all logged information between the log RBA of X'AF000' and the log RBA of X'B3000'. This summary is generated with a detail report, but will be printed to SYSSUMRY separately.

```
//STEP1 EXEC PGM=DSN1LOGP
//STEPLIB  DD DSN=PDS containing DSN1LOGP
//SYSPRINT DD SYSOUT=A
//SYSSUMRY DD SYSOUT=A
//SYSABEND DD SYSOUT=A
//BSDS     DD DSN=DSNCAT.BSDS01,DISP=SHR
//SYSIN    DD *
  RBASTART (AF000) RBAEND (B3000)
  DBID (10A) OBID(1F) SUMMARY(YES)
/*
```

Example 5: Data sharing— using DSN1LOGP on all members of a data sharing group. This example shows extract log information pertaining to the table space identified by DBID X'112' and OBID X'1D', from all members of a data sharing group.

DSN1LOGP

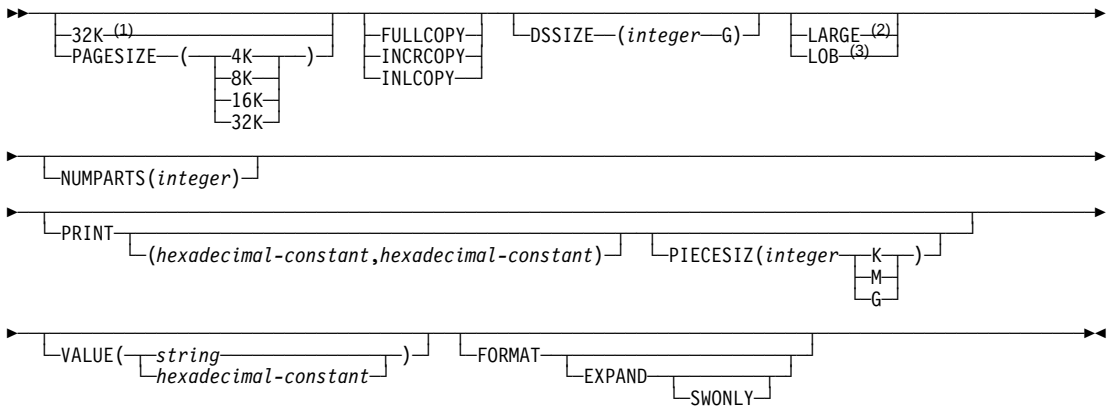
```
//STEP1 EXEC PGM=DSN1LOGP
//STEPLIB DD  DSN=PDS containing DSN1LOGP
//SYSPRINT SYSOUT=A
//SYSABEND SYSOUT=A
//GROUP DD DSN=DSNDB0G.BSDS01,DISP=SHR
//SYSIN DD *
  DATAONLY (YES)
  LRSNSTART (A7951A001AD5) LRSNEND (A7951A003B6A)
  DBID (112) OBID(1D)
/*
```

Example 6: Data sharing— using DSN1LOGP on a single member of a data sharing group. This example shows extract log information pertaining to the table space identified by DBID X'112' and OBID X'1D', from a single member of a data sharing group.

```
//STEP1 EXEC PGM=DSN1LOGP
//STEPLIB DD  DSN=PDS containing DSN1LOGP
//SYSPRINT SYSOUT=A
//SYSABEND SYSOUT=A
//M01BSDS DD DSN=DSNDB0G.DB1G.BSDS01,DISP=SHR
//SYSIN DD *
  DATAONLY (YES)
  LRSNSTART (A7951A001AD5) LRSNEND (A7951A003B6A)
  DBID (112) OBID(1D)
/*
```

DSN1PRNT

Syntax



Notes:

1 The preferred option is PAGESIZE(32K).

2 The preferred method of specifying LARGE is DSSIZE(4G).

3 Do not specify INLCOPY with the LOB parameter.

Important: If you specify more than one parameter:

- Separate them by commas (no blanks).
- Specify them in any order.

Examples

Example 1: Running DSN1PRNT

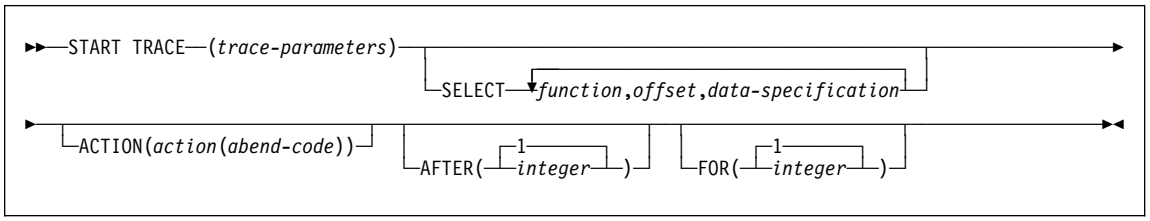
```
//jobname JOB acct info
//RUNPRNT EXEC PGM=DSN1PRNT,PARM='PRINT,FORMAT'
//STEPLIB DD DSN=prefix.SDSNLOAD
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=DSNCAT.DSNDBC.DSNDB01.SYSUTIL.I0001.A001,DISP=SHR
```

Example 2: Printing a nonpartitioned index with a 64 MB piece size

```
//PRINT2 EXEC PGM=DSN1PRNT,
//            PARM=(PRINT(F0000,F000F),FORMAT,PIECESIZ(64M))
//* PRINT 16 PAGES IN THE 61ST PIECE OF AN NPI WITH PIECE SIZE OF 64M
//SYSUDUMP DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//SYSUT1   DD DISP=OLD,DSN=DSNCAT.DSNDBD.MMRDB.NPI1.I0001.A061
```

DSN1SDMP

Syntax



Examples

Example 1: Skeleton JCL for DSN1SDMP

DSN1SDMP

```
//DSN1J018 JOB 'IFC SD',CLASS=A,
//         MSGLEVEL=(1,1),USER=SYSADM,PASSWORD=SYSADM,REGION=1024K
//*******************************************************************
//*
//*       THIS IS A SKELETON OF THE JCL USED TO RUN DSN1SDMP.
//*       YOU MUST INSERT SDMPIN DD.
//*
//*******************************************************************
//IFCSD    EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//STEPLIB  DD DISP=SHR,DSN=prefix.SDSNLOAD
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SDMPPRNT DD SYSOUT=*
//SDMPTRAC DD DISP=(NEW,CATLG,CATLG),DSN=IFCSD.TRACE,
//            UNIT=SYSDA,SPACE=(8192,(100,100)),DCB=(DSORG=PS,
//            LRECL=8188,RECFM=VB,BLKSIZE=8192)
//SDMPIN   DD *
//*******************************************************************
//*
//*       INSERT SDMPIN DD HERE.  IT MUST BEGIN WITH A VALID
//*       START TRACE COMMAND (WITHOUT THE SUBSYSTEM RECOGNITION CHAR)
//*
//*******************************************************************

                  (VALID SDMPIN GOES HERE)

/*
//*******************************************************************
//SYSUDUMP DD SYSOUT=*
//SYSTSIN  DD *
 DSN SYSTEM(DSN)
 RUN PROG(DSN1SDMP) PLAN(DSNEDCL)
 END
//*
```

Example 2: SDMPIN for ABEND and TERMINATE AGENT on -904 SQL CODE

```
//SDMPIN   DD *
* START ONLY IFCID 58, END SQL STATEMENT
 START TRACE=P CLASS(32) IFCID(58) DEST(OPX)
 FOR(1)
 ACTION(ABENDTER(00E60188))
 SELECT
* OFFSET TO FIRST DATA SECTION CONTAINING THE SQLCA.
 P4,08
* SQLCODE -904, RESOURCE UNAVAILABLE
 DR,74,X'FFFFFC78'
/*
```

Example 3: SDMPIN for ABEND and RETRY on RMID 20

```
//*       ABEND AND RETRY AN AGENT WHEN EVENT ID X'0025'
//*       (AGENT ALLOCATION) IS RECORDED BY RMID 20 (SERVICE
//*       CONTROLLER).
//*
//SDMPIN   DD *
*  ENSURE ONLY THE TRACE HEADER IS APPENDED WITH THE STANDARD HEADER
*  VIA THE TDATA KEYWORD ON START TRACE
 START TRACE=P CLASS(3,8) RMID(20) DEST(OPX) TDATA(TRA)
*  ABEND AND RETRY THE AGENT WITH THE DEFAULT ABEND CODE (00E60100)
 ACTION(ABENDRET)
*  SPECIFY THE SELECT CRITERIA FOR RMID.EID
 SELECT
*  OFFSET TO THE STANDARD HEADER
 P4,00
*  ADD LENGTH OF STANDARD HEADER TO GET TO TRACE HEADER
 LN,00
*  LOOK FOR EID 37 AT OFFSET 4 IN THE TRACE HEADER
 DR,04,X'0025'
/*
```

Example 4: Dump on SQLCODE -811 RMID16 IFCID 58

```
//SDMPIN   DD *
 START TRACE=P CLASS(3) RMID(22) DEST(SMF) TDATA(COR,TRA)
 AFTER(1)
 FOR(1)
 SELECT
* POSITION TO HEADERS (QWHS IS ALWAYS FIRST)
  P4,00
* CHECK QWHS 01, FOR RMID 16, IFCID 58
  DR,02,X'0116003A'
* POSITION TO SECOND SECTION (1ST DATA SECTION)
  P4,08
* COMPARE SQLCODE FOR 811
  DR,74,X'FFFFFCD5'
 ACTION(ABENDRET(00E60188))
/*
```

DSN1SDMP

Appendix A. Resource types for resource codes in DB2 messages

Refer to Table 3 for a list of -904 resource unavailable reason codes.

Table 3 (Page 1 of 2). Resource Types

TYPE Code	Type of Resource	Name, Content, Format
00000100	Database	DB
00000200	Table space	DB.SP
00000201	Index space	DB.SP
00000202	Table space	RD.DB.TS
00000205	Compression Dictionary	DB.SP
00000210	Partition	DB.SP.PT
00000220	Data set	DSN
00000230	Temporary file	SZ
00000240	Database procedure	DBP
00000300	Page	DB.SP.PG
00000301	Index minipage	DB.SP.PG.MP
00000302	Table space page	DB.SP.PG
00000303	Index space page	DB.SP.PG
00000304	Table space RID	DB.SP.RID
00000305	Index access/table space RID	DB.SP.RID
00000306	Index access/table space page	DB.SP.PG
00000307	Index space EOF	DB.SP.01
00000400	ICF catalog	IC
00000500	Storage group	SG
00000600	EDM pool space	
00000601	EDM data space	
00000700	Buffer pool space	BP
00000701	Group buffer pool	GBP
00000800	Plan	PL
00000801	Package	COLLECTION. PACKAGE. CONTOKEN
00000802	BINDLOCK	BINDLOCK
00000900	32KB data area	
00000901	Sort storage	
00000903	Hash anchor	DB.SP.PG.AI
00000904	RIDLIST storage	
00000905	IRLM storage	
00000906	DB2	MEMBER
00000907	Data Space	MEMBER
00000A00	Table	RD.CR.TB
00000A10	Alias	RELDEP. OWNER. ALIAS
00000B00	View	RD.CR.VW
00000C00	Index	RD.CR.IX
00000C01	Index	CR.IX
00000D00	DBID/OBID	RD.DI.OI
00000D01	DBID/OBID	DI.OI
00000D02	OBID	OI

Resource types

Table 3 (Page 2 of 2). Resource Types

TYPE Code	Type of Resource	Name, Content, Format
00000E00	SU limit exceeded	CN
00000F00	Auxiliary column	DI
00000F01	LOB lock	DIX.PIX.ROWID.VRSN
00001000	DDF	LOCATION
00001001	System conversation	LU.MODE. RTNCD. FDBK2. RCPRI. RCSEC. SENSE
00001002	Agent conversation	LU.MODE. RTNCD. FDBK2. RCPRI. RCSEC. SENSE
00001003	CNOS processing	LU. MODE. RTNCD. FDBK2. RCPRI. RCSEC. SENSE
00001004	CDB (Communication database)	LOCATION. AUTHORIZATION ID.PL
00001005	DB access agent	LOCATION
00001007	TCP/IP domain name	LINKNAME.DOMAIN.ERRNO
00001008	TCP/IP service name	LOCATION.SERVICE.ERRNO
00001102	Bootstrap data set (BSDS)	MEMBER
00002000	Table space CS-claim class	DB.SP
00002001	Table space RR-claim class	DB.SP
00002002	Table space write-claim class	DB.SP
00002003	Index space CS-claim class	DB.SP
00002004	Index space RR-claim class	DB.SP
00002005	Index space write-claim class	DB.SP
00002006	Table space partition CS-claim class	DB.SP.PT
00002007	Table space partition RR-claim class	DB.SP.PT
00002008	Table space partition write-claim class	DB.SP.PT
00002009	Index space partition CS-claim class	DB.SP.PT
00002010	Index space partition RR-claim class	DB.SP.PT
00002011	Index space partition Write-claim class	DB.SP.PT
00002100	Table space DBET entry	DB.SP
00002101	Index space DBET entry	DB.SP
00002102	Table space partition DBET entry	DB.SP.PT
00002103	Index space partition DBET entry	DB.SP.PT
00002104	Logical partition DBET entry	DB.SP.PT
00002200	Routine Parameter Storage	DB.SP.PT
A10X	Alias	RD.CR.AL

Resource types

Where	Stands for	Where	Stands for
AI	Hash anchor ID	OI	OBID in decimal of resource
ALIAS	Alias owner	OWNER	Alias owner
AUTHORIZATION ID	DB2 authorization identifier	PACKAGE	Package identifier
BP	Buffer pool identifier	PG	Hexadecimal page number
CN	Column name	COLN	Column number within the base table of the LOB column that has been marked invalid.
DIX	DBID in hexadecimal	PL	Plan identifier
COLLECTION	Collection-ID of the package	PT	Decimal partition number
CONTOKEN	Consistency token of the package	RCPRI	APPC primary return code
CR	Creator of the object	RCSEC	APPC secondary return code
DB	Database name	RD	DB2 release dependency mark
DBP	Database procedure name	RELDEP	DB2 release dependency mark
DI	DBID in decimal of resource	RID	Record Identifier
DSN	Data set name	RTNCD	VTAM primary return code
FDBK2	VTAM secondary return code	MEMBER	Group member name
GBP	Group buffer pool name	SENSE	SNA sense codename
IC	ICF catalog alias name	SG	Storage group name
IX	Index name	SP	Space name
LOCATION	Location in which the specified resource is not available	SZ	Temporary file page size
LU	Logical unit name	TB	Table name
MODE	Logical unit mode name	TS	Table space name
MP	Hexadecimal mini-page number	VW	View name

Resource types

Appendix B. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J74/G4
555 Bailey Avenue
P.O. Box 49023
San Jose, CA 95161-9023
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Programming Interface Information

This book is intended to help you to code SQL statements, commands, and utility control statements. This book primarily documents General-use Programming Interface and Associated Guidance Information provided by IBM DATABASE 2 Universal Database Server for OS/390 (DB2 for OS/390).

General-use programming interfaces allow the customer to write programs that obtain the services of DB2 for OS/390.

However, this book also documents Product-sensitive Programming Interface and Associated Guidance Information.

Product-sensitive programming interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of this IBM software product. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-sensitive programming interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

Product-sensitive Programming Interface and Associated Guidance Information is identified where it occurs, by an introductory statement to a section.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, or other countries, or both:

CICS
DATABASE 2
DB2
DB2 Universal Database
DFSMS
Distributed Relational Database Architecture
DRDA
IBM
IMS
Language Environment
OS/390
QMF
SQL/DS
VTAM

Throughout the library, the DB2 for OS/390 licensed program and a particular DB2 for OS/390 subsystem are each referred to as “DB2.” In each case, the context makes the meaning clear.

The following terms are trademarks of other companies as follows:

- Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.
- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft™ Corporation in the United States and/or other countries.

Other company, product, or service names may be trademarks or service marks of others.

How to send your comments

DB2 Universal Database for OS/390
Reference Summary
Version 6

Publication No. SX26-3844-00

Your feedback helps IBM to provide quality information. Please send any comments that you have about this book or other DB2 for OS/390 documentation. You can use any of the following methods to provide comments.

- Send your comments from the Web. Visit the DB2 for OS/390 Web site at:

  http://www.software.ibm.com/data/db2/os390

  The Web site has a feedback page that you can use to send comments.
- Send your comments by e-mail to db2pubs@vnet.ibm.com and include the name of the product, the version number of the product the number of the book. If you are commenting on specific text, please list the location of the text (for example, a chapter and section title, page number, or a help topic title).
- Complete the readers' comment form at the back of the book and return it by mail, by fax (800-426-7773 for the United States and Canada), or by giving it to an IBM representative.

Readers' Comments

DB2 Universal Database for OS/390
Reference Summary
Version 6

Publication No. SX26-3844-00

How satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Technically accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐
Grammatically correct and consistent	☐	☐	☐	☐	☐
Graphically well designed	☐	☐	☐	☐	☐
Overall satisfaction	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

May we contact you to discuss your comments? ☐ Yes ☐ No

Name _____ Address _____

Company or Organization _____

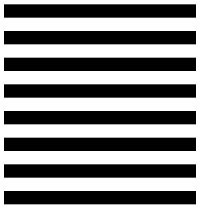
Phone No. _____

Readers' Comments
SX26-3844-00

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department BWE/H3
PO Box 49023
San Jose, CA 95161-9945

Fold and Tape **Please do not staple** Fold and Tape

SX26-3844-00

Cut or Fold
Along Line

Program Number: 5645-DB2

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SX26-3844-00