Query Management Facility

QMF Reference

Version 6

SC26-9577-00

Note!
Before using this information and the product it supports, be sure to read the general information under "Appendix D. Notices" on page 319.

Fifth Edition (December 1998)

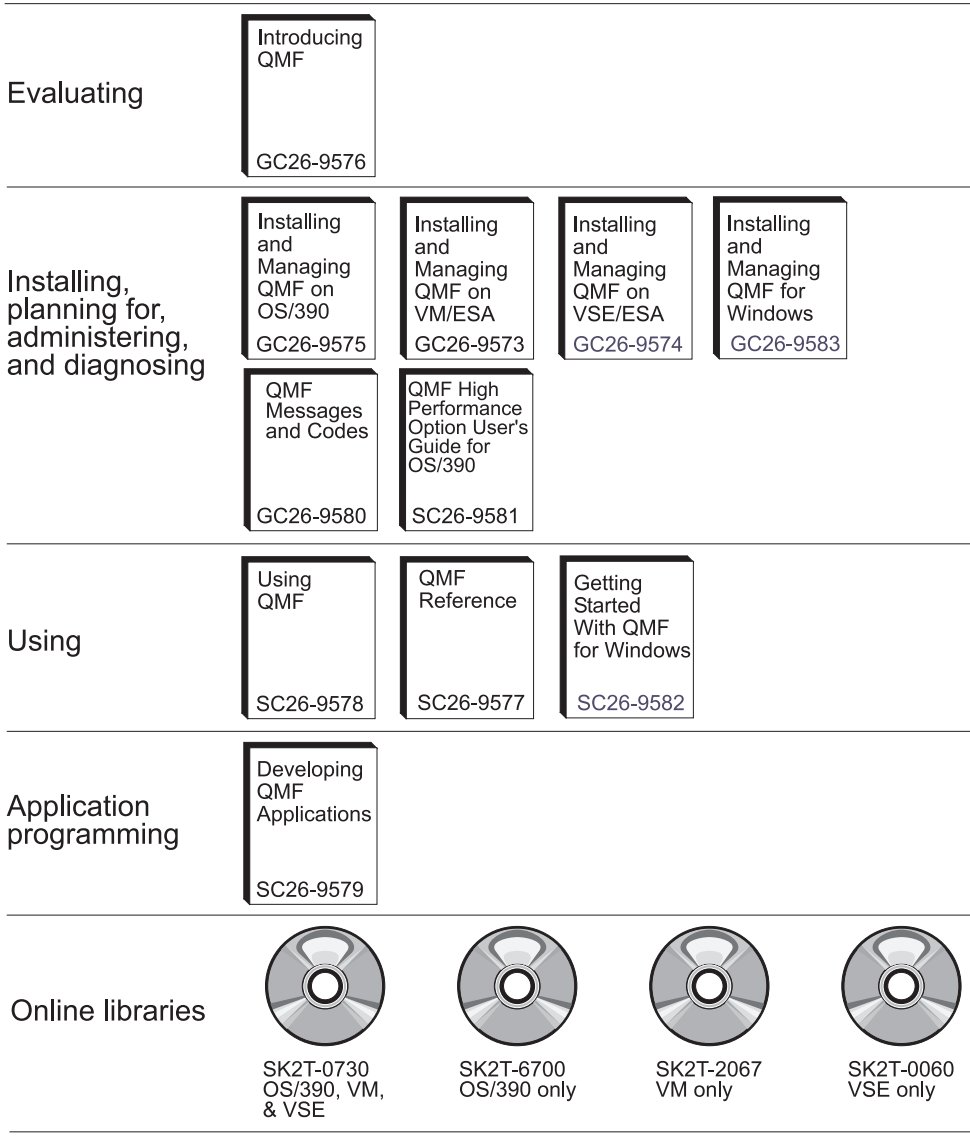
This edition applies to Query Management Facility, a feature of Version 6 of DB2 Universal Database Server for OS/390 (DB2 UDB for OS/390), 5645-DB2, and of Query Management Facility, a feature of Version 6 of DATABASE 2 Server for VM and VSE, (DB2 for VM and VSE), 5648-A70, and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces and makes obsolete the previous edition, SC26-4716-04. The technical changes for this edition are indicated by a vertical bar to the left of the change. A vertical bar to the left of figure caption indicates that the figure has changed. Editorial changes that have no technical significance are not noted.

Contents

The QMF Library

You can order manuals either through an IBM representative or by calling 1-800-879-2755 in the United States or any of its territories.



About This Book

This book is for people who have experience with the Query Management Facility product (QMF). The main topics are:

- QMF Commands
- SQL keywords used in QMF queries
- Forms, reports, and charts (including usage and edit codes)

Commands, keywords, and forms are presented in alphabetic order in their respective chapters.

The appendixes contain QMF sample tables, a list of global variables, information on QMF control tables, and QMF's support requirements for different environments. The book also includes a bibliography and a glossary.

Prerequisite Knowledge

The book *Using QMF* contains basic QMF information. Knowledge of the concepts in that guide is assumed in this reference book. In addition to the steps necessary to get started with QMF and how to use SQL queries, *Using QMF* contains detailed scenarios showing how to build queries and forms step by step. It also contains information about Query-By-Example.

A complete list of QMF publications is in "The QMF Library" on page vii. QMF publications can be obtained through your IBM representative or by calling 1-800-879-2755 in the United States or its territories.

How to Send Your Comments

Your feedback is important in helping to provide the most accurate and high-quality information.

Send your comments from the Web

Visit the Web site at:

`http://www.ibm.com./qmf`

The Web site has a feedback page that you can use to enter and send comments.

Send your comments by e-mail

to comments@vnet.ibm.com. Be sure to include the name of the

product, the version number of the product, the name and part number of the book (if applicable). If you are commenting on specific text, please include the location of the text (for example, a chapter and section title, a table number, a 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.

How to Order QMF Books

You can order QMF documentation either through an IBM representative or by calling 1-800-879-2755 in the United States or any of its territories.

For a list of QMF books, see "The QMF Library" on page vii.

Chapter 1. QMF Commands

This chapter contains the following information:

QMF Command Environments

You can enter QMF commands from TSO, CMS, or CICS environments. In TSO or CMS, you might also be using ISPF. In a small table at the beginning of each command description, an X indicates which environments accept the command. An asterisk (*) indicates that only certain aspects of the command are accepted. For example:

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	*

Entering Commands

You can issue QMF commands in several ways:

- On the command line
- With a function key
- On a prompt panel
- From a procedure
- From an application

If your installation has defined a command synonym with the same name as a QMF command, you must precede the command with `QMF` to override the synonym.

On the Command Line

Where a command line appears, you can enter any QMF command by typing it in full after the arrow. For example,

```
COMMAND ===> RUN MYQUERY (FORM=FORM2
```

To run the command, press Enter.

With a Function Key

You can enter some commands using function keys. QMF has a default set of function keys for each panel. The function keys that you see when you use QMF can differ from the defaults. This book refers to the default set.

To use parameters with a function key command, type the parameters on the command line, and then press the function key. For example, when the query panel is displayed, type (FORM=FORM2, and then press the Run function key. This command is run:

```
RUN QUERY (FORM=FORM2
```

On a Prompt Panel

QMF displays a command prompt panel if you enter a command with a syntactical error or a misspelling twice in succession, or when you enter the command name followed by a question mark on the command line. This prompt panel is useful when entering long commands.

For example, when you enter RUN ? the following command prompt panel is displayed on which you can enter the required information:

```
                    RUN Command Prompt
                                                1  to 8 of 8
Type (____________________)

Name (_____________________________________________)+
     To run an object from temporary storage, enter its type:
     QUERY or PROC.

     To run an object from the database, enter its name (and
     optionally its type).  Type can be QUERY or PROC.
_____________________________________________________________
F1=Help  F3=End  F4=List  F7=Backward  F8=Forward
```

If QMF needs additional information to complete a command, a second panel prompts you for command parameters.

You can skip the first panel of this two-step prompt by entering the command, the object type, and the object name, followed by a question mark on the command line. A panel appears containing the parameters that are applicable to that object.

A question mark is not valid in the parameters portion of a command (after the left parenthesis). Also, any parameters following the question mark are ignored. For example, (FORM=FORM2 is ignored in the following command:

```
RUN QUERY MYQUERY ? (FORM=FORM2
```

These three function keys are contained on most prompt panels:

Help Displays help information about the message being displayed at the bottom of the screen.

List Displays a list of objects from which you can select.

End Returns to the panel from which the prompt was issued.

From a Procedure

You can include almost any QMF command as a line in a procedure, including a RUN command that runs the same or another procedure. This is helpful when using commands that are too long to enter on the command line.

When you put commands into a procedure, use the full command names, parameters, and values rather than the abbreviations. The minimum acceptable abbreviation for an existing word might change in future releases and cause your procedure to fail.

Commands in Procedures with Logic

When you use QMF commands in a procedure with logic, the commands:

- *Must* be in uppercase, regardless of the profile setting.
- Can be continued by ending the line with a comma.
- Can contain substitution variables.

Commands in Linear Procedures

Commands in linear procedures can be continued over more than one line by placing a plus sign (+) as a continuation character in column 1 of each additional line. The continued line then starts in column 3.

For more information on using commands in both types of procedures, see “Procedures” on page 277.

From an Application

QMF commands within applications must be entered in uppercase, regardless of the profile setting.

Note to CICS users
The command interface is not available in CICS, as its function depends on ISPF.

The Command Interface
This interface receives QMF commands from ISPF. QMF must be started before the application, EXEC, or CLIST is run.

The Callable Interface
Receives QMF commands directly from QMF's Systems Application Architecture. (SAA) common programming interface (CPI). You can start and stop QMF from your application. ISPF is not required.

For detailed information on using commands within applications, see *Developing QMF Applications.*

Using Remote Data Access

When issuing commands using distributed unit of work or remote unit of work:

- References to tables and views apply to the current location, unless a three-part name or alias is used to refer to a different location.
- References to QMF procedures, queries, and forms in the database apply to the current location. You cannot refer to a procedure, query, or form with a three-part name.
- Data sets or files named in QMF commands must reside at the system on which QMF is executing.
- CICS data queues named in QMF commands must be defined at the system on which QMF is executing.
- References to stored profile values apply to the current location, except for the TRACE parameter. See *Managing QMF* for your operating system for more information on TRACE.
- When QMF is running in CICS/MVS, all database objects (tables, views, procedures, queries, and forms) at remote DB2 locations are read only.
- If you are running QMF in CICS/VSE, remote data access is not available.

Confirmation Panels

If there is a CONFIRM parameter on a command, you can specify YES or NO (or use the default in your profile). If the command would modify the database and the CONFIRM parameter is YES, a confirmation panel like the following is displayed:

```
                      RUN CONFIRMATION

WARNING:
Your RUN command will modify this number of rows in the
database:         1

Do you want to make this change?
1 1. YES - Make the changes permanent in the database.
  2. NO  - Restore the table to what it was before the query
     was run; make no changes.
```

Many QMF confirmation panels for changes to the database are actually prompting you to do a commit (by entering YES to keep the changes) or a rollback (by entering NO).

Because the changes were already made to the database, the database manager holds locks on the data until you reply YES or NO on the confirmation panel.

If you are using SQL/DS, the tables you are working with might be in a nonrecoverable dbspace. If so, any changes you make are committed to the database immediately, and you cannot execute a rollback. Therefore, if a table is in a nonrecoverable dbspace, specifying NO on the confirmation panel really doesn't prevent the changes from taking place.

For more information on dbspace, contact your database administrator or see *DB2 Server for VM System Administration*

Canceling Commands

The method you can use to cancel a QMF command or query that is currently in process depends on the type of terminal connection you have and your environment.

In CMS and TSO:

- If your terminal is connected directly to the system, press the Reset key and then the PA1 key.
- If your terminal is connected to an SNA network, press the ATTN key.

In CICS, the CICS operator must cancel the QMF transaction like any other CICS transaction. You cannot use the PA1 and ATTN keys in CICS. When a QMF transaction is canceled, all work is lost and the QMF environment is deleted.

Command Syntax

Read QMF syntax diagrams from left to right, top to bottom.

```
►►──────  The start of the diagram
──────►   The diagram is continued
►──────   The continuation of the diagram
──────►◄  The end of the diagram
```

Commands are always on the main path in the diagram. The minimum abbreviation for commands and parameters is shown in uppercase. If there are no parameters associated with a command, the syntax diagram looks like this:

```
►►─RETrieve──────────────────────────────────────────────────────►◄
```

Command Parameters

A command can allow two types of parameters. *Positional parameters* must be placed in a certain position within a command. *Keyword parameters* are assigned a value and can be placed in any order within a command. The first keyword parameter used in a command must be preceded by a left parenthesis.

If a command allows keyword parameters, you can use as many as you need. If you use a keyword parameter more than once in one command and provide different values for the parameter, its last value takes effect. No parameter can be longer than 80 characters.

All parameters are separated from each other with a blank, a comma followed by a blank, or a comma *not* followed by a blank (if you specified DECIMAL=PERIOD in your profile). For example, all of the following specifications are correct:

```
(MEMBER=member CONFIRM=YES
(MEMBER=member, CONFIRM=YES
(MEMBER=member,CONFIRM=YES
(MEMBER member CONFIRM=YES
(MEMBER member CONFIRM YES
```

A right parenthesis is not required, but can be used to end the command. Anything you put after it is treated as a comment; it is not processed.

Required Parameters

Parameters are on the main path if they are required. Parameter values follow an equal sign and are shown in lowercase italics.

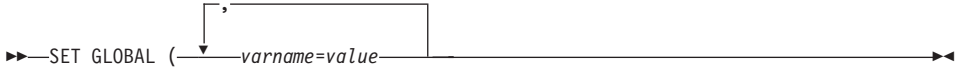
When one parameter is on the main path with others listed below, you *must* choose at least one from the list.

Optional Parameters

When a parameter is shown below the main path, it is optional. When all parameters are shown in a list below the main path, you can specify any one of them or none.

Default parameter values are shown above the path.

When a parameter can be repeated, you must separate each repetition of the parameter with a comma, like this:



Sometimes values are separated by an “or” sign (|):

ADD

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The ADD command adds rows to a table in the Table Editor and adds global variables to the global variable list.

Use the Add function key when in Add mode of the Table Editor to add rows to a table. Transactions are saved either immediately or when you end your Table Editor session. Use the SAVE parameter on the EDIT command to specify when you want transactions saved.

In the global variable list, use the Add function key or enter ADD on the command line to display the Add Variable panel so you can add a new variable.

BACKWARD

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The BACKWARD command scrolls toward the top of a panel or to the first field of the current row in the Table Editor.

Csr Scrolls the line where the cursor is positioned to the bottom of the scrollable area.

Half Scrolls backward half the depth of the scrollable area or to the top if that is nearer.

Max Scrolls to the top of the scrollable area. BACKWARD Max is equivalent to TOP.

Page Scrolls backward the depth of the scrollable area or to the top if that is closer.

n Scrolls backward *n* lines. (*n* can be any number from 1 through 9999.)

You can scroll backward until the first line is at the top of your screen.

You can scroll backward as many lines as you like: to the cursor position, a half page, all the way back, a whole page, or any number of lines (*n*) up to 9999.

If you don't specify an amount, the amount used is that shown after SCROLL ===> in the bottom right corner of the panel. You can change that amount by typing a new amount over it. The change remains in effect throughout the session, except for MAX, which remains in effect for only the next command.

You can also change the scroll amount QMF uses by setting the global variable DSQDC_SCROLL_AMT to Csr, Half, Page, or any number of lines (*n*) up to 9999.

To scroll backward in the footing text on form panels, position the cursor on the portion of the panel where the footing text is specified and press the Backward function key.

BATCH

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X		X		

BATCH is a QMF-supplied command synonym that accesses the batch query or procedure application. You must be authorized to use BATCH; consult your information center.

This application lets you run queries and procedures as QMF batch jobs rather than interactively.

For additional information, see *Installing and Managing QMF for VM/ESA* or *Installing and Managing QMF for MVS* .

BOTTOM

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

BOTTOM scrolls to the last line of queries, procedures, reports, global variable lists, and scrollable form panels. BOTTOM is equivalent to FORWARD Max.

To scroll to the bottom of footing text on form panels, position the cursor on the portion of the panel where the footing text is specified and enter the BOTTOM command.

CANCEL

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The CANCEL command:

- Discards pending modifications made during a Table Editor session.
- Returns to a primary QMF panel from a help panel.
- Cancels a confirmation panel for a command. When you press the Cancel function key from a confirmation panel, the command whose action you were asked to confirm is canceled and you return to the QMF panel you entered the command on.

The Cancel command is only available as a function key. You can use the Cancel function key from the Table Editor, Prompted Query panels, QMF help panels, some forms panels, and confirmation panels. It is also available on the List and Add Globals panels.

CANCEL is available in the Table Editor session depending on the SAVE option specified on the EDIT TABLE command (see page 29):

- When SAVE=END, changes are discarded when the Cancel function key is pressed.
- When SAVE=IMMEDIATE, CANCEL is not accepted.

CHANGE

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

In Prompted Query, the CHANGE command displays a panel on which you can make changes. In the Table Editor, the CHANGE command modifies rows in a table or view.

In Prompted Query, you can use one of the following methods to make changes:

- Move your cursor to the entry you want to change and press the Change function key.
- Type Change on the command line, move your cursor to the entry you want to change, and press Enter.

In the Table Editor, make your changes and press the Change function key.

- When SAVE=IMMEDIATE, changes are saved when the transaction is processed.
- When SAVE=END, changes are saved when the END command is processed.

CHECK

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The CHECK command checks form panels for mistakes.

To use CHECK, display a form panel, and enter CHECK on the command line or press the Check function key. You can display any form panel to start. QMF checks the one currently displayed, and then checks the remaining form panels. QMF first checks for errors. After all errors are corrected, QMF checks for warning conditions.

Error Conditions

If a form panel contains an error, QMF displays the panel on which the first error occurs, with the word ERROR at the top of the panel. If only one form panel contains an error, QMF displays the word ERROR on all the form panels. The entry area containing the error is highlighted, and the cursor is positioned next to it. The message on the message line describes the error.

You must correct the error before you can see the next error or create the report. For more information about the error and what you must do to correct it, press the Help function key. To identify the next error, enter the CHECK command again and correct the error. Continue in this way until you correct all errors.

If FORM.CALC, FORM.CONDITIONS, or a column definition panel in FORM.COLUMNS contains an expression with an error, this error might not be detected until QMF passes the values to REXX for evaluation.

Warning Conditions

If the form panels have no errors, or if you corrected all of them, QMF checks for warning conditions. If a warning condition is found, QMF displays the form panel on which the first warning condition occurs, with the word WARNING at the top of the panel. In addition, the cursor is positioned next to the entry area containing the conflicting value, and a message describes the condition.

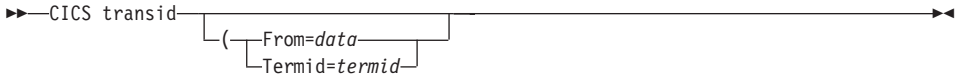
Unlike errors, warnings are not highlighted, and you can see all the warning conditions (without having to change the conflicting values) by repeatedly issuing the CHECK command. You need not change values that cause warning conditions—QMF can interpret the values and format your report. However, the report might not show the expected results. For more information about the warning and what you can do to correct it, press the Help function key. See also “Mistakes on Form Panels” on page 242.

CICS

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
				X

When you use QMF in the CICS environment, you can use the CICS command to start a CICS transaction (from the system at which QMF is executing) without ending your QMF session.

The CICS command is not valid in CMS or TSO.



transid The CICS transaction to be started.

From=*data*
The data to be used. The value for *data* is limited to 55 characters.

Termid=*termid*
The associated terminal. If you omit the TERMID parameter, no terminal is associated with your transaction.

If you are using a global variable in the FROM parameter, you must surround the global variable with single quotes (’). For example, if you issue:

```
CICS transid (FROM='&DSQAP_CICS_PQNAME'
```

the CICS command resolves correctly.

QMF supports the CICS command by issuing a CICS START command. The CICS command parameters (TRANSID, FROM, and TERMID) have the same meaning as the CICS START command. The QMF CICS command conforms to specifications as stated in the *CICS Application Programmer’s Reference* for the CICS START command.

CLEAR

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The CLEAR function key restores the values in all output fields in the Table Editor to their initial content.

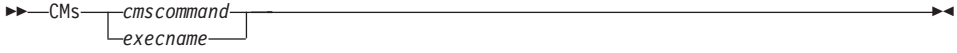
If the MODIFY confirmation category is in effect and changes are made to the panel, a confirmation panel is displayed.

CMS

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
		X	X	

When you use QMF in the CMS environment, you can use the CMS command to enter a command in the CMS environment (at the system where QMF is executing) without ending your QMF session.

The CMS command is only accepted in the CMS environment.



cmscommand or *execname*
Everything after the name of the command is sent to the CMS environment and interpreted in CMS.

For example, to display a list of all the files on your A-disk starting with the letter “S”:

```
CMS LISTFILE S* * A
```

After displaying the list of files, the QMF panel on which you issued the CMS command is displayed. If it does not run successfully, a message is displayed that contains the return code.

You can also enter VM/CP commands through the CMS command. To do so, insert CP just after CMS in the command, with a blank between the two commands.

Use the CMS command cautiously. If you are not familiar with CMS, you might adversely affect your environment. For more information on CMS commands, see *Installing and Managing QMF for VM/ESA*.

Values that contain an ampersand (&) are treated as ISPF variables when passed to CMS. If you use an ampersand, you must double it. For example, change `FUNCTION1 &COUNT=5` to `FUNCTION1 &&COUNT=5`

CONNECT

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
*	*	X	X	*

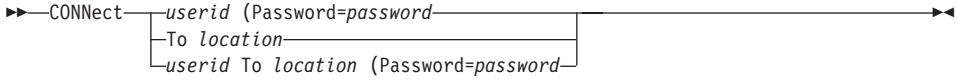
With the CONNECT command you can:

- Connect to any database that is part of the distributed network from within a QMF session.
- Change the user ID under which you are running QMF.
- Connect to another database using the specified user ID.

CONNECT is also useful for running procedures in batch mode.

The syntax of the CONNECT command varies for VM, VSE, and MVS.

To Connect from an SQL/DS Application Requester in VM:



To Connect from an SQL/DS VM Application Requester to an SQL/DS VSE or DB2 Application Server:

To Connect as a Different User ID on SQL/DS in VSE:

To Connect from a DB2 Application Requester in MVS (CICS or TSO):

userid
The SQL/DS authorization ID on whose authority the connection to the database is to be made. (This ID must have authority to connect to the database.)

You can specify *userid* in double quotes, but you cannot specify it in single quotes. If you connect to another SQL/DS location in VM without specifying an authorization ID, your VM logon ID is used as the default authorization ID at that SQL/DS location.

Password=*password*
The same password used when the authorization ID specified in this command was granted authority to connect to the database.

You must specify a password when you specify an authorization ID, and you can only specify a password if an authorization ID is specified. Otherwise, you receive an error message. For security reasons, your password is not visible on the CONNECT command prompt panel.

location
The name of the database location you want to connect to.

location is mandatory when you want to connect to a new location. For example, if you issue:

```
CONNECT userid (Password=password
```

the location to which you are connected does not change; that is, *location* defaults to the current location name.

You can specify *location* in double quotes, but you cannot specify it in single quotes. The maximum length in characters of the location value depends on the type of the application requester, as shown in Table 1.

Table 1. Maximum Length of Location Value Based on Requester Type

Requester type	Maximum Length
UDB for OS/390	16
DB2 for VM and VSE	18

Before making a new connection, QMF finishes any outstanding work (for example, a large report) at the current location. See "Report Completion and the Incomplete Data Prompt" on page 274 for more information.

When you successfully connect to a new location:

- You use the tables, views, and QMF objects at the new location. The TRACE value is an exception. Its value comes from either the DSQSDBUG value (a program parameter on the START command), or from the first user profile that is read when the QMF session is initialized.
- The profile (except for the TRACE parameter), resource control table, synonyms, and function keys are re-initialized to the values at the new (current) location. Also, the newly connected user's profile determines which resource group rows are passed to the QMF governor. Consequently, when you issue a RESET or SAVE command on your profile, the profile at the new location is read or updated.
- QMF uses the user edit routines and governor exit modules that reside at the system where QMF is executing.
- You cannot issue commands from a database object list that was not created at the current location.

- If you have queries, forms, or procedures in the QMF work area, they are unchanged when you connect to a new location. However, if you requested a list from one of the Prompted Query dialog panels, the table list associated with that query becomes obsolete. It is rebuilt when you press the List function key on one of the Prompted Query dialog panels.

If QMF fails to connect to the specified location, QMF attempts to ensure that you remain connected as you were before you issued the CONNECT command. If that is not possible, the lost connection prompt is displayed.

QMF maintains the prevailing database special register values if you reconnect to the current location without connecting to a different location in between.

Authorization IDs

Use the CONNECT command to connect to the application server named by the location parameter, as in the following example. The SQL authorization ID at the new location is system defined.

```
CONNECT TO location
```

Use the CONNECT command from an SQL/DS requester to change the SQL authorization ID at an SQL/DS server, as in the following example. This SQL authorization ID is in force only during the current connection to the server.

```
CONNECT userid (p=pw
```

To change the SQL authorization ID at a DB2 server (from any requester), use the DB2 statement SET CURRENT SQLID in a QMF SQL query. This SQL authorization ID is in force during the current connection to the server.

You cannot change the SQL authorization ID from QMF when a DB2 requester is connected to an SQL/DS server, nor can you change the SQL authorization ID at a workstation database server (from any requester).

Differences between CONNECT and the DSQSDBNM Program Parameter:

DSQSDBNM is not available in VSE.

DSQSDBNM establishes the location used during QMF initialization, as well as for the QMF session, until CONNECT to *location* is issued.

CONNECT to *location* allows you to change the current location after a QMF session is established.

Restrictions:

- You cannot use the CONNECT command while the Table Editor is active.
- An SQL CONNECT statement cannot run as a QMF SQL query because CONNECT must be a static SQL statement. Queries run through the QMF SQL query panel are always executed dynamically.

- You cannot connect to a DB2 server in an MVS environment or to a SQL/DS server in a VM environment within a distributed network from a SQL/DS database in a VSE environment, but you can connect to another user ID on the same SQL/DS database in a VSE environment.
- If you are using CICS/MVS and using remote data access, data on the remote databases are read only.
- If you use the CONNECT command and your user ID is T or TO, the user ID must be enclosed in double quotation marks.

CONVERT

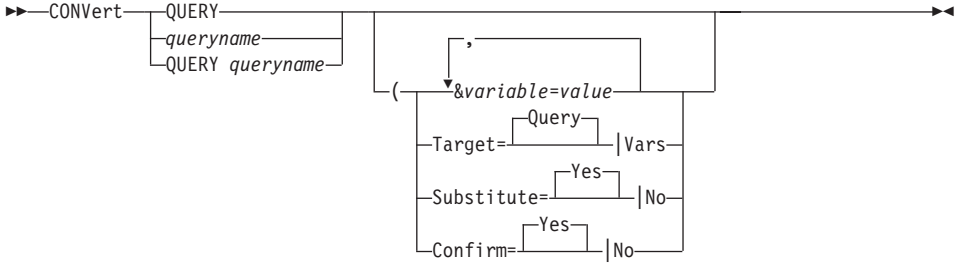
TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The CONVERT command converts QBE or relational prompted queries to queries with standard SQL syntax. CONVERT assigns values to variables and removes any original comments in a query.

A single QBE insert or update query can be converted into multiple SQL queries. These are placed into a single SQL query, and all queries after the first one are commented out.

The query being converted must be located at the database to which you are currently connected. Therefore, queries at remote databases cannot be converted unless you first connect to the remote location.

When you convert a query, the Target option determines where the converted query will be when the command finishes.



QUERY
Converts the query in QMF temporary storage.

queryname
Converts a query in the database.

When you convert a query stored in the database, the original query in the database remains unchanged.

QUERY *queryname*
Converts a query in the database.

When you convert a query in the database, the query in the database is not changed. Specifying QUERY is optional, but if used, it must precede the name.

When you convert a query, the converted query is displayed from QMF temporary storage. If a query already exists in QMF temporary storage, the converted query replaces it. When you convert a query stored in the database, the query in QMF temporary storage is replaced, but your original query in the database remains unchanged.

&*variable***=***value*
You can specify values for up to 100 variables in a query. Up to 10 values can be specified on CONVERT; you can set the others with SET GLOBAL (see page 113). QMF first looks at the command for a value, and then it looks for a global value. If the limit is exceeded, the command is rejected with an error message. Variable names that don't match parameters in your query are ignored.

If you defined your variables with the SET GLOBAL command, you do not have to specify them on CONVERT. A value specified on the CONVERT command overrides the same value set with SET GLOBAL. If you have variables in your query and do not specify values for them on your CONVERT commands, a prompt panel is displayed. All supplied parameter values appear on the prompt panel. Any variable names included in your query that aren't assigned values are listed followed by an arrow, and a message is displayed.

If you omit the variable parameter, and the object to be converted is a query that uses variables, a prompt panel is displayed.

Target=Query | Vars
Whether the target should be QMF temporary storage or a variable pool.

Query Places the converted query in QMF temporary storage, and displays it on the SQL Query panel.

This is the default.

Vars Places the converted query into the ISPF dialog manager variable pool or into the QMF global variables beginning with DSQQC.

The query in the query temporary storage area or in the database is not changed. The ISPF variable pool and the QMF global

variables are changed. (See "DSQ Global Variables That Show Results of CONVERT QUERY" on page 315 for a list of QMF global variables.)

Substitute=Yes | No
Whether substitution should take place for variables.

Yes If you have variables in your query, QMF attempts to substitute values for them. If all the variables are defined (either with the &variable parameter, or predefined global variables), no prompt panel is displayed. If QMF cannot resolve all the variables, it prompts you to enter values.

No No variable names in your query are resolved.

If you provide values for substitution variables and specify SUBSTITUTE=NO, an error message is displayed.

Confirm=Yes | No
Whether a confirmation panel is displayed:

Yes Displays a confirmation panel.

No No confirmation panel is displayed. The query is generated with the existing values of the variables.

If this parameter is not specified, the value in your profile is used.

A confirmation panel is displayed when you convert a query from QMF temporary storage and use TARGET=QUERY. This enables you to save prompted queries that were not saved before converting them to SQL queries.

Examples

To convert a query in QMF temporary storage into an SQL query and substitute a value of 38 for the variable DEPT in the converted query:

```
CONVERT QUERY (&DEPT=38
```

To convert a query in the database named QBEQUERY into an SQL query and put it into QMF temporary storage:

```
CONVERT QUERY QBEQUERY
```

To convert a query in the database named MYQUERY into an SQL query, and put it into the ISPF dialog manager pool and the global variable pool:

```
CONVERT QUERY MYQUERY (TARGET=VARS
```

DELETE

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

You can either use the Delete function key or type DELETE on the command line to remove any of the following:

- One line at a time from an SQL query or procedure
- A line from a prompted query
- A line of column information from a FORM.MAIN or FORM.COLUMNS panel
- A calculation line from a FORM.CALC or FORM.CONDITIONS panel
- A line of text from a FORM.BREAK*n*, FORM.DETAIL, FORM.FINAL, or FORM.PAGE panel
- A row from a table in the database when using Table Editor (Delete function key only)
- Error messages displayed below a query

To delete a line, position the cursor on the line to be deleted and press the Delete function key.

When using DELETE in the Table Editor, the transaction is saved immediately or when you end your Table Editor session. You can specify which method you want with the SAVE parameter on the EDIT command.

If a table or table join is deleted from a prompted query, QMF reevaluates the remaining joins to determine whether all tables left are still connected (joined).

If tables are still joined, all remaining joins are left in the query.

If all of the tables are not joined, the only joins left are for the tables connected to the first table selected for the query. QMF displays the Join Tables panel to prompt you to build any remaining joins for the other tables.

The QMF DELETE command operates differently from the SQL and QBE DELETE keywords. See your SQL reference for information on the SQL keyword. See *Using QMF* for information on the QBE keyword.

DESCRIBE

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The DESCRIBE command displays information about a QMF object or a column in a table. The Describe function key can be used from a database object list panel or a Prompted Query panel.

Using DESCRIBE on a database object list panel displays detail information about a single object. The amount of information shown is based on the type of object. On a Prompted Query panel, DESCRIBE displays a Column Description panel that shows information about the columns listed.

DISPLAY

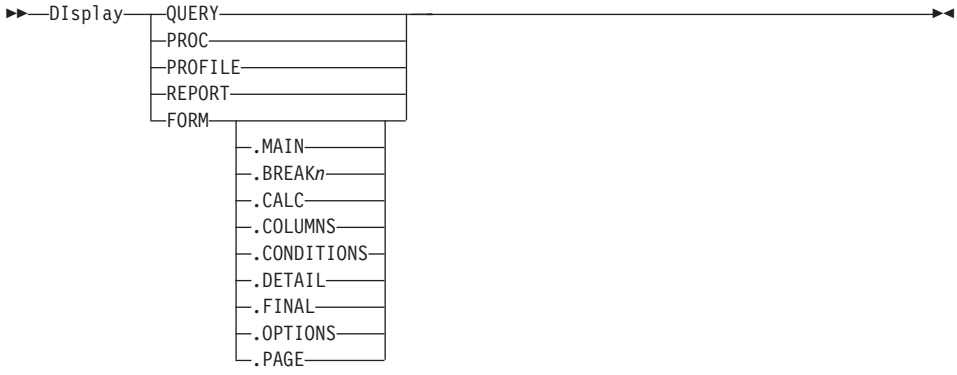
TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

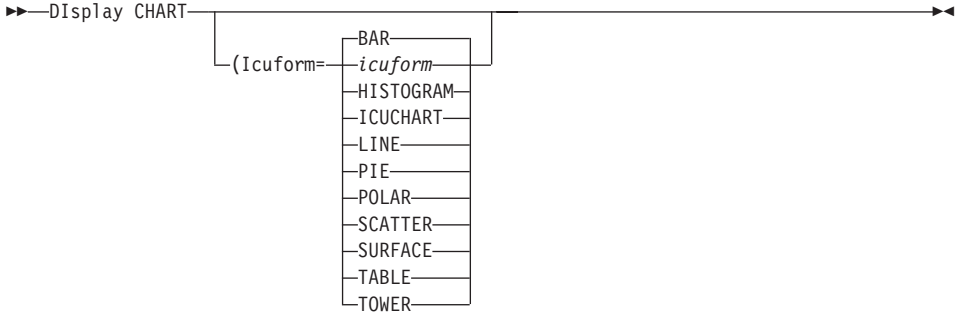
The DISPLAY command displays an object in QMF temporary storage or an existing object that is saved in the database. If the object contains a non-displayable character, QMF will not display the character.

Note: A QMF administrator can display any object owned by any user.

Display an Object in QMF Temporary Storage

You can display the contents of a query, form, procedure, profile, report, or chart in the QMF temporary storage area.





Icuform=*icuform*
The name of an ICU chart format saved in the ICU, or one of the chart formats QMF provides.

If you previously viewed a form panel, DISPLAY FORM displays the last form panel you viewed. If you have not displayed any part of the current form, DISPLAY FORM displays FORM.MAIN.

When you use DISPLAY CHART, the contents of DATA as formatted by FORM are displayed. The data can be further formatted by the Interactive Chart Utility (ICU) to represent report data graphically. To display a chart, you must have a graphics terminal.

After you work on a chart in the ICU and exit, the QMF panel on which you entered the DISPLAY CHART command is displayed again. If you want to return to a form panel, enter the DISPLAY CHART command from that form panel.

If no chart format is specified when you enter the DISPLAY CHART command, QMF uses a default chart format named DSQCFORM. DSQCFORM is a bar chart unless you (or your installation) change it to another chart type.

If you enter CHART on the DISPLAY command prompt, the DISPLAY CHART command prompt appears so that you can specify the parameters needed to display your chart.

If you are displaying a report or chart and the form is incompatible with the data or contains errors, the first form panel containing an error is displayed with the error highlighted. To see any other errors, you must correct the first error that is displayed, and then issue the CHECK command, or try to display the form or chart again to see the next error.

A displayed report looks different from a printed report. See "Differences between Printed and Displayed Reports" on page 95.

The SHOW command is similar to the DISPLAY command. The DISPLAY command displays objects from the database or from QMF temporary storage. The SHOW command shows object panels, global variables, and certain parts of panels in QMF temporary storage.

Display an Object in the Database

You can display an object in the database by specifying the object name. The object type is optional. If you specify an object type, it must precede the object name.

objectname
Names a query, form, procedure, or table that is saved in the database.

If the named object is not a table, it replaces the contents of the same object in the QMF temporary storage area.

If the object is a table, you can retrieve it from either the current location (if the name is not an alias or a three-part name) or the specified location (if the name is an alias or a three-part name).

You can display tables owned by other people. You must be authorized to display the tables, and you must use the owner qualifier.

You can display tables from other locations. To display a table from another location, both the current and other database location must support three-part names or remote unit of work, and you must include a location qualifier. Remote unit of work requires that you connect to the other location first.

A database label for a table or view is not seen in place of the actual name.

Displaying a table in the database affects QMF temporary storage areas. The data replaces the contents of the DATA object and changes the FORM object.

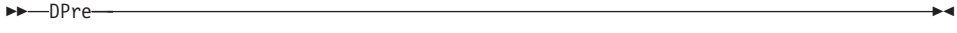
Examples

- To display a prompt panel from the QMF Home panel:
  ```
  DISPLAY ?
  ```
- To display Jennifer's query named SELECT_STAFF:
  ```
  DISPLAY QUERY JENNIFER.SELECT_STAFF
  ```
- To display the table STATUS, located at Tokyo, and owned by Okamoto:
  ```
  DISPLAY TABLE TOKYO.OKAMOTO.STATUS
  ```

DPRE

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X		X		

DPRE is a QMF-supplied command synonym that runs the Display Printed Report application.



This application lets you display a formatted report on your terminal. It displays the report that is currently in QMF temporary storage.

You must be authorized to use DPRE; consult your QMF administrator.

Printed reports look different from displayed reports. See "Differences between Printed and Displayed Reports" on page 95 for some specific differences.

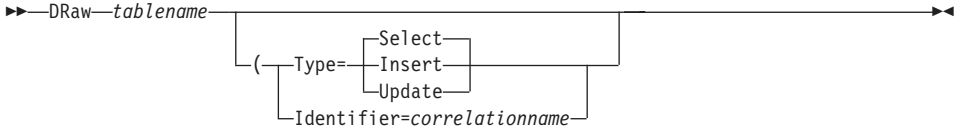
For additional information on using DPRE, see *Installing and Managing QMF for VM/ESA* or *Installing and Managing QMF for MVS*.

DRAW

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

For SQL queries, the DRAW command creates basic queries by retrieving the description of a table. With the DRAW command, you must specify the name of the table or view to be queried. You can specify the type of query you want to compose.

The DRAW command works differently in QBE. See *Using QMF* for information.



Select
Composes a basic query for selecting data from the columns of a table or view. If TYPE is not specified, SELECT is assumed. Using SELECT with the DRAW command produces a query that retrieves all rows and all columns from the specified table. You can then modify the query as desired.

For example, a SELECT query of Q.STAFF composed by DRAW looks like this:

```
SELECT ID, NAME, DEPT, JOB, "YEARS"           -- Q.STAFF
     , SALARY, COMM                           -- Q.STAFF
FROM Q.STAFF
```

If you include a location qualifier, the query looks like this:

```
SELECT ID, NAME, DEPT, JOB, "YEARS"     -- GILROY.Q.STAFF
     , SALARY, COMM                     -- GILROY.Q.STAFF
FROM GILROY.Q.STAFF
```

To use this SELECT query, type the other clauses you need. If you are selecting from more than one table, use a DRAW command for each table name you want represented.

Insert
Composes a basic query to insert data into the columns of a table or view.

The following example shows an INSERT query of Q.STAFF composed by DRAW:

```
INSERT INTO Q.STAFF (ID, NAME, DEPT, JOB, YEARS, SALARY, COMM)
   VALUES (

-- ENTER VALUES BELOW   COLUMN NAME  DATA TYPE   LENGTH    NULLS
                     , -- ID          SMALLINT              NO
                     , -- NAME        VARCHAR     9         YES
                     , -- DEPT        SMALLINT              YES
                     , -- JOB         CHAR        5         YES
                     , -- YEARS       SMALLINT              YES
                     , -- SALARY      DECIMAL     ( 7, 2)   YES
                     ) -- COMM        DECIMAL     ( 7, 2)   YES
```

To insert values into Q.STAFF, type values to the left of the column names. See your SQL reference for more information on INSERT queries.

Update

Composes a basic query to change the data in a table or view.

The following example shows an UPDATE query of Q.STAFF composed by DRAW:

```
UPDATE Q.STAFF SET
-- COLUMN NAME   ENTER VALUES BELOW    DATA TYPE  LENGTH   NULLS
   ID=                                -- SMALLINT           NO
 , NAME=                              -- VARCHAR   9        YES
 , DEPT=                              -- SMALLINT           YES
 , JOB=                               -- CHAR      5        YES
 , YEARS=                             -- SMALLINT           YES
 , SALARY=                            -- DECIMAL   ( 7, 2)  YES
 , COMM=                              -- DECIMAL   ( 7, 2)  YES
WHERE
```

To use this UPDATE query, type the changes you want to make to the right of the column names, and delete the lines you don't need. Be sure to complete the WHERE clause. For information on writing queries to update data, refer to your SQL reference.

Identifier=*correlationname*

A user-defined correlation name to be used in the query. If you do not specify this option, no correlation name is used. This option is not valid when the type specified is Insert.

The following example shows a SELECT query of Q.STAFF and Q.ORG using correlation identifiers.

The command:

```
DRAW Q.STAFF (TYPE=SELECT IDENTIFIER=S
```

gives the result:

```
SELECT S.ID, S.NAME, S.DEPT, S.JOB,             -- Q.STAFF
     , S."YEARS", S.SALARY, S.COMM              -- Q.STAFF
```

To add a second table, this command:

```
DRAW Q.ORG (TYPE=SELECT IDENTIFIER=O
```

gives this result:

```
SELECT S.ID, S.NAME, S.DEPT, S.JOB              -- Q.STAFF
     , S."YEARS", S.SALARY, S.COMM              -- Q.STAFF
     , O.DEPTNUMB, O.DEPTNAME, O.MANAGER        -- Q.ORG
     , O.DIVISION, O.LOCATION                   -- Q.ORG
FROM Q.STAFF   S
   , Q.ORG    O
```

The DRAW command places double quotes around column names that contain:

- Special characters
- QMF reserved words
- IBM SQL reserved words
- DB2 reserved words
- SQL/DS reserved words

For example the command:

```
DRAW MYTABLE
```

gives the result:

```
SELECT NORMALNAME, KEYWORDFOLLOWS, "UNION"     -- USER.MYTABLE
     , "SPECIAL+CHARS_IN!", "HAS BLANKS IN IT" -- USER.MYTABLE
     , "MIXED_CASE_%S"                         -- USER.MYTABLE
FROM USER.MYTABLE
```

Limitations of the DRAW Command:

- DRAW does not retroactively add correlation variables to an existing query when you perform subsequent draws. You must manually add any correlation variables needed to avoid ambiguity.

Examples

- To create a query from the table VISIONS:

  ```
  DRAW VISIONS
  ```

- To create a query from the table JOHNSON.STATUS located at BILLINGS, and remain connected to the current DB2 location:

  ```
  DRAW BILLINGS.JOHNSON.STATUS
  ```

EDIT

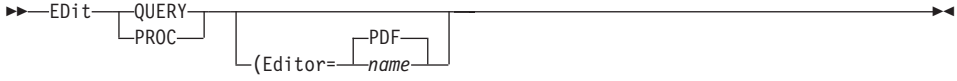
You can edit an SQL query or a QMF procedure in QMF temporary storage, or a table in the database (using the Table Editor).

See *Using QMF* to learn how to use the Table Editor.

Although you cannot use the EDIT command in CICS to edit a QMF query or procedure, you can use the QMF DISPLAY command to display a query or procedure, and then modify it using QMF.

Edit an SQL Query or a QMF Procedure

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	*	X	*	



Editor=PDF | *name*
Specifies the name of the editor to be used to edit your query or procedure.

PDF Specifies that the ISPF/PDF editor is used to edit the procedure or query. To use the PDF editor to edit a query or procedure, start QMF as an ISPF dialog.

name The name of any other editor available to you. It can also be the name of an EXEC (VM or MVS) or a CLIST (MVS) that starts an editor. For more information about available editors, see your information center.

If you want to build a new query or procedure using EDIT, reset the query or procedure first to clear the QMF temporary storage area. Do this by issuing the RESET command with the QUERY or PROC parameter.

If you want to modify an existing query or procedure, first display the query or procedure to bring it into the QMF temporary storage area. Then use the EDIT command to modify the query or procedure.

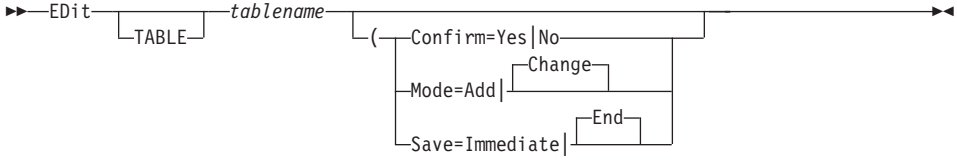
After editing your query or procedure, you can save your file or data set. This replaces whatever was in QMF temporary storage. If your query or procedure

is too large to fit in QMF's temporary storage area, it is stored in a file. If this happens, a message is displayed telling you the name of the file that your procedure or query is in.

The SAVE command in the editor is not the same as the QMF SAVE command. The editor only saves (or files) to the QMF temporary storage area. If you want the query or procedure to be saved in the database, you must use the QMF SAVE command.

Edit a Table

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	*



tablename
Specifies the name of the table or view you want to edit. You can specify the name of the table without the object type.

When QMF is running in CICS/MVS, you cannot update tables at remote locations.

Confirm=Yes|No
Whether a confirmation panel is displayed:

Yes Displays a confirmation panel, if necessary, depending on your current operation: ADD, CHANGE, DELETE, END/CANCEL, or MODIFY. When Confirm=Yes, QMF uses a set of global variables to determine whether a confirmation panel is actually displayed. See "DSQ Global Variables Associated with Table Editor" on page 304 for more information about global variables associated with the Table Editor.

No No confirmation panel is displayed.

Mode=Add|Change
Whether to add rows to a table or change (or delete) rows in a table.

Save=Immediate|End
Whether to save changes and deletions as they occur or when you end the edit session.

Immediate
Save the additions, changes, and deletions to the database when the transaction is processed.

SAVE=IMMEDIATE is not accepted in CHANGE mode unless the database supports CURSOR HOLD.

If you are editing your table with the SAVE=IMMEDIATE parameter, CANCEL is not allowed.

End Save the additions, changes, and deletions to the database when you issue the END command.

If you are editing your table with the SAVE=END parameter, the Cancel function key ends the session without saving the changes.

Examples

- To use a prompt panel to issue the EDIT command, display the EDIT command prompt panel:

  ```
  EDIT ?
  ```

- To edit the table VISIONS in CHANGE mode (the default):

  ```
  EDIT TABLE VISIONS
  ```

- To add new rows to a table named TABTWO owned by user Bill in New York (NY) and remain connected to the current DB2 location:

  ```
  EDIT TABLE NY.BILL.TABTWO (MODE=ADD
  ```

- To export the current query and place it in the ISPF/PDF editor:

  ```
  EDIT QUERY
  ```

  When the edit session ends, the edited file is imported to the current query object.

- To export the current query and place it in the XEDIT editor:

  ```
  EDIT QUERY (EDITOR=XEDIT
  ```

  When the edit session ends, the edited file is imported to the current query object.

END

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The END command ends a current operation and returns to an earlier state.

The result of the END command varies depending on what panel you are using and if an initial procedure is executing:

- If you enter END (or press the End function key) while looking at any of the following QMF panels, the QMF Home panel is displayed:

  ```
  QUERY      FORM.MAIN     FORM.COLUMNS
  PROC       FORM.CALC     FORM.OPTIONS
  PROFILE    FORM.DETAIL   FORM.BREAKn
  REPORT     FORM.FINAL    FORM.CONDITIONS
             FORM.PAGE     Global variable list
  ```

- If you enter END (or press the End function key) on:
  - The QMF Home panel, your QMF session ends.
  - A prompt panel, the panel on which you issued the command that caused the prompt is displayed. (This could be the QMF Home panel, or the panel for FORM, PROC, PROFILE, QUERY, or REPORT.)

    If you press the End function key after making an entry on the prompt panel and before pressing Enter, the entry you made is not processed.
  - A Table Editor panel, your changes are committed and the panel from which you called the Table Editor is displayed.

    When you press the End function key from a Table Editor panel, a confirmation panel is displayed so you can decide whether to end (commit your changes to the database) or not (return to the Table Editor panels).

END does not work as outlined above in the following situations:

- If QMF was started with an initial procedure, END runs the initial procedure again without displaying the QMF Home panel.
- If the current panel is the QMF Home panel and END is issued through the QMF command or callable interface, the QMF session is not terminated immediately. Instead, the EXEC, CLIST, or program containing the END command regains control. In this case, the QMF session is not terminated until the EXEC, CLIST, or program ends.
- If END is issued from a new interactive session that was started by the INTERACT command, control is returned to the application or procedure from which the INTERACT command was issued. In this case, END does not terminate the session or display the QMF Home panel.
- If the END command is issued from a new interactive session that was started as the result of issuing a command on the database object list panel, the database object list is displayed. In this case, END does not terminate the session or display the QMF Home panel.

For more information on using END in an interactive session, see *Developing QMF Applications.*

ENLARGE

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The ENLARGE command in QBE increases the size of an example table. See *Using QMF.*

ERASE

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The ERASE command removes two groups of objects:

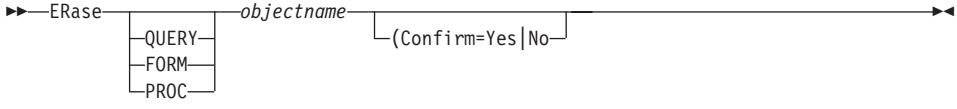
- QMF objects
- Database table objects

Note: A QMF administrator can erase any object owned by any user.

Erase a QMF Object

The ERASE command removes a query, form, or procedure from the database.

You can erase objects only from the current database location. You cannot erase a remote object by using a three-part name. Instead, first connect to the location where the object is located, and then issue the ERASE command.



objectname
The query, form, or procedure to be erased. When you specify form, all parts of the form are erased immediately. If you specify an object name that doesn't exist, a warning message is displayed.

Confirm=Yes | No
Whether a confirmation panel is displayed:

Yes Displays a confirmation panel if an object in the database will be removed by this command.

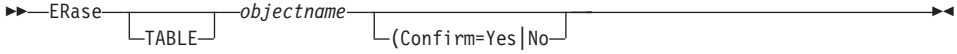
No No confirmation panel is displayed.

If this parameter is not specified, the value in your profile is used.

Erase a Database Table Object

The ERASE command also removes a table, view, synonym, or alias from the database.

You can erase objects only from the current database location. You cannot erase a remote table by using a three-part name. Instead, first connect to the location where the table is located, and then issue the ERASE command.



objectname
The table object to be erased. If you specify an object name that doesn't exist, a warning message is displayed.

Confirm=Yes | No
Whether a confirmation panel is displayed:

Yes Displays a confirmation panel if an object in the database will be removed by this command.

No No confirmation panel is displayed.

If this parameter is not specified, the value in your profile is used.

Examples

- To display a command prompt panel:
  ```
  ERASE ?
  ```
- To erase the database synonym T1:
  ```
  ERASE TABLE T1
  ```
- To erase the table DANIEL.TABLEONE:
  ```
  ERASE TABLE DANIEL.TABLEONE
  ```
- To erase a query named JBQUERY and display a confirmation panel.
  ```
  ERASE JBQUERY (CONFIRM=YES
  ```
- To erase the table DANIEL.TABLETWO at the DALLAS location while your local location is BOISE, you must first connect to DALLAS:

```
CONNECT TO DALLAS
```

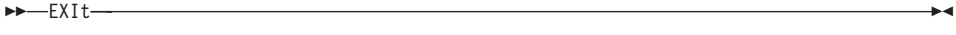
then issue the ERASE command:

```
ERASE TABLE DANIEL.TABLETWO
```

EXIT

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The EXIT command stops your QMF session.



You can issue the command on the QMF Home panel, the QUERY, REPORT, FORM, PROFILE, or global variable list panel, or you can put it in a procedure.

You can also enter the EXIT command from the QMF command area of any object on the QMF database object list panel (see page 81). You cannot enter the EXIT command on a command prompt, confirmation, or Help panel.

For users developing QMF applications: If you issue EXIT through the QMF command interface or in a procedure that is run through the command interface, your session is not terminated immediately. Instead, the EXEC, CLIST, or application program that is running from the command interface regains control. Your session is not terminated until the TSO or CMS commands complete.

EXPORT in CICS

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
				*

The EXPORT command in CICS sends the following objects to a CICS data queue (or, for charts, to a GDDM library that contains GDF files) at the system where QMF is executing:

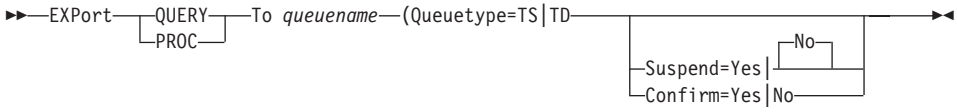
- Queries, forms, procedures, reports, charts, and data from *QMF temporary storage*.
- Queries, forms, procedures, and tables from the *database*.

The syntax for exporting objects from QMF temporary storage is different from the syntax for exporting objects from the database.

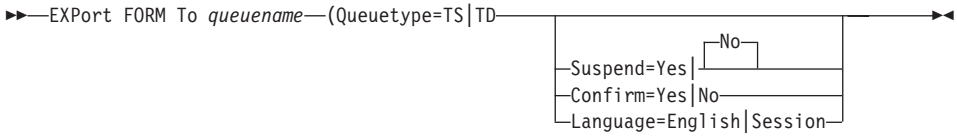
For MVS, we do not recommend using TSO datasets in CICS. For information on using TSO datasets from QMF 3.1.1, see the chapter on migration in *Installing and Managing QMF for OS/390*.

Note: A QMF administrator can export any object owned by any user.

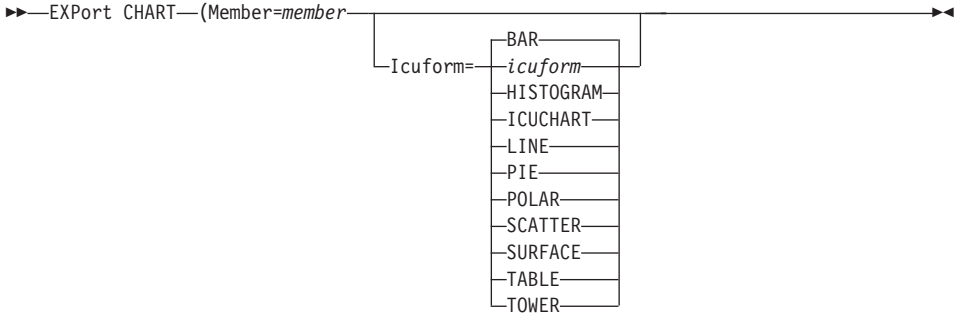
Export from QMF Temporary Storage to a CICS Data Queue

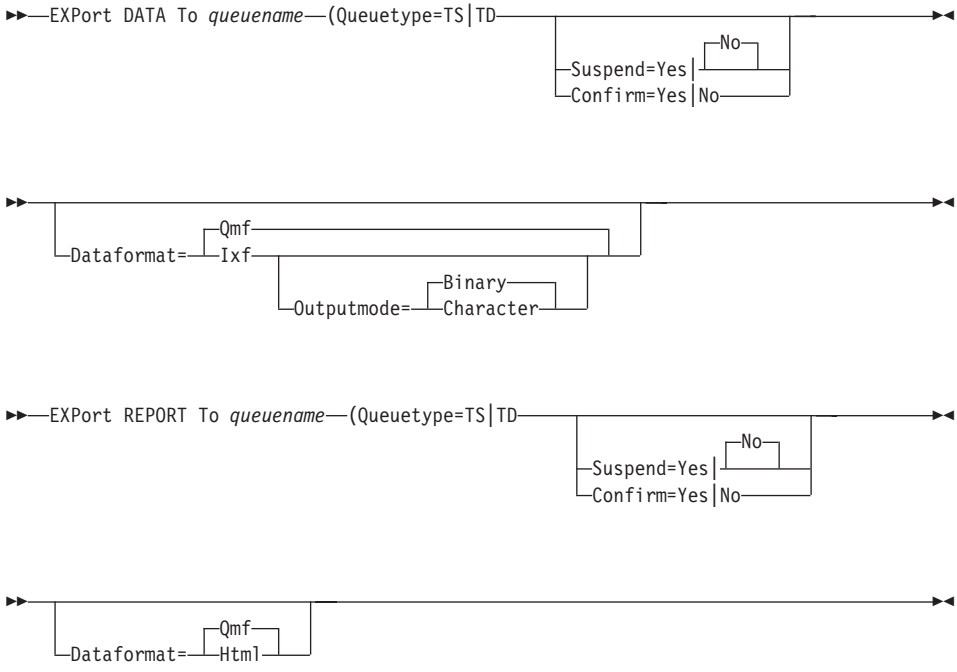


A QMF procedure that is written in English can be exported, then imported, and run in a non-English QMF session if the global variable DSQEC_NLFCMD_LANG is set. Specify 1 for English or 0 for a non-English QMF session. QMF assumes 1 unless it is changed with the SET GLOBAL command.



When a form is exported, QMF drops any FORM.DETAIL panel variation that was not modified from its default values. In this manner, you can drop unwanted FORM.DETAIL defaults by exporting and then importing the same form.





queuename

The name of a CICS temporary storage queue or a CICS transient data queue. You must specify *queuename* when using EXPORT.

An empty or partial CICS data queue might exist if there is an error in the execution of the EXPORT command. If a CICS data queue with the chosen queue name already exists, its contents are replaced by what is exported. See *Developing QMF Applications* for a detailed description of the formats of objects that are exported.

If a data or report object is being exported to the same data queue from which the current DATA object was imported, and the entire imported object won't fit into the QMF DATA storage area, the Incomplete Data Object prompt is displayed. At the prompt, choose NO so the data object is not reset and the original imported object is retained. Then export the data or report object to a different data queue.

If you are entering the EXPORT command on the command line and the queue name is a CICS temporary storage queue containing a character that has special meaning to QMF (such as a period or quote), enclose the entire queue name in single quotes and double all embedded single quotes. Do not enclose the queue name in double quotes. The length of the resulting queue name, including all quotes, cannot exceed ten characters.

Queuetype=TS|TD
Type of CICS data queue used to contain the exported QMF object. This parameter must be specified if *queuename* is specified.

TS Writes the QMF object to a CICS temporary storage queue on an "auxiliary" storage device.

When exporting to a CICS TS queue, QMF checks to see if the queue already contains an object. If so, QMF prompts you to replace the contents of the queue (proceed with the EXPORT command) or cancel the EXPORT command.

TD Writes the QMF object to a CICS transient data queue.

When exporting to a CICS TD queue, QMF does not check to see if the queue already contains an object. If the queue does contain an object, QMF appends the exported object to the object currently in the queue. This causes problems if you import the object from the queue.

To avoid problems when you import the object, make sure the TD queue is empty and ready to receive an object before you issue the EXPORT command.

Suspend=Yes|No
The action taken if the queue is busy. When the SUSPEND parameter is used, QMF issues a CICS ENQ command for the CICS data queue name. If the results of the CICS ENQ command indicate ENQBUSY, the data queue is considered busy.

Yes Waits until the queue is available.

No If the queue is not available, the export command is canceled and a message is returned. If SUSPEND is not specified, NO is assumed.

Confirm=Yes|No
Whether a confirmation panel is displayed. This parameter is only valid if the queue type is TS.

Yes Displays a confirmation panel if a CICS temporary storage queue will be replaced or changed by this command.

No No confirmation panel is displayed.

If this parameter is not specified, QMF uses the value in your profile.

The confirmation panel is not used for EXPORT CHART. The ICU controls where a chart is exported and always replaces an existing member that has the same name.

Member=*member*
MEMBER is required for the EXPORT CHART command. Charts are exported to members within the GDDM library defined by an installation for GDF (graphics data file) data.

Language=English | Session
Whether a form is exported in English or in the current session language. A QMF form that is exported in English can be run in any NLF session. However, if it is exported in any other QMF national language, it can be run (after it is imported) only in a session of that same national language.

The default language is determined by the value of the QMF global variable DSQEC_FORM_LANG.

Icuform=*icuform*
The name of the ICU chart format that you saved in the ICU, or one of the QMF chart forms.

If a chart format is not specified, DSQCFORM (a bar chart) is used.

Dataformat=Qmf | Ixf
Which file format to use for your exported data. It can be QMF or IXF. If DATAFORMAT is not specified, QMF is assumed.

The choice of QMF or IXF format for your exported data depends on the intended use of the data:

- Use the QMF format if you have an existing application that uses the QMF format or you merely intend to transport data from one QMF user to another without any processing outside of QMF.
- Use the IXF format for all other situations. These include (among others) data for new applications, data transported to products that support IXF, or data processing outside of QMF.

  If IXF is specified, you can also use the OUTPUTMODE parameter.

Dataformat=Qmf | Html
Which file format to use for your exported report. It can be QMF or HTML. If DATAFORMAT is not specified, QMF is assumed.

The choice of QMF or HTML format for your exported report depends on the intended use of the report:

- Use the QMF format if you have an existing application that uses the QMF format or you merely intend to transport the report from one QMF user to another without any processing outside of QMF.
- Use the HTML format if you want your data formatted for display on the World Wide Web.

Outputmode=Binary | Character
Accepted *only* when DATAFORMAT=IXF. It specifies which output mode

to use for the non-header numeric data in your file. It can be BINARY or CHARACTER. If OUTPUTMODE is not specified, BINARY is assumed.

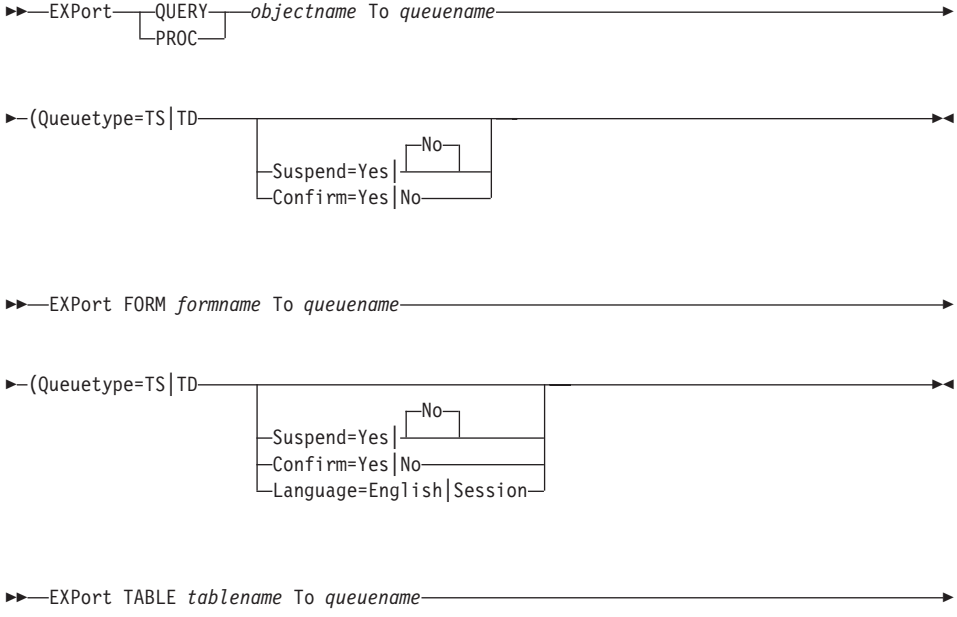
Use BINARY if your exported IXF queue will only transport data from one product to another, or if a program written in a compiled programming language such as COBOL or PL/I will process it. This results in faster processing because it is not necessary to convert the data from character form.

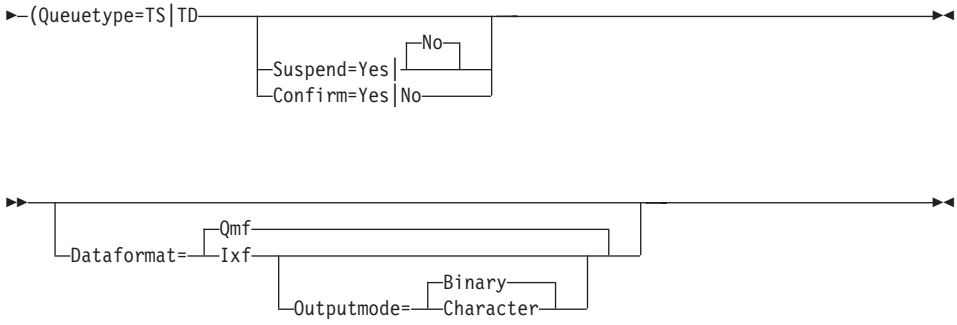
Use CHARACTER if the IXF queue will be edited.

When you request command prompting for the EXPORT command, you will receive two prompt panels. On the first panel, you can specify what type of object you want to export. On the second panel, you can specify the parameters associated with that object.

If you are exporting a report or chart, and the form is incompatible with the data or contains errors, the first form panel containing an error is displayed with the error highlighted. To see any other errors, you must correct the first error and press the Check function key.

Export from the Database to a CICS Data Queue





objectname
The name of the query or procedure to be exported.

formname
The name of the form to be exported.

tablename
The name of the table to be exported.

For descriptions of *queuename*, Confirm, Language, Member, Dataformat, and Outputmode, see page 37.

Examples

- To export data to a TD queue in IXF binary format:

```
EXPORT DATA TO TDQU
 (QUEUETYPE=TD CONFIRM=NO DATAFORMAT=IXF
```

- To export a table to a TS queue in IXF character format:

```
EXPORT TABLE KMMTABLE TO MYQUEUE
 (QUEUETYPE=TS DATAFORMAT=IXF OUTPUTMODE=CHARACTER
```

EXPORT in TSO

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X			*

The EXPORT command in TSO copies the following objects to a TSO data set at the system where QMF is executing:

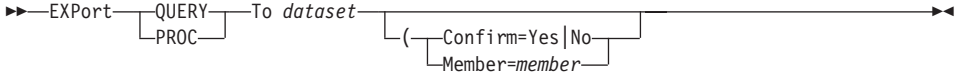
- Queries, forms, procedures, reports, charts, and data from *QMF temporary storage*.
- Queries, forms, procedures, and tables from the *database*.

The syntax for exporting objects in QMF temporary storage is different from the syntax for objects in the database.

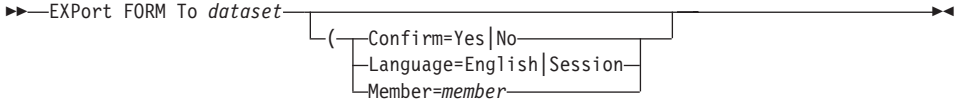
Note: A QMF administrator can export any object owned by any user.

Export from QMF Temporary Storage to a TSO Data Set

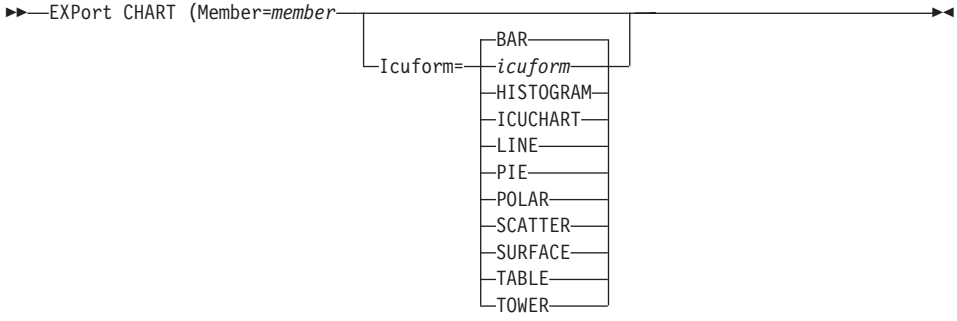
When you enter the EXPORT command, you might get a message with I/O information but no indication of an error. Such a message can be ignored.

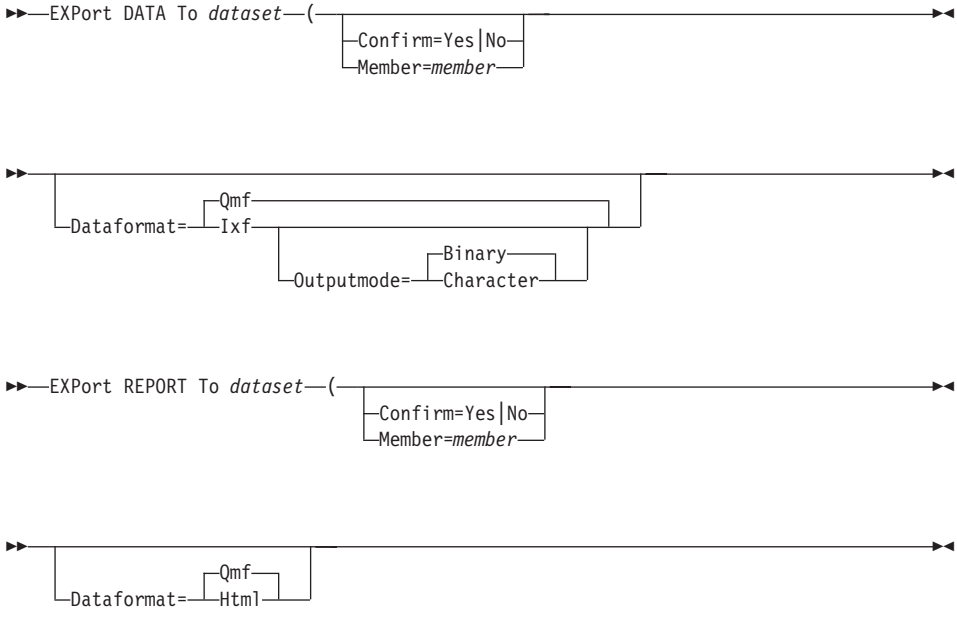


A QMF procedure that is written in English can be exported, then imported, and run in a non-English QMF session if the global variable DSSSQEC_NLFCMD_LANG is set. Specify 1 for English or 0 for a non-English QMF session. QMF assumes 1 unless it is changed with the SET GLOBAL command.



When a form is exported, QMF drops any FORM.DETAIL panel variation that was not modified from its default values. In this manner, you can drop unwanted FORM.DETAIL defaults by exporting and then importing the same form.





dataset

Can be in either of two forms:

- Any fully qualified TSO data set name. The entire name must be enclosed in single quotation marks. If you use the MEMBER parameter, the member name (*member*) is added as a suffix in parentheses.
- A partial TSO name. If *dataset* is not enclosed in single quotation marks, your TSO prefix is added as a prefix and the object type is added as a suffix. (If your TSO profile specifies NOPREFIX, no prefix is used. For new TSO users, the default for PREFIX is usually set to their user ID.)

If a data or report object is being exported to the same data set from which the current DATA object was imported, and the entire imported object won't fit into the QMF DATA storage area, the Incomplete Data Object prompt is displayed. At the prompt, choose NO so the data object is not reset and the original imported object is retained. Then export the data or report object to a different data set.

An empty or partial data set or member of a partitioned data set might exist if there is an error in the execution of the EXPORT command.

If a TSO data set with the chosen data set name already exists, its contents are replaced by what is exported if the file attributes are correct (for example, if the record format and logical record length are large enough to contain the data). See *Developing QMF Applications* for required file

attributes and a detailed description of the formats of objects that are exported. If a TSO data set with the chosen data set name does not exist, QMF dynamically allocates a data set with that name.

If you are not using the standard DASD device, you must pre-allocate your data sets before using the EXPORT command.

Confirm=Yes | No
Whether a confirmation panel is displayed:

Yes Displays a confirmation panel if a data set will be replaced or changed by this command.

No No confirmation panel is displayed.

If this parameter is not specified, the value in your profile is used.

The confirmation prompt is not used for EXPORT CHART. The ICU controls where a chart is exported and always replaces an existing member that has the same name.

Member=*member*
A member name in a TSO partitioned data set. The member name is added as another suffix in parentheses. For example, if your TSO prefix is TOM, and you enter the command:

```
EXPORT QUERY TO ROGER (MEMBER=GAMMA
```

member GAMMA is created and placed in data set:

```
'TOM.ROGER.QUERY(GAMMA)'
```

If the member name is longer than eight characters, QMF displays a message telling you that the member name is too long. You must change the member name so it contains no more than eight characters.

MEMBER is required for the EXPORT CHART command. Charts are exported to members in the partitioned data set defined by an installation for GDF data.

Language=English | Session
Whether a form is exported in English or in the current session language. A QMF form that is exported in English can be run in any NLF session. However, if it is exported in any other QMF national language, it can be run (after it is imported) only in a session of the same national language.

The default language is determined by the value of the QMF global variable DSQEC_FORM_LANG.

Icuform=*icuform*

The name of an ICU chart format that you saved in the ICU, or one of the QMF chart forms. If a chart format is not specified, DSQCFORM (a bar chart) is used.

Dataformat=Qmf | Ixf

Which file format to use for your exported data. It can be QMF or IXF. If DATAFORMAT is not specified, QMF is assumed.

The choice of QMF or IXF format for your exported data depends on the intended use of the data:

- Use the QMF format if you have an existing application that uses the QMF format or you merely intend to transport data from one QMF user to another without any processing outside of QMF.
- Use the IXF format for all other situations. These include (among others) data for new applications, data transported to products that support IXF, or data processing outside of QMF.

  If IXF is specified, you can also use the Outputmode parameter.

See *Developing QMF Applications* for the format of data when exported and for the file attributes of the exported data file.

Dataformat=Qmf | Html

Which file format to use for your exported data. It can be QMF or HTML. If DATAFORMAT is not specified, QMF is assumed.

The choice of QMF or HTML format for your exported report depends on the intended use of the report:

- Use the QMF format if you have an existing application that uses the QMF format or you merely intend to transport a report from one QMF user to another without any processing outside of QMF.
- Use the HTML format if you want your report formatted for display on the World Wide Web.

Outputmode=Binary | Character

Accepted *only* when DATAFORMAT=IXF. It specifies which output mode to use for the non-header numeric data in your file. It can be BINARY or CHARACTER. If OUTPUTMODE is not specified, BINARY is assumed.

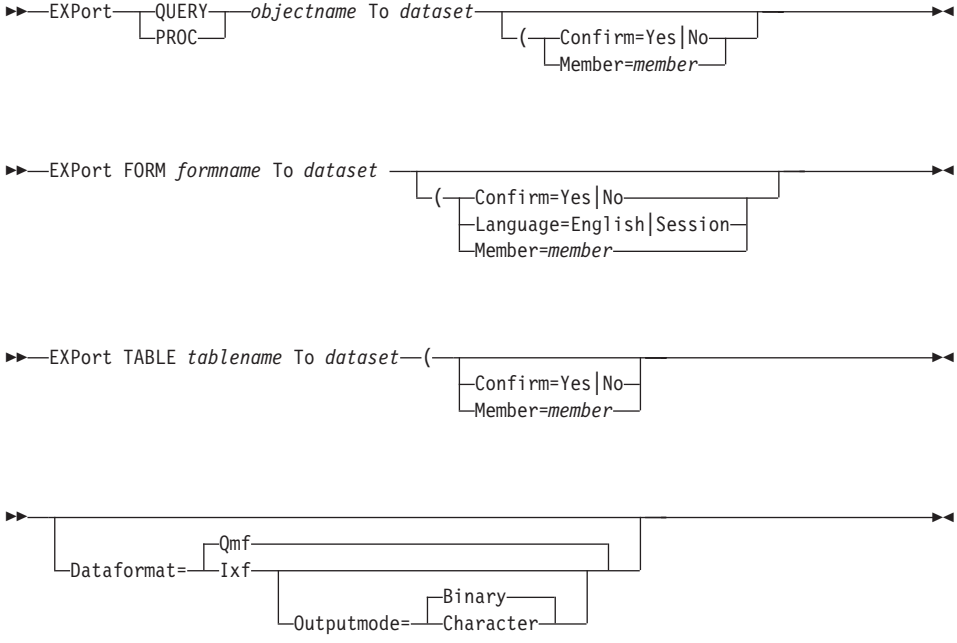
Use BINARY if your exported IXF data set will only transport data from one product to another, or if a program written in a compiled programming language such as COBOL or PL/I will process it. This results in faster processing because it is not necessary to convert the data from character form.

Use CHARACTER if the IXF data set is to be processed by a command interpreter language (such as REXX) or CLIST, or if it is to be edited. (These environments deal most effectively with data in character form.)

When you request command prompting for the EXPORT command, you will receive two prompt panels. On the first panel, you can specify what type of object you want to export. On the second panel, you can specify the parameters associated with that object.

If you are exporting a report or chart, and the form is incompatible with the data or contains errors, the first form panel containing an error is displayed with the error highlighted. To see any other errors, you must correct the first error and press the Check function key.

Export from the Database to a TSO Data Set



objectname
The name of the query or procedure to be exported.

formname
The name of the form to be exported.

tablename
The name of the table to be exported.

In TSO, you cannot export a table with a LONG VARCHAR data type column (length greater than 254).

For descriptions of *dataset*, Confirm, Language, Member, Dataformat, and Outputmode, see page 43.

Examples

- To export data in IXF character format:

```
EXPORT DATA TO JBLP
 (CONFIRM=NO DATAFORMAT=IXF OUTPUTMODE=CHARACTER
```

- To export a form using the current session language:

```
EXPORT FORM TO MYFORM (LANGUAGE=SESSION
```

- To copy the form FORMA at the current location to the data set FORMS at the system where QMF is executing:

```
EXPORT FORM FORMA TO FORMS
```

- To export a table from a remote database that does not support three-part names, first connect to that database:

```
CONNECT TO VENICE
```

  then export the table:

```
EXPORT TABLE JEAN.STATSTAB TO NONSTD
```

- To copy the table OKAMOTO.STATUS at the DB2 database in TOKYO to the data set YOURDATA on the system where QMF is executing:

```
EXPORT TABLE TOKYO.OKAMOTO.STATUS TO YOURDATA
```

EXPORT in CMS

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
		X	X	

The EXPORT command in CMS sends the following objects to a CMS file at the system where QMF is executing:

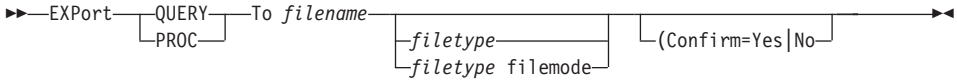
- Queries, forms, procedures, data, charts, and reports from *QMF temporary storage*
- Queries, forms, procedures, and tables from the *database*

For detailed descriptions of exported object formats, see *Developing QMF Applications* .

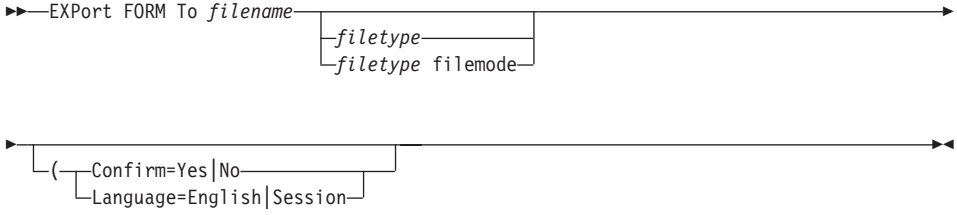
The syntax for exporting objects in QMF temporary storage is different from the syntax for objects in the database.

Note: A QMF administrator can export any object owned by any user.

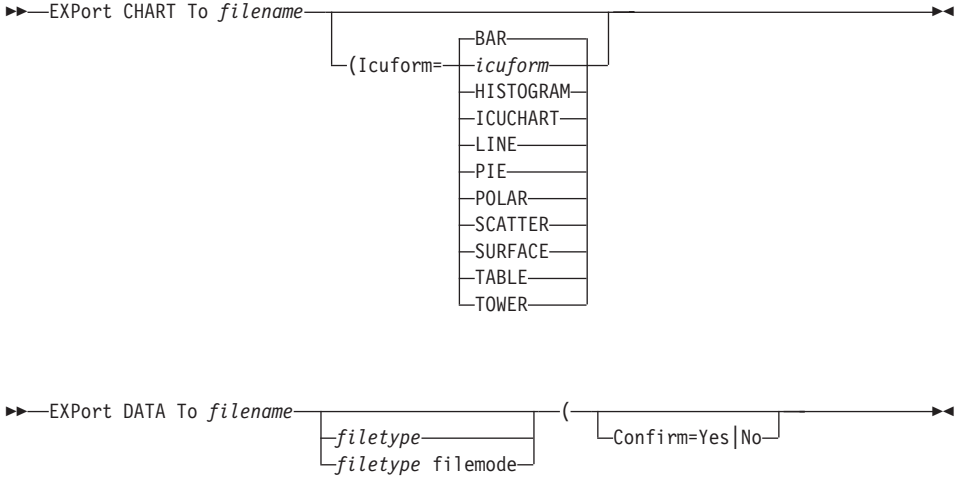
Export from QMF Temporary Storage to a CMS File

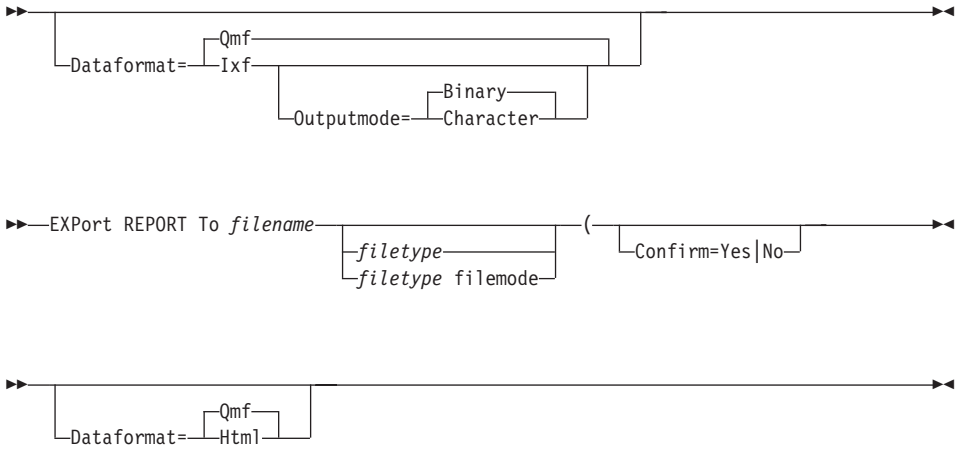


A QMF procedure that is written in English can be exported, then imported, and then run in a non-English QMF session if the global variable DSQEC_NLFCMD_LANG is set. Specify 1 for English or 0 for a non-English QMF session. QMF assumes 1 unless it is changed with the SET GLOBAL command.



When a form is exported, QMF drops any FORM.DETAIL panel variation that was not modified from its default values. In this manner, you can drop unwanted FORM.DETAIL defaults by exporting and then importing the same form.





filename, filetype, filemode
Names the CMS file that is receiving what is exported. If there is an error during execution of the EXPORT command, an empty or partial file might exist.

filename is required.

filetype can be omitted. If it is omitted, the object type is used.

filemode can be omitted. If it is omitted, the file mode is A.

For example, the following command sends the current form to a CMS file called STANDARD FORM A:

```
EXPORT FORM TO STANDARD
```

If a data or report object is being exported to the same file from which the current DATA object was imported, and the entire imported object won't fit into the QMF data storage area, the Incomplete Data Object prompt is displayed. At the prompt, choose NO so the data object is not reset and the original imported object is retained. Then export the data or report object to a different file.

Confirm=Yes|No
Whether a confirmation panel is displayed:

Yes Displays a confirmation panel if a file will be replaced or changed by this command.

No No confirmation panel is displayed.

If this parameter is not specified, the value in your profile is used.

The confirmation prompt is not used for EXPORT CHART. The ICU controls where a chart is exported and always replaces an existing file that has the same name.

Language=English | Session
Whether the form is exported in English or in the current session language. A QMF form that is exported in English can be run in any NLF session. However, if it is exported in any other QMF national language, it can be run (after it is imported) only in a session of that same national language.

The default language is determined by the value of the QMF global variable DSQEC_FORM_LANG.

Icuform=*icuform*
The name of an ICU chart format that you saved in the ICU, or one of the QMF chart forms.

If a chart format is not specified, DSQCFORM (a bar chart) is used.

Dataformat=Qmf | Ixf
Which file format to use for your exported data. It can be QMF or IXF. If DATAFORMAT is not specified, QMF is assumed.

The choice of QMF or IXF format for your exported data depends on the intended use of the data:

- Use the QMF format if you have an existing application that uses the QMF format or you merely intend to transport data from one QMF user to another without any processing outside of QMF.
- Use the IXF format for all other situations. These include (among others) new applications, transporting data to products that support IXF, or processing outside of QMF.

See *Developing QMF Applications* for the format of data when exported and for the file attributes of the exported data file.

Dataformat=Qmf | Html
Which file format to use for your exported report. It can be QMF or HTML. If DATAFORMAT is not specified, QMF is assumed.

The choice of QMF, IXF, or HTML format for your exported report depends on the intended use of the report:

- Use the QMF format if you have an existing application that uses the QMF format or you merely intend to transport the report from one QMF user to another without any processing outside of QMF.
- Use the HTML format if you want your report formatted for display on the World Wide Web.

Outputmode=Binary|Character
Accepted *only* when DATAFORMAT=IXF. It specifies which output mode to use for the non-header numeric data in your file. It can be BINARY or CHARACTER. If OUTPUTMODE is not specified, BINARY is assumed.

Use BINARY if your exported IXF data set will only transport data from one product to another, or if a program written in a compiled programming language such as COBOL or PL/I will process it. This results in faster processing because it is not necessary to convert the data from character form.

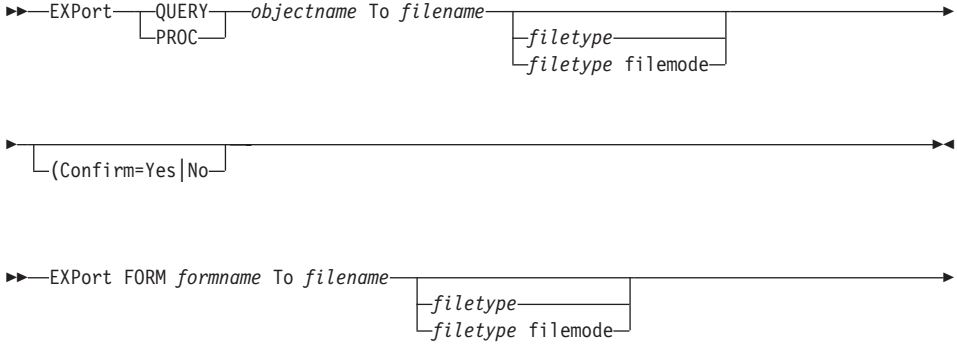
Use CHARACTER if you plan to process the IXF data set with a command interpreter language (such as REXX) or an EXEC, or if you will edit it. (These environments deal most effectively with data in character form.)

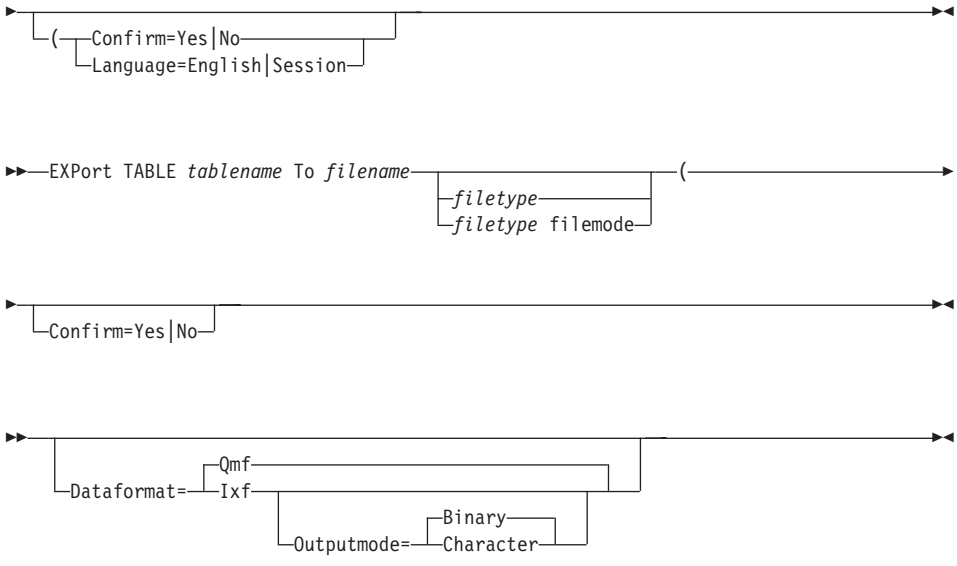
When you request command prompting for the EXPORT command, you will receive two prompt panels. On the first panel, you can specify what type of object you want to export. On the second panel, you can specify the parameters associated with that object.

If a CMS file with the chosen name, type, and mode already exists, its contents are replaced by what is exported if the record format is accepted. If no such file exists, one is created. See *Developing QMF Applications* for required attributes.

If you are exporting a report or chart, and the form is incompatible with the data or contains errors, the first form panel containing an error is displayed with the error highlighted. To see any other errors, you must correct the first error that is displayed and press the Check function key.

Export from the Database to a CMS File





objectname
The name of the query or procedure to be exported.

formname
The name of the form to be exported.

tablename
The name of the table to be exported.

For descriptions of *filename*, *filetype*, *filemode*, Confirm, Language, Dataformat, and Outputmode, see page 49.

Examples

- To export data in IXF binary format:

  ```
  EXPORT DATA TO MYFILE (CONFIRM=NO DATAFORMAT=IXF
  ```

- To copy the form FORMA at the current location to the file FORMS at the location where QMF is executing:

  ```
  EXPORT FORM FORMA TO FORMS FORM A
  ```

- If your current location is a DB2 database, you can export a table from a remote DB2 database using a three-part name:

  ```
  EXPORT TABLE VENICE.JEAN.STATSTAB TO YOURFILE TABLE A
  ```

- If your current location is a DB2 database, you can export the table OKAMOTO.STATUS from the DB2 database in TOKYO to the file YOURFILE at the system where QMF is executing by first connecting to the remote location:

```
CONNECT TO TOKYO
```

then exporting the table:

```
EXPORT TABLE OKAMOTO.STATUS TO YOURFILE
```

EXTRACT

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X		X		

The EXTRACT command accesses Data Extract (DXT) Dialogs or sends an extract request to DXT End User Dialogs from QMF. DXT lets you build extract requests that you then submit to have data extracted from various types of databases and files.

name
The name of an extract request sent to DXT for processing. The extract request *name* cannot be longer than 8 characters and must be the name of an extract request previously defined in DXT. When *name* is used, you do not leave the QMF session. You see a message indicating whether the extract request was successfully sent.

This field (*name*) can be left blank. If it is left blank, you gain access to the DXT End User Dialogs main menu panel. After exiting your DXT End User Dialogs session, you return to the QMF panel from which you issued the EXTRACT command.

Password=*password*
The password that must be used when an extract name is specified for a relational DB2 or SQL/DS table.

In VM, the password is substituted on an SQL CONNECT command that is issued from a CMS EXEC generated by DXT. In MVS, the password is substituted in the JOB statement of the JCL job that is generated by DXT. The password is not visible when entered.

To access DXT End User Dialogs, the DXT product must be installed at your installation and you must be set up as a DXT user. For more information, see your QMF administrator.

FORWARD

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The FORWARD command scrolls toward the bottom of a panel or to the last field of the current row in the Table Editor. You can scroll until the last line is at the top of your screen.

Csr Scrolls the line where the cursor is positioned to the top of the scrollable area.

Half Scrolls forward half the depth of the scrollable area or to the top if that is nearer.

Max Scrolls to the end of the scrollable area. FORWARD Max is equivalent to BOTTOM.

Page Scrolls forward the depth of the scrollable area or to the top if that is closer.

n Scrolls forward *n* lines. (*n* can be any number from 1 through 9999.)

You can scroll forward until the last line is at the bottom of your screen.

You can scroll forward as many lines as you like: to the cursor position, a half page, all the way forward, a whole page, or any number of lines (*n*) up to 9999.

If you don't specify an amount, the amount used is that shown after SCROLL ===> in the bottom right corner of the panel. You can change that amount by typing a new amount over it. The change remains in effect throughout the session, except for MAX, which remains in effect for only the next command.

You can also change the scroll amount QMF uses by setting the global variable DSQDC_SCROLL_AMT to Csr, Half, Page, or any number of lines (*n*) up to 9999.

To scroll forward in the footing text on form panels, position the cursor on the portion of the panel where the footing text is specified and enter the FORWARD command.

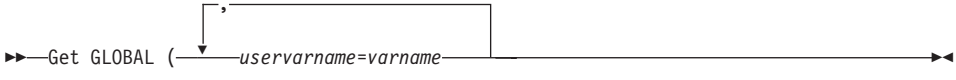
GET GLOBAL

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The GET GLOBAL command allows application programs (written in C, COBOL, REXX, FORTRAN, PL/I, RPG, or assembler language) to use the callable interface to access data from the QMF global variable pool. This command is not valid on the QMF command line.

The syntax of the command depends on the language you are using. Languages other than REXX use the extended syntax. See *Developing QMF Applications* for details.

The linear syntax, used by REXX, is shown here.



uservarname
The name of a REXX variable in your procedure with logic (or REXX EXEC).

varname
The name of a QMF global variable (see "Appendix B. QMF Global Variable Tables" on page 295 for a complete list of global variables).

For example, `GET GLOBAL (QMF_release=DSQAO_QMF_VER_RLS`

GETQMF Macro

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	

GETQMF is an edit macro, not a QMF command. It inserts a QMF report into a document.

From an edit session, you can issue the GETQMF macro to insert a QMF report into the document being edited without leaving the session. The QMF report to be inserted must be printed within a QMF session before it can be inserted into a document.

See *Using QMF* for more information on using the GETQMF macro.

```
GETQMF type option name
```

type Whether SCRIPT/VS control words are inserted.

DCF For a SCRIPT/VS document. Document Composition Facility (DCF) places the SCRIPT/VS control words before and after the QMF report. In addition, each printer page eject is replaced by a SCRIPT/VS page eject, and SCRIPT/VS control words are placed at the heading and footing of each page.

PROFS
For a PROFS document. PROFS produces the same results as DCF. It is provided in the GETQMF macro for ease of use by PROFS users.

ASIS For a QMF report as it is. If TYPE is not specified, ASIS is assumed.

option name
Whether you are creating a new report or inserting an existing one.

USEQMF
Creates a QMF report dynamically using a procedure that prints a report, where *name* is the name of the saved procedure.

FILE Inserts an existing report from a CMS file, where *name* is the name of the CMS file containing the report.

DSN Inserts an existing report from a TSO data set, where *name* is the name of the TSO data set containing the report.

HELP

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The HELP command displays information about QMF. Help can be displayed on some panels by entering the command on the command line. However, on most panels you can press the Help function key.

msgcode

A QMF message identification. QMF attempts to find message help associated with *msgcode*. If found, it is displayed. If not found, an error message is displayed. In QMF batch jobs, the message contains the message number in an L trace file.

A message code must begin with DSQ followed by five numbers, for example: `DSQ21000`. *QMF Messages and Codes* lists message numbers and text.

The information you see when you issue the HELP command depends on what is on your screen.

From the QMF Home panel:

HELP contains a list of topics about QMF and its commands, and about developing charts with the Interactive Chart Utility through QMF.

From a panel with an error message on it:

HELP contains information about the error message.

For example, to display help for message DSQ20047:

```
HELP DSQ20047
```

From other help panels:

HELP contains information about the displayed panel. There are separate sequences of HELP for these panels:

- QUERY
- PROC
- PROFILE
- REPORT
- All form panels
- Database object list
- Global variable list
- Prompted Query
- Table Editor

IMPORT in CICS

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
				*

The IMPORT command imports objects from the CICS system into QMF temporary storage or into the database. The content of the CICS data queue is copied into the named QMF temporary storage area or into the database.

For MVS, we do not recommend using TSO datasets in CICS. For information on using TSO datasets from QMF 3.1.1, see the chapter on migration in *Installing and Managing QMF for OS/390*.

Note: A QMF administrator can import any object owned by any user.

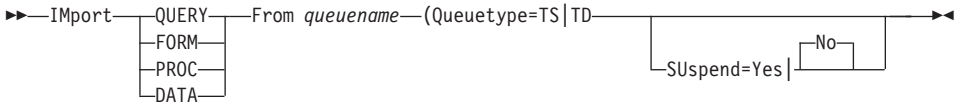
Import a CICS Data Queue into QMF Temporary Storage

You can import queries, forms, procedures, and data into QMF temporary storage. When importing a DATA object, whatever is in the FORM object is replaced by the default form for the imported DATA object.

If you attempt to import an incorrect FORM or DATA object and QMF identifies it as such before the import occurs, an error message is displayed. If you attempt to import an incorrect FORM or DATA object and QMF cannot tell that it is incorrect until the import is in progress, the current contents of the FORM, DATA, and REPORT temporary storage areas are reset to their defaults and the QMF Home panel is displayed.

When importing a DATA object from a CICS TS queue, QMF maintains an ENQ command on the TS queue until the import is completed. You can complete the import by issuing a BOTTOM command. If there is not enough QMF temporary storage to complete the report, the TS queue is considered busy until you reset the data or replace QMF temporary storage with another object.

The IMPORT DATA command processes QMF format data files and IXF files. See “Restrictions on Importing” on page 71.



queuename
The CICS data queue containing the object or data to be imported.

If you are entering the import command on the command line and the queue name is a CICS temporary storage queue containing a character that has special meaning to QMF (such as a period or quote), enclose the entire queue name in single quotes and double all embedded single quotes. Do not enclose the queue name in double quotes. The length of the resulting queue name, including all quotes, cannot exceed ten characters.

Queuetype=TS|TD

Type of CICS data queue containing the QMF object to be imported. This parameter must be specified.

TS Imports the QMF object from a CICS temporary storage queue.

TD Imports the QMF object from a CICS transient data queue.

When importing from a CICS TD queue into QMF temporary storage or into the database, you must use the correct object type for the object currently in the queue. Using an incorrect object type causes QMF to empty the queue without importing the object.

For example, if you issue `IMPORT FORM from MYDATA` but MYDATA actually contains a procedure, QMF detects a mismatch in object types and discards the contents of the queue.

QMF also discards the contents of the queue if any other types of errors are found as QMF imports the object in the queue.

When importing from a CICS TD queue into QMF temporary storage, be sure you have sufficient QMF temporary storage or spill file space to accommodate the object. If you do not have sufficient QMF temporary storage space, QMF discards the contents of the queue and the object is not imported.

Suspend=Yes|No

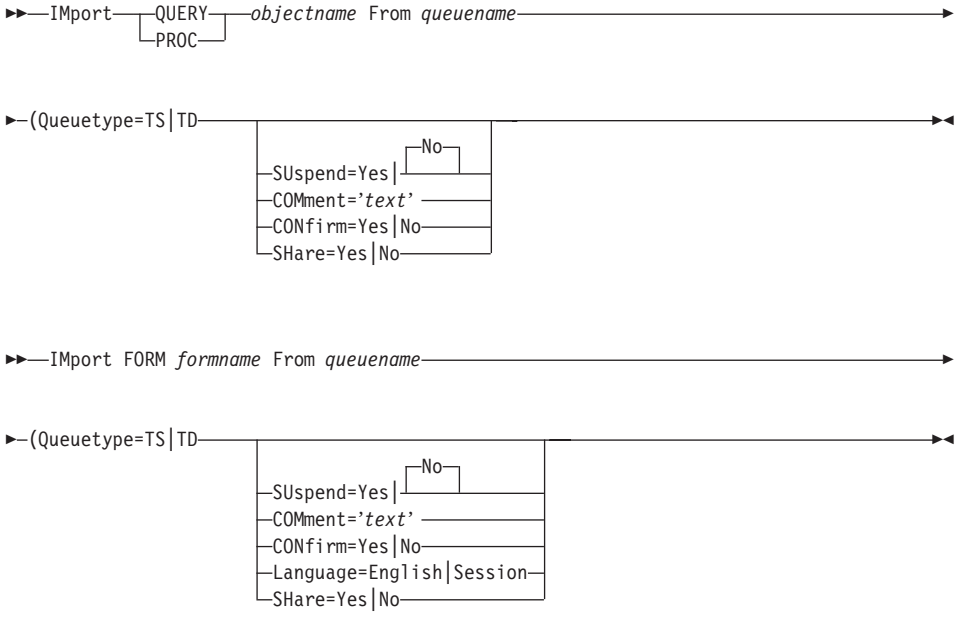
The action taken if the queue name is busy. When the SUSPEND parameter is used, QMF issues a CICS ENQ command for the CICS data queue name. If the results of the CICS ENQ command indicate ENQBUSY, the data queue is considered busy.

Yes Waits until the queue is available.

No If the queue is not available, the import command is canceled and a message is returned. If SUSPEND is not specified, NO is assumed.

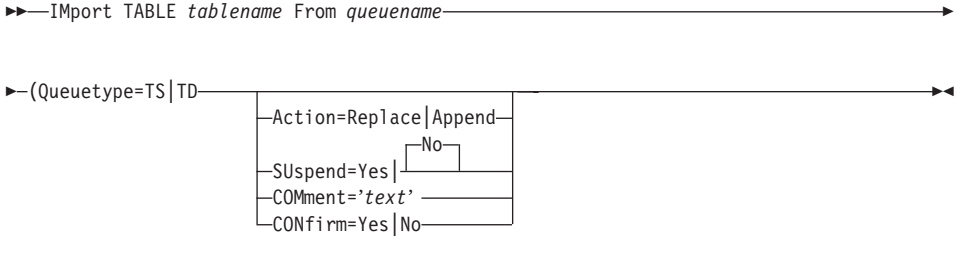
Import a CICS Data Queue into the Database

You can import queries, forms, procedures, and tables into the database.



When you import a form directly into the database, QMF drops any FORM.DETAIL panel variation that is not modified from its default values. In this manner, you can drop unwanted FORM.DETAIL defaults by exporting and then importing the same form.

The IMPORT TABLE command processes QMF format data files and IXF files. See “Restrictions on Importing” on page 71.



objectname
The name of the query or procedure to be imported.

formname
The name of the form to be imported.

tablename
The name of the table to be imported.

When you import a table, an existing object is replaced. If the object does not exist, a new table is created.

When you import a view, the object *must* exist. The contents of the view are replaced.

If your current location is DB2, you can replace or append to an existing table at a remote DB2 location by specifying that location in a three-part name for the table. Existing tables can be replaced at a remote location, but new tables cannot be created unless you first connect to that remote location with the CONNECT command.

queuename
The name of the CICS data queue containing the QMF object to be imported.

If you are entering the import command on the command line and the queue name is a CICS temporary storage queue containing a character that has special meaning to QMF (such as a period or quote), enclose the entire queue name in single quotes and double all embedded single quotes. Do not enclose the queue name in double quotes. The length of the resulting queue name, including all quotes, cannot exceed ten characters.

Queuetype=TS|TD
Type of CICS data queue containing the QMF object to be imported. This parameter must be specified.

TS Imports the QMF object from a CICS temporary storage queue on an auxiliary storage device.

TD Imports the QMF object from a CICS transient data queue.

When importing from a CICS TD queue into QMF temporary storage or into the database, you must use the correct object type for the object currently in the queue. Using an incorrect object type causes QMF to empty the queue without importing the object.

For example, if you issue `IMPORT FORM from MYDATA` but MYDATA actually contains a procedure, QMF detects a mismatch in object types and discards the contents of the queue.

QMF also discards the contents of the queue if any other types of errors are found as QMF imports the object in the queue.

Suspend=Yes | No
Action to be taken if the queue name is busy. When the SUSPEND parameter is used, QMF issues a CICS ENQ command for the CICS data queue name. If the results of the CICS ENQ command indicate ENQBUSY, the data queue is considered busy.

Yes Waits until the queue is available.

No If the queue is not available, the import command is canceled and a message is returned. If SUSPEND is not specified, NO is assumed.

COMment='*text***'**
Allows you to store a comment with any imported object.

If the comment is entered on the command line, enclose the comment in single quotation marks. If the comment contains a single quotation mark, double it by adding another single quotation mark.

If the comment is entered on the prompt panel, do not enclose the comment in single quotation marks.

You cannot replace a comment on a remote table or on a table you don't own.

You can write a comment using only double-byte characters or mixed double-byte and single-byte characters.

If entered on the prompt panel, a comment with only double-byte characters can contain up to 27 double-byte characters. Comments containing mixed double-byte and single-byte characters vary in length depending on the particular combination of characters.

For an explanation of how to calculate the lengths of fields containing both single-byte and double-byte character set data, see *Using QMF*.

CONfirm=Yes | No
Whether a confirmation panel is displayed:

Yes Displays a confirmation panel if an object in the database will be replaced or changed by this command.

No No confirmation panel is displayed.

If this parameter is not specified, the value in your profile is used.

Language=English | Session
Whether a form is imported in English or in the current session language. A QMF form imported in a specific QMF national language can be used without modification only in a session of the same QMF national language.

The default language is determined by the value of the QMF global variable DSQEC_FORM_LANG.

Share=Yes | No
Whether other users are allowed access to the imported object.

Yes Allows every QMF user to use an imported query, form, or procedure.

No Restricts the use of an imported query, form, or procedure to the current user.

If you omit the parameter, the value specified in the global variable DSQEC_SHARE is used for an object being imported for the first time. For an object being replaced, the current value of SHARE is left unchanged.

Action=Replace | Append
Whether to replace the entire database table or to append the imported data to the existing table. ACTION applies only when importing a table.

Examples

- To copy the data queue YOURDATA to the table REYNOLDS.VISIONS:

```
IMPORT TABLE REYNOLDS.VISIONS FROM YOURDATA
   (QUEUETYPE=TS
```

- To copy the data queue QUERY_A to the query REYNOLDS.QUERYA:

```
IMPORT QUERY REYNOLDS.QUERYA FROM QUERY_A
   (QUEUETYPE=TS
```

IMPORT in TSO

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X			*

The IMPORT command imports objects from the TSO system where QMF is executing into QMF temporary storage or into the database. The content of the TSO data set is copied into the named QMF temporary storage area or into the database.

For information about importing objects, see “Restrictions on Importing” on page 71.

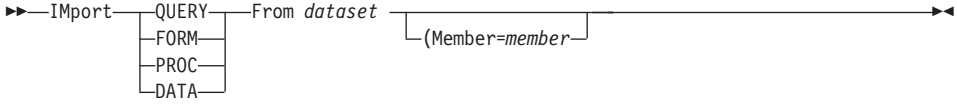
Note: A QMF administrator can import any object owned by any user.

Import a TSO Data Set into QMF Temporary Storage

You can import queries, forms, procedures, and data into QMF temporary storage. When importing a DATA object, whatever is in the FORM object is replaced by the default form for the imported DATA object.

If you attempt to import an incorrect FORM or DATA object and QMF identifies it as such before the import occurs, an error message is displayed. If you attempt to import an incorrect FORM or DATA object and QMF cannot tell that it is incorrect until the import is in progress, the current contents of the FORM, DATA, and REPORT temporary storage areas are reset to their defaults and the QMF Home panel is displayed.

The IMPORT DATA command processes QMF format data files and IXF files. See “Restrictions on Importing” on page 71.



dataset
An external data set in either of two forms:

- Any fully qualified TSO data set name. The entire name must be enclosed in single quotes.
- A partial TSO name. If *dataset* is not enclosed in single quotation marks, your TSO prefix is added as a prefix, and the object type is added as a suffix. (If you have no TSO prefix, your TSO user ID is used. If your prefix is set to blank, no prefix is used.)

Member=*member*
The member name of a TSO partitioned data set. The member name (*member*) is added (in parentheses) as a suffix to the data set name.

If you use the MEMBER parameter, the value of MEMBER is added (in parentheses) as another suffix.

For example, if your user ID is TOM, and you enter:

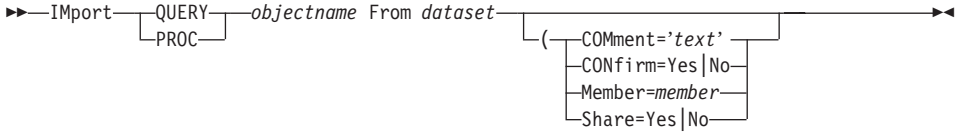
```
IMPORT QUERY FROM ROGER (MEMBER=GAMMA
```

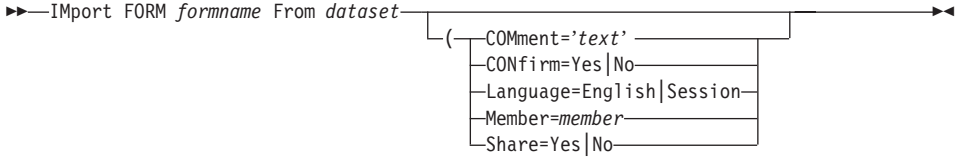
the entire name of the TSO data set imported is:

```
'TOM.ROGER.QUERY(GAMMA)'
```

Import a TSO Data Set into the Database

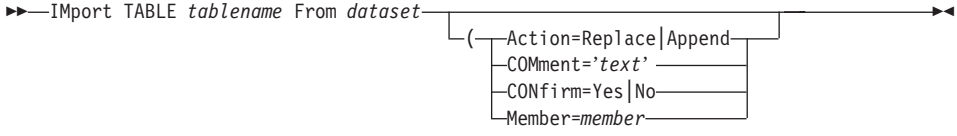
You can import queries, forms, procedures, and tables into the database.





When a form is imported directly into the database, QMF drops any FORM.DETAIL panel variation that is not modified from its default values. In this manner, you can drop unwanted FORM.DETAIL defaults by exporting and then importing the same form.

The IMPORT TABLE command processes QMF format data files and IXF files. See “Restrictions on Importing” on page 71.



objectname
The name of the query or procedure to be imported.

formname
The name of the form to be imported.

tablename
The name of the table to be imported.

When you import a table, an existing object is replaced. If the object does not exist, a new table is created.

When you import a view, the object must exist. The contents of the view are replaced.

If your current location is DB2, you can replace or append to an existing table at a remote DB2 location by specifying that location in a three-part name for the table. Existing tables can be replaced at a remote location, but new tables cannot be created unless you first connect to that remote location with the CONNECT command.

dataset
An external data set in either of two forms. See additional description under "Import a TSO Data Set into QMF Temporary Storage" on page 64.

COMment='*text***'**
Allows you to store a comment with any imported object. If the comment is entered on the command line, enclose the comment in single quotation marks. If the comment contains a single quotation mark, double it by adding another single quotation mark. If the comment is entered on a prompt panel, do not enclose the comment with single quotation marks.

You cannot replace a comment on a remote table or on a table you don't own.

You can write a comment using only double-byte characters or mixed double-byte and single-byte characters. The same rules apply as when using only single-byte characters.

If entered on the prompt panel, a comment with only double-byte characters can contain up to 27 double-byte characters. Comments containing mixed double-byte and single-byte characters vary in length depending on the particular combination of characters.

For an explanation of how to calculate the lengths of fields containing both single-byte and double-byte character set data, see *Using QMF*.

CONfirm=Yes | No
Whether a confirmation panel is displayed:

Yes Displays a confirmation panel if an object in the database will be replaced or changed by this command.

No No confirmation panel is displayed.

If this parameter is not specified, the value in your profile is used.

Language=English | Session
Whether a form is imported in English or in the current session language. A QMF form in a specific QMF national language can be used without modification only in a session of the same QMF national language.

The default language is determined by the value of the QMF global variable DSQEC_FORM_LANG.

Share=Yes | No
Whether other users are allowed access to the imported object.

Yes Allows every QMF user to use an imported query, form, or procedure.

No Restricts the use of an imported query, form, or procedure to the current user.

If you omit the parameter, the value specified in the global variable DSQEC_SHARE is used for an object being imported for the first time. For an object being replaced, the current value of SHARE is left unchanged.

Action=Replace | Append
Whether to replace the entire database table or to append the imported data to the existing table. The Action parameter applies only when importing a table.

Examples

- To copy the data set YOURDATA to the table REYNOLDS.VISIONS:

```
IMPORT TABLE REYNOLDS.VISIONS FROM YOURDATA
```

- To copy the data set QUERY_A to the query REYNOLDS.QUERYA:

```
IMPORT QUERY REYNOLDS.QUERYA FROM QUERY_A
```

- To import a table to a remote database that does not support three-part names, first connect to that database:

```
CONNECT TO VENICE
```

  then import the table:

```
IMPORT TABLE JEAN.STATSTAB FROM YOURDATA
```

- To copy the data set YOURDATA from the DB2 location where QMF is executing to the table OKAMOTO.STATUS at the TOKYO location:

```
IMPORT TABLE TOKYO.OKAMOTO.STATUS FROM YOURDATA
```

  OKAMOTO.STATUS must exist at TOKYO.

IMPORT in CMS

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
		X	X	

The IMPORT command imports objects from the location where QMF is executing into QMF temporary storage or into the database. The content of the CMS file is copied into the named QMF temporary storage area or into the database.

IMPORT in CMS

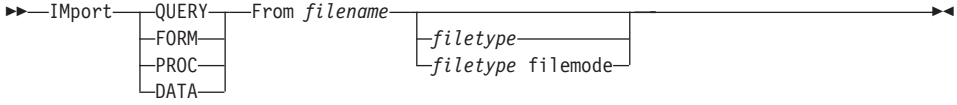
The IMPORT DATA command processes QMF format data files and IXF files. See "Restrictions on Importing" on page 71.

Note: A QMF administrator can import any object owned by any user.

Import a CMS File into QMF Temporary Storage

You can import queries, forms, procedures, and data into QMF temporary storage. When importing a DATA object, whatever is in the FORM object is replaced by the default form for the imported DATA object.

If you attempt to import an incorrect FORM or DATA object and QMF identifies it as such before the import occurs, an error message is displayed. If you attempt to import an incorrect FORM or DATA object and QMF cannot tell that it is incorrect until the import is in progress, the current contents of the FORM, DATA, and REPORT temporary storage areas are reset to their defaults and the QMF Home panel is displayed.



filename, filetype, filemode
The CMS file to be copied.

filename is required.

filetype can be omitted. (If the file type is not specified, the object type is assumed.)

filemode can be omitted. (If file mode is not specified, A is assumed.)

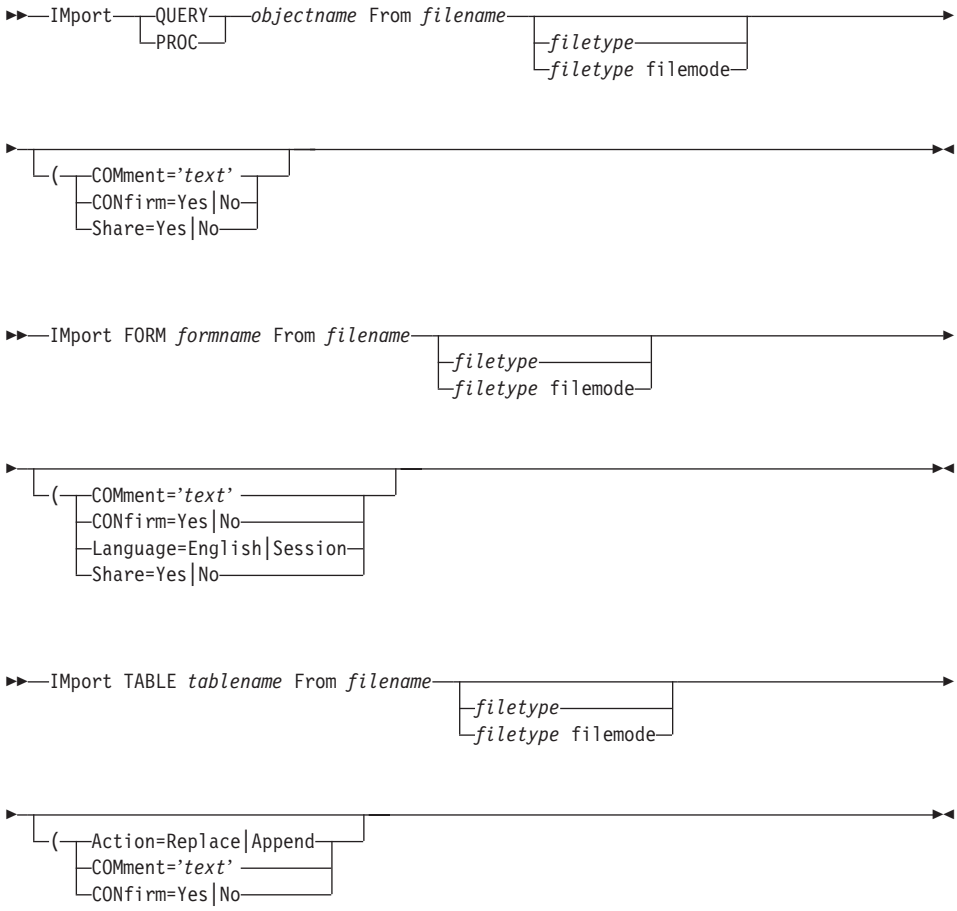
You can use an asterisk (*) instead of *filemode*. This tells CMS to search through your accessed disks in the usual order for the first occurrence of a file with the given *filename* and *filetype*.

When a form is imported directly into the database, QMF drops any FORM.DETAIL panel variation that is not modified from its default values. In this manner, you can drop unwanted FORM.DETAIL defaults by exporting and then importing the same form.

Import a CMS File into the Database

You can import queries, forms, procedures, and tables into the database.

The IMPORT TABLE command processes QMF format data files and IXF files. See "Restrictions on Importing" on page 71.



objectname
The name of the query or procedure to be imported.

formname
The name of the form to be imported.

tablename
The name of the table to be imported.

When you import a table, an existing object is replaced. If the object does not exist, a new table is created.

When you import a view, the object must exist. The contents of the view are replaced.

filename, filetype, filemode
The CMS file to be imported.

filename is required.

filetype can be omitted. (If the file type is not specified, the object type is assumed.)

filemode can be omitted. (If file mode is not specified, A is assumed.)

COMment='*text***'**
Allows you to store a comment with any imported object.

If the comment is entered on the command line, enclose the comment in single quotation marks. If the comment contains a single quotation mark, double it by adding another single quotation mark.

If the comment is entered from a prompt panel, do not surround it with single quotation marks.

You cannot replace a comment on a remote table or on a table you don't own.

You can write a comment using only double-byte characters or mixed double-byte and single-byte characters. The same rules apply as when using only single-byte characters.

If entered on the prompt panel, a comment with only double-byte characters can contain up to 27 double-byte characters. Comments containing mixed double-byte and single-byte characters vary in length depending on the particular combination of characters.

For an explanation of how to calculate the lengths of fields containing both single-byte and double-byte character set data, see *Using QMF*.

CONfirm=Yes | No
Whether a confirmation panel is displayed:

Yes Displays a confirmation panel if an object in the database will be replaced or changed by this command.

No No confirmation panel is displayed.

If this parameter is not specified, the value in your profile is used.

Language=English | Session
Whether a form is imported in English or in the current session language. A QMF form imported in a specific QMF national language can be used without modification only in a session of the same QMF national language.

The default language is determined by the value of the QMF global variable DSQEC_FORM_LANG.

Action=Replace | Append
Whether to replace the entire database table with the imported data or to append the imported data to the existing table. The Action parameter applies only when importing a table.

Share=Yes | No
Whether other users are allowed access to the imported object.

Yes Allows every QMF user to use an imported query, form, or procedure.

No Restricts the use of an imported query, form, or procedure to the current user.

If you omit the Share parameter, the value specified in the global variable DSQEC_SHARE is used for an object being imported for the first time. For an object being replaced, the current value of SHARE is left unchanged.

Examples

- To copy the file YOURDATA to the table WILDE.VISIONS:

```
IMPORT TABLE WILDE.VISIONS FROM YOURDATA
```

- To copy the file QUERY_A to the query WILDE.QUERYA:

```
IMPORT QUERY WILDE.QUERYA FROM QUERY_A
```

- If your current location is a DB2 database, you can copy the file YOURDATA from the system where QMF is executing to an existing table, OKAMOTO.STATUS, at the DB2 database at the TOKYO location:

```
IMPORT TABLE TOKYO.OKAMOTO.STATUS FROM YOURDATA
```

- To import a table to a remote database that does not support three-part names, first connect to that database using remote unit of work:

```
CONNECT TO VENICE
```

  then import the table:

```
IMPORT TABLE JEAN.STATSTAB FROM YOURFILE
```

Restrictions on Importing

The following section describes the restrictions that apply when you import data into CMS, TSO, or CICS.

IXF Format Restrictions

QMF supports the host IXF format defined by DXT. Other forms of IXF (for example, PC IXF) are not supported. You must modify the input file to conform to the IXF format defined in *Developing QMF Applications*.

Other Restrictions

You can create or change procedures, SQL queries, prompted queries, forms, and data outside QMF and import them. However, if you attempt to use a QBE query created or changed outside QMF, the QMF program can be terminated.

If an object exists in the database with the same *objectname*, the IMPORT command replaces the object, subject to these conditions:

- FORM can replace only a form, PROC only a procedure, and QUERY only a query.
- DATA can replace only a table or view. The existing table or view and DATA must have the same number of columns, and corresponding columns must have the same data types and data type lengths.
- If your current location is a DB2 database, you can replace or append to an existing table at a remote DB2 location by specifying that location in a three-part name for the table. However, new tables cannot be created unless you first connect to that remote location with the CONNECT command.
- The existing labels and column names of the table or view being replaced remain unchanged.

If you import a QMF procedure or SQL query that was created or changed while it was outside QMF, be aware of the following restrictions:

- Each line in a procedure or SQL query is 79 characters long; each line imported occupies one record. QMF does not reject records longer than 79 characters. However, it truncates the data and issues a warning message.
- IMPORT pads with blanks any line that is shorter than 79 characters. This can cause problems if the padding takes place within a character string enclosed in quotation marks.
- IMPORT does not check an imported procedure or query for printable characters. Therefore, it is possible to import characters into PROC or QUERY that cannot be displayed.

QMF searches for global variables on a case-sensitive basis. If you import a form that includes lowercase or mixed-case global variables, be sure the global variables are defined in your SET GLOBAL command exactly as you use them in the form.

For example, this SET GLOBAL command will identify the lowercase global variables in the form:

```
SET GLOBAL (x1=salary x2=dept
```

To log any warning messages you might receive while importing a form or prompted query, issue the following command *before* issuing the IMPORT command.

```
SET PROFILE (TRACE=L1
```

The messages are logged into QMF trace data. For more information about how to set the trace, see "SET PROFILE" on page 116. QMF Does not support binary data in DBCS mode. An error message is returned for IMPORT DATA or IMPORT TABLE statements when character data has the subtype BIT.

INSERT

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The INSERT command inserts any of the following:

- One line at a time into an SQL query or procedure
- A line into a Prompted Query panel
- A line of column information into a FORM.MAIN or FORM.COLUMNS panel
- A calculation line into a FORM.CALC or FORM.CONDITIONS panel
- A line of text into a FORM.PAGE, FORM.FINAL, FORM.BREAK*n*, or FORM.DETAIL panel

For example, to insert a line at the top of the scrollable area of a prompted query, position the cursor directly above the first line and press the Insert function key.

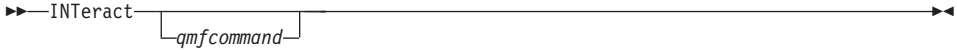
To insert a calculation line into a FORM.CALC panel, position the cursor on the line above where you want the added line, and press the Insert function key or type INSERT on the command line.

INTERACT

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

INTERACT

Use the INTERACT command from an application when you want QMF to prompt you. The INTERACT command cannot be entered from the command line.



qmfcommand
The QMF command to be run.

When an application uses the command interface to send commands through ISPF to QMF, prompts do not appear unless the command is preceded by INTERACT.

INTERACT by itself starts an interactive QMF session. The first screen displays what would be shown if the previous command were entered interactively instead of from an application. This interactive session remains active until you issue the END command. When the INTERACT command ends, control returns to the application or procedure.

For a complete discussion of the INTERACT command, see *Developing QMF Applications* .

ISPF

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X		X		

ISPF is a QMF-supplied command synonym that calls the Interactive System Product Facility (ISPF).

You must be authorized to use ISPF; consult your QMF administrator.

parameter
Which ISPF parameter to pass to ISPF. For example, if you enter 3, the third ISPF panel is displayed, or the ISPF application is called, rather than displaying the ISPF main menu.

If you do not specify a parameter, the ISPF primary option menu is displayed.

ISPF can be used from the command line, a function key, a procedure, or while invoking QMF through the QMF command interface, when preceded by the INTERACT command. It cannot be used while operating QMF in batch mode. After completion of the ISPF session, you are returned to the QMF panel from which you entered the ISPF command.

LAYOUT

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X		X		

LAYOUT is a QMF-supplied command synonym that lets you create a sample report when the data for that report is not yet available. LAYOUT also lays out constant text added in the form as it will appear. You can then see how your report will look before you put the data into it.

FORM
Creates a report using the form in temporary storage.

formname
Creates a report using the form in the database with the name *formname*.

FORM *formname*
Creates a report using the form in the database with the name *formname*.

After you develop a form that contains the specifications you plan to use in your report, there are several ways to use the LAYOUT command to generate a sample report. For example:

- To run the LAYOUT command using the form temporary storage area:

  ```
  LAYOUT FORM
  ```

- To run the LAYOUT command using the form in the database named MYFORM:

  ```
  LAYOUT MYFORM or LAYOUT FORM MYFORM
  ```

You can use your sample form to display a report with various characters representing the data. If there are no breaks in the report, the following characters are displayed:

X Character data

0 Numeric data

If the report contains breaks, the levels are shown using the following characters:

A Character data in first break

1 Numeric data in first break

B Character data in second break

2 Numeric data in second break and so on.

After you see what your form will look like, you can make changes to it without running a query.

The LAYOUT command creates and imports its data in QMF (binary) data format. This format is described in *Developing QMF Applications*.

LEFT

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The LEFT command scrolls toward the left margin of a report or a QBE query.

Csr Scrolls the column where the cursor is positioned to the left of the scrollable area.

Half Scrolls left half the width of the scrollable area or to the left margin if that is closer.

Max Scrolls to the left margin of the scrollable area.

Page Scrolls left the depth of the scrollable area or to the left margin if that is closer.

n Scrolls to the left n columns. (n can be any number from 1 through 9999.)

You can use the Left function key to scroll left in a report, or type the number of columns you want to scroll left on the command line, and then press the Left function key.

If you do not specify a scroll amount, QMF uses the amount shown after SCROLL ===> in the bottom right corner of the screen. You can change the amount by typing a new amount over it. The change remains in effect throughout the session, except for Max, which remains in effect for only the next command.

You can also change the scroll amount QMF uses by setting the global variable DSQDC_SCROLL_AMT to Csr, Half, Page, or any number of columns (*n*) up to 9999.

LIST

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The LIST command displays a list of queries, forms, procedures, local tables, or all the qualified QMF objects you are authorized to use at the specified location.

You can display a list from any QMF panel by issuing the LIST command and specifying an object type, QMF, or ALL. The first time you issue the LIST command in a QMF session, you must specify an object type, QMF, or ALL. An error message appears if you enter the LIST command with one or more parameters and do not specify an object type.

The most recently generated object list remains in temporary storage until you run the LIST command again or end the QMF session. If the QMF panel currently displayed results from issuing a command from the object list panel, you can return to your original object list panel by pressing the End function key.

For example, if you list all the tables you own, then display one of those tables, you can press the End function key to return to the list of tables you own.

Issuing the LIST command when operating in QMF batch mode results in an error. Also, to issue LIST through the QMF command interface, LIST must be preceded by INTERACT (unless it is used from a Prompted Query dialog panel).

Selection Symbols in the LIST Command

The % symbol and the _ symbol can be used to generate a list of like values.

The percent sign (%) is used to represent a string of any length and containing any characters. For example, to list all object names that contain the character J, use:

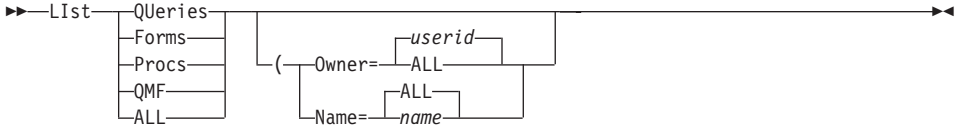
```
NAME=%J%
```

The underscore (_) is used to select objects that contain a specific character in a particular position. For example, to list all object owners that contain the character K in the second position, use:

```
OWNER=_K%
```

You can use selection symbols with *userid* and *name*. However, results are unpredictable when you use % with *userid*.

List Queries, Forms, or Procedures



Queries
Lists only queries.

Forms Lists only forms.

Procs Lists only procedures.

QMF Lists queries, forms, and procedures.

ALL Lists queries, forms, procedures, tables and views.

An object type (queries, forms, or procedures) or ALL must be specified the first time the LIST command is used in a session.

Owner=*userid* **| ALL**
Lists objects owned by the named *userid*, or lists all objects you own, plus any objects saved with SHARE=YES. If you don't specify OWNER, only the objects you own are listed.

Name=*name* **| ALL**
Lists objects with the *name* specified, or lists all objects.

The OWNER and NAME parameters can be listed in any order. The OWNER parameter produces different results depending on whether you are listing

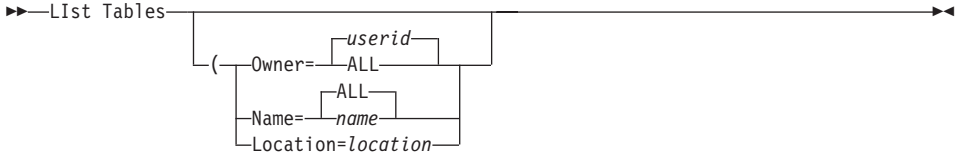
tables and views or other objects. An error message appears if LIST is entered with one or more parameters and no object type is specified.

When listing queries, forms, or procedures, the names displayed are those selected by the OWNER and NAME parameters of the command. In addition:

- If the current location is DB2, the names displayed are those owned by the current SQL authorization ID, or those saved with SHARE=YES. If the current SQL authorization ID is different from the primary authorization ID, objects owned by the primary authorization ID and saved with SHARE=NO are not displayed. These objects are not accessible when the current SQL authorization ID is different from the primary authorization ID.
- If the current location is SQL/DS, the names displayed are those owned by the SQL/DS user ID, or those saved with SHARE=YES.
- You cannot list queries, procedures, or forms at a remote location using the *location* parameter. To list these objects at a remote location, first connect to that location, then use the LIST command.

List Tables and Views

The list of tables, views, and aliases might be tailored for your installation. If so, the rules governing what objects you will see can be different from those given here.



Tables
Lists tables and views.

Owner=*userid* **| ALL**
Lists tables and views created by the named *userid*, or all the tables and views to which you are authorized. Aliases are also listed when your location is DB2. If you don't specify OWNER, only the tables and views you own are listed.

Name=*name* **| ALL**
Lists tables and views with the *name* specified, or lists all tables and views.

Location=*location*
Displays a list of tables and views at a remote location; *location* can be used only if the current location is DB2. Also, *location* must be a database system that supports three-part names (DB2 for MVS Version 2.2 or later).

To list tables at a remote location if the current location is not DB2, first connect to that location, then use the LIST command. If you specify LIST ALL and include a location value, a list of tables, views, and aliases is displayed with a message stating: `Only tables, views, and aliases have been listed.`

If the current location is DB2, the tables and views listed are those that are authorized for access by PUBLIC or PUBLIC AT ALL LOCATIONS, and by the primary or current authorization ID. If the current location is SQL/DS, the tables and views listed are those that are authorized for access by PUBLIC and the SQL/DS user ID. If aliases are supported by your database, the list also includes aliases owned by your primary or current authorization IDs.

- Tables are indicated by a T or an R in the "Object SubType" field of the list.
- Views are indicated by a V.
- Aliases are indicated by an A.

You can display the LIST command prompt panel by entering `LIST ?` on the command line. Press Enter to accept defaults (Type=ALL, Owner=*userid*, Name=ALL), or change the defaults by typing over them.

When you request a list of tables, QMF uses views to retrieve the information:

- If your current location is DB2 and you request a list from that location (if LOCATION is not specified or is specified as the current location), QMF uses the view named in the global variables DSQEC_ALIASES and DSQEC_TABS_LDB2.
- If your current location is DB2 and you request a list from a different DB2 location, QMF uses the view named in the global variables DSQEC_ALIASES and DSQEC_TABS_RDB2.
- If your current location is SQL/DS, QMF uses the view named in the global variable DSQEC_TABS_SQL.

For more information on these views, see *Managing QMF* for your operating system.

List Tables in Prompted Query

You can list tables in Prompted Query by pressing the List function key. A scrollable list of tables is displayed from which you are prompted to make selections. You can use the symbols described on page 78 to limit the search to tables with the specified characters.

The LIST command can also be used in the Expression dialog panel and when specifying values for row conditions. Pressing the List function key displays a list of the columns for the table or tables you selected. From this list, you can type the number of the column you want to use in your expression.

Display the Database Object List

The database object list contains information about the selected objects. To display the database object list, enter the LIST command with ALL specified. This displays only objects you are authorized to see. Enter the LIST command with QMF specified to display all objects, except tables, you are authorized to see. If your DB2 database supports secondary authorization IDs, the list also includes objects for which privileges were granted to your current SQL authorization ID. To display a previous database object list (assuming a list has been generated), enter the LIST command with no *name* specified.

If you connected to a new location since you created the object list being displayed, your list is now obsolete. You must refresh the list, or cancel it and create a new one. Commands issued in the **Action** column of an obsolete list are not executed.

```
                  Object List

                                          --------Dates--------
Action   Name               Owner    Type  Modified   Last Used
                                              n to n of n
         xxxxxxxxxxxxxxxxxx xxxxxxxx xxxxx yyyy-mm-dd yyyy-mm-dd
         xxxxxxxxxxxxxxxxxx xxxxxxxx xxxxx yyyy-mm-dd yyyy-mm-dd
         xxxxxxxxxxxxxxxxxx xxxxxxxx xxxxx yyyy-mm=dd yyyy-mm-dd
_____________________________________________________________
_
F1=Help     F4=Command  F5=Describe    F6=Refresh   F7=Backward
F8=Forward  F9=Clear    F10=Comments   F11=Sort     F12=Cancel
```

You can issue QMF commands from this panel by entering the command under the **Action** column. You cannot issue commands from a database object list that was not created at the current location.

See “Function Key Commands” on page 83 for actions you can take using function keys from this panel.

Examples of LIST

To generate and display a new object list that contains all the queries that begin with the name APP1 regardless of owner:

```
LIST QUERIES (OWNER=ALL,NAME=APP1%
```

To generate and display a new object list containing all the forms you own:

```
LIST FORMS
```

To generate and display a new object list containing all the tables belonging to the current ID at the remote location DALLAS:

```
LIST TABLES (LOCATION=DALLAS
```

QMF Commands Entered in the Action Column

The QMF command area is at the far left of the QMF Database Object List. QMF commands typed next to the name of the object are run for that object. Parameters can be specified immediately following the command. You can type a command across the entire width of the screen, overlaying portions of the list. Use a slash (/) in place of the object type and the object name when you key a command in the **Action** column. For example, entering:

```
RUN / (FORM=MYFORM
```

next to a query results in the following QMF command:

```
RUN QUERY ownerid.name (FORM=MYFORM
```

If the place-holder character is not specified, the object type and name are placed at the end of the command.

An asterisk is placed before the command name on the line next to the object after a command runs successfully.

These QMF commands can be entered in the **Action** column:

CONVERT
Converts an SQL, QBE, or relational prompted query to an SQL query with standard syntax.

DISPLAY
Retrieves an object from the database and places it in temporary storage. To return to your object list after displaying an object, press F3.

EDIT Opens a Table Editor session for a table when entered next to a table name. EDIT must be the last command entered on the database object list. You cannot enter EDIT next to a query or procedure because you can only edit queries or procedures in temporary storage, not from the database.

ERASE
Deletes an object from the database.

EXIT Ends your QMF session. If the LIST command is run as part of a procedure that starts your QMF session, you must use EXIT to leave QMF.

EXPORT
Exports an object from the database to a TSO data set, a CMS file, or a CICS data queue.

IMPORT
Imports an object into the database from a TSO data set, a CMS file, or a CICS data queue.

PRINT
Prints an object in the database.

QMF If your installation has defined any commands with the same name as a QMF command, you can precede the QMF command with QMF to ensure that you get the expected results. See "QMF" on page 95.

RUN Run a query or procedure stored in the database.

Function Key Commands

Function keys are used in processing the database object list. Unless otherwise stated, QMF commands entered on the command line are run before function key commands are processed. The following function keys are supported:

Help Provides information about the database object list. You can run any pending commands in the Action column after you return to the database object list from the help panel.

Command
Provides a panel for entering QMF commands. This function key is accepted only when there is an action column on the panel.

Describe
With the cursor on the desired object name, pressing this key displays the description panel.

Refresh
Obtains a new list from the database and clears existing commands and feedback from the action entry area. This function key is accepted only when there is an action column on the panel.

Backward
Scrolls the object list backward. If QMF commands are pending in the QMF command area, they are run after the scrolling completes.

Forward
Scrolls the object list forward. If QMF commands are pending in the QMF command area, they are run after the scrolling completes.

Clear Clears existing commands and feedback from the action entry area. This function key is accepted only when there is an action column on the panel.

Comments
The beginning comments are appended to the primary selection panel. If the current panel has comments, pressing this key toggles between

the comments and the Date Modified and Date Last Used fields. Pressing this key also removes existing commands and feedback from the action entry area.

Comments *cannot* be entered directly on the LIST panel.

Comments entered on a query or procedure panel can be viewed only when the query or procedure is displayed; they are not available on the LIST panel.

If QMF commands are pending in the QMF command area, the Comments field is not displayed, and this function key runs the pending commands.

Sort Sorts the object list by object name within object owner or within object type, or by modified date, last used date, or type. You can set a default sort order with the global variable DSQDC_LIST_ORDER. Note that the last used date may not be updated, for performance reasons, when using a QMF object while the current QMF report is not yet complete.

Cancel

Returns control to the QMF object panel from which the LIST command was issued. The object list panel, including any pending QMF commands, is preserved “as is”.

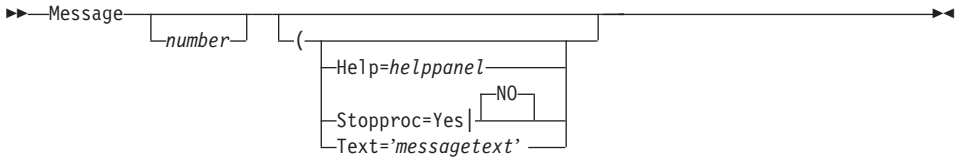
MESSAGE

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	*	X	*	*

The MESSAGE command:

- Displays a message from the ISPF library
- Generates QMF-like messages when an application ends
- Assigns a Help panel for a message generated by a command
- Suppresses the execution of QMF linear procedures

You can issue the MESSAGE command from the QMF command interface or from a procedure.



number
The identification number of a message definition in an ISPF message library. The number is optional. The designated library must be concatenated to your ISPMLIB file. If ISPF is not available, *number* is not allowed.

Help=*helppanel*
The panel identification number of a panel that appears as help for a message. If ISPF is not available, HELP=*helppanel* is not allowed.

Stopproc=Yes | No
A termination switch for linear procedures.

Text='*messagetext***'**
The text of the message.

Use MESSAGE from user applications (procedures, programs, EXECs, CLISTs) to pass a message to the QMF message area.

For a complete discussion of the MESSAGE command, see *Developing QMF Applications* .

NEXT

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The NEXT command:

- Navigates forward through the set of variations associated with the FORM.DETAIL panel.
- Displays the next column or the next definition from the Column Definition or Column Alignment panel.
- Displays the next row in the set of accessed rows in the Table Editor.

Column
Displays the next column from the Column Definition or Column Alignment panel.

Definition
Displays the next column with a nonblank definition expression from the Column Definition panel.

The Column and Definition parameters:

- Direct panel navigation while a Column Alignment or Column Definition panel is active.
- Can be entered using a function key or from an application.

On a FORM.DETAIL panel, the NEXT command:

- Displays the next panel variation (unless you are currently at the last variation and it wasn't modified).
- Can be entered from the command line or from an application.

PREVIOUS

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The PREVIOUS command:

- Navigates backward through the set of variations associated with the FORM.DETAIL panel.
- Displays the previous column or the previous definition from the Column Definition or Column Alignment panel.
- Displays the row just added (Add Mode) or the most recent successful search criteria (Search Mode) when using the Table Editor.

Column
Displays the previous column from the Column Definition or Column Alignment panel.

Definition
Displays any preceding column with a nonblank definition expression from the Column Definition panel.

The Column and Definition parameters:

- Direct panel navigation while a Column Alignment or Column Definition panel is active.
- Can be entered using a function key or from an application.

On a FORM.DETAIL panel, the PREVIOUS command:

- Displays the previous panel variation, unless you are currently at the first variation.
- Can be entered from the command line or from an application.

In the Table Editor, the PREVIOUS command can only be entered using a function key.

PRINT

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The PRINT command prints a copy of an object in the QMF temporary storage area or an object stored in the database (at the current location).

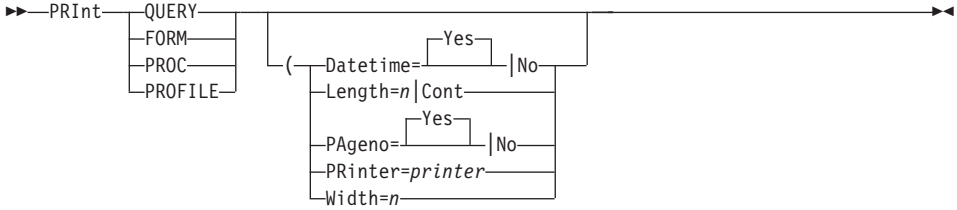
The rules for printing QMF objects vary depending on the type of object you are printing and the operating system you are using. See “Printing QMF Objects” on page 279 for more information.

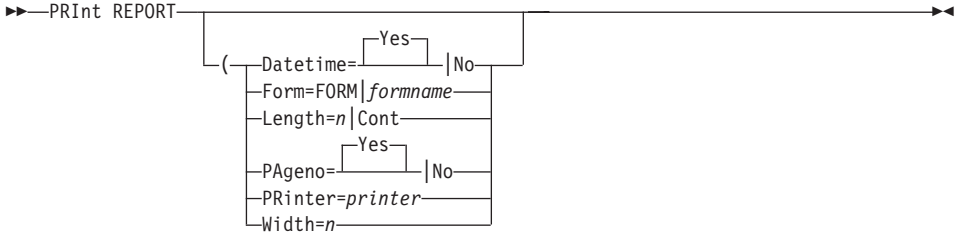
Notes for DBCS users: With a DBCS printer, you can print reports containing DBCS data even if you do not have a terminal that displays DBCS data. This is possible by invoking QMF in a special way. See your QMF administrator center about how to do this.

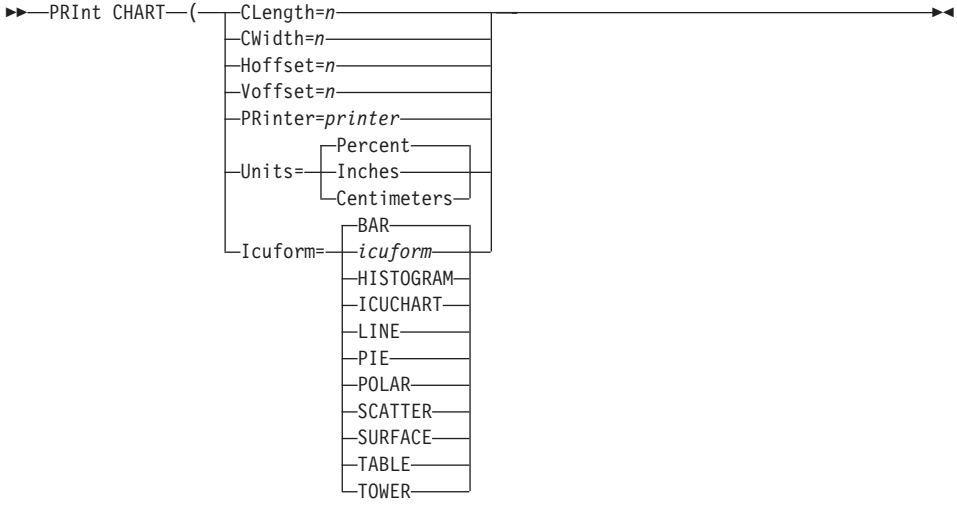
If you are using DBCS data and QMF splits the page, printing resumes on the second and subsequent pages of the report at the fourth byte position from the left side of the page.

Printing in CMS and TSO from QMF Temporary Storage

You can print queries, forms, procedures, profiles, reports, and charts from temporary storage.







Datetime=Yes|No
Controls the printing of the system date and time in the page footing. When you print the report, the date and time are printed in the page footing *unless* you use DATETIME=NO *or* specify &DATE and &TIME in the page text on FORM. If DATETIME=YES, the date and time are printed in the default format assigned by your installation.

Length=*n*|Cont
The maximum number of lines to print on any page.

QMF has no way to check that the paper in your printer is as long as the value you choose for LENGTH.

n Can range from 1 to 999. If this parameter is omitted, the value of Length in your profile is used. Use the following minimum lengths when printing:

- 25 for FORM, SQL queries, prompted queries, and procedures
- 7 for PROFILE and QBE queries
- A number that is *at least* 2 greater than the total number of lines needed for the column headings, detail headings, and page headings and footings that appear on each page of a report, table, or view

You can print a profile or QBE query to DSQPRINT DDNAME, rather than to a GDDM printer, with a minimum length of 5.

The maximum number of lines that will print for FORM is 66, because of GDDM restrictions. For more information on GDDM support for the QMF PRINT command, see *Managing QMF* for your operating system

CONT

Causes continuous printing without page breaks. It is not accepted when printing forms or prompted queries or when you specify a printer name.

PAgeno=Yes | No

Controls the printing of page numbers. To suppress them, specify No. When the form contains &PAGE, PAGENO doesn't control page numbers in the report.

PRinter=*printer*

The nickname of the GDDM printer to be used for output.

The first character of *printer* must be alphabetic.

When entering this parameter on the command line in MVS or VM, you can specify a blank (' ') as the printer nickname to send the output to a file, data set, or printer (depending on what is specified by the ddname or FILEDEF). If `Printer=' '` is specified on a PRINT REPORT command, DSQPRINT DDNAME is assumed. The attributes of record format (RECFM) and record length (LRECL) for DSQPRINT DDNAME can be found in *Installing and Managing QMF on VM/ESA* or *Installing and Managing QMF on OS/390*.

If you do not specify the PRINTER parameter, QMF uses the printer listed in your profile.

You must specify a printer when printing a form or prompted query. Otherwise, QMF uses the value for PRINTER in your profile. If the value for PRINTER in your profile is blank, an error message is displayed.

Width=*n*
The maximum number of characters to print on any line.

QMF has no way to check that the paper in your printer is as wide as the value you choose for WIDTH.

n can range from 22 to 999 for single-byte characters and from 10 to 498 for double-byte characters. For mixed single-byte and double-byte characters, the maximum print width varies depending on the particular combination of characters. For more information on how to calculate the width of mixed fields, see the chapter on double-byte character set data in *Using QMF* . If WIDTH is omitted, the value of width in your profile is used.

When you print a report, lines longer than WIDTH are formatted on a subsequent page unless you've specified line wrapping on FORM.OPTIONS. When you print other objects, such as FORM or QUERY, lines longer than WIDTH are cut off at the right and the excess is lost.

Form=FORM|*formname*
Only accepted with PRINT REPORT. *formname* is the name of a form in the database. FORM refers to the form currently in temporary storage.

If you specify `FORM=formname`, the named form in the database is loaded into temporary storage and is used as the form when printing the report.

If you don't specify the FORM parameter, the form in temporary storage is used when printing the report.

CLength=*n*
The length of the chart area.*n* can be expressed in percent, inches, or centimeters (the value specified in the UNITS parameter). If UNITS is percent, the default for CLENGTH is 95. If UNITS is inches or centimeters, the default for CLENGTH is 6.

CWidth=*n*
The width of the chart area.*n* can be expressed in percent, inches, or centimeters (the value specified in the UNITS parameter). If UNITS is percent, the default for CWIDTH is 95. If UNITS is inches or centimeters, the default for CWIDTH is 6.

Hoffset=*n*
The horizontal offset from the left side of the page to the left edge of the chart area. *n* can be expressed in percent, inches, or centimeters (whatever is specified in the UNITS parameter). If HOFFSET is not specified, zero is assumed.

Voffset=*n*
The vertical offset amount from the top of the page to the top of the chart area. *n* can be expressed in percent, inches, or centimeters (whatever is specified in the UNITS parameter). If VOFFSET is not specified, zero is assumed.

Units=Percent | Inches | Centimeters
The unit of measure to be used for CLENGTH, CWIDTH, VOFFSET, and HOFFSET. The value assigned to UNITS can be PERCENT, INCHES, or CENTIMETERS. If UNITS is not specified, PERCENT is assumed.

Icuform=*icuform*
The name of an ICU chart format saved in the ICU (*icuform*), or one of the chart formats QMF provides.

If no chart format is specified when the PRINT CHART command is entered, QMF uses a default chart format named DSQCFORM. (If DSQCFORM was not changed by you or your installation to another chart type, it is a bar chart.)

You can also specify ICUCHART, which is the ICU default chart format, when you use the ICUFORM parameter.

PAGENO and DATETIME cannot be specified as parameters with the PRINT CHART command. The page number, date, and time can be included in the chart title by specifying &PAGE, &DATE, and &TIME, respectively, on the FORM.PAGE panel.

When you print a report, the report is printed according to the form specifications.

If you enter anything other that CHART (such as QUERY) on the PRINT command prompt, another command prompt appears so that you can specify parameters.

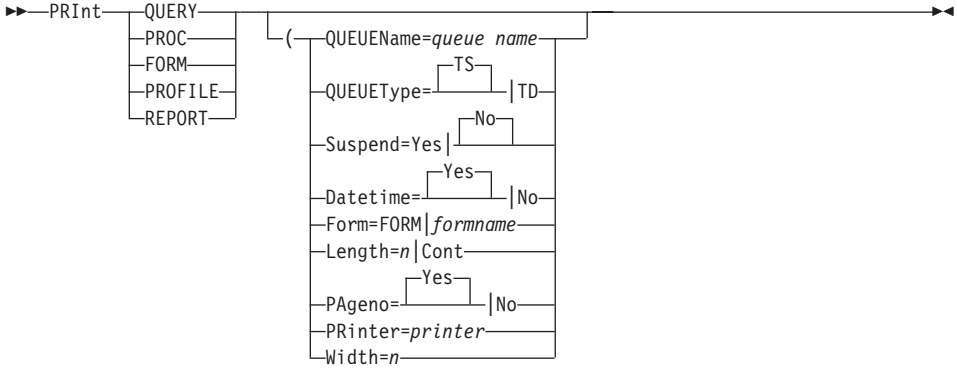
If CHART is entered on the PRINT command prompt, the PRINT CHART command prompt appears so that you can specify the appropriate parameters needed to print your chart.

When you print a chart, the form specifications are applied to the data and the chart is formatted by the GDDM Interactive Chart Utility.

If you are printing a report or chart, and the form is incompatible with the data or contains errors, the first form panel containing an error is displayed with the error highlighted. To see any other errors, you must correct the first error that is displayed.

Printing in CICS from QMF Temporary Storage

To print an object from temporary storage using CICS, specify the name of a query, procedure, profile, or report.



QUEUEName=*queue name*
The CICS data queue to contain the QMF object being printed.

QUEUEType=TS|TD
The type of CICS storage used to contain the QMF object.

TS Writes the QMF object to a CICS temporary storage queue on an auxiliary storage device.

TD Writes the QMF object to a CICS transient data queue.

Suspend=Yes|No
The action taken if the print queue is busy. When the SUSPEND parameter is used, QMF issues a CICS ENQ command for the CICS data queue name. If the results of the CICS ENQ command indicate ENQBUSY, the data queue is considered busy.

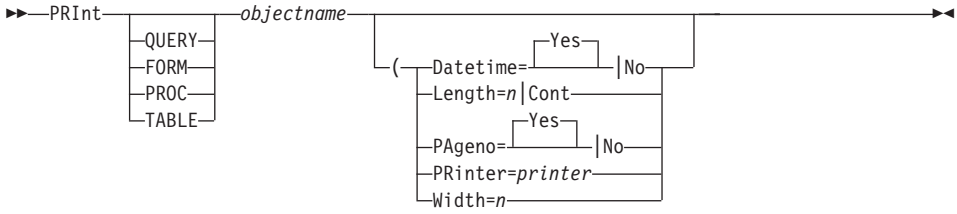
Yes The print command waits until the queue is available, then writes to it.

No Cancels the print command and returns a message to the user.

The descriptions of the other parameters are the same as for “Printing in CMS and TSO from QMF Temporary Storage” on page 87.

Printing in CMS and TSO from the Database

To print an object from the database, specify the name of a query, form, procedure, or table. The object type is optional.



objectname
The name of the object to be printed.

The descriptions of the parameters are the same as under "Printing in CMS and TSO from QMF Temporary Storage" on page 87.

When you print a form, all the parts of the form are printed at the same time.

When you specify a table, the table is printed using the default form. If the table or view is too large to fit on the page, the data is truncated. If column labels exist for the table or view, they appear as column headings when the table or view is printed.

To print a table or view with other than the default form, use these commands:

```
DISPLAY tablename
PRINT REPORT (FORM= formname
```

If the form requires that the rows of the report be in order (for example, if the form uses breaks), use the RUN QUERY command, where the query resembles this:

```
SELECT * FROM tablename
ORDER BY column-list
```

When detail headings disappear from the screen, QMF generates a set that appears at the top of each column. These QMF-generated detail headings contain plus signs (+) mixed with the underscores.

See "DPRE" on page 25 to see how to display a report on your terminal just as it will be printed.

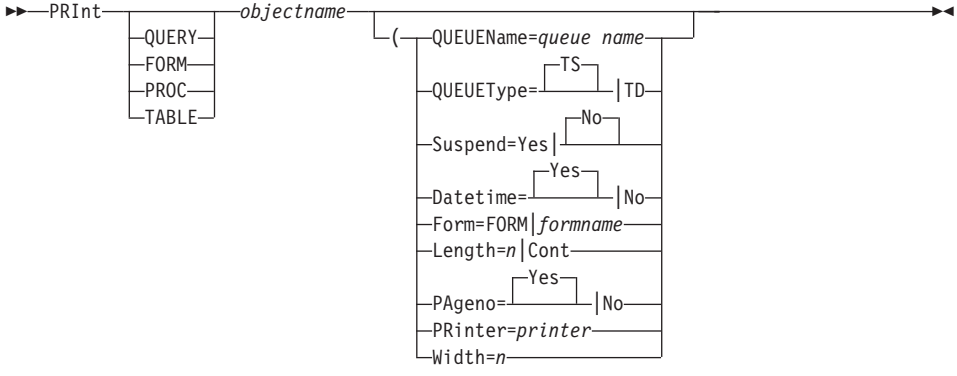
GDDM Printing

GDDM printing of QMF objects in VM varies if some of the default values (ADMLIST and ADMPRINT) were modified by your installation. When using GDDM, remember the following facts:

- DSQPRINT printing requires a FILEDEF. However, when printing with GDDM, you don't need a matching FILEDEF for your printer nicknames. GDDM places the output from your PRINT command in a file called *xxxxxxxx* ADMLIST or *xxxxxxxx* ADMPRINT, where *xxxxxxxx* is the printer nickname you used.
- GDDM printing determines whether an ADMLIST or ADMPRINT file is created, depending on the device token specified in the nickname. System printer output is placed in ADMLIST; queued printer output is placed in ADMPRINT.

Printing in CICS from the Database

To print an object from the database using CICS, specify the name of a query, form, procedure, or table.



When you issue the PRINT command in CICS and do not specify QUEUEName and QUEUEType, and PRINTER=' ', the default values specified in the QMF global variables are used. See "DSQ Global Variables Associated with CICS" on page 302 for details.

If you are using procedures from QMF VSE Version 1 in QMF VSE Version 3.1.1, and the procedures contain PRINT commands using the PRINTER parameter, you must change them to use a GDDM printer nickname and CICS storage (either CICS temporary storage or a CICS transient data queue).

You can print reports to a CICS extrapartition destination by using the destination control table (DCT) statement to define a print file or data set. You can then print the report using the QUEUENAME and QUEUETYPE parameters of the PRINT command.

Differences between Printed and Displayed Reports

A report printed in any environment (CMS, TSO, or CICS) differs from a report displayed on a screen in the following ways:

Part of Report	Displayed Report	Printed Report
Number of pages	One page that can be scrolled	One or more pages
Page headings and footings	Appear only once	Appear at the top and bottom of each page
Detail headings	Before the first detail line at the beginning of a report and on every screen following	Before the first detail line at the beginning of a report and on every page following
Fixed columns	Remain in place when report is scrolled horizontally	Repeated on the left side of each page

QMF

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

Use the QMF command to issue a QMF command whose name is the same as an installation-defined command.

qmfcommand
The QMF command to be run.

You can run the QMF command from the command line, from a procedure, from the Database Object List panel, or from an application.

For example, to list objects in the QMF database (even if your installation has defined the LIST command to have a different function), enter:

```
QMF LIST
```

REDUCE

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

REDUCE is used in reports and in QBE. See *Using QMF*.

REFRESH

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

REFRESH is used:

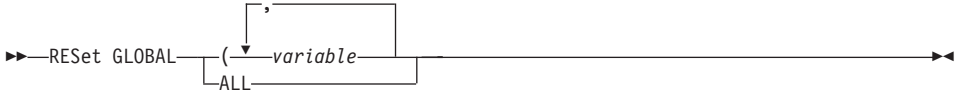
- On the database object list to recreate the list.
- In the Table Editor to set the currently displayed row to the value contained in the corresponding column in the database.

A confirmation panel is displayed before the row is actually changed.

RESET GLOBAL

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The RESET GLOBAL command deletes the values for one or more variables that were set using the SET GLOBAL command.



variable
The name of the variable whose value is to be deleted. Up to 10 variables can be specified.

ALL
Deletes the names and values of all the variables previously set with the SET GLOBAL command. If you have several global variables defined or you don't remember the names of your global variables, you can use this parameter to reset all global variables at one time.

To display a prompt panel on which you can fill in variables and the values you want to reset:

```
RESET GLOBAL ?
```

To delete previously set values for all global variables:

```
RESET GLOBAL ALL
```

To delete the values for the variables DEPT and LOCATION:

```
RESET GLOBAL (DEPT LOCATION
```

Any other variables remain in effect for the remainder of your current QMF session.

You can use global variables in queries, procedures, and forms, but not in the Table Editor.

RESET Object

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The RESET *object* command restores the current object (query, procedure, profile, data, or form) in temporary storage to its initial state.

RESET acts on objects as follows:

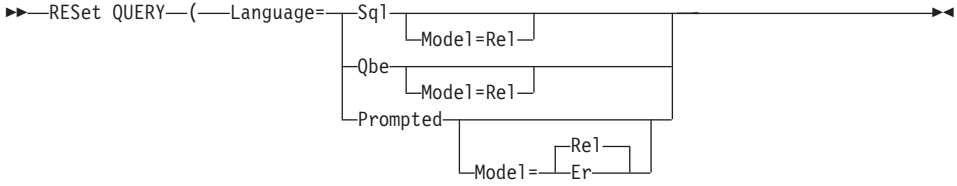
QUERY
For SQL and QBE: Displays a cleared query panel.
For Prompted Query: Starts a new dialog.

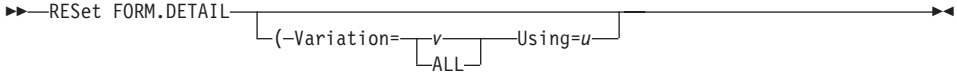
PROC Displays an empty procedure panel.

PROFILE
Resets your profile to the values saved in the database at the current location.

DATA Erases the data in the DATA temporary storage area.

FORM
Sets all parts of the form to defaults that fit the contents of DATA. If there is nothing in DATA, all parts of the forms are set to their default values and no column information is available.





Language=Sql|Qbe|Prompted
Whether you want to display a blank SQL, QBE, or Prompted Query panel. If the LANGUAGE parameter is used, the value of LANGUAGE in your profile remains unchanged.

If LANGUAGE is not specified, the value for LANGUAGE in your QMF profile is used.

Model=Rel|Er
When the object type is QUERY, MODEL can be specified as REL (relational) for any type of query. If MODEL is not specified, REL is assumed.

Consistency is maintained between the language set in your profile and the model specified on the RESET command. For example:

- If your profile specifies LANGUAGE=PROMPTED, you can specify only REL on the Model parameter.
- If either SQL or QBE is specified, Model is assumed to be REL. If you then issue RESET (LANGUAGE=PROMPTED, you are prompted with whatever model is in your profile.
- If MODEL is not specified and LANGUAGE is set to PROMPTED (either on RESET or in your profile), the value of MODEL in your profile is assumed.

Variation=*v* | ALL

A specific variation (*v*) or all the variations. *v* can be any value from 1 to 99. If the number is greater than the number of existing variations, the next number higher than the current number is used. If the VARIATION parameter is omitted and FORM.DETAIL has more than one variation, only the currently displayed variation is reset. If the current QMF panel is not the FORM.DETAIL panel, an error message is displayed.

Using=*u*

An integer from 1 to 99 indicating the FORM.DETAIL panel variation to use. With USING, you can specify the variation to be reset (*v*), or you can reset the last variation displayed. This can be helpful if you make a number of modifications to a detail panel and want to create another with similar changes. The USING parameter can only be used with the FORM.DETAIL panel. If the specified USING variation is greater than the number of existing variations, an error message is displayed.

Examples

- To display an empty SQL Query panel:

```
RESET QUERY (LANGUAGE=SQL
```

- To erase the data in temporary storage:

```
RESET DATA
```

- To display a FORM.BREAK*n* panel set to the default values for your data (*n*=6):

```
RESET FORM.BREAK6
```

- To reset only variation 1:

```
RESET FORM.DETAIL (VAR 1
```

- To create variation 2 and include the contents of variation 1:

```
RESET FORM.DETAIL (VAR 2 USING 1
```

- To reset FORM.DETAIL to defaults with 1 variation:

```
RESET FORM.DETAIL (VAR ALL
```

RETRIEVE

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The RETRIEVE command or ? (question mark) displays commands or parts of commands that were entered on the command line.

The retrieved commands are displayed on the command line. Only commands issued from the QMF command line can be retrieved. RETRIEVE has no prompt panel, names, or parameters.

You can enter RETRIEVE multiple times in succession to retrieve text that was entered farther back in your QMF session. For example, if you most recently issued the command RUN QUERY and before that issued DISPLAY QUERY, enter RETRIEVE twice to display the text `DISPLAY QUERY` on the command line. If you use ?, you can enter several question marks at once (without spaces in between) to display the text entered as many interactions ago as there are questions marks.

The confirmation message that you receive after entering RETRIEVE indicates how far back the retrieved command was entered relative to the command that was most recently entered. When the oldest command is retrieved, and the RETRIEVE command is entered again, command retrieval wraps around, and the most recently entered command is again displayed.

After the text is retrieved, you can press Enter to reissue the command. If the text is not complete, make sure to modify it before pressing Enter, or press a function key with a command compatible with the text. Characters in retrieved text are converted (or not converted) into uppercase according to the CASE parameter specified in your profile.

When the RETRIEVE command is used with text already on the command line, the following is true:

- A ? or multiple ?? can be entered where there is no space between the ? and the rest of the text. For example, `??SPLAY QUERY` is accepted.
- RET can be entered with text on the command line, but there must be at least one blank space between RET and the rest of the text. For example:

  ```
  RET LAY QUERY
  ```

  is accepted.

```
RETLAY QUERY
```

is not accepted.

RIGHT

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The RIGHT command scrolls toward the right margin of a report or a QBE query.

Csr Scrolls the column where the cursor is positioned to the right of the scrollable area.

Half Scrolls right half the width of the scrollable area or to the right margin if that is closer.

Max Scrolls to the right margin of the scrollable area.

Page Scrolls right the depth of the scrollable area or to the right margin if that is closer.

n Scrolls to the right n columns. (n can be any number from 1 through 9999.)

You can use the Right function key to scroll right in a report, or you can type the number of columns you want to scroll right on the command line, and then press the Right function key.

If you do not specify a scroll amount, QMF uses the amount shown after SCROLL ===> in the bottom right corner of the screen. You can change the amount by typing a new amount over it. The change remains in effect throughout the session, except for Max, which remains in effect for only the next command.

You can also change the scroll amount QMF uses by setting the global variable DSQDC_SCROLL_AMT to Csr, Half, Page, or any number of columns (*n*) up to 9999.

RUN

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	*

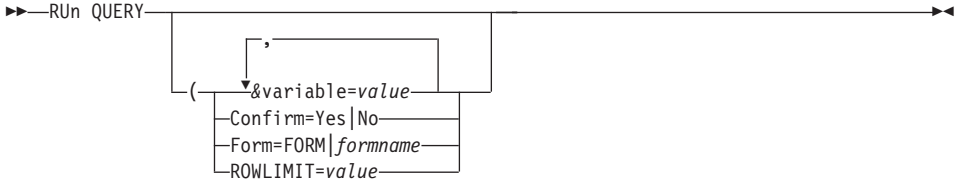
The RUN command runs queries or procedures from QMF temporary storage or from the database (at the current location).

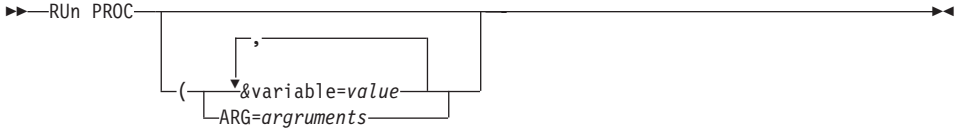
When the current location is DB2, QMF automatically adds the FOR FETCH ONLY clause to any SELECT queries run using the RUN QUERY command. Therefore, do not add the FOR FETCH ONLY clause to queries that you run using QMF.

Note: A QMF administrator can run any query or procedure owned by any user.

Run from QMF Temporary Storage

To run an object that is in QMF temporary storage, specify only the object type: QUERY or PROC.

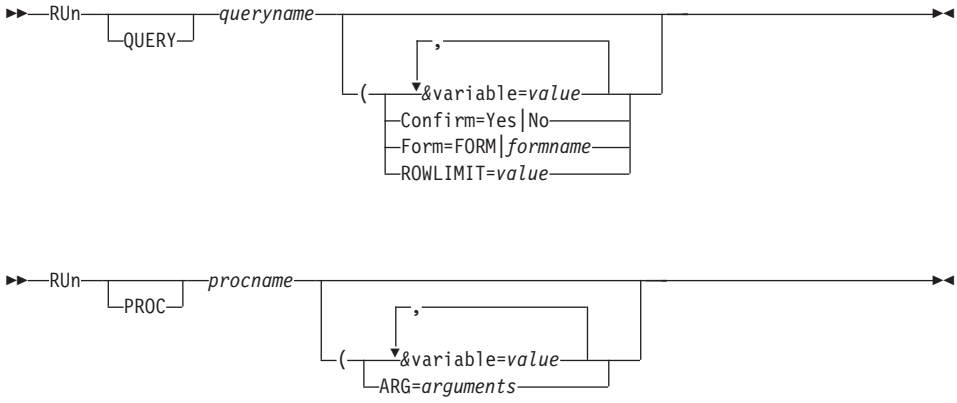




Run from the Database

To run an object that is in the database, specify the object name. Specifying the object type (QUERY or PROC) is optional. If you specify the object type, it must precede the object name.

A QMF form or procedure saved in a specific QMF national language can be run in any NLF session. However, if it was saved in any other QMF national language, it can be run only in a session of that same national language.



queryname
The name of the query to be run.

&*variable***=***value*
Assigns *value* to a variable (&*variable*). This parameter can be used if you are running a query or procedure that uses variables. Substitution variable values can be up to 55 characters long.

Values for variables can be set on the SET GLOBAL command before issuing the RUN command (see "SET GLOBAL" on page 113). However, a value specified on the RUN command overrides the same value set with SET GLOBAL.

If you don't set values for your variables before running your query or procedure, QMF displays a prompt panel so you can fill in the values. Be sure the value assigned to the variable is no longer than 55 single-byte characters (or the equivalent in double-byte characters).

You can specify values for up to 100 variables in a query or procedure. You can specify up to 10 variables on the RUN command; others must be set using SET GLOBAL. QMF first checks the command for a value, then it checks for a global value. If the limit is exceeded, the command is rejected with an error message. Variable names that don't match parameters in your query are ignored. For information on variable names, see "Rules for Variable Names and Values" on page 114.

If your linear procedure sets a variable using SET GLOBAL, that value is not available to commands in that same procedure. However, it is available to queries and procedures called by that procedure.

If you omit the *&variable* parameter, and the object to be run is a query that uses variables with no global variables set, a prompt panel is displayed on which you can fill in variable values.

On the RUN command, variables cannot be replaced by other variables.

For more information about variables, see “SET GLOBAL” on page 113.

Confirm=Yes | No
Whether a confirmation panel is displayed.

Yes Displays a confirmation panel if an object in the database will be changed by this command.

Displays a confirmation panel if the cost estimate to run the query or procedure exceeds the limit specified in the Resource Limit Facility (DB2 Governor) and prevents the object from running.

No No confirmation panel is displayed.

If this parameter is not specified, the value in your profile is used.

CONFIRM is not accepted when running a procedure.

Form=FORM | *formname*
Can be FORM or the name of a form in the database.

FORM
The selected data is formatted by the form currently in the FORM object.

formname
Names a form in the database. That form is put into the FORM object and used to format the display of data.*formname* takes effect only if what is run is a query that selects data. If a procedure is run, this parameter is not accepted.

If you omit the parameter, QMF puts the default form for the selected data into the FORM object and formats the display with that.

rowlimit=*value*
The maximum number of rows to be returned by the query. *value* is n integer from 1 to 99999999.

procname
The name of the procedure to be run.

Arg=*arguments*
Passes arguments to a procedure with logic. The maximum length of an argument is 80 characters. Arguments are received through the REXX PARSE ARG command.

If you are running a query and the form is incompatible with the data or contains errors, the first form panel containing an error is displayed with the error highlighted. To see any other errors, you must correct the first error that is displayed.

Specify Values on the Command Line

QMF assumes it is at the end of a value for a variable specified on the RUN command when it finds a blank, comma, left or right parenthesis, single quote, double quote, or an equal sign. If the value is enclosed in quotation marks, they are included in the value. If the value is enclosed in parentheses, the parentheses are *not* included in the value. To include parentheses in your final value, you must double them.

When QMF finds a substitution variable in the command line, it checks for one of those characters to find the end of the value.

For example, in processing from the command line, if QMF finds a single or a double quotation mark, it tries to find a match for it.

Strings that start with a quotation mark must end with a similar quotation mark; if QMF does not find another quotation mark to pair with the first one, it takes the rest of the command specification and includes it with the beginning quotation mark as part of the value.

To include a blank, comma, right or left parenthesis, single quote, double quote, or equal sign in your variable, you can enclose the value specification in parentheses. For example, this RUN command considers the *value* specification for the variable &X ended at the first comma, and does not accept NAME as a RUN keyword:

```
RUN QUERY (&X=DEPT,NAME,SALARY
```

The same query can be specified on the command line and is properly processed by adding parentheses:

```
RUN QUERY (&X=(DEPT,NAME,SALARY)
```

When the RUN command within a procedure runs a query, this variable parameter can pass a value to a variable within the query. For example, suppose the query uses a variable named &DEPARTMENT. &&DEPARTMENT = 66 assigns the value 66 to the variable &DEPARTMENT in the query without making &DEPARTMENT a variable of the procedure. &&DEPARTMENT = &DEPT makes &DEPT a variable of the procedure, and assigns its value to &DEPARTMENT in the query.

Any CMS, TSO, or CICS commands contained in the procedure specified in the RUN PROC command are executed on the system where QMF is executing. For example, if you have a procedure CALCS consisting of QMF and TSO commands stored at the DB2 subsystem in Dallas, do not attempt to run that procedure if QMF is executing on a VM system (TSO commands are not valid on VM).

When you issue a RUN QUERY command, it runs a query stored at the current location (or optionally uses a form found at the current location). For example, if the query STATSCHK contains the following statement:

```
SELECT * FROM JOHNSON.STATUS
```

the command:

```
RUN QUERY STATSCHK (FORM=FORMCHK
```

retrieves the query, form, and data from the current location.

However, if the query is as follows:

```
SELECT * FROM BILLINGS.JOHNSON.STATUS
```

the command:

```
RUN QUERY STATSCHK (FORM=FORMCHK
```

retrieves the data from the BILLINGS location, and the query and form from the current location.

To create a "top *n* rows" style report you can use ORDER BY with ROWLIMIT. For example, to create a report containing the five managers who have been employed for the longest period of time, you could use the following query and QMF command.

SQL query:

```
SELECT NAME, YEARS
  FROM Q.STAFF
 WHERE JOB='MGR'
 ORDER BY YEARS DESC
```

QMF command:

```
RUN QUERY (ROWLIMIT=5
```

Resulting report:

```
NAME        YEARS
---------  ------
JONES          12
QUILL          10
HANES          10
LU             10
LEA             9
```

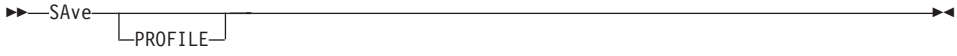
SAVE

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The SAVE command stores objects in the database at the current location.

Saving Your Profile

To save current profile values, enter the SAVE command as follows:



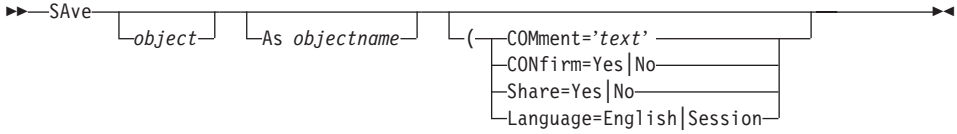
Do *not* specify a profile name with the SAVE PROFILE command. You can have only one saved profile.

To restore your temporarily changed PROFILE to the permanent one saved in the database, enter the command:

```
RESET PROFILE
```

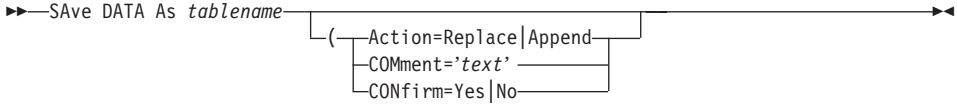
Saving QMF Objects

To save a QMF query, form, or procedure, enter the SAVE command as follows:



Saving Data

To save data in a table, enter the SAVE command as follows:



object
The type of object to save. Valid object types are:

QUERY
If you specify QUERY as the value for *object*, QMF saves the query that is in QMF temporary storage.

FORM
If you specify FORM as the value for *object*, QMF saves the form that is in QMF temporary storage.

PROC
If you specify PROC as the value for *object*, QMF saves the procedure that is in QMF temporary storage.

If you do **not** specify a value for *object*, the default is the object type for the panel currently displayed. If a valid object panel is not displayed, QMF prompts you to enter an object type.

objectname
The name to assign to your saved query, form, or procedure.

If the object in QMF temporary storage currently has a name, QMF uses that name as the default. Otherwise, QMF prompts you to enter an *objectname*.

tablename
The name you assign to your saved data. If the table exists, QMF attempts to replace the data in the table. If the table does not exist, QMF attempts to create a new table in the table space that is specified in the SPACE parameter of the your profile, using the attributes that correspond to the data you are saving.

You can replace existing tables at remote locations by qualifying the table name with a location name and an owner name.

COMment='*text***'**
Allows you to store a comment with any saved object except PROFILE or a remote table. The comment is displayed with the *objectname* when you enter the LIST command to see queries, procedures, forms, or tables that you are authorized to use.

A comment cannot be replaced in a table at a remote location.

If you enter the comment on the command line, enclose the comment in single quotation marks. If the comment contains a single quotation mark (apostrophe), double it. For example,

```
SAVE DATA AS HTABLE
     (COMMENT='Copy of Henry''s table'
```

Omit the quotation marks when entering a comment on a prompt panel.

A comment is converted to uppercase when CASE=UPPER is specified in your profile. Comments saved with the COMMENT parameter of the SAVE command or in a procedure differ from those entered on the query panel (which are only displayed when the saved query is displayed).

You can write a comment using only double-byte characters or mixed double-byte and single-byte characters. The same rules apply as when using only single-byte characters:

- On the prompt panel, enter the comment without surrounding it with single quotation marks.
- On the command line, surround the comment with single quotation marks.

A comment can be up to 57 single-byte characters long (or the equivalent in double-byte characters), excluding any quotation marks that enclose it.

If entered on the prompt panel, a comment with only double-byte characters can contain up to 27 double-byte characters.

Comments containing mixed double-byte and single-byte characters vary in length, depending on the particular combination of characters.

For an explanation of how to calculate the lengths of fields containing mixed data, see the chapter on double-byte character set data in *Using QMF*.

CONfirm=Yes | No

Whether a confirmation panel is displayed:

Yes Displays a confirmation panel if an object in the database will be replaced or changed by this command.

No No confirmation panel is displayed.

If this parameter is not specified, the value in your profile is used.

Share=Yes | No

Whether the objects will be shared with other QMF users:

Yes Allows access to saved queries, procedures, and forms.

No Restricts access to saved queries, procedures, and forms to the current user. Other users cannot use the object.

This parameter is *not* allowed when saving the profile or data objects.

If you omit the parameter, the value specified in the global variable DSQEC_SHARE is used for an object being saved for the first time. For an object being replaced, the current value of SHARE is left unchanged.

Language=English | Session
Whether a form is saved in English or in the current session language. Use this parameter only when you specify FORM as the type of object to save.

The default language is determined by the value of the QMF global variable DSQEC_FORM_LANG.

Action=Replace | Append
Whether to replace the entire database table with the saved data or to append the saved data to the existing table. Action is accepted only when saving data.

Rules

- If you enter the SAVE *object* AS *objectname* command, and an object already exists in the database with the name you specify, QMF replaces the object, subject to these conditions:
  - A form can replace only a form, a procedure can replace only a procedure, and a query can replace only a query.
  - Data can replace only a table or view. The existing table or view and data must have the same number of columns, and corresponding columns must have the same data types.
  - You can replace a table at a remote location, but you cannot create a table at a remote location.
- If you save a QMF form or procedure in a specific QMF national language, it can be used only in a session of the same QMF national language. QMF assumes English unless you change the Language option with the SET GLOBAL command.
- Queries, procedures, and forms are saved as rows in the system control table called Q.OBJECT_DATA. Data is saved in a space that can be named in your profile or is provided by default. See your information center if you want to save data and do not have a space assigned.
- When you save a QMF object, QMF updates the Modified and Last Used fields. The first time you save an object, QMF updates the Created field. These fields, which appear on the database object list, indicate when you most recently modified, most recently accessed or ran, or created a particular data object. QMF updates the Last Used field at most once each day for any object, the first time the object is used that day.
- If you enter the SAVE command on the command line, enclose comments in single quotes. If a comment contains a single quote (apostrophe), use two single quotes in its place.

- If you enter your comments on the SAVE prompt panel, do not enclose them in quotes.
- Query by Example (QBE) queries are saved as they appear on the screen. Any error messages encountered while running the query are saved with the query. To delete the error messages, use the Delete function key or the DELETE command.
- When you save a form, QMF drops any FORM.DETAIL panel variation that you did not modify from its default values. You can drop unwanted FORM.DETAIL variations by resetting a variation to its defaults, and then saving and displaying the form again.
- When you create a new table by issuing the SAVE DATA command, the column names and the labels (if present) are saved with the table. When you display the new table after saving it, the column headings shown on the default form are either the column names or the labels, if they exist.

Examples

- To display a prompt panel for saving a form:

  ```
  SAVE FORM ?
  ```

- On the Query Panel, to save a new query that was not loaded from the database, enter either of the following commands:

  ```
  SAVE
  SAVE QUERY
  ```

  QMF prompts you for a name for the query.

  You can also enter either of the following commands:

  ```
  SAVE AS S1
  SAVE QUERY AS S1
  ```

  where S1 is the name to assign to the query.

- On the Query Panel, to save a query named S1 that is currently loaded in temporary storage, enter either of the following commands:

  ```
  SAVE
  SAVE QUERY
  ```

- On the Query Panel, to save a query named S1 that is currently loaded in temporary storage under a new name, enter either of the following commands:

  ```
  SAVE AS S2
  SAVE QUERY AS S2
  ```

  where S2 is the new name under which you want to save the query.

- On any panel other than the Query Panel, to save a query named S1 that is currently loaded in temporary storage, enter either of the following commands:

```
SAVE
```

QMF prompts you for the object type, and then saves the query as S1.

```
SAVE QUERY
```

QMF saves the query as S1.

- On any panel other than the Query Panel, to save a query named S1 that is currently loaded in temporary storage with the name S2, enter the following command:

  ```
  SAVE QUERY AS S2
  ```

- To include a comment with saved data:

  ```
  SAVE DATA AS FORECAST (COMMENT='Current sales forecast'
  ```

- To replace the existing table JOHNSON.STATUS at the BILLINGS location, while running QMF at DALLAS:

  ```
  SAVE DATA AS BILLINGS.JOHNSON.STATUS
  ```

- To create and save the table JOHNSON.STATUS at the BILLINGS location, while running QMF in DALLAS, first connect to BILLINGS:

  ```
  CONNECT TO BILLINGS
  ```

  then issue the command:

  ```
  SAVE DATA AS JOHNSON.STATUS
  ```

SEARCH

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

In the Table Editor, the SEARCH command locates specified information in the database.

When you are in SEARCH mode, enter your search criteria and press the Search function key to retrieve rows whose columns match your search criteria.

To search for data when you know only part of a value, use either or both of the following symbols in your search criteria:

- A percent sign (%) is used to represent a string of any length and containing any characters.

- An underscore (_) is used to specify a specific character in a particular position. Use more than one underscore in succession to represent an exact number of missing characters.

You can use % and _ in the same value.

For example, searching for _OS% might return the names ROSS or BOSLEY. The percent sign stands for “anything at all”—any number of characters or none.

When searching for data with a specific ending, be aware of the data type of the column you are searching. If the column has a fixed width and the data in the column varies in width, use a trailing percent sign to represent any blanks that might follow your search criteria. If the column you are searching has a data type of VARCHAR, there are no trailing blanks.

You can also use the underscore (_) to specify a character string that ignores a given number of characters. For example, entering J_n% as the search criteria in the name column could result in Jane Taylor and June Kingston being found, no matter how long the search field is. However, if you did not use a trailing percent sign (J_n), no entries would be found.

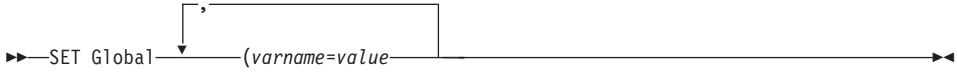
SET GLOBAL

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

You can define up to 10 global variables at a time on the SET GLOBAL command. You can issue SET GLOBAL multiple times to set up to 100 variables in a query or procedure. The values you set remain in effect for your entire QMF session, unless you specifically remove them with the RESET GLOBAL command or specify a new value with another SET GLOBAL command.

The syntax of the command depends on the language you are using. The linear syntax, which is used by REXX, is shown here.

Languages other than REXX use the extended syntax of this command. See *Developing QMF Applications* for details.



varname=value
Assigns a value to a variable name.

Rules for Variable Names and Values

- Global variables can be used in queries, procedures, and forms. Preface a variable with one or more ampersands (&) when you use it in a QMF object.
- Global variable names are limited to 17 characters when entered on the command line; 18 characters when entered through the callable interface. Use of 17 character names is recommended because of limitations of the SET GLOBAL command.
- A global variable name can contain a numeric character, but the first character of a global variable name cannot be numeric.
- The first character of a global variable name must be an alphabetic character (A through Z) or one of these special characters:

  ¢ ! $ ~ { } ? @ # % \

- A global variable name cannot contain blanks or any of the following characters:

  . , ; : < > ( ) | + − * / = & ¬ ' "

- In REXX, values for global variables are limited to 55 characters.
- On the SET GLOBAL command, variable names are not preceded with an ampersand like they are on the RUN and CONVERT commands.
- Global variable names with question marks are not recognized by the QMF form.
- Global variables set to form variable names or aggregation variable names are not recognized by the QMF form.
- Global variable names cannot begin with DSQ, because QMF reserves these letters for QMF predefined global variables.
- Global variable values used in a query cannot begin with a double hyphen (--).
- Trailing blanks are not recognized in global variable names.
- When you are setting many variables, it's easier to keep track of them if you use a procedure (see "Procedures" on page 277).

Rules for Using Quotation Marks and Parentheses in a SET GLOBAL Command

- If a variable is a numeric string, do not enclose the string in quotation marks.
- If a variable is a character string that is a *name* (such as the name of a column, a table, or an operator):

- Enclose the complete string in a set of single quotation marks. (These quotation marks are not considered part of the value.)
- Double all embedded quotation marks.

For example, if the SELECT statement is:

```
SELECT DEPT, &COL FROM &TABLE
```

The SET GLOBAL command is:

```
SET GLOBAL (COL='NAME', TABLE='Q.STAFF'
```

- If the variable is a character string that is to be used as a value contained within a column (unique to the WHERE clause in an SQL statement) you can use either of two methods to specify a string.

  Method 1 (quotes)

  1. Start with original string.
  2. Double all quotation marks (if any).
  3. Enclose the string in three sets of single quotation marks.
  4. Double all the embedded quotes (all but the outside most ones).

  Method 2 (quotes and parentheses)

  1. Start with original string.
  2. Double all quotation marks (if any).
  3. Enclose the string in one set of single quotation marks.
  4. Enclose the string in one set of parentheses.

  For example, if the SELECT statement is:

  SELECT DEPT FROM &TABLE WHERE NAME=&ABC

  the Method 1 SET GLOBAL command is (substituting JAMES for variable ABC):

  SET GLOBAL (ABC='''JAMES''', TABLE='Q.STAFF'

  The Method 2 example for the same SELECT statement is (substituting O'BRIEN for variable ABC):

  SET GLOBAL (ABC=('O''BRIEN'), TABLE='Q.STAFF'

- If the variable contains a blank, comma, single quote, double quote, or an equal sign, the entire value must be enclosed in a set of parentheses. However, if the value also includes left or right parentheses, you must enclose the entire value in single quotes *instead* of parentheses, and double any embedded single quotes.

For example, if the SELECT statement is:

```
SELECT &COLS FROM Q.STAFF
```

The SET GLOBAL command is:

```
SET GLOBAL (COLS=(NAME, JOB, SALARY)
```

- If the variable is a numeric string, you don't need to use quotation marks.

Setting Variables on a Prompt Panel

If you issue SET GLOBAL ?, a prompt panel is displayed where you can fill in variables and the values you want to set. Use the Tab key to move between the variable name column and the value input area. Press Enter to issue the SET GLOBAL command.

SET GLOBAL Command in QMF Linear Procedures

Within a QMF linear procedure, if you use substitution variables in your SET GLOBAL command and wish to reset these variables after each run, you must code a RESET GLOBAL command at the end of the procedure. Otherwise, the previous set of substitution values are used.

SET GLOBAL Command in the Callable Interface

The syntax of this command depends on which language you are using.

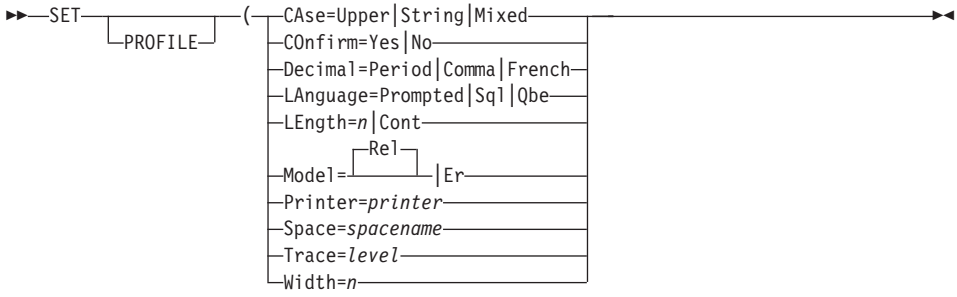
The rules are the same for setting global variables in the callable interface as those under "Rules for Variable Names and Values" on page 114. However, global variables set using the extended syntax of the SET GLOBAL command can have values of up to 32,768 characters, while those set using the linear syntax of the command are limited to values of 55 characters.

See *Developing QMF Applications* for details on the callable interface.

SET PROFILE

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The SET PROFILE command sets values in your profile. A change to your profile is in effect only during your current terminal session (or until you make another change to the same value). To make the change permanent, issue the command SAVE PROFILE.



CAse=Upper | String | Mixed
Whether input from the terminal is converted to uppercase. (The CASE parameter has no effect on double-byte characters.)

Even though the commands in this chapter are shown in uppercase, you can enter them in lowercase if your profile is set to UPPER or STRING. To use mixed-case, set your profile to MIXED.

Upper All input is converted into uppercase. Lowercase letters are *not* distinguished from uppercase letters.

String Input is converted into uppercase except for:

- Character strings enclosed in single or double quotation marks
- Comments in queries and linear procedures
- All text in procedures with logic
- Column headings and text fields in forms
- Data entered in the Table Editor

Lowercase letters in names are distinguished from uppercase letters if the name is enclosed in quotation marks.

Mixed No input is converted to uppercase. Lowercase letters are not the same as uppercase letters. With this parameter, all reserved words and commands must be entered in uppercase.

COnfirm=Yes | No
Whether a confirmation panel is displayed:

Yes Displays a confirmation panel if an ERASE, EXPORT, IMPORT, RUN, or SAVE command is specified without the CONFIRM parameter, and if a file, data set, or the database will be replaced or changed by this command.

No No confirmation panel is displayed.

Decimal=Period | Comma | French
How large numbers are punctuated in a display. For example, using the number 123456789:

Period Displays the number as 1,234,567.89. This is the style common in the United States.

Comma
Displays the number as 1.234.567,89. This style is common in much of Europe.

French
Displays the number as 1 234 567,89. This style is common in France.

You must use the corresponding edit code (D, K, or P) in the form. These edit codes provide thousands separators for numeric values.

LAnguage=Prompted|Sql|Qbe
Which type of query panel (Prompted, SQL, or QBE) is displayed after issuing the command RESET QUERY. The value of LANGUAGE in your profile is in effect unless you use the LANGUAGE parameter on the RESET QUERY command.

LEngth=*n***|Cont**
The maximum number of lines to print on any page when using the PRINT command. *n* can range from 1 to 999. CONT causes continuous printing, without page breaks. CONT is not accepted with a printer name, when printing the Form, or in Prompted Query.

The value of LENGTH in your profile is in effect only if you do *not* use the LENGTH parameter on PRINT.

Model=Rel|Er
Which type of prompted query you plan to write (e.g., Relational (Rel)).

Printer=*printer*
The nickname of the GDDM printer to be used. When entering this parameter on the command line, you can specify a blank (' ') as the printer nickname, to send output to a file, a data set, or a printer (depending on what is specified in the ddname or FILEDEF). If you do not use the PRINTER parameter at all, QMF defaults to the printer specified in your profile. You must specify a name when printing the form or a prompted query.

Space=*spacename*
The dbspace that holds tables when you use the SAVE DATA and IMPORT commands. You can set SPACE to any valid dbspace name.

Trace=*level*
Turns on or off the QMF trace facility. You can set TRACE to ALL, NONE, or a sequence of alternating function-identifier and trace-level codes. For more information on the TRACE command, see *Managing QMF* for your operating system.

In a CICS environment, you can specify the name and the type of queue that the trace data is written to by using the global variables DSQAO_CICS_TQNAME and DSQAO_CICS_TQTYPE. See "DSQ Global Variables Associated with CICS" on page 302 for details.

Width=*n*

The maximum number of characters to print on any line when using PRINT. *n* can range from 22 to 999 for single-byte characters and from 10 to 498 for double-byte characters. For lines containing mixed single- and double-byte characters, the maximum print length varies depending on the particular mix of characters. If you omit this parameter on the PRINT command, the value of WIDTH in your profile is used.

For an explanation of how to calculate the lengths of fields containing mixed data, see *Using QMF*.

SHOW

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

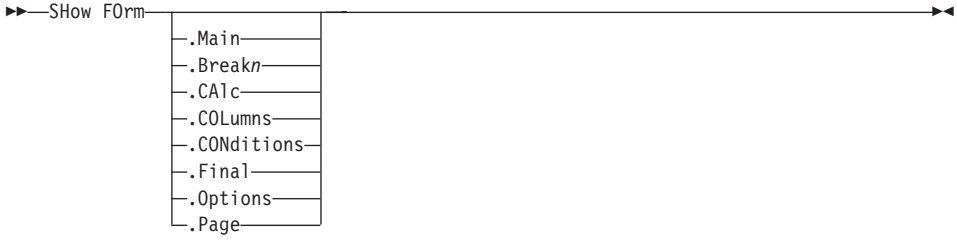
Use the SHOW command to:

- Navigate among object panels.
- Show a variation from a FORM.DETAIL panel.
- Show a list of global variables.
- Show fields that are too long to fit on the panel.
- Show the SQL translation of a relational prompted query.
- Show a command panel from the database object list where you can specify any QMF command or synonym.

The SHOW command is similar to the DISPLAY command. The SHOW command shows object panels, global variables, and certain parts of panels in QMF temporary storage. The DISPLAY command displays objects from the database or from QMF temporary storage.

Show an Object or Form Panel

Navigate among object or form panels in temporary storage. Panel names can be specified with their minimum unique abbreviations. For example, the FORM.MAIN panel can be specified as F.M.



Break*n*
Which FORM.BREAK panel to display. *n* can be any value from 1 to 6. If no number is specified for *n* with FORM.BREAK*n*, FORM.BREAK1 is assumed.

You can type the name of a panel on the command line and press the Show function key to display a particular panel.

When you enter SHOW REPORT or SHOW CHART and the form contains errors, QMF displays the form panel on which the first error occurs. The entry area containing the error is highlighted. If there are other errors, the next one detected is highlighted after you correct the first one.

On a form panel, SHOW REPORT and SHOW CHART can fail if the form is incompatible with the data or if the form contains errors. QMF displays the form panel on which the first error occurs, highlighting the entry area containing the error. To see other errors, correct the first error displayed and press Enter.

You can enter the SHOW command with no parameters or press the Show function key without specifying a panel to display the SHOW command prompt panel:

```
                  SHOW Command Prompt

Enter the name or number of the panel to show. (_____________)
                                                 1 to 13 of 17

1.  PROFile            Current user profile
2.  PROC               Current procedure
3.  Query              Current query
4.  Report             Current report
5.  CHARt              Default chart
6.  Globals            Global variable list
7.  FOrm               Current form
8.   Form.Main           Basic report formatting
9.   Form.COLumns        Column attributes
10.  Form.CONditions     User-defined conditions
11.  Form.CAlc           User-defined calculations
12.  Form.Page           Page heading and footing text
13.  Form.Detail         Detail text
14.  Form.Final          Final footing text
15.  Form.BreakN         BreakN text (where N is 1 to 6)
                           Example:  FORM.BREAK2 = F.B2 = 15.2
16.  Form.Options        Choices about a report's appearance
-------------------------------------------------------------------
F1=Help  F3=End  F7=Backward  F8=Forward
```

Show a Variation

Display a variation of a FORM.DETAIL panel.

The variation *v* can be any value from 1 to 99. If the number is greater than the number of existing variations, the next number higher than the current number is used. This parameter does not appear in the SHOW command prompt panel because the number is typed directly on the FORM.DETAIL panel.

To show variation 2 of FORM.DETAIL:

```
SHOW FORM.DETAIL (VAR 2
```

Show a List of Global Variables

Display the global variable list panel. See Appendix B for tables listing QMF global variables.

On the global variable list panel, you can:

- Change the value of an existing global variable by typing over its value and pressing Enter. Variable values that can be changed are surrounded by parentheses, like this:

  ```
  DSQDC_COST_EST     ( 1____________________)
  ```

  Note that when you change the value of an existing global variable on the global variable list panel, QMF removes trailing blanks from the value. To preserve trailing blanks in a value for a global variable, use the SET GLOBAL command to change the value.
- Add a new global variable by pressing the Add function key.
- Reset a user-defined global variable by entering the RESET command. For example, if you set a global variable named XYZ, you can remove it from the global variable list by entering:

  ```
  RESET GLOBAL (XYZ
  ```

  To reset all the global variables you have set using the SET GLOBAL command, enter:

  ```
  RESET GLOBAL ALL
  ```

  You cannot remove a variable that begins with DSQ from the list of global variables.

The list is presented with your global variables first (in alphabetic order), followed by QMF's global variables (also in alphabetic order).

If you type a value containing an error, the error is highlighted. Generally, you must correct this error before you can continue using the global variable list.

Global variables added or changed on this panel can be up to 32,768 characters long.

Show an Entire Field

Display an entire field that was truncated on the screen. The Show Field function key is used in Relational Prompted Query, the Table Editor, and on global variable list panels.

In Relational Prompted Query, you can display a panel containing fields that are too long to fit on the base panel. The area is scrollable so you can see the entire field.

In the Table Editor, you can determine what kind of data is accepted for a specific column, and you can expand a truncated field.

On a global variable list panel, if a variable is too long to display on one line, a greater-than sign (>) appears to the right of the value.

You can display the entire value in a scrollable area by placing the cursor on the line containing the long variable value and pressing the Show Field function key.

Show a QMF Command

Display a command panel that lets you enter any QMF command or synonym. The Command function key is used from the database object list.

Show an SQL Translation of a Relational Prompted Query

Display the SQL equivalent of a prompted query in the echo area.

You can browse the SQL query and scroll forward and backward, but you cannot edit it.

SORT

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The SORT command sorts items in a database object list. You can issue this command only by pressing the Sort function key. When you request SORT, a panel is displayed that lets you select the order of the names.

SPECIFY

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

You can use the SPECIFY command in Prompted Query and on FORM.COLUMNS.

Prompted Query

In Prompted Query, SPECIFY displays a list from which you can select the panel you want to see next.

Columns
Name your columns.

Duplicates
Specify whether duplicate entries are to be shown.

Rows Fill in the rows.

Sort Sort the rows.

Tables Name the tables to be used.

If you enter SPECIFY alone, the first selection panel appears so you can choose the next panel you want to see: columns, duplicates, rows, sort, or tables.

If you specify your selection on the command line, the first selection panel is skipped. For example:

```
SPECIFY ROWS
```

takes you directly to the Rows panel.

FORM.COLUMNS Panel

On the FORM.COLUMNS panel, SPECIFY displays a panel from which you can specify additional information about columns in the form or define new columns in the form.

Alignment
Displays the column number, column heading, heading alignment, and data alignment values. Only the heading and data alignment values can be modified.

DEfinition
Displays the column number, column heading, and the definition for the column (if any). Only the definition value can be modified.

You can issue the SPECIFY command with the cursor on the column information line.

- For column alignment, the cursor position (when issuing the SPECIFY command) determines which column appears in the alignment panel.
- For column definition, the cursor position (when issuing the SPECIFY command) determines which column appears in the definition panel.

If the cursor is not on the column information line, a panel is displayed beginning with the first column. You can move through the panels sequentially by using the NEXT and PREVIOUS function keys.

On a FORM.COLUMNS panel with column definition, you can do the following:

- Define a column based on other columns.
- Group results based on ranges of values.
- Define user functions for individual data values.
- Display partial columns.
- Set control breaks for partial columns.
- Apply multiple usages to a single column.

You can cancel your SPECIFY command with the Cancel function key.

START

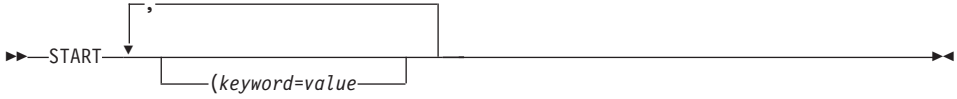
The START command begins a new QMF session. The syntax of the command depends on the language you are using. The linear syntax, which is used by REXX, is shown here.

START

Languages other than REXX (C, COBOL, FORTRAN, PL/I, RPG, or assembler language) use the extended syntax of this command. See *Developing QMF Applications* for details.

QMF requires that the START command not be abbreviated.

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X



keyword
Name of the start command keyword.

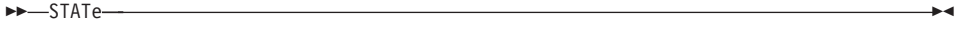
value
A value for the specified start command keyword.

See *Developing QMF Applications* for a list of valid start command keywords and values.

STATE

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X		X		

The STATE command saves the values of selected QMF "state" variables in the QMF global variable pool. STATE is an application support command and can be run only through the QMF command interface.



Use STATE from an application, an EXEC, or a CLIST.

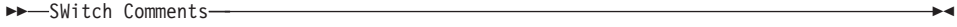
When the STATE command is issued, new variables are set to the database location associated with the current object.

See "Appendix B. QMF Global Variable Tables" on page 295 for tables listing QMF global variables.

SWITCH

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

Use the SWITCH command to display or remove comments in a database object list and in tables in Prompted Query.



When the SWITCH command is issued:

- If comments are displayed on the panel, they are removed.
- If no comments are displayed on the panel, the current list panel is displayed again *with* a Comments column. The comments for each object (or blank) are shown on the panel. They are truncated to fit on the screen.

The function key that performs the Switch Comment command is labeled "Comments".

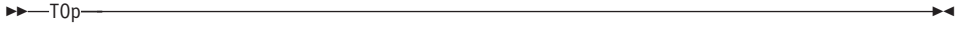
TOP

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X	X	X	X

The TOP command scrolls to the beginning of queries, procedures, reports, global variable lists, and scrollable form panels. TOP is equivalent to BACKWARD MAX.

The minimum unique abbreviation of this command differs depending on whether you are in the TSO or CMS environment.

In TSO:



In CMS or CICS:

TOP

To scroll to the top of heading text on form panels, position the cursor on the portion of the panel where the heading text is specified and enter the TOP command.

TSO

TSO With ISPF	TSO Without ISPF	CMS With ISPF	CMS Without ISPF	CICS
X	X			

Use the TSO command to issue a command in the TSO environment (at the system where QMF is executing) without terminating your use of QMF.

For example, you can issue a TSO SEND command, which sends a message to another user.

```
TSO SEND 'RECEIVED PROC2. THANK YOU.' U(JOE)
```

Everything after the name of the command (TSO) is sent to the TSO environment and then interpreted in TSO. If the command runs successfully, the same panel on which you issued the TSO command is displayed. If what you send to TSO does not run successfully, an error message from TSO is displayed.

Use the TSO command with care; it could adversely affect your QMF environment.

Chapter 2. SQL Keywords and Functions Used in QMF Queries

Selected SQL keywords that are used in QMF queries are described here. SQL functions are described beginning at “SQL Scalar Functions” on page 176. Some words are “keywords” in database management systems, and, in many cases, cannot be used as the name of a table, view, column, or index in a query, unless they are enclosed in double quotation marks. See your SQL reference for a list of reserved words in your database manager.

SQL Keywords

This is not a complete list of SQL keywords that are available. For more information, consult the SQL reference for your database manager.

ADD

You can add columns to a table only if *you* created the table or are specifically authorized to do so. The following example adds one column to the description of table PERS:

```
ALTER TABLE PERS
ADD PHONENO SMALLINT
```

The new column is initially filled with null values. Use the UPDATE statement to provide actual values for the new column.

In DB2, you can define a column as NOT NULL WITH DEFAULT, but you can’t define an added column to be NOT NULL.

NOT NULL WITH DEFAULT is invalid in SQL/DS.

ALL

A subquery generally returns only one value. However, it is possible for a query to return a set of values.

To permit a query to return a set of values, rather than an individual value, use the ALL keyword with the following comparison operators:

```
=   ¬=   >   >=   <   <=
```

ALL

With ALL, each value in the returned set must be satisfied.

The symbol ¬= is an alternative symbol for < > (not equal to). It is an American National Standards Institute (ANSI) SQL operator. If you are using remote data access, the preferred symbol is < >.

The following query produces a report that lists the department with the highest average salary. Use of the ALL keyword specifies that the department selected by the main SELECT statement must have an average salary equal to or greater than all average salaries of other departments.

```
SELECT DEPT, AVG(SALARY) FROM Q.STAFF
GROUP BY DEPT
HAVING AVG(SALARY) >= ALL
      (SELECT AVG(SALARY) FROM Q.STAFF
       GROUP BY DEPT)
```

Operators other than the equal sign (=) can be used with the ALL keyword. If any of the results produced by the subquery are NULL, the result of the condition with ALL is unknown.

ALTER TABLE

You can alter a table only if *you* created the table or are specifically authorized to do so. The ALTER TABLE statement specifies which existing table to change. For example, following ALTER TABLE, you can use the ADD statement to add a new column on the right side of a table. (See “ADD” on page 129.)

AND

You can select rows based on multiple conditions connected by AND or OR. Two conditions connected by AND select only rows that satisfy both conditions. For example:

This query:

```
SELECT ID, NAME, YEARS, SALARY
FROM Q.STAFF
WHERE YEARS = 10 AND SALARY > 20000
```

Produces this report:

```
   ID  NAME       YEARS     SALARY
-----  ---------  -----  ---------
   50  HANES         10   20659.80
  210  LU            10   20010.00
```

Compare the results using AND with “OR” on page 157.

Parentheses

If you use both AND and OR, use parentheses to specify the order that AND and OR conditions are evaluated. Compare the following examples:

With parentheses:

```
WHERE (JOB='SALES' AND COMM > 1200) OR YEARS > 10
```

Selects employees that satisfy *at least one* of these conditions:

- Their job is sales and their commission is more than $1200
- *OR*, they have more than 10 years of service.

Result: 90, 260, 310, 340.

With the parentheses moved:

```
WHERE JOB='SALES' AND (COMM > 1200 OR YEARS > 10)
```

Selects employees that satisfy *both* these conditions:

- Their job is sales
- *AND*, either their commission is more than $1200 or they have more than 10 years of service.

Result: 90, 310, 340.

You can use more than one level of parentheses. The condition is evaluated from the innermost level of nested parentheses outward, as in algebraic expressions.

If you do not use parentheses, all conditions connected by AND are evaluated and connected first, and then conditions connected by OR. That is, if A, B, and C are conditions, these two phrases produce the same results.

```
A AND B OR C    means    (A AND B) OR C
```

ANY

A subquery generally returns only one value. However, it is possible for a query to return a set of values. To permit a query to return a set of values, rather than an individual value, the ANY keyword can be used with the comparison operators:

```
=   ¬=   >   >=   <   <=
```

With ANY, at least one value in the set returned must be satisfied.

ANY

IN can be used in a subquery in place of = ANY, and SOME is a synonym for ANY.

The symbol ¬= is an alternative symbol for < > (not equal to). It is an ANSI SQL operator. If you are using remote data access, the preferred symbol is < >.

The following query produces a list of employees who work in the Eastern division. First, the subquery finds the department numbers in the Eastern division. Then, the main query finds the employees who work in *any* of these departments.

This query:

```
SELECT NAME, ID
FROM Q.STAFF
WHERE DEPT = ANY
 (SELECT DEPTNUMB FROM Q.ORG WHERE DIVISION='EASTERN')
```

Produces a list of names and IDs of employees who work in the Eastern division.

The keyword ANY was used in this query because there are multiple departments in the Eastern division. If ALL is used instead of ANY, the result is an empty set. No employee works in *all* the departments of the Eastern division.

AS

You can use an AS clause in a SELECT statement to name or rename a result column in a query. The name must not be qualified and does not have to be unique.

For example:

```
SELECT NAME, SALARY*0.05 AS "RAISE"
  FROM Q.STAFF
```

If the AS clause is not specified and the result column is derived from a column name, the result column name is the unqualified name of that column.

AVG

AVG is a column function. The following example includes more than one column function in the SELECT statement. It calculates and displays, for Department 10, the sum of employee salaries, the minimum, average, and maximum salary, and the number of employees (COUNT) in the department.

This query:

```
SELECT SUM(SALARY), MIN(SALARY), AVG(SALARY),
   MAX(SALARY), COUNT(*)
FROM Q.STAFF
WHERE DEPT = 10
```

Produces this report:

```
SUM(SALARY) MIN(SALARY)      AVG(SALARY) MAX(SALARY) COUNT(EXPRESSION)
----------- ----------- --------------- ----------- -----------------
   83463.45    19260.25 20865.8625000000    22959.20                 4
```

Write a column function like this:

`AVG(`*`expression`*`)`

The parentheses are required. *expression* is most often a column name, but can also be:

- An arithmetic expression containing at least one column name.
- DISTINCT, followed by a column name.

A column name in a function must not refer to a long string column or a column derived from a column function (a column of a view can be derived from a function). Column functions cannot be nested within other column functions. Null values are not included in the calculation made by a column function.

BETWEEN x AND y

You can retrieve data from each row whose column, named in a WHERE clause, has a value within two limits. Use BETWEEN in place of an AND condition when using greater than or equal to (>=) and less than or equal to (<=).

The limits you specify are inclusive. Enter the lower boundary (smaller value) of the BETWEEN condition first, then the upper boundary (larger value). The following example selects employees who have a salary between $20,000 and $21,000. GRAHAM has a salary of exactly $21,000.

This query:

```
SELECT ID, NAME, SALARY
FROM Q.STAFF
WHERE SALARY BETWEEN 20000 AND 21000
```

Produces this report:

```
  ID  NAME          SALARY
----- ---------  ---------
  50  HANES       20659.80
 210  LU          20010.00
 310  GRAHAM      21000.00
```

Examples:

- Select everyone whose name is alphabetically between HANES and MOLINARE:

  ```
  SELECT ID, NAME
  FROM Q.STAFF
  WHERE NAME BETWEEN 'HANES' AND 'MOLINARE'
  ```

- Select everyone who has between 10 and 12 years of service (inclusive):

  ```
  SELECT ID, NAME, YEARS
  FROM Q.STAFF
  WHERE YEARS BETWEEN 10 AND 12
  ```

- Select employees whose salary is *NOT* in the range of $19,000 to $21,

  ```
  SELECT ID, NAME, YEARS
  FROM Q.STAFF
  WHERE SALARY NOT BETWEEN 19000 AND 21000
  ```

  Each employee whose salary is less than $19,000 or more than $21,000 is included in the report. Employees with salaries between and including $19,000 and $21,000 are not included.

COUNT

The COUNT function counts only non-null values. Therefore, the data type of the result of the COUNT function always has the NOT NULL attribute. There are two uses of COUNT:

- COUNT(DISTINCT *colname*) — Counts rows returned in which there is a non-null value in a named column. It eliminates duplicates from the count.

  This form *must* be used with a column name; it cannot be used with an expression. See also “DISTINCT” on page 140.

  ```
  SELECT COUNT(DISTINCT DIVISION)
  FROM Q.ORG
  ```

  The result is 4.

- COUNT(*) — Counts rows returned regardless of the value of any column. This form is *not* used with a column name.

  ```
  SELECT SUM(SALARY), MIN(SALARY), AVG(SALARY),
     MAX(SALARY), COUNT(*)
  FROM Q.STAFF WHERE DEPT = 10
  ```

This example includes more than one column function in the SELECT statement. It calculates and displays, for Department 10, the sum of employee salaries, the minimum, average, and maximum salary, and the number of employees (COUNT) in the department, and produces the following report:

```
SUM(SALARY) MIN(SALARY)      AVG(SALARY) MAX(SALARY)
----------- ----------- ---------------- -----------
   83463.45    19260.25 20865.8625000000    22959.20

                                           COUNT(EXPRESSION 1)
(Continuation of report)                   -------------------
                                                             4
```

CREATE SYNONYM

The CREATE SYNONYM statement defines an alternative name for a table or view. This lets you refer to a table owned by another user without having to enter the fully qualified name. You can also create synonyms for your own tables and views. The synonym remains defined until it is dropped.

The following example creates a new name for the table Q.APPLICANT.

```
CREATE SYNONYM APPLS FOR Q.APPLICANT
```

After executing this statement, you can write APPLS instead of Q.APPLICANT.

A synonym is only of value when it is shorter than the fully qualified table name (which can be up to 26 characters, not counting the intervening period). Or it can be a valuable protection for your queries if you are using tables created by someone else.

For example, suppose that table Q.APPLICANT is dropped and re-created by user BDJ1385L. All your queries were written using the synonym APPLS. If you use SQL/DS, your first step is to drop the synonym by using this command:

```
DROP SYNONYM APPLS
```

If you use SQL/DS or DB2, make this change:

```
CREATE SYNONYM APPLS FOR BDJ1385L.APPLICANT
```

If you share a query that uses a synonym, it will not work for the other user until that user creates the same synonym. You cannot share synonyms you define under your authorization identifier. However, other users can define the same synonyms with the same meanings.

DBCS Data

If your installation uses DBCS data, do not create a synonym that contains double-byte characters that are internally represented as double quotation marks unless your database specifically supports double-byte characters in table names. For more information about how to write names containing double-byte characters, see “Names with Double-byte Characters” on page 272.

CREATE TABLE

The CREATE TABLE statement defines a table. You provide the name of the table and the names and attributes of its columns. You can create a table only if you have authorization to do so. You can also grant or revoke authorization for other people to use a table you created. See “GRANT” on page 143 and “REVOKE” on page 160.

The syntax of the CREATE TABLE statement is:

```
CREATE TABLE tablename (column1 type1 NOT NULL,
column2 type2 . . .)
  IN space-name
```

tablename
: The name you assign to the table.

  If your installation uses DBCS data, names of tables cannot contain double-byte characters that are internally represented as double quotation marks unless your database specifically supports double-byte characters in table names. (See “Names with Double-byte Characters” on page 272.)

column1 type1
: The name you assign to the first column, and the data type describing it.

  If the data type is CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, or DECIMAL, you must specify the maximum length of a data element, in parentheses. For DECIMAL, you must also specify the number of places after the assumed decimal point.

column2 type2
: The name you assign to the second column and the data type describing it.

NOT NULL
: Optional for any column you define. If you use NOT NULL in the table definition, then any attempt to have no value in the

corresponding column of the table produces an error message. Omitting NOT NULL allows null values in the column.

IN *space-name*
Refers to a table space or a dbspace in which the table is to be created. This clause is needed only if your installation does not provide a space to be used by default.

You can find the *space-name* used when QMF creates tables for SAVE DATA or IMPORT TABLE by issuing the QMF command DISPLAY PROFILE. See *Managing QMF* for your environment

The following CREATE statement defines a table called PERS. The columns in PERS have the same characteristics as Q.STAFF, but contain no data.

```
CREATE TABLE PERS
(ID SMALLINT NOT NULL,
NAME VARCHAR(9),
DEPT SMALLINT,
JOB CHAR(5),
YEARS SMALLINT,
SALARY DECIMAL(7,2),
COMM DECIMAL(7,2))
IN space-name
```

ID The employee number is a small integer and null cannot be specified for it.

NAME
The maximum length of the name is 9.

DEPT The department number is small integer.

JOB The name of the job has 5 characters.

YEARS
The number of years is small integer.

SALARY
A 7-digit number with 2 decimal positions.

COMM
A 7-digit number with 2 decimal positions. (Don't forget the final parenthesis.)

You can use NOT NULL with any set of columns in the CREATE TABLE statement; in the example, it appears with column ID. In effect, it means that any row entered into PERS must have, at the very least, an employee number.

This statement defines the Q.APPLICANT table:

```
CREATE TABLE APPLICANT
(TEMPID  SMALLINT  NOT NULL,
NAME     VARCHAR(9),
```

```
ADDRESS  VARCHAR(17),
EDLEVEL  SMALLINT,
COMMENTS VARCHAR(29))
IN space-name
```

This statement defines the Q.INTERVIEW table:

```
CREATE TABLE INTERVIEW
(TEMPID   SMALLINT,
INTDATE   DATE,
STARTTIME TIME,
ENDTIME   TIME,
MANAGER   SMALLINT,
DISP      VARCHAR(6),
LASTNAME  VARCHAR(9),
FIRSTNAME VARCHAR(9))
IN space-name
```

Defining the table does *not* put data into it. For methods of entering data into it, see “INSERT INTO” on page 150.

CREATE VIEW

View is an imaginary table that appears to contain data selected from existing tables. The view can rename and rearrange columns, omit unwanted columns or rows, define columns by expressions, group results, and combine more than one table. Views make it possible to view data that exists in parts of one or more tables. No data actually exists in a view.

Any SELECT statement that does *not* contain ORDER BY can be used as the basis of a view; the selected columns and rows become the columns and rows of the view. In the following example, NAME, ID, and JOB from Q.STAFF become the columns of D42. The column names for D42 are LAST NAME, EMP. ID, and JOB.

```
CREATE VIEW D42
 ("LAST NAME", "EMP. ID", JOB)
   AS SELECT NAME, ID, JOB
FROM Q.STAFF
WHERE DEPT = 42
```

Issue the command:

```
DISPLAY TABLE D42
```

to display this view:

```
LAST NAME EMP. ID  JOB
--------- -------  -----
KOONITZ         90  SALES
PLOTZ          100  MGR
YAMAGUCHI      130  CLERK
SCOUTTEN       200  CLERK
```

There are two main reasons for using a view:

- To simplify writing a query to use its data, as in the example above.
- To prevent access to data. Anyone using the view D42, defined above, cannot see salary data.

Use a view by its name, like you use a table. You can select from it, writing the same kind of SELECT statement as if it were a table. For example, run this query:

```
SELECT * FROM D42
WHERE JOB='CLERK'
```

With a few restrictions, you can insert, update, and delete rows in a view. Corresponding changes are made to the tables the view is based on.

There are a few things you can't do with a view:

- You cannot insert, update, or delete using a view if the view contains:
  - Data from more than one table.
  - A column defined by one of the column functions, for example, SUM(SALARY).
  - Data selected by the DISTINCT or GROUP BY keywords.
- You cannot update or insert (you can delete) if the view contains a column defined by an expression (like SALARY / 12).
- You cannot use UNION when creating a view.
- You cannot join a view that was created using a GROUP BY to another table or view.

DELETE

You can delete rows from a table only if *you* created the table or are specifically authorized to do so. You can delete information from a table by row. Individual fields in a row or complete columns of information cannot be deleted.

The DELETE statement consists of two parts:

DELETE FROM
The table from which rows are to be deleted.

WHERE
The rows to be deleted.

If DELETE is entered with no WHERE clause specified, all rows of the table are deleted. The table still exists, but it no longer contains any rows.

The following statement deletes employee number 140 from the table PERS.

```
DELETE FROM PERS
WHERE ID = 140
```

In this example, ID is used rather than employee name to avoid deleting more rows than anticipated, because there could be more than one employee with the same name.

You can delete more than one row with one DELETE statement. Include a condition to show which rows to delete. The next example deletes everyone in Department 10:

```
DELETE FROM PERS
WHERE DEPT = 10
```

For information about authorization, see “GRANT” on page 143.

DISTINCT

Use DISTINCT before the column names in an SQL statement to prevent duplicate rows from being selected. The following example specifies, in effect, “List only the unique divisions that exist in the table Q.ORG”:

This query:

```
SELECT DISTINCT DIVISION
FROM Q.ORG
```

Produces this report:

```
DIVISION
---------
CORPORATE
EASTERN
MIDWEST
WESTERN
```

Compare the result in the previous example with the following:

This query:

```
SELECT DIVISION
FROM Q.ORG
```

Produces this report:

```
DIVISION
---------
WESTERN
WESTERN
CORPORATE
EASTERN
EASTERN
EASTERN
MIDWEST
MIDWEST
```

DISTINCT can also select distinct combinations of data, for example:

```
SELECT DISTINCT DEPT, JOB
 FROM Q.STAFF
 ORDER BY DEPT
```

The report resulting from this example shows the distinct combinations of department number and job; or, for every department, the jobs represented in it.

When using DISTINCT, remember these things:

- DISTINCT comes *after* SELECT.
- DISTINCT comes *before* the first column name and is *not* separated from the column name with a comma.
- DISTINCT applies to *all* the columns being selected.

DISTINCT is also a special case of COUNT (see "COUNT" on page 134). There are two uses of COUNT:

- COUNT(*) is *not* used with a column name.
- COUNT(DISTINCT colname) *must* be used with a column name and *cannot* be used with an expression.

Use DISTINCT with other column functions when you want only the DISTINCT values for the columns within a group to be used. For example, AVG(DISTINCT PRICE) ignores duplicate prices in the column and just averages a list in which each price appears once. AVG(PRICE) averages all the prices in the column without regard to the fact that some are duplicates of others.

Write a column function like this:

```
COUNT(DISTINCT expression)
```

The parentheses are necessary.

Example of a COUNT(DISTINCT column function):

DISTINCT

```
SELECT COUNT(DISTINCT EDLEVEL), AVG(EDLEVEL)
FROM Q.APPLICANT
```

Examples:

- List the different values that appear for YEARS:

  ```
  SELECT DISTINCT YEARS
  FROM Q.STAFF
  ORDER BY YEARS
  ```

- List the department numbers for departments in which at least one employee has 10 or more years of service:

  ```
  SELECT DISTINCT DEPT
  FROM Q.STAFF
  WHERE YEARS >= 10
  ```

DROP

The DROP statement erases tables, views, synonyms, aliases, and other things (like indexes and authorizations) from the database. You must have authority to drop tables or views from the database. To drop a synonym, you must be the owner of the synonym. To drop an alias, you must be the owner or have SYSADM or SYSCTRL authority.

The syntax of the DROP statement is:

DROP *object object-name*

object TABLE, VIEW, SYNONYM, or ALIAS

object-name
The name by which the object is known in the database.

For example:

This statement:
Erases this object:

DROP TABLE PERS
The table PERS

DROP VIEW D42
The view D42

DROP SYNONYM APPLS
The synonym APPLS

DROP ALIAS PETROCK
The alias PETROCK

Attention: Use DROP TABLE with extreme caution. Dropping a table destroys the data in it, and hence destroys any view based on the table. It also revokes any authorization granted on the table, or on any view based on the table.

Running any of the following:

```
DROP TABLE name
DROP VIEW name
DROP SYNONYM name
DROP ALIAS name
```

is equivalent to running the single QMF command:

```
ERASE TABLE name
```

DROP VIEW does not affect any tables the view is based on, and does not destroy tables in the database. A view that was dropped can easily be created again. However, DROP VIEW revokes any authorization granted on the view.

DROP SYNONYM removes the synonym from a dictionary of synonyms, so it no longer refers to anything in the database. It has no effect on the tables or views the synonym accessed. If APPLS is in the synonym table for Q.APPLICANT, executing the example query DROP SYNONYM APPLS does not affect Q.APPLICANT. The query removes APPLS from a dictionary in the synonym table, so it no longer refers to anything in the database.

EXISTS

The EXISTS statement determines whether a row exists that satisfies a given condition, as shown in the subquery of the following query:

```
SELECT ID, NAME, DEPT
FROM Q.STAFF CORRVAR
WHERE EXISTS
  (SELECT * FROM Q.ORG WHERE MANAGER = CORRVAR.ID)
```

See “IN” on page 149 for other methods of conditionally selecting values.

GRANT

The GRANT statement gives users authorization to perform one or more operations on a table. You must be authorized to INSERT, UPDATE, DELETE, ALTER, or SELECT rows in a table you do not own. Authorization must be granted by the creator of the table or by someone to whom the creator granted such authorization. (See also “REVOKE” on page 160.)

The syntax of the GRANT statement is:

```
GRANT operation-list ON tablename
TO user-list WITH GRANT OPTION
```

operation-list
: One or more of the following, separated by commas: ALTER, DELETE, INSERT, SELECT, UPDATE (*column-list*). ALL grants authorization to do all operations.

tablename
: Names a table or view for which the authorization is granted.

user-list
: Lists each user ID with commas between. PUBLIC can be specified in place of *user-list* to grant authorization to all users.

WITH GRANT OPTION SQL keyword
: Authorizes another user to use the GRANT keyword to grant the same authorization to other users. It is optional.

This statement:

```
GRANT SELECT ON PERS TO PUBLIC
```

Grants authorization to all other users to write SELECT queries using table PERS.

This statement:

```
GRANT INSERT, DELETE ON PERS TO HSAM4419
```

Grants authorization to user HSAM4419 to insert and delete rows in PERS.

This statement:

```
GRANT UPDATE ON PERS TO SMITH WITH GRANT OPTION
```

Grants authorization to SMITH to update PERS and to grant this authorization to other users.

For more information on granting authorization, see *Managing QMF* for your operating system.

GROUP BY

GROUP BY identifies a selected column to use for grouping results. It divides the data into groups by the values in the column specified, and returns one row of results for each group. You can GROUP BY more than one column name (separate column names with commas). Always place GROUP BY *after* FROM and WHERE in a query, and *before* HAVING and ORDER BY.

All selected columns without an associated aggregation must appear in the GROUP BY clause.

GROUP BY accumulates the results by group, but does not necessarily order the groups; you need ORDER BY to do that. When you retrieve multiple rows from a table, the GROUP BY, HAVING, and ORDER BY clauses can be used to indicate:

- How you want the rows grouped (GROUP BY)
- A condition that the rows, as a group, must meet (HAVING)
- The order in which you want the rows returned to you (ORDER BY)

The following query selects the average salary for each department.

This query:

```
SELECT DEPT, AVG(SALARY)
FROM Q.STAFF
GROUP BY DEPT
```

Produces this report:

```
    DEPT         AVG(SALARY)
  ------  ------------------
      10   20865.8625000000
      15   15482.3325000000
      20   16071.5250000000
      38   15457.1100000000
      42   14592.2625000000
      51   17218.1600000000
      66   17215.2400000000
      84   16536.7500000000
```

In the above example, GROUP BY divides the table into groups of rows with the same department number, and returns one row of results for each group. DEPT can be selected without a built-in function because it is used with GROUP BY, and because every member of each group has the same DEPT. As stated above, all column names included in a SELECT clause must either have an associated built-in function or must appear in the GROUP BY clause. For example, if DEPT is not used in the GROUP BY clause (in the example above), the list of average salaries has little meaning.

This is correct:

```
SELECT DEPT, AVG(SALARY), JOB
FROM Q.STAFF
GROUP BY DEPT, JOB
```

This is incorrect:

```
SELECT DEPT, AVG(SALARY), JOB
FROM Q.STAFF
GROUP BY DEPT
```

Generally, GROUP BY produces one row of a report for each different value of the grouping column. When there are several columns named in the GROUP BY clause, a different group of rows is produced each time a value in one of the columns changes. However, if there are null values in the column, each null value is treated as a separate group, consisting of one member.

Using GROUP BY in SQL is an alternative to using the usage code GROUP on the form (as described in "GROUP Usage Code" on page 259). GROUP BY provides an extension to the grouping that can be specified on the form and it allows *conditional* selection of data, which cannot be done on the form. For example, to see the smallest, largest, and average of total department salaries:

1. Write and run this query:

   ```
   SELECT DEPT, SUM(SALARY), SUM(SALARY), SUM(SALARY)
   FROM Q.STAFF
   GROUP BY DEPT
   ```

2. And use these usage codes on the form:

   ```
   NUM COLUMN HEADING                    USAGE
   --- --------------------------------  -------
    1  DEPT
    2  SUM(SALARY)                       MINIMUM
    3  SUM(SALARY)1                      AVERAGE
    4  SUM(SALARY)2                      MAXIMUM
   ```

The report contains four columns, of which the last three are almost identical. All three show the total salary for each department; but the final row shows the minimum, average, and maximum of the totals.

Examples:

- List the largest and smallest salary by job for each department, excluding managers:

  ```
  SELECT DEPT, JOB, MIN(SALARY), MAX(SALARY)
  FROM Q.STAFF
  WHERE JOB < >'MGR'
  GROUP BY DEPT, JOB
  ```

- List, for each number of years of service, the number of employees with that number of years and their average salaries:

```
SELECT YEARS, COUNT(*), AVG(SALARY)
FROM Q.STAFF
GROUP BY YEARS
```

Remember that HAVING *must* be used with grouped data. When the HAVING statement and the GROUP BY statement are both used, the HAVING statement must follow the GROUP BY statement.

- List the smallest, largest, and average salary in each department, excluding managers, for departments with an average salary greater than 12000:

```
SELECT DEPT, MIN(SALARY), MAX(SALARY), AVG(SALARY)
FROM Q.STAFF
WHERE JOB < > 'MGR'
GROUP BY DEPT
HAVING AVG(SALARY) > 12000
```

- List, for each number of years of service, the number of employees with that number of years and their average salaries, but only for groups with more than 2 employees:

```
SELECT YEARS, COUNT(*), AVG(SALARY)
FROM Q.STAFF
GROUP BY YEARS
HAVING COUNT(*) > 2
```

HAVING

The HAVING clause filters results obtained by the GROUP BY clause. In the following example, the clause HAVING COUNT(*) > 4 eliminates all departments with four or fewer members from the final result. It is similar to the example shown in “GROUP BY” on page 145.

This query:

```
SELECT DEPT, AVG(SALARY)
FROM Q.STAFF
GROUP BY DEPT
HAVING COUNT(*) > 4
```

Produces this report:

```
 DEPT      AVG(SALARY)
------ ---------------
   38  15457.110000000
   51  17218.160000000
   66  17215.240000000
```

Both WHERE and HAVING eliminate data from your report. The WHERE condition is used with column selection. It determines whether an individual row is included. The HAVING condition is used with built-in functions. It determines whether a whole group is included.

HAVING is always followed by a column function (such as SUM, AVG, MAX, MIN, or COUNT). HAVING can also be followed by a subquery that finds a grouped value to complete the HAVING condition. Use WHERE to eliminate unwanted *row* data and HAVING to eliminate unwanted *grouped* data.

For example:

This is correct: `HAVING MIN(YEARS) > 6`

This is incorrect: `HAVING YEARS > 6`

Example 1

List the smallest, largest, and average salary in each department, excluding managers, for departments with an average salary greater than 12000:

```
SELECT DEPT, MIN(SALARY), MAX(SALARY), AVG(SALARY)
FROM Q.STAFF
WHERE JOB < > 'MGR'
GROUP BY DEPT
HAVING AVG(SALARY) > 12000
```

Produces this report:

```
  DEPT  MIN(SALARY)  MAX(SALARY)         AVG(SALARY)
------  -----------  -----------  ------------------
    15     12258.50     16502.83   13756.5100000000
    20     13504.60     18171.25   15309.5333333333
    38     12009.75     18006.00   14944.7000000000
    42     10505.90     18001.75   13338.7500000000
    51     13369.80     19456.50   16235.2000000000
    66     10988.00     21000.00   16880.1750000000
    84     13030.50     17844.00   15443.0000000000
```

Remember that HAVING can only be used with grouped data. When the HAVING statement and the GROUP BY statement are both used, the HAVING statement must follow the GROUP BY statement.

Example 2

List, for each number of years of service, the number of employees with that number of years and their average salaries, but only for groups with more than 2 employees:

```
SELECT YEARS, COUNT(*), AVG(SALARY)
FROM Q.STAFF
GROUP BY YEARS
HAVING COUNT(*) > 2
```

Produces this report:

```
 YEARS  COUNT(EXPRESSION 1)         AVG(SALARY)
------  -------------------  ------------------
     5                    5   15552.0400000000
```

```
     6                  6    16930.0250000000
     7                  6    18611.8050000000
    10                  3    20162.6000000000
     -                  4    13694.0625000000
```

IN

You can retrieve data from each row whose column, named in the WHERE clause, has a value equal to one of several listed values using OR. When applying search conditions to a column, sometimes it is easier to use the IN statement instead of multiple OR statements. When IN is used, at least two values must be specified within the parentheses. Enclose the list of values (excluding NULL, which cannot be used with IN) in parentheses. Separate one value from the next with a comma; a blank between values is optional.

The order of the objects in the list is not important; you get the same rows in any case. The order of objects in the list does not affect the ordering of the result. To order the result, use ORDER BY.

This query:

```
SELECT DEPTNUMB, DEPTNAME
FROM Q.ORG
WHERE DEPTNUMB IN (20, 38, 42)
```

Produces this report:

```
DEPTNUMB DEPTNAME
-------- --------------
      20  MID ATLANTIC
      38  SOUTH ATLANTIC
      42  GREAT LAKES
```

In the query above, `IN(20, 38, 42)` is equivalent to `(DEPTNUMB = 20 OR DEPTNUMB = 38 OR DEPTNUMB = 42)`.

Examples:

- Select every department in the Eastern and Midwestern divisions:

  ```
  SELECT DEPTNAME, DIVISION, LOCATION
  FROM Q.ORG
  WHERE DIVISION IN ('EASTERN', 'MIDWEST')
  ```

- Select every salesperson and clerk in departments 15, 20, and 38:

  ```
  SELECT ID, NAME, JOB, DEPT
  FROM Q.STAFF
  WHERE JOB IN ('CLERK', 'SALES')
  AND DEPT IN (15, 20, 38)
  ```

- Select everyone with 1, 2, or 3 years of service, or whose years value is null:

```
SELECT ID, NAME, YEARS
FROM Q.STAFF
WHERE YEARS IN (1, 2, 3) OR YEARS IS NULL
```

INSERT INTO

INSERT is an SQL statement that adds data to a table.

The INSERT statement has the format:

```
INSERT INTO tablename
VALUES (value1, value2, ...)
```

where *tablename* is the name of the table or view into which you want to insert data, and *value1, value2,* and so on, are the values you insert.

The list of data values after VALUES must correspond with the list of columns in the table into which they are inserted. There must be the same number of values as columns, and each value must have a data type that agrees with its column. As shown in the following example, null values can be inserted by writing NULL.

This statement:

```
INSERT INTO PERS
VALUES (400, 'HARRISON', 20, 'SALES', NULL, 18000.66, 0)
```

Inserts this line into the table PERS:

```
   ID  NAME         DEPT  JOB     YEARS     SALARY       COMM
------ ---------  ------  -----  ------  ---------  ---------
  400  HARRISON       20  SALES       -   18000.66       0.00
```

Table PERS is a copy of Q.STAFF; instructions for creating it are in “CREATE TABLE” on page 136. If you don’t want to use the CREATE TABLE statement, you can also create PERS with these two commands:

```
DISPLAY Q.STAFF
SAVE DATA AS PERS
```

Insert Some Column Values in a Row

If you want to insert a row without providing values for all of the columns in a row, you can use a list of columns with the INSERT statement.

Specify the values you want to insert into the columns, as in this example:

```
INSERT INTO PERS (ID, NAME, JOB, SALARY)
VALUES (510, 'BUCHANAN', 'CLERK', 11500.75)
```

An easy way to create an INSERT query is by using the DRAW command with the option, (TYPE=INSERT. Columns for which values are not specified are given no value (NULL). If a column is defined as NOT NULL, you must specify values for it.

Copy Rows from One Table to Another

Rows can be inserted into a table by copying data from another table and identifying columns to be inserted with a subquery instead of using the VALUES clause with INSERT. The information retrieved by the subquery is placed into the table as if multiple INSERT commands had been entered.

The following statement copies the ID, NAME, JOB, and YEARS columns for members of Department 38 from Q.STAFF into PERS:

```
INSERT INTO PERS (ID, NAME, JOB, YEARS)
SELECT ID, NAME, JOB, YEARS
FROM Q.STAFF
WHERE DEPT = 38
```

Values must be specified for all columns that are defined as NOT NULL.

A one-to-one correspondence does *not* have to exist between columns being selected and columns being inserted; however, there should not be more columns selected than there are columns being inserted. If fewer columns are being selected than are being inserted, the remaining columns are inserted with nulls. Rows cannot be selected for insertion into the same table.

For information about authorization, see “GRANT” on page 143.

IS

The IS keyword is used only with NULL and NOT NULL. See “NULL” on page 156 for examples.

LIKE

To select character data when you only know part of a value, use LIKE in a WHERE clause, plus a symbol for the unknown data:

- A percent sign (%) is the symbol for any number of characters or none.
- An underscore (_) is the symbol for any single character. Use more than one underscore in succession to represent an exact number of unknown characters.

You can also use % and _ together. For example, to select every name with AN or ON as the second and third letters:

```
SELECT ID, NAME
FROM Q.STAFF
WHERE NAME LIKE '_AN%' OR NAME LIKE '_ON%'
```

LIKE can be used only with character and graphic data. For character data, the value after LIKE must always be enclosed in single quotation marks. If you are using graphic data, the value after LIKE must be preceded by the single-byte character 'G' enclosed in single quotation marks. The percent sign and the underscore must be double-byte characters.

Select a String of Characters: LIKE '%abc%'

You can select rows containing a string of characters that might be part of a word or number you know exists in the data. In the following example, WHERE ADDRESS LIKE '%NY' means, “Where the address ends with 'NY' with anything before that.” The percent sign (%) stands for anything—any number of preceding characters or none.

This query:

```
SELECT NAME, ADDRESS
FROM Q.APPLICANT
WHERE ADDRESS LIKE '%NY'
```

Produces this report:

```
NAME      ADDRESS
--------- -----------------
JACOBS    POUGHKEEPSIE, NY
REID      ENDICOTT, NY
LEEDS     EAST FISHKILL, NY
```

When using LIKE to search for data with a specific ending, be aware of the data type of the column you are searching. If the column has a fixed width and the data in the column varies in width, add blanks to the character string to match the blanks in the column data.

For example, if the ADDRESS column in the example has a data type of CHAR(17), the width of the column is fixed with blanks filling the space where the data is not as wide as the column. Searching with an ending character string requires that you anticipate, and search for, the string with every possible number of trailing blanks that could be encountered in the data.

If the ADDRESS column has a data type of VARCHAR, the width of the column varies with the data in it, because blanks are not appended to the data. In the database, no blanks follow the data in each row of the column.

Example:

Select everyone whose name begins with W:

```
SELECT ID, NAME
FROM Q.STAFF
WHERE NAME LIKE 'W%'
```

Ignore Characters: LIKE '_a_'

You can also use the underscore (_) to specify a character string that ignores a given number of characters. Use a specific number of underscores to specify that the same number of characters are to be ignored in the search. For example,

```
WHERE PARTNO LIKE '_G2044_ _'
```

is used to search a column of 8-character part numbers for the combination "G2044" occurring in positions 2 through 6. The first and last two characters are ignored.

MVS requires single quotes around an all-digit value.

Examples:

- Select every name that has an S in some position after the first:

  ```
  SELECT ID, NAME
  FROM Q.STAFF
  WHERE NAME LIKE '_%S%'
  ```

- Select every name that ends in SON:

  ```
  SELECT ID, NAME
  FROM Q.STAFF
  WHERE NAME LIKE '%SON'
  ```

  This example works because the NAME column has data type VARCHAR, which has no blanks following it in the database. If a column has data type CHAR, with a fixed width, the query has to anticipate all lengths of names ending in SON, and has to include those combinations in the search value.

MAX and MIN

MAX and MIN operate on columns that contain character, graphic, numeric, or date/time values.

You can write a column function like this:

`MAX(`*`expression`*`)` or `MIN(`*`expression`*`)`

The parentheses are required. *expression* is most often a column name, but can be:

- An arithmetic expression containing at least one column name.
- DISTINCT, followed by a column name.

A column name in a function must not refer to a long string column or a column derived from a column function. (A column of a view can be derived from a function.) Column functions cannot be nested within other column functions.

The data type of the result of the MAX or MIN function always allows nulls even if the operand of these functions is NOT NULL. Null values are not included in the calculation made by a built-in function.

The following example includes more than one column function in the SELECT statement. It calculates and displays, for department 10, the sum of employee salaries, the minimum, average, and maximum salary, and the number of employees (COUNT) in the department.

```
SELECT SUM(SALARY), MIN(SALARY), AVG(SALARY),
   MAX(SALARY), COUNT(*)
FROM Q.STAFF
WHERE DEPT = 10
```

If you use MAX or MIN with character data, remember that a binary collating sequence is applied when comparing data.

NOT

You can exclude a condition by putting NOT before it. The following example selects all divisions that are not EASTERN or WESTERN.

This query:

```
SELECT DEPTNUMB, LOCATION,
 DIVISION FROM Q.ORG
WHERE NOT
 (DIVISION = 'EASTERN' OR DIVISION = 'WESTERN')
```

Produces this report:

```
DEPTNUMB LOCATION       DIVISION
-------- -------------  ---------
      10  NEW YORK       CORPORATE
      42  CHICAGO        MIDWEST
      51  DALLAS         MIDWEST
```

To make it clear what the NOT condition applies to, use parentheses. If you use NOT with AND or OR without parentheses, conditions preceded by NOT

are negated before they are connected by AND or OR. That is, if A, B, and C are conditions, these two phrases are equivalent:

```
NOT A AND
B OR C means ((NOT A) AND B) OR C
```

With greater than, less than, or equals, NOT must precede the entire condition, as in WHERE NOT YEARS = 10. You can also negate the equal sign with the not symbol (¬).

These are correct:

```
WHERE YEARS ¬ > 10
WHERE NOT YEARS = 10
```

This is incorrect:

```
WHERE YEARS NOT = 10
```

The symbol ¬= is an alternative operator for < > (not equal to). It is an ANSI SQL operator. If you are using remote data access, the preferred symbol is < >.

NOT with NULL, LIKE, IN, and BETWEEN

You can use NOT NULL, NOT LIKE, NOT IN, or NOT BETWEEN. For example:

```
WHERE YEARS IS NOT NULL
```

It is *only* in these cases that NOT can follow the entire condition.

Examples:

- Select everyone whose salary is NOT between $17,000 and $21,000:

```
SELECT ID, NAME, SALARY
FROM Q.STAFF
WHERE SALARY NOT BETWEEN 17000 AND 21000
```

- Select everyone who does NOT earn a salary less than $18,000 and also earns a commission less than $500:

```
SELECT ID, NAME, SALARY, COMM
FROM Q.STAFF
WHERE NOT (SALARY < 18000 AND COMM < 500)
```

- Select only managers in Q.STAFF who are NOT managers of departments in the Q.ORG table:

```
SELECT ID, NAME, DEPT
FROM Q.STAFF
WHERE JOB = 'MGR'
AND ID NOT IN (SELECT MANAGER FROM Q.ORG)
```

NULL

If a table is created and only partially filled with data, the locations in which nothing is entered contain a code word called NULL, meaning "value unknown." NULL is *not* the same as any of these values:

- A numerical value of zero
- A character string of all blanks
- A character string of length zero
- The character string NULL (of length 4)

Each of these is a legitimate value that can be entered in some row and column of some table. NULL occurs where no value was entered, or where the value was specifically set to NULL. It prints and displays as a single hyphen (-).

This is correct: `WHERE columnname IS NULL`

This is incorrect: `WHERE columnname = ' '`

The VALUE scalar function can be used to change how a null is printed and displayed. See "String Functions" on page 178.

To select rows that have NULL in a column, enter:

```
WHERE columnname IS NULL
```

Examples:

- Select everyone who does not receive a commission:

```
SELECT ID, NAME
FROM Q.STAFF
WHERE COMM IS NULL
```

- Select everyone whose commission is zero:

```
SELECT ID, NAME
FROM Q.STAFF
WHERE COMM = 0
```

  0 (zero) is not the same as NULL. No row in the sample table satisfies this condition.

- Select everyone who *does* get a commission:

```
SELECT ID, NAME
FROM Q.STAFF
WHERE COMM IS NOT NULL
```

OR

You can select rows based on multiple conditions connected by AND or OR. Two conditions connected by OR select every row that satisfies either one of the conditions.

This query:

```
SELECT ID, NAME, YEARS, SALARY
FROM Q.STAFF
WHERE YEARS = 10 OR SALARY > 20000
```

Produces this report:

```
  ID  NAME       YEARS     SALARY
----- ---------  -----  ---------
  50  HANES         10   20659.80
 140  FRAYE          6   21150.00
 160  MOLINARE       7   22959.20
 210  LU            10   20010.00
 260  JONES         12   21234.00
 290  QUILL         10   19818.00
 310  GRAHAM        13   21000.00
```

Compare the results of OR with “AND” on page 130.

For information on how parentheses clarify the meaning of a query, see “Parentheses” on page 131.

ORDER BY

As part of the SQL SELECT statement, you can specify the sequence in which selected rows are displayed. You can also eliminate duplicate rows in a selection.

ORDER BY specifies the order in which rows appear in a report. If you use ORDER BY, it must be the last clause in the entire statement. Any columns named after ORDER BY must also be named after SELECT.

The format of the ORDER BY clause is:

```
ORDER BY columnname DESC     (for descending order)
```

If you do not specify an ordering sequence, ascending order is assumed.

The following report shows rows in *ascending* order.

This query:

```
SELECT NAME, JOB, YEARS
FROM Q.STAFF
WHERE DEPT = 84
ORDER BY JOB
```

Produces this report:

```
NAME      JOB    YEARS
--------- -----  -----
GAFNEY    CLERK      5
QUILL     MGR       10
DAVIS     SALES      5
EDWARDS   SALES      7
```

Sorting Sequence

The sequence for sorting character data in numeric order is:

1. Special characters, including blank
2. Lowercase letters, in alphabetic order
3. Uppercase letters, in alphabetic order
4. Numbers
5. NULL

The sequence for sorting numbers is ascending order. The sequence for sorting DATE, TIME, and TIMESTAMP values is chronological. The sequence for sorting DBCS data is determined by the internal value of the data and generally is not meaningful.

Examples:

- List employees in descending order by salary:

  ```
  SELECT ID, NAME, SALARY
  FROM Q.STAFF
  ORDER BY SALARY DESC
  ```

- List employees in ascending order by last name:

  ```
  SELECT ID, NAME, SALARY
  FROM Q.STAFF
  ORDER BY NAME
  ```

Order by More than One Column

To order by more than one column, put the column name or the column number in a list after ORDER BY. You can mix column names and column numbers in the same list.

If you want to order by a defined column, you *must* use its column number. See “Order Columns by Column Number” on page 160.

A column name in an ORDER BY clause, possibly followed by ASC or DESC, is a sort specification. Sort specifications in a list are separated by commas. The first column that follows the ORDER BY clause is put in order first, the second column is ordered within the limits of the first ORDER BY column, and so on.

To order by years within job:

```
SELECT NAME, JOB, YEARS
FROM Q.STAFF
WHERE DEPT=84
ORDER BY JOB, YEARS DESC
```

Produces this report:

```
NAME      JOB    YEARS
--------- -----  -----
GAFNEY    CLERK      5
QUILL     MGR       10
EDWARDS   SALES      7
DAVIS     SALES      5
```

To order by job within years:

```
SELECT NAME, JOB, YEARS
FROM Q.STAFF
WHERE DEPT=84
ORDER BY YEARS DESC, JOB
```

Produces this report:

```
NAME      JOB    YEARS
--------- -----  -----
QUILL     MGR       10
EDWARDS   SALES      7
GAFNEY    CLERK      5
DAVIS     SALES      5
```

Examples:

- List employees in descending order by years of service, and within each year, in descending order by salary:

  ```
  SELECT YEARS, ID, NAME, SALARY
  FROM Q.STAFF
  ORDER BY YEARS DESC, SALARY DESC
  ```

- List employees in ascending order by salary within department:

  ```
  SELECT DEPT, ID, NAME, SALARY
  FROM Q.STAFF
  ORDER BY DEPT, SALARY
  ```

Order Columns by Column Number

To order by a column defined by an expression, use its column number, as in this example:

```
SELECT ID, NAME, SALARY+COMM
FROM Q.STAFF
WHERE COMM IS NOT NULL
ORDER BY 3
```

You cannot use an expression like SALARY+COMM after ORDER BY.

You can use more than one column number in a list after ORDER BY, and you can use column names and column numbers in the same list. For example, in the query above, SALARY+COMM is column 3 and NAME is column 2. The last line of the query can be written:

```
ORDER BY 3 DESC, NAME
```

To list employees in descending order by salary within department:

```
SELECT DEPT, ID, NAME, SALARY
FROM Q.STAFF
ORDER BY 1, 4 DESC
```

REVOKE

The REVOKE statement removes authorization allowed by a GRANT statement. The syntax of the REVOKE statement is:

REVOKE *operation-list* ON *tablename* FROM *user-list*

operation-list
: Lists one or more of the following, separated by commas: ALTER, DELETE, INSERT, SELECT, UPDATE, or ALL to revoke authorization to do any operations.

tablename
: Names the table or view for which the authorization is revoked.

user-list
: Lists each user ID with commas between. PUBLIC can be specified in place of *user-list*. The use of PUBLIC does not revoke a privilege from any user ID for which authorization was specifically granted. Such privilege has to be revoked specifically also.

REVOKE and GRANT are similar, with the following exceptions:

- With REVOKE, you cannot specify a column list after UPDATE. UPDATE revokes the authorization to update any column. To revoke authorization to update specific columns and allow it to remain for others:

1. Revoke the authorization to update any column.
2. Grant the authorization to update a specific list of columns.

- If you grant an authorization to JONES who grants it to JACOBS, and you revoke the authorization from JONES, authorization is also revoked from JACOBS.

The following statement revokes authorization to write SELECT queries using table PERS from user Jacobs:

```
REVOKE SELECT ON PERS FROM JACOBS
```

The following statement revokes authorization to update any column in PERS from user HSAM4419:

```
REVOKE UPDATE ON PERS FROM HSAM4419
```

SELECT

With the SELECT statement, you can specify the name of each column you want to retrieve from a table. You can name one or more columns from a table or view, or you can select all the columns. Each SELECT statement can select information from several tables. See also "DISTINCT" on page 140.

See the SQL reference for your database manager for table, view, and column limits in a SELECT statement.

If your SELECT statement specifies a table with binary data, QMF displays the table only if you provide a form with appropriate hex, bit, or user edit codes to display it reliably.

Select Every Column from a Table

To retrieve *all* the columns from a table, use an asterisk (*) instead of naming the columns. The format of a SELECT statement used for this selection is:

```
SELECT * FROM tablename
```

tablename is the name of the table or view you are searching. For example, this statement produces *all* the columns in Q.ORG:

```
SELECT * FROM Q.ORG
```

This query produces all the columns but only the rows where the department number is 10:

```
SELECT *
FROM Q.STAFF
WHERE DEPT = 10
```

Select Some Columns from a Table

To select some of the columns from a table, enter SELECT, followed by the exact names of the columns, in the same order (left to right) you want them in your report. Separate column names by a comma; blanks are allowed but are not required.

With automatic reordering, the following statement produces a report with the department names on the left and the department numbers on the right:

```
SELECT DEPTNAME, DEPTNUMB
FROM Q.ORG
```

You can change the order of columns in the report by changing the form. But the order of the columns on the form is the same as the order in which they are named in the query.

You can select a column more than once. Doing so allows you to use multiple aggregating functions on the form.

You can select up to 750 column names (or expressions) in MVS; up to 255 in VM and VSE.

You can use a column name in a WHERE clause without using the column name in the SELECT clause.

Examples:

- Select only the ID and NAME columns from the Q.STAFF table:

  ```
  SELECT ID, NAME
  FROM Q.STAFF
  ```

- Select the NAME and ID columns from the Q.STAFF table, and list NAME first:

  ```
  SELECT NAME, ID
  FROM Q.STAFF
  ```

Add Descriptive Columns

You can add a column of purely descriptive information to your report by putting a quoted constant in the column list of your SELECT statement. The constant within surrounding quotation marks can be up to 256 characters in length, and can be alphabetic, numeric, or any combination of alphanumerics. The following example lists the names and addresses of people in the Q.APPLICANT table with 14 years of education, and identifies each as an applicant.

This query:

```
SELECT NAME, ADDRESS, 'APPLICANT'
FROM Q.APPLICANT
WHERE EDLEVEL = 14
ORDER BY NAME
```

Produces this report:

```
NAME      ADDRESS            EXPRESSION 1
--------- -----------------  ------------
CASALS    PALO ALTO,CA       APPLICANT
REID      ENDICOTT,NY        APPLICANT
RICHOWSKI TUCSON,AZ          APPLICANT
```

The report includes three columns: one containing names, one containing addresses, and a newly created column containing the word APPLICANT for every row selected. The database manager adds a column name to the newly created column. This name varies, depending on the database manager used at your installation. You can change this column name using the form panels.

Subqueries

Subqueries select data from a table. The data is then used to test a condition in the WHERE clause of the main query. For example, this query produces a list of employees who work in the Eastern division:

```
SELECT NAME, ID
FROM Q.STAFF
WHERE DEPT = SOME
  (SELECT DEPTNUMB            +
  FROM Q.ORG                  ¦  subquery
  WHERE DIVISION='EASTERN')   +
```

First, the subquery finds the department numbers in the Eastern division. Then, the main query finds the employees who work in *some* of these departments.

When there are several subqueries, the last one is executed first; the first one last.

Examples:

```
SELECT DEPT, NAME, SALARY
FROM Q.STAFF CORRVAR
WHERE SALARY =
  (SELECT MAX(SALARY)              ┐
  FROM Q.STAFF                     │   subquery
  WHERE DEPT = CORRVAR.DEPT)       ┘
```

```
SELECT ID, NAME
FROM Q.STAFF
WHERE DEPT IN
  (SELECT DISTINCT DEPTNUMB      ┐
  FROM Q.ORG                     │   subquery
  WHERE DIVISION = 'MIDWEST')    ┘
ORDER BY ID

SELECT DEPT, AVG(SALARY)
FROM Q.STAFF
GROUP BY DEPT
HAVING AVG(SALARY) >
   (SELECT AVG(SALARY) FROM Q.STAFF)  ]  subquery
```

SOME

Use the SOME keyword with comparison operators to permit a query to return a set of values, rather than a single value. You can use SOME with the following comparison operators:

```
=   ¬=   >   >=   <   <=   < >
```

The symbol ¬= is an alternative symbol for < > (not equal to). It is an ANSI SQL operator. If you are using remote data access, the preferred symbol is < >.

ALL, ANY, and IN can also be used to return a set of values:

- When ALL is used, all values in the set returned are satisfied.
- When ANY or SOME is used, at least one value in the set returned is satisfied.
- IN can be used in a subquery in place of either = SOME or = ANY.

The following query produces a list of employees who work in the Eastern division. First, the subquery finds the department numbers in the Eastern division. Then, the main query finds the employees who work in *some* of these departments.

```
SELECT NAME, ID
FROM Q.STAFF
WHERE DEPT = SOME
   (SELECT DEPTNUMB FROM Q.ORG WHERE DIVISION='EASTERN')
```

The keyword SOME is used in this query because there are multiple departments in the Eastern Division. If ALL is used instead of SOME (or ANY), the result is an empty set. No employee works in *all* the departments of the Eastern division.

SUM

SUM is valid only on columns that contain numeric values.

The data type of the result of the SUM always allows nulls, even if the operand of these functions is NOT NULL. Null values are not included in the calculation made by a built-in function.

The following example includes more than one column function in the SELECT statement. It calculates and displays, for Department 10, the sum of employee salaries, the minimum, average, and maximum salary, and the number of employees (COUNT) in the department.

This query:

```
SELECT SUM(SALARY), MIN(SALARY), AVG(SALARY),
   MAX(SALARY), COUNT(*)
FROM Q.STAFF
WHERE DEPT = 10
```

Produces this report:

```
SUM(SALARY) MIN(SALARY)      AVG(SALARY) MAX(SALARY) COUNT(EXPRESSION)
----------- ----------- --------------- ----------- -----------------
  83463.45     19260.25 20865.8625000000    22959.20                 4
```

You can write a column function like this:

SUM(*expression*)

The parentheses are required. *expression* is most often a column name, but can also be:

- An arithmetic expression containing at least one column name.
- DISTINCT, followed by a column name.

A column name in a function must not refer to a long string column or a column derived from a column function (a column of a view can be derived from a function). Column functions cannot be nested within other column functions.

UNION

UNION merges the rows of two or more tables into one report. To make sense, these rows should relate to one another, have the same width, and have the same data type. Using UNION, you can merge values from two or more tables into the same columns but different rows of the same report. You can use UNION more than once in a query.

Examples in this section that use UNION ALL require enhanced UNION support. See “Appendix C. QMF Functions that Require Specific Support” on page 317.

The following example selects the name and employee columns from Q.STAFF and the name and applicant columns from Q.APPLICANT.

```
SELECT NAME, 'EMPLOYEE '
FROM Q.STAFF
WHERE YEARS < 3
 UNION
SELECT NAME, 'APPLICANT'
FROM Q.APPLICANT
WHERE EDLEVEL > 14
```

Results:

```
NAME      EXPRESSION 1
--------- ------------
BURKE     EMPLOYEE
GASPARD   APPLICANT
JACOBS    APPLICANT
```

The portion of the query that selects from Q.STAFF also creates a column in the report with the constant EMPLOYEE in it. The portion of the query that selects from Q.APPLICANT does the same with the constant APPLICANT. A default column name is assigned to that column, but can easily be changed on the form.

In any query, the lengths of the columns are matched. In the previous example, EMPLOYEE is padded with a blank to match the length of APPLICANT.

The next example selects from Q.STAFF and Q.INTERVIEW all the managers and the people they interviewed:

```
SELECT NAME, '             '
FROM Q.STAFF, Q.INTERVIEW
WHERE MANAGER = ID
UNION
SELECT NAME, 'NO INTERVIEWS'
FROM Q.STAFF
WHERE JOB = 'MGR'
  AND ID NOT IN (SELECT MANAGER FROM Q.INTERVIEW)
```

Results:

```
NAME       EXPRESSION 1
---------  -------------
DANIELS    NO INTERVIEWS
FRAYE
HANES
JONES      NO INTERVIEWS
```

```
LEA
LU          NO INTERVIEWS
MARENGHI    NO INTERVIEWS
MOLINARE
PLOTZ
QUILL
SANDERS
```

Retain Duplicates in UNION

UNION implies that only DISTINCT rows are selected from the columns named in both SELECT statements.

If you want to keep duplicates in the result of a UNION operation, specify the optional keyword ALL after UNION. When UNION ALL is specified, redundant duplicate rows are not eliminated from the result.

The following example selects all sales people in Q.STAFF who have been employed for more than 5 years, or who earn a commission greater than $850. The sales people who meet both conditions appear twice in the resulting report:

```
SELECT * FROM Q.STAFF
WHERE JOB = 'SALES' AND YEARS > 5
UNION ALL
SELECT * FROM Q.STAFF
WHERE JOB = 'SALES' AND COMM > 850
ORDER BY 2
```

Produces this report:

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
340	EDWARDS	84	SALES	7	17844.00	1285.00
340	EDWARDS	84	SALES	7	17844.00	1285.00
310	GRAHAM	66	SALES	13	21000.00	200.30
90	KOONITZ	42	SALES	6	18001.75	1386.70
90	KOONITZ	42	SALES	6	18001.75	1386.70
40	O'BRIEN	38	SALES	6	18006.00	846.55
20	PERNAL	20	SALES	8	18171.25	612.45
70	ROTHMAN	15	SALES	7	16502.83	1152.00
70	ROTHMAN	15	SALES	7	16502.83	1152.00
220	SMITH	51	SALES	7	17654.50	992.80
220	SMITH	51	SALES	7	17654.50	992.80
150	WILLIAMS	51	SALES	6	19456.50	637.65
280	WILSON	66	SALES	9	18674.50	811.50

If UNION rather than UNION ALL is specified, determining which sales people satisfied both conditions requires closer inspection, as shown in this report:

```
   ID  NAME         DEPT  JOB     YEARS      SALARY        COMM
------ ---------  ------  -----  ------  ----------  ----------
  340  EDWARDS        84  SALES       7    17844.00     1285.00
  310  GRAHAM         66  SALES      13    21000.00      200.30
   90  KOONITZ        42  SALES       6    18001.75     1386.70
   40  O'BRIEN        38  SALES       6    18006.00      846.55
   20  PERNAL         20  SALES       8    18171.25      612.45
   70  ROTHMAN        15  SALES       7    16502.83     1152.00
  220  SMITH          51  SALES       7    17654.50      992.80
  150  WILLIAMS       51  SALES       6    19456.50      637.65
  280  WILSON         66  SALES       9    18674.50      811.50
```

With subqueries, the order of evaluation of each subquery has no effect on the result of the operation. However, when you use UNION ALL and UNION to combine two SELECT queries, the result of the operation depends on the order of evaluation. Parentheses are resolved first, starting with the innermost one. Then, each clause is resolved from left to right.

For example, the following queries yield different results:

- In this example, all rows of TABLE1 are merged with all rows of TABLE2 to form an intermediate table, which is merged with TABLE3 with the elimination of duplicates.

  ```
  (TABLE1 UNION ALL TABLE2) UNION TABLE3
  ```

- In this example, all rows of TABLE2 are merged with TABLE3 with the elimination of duplicates, to form an intermediate table that is merged with all rows of TABLE1.

  ```
  TABLE1 UNION ALL (TABLE2 UNION TABLE3)
  ```

Rules for Using UNION

- You can put UNION between two SELECT statements only if the two statements select the same number of columns and the corresponding columns are compatible data types, for example, numeric to numeric or string to string.
- Corresponding columns in select statements merged by UNION need not have the same name. Because the names of the interleaved columns are likely to be different, do *not* use a column name after an ORDER BY. Instead, always use a column number, such as ORDER BY 1.
- The lengths and data types of the columns named in the SELECT statements need only be comparable. They must both be either numeric, character, graphic, date, time, or timestamp. They cannot be a mixture of these groups. For example:

  ```
  SELECT ID
  ⋮

  UNION
  SELECT DEPT
  ⋮
  ```

If ID is CHAR(6) and DEPT is CHAR(3), the column in the result table is CHAR(6). The values in the resulting table that are derived from DEPT are padded on the right with blanks.

When to Use UNION — When to Join

When to use UNION to merge tables and when to *join* tables depends on what kind of results you want in your report.

- UNION interleaves rows from two queries into one report.
- *Joining* tables doesn't interleave the rows, but joins each row from one table horizontally to each row from another table. When joining, it is essential that you use a condition to limit the number of combinations so that every row isn't joined to every other row.

The following query doesn't produce a report that is as readable or meaningful as the UNION query in "UNION" on page 165. Because no common column was used in the WHERE condition in this query to join the two tables, the report contains duplicates.

This query:

```
SELECT S.NAME, 'EMPLOYEE ', A.NAME, 'APPLICANT'
FROM Q.STAFF S, Q.APPLICANT A
WHERE YEARS < 3 AND EDLEVEL > 14
```

Produces this report:

```
NAME      EXPRESSION 1  NAME1      EXPRESSION 2
--------- ------------  ---------  ------------
BURKE     EMPLOYEE      JACOBS     APPLICANT
BURKE     EMPLOYEE      GASPARD    APPLICANT
```

You can also use UNION between two SELECT statements that refer to the same table. For example, to list all employees by number within department, and identify those with ten years of service:

```
SELECT DEPT, ID, NAME, YEARS, 'TEN YEARS'
FROM Q.STAFF
WHERE YEARS = 10
 UNION
SELECT DEPT, ID, NAME, YEARS, '        '
FROM Q.STAFF
WHERE NOT YEARS = 10
ORDER BY 1, 2
```

UPDATE

The UPDATE statement changes the values of specified existing columns in rows of a table. You can update a table only if you created the table, or are specifically authorized to update the table. For information about authorization, see “GRANT” on page 143.

The UPDATE statement consists of three parts:

1. UPDATE specifies the table to update.
2. SET specifies the column to update and the new value to place in the table.
3. WHERE specifies which row to update.

The following example updates table PERS for employee 250: It changes job to “sales” and increases salary by 15%.

```
UPDATE PERS
SET JOB='SALES', SALARY=SALARY * 1.15
WHERE ID = 250
```

An easy way to create an UPDATE query is by using the DRAW command with the option, TYPE=UPDATE.

You can use a single UPDATE statement to update more than one row in a table, as shown in the first of the following examples, or to update all rows for a column (when the WHERE clause is omitted).

Examples:

- Give every clerk in PERS a $300 increase:

  ```
  UPDATE PERS
  SET SALARY = SALARY+300
  WHERE JOB = 'CLERK'
  ```

- Increase everyone’s years of service by 1 in table PERS:

  ```
  UPDATE PERS
  SET YEARS = YEARS + 1
  ```

WHERE

Use WHERE in your SELECT statement to allow QMF to select just those rows from a table that meet a certain condition or set of conditions, without retrieving every row in a table. The WHERE clause specifies a search condition (one or more selection criteria) that identifies the row or rows you want to retrieve, update, or delete.

The search condition of a WHERE clause specifies that a comparison be made between two values. Usually, a column's value is compared with a fixed value specified in the WHERE clause. The only rows selected are the ones that satisfy the search condition. In the following example, the search condition specifies that the value in the DEPT column must be 20.

This query:

```
SELECT DEPT, NAME, JOB
FROM Q.STAFF
WHERE DEPT = 20
```

Produces this report:

```
DEPT NAME        JOB
---- ----------  -----
 20  SANDERS     MGR
 20  PERNAL      SALES
 20  JAMES       CLERK
 20  SNEIDER     CLERK
```

Both WHERE and HAVING eliminate data you don't want in your report:

- The WHERE condition is used with column selection. It determines whether an individual row is included.

  Use WHERE to eliminate unwanted *row* data.

- The HAVING condition is used with built-in functions. It determines whether a whole group is included.

  HAVING is always followed by a column function (such as, SUM, AVG, MAX, MIN, or COUNT). HAVING can also be followed by a subquery that finds a grouped value to complete the HAVING condition.

  Use HAVING to eliminate unwanted *grouped* data.

For example, to list the smallest, largest, and average salary in each department, excluding managers, for departments with an average salary greater than 12000:

This query:

```
SELECT DEPT, MIN(SALARY),
 MAX(SALARY), AVG(SALARY)
FROM Q.STAFF
WHERE JOB < > 'MGR'
GROUP BY DEPT
HAVING AVG(SALARY) > 12000
```

Produces this report:

```
   DEPT  MIN(SALARY)  MAX(SALARY)         AVG(SALARY)
 ------  -----------  -----------  ------------------
     15     12258.50     16502.83    13756.5100000000
     20     13504.60     18171.25    15309.5333333333
```

```
38     12009.75     18006.00    14944.7000000000
42     10505.90     18001.75    13338.7500000000
51     13369.80     19456.50    16235.2000000000
66     10988.00     21000.00    16880.1750000000
84     13030.50     17844.00    15443.0000000000
```

In addition to making an equality comparison (=), you can compare a column value in the following ways. The condition defined in the first column is specified by entering the corresponding words or symbols in the second column.

Condition
Word or Symbol

Equal to
=

Not equal to
< >

Alternative to not equal to
¬=

Greater than
>

Greater than or equal to
>=

Not greater than
¬> (in DB2 only)

Less than
<

Less than or equal to
<=

Not less than
¬< (in DB2 only)

Multiple conditions
AND, OR

Values within a range
BETWEEN x AND y

Values matching any in a list
IN (x, y, z)

Selects a string of characters
LIKE '%abc%'

Ignores certain characters
LIKE '_a_'

Negative conditions
NOT

A not sign (¬) can cause parsing errors in statements passed from one DBMS to another. To avoid this possible problem in statements to be executed at a remote location, substitute an equivalent for any operation in which the not sign appears. For example, substitute <> for ¬=, <= for ¬>, and >= for ¬<.

Values to be compared with columns of character data must be enclosed in single quotes (as in WHERE NAME = 'JONES'). Numeric data is not enclosed in quotes.

If you are using graphic data, the value after WHERE must be preceded by the single-byte character 'G' and be enclosed in single quotation marks. The percent sign and the underscore must be double-byte characters.

Equality and Inequality Symbols in a WHERE Clause

You can write a WHERE search condition using any of the symbols of equality or inequality in "WHERE" on page 170. For example, to select only employees who have made commissions of $1,000 or more:

This query:

```
SELECT ID, COMM
FROM Q.STAFF
WHERE COMM >= 1000
```

Produces this report:

```
  ID     COMM
----- -------
  70  1152.00
  90  1386.70
 340  1285.00
```

Additional examples:

- Select everyone with 10 years of service or more:

  ```
  SELECT ID, NAME, YEARS
  FROM Q.STAFF
  WHERE YEARS >= 10
  ```

- Select everyone with more than ten years of service:

  ```
  SELECT ID, NAME, YEARS
  FROM Q.STAFF
  WHERE YEARS > 10
  ```

- Select every manager:

```
SELECT ID, NAME, YEARS
FROM Q.STAFF
WHERE JOB = 'MGR'
```

- Select everyone whose name occurs later in the alphabet than SMITH:

```
SELECT NAME, ID
FROM Q.STAFF
WHERE NAME > 'SMITH'
```

- Select every employee name in Q.STAFF that is not in department 10:

```
SELECT NAME, ID
FROM Q.STAFF
WHERE DEPT < > 10
```

Calculated Results

You can use calculated values as part of a search condition. You can also display them for selected rows just as you display column values.

You can use an arithmetic expression in the SELECT clause or in the WHERE clause of the query:

- When the expression is part of the SELECT clause, the column with the results of the calculation appears in the report.
- When the expression is part of the WHERE clause, it is part of the search condition and does not alter column values.

The following two queries illustrate the use of an arithmetic expression in a SELECT clause.

- This query selects every employee's *annual* salary from the Q.STAFF table:

```
SELECT ID, SALARY
FROM Q.STAFF
```

- This query selects every employee's *monthly* salary, which must be calculated:

```
SELECT ID, SALARY/12
FROM Q. STAFF
```

  SALARY/12 is called an *expression*. It means the result of dividing SALARY by 12.

This query:

```
SELECT DEPT, NAME, SALARY
FROM Q.STAFF
WHERE DEPT = 38
```

Produces this report:

```
DEPT NAME            SALARY
---- ----------  ----------
 38  MARENGHI      17506.75
 38  O'BRIEN       18006.00
 38  QUIGLEY       16808.30
 38  NAUGHTON      12954.75
 38  ABRAHAMS      12009.75
```

This query:

```
SELECT DEPT, NAME, SALARY/12
FROM Q.STAFF
WHERE DEPT = 38
```

Produces this report:

```
 DEPT  NAME             EXPRESSION 1
------ ---------  ------------------
   38  MARENGHI      1458.8958333333
   38  O'BRIEN       1500.5000000000
   38  QUIGLEY       1400.6916666666
   38  NAUGHTON      1079.5625000000
   38  ABRAHAMS      1000.8125000000
```

Arithmetic operators:

Operator	Operation
+	add
−	subtract
*	multiply
/	divide

Within expressions, you can use column names (as in RATE*HOURS), columns and constants (as in RATE*1.07), and built-in functions (as in AVG(SALARY)/2). An expression can consist of numeric constants (such as 3*7) or character constants (such as SALARY + COMM).

When a table is created, each column in it is defined to hold a certain type of data. Arithmetic operations can be performed only on numeric data types, and the results of an operation can depend on the data types of the operands.

Example:

- Select the name and total earnings (salary plus commission) of every employee who earns more than $20,000 a year:

  ```
  SELECT NAME, SALARY + COMM
  FROM Q.STAFF
  WHERE SALARY + COMM > 20000
  ```

The above query does *not* list anyone whose salary alone is greater than $20,000 when the amount of commission is null. The result of operating on an unknown is unknown.

- List anyone whose commission is 5% or more of their total earnings:

```
SELECT NAME, SALARY, COMM
FROM Q.STAFF
WHERE COMM >= 0.05 * (SALARY + COMM)
```

SQL Scalar Functions

Three types of scalar functions are described here:

- Date/time functions
- Conversion functions
- String functions

Date/Time Functions

The date/time functions do the following:

- DATE, TIME, and TIMESTAMP change the data type of their argument to the associated date/time data type.
- CHAR changes the data type of its argument (a DATE or TIME value) to the CHAR data type.
- DAYS calculates the number of days between one date and another.
- YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, and MICROSECOND select parts of DATE, TIME or TIMESTAMP values.

Each date/time function is followed by an argument enclosed in parentheses. The following example lists the projects, by number, of each project scheduled to begin in 1990. It does this by applying the YEAR date/time function to the STARTD column of the Q.PROJECT table.

This query:

```
SELECT PROJNO, STARTD, ENDD, TIMESTAMP
FROM Q.PROJECT
WHERE YEAR(STARTD) = 1998
```

Produces this report:

```
PROJNO  STARTD      ENDD        TIMESTAMP
------  ----------  ----------  --------------------------
1409    1998-06-15  1999-12-31  1996-03-13-09.12.57.149572
1410    1998-09-29  2000-03-31  1996-03-13-12.18.23.402917
```

Date/time functions (see Table 2, following) can be used wherever an expression can be used. The first or only argument of each of these functions

is an expression giving the value to be manipulated.

Table 2. Date/Time Functions

Function	Argument	Result
DATE	Date, timestamp, or string representation of a date	Date
TIME	Time, timestamp, or string representation of a time	Time
TIMESTAMP	Timestamp, string representation of timestamp, *or* a date or string representation of a date *and* a time or string representation of a time	Timestamp
DAY, MONTH, or YEAR	Date or timestamp, or date duration	Day, month, or year part
HOUR, MINUTE, or SECOND	Time or timestamp, or time duration	Hour, minute, or second part
MICROSECOND	Timestamp	Microsecond part
DAYS	Date, timestamp, or string representation of a date	Days since Dec 31, 0000
CHAR	Date or time and the specified date/time output format	String representation in specified date/time format. If format is not specified, ISO format will be returned.

Conversion Functions

Scalar functions (see Table 3, following) allow the conversion of a value from one data type to another.

Table 3. Conversion Functions

Function and Syntax	Argument	Result
DECIMAL(V,P,S)	V = A number P = Precision of the result S = Scale of the result	Decimal representation of V
DIGITS(argument)	A binary integer or decimal number	A character string representing the digits of the argument
FLOAT(argument)	A number	Single-precision floating point number representing the argument

Table 3. Conversion Functions (continued)

Function and Syntax	Argument	Result
HEX(argument)	Any data type other than a long character or long graphic string	A character string representing actual hex digits of the argument
INTEGER(argument)	A number within the range of binary integers	Fullword representation of the argument
VARGRAPHIC(argument)	Short character string	Graphic string that is the DBCS representation of the argument

This query:

```
SELECT SALARY,          --SALARY
DECIMAL(SALARY,9,3),    --COL1
DIGITS(SALARY),         --COL2
FLOAT(SALARY),          --COL3
HEX(NAME),              --COL4
VARGRAPHIC(JOB)         --COL5
FROM Q.STAFF
WHERE DEPT = 10
```

Produces this report:

```
 SALARY          COL1 COL2             COL3  COL4                COL5
-------- ------------ ------- -------------  ------------------- --------
22959.20    22959.200 2295920  2.295920E+04  D4D6D3C9D5C1D9C5    -M-G-R
20010.00    20010.000 2001000  2.001000E+04  D3E4                -M-G-R
19260.25    19260.250 1926025  1.926025E+04  C4C1D5C9C5D3E2      -M-G-R
21234.00    21234.000 2123400  2.123400E+04  D1D6D5C5E2          -M-G-R
```

String Functions

Three scalar functions (see Table 4, following) enable the manipulation and retrieval of string segments: SUBSTR, LENGTH, and VALUE.

Table 4. String Functions

Function and Syntax	Argument	Result
LENGTH(argument)	Any data type	Integer represents the length of V
SUBSTR(S,N,L)	S: Character or graphic string to be evaluated. N: Binary integer represents the starting position of substring in S. L: Binary integer represents the length of substring.	Substring of S
VALUE(*arg1*,*arg2*)	Arguments must have compatible data type.	A non-null value representing *arg1* if *arg1* is non-null, or representing *arg2* if *arg1* is null.

The length function returns the actual variable length of the data if the data type is VARCHAR; it returns the fixed length if the data type is CHAR.

The following statement lists applicant status for each applicant in the Q.INTERVIEW table who was interviewed by manager 270. For any applicant, if the DISP column was not filled in (and therefore, contains a null value), the result for that row is “unknown” rather than the null symbol (-).

```
SELECT VALUE(DISP, 'unknown')
FROM Q.INTERVIEW
WHERE MANAGER = 270
```

The first or only argument of each of these functions is an expression giving the value to be manipulated or retrieved. For LENGTH, the value of this expression can be any data type. For SUBSTR, the value must be a character string or a graphic string. For VALUE, two or more values must be specified, and their data types must be comparable.

For example, this query finds the first initial and last name of an applicant with the temporary ID number 400.

```
SELECT SUBSTR(FIRSTNAME,1,1)||LASTNAME
FROM Q.INTERVIEW
WHERE TEMPID = 400
```

Concatenation

The concatenation operator (CONCAT) joins two values of an expression into a single string. The alternate operator for CONCAT is ||. Because vertical bars can cause parsing errors in statements passed from one DBMS to another, CONCAT is the preferred operator for statements executed at remote locations.

The concatenation operator observes the following rules:

- The operands of a concatenation operator must both be character strings or both be graphic strings.
- The length of the result is the sum of the lengths of the operands.
- The data type of the result is:
  - VARCHAR when one or more operands is VARCHAR
  - CHAR when both operands are CHAR
  - VARGRAPHIC when one or more operands are VARGRAPHIC
  - GRAPHIC when both operands are GRAPHIC
- If either operand is a null value, the result is the null value. For example:

  ```
  VALUE(FNAME, 'unknown') CONCAT VALUE(LNAME, 'unknown')
  ```

  To avoid a null value, use the VALUE function. For more information on VALUE, see “String Functions” on page 178.
- Concatenation cannot be specified in a LIKE clause, nor in the SET clause of an UPDATE statement.

Examples

- If FNAME is CHAR(6) with a value of BEN, and LNAME is CHAR(8) with a value of JOHNSON, `FNAME CONCAT LNAME` results in `BEN JOHNSON` with a length of 14. (There are 3 blank spaces between the first and the last names.)

  This example requires a specific release of DB2 or SQL/DS. See “Appendix C. QMF Functions that Require Specific Support” on page 317.
- This query lists all last names in Q.INTERVIEW that begin with letters greater than M, and combines those last names with their respective first names.

  ```
  SELECT LASTNAME CONCAT ', ' CONCAT FIRSTNAME
  FROM Q.INTERVIEW
  WHERE LASTNAME > 'M'
  ```

Chapter 3. Forms, Reports, and Charts

QMF creates reports from data stored in your database. A QMF *form* consists of a number of panels used to control report formatting. When you select data (by running a query, importing data, or displaying a table or view), you can use QMF form panels to format the data into a report or chart. You can also use form panels to instruct QMF to perform specific calculations on report data, such as adding columns or calculating percentages.

This chapter shows the QMF form panels and describes the entry areas on each panel. The chapter also includes information on using REXX with QMF forms; edit and usage codes; and variables used in forms.

Using QMF Forms

QMF automatically generates form panels when a table is displayed or a SELECT query is run without specifying a form. The resulting report is based on certain default choices made by QMF about the format of the report. You can see the *default form* by typing `DISPLAY FORM.MAIN` (or `DISPLAY FORM`) after you run a query without specifying a form name on the RUN command.

Each form panel has entry areas to which information is added or changed. In this chapter (beginning with "FORM.MAIN" on page 185), a letter is assigned to each entry area on a panel (such as **C**) and corresponds to the description following the panel. If there is a default value, it is shown in the entry area on the panel. Each entry area is described in terms of its effect on *reports*. If an entry area affects *charts*, that description follows.

Creating Reports in QMF

Reports are initially created by applying a default form to the data retrieved from your query. To alter a report's default format (for example, to change the column widths, add page headings, or change the spacing between lines of a report), you change the data displayed on the form panels. Data entered into an entry area can be translated to uppercase, depending on your profile case option setting.

Display a Report without Any Data

With the LAYOUT command, you can view a report before the data is available. Variable data is displayed using the letters A, B, C, D, E, F, and X,

and the numbers 0, 1, 2, 3, 4, 5, and 6. All other text (including headings) is displayed as entered. You can tailor the different form panels to produce a representative report independent of the data. Combined with the LAYOUT command, forms with complex variables can be used repeatedly. See "LAYOUT" on page 75. For scenarios using the LAYOUT command and using forms to create reports and charts, see *Using QMF*.

Symbols Used in Reports to Indicate Errors

When QMF cannot display a value in a report, it displays a special symbol in place of the value. The symbol that is displayed depends on the underlying cause. Please refer to Table 5 for a list of the symbols and their meaning.

Table 5. QMF Error Symbols

Symbol Displayed	Cause
********	The column is not wide enough to display the formatted value. Only numeric columns display this symbol. (Character columns truncate instead.)
>>>>>>>>	The value exceeds the maximum value allowed by the data type for that column. This is called an *overflow condition*, and is usually detected by QMF.
????????	The value is undefined. The following conditions will result in an *undefined value* in the report: • Numeric underflow • Numeric overflow detected by the database • Dividing a value by zero (in a query, calculation, or column definition) • Expressions that REXX is unable to evaluate • REXX expressions that evaluate to a nonnumeric value • Aggregations calculated using undefined values (except FIRST and LAST)
' ' (blanks)	The data has no instance (DSQNOINS) or no relationship (DSQNOREL).

Quick Reference to Form Panels for Reports

Table 6, following, lists some common additions or changes that alter the format of a report, and lists the appropriate form panel (or panels) you should normally use.

Table 6. Report Quick Reference

To Add or Change:	Use the Form Panel:
Break text	
Default break text	MAIN, OPTIONS
Break text width	OPTIONS

Table 6. Report Quick Reference (continued)

To Add or Change:	**Use the Form Panel:**
Break heading text	BREAK*n*
Break footing text	MAIN, BREAK*n*
Break summary	BREAK*n*
Placement on page	BREAK*n*
Outlining	MAIN, OPTIONS
Calculations	CALC
Column	
Alignment	COLUMNS (Specify)
Definition	COLUMNS (Specify)
Heading	MAIN, COLUMNS
Usage	MAIN, COLUMNS
Indent	MAIN, COLUMNS
Width	MAIN, COLUMNS
Editing	MAIN, COLUMNS
Sequencing	MAIN, COLUMNS
Automatic ordering	OPTIONS
Headings repeated at breaks	BREAK*n*
Headings repeated at detail blocks	DETAIL
Conditional formatting	CONDITIONS
Detail block text	
Remove tabular information	DETAIL
Specify placement of tabular information	DETAIL
Include text with column values	DETAIL
Detail heading text	DETAIL
Final text	
Placement on page	FINAL
Width	OPTIONS
Final summary	FINAL
Fixed columns	OPTIONS
New page	
For breaks	MAIN, BREAK*n*
For detail block text	DETAIL
For final text	FINAL
Page heading and footing	MAIN, PAGE

Table 6. Report Quick Reference (continued)

To Add or Change:	Use the Form Panel:
Associate a panel variation with a condition	DETAIL
Separator lines	OPTIONS
Spacing between detail blocks	OPTIONS, DETAIL

Creating Charts in QMF

Certain entry areas on the form panels determine what appears on a chart, such as chart headings, legends, axis labels, and data plotted on the X- and Y-axes. However, not all entry areas on all panels affect charts. The descriptions of the form panels (beginning with "FORM.MAIN" on page 185) point out both those panels and panel entry areas that affect charts and how these panels can be modified.

Table 7, following, lists some common additions or changes that alter your chart within QMF, and lists the appropriate form panel (or panels) you should normally use.

Table 7. Chart Alteration Panel Quick Reference

To Add or Change:	Use the Form Panel:
Legend labels (Y data column headings)	MAIN, COLUMNS
X-axis data labels (BREAK or GROUP columns)	MAIN, COLUMNS
Y-axis data (numeric data columns)	MAIN, COLUMNS
Chart heading (page heading)	MAIN, PAGE
Vertical position of chart heading	PAGE
Function name in legend label	OPTIONS

FORM.MAIN

Use FORM.MAIN to make simple changes to a report or chart. Other panels (see Table 8, below) work with FORM.MAIN to modify the appearance of reports or charts.

Table 8. Report/Chart Appearance Change Guide

Form Name	Function	See page
FORM.MAIN	Basic format of a report or chart	185
FORM.BREAK*n* (*n* = 1 to 6)	Text before and after breaks in a report	189
FORM.CALC	Expressions for calculations in a report	198
FORM.COLUMNS	Use of columns in a report or chart	203
FORM.CONDITIONS	Expressions for conditional formatting	214
FORM.DETAIL	Text included with column values or headings of a report	216
FORM.FINAL	Content and placement of final text in a report	223
FORM.OPTIONS	Miscellaneous adjustments to a report	229
FORM.PAGE	Content and placement of page headings and footings in a report or chart	236

Everything entered on FORM.MAIN is automatically reflected in a corresponding entry area on one of the other form panels. However, not all of the entry areas on the other panels are reflected on FORM.MAIN.

There are two areas on the FORM.MAIN and FORM.COLUMNS panels that are not entry areas. The `Total Width of Report Columns` and `NUM` areas are described under “Nonentry Areas” on page 187.

```
FORM.MAIN

COLUMNS:              Total Width of Report Columns: 66
          A                              B       C       D       E       F
NUM   COLUMN HEADING                   USAGE    INDENT   WIDTH   EDIT   SEQ
---   ------------------------------  -------  ------   -----   -----  ---
  1   ID                                        2        6       L      1
  2   NAME                                      2        9       C      2
  3   DEPT                                      2        6       L      3
  4   JOB                                       2        5       C      4
  5   YEARS                                     2        6       L      5

  PAGE:       HEADING  ===>        G
              FOOTING  ===>
  FINAL:      TEXT     ===>        H
  BREAK1:     NEW PAGE FOR BREAK? ===> NO
              FOOTING  ===>                  I
  BREAK2:     NEW PAGE FOR BREAK? ===> NO
              FOOTING  ===>
  OPTIONS:    OUTLINE? ===> YES      DEFAULT BREAK TEXT? ===> YES      J

 1=Help        2=Check     3=End     4=Show        5=Chart       6=Query
 7=Backward    8=Forward   9=       10=Insert     11=Delete     12=Report
 OK, FORM.MAIN is displayed.
 COMMAND ===>                                            SCROLL ===> PAGE
```

Entry areas **A** through **F** correspond to identical entry areas on the FORM.COLUMNS panel. If all the columns in the form are not visible on the FORM.MAIN panel, you can scroll forward and backward to see them.

With these entry areas you can:

A Assign column headings (page 203)

B Choose how to process columns (page 205)

C Adjust indentation of columns (page 206)

D Adjust width of columns (page 206)

E Specify formatting of columns (page 208)

F Change the sequence of columns (page 209)

Reports: The order of columns in the form is determined by the way they are specified in the query. Change the order of columns in the report by using the automatic reorder option or by changing the sequence (SEQ) column (**F**) on the FORM.MAIN panel. For a description of the automatic reordering option, see page 234.

Charts: Of these six entry areas, COLUMN HEADING, USAGE, WIDTH, and EDIT apply to charts. The codes that appear in the USAGE entry area affect processing. For more information, see “FORM.COLUMNS” on page 203; “Usage Codes” on page 251; and “Edit Codes” on page 260.

Entry areas **G** through **J** have corresponding form panels. The page number on which these corresponding form panels are described follows the entry area name.

G **PAGE** (page 236)

Reports: Enter one line of page heading and footing text for a report. QMF determines the horizontal and vertical placement of the heading and footing lines. The PAGE entry area corresponds to two entry areas on the FORM.PAGE panel.

Charts: Whatever appears in the PAGE entry area for a report heading also appears on a chart as the heading. Footing text *cannot* be specified for a chart.

H **FINAL** (page 223)

Reports: Enter one line of final text for a report. The default placement of the line can be changed on the FORM.FINAL panel. The FINAL entry corresponds to one entry on the FORM.FINAL panel.

I **BREAK1 and BREAK2** (page 189)

Reports: Enter footing text for up to two levels of breaks, and specify whether to start a new page each time the value in the control column changes. QMF determines the horizontal and vertical placement of the break footings. The BREAK1 and BREAK2 entry areas correspond to entry areas on the FORM.BREAK1 and the FORM.BREAK2 panels.

J **OPTIONS** (page 229)

Reports: Change two options that affect the overall format of a report. For reports with breaks, use the OUTLINE option to determine whether QMF displays the value of the break column on each tabular data line of the report. YES displays the value in the BREAK column only when the value itself changes.

For reports with breaks, use the DEFAULT BREAK TEXT option to determine whether to generate default break footing text to mark the BREAK aggregation line. When you do not enter any break footing text, YES displays a default break footing of asterisks.

This entry area corresponds to two entry areas on the FORM.OPTIONS panel.

Nonentry Areas

Total Width of Report Columns

Reports: This area shows the character width of the columns of the report.

You cannot change this area directly. But when you change INDENT, WIDTH, or edit codes for a column, or use a usage code of OMIT or ACROSS, the new total width of the report columns (in characters) appears after the colon.

If you use an edit code of G with DBCS data, each double-byte character counts as two positions. For more information about calculating the width of a column containing DBCS data, see *Using QMF* .

If you use the usage code ACROSS, the width appears as an algebraic expression of the form: $a + (N \times b)$.

a A constant value

N An unknown that stands for the number of sets of columns that are duplicated across the page, one set for each distinct value in the ACROSS column.

b The width of each group of columns

NUM ***Reports:*** This area shows the number of each column in the order in which it was selected by the query that was run. You cannot change this area, but you can change the order of your columns by using the SEQ entry area.

You can tell QMF which column you want to use as a substitution variable by using the column number. For example, `&6` refers to the sixth column selected by the query, even though it might not appear in the sixth position of the report.

Usually, columns appear on the report from left to right in order by their sequence numbers. However, when you use BREAK, GROUP, or an aggregation function on FORM.MAIN or FORM.COLUMNS and specify YES for `Automatic reordering of report columns?` on FORM.OPTIONS, QMF automatically reorders the columns in a report.

With automatic column reordering, if you use one or more of the BREAK codes as a usage, the control columns are moved to the left of the report. They appear there in order by their BREAK code numbers.

Also, columns whose usage is one of the aggregating usages (AVERAGE, COUNT, FIRST, LAST, CALC*id*, MAXIMUM, MINIMUM, STDEV, SUM, CPCT, CSUM, PCT, TPCT, or TCPCT) are moved to the right of the report and appear there in order by their column numbers.

For more information about width and order of columns, see **C** *Report text line width* (page 231) and **J** *Automatic reordering of report columns* (page 234).

FORM.BREAKn

Use the FORM.BREAK*n* panels to make choices about the text and its placement for up to six breaks in a report. QMF places that text after its associated break in the report.

FORM.BREAK*n* does not affect charts.

Specify a break usage code in the USAGE entry area (**B**) on FORM.MAIN or FORM.COLUMNS opposite one of the column names (see pages 185 and 203). That column then becomes the *control column* and a break occurs in the report whenever the value in this control column changes.

When evaluating values in VARCHAR columns, QMF differentiates between a value padded with blanks or hexadecimal zeros and the same values without these trailing characters. Using FORM.BREAK*n* in such cases creates a break.

You can use the same level of break on multiple columns. In this case, a break occurs when a value changes in any one of those columns.

Area **I** on FORM.MAIN specifies footing text for BREAK1 and BREAK2 in a report and whether to start a new page each time the value in the control column changes. Whatever you specify in area **I** of FORM.MAIN is reflected on FORM.BREAK1 and FORM.BREAK2. What you specify on areas **H** and **N** on BREAK1 and BREAK2 is reflected on FORM.MAIN.

There are six FORM.BREAK*n* panels — one for each possible level of break. They are all the same, except for the panel title.

```
FORM.BREAK1

A New Page for Break?        ===> NO    B Repeat Detail Heading?   ===> NO
C Blank Lines Before Heading ===> 0     D Blank Lines After Heading ===> 0
E LINE F ALIGN G BREAK1 HEADING TEXT
   ----    ------   ----+----1----+----2----+----3----+----4----+----5----+
   1       LEFT
   2       LEFT
   3       LEFT
           *** END ***

H New Page for Footing?      ===> NO    I Put Break Summary at Line ===> 1
J Blank Lines Before Footing ===> 0     K Blank Lines After Footing ===> 1
L LINE M ALIGN N BREAK1 FOOTING TEXT
   ----    ------   ----+----1----+----2----+----3----+----4----+----5----+
   1       RIGHT
   2       RIGHT
   3       RIGHT
           *** END ***

1=Help       2=Check     3=End     4=Show       5=Chart       6=Query
7=Backward   8=Forward   9=       10=Insert     11=Delete     12=Report
OK, FORM.BREAK1 is displayed.
COMMAND ===>                                          SCROLL ===> PAGE
```

A New Page for Break?

Specify whether to begin a new page whenever the value in the control column for the break changes. This value affects printed and exported reports. It does not affect displayed reports. A new page is started if the report is not already at the top of the page.

Specifying YES for more than one break level can produce more pages than expected in your printed or exported report. This happens when multiple breaks occur at the same time.

If you specify two or more breaks and also specify YES for New Page for Break on each break, a page is generated for each specified break whenever the highest break level occurs. Multiple breaks frequently occur together, since the highest break level forces all lower break levels to occur. In particular, all breaks occur for the first row of data in a report.

B Repeat Detail Heading?

Specify whether the detail heading is to be repeated at the beginning of each new break level following the break heading text and before the detail block text.

In printed reports, if a break begins at the top of a page and you specify YES, only one set of detail headings appears.

Detail headings consist of the detail heading text specified on the FORM.DETAIL panel, plus column headings (unless you suppress column headings on the FORM.DETAIL panel). See “FORM.DETAIL” on page 216.

Specifying YES for Repeat Detail Headings on FORM.DETAIL overrides the specifications given here.

C Blank Lines Before Heading

Enter the number of blank lines before the first line of the break heading text, if specified, or before the first break member line if there is no break heading text. The value can be any number from 0 through 999.

D Blank Lines After Heading

Enter the number of blank lines after the last line of the break heading text, if specified. This entry can be any number from 0 through 999.

E LINE

Identify the lines of break heading text and specify their position relative to themselves and to the line at which the break heading starts (as indicated in the Blank Lines Before Heading entry area). You can specify any number from 1 through 999 or a blank. If blank, QMF ignores any associated text.

The numbers you choose need not start with 1 or be consecutive.

For example, these values on FORM.BREAK1:

```
LINE  ALIGN  BREAK1 HEADING TEXT
----  -----  ------------------
3     LEFT   DEPARTMENT &4
2     LEFT   BEGINNING OF LISTING
```

display as:

```
BEGINNING OF LISTING
DEPARTMENT 35
```

Notice that a blank line appears before the first line of text.

F ALIGN

Specify where each line of the break heading text is to be placed horizontally in the report. You can place the lines anywhere in the width of the report. For an online report, the width is the width of the displayed report; for a printed report, the width is the page width.

Left Left-justifies the break heading text.

Right Right-justifies the break heading text.

Center
Centers the break heading text.

n Begins the break heading text in the *n*th position of the line. *n* can be any number from 1 through 999999.

Append

Attaches the line to the end of the previous line of break heading text. If append is used on the first line of break heading text, the line of text is left-justified.

The appended line of text must have the same LINE value as the line of text it is being appended to. If the report is not wide enough to accommodate the appended line of text, some of the text might be truncated.

For example, the following entries on FORM.BREAK1:

```
Blank Lines Before Heading ===> 0
LINE  ALIGN   BREAK1 HEADING TEXT
----  ------  ----------------------
1     LEFT    DEPARTMENT
1     APPEND   &4
3     LEFT
```

align the columns in the resulting report as shown:

```
      DEPT        COMM  JOB        SALARY
    ------  ----------  -----  ----------
  DEPARTMENT 66
        66       55.50  CLERK    10988.00
                    -   MGR      18555.50
                844.00  SALES    16858.20
                200.30  SALES    21000.00
                811.50  SALES    18674.50
                                 ----------
                            *    86076.20

  DEPARTMENT 84
        84      188.00  CLERK    13030.50
                    -   MGR      19818.00
```

G BREAK1 HEADING TEXT

Enter the heading text you want associated with the break. Every time the value in the break column changes, the text specified in this entry is displayed in the report. You can add up to 999 lines of break heading text using the INSERT command. Each line of text can be up to 55 characters long. You can add text to the line by using APPEND as the ALIGN value, or by specifying a specific horizontal position.

If your installation supports DBCS data, see "Names with Double-byte Characters" on page 272.

By default, break heading text extends from the left to the right margin of a report. However, you can choose the width of break heading text on the `Report text line width` entry on FORM.OPTIONS (see page 229).

To make the break heading text appear in a report in uppercase and lowercase, specify in your profile a CASE value of either STRING or MIXED.

STRING
Displays break heading text as entered, but converts any other input to uppercase.

MIXED
Displays all input exactly as entered.

Break heading text can contain the following variables:

Global variables
Use SET GLOBAL to set variables for use in break heading text. See "SET GLOBAL" on page 113 for details about this command.

&*n* *n* is a number that represents the current row in column *n* on the form used for this report. Column *n* is not necessarily the *n*th column that you see in a report. It is the *n*th column listed on FORM.MAIN and FORM.COLUMNS. For example, this break heading text:

```
BEGINNING OF DEPARTMENT &3
```

might display this line on a report:

```
BEGINNING OF DEPARTMENT 38
```

The following variables can also be used with DATE, TIME, and TIMESTAMP values in break heading text:

&DATE
The current date is formatted according to the installation default, which reflects one of the following date formats:

- USA (United States of America)
- EUR (European)
- ISO (International Standards Organization)
- JIS (Japanese Industrial Standard)
- An alternative date format supplied by your installation

&TIME
The current time is formatted according to the installation default, which reflects one of the formats listed under &DATE.

&PAGE
The page number is printed on each page when the report is formatted.

If a page in a report is wider than either the printer width or the default printing width specified in your PROFILE, QMF splits the page. It gives all parts of the split page the same page number, but with subscripts. (If you are using DBCS data and QMF splits the page, printing resumes on the second and subsequent pages of the report at the fourth byte position from the left side of the page.)

&ROW
The number of the first data row within the current break level is printed or displayed in your report.

H New Page for Footing?
Specify whether to begin a new page (if the report is printed) before displaying any break footing text specified. A new page is started if the report is not already at the top of the page.

I Put Break Summary at Line
Specify whether the break summary is to be formatted, and, if so, where it is to be placed in relation to the lines of break footing text. The value for this entry can be any number from 1 through 999 or the word NONE (*no* break summary).

J Blank Lines Before Footing
Specify the number of blank lines before the first line of break footing. This entry can be any number from 0 through 999 or the word BOTTOM.

K Blank Lines After Footing
Specify the number of blank lines after the last line of the break footing text. The value for this entry can be any number from 0 through 999.

If you specify a break *and* you have a column-wrapped column with a usage code of FIRST, LAST, MIN, or MAX, you might need to increase the value in this field to see all the wrapped lines in the break summary. For information on column wrapping, see the CW entry in “Edit Codes for Character Data” on page 261.

L LINE
Identify the lines of break footing text and specify their position relative to themselves and to the line at which the break footing starts (as indicated in the *Blank Lines Before Footing* entry area). You can specify any number from 1 through 999 or a blank. A blank ignores any associated text.

The numbers you choose need not start with 1 or be consecutive.

For example, these values on FORM.BREAK1:

```
LINE  ALIGN  BREAK1 FOOTING TEXT
----  -----  ------------------
3     LEFT   DEPARTMENT &4
2     LEFT   END OF LISTING
```

Display as:

```
END OF LISTING
DEPARTMENT 35
```

M ALIGN

Specify where each line of the break footing text is to be placed horizontally in the report. For breaks without break summaries, you can place the lines of break footing text anywhere in the width of the report. The width of the report is shown at the top of FORM.MAIN.

For breaks with break summaries created with usage codes (except OMIT, BREAKn, GROUP, or ACROSS), QMF places the lines of break footing text anywhere from the left margin to the beginning of the indent area associated with the leftmost column of summary data.

Left Left-justifies the break footing text.

Right Right-justifies the break footing text.

Center
Centers the break footing text.

n Begins the break footing text in the *n*th position of the line. *n* can be any number from 1 through 999999.

Append
Positions the line at the end of the previous line of break footing text. If APPEND is used for a line of text that is not appended to another line, the line of text is left-justified.

The appended line of text must have the same LINE value as the line of text it is being appended to. If the report is not wide enough to accommodate the appended line of text, some of the text might be truncated.

For example, the following entries on FORM.BREAK1:

```
LINE  ALIGN   BREAK1 FOOTING TEXT
----  ------  ----------------------
1     RIGHT   TOTAL
1     APPEND   SALARIES--DEPT. &4;
3     RIGHT
4     RIGHT
5     RIGHT
```

align columns as shown in the resulting report.

```
 DEPT        COMM  JOB        SALARY
------  ----------  -----  ----------
   66       55.50  CLERK    10988.00
               -   MGR      18555.50
          844.00  SALES    16858.20
          200.30  SALES    21000.00
          811.50  SALES    18674.50
                             ----------
TOTAL SALARIES--DEPT. 66     86076.20

   84      188.00  CLERK    13030.50
               -   MGR      19818.00
          806.10  SALES    15454.50
         1285.00  SALES    17844.00
                             ----------
TOTAL SALARIES--DEPT. 84     66147.00
```

N BREAK1 FOOTING TEXT

Enter the footing text you want associated with the break. Every time the value in the break column changes, the text specified in this entry is displayed in the report. You can add up to 999 lines of break footing text using the INSERT command. Each line of text can be up to 55 characters long. You can add text to the line by using APPEND as the ALIGN value, or by specifying a specific horizontal position.

If your installation supports DBCS data, see "Names with Double-byte Characters" on page 272.

By default, break footing text extends from the left margin of a report either to the beginning of the break summary data (if any), or to the right margin of a report. However, you can choose the width of break footing text on the `Report text line width` entry on FORM.OPTIONS (see page 229).

To make the break footing text appear in a report in uppercase and lowercase, specify in your profile a CASE value of either STRING or MIXED.

STRING
: Displays break footing text as entered, but converts any other input to uppercase.

MIXED
: Displays all input exactly as entered.

Break footing text can contain the following variables:

Global variables
: Use SET GLOBAL to set variables for use in break footing text. See "SET GLOBAL" on page 113 for details.

&*n* *n* is a number that stands for the most current value in column *n* on the form used for this report. Column *n* is not necessarily the *n*th column that you see in a report. It is the *n*th column selected from the database, or the *n*th column listed on FORM.MAIN and FORM.COLUMNS.

For example, this break footing text:

```
END OF DEPARTMENT &3
```

Might display this line on a report:

```
END OF DEPARTMENT 38
```

&COUNT
The number of rows retrieved or printed since the last break at the same level. This value increases from data row to data row.

&ROW
The number of the last data row is printed or displayed in your report.

&CALC*id*
Calculated value

&DATE
The current date

&TIME
The current time

&PAGE
The current page number

For a description of &CALC*id*, see "FORM.CALC" on page 198.

For descriptions of &DATE, &TIME, and &PAGE, see page 193 under *BREAK1 HEADING TEXT*.

&*an* *n* is a valid column number and *a* is one of the following QMF aggregation functions: AVG, COUNT, CPCT, CSUM, FIRST, LAST, MAX, MIN, PCT, STDEV, SUM, TCPCT, TPCT. The values of the aggregations are based on running values within the current break level.

For example, assume the fourth column of the report contains salaries and you want to summarize the salaries in each group in break footing text.

Write in the BREAK1 FOOTING TEXT:

```
TOTAL SALARY FOR DEPARTMENT &3 IS &SUM4
```

For example, the resulting line of break footing text in the report would be:

```
TOTAL SALARY FOR DEPARTMENT 38 IS $77,285.55
```

If you specify the aggregation variable in break footing text, you need not specify that same aggregation as the usage for that column. However, the aggregation must be compatible with the edit code and data type of the column. For example, you cannot specify &SUM3 in your final text if the data in column 3 has a character edit code.

If you use an aggregation variable with percent (PCT, TPCT, or TCPCT) in break footing text, and if you associate it with a column that has a D edit code, QMF formats the percent value as if it had an L edit code. Likewise, if you use the aggregation variable standard deviation and associate it with a column that has a P or a D edit code, QMF formats the standard deviation as if it had an L edit code.

For more information, see the L code under “Edit Codes for Numeric Data” on page 263 and “Variables Used in Forms” on page 268.

FORM.CALC

Note to CICS users

FORM.CALC uses expressions written in REXX, which is not available in CICS.

On the FORM.CALC panel you can enter expressions for report calculations. It initially contains only one row—a place for one expression. However, up to 998 additional rows can be inserted.

Each entry area is described in terms of its effect on reports. FORM.CALC does not affect charts.

```
FORM.CALC

                                                        C       D      E
 A              B                                       PASS    For &CALCid
 ID   CALCULATION EXPRESSION                            NULLS?  WIDTH  EDIT
 ---  ------------------------------------------------  ------  -----  -----
                                                        NO      10     C

      *** END ***

 1=Help        2=Check     3=End     4=Show       5=Chart       6=Query
 7=Backward    8=Forward   9=       10=Insert    11=Delete     12=Report
 OK, Cursor positioned.
 COMMAND ===>                                              SCROLL ===> PAGE
```

A ID

Enter a one to three character identifier for the corresponding calculation expression. The identifier is any number from 1 through 999. When appended to the usage code CALC*id* (see "Usage Codes" on page 251) or the &CALC variable (&CALC*id*), it identifies which expression on FORM.CALC is to be used in a calculation.

The &CALC*id* variable can be used only in detail block text, final text, and break footing text. CALC*id* and &CALC*id* activate the evaluation of the calculation expression on FORM.CALC whose ID equals *id*.

For an &CALC variable, the evaluated result is edited according to the width and edit code specified for the expression in the FORM.CALC panel (subject to the special factors described in "Summary of Editing Expressions" on page 202). For a CALC*id* usage code, the evaluated result is edited according to the width of the columns and the edit code of the CALC.

B CALCULATION EXPRESSION

Enter an expression. It can contain up to 50 characters. You cannot execute QMF commands (using the callable or command interfaces) from within a REXX EXEC used in FORM.CALC.

Other than &CALC*id*, any valid form variable can be used in the expressions. The following variables are valid:

Global variables

Use SET GLOBAL to set variables for use in calculation expressions. See "SET GLOBAL" on page 113 for details about this command.

Column variables: &*n*

n is a column number.

Aggregation variables: &*an*

n is a valid column number, and *a* is one of the following QMF aggregation functions: AVG, COUNT, CPCT, CSUM, FIRST, LAST, MAX, MIN, PCT, STDEV, SUM, TCPCT, TPCT.

&ROW
Print the number of the data row at the time the calculation is evaluated. The &ROW variable is replaced just before the &CALC*id* variable or CALC usage code is evaluated.

&COUNT
Row count

&DATE
The current date

&TIME
The current time

&PAGE
The current page (always 1 for displayed reports)

For a description of &COUNT, see page 197 under *BREAK1 FOOTING TEXT*.

For descriptions of &DATE, &TIME, and &PAGE, see page 193 under *BREAK1 HEADING TEXT*.

When an expression is entered, its variables are validated. Column variables are checked for valid column numbers and for compatible usages or edit codes or both. For example, if the sixth column has an edit code of C and the expression uses &SUM6, an error exists and a message is issued.

Be sure to use substitution variables that are compatible with the expression. QMF does not check for nonnumeric substitution variables in an arithmetic expression.

If you encounter a syntax error on the expression, you must correct it either in the REXX EXEC itself or in the REXX expression. Be sure to follow the REXX coding rules.

For example, you include in the expression an EXEC name that does not exist. After you correct the EXEC name or create the EXEC, show F.CALC and make any necessary modifications. If you don't need to make any other changes, retype one of the characters in the expression. Doing this causes QMF to validate the variables again to ensure you have built your form correctly. If you don't revalidate your form, you might get unpredictable results.

C PASS NULLS
Enter YES or NO.

YES Allows you to use the following QMF-provided values to change the default handling in the corresponding situations:

Value Situation

DSQNULL
Data is null

DSQUNDEF
Data is undefined

DSQOFLOW
Data has numeric overflow

DSQNOINS
Data has no instance

DSQNOREL
Data has no relationship

For example, any database variable that is null (a database null) is replaced with the character string DSQNULL before the expression is passed to REXX for evaluation. You can provide a REXX expression or EXEC that checks for the string and substitutes 0 (or whatever is appropriate for your purpose) for the database null.

If a null value is returned by the REXX expression, you can pass it to your report.

NO Returns a null for the values listed above. Nothing is passed to REXX for evaluation.

D WIDTH

Enter the width (in single-byte characters) to which the evaluated result of the corresponding expression is edited in report text. It is applicable only to results obtained for &CALC*id* variables. If the CALC*id* usage cannot be edited according to the edit code for the column, the edit code of the CALC*id* is used.

WIDTH is a 5-character entry field. It must contain a number from 1 through 32767. The default is 10.

E EDIT

Enter the edit code to be used when the evaluated result of the corresponding expression is edited in report text. It is applicable only to results obtained for &CALC*id* variables. Results of CALC*id* usages are edited using the edit code specified for the column on FORM.MAIN or FORM.COLUMNS.

EDIT is a 5-character field. The default is C for character data when a line is inserted in FORM.COLUMNS. Only the following edit codes are accepted:

Numeric

D E I J K L P

You can use optional suffixes with these numeric edit codes. Z is an optional suffix for all numeric edit codes and can be used to suppress zero values. C is an optional suffix for the **D** edit code and causes QMF to use a currency symbol specified with the global variable DSQDC_CURRENCY instead of the default currency symbol. You can add a decimal scale value from 0 to 99 to any numeric edit code except E.

Character

C Character editing (default)

User-defined

U*xxxx*, **V***xxxx*

User edit codes for numeric or character editing.

Summary of Editing Expressions

Table 9, following, summarizes the results returned when an edit code is applied to an expression. For details on edit codes for calculations, see "Edit Codes" on page 260.

Table 9. Edit Code Summary

Result from User Expression	Applicable Edit Code	Edited Result	
Numeric	Numeric	Edited according to edit code	
	Nonnumeric	Character representation of result edited according to edit code	
	U*xxxx*, V*xxxx*	As edited by user edit routine (expression result for U*xxxx* is passed to routine as extended floating point data)	
Nonnumeric	Numeric	As if C (character)	
	Nonnumeric	C*xx*	Character
		U*xxxx*, V*xxxx*	As edited by user edit routing

Note: In COBOL, a long floating point format for the first eight bytes of numeric data should provide suffiecient accuracy. If not, use the V*xxxx* edit code for maximum accuracy.

FORM.COLUMNS

Use FORM.COLUMNS to make choices about the uses of the columns. What you specify on FORM.COLUMNS is reflected on FORM.MAIN. Conversely, what you specify on FORM.MAIN (areas **A** through **F**) is reflected on FORM.COLUMNS.

```
FORM.COLUMNS

COLUMNS:          Total Width of Report Columns: 66
            A                              B       C       D       E       F
 NUM    COLUMN HEADING                   USAGE   INDENT  WIDTH   EDIT    SEQ
 ---    -------------------------------  ------  ------  -----   -----   ---
   1    ID                                       2       6       L       1
   2    NAME                                     2       9       C       2
   3    DEPT                                     2       6       L       3
   4    JOB                                      2       5       C       4
   5    YEARS                                    2       6       L       5
   6    SALARY                                   2       10      L2      6
   7    COMM                                     2       10      L2      7
   8    Total Earnings                           2       12      L2      8
        *** END ***

 1=Help        2=Check     3=End       4=Show        5=Chart     6=Query
 7=Backward    8=Forward   9=Specify  10=Insert     11=Delete   12=Report
 OK, FORM.COLUMNS is displayed.
 COMMAND ===>                                          SCROLL ===> PAGE
```

A COLUMN HEADING

Reports: Assign column headings. On the default form, column headings can be any of the following:

- The label assigned to the column (if your installation uses labels)
- The name of the column in the table from which it was selected
- A heading constructed by QMF for columns of constants or calculated values

If your installation supports DBCS data, see “Names with Double-byte Characters” on page 272.

You can enter any new heading of up to 40 characters over a heading shown in the COLUMN HEADING area. The heading, like the original column name, can contain blanks or special characters; of these, the underscore character (_) is reserved for multiple-line headings.

To create multiple-line headings, use an underscore in a column heading to specify a break between lines. For example:

```
EMPLOYEE_NAME    displays as:    EMPLOYEE
                                 NAME
```

A single underscore before or after an entire column heading has no effect. For example, _EMPLOYEE NAME does not add a blank line. However, consecutive underscores within text produce one or more blank lines in a column title. You can have up to nine lines in a column heading.

For example, these two column names:

```
1  ONE_TWO_THREE_FOUR_FIVE_SIX_SEVEN
2  SIX_ _LINE_ _ _TITLE
```

Display as:

```
 ONE          SIX
 TWO
THREE        LINE
FOUR
FIVE
 SIX         TITLE
SEVEN
```

If you are using double-byte characters in column headings, you can specify a break between lines if the underscore you use is a single-byte character.

To create column headings in uppercase and lowercase, specify in your PROFILE a CASE value of either STRING or MIXED.

STRING
Displays column heading text as entered, but converts any other input to uppercase.

MIXED
Displays all input exactly as entered.

Headings are aligned (justified) to the left over a column of character data, and to the right over a column of numeric data. If there is more than one line in the heading, the longest line is justified, and shorter lines are centered within the longest line. You can override these defaults by entering a new alignment value. See “Column Alignment” on page 210 for more information.

If any line of a heading is longer than the width of the column, it fills the whole width of the column and is cut off on the right.

Global variable substitution is not performed for column headings.

Charts: Most of the preceding information on how changes to COLUMN HEADING affect reports is also true for charts. Column headings for data plotted on the Y-axis appear in the legend of a

chart. Therefore, you probably want these column headings to be as concise as possible, or the legend will take up too much space on the chart.

B USAGE

Reports: Specify how you want a column processed for a report. If the usage code for a column is blank, the values in the column are listed with no other processing unless one or more columns in the report has a usage of GROUP and at least one column has an aggregation usage. In that case, blank columns are omitted. A number of aggregation functions, listed in Table 10, can be entered in the area.

Table 10. Aggregation Functions

Aggregation	Usage Code	Minimum Abbreviation	Page
Across	ACROSS	AC	251
Average	AVERAGE (or AVG)	AV	253
Break1	BREAK, BREAK1	B, B1	189
Break1x	BREAKX, BREAK1X	BX, B1X	189
Break2	BREAK2	B2	189
Break2x	BREAK2X	B2X	189
Break3	BREAK3	B3	189
Break3x	BREAK3X	B3X	189
Break4	BREAK4	B4	189
Break4x	BREAK4X	B4X	189
Break5	BREAK5	B5	189
Break5x	BREAK5X	B5X	189
Break6	BREAK6	B6	189
Break6x	BREAK6X	B6X	189
Calculate	CALC*id*	CA	198
Count	COUNT	CO	253
Cumulative percent	CPCT	CP	254
Cumulative sum	CSUM	CS	254
First	FIRST	F	253
Group	GROUP	G	259
Last	LAST	L	253
Maximum	MAXIMUM	MA	253
Minimum	MINIMUM	MI	253

Table 10. Aggregation Functions (continued)

Aggregation	Usage Code	Minimum Abbreviation	Page
Omit	OMIT	O	260
Percent	PCT	P	254
Standard deviation	STDEV	ST	253
Sum	SUM	SU	253
Total cumulative percent	TCPCT	TC	254
Total percent	TPCT	TP	254

C INDENT

Reports: Specify the number of blank spaces to the left of a column. The blank spaces separate the column from the previous column or from the left margin. INDENT can be any number from 0 through 999. For columns using a graphic edit code, the minimum indent is 1. The default INDENT for each column is 2.

INDENT is always specified in single-byte characters.

D WIDTH

Reports: Specify the number of character positions reserved for displaying data from a column, or the column heading. WIDTH can be any number from 1 through 32767.

If the column you are displaying uses a graphic edit code, the width can be any number from 1 through 16383. For more information about how to calculate the width of a column containing DBCS data, see *Using QMF*.

For a column that uses a graphic edit code, the width of the column, when displayed or printed, is twice the column width, plus one character space.

When assigning a width for numeric data, include space for the following characters as well as for digits:

- A minus sign (except with edit code J)
- A decimal point (when edit codes specify them)
- Separators for groups of thousands (with edit codes D, K, and P)
- A currency symbol (with edit code D)
- A percent sign (with edit code P)

If the length of a value to be displayed exceeds the width of the column:

- If it is numeric data, it is replaced with a row of asterisks (********)

  In some cases, you can avoid a numeric overflow by using a different data type. For example, in an arithmetic operation, if all operands are decimal numbers and an overflow occurs, you can change at least one operand to a floating point number. In this example, the operand can be a floating point constant or a floating point table column.
- If it is character, date, time, or timestamp data, it is cut off at the right or left (depending on the alignment specified for the data)

Resolve column width problems by changing WIDTH and displaying the report again. Alternatively, you can tell QMF to keep the column width the same, but to wrap data that won't fit on a line to the next line in the column. Column wrapping applies only to nonnumeric data. For more information about column wrapping, see "Edit Codes" on page 260.

The width of a column on the default form is at least as great as the longest line in the column heading. Otherwise, the assigned width depends on the data type of the column, as shown in Table 11.

Table 11. Default Width of Data Types

Data Type	Width on Default Form
SMALLINT	6
INTEGER	11
DECIMAL	The width of the column in the database, plus 3 character spaces.
FLOAT	10
CHAR	The width of the column in the database.
VARCHAR	The maximum width of the column in the database.
LONG VARCHAR	The smaller of: • The column width • A width determined by QMF, based on the quantity and type of other columns in the report
GRAPHIC	The width of the column in the database.
VARGRAPHIC	The width of the column in the database.
LONG VARGRAPHIC	The smaller of: • The column width. • A width determined by QMF, based on the quantity and type of other columns in the report.

Table 11. Default Width of Data Types (continued)

Data Type	Width on Default Form
DATE	10, or if your date format is locally defined by your installation, the larger of: • The width of the column heading • The width of the locally defined date format
TIME	8, or if your time format is locally defined by your installation, the larger of: • The width of the column heading • The width of the locally defined time format
TIMESTAMP	26

When inserting a line on FORM.COLUMNS, the default width is 10.

For single-precision floating point data, values with a data type of FLOAT are treated the same for single-precision or double-precision.

Charts: Specify the number of character positions for labels on the X-axis of a chart.

If the width exceeds the allotted space, the labels might be omitted. Truncating the width of column headings is one way to handle the problem of omitted labels. When labels are truncated, more fit in the allotted space.

Single-precision floating point data is treated the same as double-precision floating point data for chart formatting.

Values from columns with DATE, TIME, and TIMESTAMP data types, (treated as character strings) cannot appear on the Y-axis.

E EDIT

Reports: Specify how QMF formats data for display. The default is C when inserting a line in FORM.COLUMNS.

Charts: The X-axis labels come from columns using GROUP or BREAK (or from the leftmost column of the report when there is no GROUP or BREAK). The effect that edit codes have on the data in those columns appears in the X-axis labels. For example, if data selected for the X-axis is column wrapped, only the first line is incorporated into the labels.

Also, numeric columns that are edited with U*xxxx* or V*xxxx* cannot be used for Y data.

Finally, when column substitution values (*&n*) are used in the page heading (and therefore, in the chart heading), they are edited according to the edit code for that column in the form.

Table 12 lists the edit codes that can be specified for each data type and the page that contains more information.

Table 12. Edit Codes for Data Types

Data Type	Edit Codes	Page
Character	C CDx CW CT X B XW BW U*xxxx* V*xxxx*	261
Graphic	G U*xxxx* GW V*xxxx*	263
Numeric	E<Z> D<Z><C> I<Z> J<Z> K<Z> L<Z> P<Z> U*xxxx* V*xxxx*	263
DATE	TDY*x* C TDM*x* CW TDD*x* CT TDYA*x* CD*x* TDMA*x* U*xxxx* TDDA*x* V*xxxx* TDL	264
TIME	TTSx C TTCx CW TTAx CT TTAN CDx TTUx U*xxxx* TTL V*xxxx*	265
TIMESTAMP	TSI CDx C U*xxxx* CW V*xxxx* CT	266

You can use character edit codes with DATE, TIME, and TIMESTAMP columns to allow wrapping of those columns.

F SEQ

Reports: Enter numbers in this column to change the sequence of the columns in your report. Initial settings are the same as for the NUM column. Any numbers from 1 through 999 are allowed. If two

numbers are the same, those columns appear in the same order they are listed on the form. The Automatic reordering of report columns option on the FORM.OPTIONS panel must be set to NO (the default) for SEQ to have an effect on column reordering.

When variables are resolved, the column number is taken from NUM, not SEQ.

SEQ numbers are ignored in ACROSS reports.

Specifying Column Attributes

Using the SPECIFY command, you can change the alignment of a column heading or the data within a column, or you can define a column. There are two ways to access the alignment and definition panels.

- Press the Specify function key to display the Specify panel, then choose Alignment or Definition.
- Enter SPECIFY alignment or SPECIFY definition (or a valid abbreviation) on the command line, then move the cursor to the desired column and press Enter. This bypasses the Specify panel and takes you directly to the Alignment or Definition window.

Column Alignment

If you specify alignment, a small panel overlays the FORM.COLUMNS panel showing the alignment specifications for the column you chose. For example:

```
                        Alignment
Column number   :    3
Column Heading  :  DEPT_HEADING_CAN_BE UP TO_40 CHARS LONG!

Heading alignment  :  [DEFAULT  ]
Data alignment     :  [LEFT     ]

_____________________________________________________________
F1=Help  F5=Previous Column  F6=Next Column  F12=Cancel
```

Choices for heading and data alignment are LEFT, RIGHT, CENTER, and DEFAULT. The default for the heading and data of a column containing character data is right-justified, while the default for the heading and data of a column containing numeric data is left-justified.

To change an alignment value, type the new value over the current value. Use the tab key to move between the heading and data alignment entry fields. from one column alignment specification to another.

Column alignment applies mainly to tabular data. However, if you use **_B** with a substitution variable, the data is aligned as follows:

1. The data is edited according to the edit code and width of the column.

2. If the alignment is not DEFAULT, leading and trailing blanks are removed.
3. The value is aligned according to the specified alignment value.
   - If the data is character, trailing blanks are removed.
   - If the data is numeric, leading blanks are removed.
   - If &_B is used, no blanks are removed.

In tabular reports, leading and trailing blanks are removed if the value for data alignment is LEFT, RIGHT, or CENTER. The blanks are not removed if the data alignment value is DEFAULT.

If you are using edited character data with leading blanks, or edited numeric data with trailing blanks, the blanks are not removed regardless of the alignment value.

Column Definition

Note to CICS users
Column definition is not available in CICS, because its function depends on REXX.

Column definition allows you to define a new column of data using an expression. There are some differences between columns retrieved by a query and columns you define. The main difference is in the data type and length assigned to user-defined columns.

When you define a column, you are prompted to enter an expression to define the column and whether null values should be included when REXX evaluates the expression. QMF determines the data type and column length based on the edit code and column width specified for that column on FORM.COLUMNS. However, if you use a usage code for the defined column that does not agree with the edit code for the column, the usage code determines the data type.

Another difference between user-defined columns and those retrieved from the database is that values for user-defined columns are not retained when the data is saved or exported.

Column wrapping can also appear to work differently for defined columns.

- If the data for a defined column is less that 254 bytes, there is no apparent difference in how column wrapping works.
- If the data for a defined column is greater than 254 bytes and the column width is 254 or less, the data is wrapped up to and including the 254th byte, but the remainder of the data is truncated.

- If the data for a defined column is greater that 254 bytes and the column width is 255 or more, the data is wrapped at the width of the column.

A LONG VARCHAR column can only have a usage code of OMIT (or be left blank).

When you specify Definition from FORM.COLUMNS, a panel is displayed where you can enter an expression (up to 50 characters) defining your new column. For example:

```
                        Definition

Column number :     8
Column Heading:   Total Earnings

Type an expression to define this column.
Expression  [ totearn(&6 &7)                                ]
Pass Nulls? [ YES    ]
_____________________________________________________________
F1=Help  F5=Previous Column  F6=Next Column
F10=Previous Definition  F11=Next Definition  F12=Cancel
```

You can define the new column in terms of:

- A character or numeric constant
- The following form variables (see page 193 under *BREAK1 HEADING TEXT* for general descriptions of QMF form variables):
  - &n
  - &DATE
  - &TIME
  - &ROW
  - Any global variable conforming to “Rules for Variable Names and Values” on page 114
- A valid REXX expression or function
- An expression involving any of the above

If you include a REXX expression in your column definition, you might receive unexpected results if the value returned by REXX is longer than 32767 characters.

Use the Previous and Next function keys to move from one column definition panel to another.

Pass Nulls: If the PASS NULLS question is answered YES, you can use the following QMF-provided values to change the default handling in the corresponding situations:

Value Situation

DSQNULL
Data is null

DSQUNDEF
Data is undefined

DSQOFLOW
Data has numeric overflow

DSQNOINS
Data has no instance

DSQNOREL
Data has no relationship

For example, any database variable that is null (a database null) is replaced with the character string DSQNULL before the expression is passed to REXX for evaluation. You can provide a REXX expression or EXEC that checks for the string and substitutes 0 (or whatever is appropriate for your purpose) for the database null.

If a null value is returned by the REXX expression, you can pass it to your report.

If the PASS NULLS answer is NO, a null is returned for the values listed above. Nothing is passed to REXX for evaluation.

Edit Codes and Data Types

QMF determines the data type and column length of a defined column based on the edit code and column width specified for that column on the FORM.COLUMNS panel. Table 13, following, summarizes the results.

Table 13. Edit Codes and Data Types

Edit Code	Data Type	
	Column width <=254	**Column width >=255**
Character (C, CW, CT, CDx, B, BW, X, XW)	VARCHAR (max. length=254	LONG VARCHAR (max. length=32767)
Numeric (D, E, I, J, K, L, P)	Numeric - Extended floating point	
U and V user edit codes (no numeric usage)	VARCHAR (max. length=254)	LONG VARCHAR (max. length=32767
U and V user edit codes (at least one numeric usage)	Numeric - Extended floating point	

SQL/DS and DB2 databases do not support an extended floating point data type. Therefore, you might find it advantageous to define a numeric column as extended floating point, for example, when working with data that would ordinarily cause an overflow condition if it were used as a database data type (such as DECIMAL or INTEGER).

Printing Considerations

When you print a FORM, the column definition and alignment information are printed on a page following the FORM.COLUMNS instead of the `Specify Alignment` and `Specify Definition` windows that appear on your screen. The NUM field is repeated with the column definition and alignments. For example:

```
1                                      FORM:
FORM.COLUMNS

        HEADING   DATA                                          PASS
 NUM    ALIGN     ALIGN      DEFINITION                         NULLS?
 ---    --------  ---------  ---------------------------------  ------
   1    DEFAULT   DEFAULT                                       NO
   2    CENTER    CENTER                                        NO
   3    DEFAULT   DEFAULT                                       NO
   4    LEFT      DEFAULT                                       NO
   5    DEFAULT   DEFAULT                                       NO
   6    DEFAULT   DEFAULT                                       NO
   7    DEFAULT   DEFAULT                                       NO
   8    RIGHT     RIGHT       &6 + &7                           NO
   9    DEFAULT   DEFAULT     (&6 + &7)  *  &5                  NO
        *** END ***

   05/05/91   11:10 AM                                PAGE   3
```

FORM.CONDITIONS

> **Note to CICS users**
> FORM.CONDITIONS uses expressions written in REXX, which is not supported in CICS.

Use FORM.CONDITIONS to enter expressions for conditional formatting. Conditional formatting allows you to create expressions that determine when the formatting variations specified in FORM.DETAIL appear.

You can use conditional formatting to specify detail text for grouped data. The condition is evaluated using data from the first row of the group. If the condition evaluates to `true`, the detail text for that variation is printed. If the condition evaluates to `false`, the detail text for that variation is not printed for that group.

```
FORM.CONDITIONS

 A          B                                                    C
 ID   CONDITIONAL EXPRESSION                                     PASS
 ---  ---------------------------------------------------------  NULLS?
                                                                 ------
                                                                 NO
          *** END ***

 1=Help        2=Check     3=End     4=Show        5=Chart       6=Query
 7=Backward    8=Forward   9=       10=Insert     11=Delete     12=Report
 OK, FORM.CONDITIONS is displayed.
 COMMAND ===>                                          SCROLL ===> PAGE
```

A ID

Enter a one to three character identifier for the corresponding conditional expression. The identifier is any number from 1 through 999. When appended to the *C* selection code in the **N** `Select Panel Variation?` of the FORM.DETAIL panel (page 223), it identifies which expression in FORM.CONDITIONS determines whether the detail variation gets formatted.

B CONDITIONAL EXPRESSION

Enter a valid REXX expression. The difference between an expression in FORM.CALC and in FORM.CONDITIONS is that a condition results in a value of either `true` or `false`. An expression evaluating to 1 is true; an expression evaluating to anything else is assumed to be false. Nonnumeric data, including blanks and nulls, are assumed to be false. You can use any valid global variables in conditional expressions. However, the only QMF form variables you can use in in conditional expressions are &ROW, &DATE, &TIME, and &n

For more information, see “Using REXX with QMF Forms” on page 244.

C PASS NULLS

Enter YES or NO.

YES Allows you to use the following QMF-provided values to change the default handling in the corresponding situations:

Value	Situation
DSQNULL	Data is null
DSQUNDEF	Data is undefined
DSQOFLOW	Data has numeric overflow
DSQNOINS	Data has no instance

DSQNOREL
Data has no relationship

For example, any database variable that is null (a database null) is replaced with the character string DSQNULL before the expression is passed to REXX for evaluation. You can provide a REXX expression or EXEC that checks for the string and substitutes 0 (or whatever is appropriate for your purpose) for the database null.

If a null value is returned by the REXX expression, you can pass it to your report.

NO Returns a null for the values listed above. Nothing is passed to REXX for evaluation.

FORM.DETAIL

Use FORM.DETAIL to:

- Specify text to precede column headings.
- Combine tabular data with text.
- Omit tabular data and show data values entirely as text.

FORM.DETAIL consists of *detail variations* that you define. You can create up to 99 variations, and each variation can correspond to conditions entered on FORM.CONDITIONS. Unless each condition is mutually exclusive, different detail variations can be displayed for the same data row.

FORM.DETAIL does not affect charts.

```
FORM.DETAIL                                                  A  VAR   1 of  1

 B Include Column Headings with Detail Heading? ===> YES
 C LINE D ALIGN E DETAIL HEADING TEXT
 ----    ------   ----+----1----+----2----+----3----+----4----+----5----+
 1       LEFT
 2       LEFT
         *** END ***

 F New Page for Detail Block? ===> NO    G Repeat Detail Heading?   ===> NO
 H Keep Block on Page?        ===> NO    I Blank Lines After Block  ===> 0
 J Put Tabular Data at Line (Enter 1-999 or NONE) ===> 1
 K LINE L ALIGN M DETAIL BLOCK TEXT
 ----    ------   ----+----1----+----2----+----3----+----4----+----5----+
 1       LEFT
 2       LEFT

         *** END ***
 N Select Panel Variation? ===> YES

 1=Help        2=Check     3=End     4=Show        5=Chart       6=Query
 7=Backward    8=Forward   9=        10=Insert     11=Delete     12=Report
 OK, FORM.DETAIL is displayed.
 COMMAND ===>                                            SCROLL ===> PAGE
```

A VAR 1 of 1

The first number represents the current panel variation, and the second represents the total number of variation panels (maximum is 99). The default form displays `VAR 1 of 1`.

You can create a new detail variation by entering a value one greater than the total number of variation panels over the current panel variation value. New panels must be added sequentially.

You can navigate to existing panel variations by entering the identifying value over the current panel variation value. You can also display different panel variations by entering the NEXT and PREVIOUS commands on the command line. (See "NEXT" on page 85 and "PREVIOUS" on page 86 for more information.)

Sections **B** through **E** specify text to be followed in a report by column headings specified on FORM.COLUMNS.

B Include Column Headings with Detail Heading?

YES Column headings become part of the detail headings. The resulting detail heading is repeated whenever requested on BREAK panels or in **G** *Repeat Detail Heading?* (page 220).

NO Column headings are suppressed.

C LINE

Identify lines of detail heading text and their relative positions. Any number of lines can be specified. The line numbers can be any number from 1 through 999 or blank.

If you use the same LINE value for more than one line, those lines are joined according to the ALIGN value for the additional line or lines. Lines with the same LINE value overlay each other if they are longer than the report width, or if their ALIGN values conflict.

D ALIGN

Specify where each line of detail heading text is to be placed horizontally in the report. You can place the lines anywhere within the width of the report.

Left Left-justifies the detail heading text.

Right Right-justifies the detail heading text.

Center
Centers the detail heading text.

n Begins the detail heading text in the *n*th position of the line. *n* can be any number from 1 through 999999.

Append
If APPEND is used for a line of text that is not appended to another line, the line of text is left-justified.

The previous line of text and the appended line of text must have the same LINE value if they are to be placed on the same line. If the report is not wide enough to accommodate the appended line of text, some of the text might be truncated.

E DETAIL HEADING TEXT

Specify the detail heading text. You can add up to 999 lines of text using the INSERT command. Each line of text can be up to 55 characters long. You can add text to the line by using APPEND as the ALIGN value, or by specifying a specific horizontal position.

If your installation supports DBCS data, see "Names with Double-byte Characters" on page 272.

Detail heading text always precedes column headings in a report. Detail headings consist of detail heading text, column headings, or both. Unless omitted, detail heading text and column headings constitute detail headings.

By default, a detail heading can extend from the left margin to the right margin of the report. Any text that extends beyond the right margin is not displayed or printed. You can alter the width by changing the report text width on the FORM.OPTIONS panel. If you do not explicitly specify a width, the right margin is determined by the width of the tabular data.

When printing a report, all the detail headings selected for the current row of data when the page heading is formatted are printed. If the

number of lines for the detail heading exceed the number of available lines on the page, the excess detail heading lines are lost.

Detail headings can contain the following variable values:

Global variables
Use SET GLOBAL to set variables for use in detail heading text. See "SET GLOBAL" on page 113 for details about this command.

&*n* The value in the *n*th column on the form used for this report. For example, this detail heading:

```
ID NUMBER:  &1   EMPLOYEE NAME:  &2
```

Can produce the following heading in a report:

```
ID NUMBER:  50    EMPLOYEE NAME:  HANES
```

The &*n* value is the value of column *n* from the current row at the start of the new page. Detail headings for unconditionally selected variations are shown at the top of each screen in displayed reports. However, the value for &*n* appears only on the first screen of a displayed report. If you want to display the report online with page breaks, issue the DPRE command. See "DPRE" on page 25 for more information on this command.

With this special syntax, the width of the substitution value is determined by the width specified by the associated column on the FORM.COLUMNS or FORM.MAIN panel.

&ROW
The number of the current data row when the detail heading is formatted.

&DATE
The date the print command was executed (in printed reports) or the current date (in displayed reports)

&TIME
The time the print command was executed (in printed reports) or the current time (in displayed reports)

&PAGE
The current page number

For descriptions of &DATE, &TIME, and &PAGE, see page 193 under *BREAK1 HEADING TEXT*.

Sections **F** through **M** specify report data that can be repeated in a report for each data row. This data, called a detail block, is the tabular data (if selected) and text associated with a single data line or a single detail line (for example, a row from a table).

F New Page for Detail Block?
Specify whether to start each occurrence of the detail block on a new page in a printed report. A new page is started if the report is not already at the top of the page.

G Repeat Detail Heading?
Specify whether to repeat the detail heading before each occurrence of the detail block text. The detail heading includes any detail heading text specified on the FORM.DETAIL panel, followed by column headings (if not suppressed) listed on the FORM.COLUMNS panel.

NO The detail heading is formatted at the beginning of each screen for online reports or each page for printed reports.

YES The detail heading is formatted before each occurrence of detail block text.

H Keep Block on Page?
Specify whether to keep each detail block text together on one page of your printed report.

NO Detail blocks can be split across two or more pages of your printed report.

YES You can prevent detail blocks from being split across pages. If a detail block is too long to be printed on one page, it is started on a new page.

I Blank Lines After Block
Specify how many blank lines after detail block text.

The detail spacing option on the FORM.OPTIONS panel also affects the number of blank lines after detail block text.

J Put Tabular Data at Line (Enter 1-999 or NONE)
Specify whether to generate the tabular data (in the tabular format specified on FORM.COLUMNS or FORM.MAIN) and where this tabular data should be placed. The number corresponds to the number of the detail block text line on which the tabular data should be placed. NONE (or N) indicates not to format the tabular data. NONE doesn't affect break text or aggregation values.

This option can be used to mix text with tabular data. When a number is specified, tabular data overlays or combines with any detail block text on the same line.

If NONE is specified, tabular data is not formatted, but the column values can be included in the detail block text by using column substitution values.

K LINE

Identify the lines of detail block text and specify their relative positions. Any number of tabular data lines can be specified. You can specify any number from 1 through 999 or a blank. See **C** *LINE* on page 217 for additional information.

L ALIGN

Specify where each line of detail block text is to be placed horizontally in the report. You can place the lines anywhere within the width of the report. Valid values are LEFT, RIGHT, CENTER, APPEND, or any number from 1 through 999999.

The ALIGN values do not affect the horizontal placement of tabular data. To change the placement of tabular data, modify the column widths or indents on FORM.COLUMNS or FORM.MAIN. See **D** *ALIGN* on page 218 for additional information.

M DETAIL BLOCK TEXT

Specify the detail block text. You can add up to 999 lines of detail block text using the INSERT command. Each line of text can be up to 55 characters long. You can add text to the line by using APPEND as the ALIGN value, or by specifying a specific horizontal position.

If your installation supports DBCS data, see "Names with Double-byte Characters" on page 272.

By default, detail block text extends from the left margin to the right margin of the report. Any text that extends beyond the right margin is not displayed or printed. You can alter the width by changing the report text width on the FORM.OPTIONS panel. If you do not specify a width, the right margin is determined by the width of the tabular data.

Detail block text can contain literal text along with the following variable values:

Global variables

Use SET GLOBAL to set variables for use in detail block text. See "SET GLOBAL" on page 113 for details about this command.

&*n* The value in the *n*th column on the form used for this report. For example, this detail block text:

```
DEPARTMENT: &3   EMPLOYEE NAME: &2
```

Could produce the following line in a report:

```
DEPARTMENT:  20    EMPLOYEE NAME:  SANDERS
```

&COUNT
The number of rows displayed or printed since the last break. This value is a running count and increases from data row to data row.

&ROW
The number of the data row for the detail block is printed or displayed in your report.

In detail block text with a group summary report, the number of the data row for the last row in the group is printed.

&CALC*id*
Calculated value

&DATE
The current date

&TIME
The current time

&PAGE
The current page number

For a description of &CALC*id*, see "FORM.CALC" on page 198.

For descriptions of &DATE, &TIME, and &PAGE, see page 193 under *BREAK1 HEADING TEXT*.

&*an* *n* is a valid column number, and *a* is one of the following QMF aggregation functions: AVG, COUNT, CPCT, CSUM, FIRST, LAST, MAX, MIN, PCT, STDEV, SUM, TCPCT, TPCT. The values of the aggregations are based on running values within the current break level.

In detail block text, the values for aggregations are based on the data values since the last break through the current row. Calculated values such as AVG and STDEV also are based on data values since the last break. For example, &AVG6 is the sum of column six (through the current row) divided by COUNT.

At the detail level, &SUM and &CSUM produce the same result. &SUM6 and &CSUM6 in the detail block text each produces the total value of column 6 through the current row.

If you use an aggregation variable with percent (PCT, TPCT, or TCPCT) in detail block text, and if you associate it with a column that has a D edit code, QMF formats the percent value in the detail block text as if it had an L edit code. Likewise, if

you use the aggregation variable standard deviation in detail block text and associate it with a column that has a P or a D edit code, QMF formats the standard deviation in the detail block text as if it had an L edit code.

For more information, see the L code under "Edit Codes for Numeric Data" on page 263 and "Variables Used in Forms" on page 268.

N Select Panel Variation

Specify when to select a panel variation. You must enter one of the following allowable values—blanks are not allowed:

YES Always selected for formatting in the report. It is the default when the variation number is 1.

NO Never selected for formatting. It is the default when the variation number is from 2 through 99. This value can be used to temporarily inhibit the formatting of a variation in a report.

The following two choices allow you to selectively format your report. You can associate an entire panel of detail text and formatting options with a specific condition on the FORM.CONDITIONS panel (conditional formatting), or a specific data column that corresponds to a *branch* of *tree* data.

C1-C999

Can be selected to identify a condition on FORM.CONDITIONS. If the condition is true, the associated FORM.DETAIL variation is formatted.

E1-E999

Can be selected for formatting when data exists for the indicated column. The column is identified by the number following E. This number corresponds to the NUM value for a column on FORM.MAIN or FORM.COLUMNS.

FORM.FINAL

Use FORM.FINAL to make detailed choices about the content and placement of a report's final text. QMF places the text at the end of the report, and you can use it, for example, to identify a report's final summary data.

Area **H** on FORM.MAIN (see page 185) specifies the final text for a report. Whatever you specify in this area of FORM.MAIN is reflected on FORM.FINAL. Similarly, the first line of final text is reflected on FORM.MAIN.

```
FORM.FINAL

A New Page for Final Text?===> NO     B Put Final Summary at Line ===> 1
C Blank Lines Before Text ===> 0
D LINE E ALIGN F FINAL TEXT
----    ------   ----+----1----+----2----+----3----+----4----+----5----+
1       RIGHT
2       RIGHT
3       RIGHT

        *** END ***

1=Help        2=Check     3=End     4=Show        5=Chart       6=Query
7=Backward    8=Forward   9=       10=Insert     11=Delete     12=Report
OK, FORM.FINAL is displayed.
COMMAND ===>                                            SCROLL ===> PAGE
```

A New Page for Final Text?

Reports: Specify whether to place the final text on a page separate from the body in a printed report. A new page is started if the report is not already at the top of the page.

B Put Final Summary at Line

Reports: Specify whether to generate the final summary of a report, and, if so, where to place it in relation to the final text. The value for this entry can be any number from 1 through 999 or the word NONE. The number is the number of the line of final text next to which you want to place the final summary. NONE (or N) omits the final summary.

If you expect the final summary value of a wrapped column to be greater than one line long, include final text on the line corresponding to the last line you expect for your wrapped final summary value. This is only necessary if the wrapped column has a usage code of MAX, MIN, FIRST, or LAST.

For example, if the column NAME (from Q.STAFF) is set to a width of 2, has an edit code of CW, and a usage code of MAX, you must place some final text (perhaps just a period) on the fifth line of FORM.FINAL to see the entire final summary value for that column (YAMAGUCHI).

Two data lines per summary in an across report can appear *only* if the across summary column *and* final summary are both present. This occurs when a column in the form has a usage of CSUM, CPCT, PCT, TPCT, or TCPCT.

When the across summary column is omitted on FORM.OPTIONS, the ACROSS-across values are also omitted and only one line is formatted per group (with ACROSS-down values).

When the final summary is omitted on FORM.FINAL, the ACROSS-down values are omitted and only one line is formatted per group (with the ACROSS-across values).

Charts: When there are two summary lines, but only one is charted by the Interactive Chart Utility (ICU), the second summary data line contains values only in columns for which PCT, CPCT, or CSUM is specified. In these columns:

- The value in the first line is the summary value for that category relative to the ACROSS-across (group) total.
- The value in the second line is the summary value for that category relative to the ACROSS-down (category) total.

See *Using QMF* for information about how QMF works with the ICU.

C Blank Lines Before Text

Reports: Specify the number of blank lines between the body of the report and the first line of final text. The value for this entry can be any number from 1 through 999 or the word BOTTOM. The default is 0.

For example, if you want one blank line between the body of the report and the first line of final text, type 1 in this entry. If you want the final text to be separated from the body by two blank lines, type 2 in this entry.

If you want the final text displayed at the bottom of the current page (regardless of where the body of the report ends) type BOTTOM (or B) in this entry.

D LINE

Reports: Identify the lines of final text and specify their position relative to themselves and to the line at which the final text starts (as indicated in *Blank Lines Before Text*).

The numbers you choose need not start with 1 or be consecutive. You can choose spacing between the lines of the final text and between the body of the report and the first line of final text. For example, if you have three lines of final text, and you choose LINE values of 1, 3, and 5 for the text, QMF starts the final text at the line you indicated in `Blank Lines Before Text` and places one blank line between lines of text. If you do not use 1 as one of your LINE values, QMF does not begin the final text at the line you specified in `Blank Lines Before Text`. It leaves extra blank lines, up to the first specified line number. A blank LINE value tells QMF to ignore any associated text.

For example, these values on FORM.FINAL:

```
LINE  ALIGN  FINAL TEXT
----  -----  -----------------
2     LEFT   GRAND TOTALS FOR
3     LEFT   ALL DEPARTMENTS
```

Display as:

```
GRAND TOTALS FOR
ALL DEPARTMENTS
```

Notice that a blank line appears before the first line of text.

In the example, if you indicated a value of 0 in `Blank Lines Before Text`, you might expect the text `GRAND TOTALS FOR` on the line immediately following the body of the report. But, because the first line of text has a LINE value of 2, QMF skips one blank line (for the *missing* first line of the final text), and then prints the first line from FORM.FINAL on the second line of the final text in the report.

If you use the same LINE value for more than one line, those lines are joined according to the ALIGN value for the additional line or lines. Lines with the same LINE value overlay each other if their ALIGN values are the same or otherwise conflict. For example, you can specify the same LINE value for two lines of final text, with an ALIGN value of LEFT for the first line and an ALIGN value of CENTER for the second line. If the text on the first line extends past the center of the report, the second line overlays part of the first line.

E ALIGN

Reports: Specify where each line of final text is placed horizontally in a report. If a report contains final summary data, the line length for the final text is from the left margin to the beginning of the summary data.

However, if a report does not contain final summary data, the line length for the final text is the complete length of the line (from the left to the right margin). For an online report, the line length is the width of the displayed report; for a printed report, the line length is the width of the printed report.

Left Left-justifies the line of final text.

Right Right-justifies the line of final text. This is the default.

Center
Centers the line of final text.

n Begins the line of final text in the *n*th position of the line. *n* can be any number from 1 through 999999.

Append

Positions the line at the end of the previous line of final text. If append is used on the first line of final text (that is, on the line of text with the lowest LINE value), the line of text is left-justified.

The appended line of text must have the same LINE value as the line of text it is being appended to. If the report is not wide enough to accommodate the appended line of text, some of the text might be truncated.

For example, the following entries on FORM.FINAL:

```
Blank Lines Before Text ===> 0
LINE  ALIGN   FINAL TEXT
----  ------  -----------------
1     RIGHT   TOTAL
1     APPEND  SALARIES
3     RIGHT
```

Produce a report like this:

```
  DEPT        COMM  JOB        SALARY
------  ----------  -----  ----------
    66       55.50  CLERK    10988.00
                    .
                    .
                    .
          1285.00  SALES    17844.00
                           ----------
                       *    66147.00

                           ==========
          TOTAL SALARIES   152223.20
```

F FINAL TEXT

Reports: You can add up to 999 lines of final text using the INSERT command. Each line of text can be up to 55 characters long. You can add text to the line by using APPEND as the ALIGN value, or by specifying a specific horizontal position.

If your installation supports DBCS data, see "Names with Double-byte Characters" on page 272.

By default, final text extends from the left margin of a report to the beginning of the summary data (if a report has summary data) or to the right margin of a report. However, you can specifically choose the width of final text by changing the `Report text line width` entry on FORM.OPTIONS (see page 229).

To make the final text appear in a report in uppercase and lowercase, specify a CASE value of either STRING or MIXED in your profile:

STRING
Displays final text as entered, but converts any other input to uppercase.

MIXED
Displays all input exactly as entered.

Final text can contain the following variable values:

Global variables
Use SET GLOBAL to set variables for use in final text. See "SET GLOBAL" on page 113 for details about this command.

&*n* The last value in the *n*th column on the form used for this report.

&COUNT
The number of rows displayed or printed since the last break. This value is a running count and increases from data row to data row.

&ROW
The number of the last data row of the entire report is printed or displayed in your report.

&CALC*id*
Calculated value

&DATE
The current date

&TIME
The current time

&PAGE
The current page number

For a description of &CALC*id*, see "FORM.CALC" on page 198.

For descriptions of &DATE, &TIME, and &PAGE, see page 193 under *BREAK1 HEADING TEXT*.

&*an* *n* is a valid column number, and *a* is one of the following QMF aggregation functions: AVG, COUNT, CPCT, CSUM, FIRST, LAST, MAX, MIN, PCT, STDEV, SUM, TCPCT, TPCT. The values of the aggregations are based on running values within the current break level.

If you use an aggregation variable with percent (PCT, TPCT, or TCPCT) in detail block text, and if you associate it with a column that has a D edit code, QMF formats the percent value in the detail block text as if it had an L edit code. Likewise, if you use the aggregation variable standard deviation in detail block text and associate it with a column that has a P or a D edit code, QMF formats the standard deviation in the detail block text as if it had an L edit code.

For more information, see the L code under "Edit Codes for Numeric Data" on page 263 and "Variables Used in Forms" on page 268.

FORM.OPTIONS

Use FORM.OPTIONS to adjust the appearance of your report.

Area **J** on FORM.MAIN (*OUTLINE* and *DEFAULT BREAK TEXT*— page 185) specifies two options that affect the overall appearance of a report. What you specify in that area of FORM.MAIN is reflected on FORM.OPTIONS. Similarly, some of what you specify on FORM.OPTIONS is reflected on FORM.MAIN.

```
FORM.OPTIONS

    What do you want for
 A Detail spacing?                                ===> 1
 B Line wrapping width?                           ===> NONE
 C Report text line width?                        ===> DEFAULT
 D Number of fixed columns in report?             ===> NONE

    Do you want
 E Outlining for break columns?                   ===> YES
 F Default break text (*)?                        ===> YES
 G Function name in column heading when grouping? ===> YES
 H Column wrapped lines kept on a page?           ===> YES
 I Across summary column?                         ===> YES
 J Automatic reordering of report columns?        ===> NO
 K Page renumbering at the highest break level?   ===> NO

    Do you want separators for
 L Column heading? ===> YES      M Break summary? ===> YES
 N Across heading? ===> YES      O Final summary? ===> YES

 1=Help        2=Check     3=End     4=Show        5=Chart       6=Query
 7=            8=          9=       10=           11=           12=Report
 OK, FORM.OPTIONS is displayed.
 COMMAND ===>                                          SCROLL ===> PAGE
```

A Detail spacing?

Reports: Select spacing between tabular data lines or detail blocks. The spacing within detail block text is not affected. The value can be any number from 1 through 999. The default is single spacing with no blank line between each block of text.

The Blank Lines after Block option on the FORM.DETAIL panel (page 216) also affects the spacing between detail blocks.

B Line wrapping width?

Reports: Specify whether the columns in a report are to be wrapped, and if so, at what width. The value for this entry can be any number from 1 through 999 or the word NONE. The default is NONE, indicating that the lines in a report are not to be wrapped.

Lines cannot be wrapped in ACROSS reports or reports with column wrapping. Detail heading text and detail block text are not wrapped. They are truncated at the report text line width. However, if the value for report text width is DEFAULT, and the line wrapping width is not NONE, the detail heading text and detail block text are truncated at the line wrapping width.

If the value in this entry area is greater than the print width, the data in the columns of a report is truncated on the right.

If you want line wrapping (that is, the detail lines in a report begin on one line and continue on one or more subsequent lines), type a number in this entry area to indicate the maximum width of the lines of data you want in the report. As many whole columns as possible are positioned across the report. Any remaining columns are placed on one or more subsequent lines of the report. All wrapped lines begin with the column indent, then include the tabular data.

If a column and its indent are too wide to fit within the line wrapping width specified, a new line does not begin for the column and the column is cut off on the right.

Only column headings, tabular data, and column summaries are wrapped when you specify a width. All other data in the report is formatted as usual.

Following is part of a report with line wrapping (at a width of 35) and tabular data line spacing of 2.

```
    ID  NAME         DEPT  JOB
------  ---------  ------  -----
 YEARS       SALARY        COMM
------  ----------  ----------
   160  MOLINARE       10  MGR
     7    22959.20           -

   210  LU             10  MGR
    10    20010.00           -

   240  DANIELS        10  MGR
     5    19260.25           -
```

C Report text line width?

Reports: Specify the width of the final text, detail heading text, detail block text, and break text in a report. The values in this entry area can be DEFAULT, COLUMNS, or any number from 1 through 999999.

DEFAULT

Break footing text and final footing text use the full width of all columns *up to the first summary column* as indicated in FORM.COLUMNS and FORM.MAIN.

COLUMNS

All text areas use the full width of all columns as indicated in FORM.COLUMNS and FORM.MAIN. (This option is the same as DEFAULT for detail heading text and detail block text.)

A number from 0 through 999999

The width in characters for all text types. 0 indicates that no text is formatted.

D Number of fixed columns in report?

Reports: Specify the number of columns that remain in place when you scroll reports horizontally on the screen. When fixed columns are specified, the report is divided into a fixed area and a scrollable area. For printed reports of more than one page, fixed columns are repeated on the left side of each page. The scrollable area of a printed report refers to the area that changes during page splitting.

The value can be any number from 1 through 999 or the default NONE.

If the number specified is greater than the number of columns in the report, all columns are fixed. Columns with OMIT usages are not counted as fixed columns.

Fixed columns can be used with column reordering (SEQ). If the columns were reordered and you select a number of columns, *n*, as fixed columns, the first *n* columns of the new order are the fixed columns. This applies to automatic reordering and user reordering.

The fixed column area of a report can affect the text of the report. The portions of break, detail, and final text that are within the fixed area are repeated on the left side of any printed pages of the report. The portion of break, detail, and final text that are within the scrollable area appear on the first page of a printed report, but do not appear on subsequent pages when page splitting occurs.

Page heading and footing text are not affected by fixed column settings in either displayed or printed reports.

Fixed columns can conflict with other report options. You cannot use line wrapping with fixed columns (see **B** *Line wrapping width?* on

page 230). Also, if the total width of all fixed columns in a report is greater than the displayable screen width, both the displayed and printed versions of the report are affected. For displayed reports, you can scroll the report up and down, but you cannot scroll it to the left or right. For printed reports, this message is displayed:

```
The report cannot be printed; the fixed area is
too wide.
```

E Outlining for break columns?

Reports: If you assigned a usage code of BREAK to one of your columns, use this entry area to determine whether the value in the BREAK column is to be displayed only when the value changes or on every line in a report.

YES Displays the value in the BREAK column only when the value changes.

NO Displays the value in the BREAK column on every tabular data line in the report.

Outlining begins at the top of a page. The value is printed at the top of a page even if it hasn't changed from the bottom line of the previous page.

F Default break text (*)?

Reports: If a report contains breaks for which you did not indicate break footing text, use this entry area to specify whether to generate break footing text to mark the BREAK aggregation line.

The default break text consists of one asterisk for the highest numbered break level text, two asterisks for the next-highest numbered break level text, and so on.

G Function name in column heading when grouping?

Reports: If a report has combined data (for example, as a result of summing a column) and you use the usage code GROUP to suppress the tabular data lines, this entry area determines the heading of the aggregated column.

YES Displays a word indicating the type of aggregation as part of the column heading.

NO Suppresses the aggregation name in the column heading.

Charts: If you use YES for charts, the function name appears in the legend on a chart. NO is recommended.

H Column wrapped lines kept on a page?

Reports: If you specified column wrapping for one or more columns in a report, this entry area determines whether the wrapped columns can be split between two pages.

YES Unless the wrapped column is longer than the page depth.

NO Allows wrapped columns to be split between pages if necessary.

I Across summary column?

Reports: Specify whether to display the automatically generated across summary column. `Across summary column` produces additional columns that summarize (total) *across* the specified columns.

In the following ACROSS report, you can read the lines for departments 10 through 84 across to see the average salary for each job and the department average in the last column. The job salary averages are under the final summary separators at the bottom of each column.

```
          <---------------- JOB ----------------->
          <- CLERK -->  <-- MGR --->  <- SALES -->  <- TOTAL -->
            AVERAGE       AVERAGE       AVERAGE       AVERAGE
 DEPT       SALARY        SALARY        SALARY        SALARY
------    ----------    ----------    ----------    ----------
    10                    20865.86                    20865.86
    15      12383.35      20659.80      16502.83      15482.33
    20      13878.68      18357.50      18171.25      16071.53
    38      12482.25      17506.75      17407.15      15457.11
    42      11007.25      18352.80      18001.75      14592.26
    51      13914.90      21150.00      18555.50      17218.16
    66      10988.00      18555.50      18844.23      17215.24
    84      13030.50      19818.00      16649.25      16536.75
          ==========    ==========    ==========    ==========
            12612.61      19805.80      17869.36      16675.64
```

The across summary column is displayed to the right of the columns in a report.

It is possible to get two data lines per summary in any across report for which at least one column has a usage of PCT, CPCT, or CSUM. However, this *only* happens if the across summary column and final summary are both present or both absent in the report.

When two data lines per summary are returned, the second summary data line contains values only in those columns for which PCT, CPCT, or CSUM is specified. In such columns, the value in the first line is the summary value for that subcategory relative to the

ACROSS-across (group) total. The value in the second line is the summary value for that subcategory relative to the ACROSS-down (subcategory) total.

When the across summary column is omitted (on FORM.OPTIONS), the ACROSS-across values are also omitted and only one line is formatted per group (with the one line containing the ACROSS-down values).

When the final summary is omitted (on FORM.FINAL), the ACROSS-down values are omitted and only one line is formatted per group (with the one line containing the ACROSS-across values).

Charts: Only one of the two possible across summary lines of data can be transferred to the ICU. Charts cannot display both lines of data. If two values exist for a column in each group, the value on the second line (ACROSS-down) is the value that is passed to the ICU and shows on the chart.

You can force the ACROSS-across values to be charted if the final summary is omitted. This causes the ACROSS-down values to be omitted.

J Automatic reordering of report columns?

Reports: Specify whether the columns in a report are automatically reordered when you specify a usage of BREAK*n*, GROUP, or one of the aggregating functions (such as AVERAGE, COUNT, FIRST, LAST, MAXIMUM, MINIMUM, STDEV, SUM, CPCT, CSUM, PCT, TPCT, or TCPCT).

The default is NO. The columns are not automatically reordered. They appear in a report in the order in which they are shown on FORM.MAIN or FORM.COLUMNS—even if you use a usage code of BREAK*n*, GROUP, or one of the aggregating functions. If you specify YES, the columns are reordered according to the following rules:

- BREAK*n* columns to the far left
- GROUP columns to the left after BREAK*n* columns
- All nonaggregated columns to the left after BREAK*n* and GROUP columns
- All aggregated columns to the far right

If you use ACROSS as a usage, the value in this entry area is ignored because the purpose of an ACROSS report is defeated if the columns cannot be reordered.

Charts: If automatic reordering of report columns is set to YES, it can have an effect on which Y data column is selected for the X-axis in a chart. The following conditions must be met for automatic column reordering to have an effect:

- No GROUP or BREAK*n* usage codes are used on the form to select Y data columns for the X-axis of the chart.
- An aggregation function (such as AVERAGE, SUM, or COUNT) is used on the form with one of the columns.

If these conditions are met, the aggregated columns are moved from the left side of the report to the far right. For example, suppose that YEARS originally appeared on the left side of your report; therefore, the YEARS column was plotted on the X-axis when you displayed your chart. (You did not specify GROUP or BREAK to select data columns for the X-axis.)

Additionally, suppose you decide to use the aggregation function of AVERAGE with YEARS; the YEARS column now moves to the far right of the report. Because it is no longer the leftmost column, it is not plotted on the X-axis of your chart. The column that now appears at the left of your report is plotted on the X-axis.

K Page renumbering at the highest break level?

Reports: Specify whether a printed report begins a new page beginning with the number 1 whenever the value in the control column with the highest break level changes. The highest break level is the one with the lowest number. This option affects only printed reports, because QMF treats online reports as one long page.

Use the default for this option, NO, to indicate that you do not want to restart the numbering of a report whenever the value in the highest level break column changes; enter YES in this entry area to start page renumbering. If you indicate YES, that value is ignored unless you use at least one BREAK usage on the form and enter YES in the New Page for Break entry area on the corresponding FORM.BREAK*n* panel.

L Column heading?

Reports: Specify whether the dashed lines that separate the column headings from the tabular data lines in the report are to be displayed.

M Break summary?

Reports: Specify whether the equal signs that separate the break summary from the break member lines are to be displayed.

N Across heading?

Reports: Specify whether the dashed lines and arrows that mark columns in across reports are to be displayed.

O Final summary?

Reports: Specify whether the equal signs that separate the final summary from the body of the report are to be displayed.

FORM.PAGE

Use FORM.PAGE to make detailed choices about the content and placement of the page headings and footings in a report. For online and printed reports, QMF places headings at the top of an online report and footings at the bottom. Headings and footings appear at the top and bottom of each page of a printed report.

Area **G** on the FORM.MAIN panel (see **G** PAGE on page 187) specifies page headings and footings for a report. Whatever you specify in area **G** of FORM.MAIN is shown on FORM.PAGE. Similarly, the first line of page heading and footing that you specify on FORM.PAGE is shown on FORM.MAIN.

```
FORM.PAGE

A Blank Lines Before Heading ===> 0      B Blank Lines After Heading ===> 2
C LINE D ALIGN E PAGE HEADING TEXT
    ----   ------   ----+----1----+----2----+----3----+----4----+----5----+
    1      CENTER
    2      CENTER
    3      CENTER
    4      CENTER

F Blank Lines Before Footing ===> 2      G Blank Lines After Footing ===> 0
H LINE I ALIGN J PAGE FOOTING TEXT
    ----   ------   ----+----1----+----2----+----3----+----4----+----5----+
    1      CENTER
    2      CENTER
    3      CENTER
    4      CENTER
           *** END ***

1=Help        2=Check     3=End     4=Show        5=Chart       6=Query
7=Backward    8=Forward   9=       10=Insert     11=Delete     12=Report
OK, FORM.PAGE is displayed.
COMMAND ===>                                            SCROLL ===> PAGE
```

A Blank Lines Before Heading

Reports: Specify the number of blank lines between the top of a page and the first line of the page heading. The value can be any number from 1 through 999.

Charts: An entry in this area determines vertical placement of the heading on the chart. However, too many blank lines can change the labels on the Y-axis.

B Blank Lines After Heading

Reports: Specify the number of blank lines between the last line of

page heading and the body of the report. The value can be any number from 1 through 999. The default is 2.

C LINE

Reports: Identify the lines of page heading text and specify their position relative to themselves and to the line at which the page heading starts (as indicated in the `Blank Lines Before Heading` entry area).

The numbers you choose need not start with 1 or be consecutive. You can choose spacing between the lines of the page heading and between the top of the page and the first line of page heading text. A blank ignores any associated text.

For example, these values on FORM.PAGE:

```
LINE  ALIGN   PAGE HEADING TEXT
----  ------  ----+----1----+----2----
4     LEFT    MONTHLY INVENTORY
4     RIGHT   PAGE &PAGE
2     CENTER  ABC COMPANY
```

Display as:

```
                ABC COMPANY

    MONTHLY INVENTORY       PAGE 1
```

Charts: Use LINE to position the lines of heading text vertically relative to themselves and to the line at which the chart (page) heading starts.

D ALIGN

Reports: Specify where each line of the page heading text is placed horizontally in the report. You can place the lines anywhere in the width of the report. For an online report, the width is the width of the displayed report; for a printed report, the width is the page width.

Left Left-justifies the line of page heading text.

Right Right-justifies the line of page heading text.

Center
Centers the line of page heading text.

n Begins the line of page heading text in the *n*th position of the line. *n* can be any number from 1 through 999999.

Append
Attaches the line at the end of the previous line of page heading text. If append is used on the first line of page heading text, the line of text is left-justified.

The appended line of text must have the same LINE value as the line of text it is being appended to. If the report is not wide enough to accommodate the appended line of text, some of the text might be truncated.

For example, the following entries on FORM.PAGE:

```
LINE  ALIGN    PAGE HEADING TEXT
----  ------  ----+----1----+----2----
1     CENTER   ABC COMPANY MANAGERS --
1     APPEND    &DATE, &TIME
3     CENTER
4     CENTER
5     CENTER
```

Align the columns like this:

```
  ABC COMPANY MANAGERS -- 98/08/04, 14:20

    ID  NAME         DEPT  JOB     YEARS      SALARY        COMM
------  ---------  ------  -----  ------  ----------  ----------
    10  SANDERS        20  MGR         7    18357.50           -
    30  MARENGHI       38  MGR         5    17506.75           -
```

Charts: ALIGN does not affect a chart heading, *except* when LINE is used to place more than one line of text on the same line of the heading.

E PAGE HEADING TEXT

Reports: Enter the text you want to appear either at the top of each page of a printed report or before the first line of a report displayed at a terminal. You can add up to 999 lines of page heading text using the INSERT command. Each line of text can be up to 55 characters long. You can add text to the line by using APPEND as the ALIGN value, or by specifying a specific horizontal position.

If your installation supports DBCS data, see "Names with Double-byte Characters" on page 272.

To make the page heading text appear in a report in uppercase and lowercase, specify in your PROFILE a CASE value of either STRING or MIXED:

STRING
Displays the page heading text as entered, but converts any other input to uppercase.

MIXED
Displays all input exactly as entered.

Page headings can contain the following variable values:

&*n* *n* is a number that stands for the first value in column *n* on the current page of this report. Column *n* is the nth column selected from the database, or the nth column listed on FORM.MAIN and FORM.COLUMNS.

&ROW
The number of the first data row on the current page is printed or displayed in your report.

&DATE
The current date

&TIME
The current time

&PAGE
The current page number

When &DATE, &TIME, or &PAGE are entered in page heading text, the system date, time, or page number do not appear at the bottom of printed reports. This applies only to these three variables entered on FORM.PAGE.

For descriptions of &DATE, &TIME, and &PAGE, see page 193 under *BREAK1 HEADING TEXT*.

Charts: The preceding description regarding PAGE HEADING TEXT applies to charts, except for part of the description of ALIGN. The only time that the value specified for ALIGN affects a chart heading is when LINE is used to place one or more lines of text entered on FORM.PAGE on the same line in the formatted report. If you're not using the LINE function, the chart heading is automatically centered.

F Blank Lines Before Footing
Reports: Specify the number of blank lines between the body of the report and the first line of page footing. The value for this entry can be any number from 1 through 999. The default is 2.

G Blank Lines After Footing
Reports: Specify the number of blank lines between the last line of page footing and the bottom of the page. The value for this entry can be any number from 1 through 999.

If a report contains break summary data and one or more wrapped columns, you might need to increase the value in this entry area to see all the lines of summary data. For more information, see the CW code under "Edit Codes for Character Data" on page 261.

H LINE

Reports: Identify the lines of page footing text and specify their position relative to themselves and to the line at which the page footing starts (as indicated in the `Blank Lines Before Footing` entry area). You can specify any number from 1 through 999 or a blank.

For example, these values on FORM.PAGE:

```
LINE  ALIGN   PAGE FOOTING TEXT
----  ------  ----+----1----+----2----
3     LEFT    MONTHLY INVENTORY
3     RIGHT   PAGE &PAGE
2     LEFT    ABC COMPANY
```

Display as:

```
ABC COMPANY
MONTHLY INVENTORY       PAGE 1
```

Notice that a blank line appears before the first line of text.

I ALIGN

Reports: Specify where each line of the page footing text is to be placed horizontally in the report. You can place the lines of text anywhere between the left and right margin. For an online report, the width is the width of the displayed report; for a printed report, the width is the page width.

Left Left-justifies the line of page footing text.

Right Right-justifies the line of page footing text.

Center
Centers the line of page footing text.

n Begins the line of page footing text in the *n*th position of the line. *n* can be any number from 1 through 999999.

Append
Positions the line at the end of the previous line of page footing text. If Append is used on the first line of page footing text (the line of text with the lowest LINE value), the line of text is left-justified.

The appended line of text must have the same LINE value as the line of text it is being appended to. If the report is not wide enough to accommodate the appended line of text, some of the text might be truncated.

For example, the following entries on FORM.PAGE:

```
LINE  ALIGN   PAGE FOOTING TEXT
----  ------  ----+----1----+----2----
1     CENTER  ABC COMPANY MANAGERS --
1     APPEND  &DATE, &TIME
```

align columns like this:

```
    ID  NAME         DEPT  JOB     YEARS      SALARY        COMM
------  ---------  ------  -----  ------  ----------  ----------
                             .
                             .
                             .

    10  SANDERS        20  MGR         7    18357.50           -
    30  MARENGHI       38  MGR         5    17506.75           -


                  ABC COMPANY MANAGERS -- 98/08/04, 16:20
```

J PAGE FOOTING TEXT

Reports: Enter the text you want to appear either at the bottom of each page of a printed report or before the last line of a report displayed at a terminal. You can add up to 999 lines of page footing text using the INSERT command. Each line of text can be up to 55 characters long. You can add text to the line by using APPEND as the ALIGN value, or by specifying a specific horizontal position.

If your installation supports DBCS data, see "Names with Double-byte Characters" on page 272.

To make the page footing text appear in a report in uppercase and lowercase, specify in your profile a CASE value of either STRING or MIXED:

STRING
Displays page footing text as entered, but converts any other input to uppercase.

MIXED
Displays all input exactly as entered.

Page footings can contain the following variable values:

Global variables
Use SET GLOBAL to set variables for use in page footing text. See "SET GLOBAL" on page 113 for details about this command.

&*n*
n is a number that represents the last row in column *n* processed for the current page of this report. Column *n* is the nth column selected from the database, or the nth column listed on FORM.MAIN and FORM.COLUMNS.

&ROW
The number of the last data row on the current page is printed or displayed in your report.

&DATE
The current date

&TIME
The current time

&PAGE
The current page number

When &DATE, &TIME, or &PAGE are entered in page footing text, they appear (instead of the system date, time, or page number) at the bottom of the printed report. This applies only to these three variables entered on FORM.PAGE.

For descriptions of &DATE, &TIME, and &PAGE, see page 193 under *BREAK1 HEADING TEXT*.

Mistakes on Form Panels

QMF distinguishes between two types of mistakes:

Error conditions
Mistakes that require correction before the form can be used

Warning conditions
Mistakes that do *not* require correction before the form can be used

Error Conditions

An error condition results from entering an invalid value in an entry area. For example, typing `Y0` in the OUTLINE field on FORM.OPTIONS results in an error because `Y0` is not an allowed value for the entry area.

An error can also occur if there is a conflict that prevents the report from being displayed. For example, SUM is a valid entry for USAGE on a numeric column. However, SUM produces an error if entered for a column with character data.

You must correct errors before using the form. However, you can save, import, export, display, and print forms that contain errors.

After you correct errors, QMF identifies any warning conditions.

Warning Conditions

A warning condition results when the values in two or more entry areas conflict. Unlike an error, a warning condition need not be corrected before you use the form. Instead, QMF warns you of the conflict and interprets the condition to format the report or chart.

You can either accept the report or chart as is, or change one or more of the conflicting entries to correct the form.

Table 14, following, lists some common warning conditions and how QMF formats the report. These warning conditions can also affect the chart representing that report.

Table 14. Warning Conditions

Condition	QMF Action
More than one ACROSS usage	Accepts first ACROSS; omits remaining ACROSS columns from report
ACROSS usage without GROUP usage	Omits ACROSS column from report
GROUP usage without aggregating usage	Omits GROUP column from report
ACROSS and GROUP usage with one or more blank usages	If aggregation used, omits columns with blank usages from report; otherwise, omits ACROSS and GROUP columns from report
GROUP usage with at least one aggregation usage and one or more blank usages	Omits columns with blank usages from report
Line wrapping with ACROSS usage or with column wrapping edit code	Ignores line wrapping
ACROSS usage without automatic column reordering	Ignores value of column reordering option; produces standard ACROSS report

Checking for and Correcting Mistakes

Normally, pressing Enter while displaying a form panel positions the cursor on the command line. However, if you press Enter immediately after entering one or more erroneous values, QMF highlights any errors and sends you a message describing the first one. *Pressing Enter does not identify any errors made during a previous interaction.*

If you press Enter again (with or without correcting the first error), QMF positions the cursor on the command line. To receive a message about the next error in the form, use the CHECK subcommand (see “CHECK” on page 11).

Mistakes on Form Panels

QMF checks a form for errors whenever you issue a command that uses a form—for example, DISPLAY REPORT, PRINT CHART, PRINT REPORT, EXPORT REPORT, EXPORT CHART, or RUN QUERY with the FORM option. (You can issue the command either by entering it on the command line or by using a function key.) QMF also checks for errors when you display the form.

If a form contains an expression with an error, this error is not detected until QMF passes the values to REXX for evaluation. If you enter a QMF command (other than CHECK, DISPLAY REPORT, DISPLAY CHART, PRINT REPORT, PRINT CHART, or RUN QUERY with the FORM option) while displaying a FORM, QMF processes your command whether or not the FORM contains errors. The displayed message pertains to the command you entered.

Therefore, you can display, save, import, or export a FORM even if the FORM contains errors or warning conditions. Saved, imported, or exported forms are saved or transported in their present state, with mistakes and ERROR and WARNING indicators in place.

Form and Data Incompatibility

There might be times when you modify a form in such a way that the form is inconsistent with the data. This situation is treated differently from error and warning conditions. There is no error message at the top of the screen when the cursor is positioned, and the CHECK command does not identify the problem. Instead, when you try to display the report, a message is displayed and the form panel containing the incompatibility is displayed.

Examples of Possible Incompatibilities:

- The number of columns in the form (excluding defined columns) and in the data must be equal.
- Edit codes in the form must match the data type for each column in the data.
- Every LONG VARCHAR and LONG VARGRAPHIC column in the data must have a blank or an OMIT usage code in the form.

Using REXX with QMF Forms

Note to CICS users
FORM.CALC, FORM.CONDITIONS, and Column Definition use expressions written in REXX, which QMF does not support in CICS.

Expressions used in FORM.CALC, FORM.CONDITIONS, and FORM.COLUMNS (Column Definition) can consist of terms (*strings*, *symbols*,

and *functions*) interspersed with operators and parentheses. Do not execute QMF commands (using the callable or command interfaces) from within a REXX expression or EXEC.

Strings are literal constants enclosed in single or double quotation marks. For example, `'High'` and `"Low"`.

Symbols are numeric literals (numbers), variables, or nonnumeric literals without quotation marks.

- *Numeric literals* can be expressed in integer, decimal, or exponential notation. For example:

```
123
 25.45
   .432
  1.7E4   (equivalent to 17000)
  7.6e-3  (equivalent to .0076)
```

  Commas are not allowed, except as decimal points. (QMF allows commas for decimal points only when they are defined as such to the database manager.)
- *Variables* are restricted by how the expression is used. See the table in "Variables Used in Forms" on page 268 for a summary of allowable variables.
- *Nonnumeric literals* are symbols that are neither numbers nor variables. These are handled like strings in the evaluation of expressions.

Functions have the following syntax:

```
function-name([[expression][,][expression][,] ...])
```

where 0 to *n* `expression` arguments can exist (*n* is the maximum number of comma-separated expressions allowed by REXX).

Function-name must identify either a built-in function or an external function, for example, a REXX program. Evaluation of an expression is left to right, modified by parentheses and by operator precedence in the usual algebraic manner (with the exception of the minus prefix). See "Operator Priorities" on page 249.

Using Calculated Values in Reports

There are three ways to include calculated values in a QMF report:

- Include calculations in the query with SQL statements.
- Define a new column based on an expression.
- Specify and use expressions defined on the FORM.CALC panel.

The first method of including calculations in a report is handled by the database, and the other two are handled by QMF from specifications made on the form. When calculations are specified on the form, they are evaluated using REXX.

QMF verifies conditions, column definitions, and expressions whenever a form is loaded, imported, displayed, or run with a query. When you modify a condition, column definition, or expression, QMF verifies it again. This can result in a REXX error if QMF passes unexpected data during verification. To avoid this kind of REXX error, include your calculation, along with validation statements, in a REXX EXEC.

When using FORM.CONDITIONS or Column Definition, make sure the expression or EXEC returns the same value if invoked multiple times with the same parameters. If the EXEC doesn't return the same value, breaks might not resolve as expected, and summary values might not match printed results.

There can be a significant difference in performance, capability, and flexibility of calculations performed by the database and those evaluated using REXX. A REXX program can return values dependent upon complex logic or the values processed by REXX functions. Although REXX offers more function and programming options, there can be some drawbacks to relying on REXX for all of the calculations in a report.

REXX requires a certain amount of resource to evaluate expressions. If REXX is called repeatedly for completion of a report, you might notice an impact on performance. Because of this, you might choose to specify some calculations in the query. For example, to create a new column in a report based on the following:

```
((Column A - Column B) * 100) / Column B
```

you can enter the expression in SQL and rerun the query, or enter the expression as the definition for a new column in the form and display the report. Because the column defined in the form requires a call to REXX for every detail row processed for the report, you might decide to define the new column in the query.

How QMF and REXX Interact

When executing REXX expressions and EXECs:

1. QMF evaluates substitution and global variables in an expression and adds the prefix `DSQ$#VAL=` to the expression.
2. QMF then creates a literal string by applying double quotation marks to any global variables or any substitution variables resulting in character data. This prevents these variables from being interpreted as REXX variables or modified by REXX.

3. QMF passes the expression to an interpret instruction.
4. The interpret instruction executes your REXX expression or EXEC.

Executing the same REXX EXEC in CMS and TSO can produce different results.

Because QMF does not place double quotation marks around numeric values in REXX expressions, any negative values in your expression might not be treated as such. To avoid having negative signs treated as the subtraction arithmetic operator, you can separate the variables that get passed to REXX with commas (instead of spaces) or enclose any negative values (including substitution variables that might result in negative values) with double quotation marks. For example, `myexec(A -1)` results in an evaluation error, but `myexec(A,-1)` and `myexec("A" "-1")` do not. being interpreted as arithmetic operators. However, if you use commas, be aware that:

- There are limits on the number of commas allowed in an expression.
- You might need to modify your parse statement to include commas.

REXX limits the maximum length of a single string. Therefore, when using columns containing data that exceed this limit, your REXX EXEC might produce unexpected results. Also, because QMF adds characters to strings (as noted above), a string can exceed the limit after it is processed by QMF.

If REXX passes a string longer than 32,767 bytes to QMF, the string is truncated to 32,767 bytes.

For information about limits on commas and string length in expressions, see the *TSO/E Procedures Language MVS/REXX Reference* (for TSO) or the *Virtual Machine/Enterprise Systems Architecture REXX/VM Reference*.

When using REXX within QMF, performance might be adversely affected. To improve performance, start QMF using the REXX callable interface.

When Expressions Are Evaluated by REXX

Expressions specified on the FORM.CALC panel and used as substitution variables (&CALCn) in text areas of the form are passed to REXX for evaluation at different times, depending on where they are placed in the form.

- Calculations are processed when they are formatted:
  - References on FORM.DETAIL panels with SELECT=NO or SELECT=Cn (where `n` condition is false) are not evaluated.
  - If the calculation is listed on separate lines in one variation, it might be evaluated multiple times.

  - If the calculation is referenced on several selected FORM.DETAIL variations (in which the Select Panel Variation field is YES or Cn, where condition n is "true"), the calculation might be evaluated multiple times.
- Expressions specified on the FORM.CALC panel and used as a usage code on the FORM.COLUMNS panel are evaluated by REXX whenever the value is needed for formatting.
- Expressions specified on the FORM.COLUMNS Definition panel to define a new column are evaluated by REXX each time a row is fetched. Rows can be fetched more than once (for example, to support printing a report in which page-splitting is required) or to support a usage code (such as TCPCT) that requires all the data to be retrieved first.
- Expressions specified on the FORM.CONDITIONS panel and referred to on a FORM.DETAIL panel variation are evaluated by REXX at least once for every detail row formatted in a report.

REXX Operators

CICS users

FORM.CALC, FORM.CONDITIONS, and Column Definition use expressions written in REXX,which QMF does not supoport in CICS.

Each operator (except the prefix operator) acts on two terms. These terms can be symbols, functions, or subexpressions in parentheses. Each prefix operator acts on the term or subexpression that follows it. The following operators are allowed in QMF expressions:

Arithmetic Operators

+ Add

− Subtract

***** Multiply

/ Divide

% Divide and return only the integer part of the quotient

// Divide and return only the remainder (not *modulo* because the result can be negative)

****** Raise a number to a whole-number power (exponentiation)

Prefix −
Negate the following term

Prefix +
Take the following term as is

Comparative Operators

== Exactly equal (identical)

= Equal (numerically or when padded)

¬==, /==
Not exactly equal (inverse of ==)

¬=, /= Not equal (inverse of =)

> Greater than

< Less than

< > Not equal

>= Greater than or equal

¬< Not less than

<= Less than or equal

¬> Not greater than

Concatenation Operator

|| Concatenate terms (can use no blanks or one blank)

REXX provides other concatenation operators. See the *TSO/E Procedures Language MVS/REXX Reference* or the *Virtual Machine/Enterprise Systems Architecture REXX/VM Reference* for more information.

Logical (Boolean) Operators

& AND (returns 1 if *both* terms are true)

| Inclusive OR (returns 1 if *either* term is true)

&& Exclusive OR (returns 1 if either term is true, but not both)

Prefix ¬
Logical NOT (negates; 1 becomes 0 and vice versa)

Operator Priorities

Expression evaluation is from left to right. Modify this by using parentheses and operator priority.

Use parentheses to clarify the meaning when the priority of operators is not obvious. An expression in parentheses is evaluated first.

When the sequence:

```
term1 operator1 term2 operator2 term3 ...
```

is encountered, and operator2 has a higher priority than operator1, the expression (term2 operator2 term3 ...) is evaluated first, applying the same rule repeatedly, as necessary.

For example, * (multiply) has a higher priority than + (add), so 3 +2*5 evaluates to 13, rather than 25, which results if strict left-to-right evaluation occurred.

The order of priority of the operators (from highest to lowest):

+ − ¬ Prefix operators

****** Exponentiation

*** / % //**
Multiply and divide

+ − Add and subtract

|| Concatenation with or without blank

=, >, ...
All comparison operators

& And

|, && Or, exclusive or

The & and && operators must be followed by a blank in calculation expressions to differentiate them from substitution variables.

For operators of equal priority (the multiply and divide operators, for example), the left-to-right rule prevails.

The only difference between these priorities and conventional algebra is that the prefix minus operator has a higher priority than the exponential operator. Thus −3**2 evaluates to 9, not −9.

Report Calculation Expression Examples

The following assumptions produce the results shown:

```
&SUM1 has the value 1600
&SUM2 has the value 400
&DATE has the value "87/12/15"
```

Expression:
Result:

&SUM2/25
16

&SUM2-&SUM1*.25
0

&SUM1+&SUM2 < 4000
1 (true)

' ' = '' 1 (true)

' ' == ''
0 (false)

&SUM1+(&DATE<'88')*&SUM2
2000

date(u) (built-in function)
"12/15/87"

And this expression:

```
substr(&DATE,4,5) || "/" ||
substr(&DATE,7,8) || "/" ||
substr(&DATE,1,2)
```

produces the same result as *date(u)*.

See *Using QMF* for additional examples of FORM.CALC.

Usage Codes

QMF usage codes define how to use column data to produce reports and charts.

This section contains brief descriptions of each of the QMF usage codes. For additional information, see *Using QMF*. It contains basic information and exercises on usage codes. It also contains detailed information and examples of how reports and charts can be changed using usage codes.

ACROSS Usage Code

Reports: A column can have a usage of ACROSS only if one or more columns have a usage of GROUP. In that case, the summary line for each group value can contain several sets of results from the columns that use aggregations. There is one set for each group of values in the column that uses ACROSS. The heading for a column that uses ACROSS has three levels:

1. The column heading as entered on the form
2. The set of values within the column

3. For each value in the set, the column headings for columns with aggregations

If more than one column has a usage of ACROSS, QMF accepts the first ACROSS and omits the remaining ACROSS columns from the report. If one column has a usage of ACROSS, no other column should have a blank usage. If you leave a column usage blank in an across report, QMF runs the report but omits all columns with blank usages.

For an example of an across summary report with a usage of AVG, see  *Across summary column?* on page 233.

Charts:

The information about reports also applies to charts. ACROSS on charts displays a category of data (such as JOB) broken down into subcategories (such as SALES and CLERK) within a larger category (such as DEPARTMENT). The data for these subcategories is displayed in a bar chart. Color terminals display the bars in different colors for different subcategory bars.

Aggregation Usage Codes

Two types of aggregations are described here:

- Those that summarize the data in a column:

```
AVERAGE     COUNT       FIRST     LAST
MAXIMUM     MINIMUM     STDEV     SUM
```

- Those that replace the data value with a calculation and produce interim and final results:

```
CSUM     PCT     CPCT     TPCT     TCPCT
```

Table 15, following, shows which aggregation usage codes are valid when used with different data types.

Table 15. Valid Usage Codes for Data Types

Datatype	Valid Usage Codes
Numeric	AVG, COUNT, CPCT, CSUM, FIRST, LAST, MAX, MIN, PCT, STDEV, SUM, TCPCT, TPCT
Character, Date, Time, Timestamp	COUNT, FIRST, LAST, MAX, MIN

Note: LONG VARCHAR and LONG VARGRAPHIC columns cannot be aggregated. The only valid usage codes for these data types are blank and OMIT.

Summarize Data in a Column

Reports: Aggregation usage codes summarize the data in a column. The results of an aggregation can appear in the middle of the report as subtotals or at the end of the report as totals.

AVERAGE
Average of the values in the column

COUNT
Count of the values in the column

FIRST First value in the column

LAST Last value in the column

MAXIMUM
Maximum value in the column

MINIMUM
Minimum value in the column

STDEV
Standard deviation of the values in the column

SUM Sum of the values in the column

When you use MAXIMUM and MINIMUM on character, date, time, timestamp, or graphic data, QMF uses an EBCDIC collating sequence to compare the data. To determine MAXIMUM and MINIMUM for numeric data, QMF uses algebraic compares. Nulls can be included in the result for MAX, MIN, FIRST, and LAST.

A date/time function applied to a DATE, TIME, or TIMESTAMP value changes the data type of that value to numeric. Therefore, the resulting value can be aggregated.

The format of the result is determined by the edit code of the column, except for COUNT, STDEV, and percentage aggregations. COUNT can be applied to data of any type, but always produces an integer result; hence, its result is formatted with edit code K. STDEV, PCT, CPCT, TPCT, and TCPCT are formatted with edit code L. (See “Edit Codes for Numeric Data” on page 263.)

Charts: The information on reports for these usage codes is also true for charts.

AVERAGE, MAXIMUM, MINIMUM, STDEV, and SUM can all be useful in charting QMF data. Entries such as FIRST and LAST might not be useful in a chart format.

The following values are sent as null values to the ICU when you display a chart of the report:

- Null values in a report
- Data values too long for the width of the column
- Undefined values
- Arithmetic overflow values

Replace Data Value with a Calculation

Reports The following codes name aggregations that replace each detail line value in a column with a calculation and show a final result of the aggregation at the end of the report. They can also appear in the middle of the report as subtotals.

CSUM
The cumulative sum for each value in a column.

PCT The percentage each value is of the total:

- In reports with BREAK or ACROSS usages, PCT shows what percentage each value in the break or across group is of the break or across total.
- In all other reports, PCT shows the percentage each value in the column is of the column total.

CPCT The cumulative percentage for each value in a column:

- In reports with BREAK or ACROSS usages, CPCT shows the cumulative percentage of the break or across total for each value in the break or across group.
- In all other reports, CPCT shows the cumulative percentage each value in the column is of the column total.

TPCT The total percentage each value is of the column total:

- In reports with BREAK or ACROSS usages, TPCT shows what percentage each value in the column is of the column total.
- In all other reports, TPCT displays the column total.

TCPCT
The total cumulative percentage for each value in a column:

- In reports with BREAK or ACROSS usages, TCPCT shows the cumulative percentage each value in the column is of the column total.
- In all other reports, TCPCT displays the column total.

These aggregations work only on numeric data. Nulls in the column are not included in the result, but undefined values and numeric overflow are evaluated. The format of the result is determined by the edit code of the column.

Four versions of a report follow. The only difference is a result of the aggregation specified on the form for the salary column.

Report 1: SUM SALARY (Total)

NAME	JOB	SUM SALARY
MOLINARE	MGR	22959.20
LU	MGR	20010.00
DANIELS	MGR	19260.25
JONES	MGR	21234.00
		83463.45

Report 2: CSUM SALARY (Cumulative Total)

NAME	JOB	CSUM SALARY
MOLINARE	MGR	22959.20
LU	MGR	42969.20
DANIELS	MGR	62229.45
JONES	MGR	83463.45
		83463.45

Report 3: PCT SALARY (Percentage)

NAME	JOB	PCT SALARY
MOLINARE	MGR	27.51
LU	MGR	23.97
DANIELS	MGR	23.08
JONES	MGR	25.44
		100.00

Report 4: CPCT SALARY (Cumulative Percentage)

NAME	JOB	CPCT SALARY
MOLINARE	MGR	27.51
LU	MGR	51.48

```
   DANIELS    MGR         74.56
   JONES      MGR        100.00
                       ==========
                          100.00
```

Two versions of the same report with a break follow.

The first report uses PCT to show:

- The percentage each salary is of its break group total
- The percentage each break group is of the column total

```
                            PCT
JOB    NAME              SALARY
-----  ---------  ----------
CLERK  JAMES              25.71
       KERMISCH           23.34
       NGAN               23.81
       SNEIDER            27.14
                       ----------
                *          41.61

MGR    HANES              52.95
       SANDERS            47.05
                       ----------
                *          30.91

SALES  PERNAL             52.41
       ROTHMAN            47.59
                       ----------
                *          27.47

                       ==========
                          100.00
```

This report uses TPCT to show:

- The percentage each salary is of the column total
- Subtotals at the breaks

```
                            TPCT
JOB    NAME              SALARY
-----  ---------  ----------
CLERK  JAMES              10.70
       KERMISCH            9.71
       NGAN                9.91
       SNEIDER            11.29
                       ----------
                *          41.61

MGR    HANES              16.37
       SANDERS            14.54
                       ----------
                *          30.91
```

```
SALES  PERNAL          14.40
       ROTHMAN         13.08
                     ----------
                *       27.47

                     ==========
                         100.00
```

Whenever you use a percentage usage code (PCT, CPCT, TPCT, and TCPCT), QMF shows the total percentage as 100. However, occasionally the individual percentages add up to a number slightly higher or lower than 100. That happens because QMF sometimes rounds off the individual percentages when it calculates them.

Charts:

The information on reports for these usage codes is also true for charts. Some of these codes might not be as meaningful in a chart as in a report:

- Cumulative percentages or sums can be difficult to express in a meaningful way graphically.
- Errors that cause undefined data values are considered null values. These values appear as question marks in a report.
- If any of the following symbols are contained in a report to be charted, they are considered null values:
  - Hyphens represent null values in a report
  - Asterisks represent data values too long for the width of the column
  - Greater-than signs (>) represent arithmetic overflow
  - Question marks (?) represent undefined values

BREAK Usage Codes

The BREAK usage codes provide six levels of breaks (or groupings) in a report.

Reports:

When usage is BREAK1, it is a control column for level-1 breaks. Any change in the value of the column causes a break: subtotals are displayed for columns whose usage is one of the aggregation usages, and the level-1 break text is displayed.

Rules for using BREAK:

- To show a break in your report for each change of value in a column, your query must use ORDER BY in SQL. The report then shows exactly as many

breaks as there are different values in the column. Without ORDER BY, the report could show as many breaks as there are lines in the report.

- If the answer set for the query is large, QMF might perform multiple retrievals of data from the database. To ensure that the data is returned in the same order each time, be sure to include an ORDER BY in the query. Similarly, if BREAK is used on a defined column, ensure that multiple evaluations of the column will result in the same results each time.
- More than one column can have a usage of BREAK. The columns are then considered together for the purpose of determining breaks. For example, if a table contains columns for YEAR, MONTH, and DAY, giving each a usage code of BREAK1 causes a level-1 break at every change in date.
- A usage code of BREAK2 controls the column for level-2 breaks. The column is displayed just to the right of a control column for level-1 breaks (if the automatic column reordering option on FORM.OPTIONS is set to YES). There can be up to six levels of breaks. The sequence of break numbers might have gaps. (You can use BREAK2, BREAK3, and BREAK5 in a form without using BREAK1 or BREAK4.)

The BREAK, GROUP, and aggregation usage codes can change the order of the columns on the report (though not on the form). You can tell QMF to automatically reorder the columns in a report. If you do, control columns are moved to the left of the report, and columns using aggregations are moved to the right. For information, see J *Automatically reordering of report columns* (page 234).

By default, columns are not reordered.

You can use BREAK*n*X (*n*=1 to 6) to omit the control column from a report.

Charts:

The BREAK1 usage code can be used to modify the chart. The values in a column with a BREAK usage code are selected for the X-axis. The remaining numeric columns are plotted as Y-axis data, and remaining nonnumeric columns are ignored.

You can use BREAK*n*X (*n*=1 to 6) to omit the control column from a chart. You can also use it to get evenly spaced X-axis points for numeric data.

The QMF-provided chart formats are tailored to handle discrete versus continuous data.

CALCid Usage Code

Reports:

The CALC*id* usage code activates the evaluation of the calculation expression in FORM.CALC whose ID equals *id* for group, break, or final column summaries in the report. The result is edited according to the edit code specified on FORM.CALC and the width given on FORM.COLUMNS.

When CALC*id* is used as a usage code, the calculation is applied to the last row of data. If the column value is used in the calculation, only the last row of data is evaluated. This differs from other usage codes in which every row of data is evaluated.

GROUP Usage Code

Reports:

The GROUP usage code displays only one line of summary data for each set of values in the column. The summary line can display only values that are the same for each member of the group, such as the value in a control column, or the results of columns whose usage is one of the aggregations.

When you want a report to show a summary line for each group of values in a column, use a query that includes the GROUP BY and ORDER BY SQL clauses. GROUP BY accumulates the results of the query by group; ORDER BY orders the groups. The report then shows exactly as many summary lines as there are different values in the column. Without ORDER BY in the query, the report could show as many summary lines as there are lines in the report.

Using GROUP BY and ORDER BY can also improve the performance of a query.

Rules for using GROUP:

- The query that selects the data must use ORDER BY in SQL. Without ORDER BY, the report can produce unexpected results.
- More than one column can have usage GROUP. If so, a change in value in *any* column starts a new group. With two usage codes of GROUP, the report could have many more lines of grouped values.
- The report runs but omits all columns with blank usages if all the following are true:
  - One or more columns in a report has usage GROUP
  - Any other column has an aggregation usage
  - Any remaining columns have blank usages
- If any column has usage GROUP and all other columns have blank usage codes, the report omits the column containing the GROUP usage.

Charts:

CALCid Usage Code

The effect of GROUP as it is used to format a report is similar to its effect on a chart.

OMIT Usage Code

Reports and charts: If the usage code is OMIT, the column and its values are excluded from the tabular report or chart. The values in the column can still appear in the report by use of form variables (such as &*n*).

Date and Time Usage Codes

Arithmetic functions *cannot* be specified for DATE, TIME, and TIMESTAMP values.

Usage codes allowed with DATE, TIME, and TIMESTAMP values:

ACROSS
GROUP

BREAKn (n=1,2,...,6)
LAST

BREAK*n*X (n=1,2,...6)
MAXIMUM

COUNT
MINIMUM

FIRST OMIT

Usage codes not allowed with DATE, TIME, and TIMESTAMP values:

AVERAGE
STDEV

CPCT SUM

CSUM TCPCT

PCT TPCT

Edit Codes

Edit codes determine the formatting of character, graphic, numeric, and, for installations that support it, date and time data. For information on the effect edit codes have on defined columns, see “Edit Codes and Data Types” on page 213.

Edit Codes for Character Data

Use CW, CT, and CDx edit codes with DATE, TIME, and TIMESTAMP values to allow column wrapping.

C Makes no change in the display of a value.

CW Makes no change in the display of a value, but if the value cannot fit on one line in the column, tells QMF to wrap the text according to the width of the column. Instead of cutting off the data at the end of the column, QMF puts as much data as it can on a line in the column, and then continues wrapping the data on the next line in the column.

Data in column-wrapped columns (CW, CT, CD, XW, and BW edit codes) is always aligned using default alignment. (Alignment for headings in column wrapped columns can be modified.) LEFT, CENTER, and RIGHT alignment are ignored for these edit codes. (See "Column Alignment" on page 210.)

If your installation uses DBCS data, you can use the CW edit code on columns of mixed double-byte and single-byte character data. The minimum width of such a column is 4.

The following examples show a report before and after the width of the LOCATION column is reduced and its edit code changed to CW.

Before column wrapping:

```
DEPTNAME        LOCATION
--------------  -------------
HEAD OFFICE     NEW YORK
PACIFIC         SAN FRANCISCO
```

After column wrapping:

```
DEPTNAME        LOCAT
--------------  -----
HEAD OFFICE     NEW Y
                ORK
PACIFIC         SAN F
                RANCI
                SCO
```

CT Makes no change in the display of a value, but if the value cannot fit on one line in the column, tells QMF to wrap the column according to the text in the column. Instead of cutting off the data at the end of the column, QMF fits as much data as possible on a line, interrupts the line when it finds a blank, and continues wrapping the data on the next line. If a string of data is too long to fit in the column and does not contain a blank, QMF wraps the data by width until it finds a blank and can continue wrapping by text.

If your installation uses DBCS data, you can use the CT edit code on columns of mixed double-byte and single-byte character data. QMF interrupts the line when it finds an SBCS blank. The minimum width of such a column is 4.

The following examples show a report before and after the width of the LOCATION column is reduced and its edit code changed to CT.

Before column wrapping:

```
DEPTNAME        LOCATION
--------------  -------------
HEAD OFFICE     NEW YORK
PACIFIC         SAN FRANCISCO
```

After column wrapping:

```
DEPTNAME        LOCAT
--------------  -----
HEAD OFFICE     NEW
                YORK
PACIFIC         SAN
                FRANC
                ISCO
```

CDx Tells QMF to wrap the column according to a delimiter in the text. QMF begins a new line in the column each time it sees a special delimiter in the text. For this edit code, replace the x with the special delimiter. It can be any character, including a blank, and does not appear in the output.

If your installation uses DBCS data, you can use the CDx edit code on columns of mixed double-byte and single-byte character data. The minimum width of such a column is 4, and the delimiter must be outside of the DBCS string.

If a string of data is too long to fit in the column and does not contain a delimiter, QMF wraps the data by width until it finds a delimiter and can continue wrapping by it. If a string of data contains multiple successive delimiters, QMF shows a blank line for each one after the first. For example, if the data contains two delimiters, QMF begins a new line when it gets to the first delimiter, skips a line when it gets to the second delimiter, and then continues wrapping the output.

The following examples show a report before and after the width of the LOCATION column is reduced and its edit code changed to CD&.

Before column wrapping:

```
DEPTNAME        LOCATION
--------------  -------------
HEAD OFFICE     NEW YORK
PACIFIC         SAN FRANCISCO
```

After column wrapping:

```
DEPTNAME        LOCATION
--------------  ---------
HEAD OFFICE     NEW
                YORK
PACIFIC         SAN
                FRANCISCO
```

X Formats data as a series of hexadecimal characters.

XW Formats data as a series of hexadecimal characters. Column wrapping for XW follows the same rule as for CW.

B Formats data as a series of 0's and 1's.

BW Formats data as a series of 0's and 1's. Column wrapping for BW follows the same rule as for CW.

When you use edit codes CW, CT, CD, XW, and BW, column wrapping is only performed when tabular data is displayed or printed. A reference to &*n* in a text line only displays the first line of the wrapped data.

Edit Codes for Graphic Data

G Makes no change in the display of a value.

GW Makes no change in the display of a value, but if the value cannot fit on one line in the column, tells QMF to wrap the text according to the width of the column. Instead of cutting off the data at the end of the column, QMF puts as much data as it can on a line in the column, and then continues wrapping the data on the next line in the column.

Edit Codes for Numeric Data

E<Z> Displays numbers in scientific notation. For example, with this code, the number -1234.56789 would display as -1.234E+03. **E** is used on the default form for columns with data type FLOAT.

QMF shows up to 17 significant digits when editing floating point data, or up to 34 significant digits when editing extended floating point data, even if the width of the column can accommodate more. The number of significant digits is less for other data types.

Edit code **Z** in the second position suppresses zero values.

D<Z><C>, I<Z>, J<Z>, K<Z>, L<Z>, and P<Z>
Display numbers in decimal notation, with different combinations of leading zeros, minus signs for negative numbers, thousands separators, currency symbols, and percent signs as shown in Table 16 on page 264.

Each code can be followed by a number (from 0 to 99) that tells how many places to allow after the decimal point. Numbers with more places after the decimal are rounded; numbers with fewer places are padded with zeros.

On the default form, **L** is used for all columns with numeric data types other than FLOAT. The number of decimal places used is the same as in the column definition.

You might notice small variances for a value when different edit codes are applied to it. For example, the value 0.068124999 displays as 0.068125 using an edit code of L6. However, using an edit code of L5 results in 0.06812. In this case, the digit 2 is not rounded to 3 because the following digit in the original number is less than five.

Edit code **Z** in the second position suppresses zero values. An optional edit code **C** in the second or third position displays the user-defined currency symbol instead of the standard currency symbol. You can define a currency symbol by using the global variable DSQDC_CURRENCY. If you use both Z and C, C must follow Z.

Table 16, following, shows what edit codes D, I, J, K, L, and P provide, and how each formats the number -1234567.885. The display assumes that:

- WIDTH is 15.
- The value of DECIMAL in the profile is PERIOD. (The characters used for the thousands separators and the decimal point depend on that value.)

Table 16. Attributes and Examples of Decimal Edit Codes

Edit Code	Leading Zeros	Negative Sign	Thousands Separators	Currency Symbol	Percent Sign	Example
D2	N	Y	Y	Y	N	-$1,234,567.89
DC2	N	Y	Y	Y	N	-DM1,234,567.89
I2	Y	Y	N	N	N	-00001234567.89
J2	Y	N	N	N	N	000001234567.89
K2	N	Y	Y	N	N	-1,234,567.89
L2	N	Y	N	N	N	-1234567.89
P2	N	Y	Y	N	Y	-1,234,567.89%

Edit Codes for Date Data

In the following edit codes, **x** represents the character to be used as a delimiter between date values. It can be any special character, including blank, but not letters or numbers.

Four-Digit Year:

TDYx	Year first	YYYYxMMxDD
TDMx	Month first	MMxDDxYYYY
TDDx	Day first	DDxMMxYYYY

Abbreviated Two-Digit Year:

TDYAx	Year first	YYxMMxDD
TDMAx	Month first	MMxDDxYY
TDDAx	Day first	DDxMMxYY

Alternative Date Format:

TDL Locally defined. See your QMF administrator for format information.

Date edit code examples: The examples in Table 17, following, show the date July 17, 1989, formatted with various date edit codes.

Table 17. Date Edit Code Examples

Edit Code	Format	Notes
TDD.	17.07.1989	European format
TDY-	1989-07-17	International Standards Organization (ISO) and Japanese Industrial Standard (JIS) formats
TDM/	07/17/1989	USA format
TDD-	17-07-1989	Four-digit year, day first, delimiter: dash (-)
TDDA/	17/07/89	Two-digit year, day first, delimiter: slash (/)
TDDA.	17.07.89	Two-digit year, day first, delimiter: period (.)
TDDA-	17-07-89	Two-digit year, day first, delimiter: dash (-)
TDDA	17 07 89	Two-digit year, day first, delimiter: blank ( )
TDMA/	07/17/89	Two-digit year, month first, delimiter: slash (/)
TDMA-	07-17-89	Two-digit year, month first, delimiter: dash (-)
TDYA/	89/07/17	Two-digit year, year first, delimiter: slash (/)

Edit Codes for Time Data

In Table 18, following, **x** represents the character to be used as a delimiter between time values. It can be any special character, including blank, but not letters or numbers.

Table 18. Clock Format Edit Codes

Edit Code	Format	Notes
TTSx	HHxMMxSS	24-hour clock, including seconds
TTCx	HHxMMxSS	12-hour clock, including seconds

Table 18. Clock Format Edit Codes (continued)

Edit Code	Format	Notes
TTAx	HHxMM	Abbreviated (no seconds)
TTAN	HHMM	Abbreviated (no seconds, no delimiter)
TTUx	HHxMM AM HHxMM PM	USA format
TTL	Locally defined.	See your QMF administrator for format information

Time edit code examples

The examples in Table 19, following, show the time, 1:25:10 PM, formatted with various time edit codes.

Table 19. Time Format Edit Codes

Edit Code	Format	Notes
TTS.	13.25.10	ISO, European formats
TTS:	13:25:10	JIS format
TTU:	01:25 PM	USA format
TTS,	13,25,10	Hours, minutes, seconds (24 hr.), delimiter: comma (,)
TTC:	01:25:10	Hours, minutes, seconds (12 hr.), delimiter: colon (:)
TTA.	13.25	Hours, minutes (24 hr.), delimiter: period (.)
TTA,	13,25	Hours, minutes (24 hr.), delimiter: comma (,)
TTAN	1325	Hours, minutes (24 hr.), no delimiter

Edit Codes for Timestamp Data

The timestamp is a seven-part value designating date and time, including microseconds. There is only one edit code (TSI) for the timestamp data type. The TSI edit code can only be used with columns that have a timestamp data type.

TSI *yyyy-mm-dd-hh.mm.ss.nnnnnn*

yyyy Four-digit value representing the year

mm Two-digit value representing the month

dd Two-digit value representing the day

hh Two-digit value representing the hour

mm Two-digit value representing the minutes

ss Two-digit value representing the seconds

nnnnnn
Six-digit value representing the number of microseconds

The timestamp value:
1991-12-29-23.25.15.123000

Formatted with the TSI edit code:
1991-12-29-23.25.15.123000

User-Defined Edit Codes

Additional edit codes, U*xxxx* and V*xxxx*, are available for special purposes. *xxxx* can be any combination of characters, excluding embedded blanks or nulls. See your QMF administrator for the user-edit codes available to you and the type of data each supports. See *Managing QMF* for your environment for more information about user edit codes.

Considerations for Aggregation Functions and Edit Codes

QMF calculates the result of an aggregation function based on the actual values stored in the database table, not on the values resulting from the edit code for a column. To obtain the aggregation result using the values resulting from the edit code for a column, you must use an alternative method such as defining a new column, and then using a REXX function.

For example:

1. Create and save the following query, naming it Q1:
   ```
   SELECT 10.5 from Q.ORG
   ```
2. Issue the commandRUN Q1 (ROW 2. The report appears as follows:
   ```
     COL1
   ------
     10.5
     10.5
   ```
3. Issue the command SH F. COL.
4. Position the cursor under COL1, and press the Insert function key.
5. Type COLNEW under COLUMN HEADING, SUM under USAGE for both COL1 and COLNEW, and change the edit code for COLNEW to L as shown below:
   ```
   FORM.COLUMNS                                          MODIFIED

                        Total Width of Report Columns: 20
    NUM COLUMN HEADING                          USAGE   INDENT WIDTH EDIT  SEQ
    --- ------------------------------------- ------- ------ ----- ----- ---
      1 COL1                                    SUM     2      6     L1    1
      2 COLNEW                                  SUM     2      10    L     1
        *** END ***
   ```

6. Position the cursor under COLNEW, and press the Specify function key.
7. Choose Definition, and then press Enter.
8. Type the following REXX expression, and then press Enter:
   ```
   format(&1,5,0)
   ```
9. Press F12 to cancel the Specify window.
10. Press the Report function key to display the following report:

```
  COL1      COLNEW
------  ----------
  10.5          11
  10.5          11
======  ==========
  21.0          22
```

Note that COLNEW has rounded values for each row and that the sum is the sum of the rounded values.

Variables Used in Forms

You can use global variables (both user-defined and QMF-supplied) and form variables in QMF forms. A variable can replace a string of text or a numeric value. You can assign different values to the variable to produce different reports without changing the form.

Single or double quotation marks do not affect variables used in the form.

Global variables in forms make it possible for multiple queries to share the same form. For example, using the SET GLOBAL command, you can set a string of text such as *Annual Report for 1993* to a variable *&ann* and use it in a form. (See "SET GLOBAL" on page 113.) You can use the SHOW GLOBAL command to display some or all of the global variables available.

Normally, QMF removes trailing blanks from character values for substitution variables. For numeric values, leading blanks are removed. To retain leading or trailing blanks in substitution variables in the report, append **_B** to any variable on a form panel. For example: &3_B. This special syntax is meaningful only for substitution variables in the form panels. It does not apply to substitution variables used in queries or procedures, or to the variables &ROW, &DATE, &TIME, and &PAGE.

QMF supplies variables called *form variables* that return system information or information about your report. The form variables are:

&ROW	&COUNT	&DATE	&CALC*id*
&TIME	&*n*	&PAGE	&*an*

These variables are defined in the context of the form panel they are entered on and where they appear in the report. They are discussed (if applicable) in the individual sections for each form panel.

Table 20 shows which variables are allowed on the various form panels.

Table 20. Variables Allowed on Form Panels

	F.PAGE		F.BREAK*n*		F.CALC	F.COLUMNS	F.CONDITIONS	F.DETAIL		F.FINAL
Head	**Foot**	**Head**	**Foot**		**Column Definition**		**Head**	**Block**		
&ROW	x	x	x	x	x	x	x	x	x	x
&DATE	x	x	x	x	x	x	x	x	x	x
&TIME	x	x	x	x	x	x	x	x	x	x
&PAGE	x	x	x	x	x			x	x	x
&COUNT				x	x				x	x
&CALC*id*				x					x	x
&*n*	x	x	x	x	x	x	x	x	x	x
&*an*				x	x				x	x
Global variables	x	x	x	x	x	x	x	x	x	x

Chapter 4. General Topics

This chapter contains information about:

- Naming Conventions
- Names with Double-byte Characters
- Commas Instead of Decimal Points
- QMF Temporary Storage Areas
- Report Completion and the Incomplete Data Prompt
- Methods of Writing Queries
- Procedures
- Printing QMF Objects
- The Table Editor
- Online Help
- Remote Data Access
- The Governor Interrupt

Naming Conventions

The following rules apply when naming objects saved in the database.

- Names for queries, forms, procedures, tables, and views must be unique. (You cannot have a query and a form with the same name.)
- Names cannot start with a number.
- A name enclosed in double quotation marks can start with any character except a double quotation mark or a blank.
- You can use any character in a QMF object name *except* the following special characters:

  . , ; : < > ( ) | + − * / = & ¬ ' "

  In some non-English single-byte character sets, the not sign (¬) displays as a circumflex (^); the vertical bar (|) displays as an exclamation point (!).
- Avoid using the special characters listed above in a name. If you use any of the special characters in SQL names, you *must* enclose the entire name in double quotation marks ("*name*"). Names enclosed in double quotation marks can contain any characters (including blanks) except a double quotation mark. See your SQL reference for rules for using special characters in SQL names.

- A name cannot be longer than 18 characters. However, a name can be *qualified* by a location identifier of up to 18 characters and may include a user identifier of up to 8 characters. For example, this is a fully qualified name:

  ```
  NEW_YORK.Q.STAFF
  ```

  It specifies a table owned by the NEW_YORK location created by the user Q with the name of STAFF.
- Do not use QMF reserved words for names because, when used in a QMF command, they will never refer to something in the database. The QMF reserved words are:

  ```
  CHART   FORM     QUERY   DATA     TABLE   PROC   REPORT   FORM     PROFILE
  ```
- Do not use SQL reserved words for names. See your SQL reference for a list of reserved words.

Names with Double-byte Characters

If your installation supports double-byte character set (DBCS) data, you can use double-byte characters alone or mixed with single-byte character set (SBCS) data in your names. The following rules apply when using double-byte characters:

- Names with both double-byte and single-byte characters can contain the same single-byte characters described under “Naming Conventions” on page 271.
- You can specify column headings in a form with mixed double-byte and single-byte characters. A heading consisting of double-byte characters only can be up to 19 double-byte characters long.
- Names containing only double-byte characters can contain no more than eight double-byte characters. But a name can be *qualified* by a user identification. The qualifier can contain as many as eight single-byte characters and *cannot* contain double-byte characters.
- If your database specifically supports double-byte characters in table names, all names can contain any double-byte characters.
- If your database does not specifically support DBCS data in table names, all names can contain any double-byte characters *except* those that are represented internally as a double quotation mark (X'7F').

For information on the use and handling of DBCS data, see *Using QMF* .

Commas Instead of Decimal Points

If you use commas instead of decimal points to indicate decimals in the database and a number ends in a comma, the number is interpreted as an integer. For example:

```
RUN PROC (&1=3, is interpreted as: RUN PROC (&1=3
```

If you use commas to indicate decimals in the database, commas used as separators must have a blank after them to distinguish them from decimal indicators.

QMF Temporary Storage Areas

Some objects in QMF are temporary. These temporary objects reside in QMF temporary storage areas. You have to save them or they disappear, either when you exit QMF or when you write something else over them.

When you save the contents of any of these QMF temporary storage areas, they are stored in the database.

There are five QMF temporary storage areas:

QUERY
Holds a query you are writing, recently imported, or recently ran. To display the contents of QUERY, enter SHOW QUERY.

PROC Holds a procedure you are writing, recently imported, or recently ran. To display the contents of PROC, enter SHOW PROC.

PROFILE
Holds your profile. To display the contents of PROFILE, enter SHOW PROFILE.

FORM
Holds an object that specifies how to format data. To display the contents of FORM, enter SHOW FORM.

DATA Holds the data you imported or selected by the last query you ran or displayed. DATA is formatted by FORM to yield a report.

To display the contents of DATA, enter SHOW REPORT. This does not show DATA directly (nothing does); it shows the contents of DATA as formatted by FORM.

To display DATA in chart form using the Interactive Chart Utility (ICU), enter SHOW CHART.

The contents of a QMF temporary storage area are replaced when you do any of the following:

- Import a CICS data queue, TSO data set, or a CMS file into QUERY, PROC, DATA, or FORM.
- Run a query from the database. The query in the database replaces the contents of QUERY in QMF temporary storage.
- Run a procedure from the database. The procedure in the database replaces the contents of PROC in QMF temporary storage. And, if the procedure contains a command to run a query, that query replaces the contents of QUERY.
- Run a query that displays data. The new data replaces the contents of DATA (whether you entered the RUN command on the command line or from a procedure). When you change the contents of DATA, you change the contents of FORM.
- Display a table in the database. The data replaces the contents of the DATA object and changes the FORM object.

Tables in the database, such as Q.STAFF, are permanent. You must be authorized to erase tables from the database.

Report Completion and the Incomplete Data Prompt

When you run a query or display a table or view, QMF retrieves only enough rows from the database to display the report. This allows QMF to display the report as soon as possible, although QMF might need to retrieve more rows to finish the report.

If you do not complete the report (by either resetting the data or scrolling to the bottom of the report), QMF completes it when you request the next operation that involves the database. The following commands cause QMF to complete the report before the command runs.

CONNECT

DISPLAY
tablename (from the database)

DPRE

DRAW
tablename

EDIT TABLE

ERASE

EXPORT
(from the database)

IMPORT
(to the database)

LIST

PRINT
(from the database)

REFRESH
(of a database object list)

RUN (an object in the database)

RUN QUERY
(from the database)

RUN QUERY
(a non-SELECT query)

SAVE (data, form, procedure, or profile)

If the QMF temporary storage area becomes full while QMF completes your report, QMF displays the following Incomplete Data Object prompt.

```
DXYESIR2              INCOMPLETE DATA OBJECT

The temporary storage area does not contain all of the rows of
DATA.  Because there is not enough storage for QMF to capture
all the rows and columns of data, DATA must be RESET or the
current command must be withdrawn.

Do you want to RESET the DATA object?

_ 1.  YES - RESET the DATA object.
  2.  NO  - Do not RESET the DATA object.

_____________________________________________________________
  F1=Help  F12=Cancel
```

YES Removes all the data in QMF temporary storage, so that none of it is available to you. If you are finished with the contents of the DATA object, choose YES.

NO Cancels the command and leaves the DATA object as is.

For information about controlling the capacity of QMF temporary storage, see *Managing QMF*

Changing QMF's Response to Long-Running Queries

Some QMF commands will not run until all the rows of a query are stored in the temporary storage area. If a query is in the process of running, and you issue a new command, QMF's default response is to finish the query, and then run the new command. You can change QMF's response to this condition by setting the DSQEC_RESET_RPT global variable as follows:

```
SET GLOBAL DSQEC_RESET_RPT=n
```

where *n* can be:

0 Reset Report Prompt Panel is not displayed and QMF runs the query.

1 Reset Report Prompt Panel is displayed. This panel prompts the user to stop or continue the query.

2 Reset Report Prompt Panel is not displayed and the query is stopped.

Avoiding Using Nulls as Data When Editing a QMF Object

QMF uses GDDM for its panels, and nulls (X'00') are susceptible to GDDM screen presentation. Therefore, avoid using nulls on QMF panels, such as the Edit Query panel. Instead, use an alternative, such as a constant hex representation or the database HEX function in an SQL query.

For example, to change a byte to a null value (binary zero) in a table named TEST that has a column named FLD1 with a hex value of 03C1549F, run this update statement:

```
UPDATE TEST SET FLD1=X'0300549F' WHERE FLD=X'03C1549F'
```

Now this field can be displayed using the database HEX function:

```
SELECT HEX(FLD1) FROM TEST
```

Methods of Writing Queries

In addition to writing queries in SQL, you can use Prompted Query or Query-by-Example (QBE).

Prompted Query

Prompted Query prompts you step by step to build a query. To start Prompted Query, specify LANGUAGE=PROMPTED on a SET PROFILE or RESET QUERY command.

When you begin working with a new prompted query, QMF displays a dialog panel on the right side of the screen to guide you through creating a query. As you work with the dialog panels, the prompted query is built in the echo area on the left side of the screen.

For detailed scenarios of the process of creating queries with Prompted Query, see *Using QMF*. Online help is also available.

Query-by-Example (QBE)

QBE is a graphic alternative to writing queries in SQL. See *Using QMF* for details about how to use Query-by-Example.

Procedures

You can create a procedure that contains a series of QMF commands and run it with a single RUN command. This is helpful when you are using commands that are too long to enter on the command line. However, use caution when you use system-specific commands within a procedure. For example, if a procedure contains CMS commands and QMF is running in TSO, you cannot run the procedure successfully.

When you run a procedure, the contents of QMF temporary storage areas DATA, FORM, and QUERY change just as with commands entered on the command line.

Because minimum unique abbreviations might change in future releases, you should use the full names for commands, options, and values in procedures (rather than abbreviated names).

You can create either of two types of procedures: procedures with logic or linear procedures. If the first statement of a procedure is a REXX comment, QMF assumes it is a *procedure with logic*. Otherwise QMF assumes it is a *linear procedure*.

Procedures with logic and linear procedures can call each other in any combination. A procedure with logic can run a linear procedure and vice versa. There is no limit on the length of any procedure.

Procedures with Logic

Note to CICS users

Procedures with logic are not available in CICS, as their function depends on REXX.

Procedures with logic let you use the REXX language to perform conditional logic and calculations, build strings, and pass commands back to the host environment.

Procedures with logic have their own REXX variable pool. You can use procedures with logic to get and set QMF global variables. QMF commands in procedures with logic can contain substitution variables.

QMF commands in procedures with logic *must* be in uppercase regardless of your profile setting.

Substitution variables
The value of a substitution variable is found within the QMF command as it is sent back to QMF. It is resolved at the time each command is executed.

It can refer either to a private procedure variable that exists for the duration of the procedure or to a global variable.

Global variables
The value of the global variable is immediately available to the procedure.

Use the GET GLOBAL command to copy a global variable into a variable, or use the SET GLOBAL command to set new global variables.

Return codes and procedure termination
Success or failure of a command is indicated by a return code. You must test the return code and take appropriate action.

You can move to the `ERROR` label whenever a nonzero return code occurs by using the `SIGNAL ON ERROR` statement.

Continuation lines
Indicated by a comma at the end of the previous line. Command keywords and substitution variables cannot span lines.

Comments
Indicated by: /**comment**/

Linear Procedures

Linear procedures can contain:

- Any QMF command
- Comment lines
- Blank lines
- RUN commands that run other procedures or queries
- Substitution variables

When a variable is set using SET GLOBAL in a linear procedure, the value is unavailable to commands in that same procedure because all substitution variables in a linear procedure must be resolved before the procedure is run. You are prompted for any unresolved variables in your procedure. However, the variable is available to any queries or procedures called by the procedure in which it was set.

Substitution variables
QMF scans the entire procedure for substitution variables, and the values are resolved before the procedure is run.

Global variables
Access global variable values in linear procedures by using substitution variables.

After the global variables are set, if you need to reset them, you must code a `RESET GLOBAL` statement at the end of your procedure. Otherwise, the previous set of substitution values will continue to be used.

Return codes and procedure termination
Success or failure of a command is indicated by a return code. If a command is not successful, the procedure ends and the incorrect command is displayed at the top of the procedure area.

Continuation lines
Indicated by a plus sign (+) in column one of the continued line. Command keywords, substitution variables, and comments cannot span lines.

Comments
Indicated by: *--comment*

System Initialization Procedure

When you start QMF, the system initialization procedure runs to configure the QMF session. For more information, see the version of *Installing and Managing QMF* for your platform.

Printing QMF Objects

The rules for printing QMF objects vary depending on the type of object you are printing and the operating system you are using.

Reports, Tables, Profiles, Procedures, SQL Queries, and QBE Queries

- No printer nickname is required for non-GDDM printing.
- To print without GDDM, enter:

```
PRINTER=' '
```

- GDDM gets control only if the nickname is supplied on the PRINT command or in your profile.
- If no nickname is supplied, (PRINTER=' ') output goes to DSQPRINT. If a nickname is used, output goes to GDDM. See *Managing QMF* for your operating system for more information.

Charts

- A valid GDDM printer nickname is required.
- The default printer name in your profile is used if no printer name is supplied.
- Device token must be a valid printer or plotter such as a 3287 printer.
- GDDM Interactive Chart Utility always gets control when the PRINT command is issued. See *Managing QMF* for your operating system for more information.

Prompted Queries and Forms

- A valid GDDM printer nickname is required.
- GDDM always gets control when the PRINT command is issued.
- Output goes to:
  - In TSO and CICS/MVS, the ddname associated with the nickname.
  - In CMS, *xxxxxxxx* ADMLIST or ADMPRINT (where *xxxxxxxx* is the nickname).
  - In CICS/VSE, the transient data queue associated with the nickname.

The Table Editor

The Table Editor provides a convenient method of adding or changing rows in tables. Without writing a query, you can make changes to columns you are authorized to update.

You can add rows to a table, delete rows from a table, or search for and change existing rows in a table.

To access the Table Editor, depending on whether you want to change existing rows or add rows to your table, enter:

```
EDIT tablename (MODE=CHANGE
```

or

```
EDIT tablename (MODE=ADD
```

Use function keys to enter Table Editor commands. A different set of function keys is displayed depending on whether you are in ADD or CHANGE mode.

Additionally, in those modes, when you edit columnar data having a type of VARCHAR, VARGRAPHIC, or LONG VARGRAPHIC, Table Editor automatically strips trailing blanks.

When performing a search, you must ensure that the length of your search string equals the column length, or the database will not find a match. If the length of your data is shorter than the column length, you must pad the search string with wildcards to equal the column length. You can use the underscore (_) wildcard to represent one character, or the percent sign (%) wildcard to represent multiple characters. For example:

- FLD1 is defined as a 5 character field.
- Its value is AB_D, which is 4 characters long and contains the reserved wildcard character "_".
- When doing a search, enter a value that represents all 5 character positions; for example AB_D_, AB_D%, AB_% or AB%. If you enter the actual four character value AB_D, QMF generates:

  ```
  SELECT FLD1 FROM tablename WHERE FLD1 LIKE 'AB_D'
  ```

  The database will not find the match in this case, since FLD1 is a 5 character field. To find the match, you must enter AB_D_ or one of the forms listed previously. For example, with AB_D_, QMF generates:

  ```
  SELECT FLD1 FROM tablename WHERE FLD1 LIKE 'AB_D_'
  ```

  and with AB%, QMF generates:

  ```
  SELECT FLD1 FROM tablename WHERE FLD1 LIKE 'AB%'
  ```

  The database finds the correct row in either case, because the wildcards account for all five character positions required by the database for FLD1.

When you press a function key, a different set of labels appears. For example, you can press a function key labeled SEARCH while in CHANGE mode to look for the rows you want to change. SEARCH displays another set of function keys.

Table 21, following, lists function keys that are displayed on the various panels of the modes indicated.

Table 21. Mode Function Keys

CHANGE Mode	ADD Mode	SEARCH Mode
BACKWARD	ADD	BACKWARD
CANCEL	BACKWARD	CANCEL
CHANGE	CANCEL	CLEAR
DELETE	CLEAR	END

Table 21. Mode Function Keys (continued)

CHANGE Mode	ADD Mode	SEARCH Mode
END	END	FORWARD
FORWARD	FORWARD	HELP
HELP	HELP	PREVIOUS
NEXT	PREVIOUS	SEARCH
REFRESH	SHOW FIELD	SHOW CHANGE
SHOW FIELD		SHOW FIELD
SHOW SEARCH		

In SHOW FIELD, the Enter key closes the panel and saves the information; the Cancel function closes the panel without saving the information.

You can specify either that you want your changes saved every time you press Enter or not until you are finished with all your changes.

You can specify whether you want a chance to change your mind by having a confirmation panel displayed if the change you make could cause unexpected results.

See *Using QMF* for details about how to use the Table Editor. Online help is also available in the Table Editor.

Online Help

There are three general classifications of help in QMF.

Object help
Descriptions of QMF panels

Message help
Explanations of messages generated because of user errors

Field-sensitive help
Information for entry fields on QMF form panels

Object Help

You can press the HELP function key for information any time you are viewing a QMF panel that is not displaying an error message. For example, pressing the Help function key when the QMF Home panel is displayed lets you select topics of general interest and specific information about commands, forms, and all other parts of QMF.

For more information about the Help facility, see "HELP" on page 56.

Message Help

If you make a typing error, a message appears just above the command line. For example:

```
RNU is not a command.
COMMAND ===> RNU ROUTINE123
```

You can correct the command on the command line and press Enter.

If the error isn't clear from the message, press the Help function key or enter the HELP command for more information. If you need even more information, press the More Help function key. Press the Cancel function key when you want to return to your panel.

Field-Sensitive Help

Field-sensitive help provides direct access to online help information for the entry fields on all forms panels. To obtain field-sensitive help, position your cursor in an entry area and press the Help function key.

Remote Data Access

There are two ways to access data at remote locations: using *distributed unit of work* or *remote unit of work*. Remote data access is fully supported in the VM and MVS environments. In the VSE environment, VSE provides DRDA-remote unit of work server functions. Distributed unit of work allows you to access data at a remote location and use it at your current location. Remote unit of work lets you connect to a remote location and access and use data at that location. Additionally, when you make a connection with remote unit of work, you can access data from yet another location and use it at the location you are currently connected to.

Distributed Unit of Work Access (DB2 for OS/390 only)

If your current location is a DB2 for OS/390 database, you can read and update tables and views managed by remote DB2 for OS/390 databases that are part of the communications network defined to your local DB2 for OS/390 database. You cannot access queries, procedures, or forms at a remote location.

In your query, you can specify a remote table or view by using a *three-part name* or an *alias*. A three-part name includes the name of the location where the table exists, the name of the table owner, and the name of the table. The parts are separated by periods:

```
NEW_YORK.JBP.STAMPS
```

An alias is a locally defined name used to refer to a table or view at the same or a remote DB2 for OS/390 database. You can list aliases that are owned by your primary and current DB2 authorization IDs. Authorization to use the table or the view to which the alias refers is checked when you use the alias in queries or QMF commands.

You can access remote tables or views with the following commands:

Command
Restrictions

DISPLAY
Must use TABLE object type

DRAW
Must use TABLE object type

EDIT None

EXPORT
Must use TABLE object type

IMPORT
Must use TABLE object type

PRINT
None

SAVE Must use DATA object type

You can replace a remote table using a SAVE or IMPORT command.

Remote Unit of Work Access

QMF allows you to connect to any of the DB2 or SQL/DS databases within a distributed network. When you connect to a remote location, it becomes your *current location*. These connections can be made between "like" (DB2–DB2) and "unlike" (SQL/DS–DB2) locations. You can establish this connection during QMF initialization (using the DSQSDBNM program parameter of the START command) or from within a QMF session (with the QMF CONNECT command).

When you are connected to a remote location, all SQL statements you issue (except CONNECT) are directed to the database at the remote location for processing. Therefore, you can access data and QMF objects at a remote location in much the same way you would access data and objects at your own location. For example, you can create a table or replace comments on a table at a remote location by first connecting to that location with remote unit of work.

For more information on preparing for remote unit of work, see *Installing and Managing QMF for VM/ESA* or *Installing and Managing QMF for MVS*. For more information on using remote unit of work, see *Using QMF*.

The Governor Interrupt

Your installation can set database resource limits on queries or procedures that you run. If your query or procedure exceeds a time limit or retrieves more rows from the database than the limit set by your installation, processing is interrupted. A panel is displayed that lets you specify whether you want to continue or cancel the query or procedure. In TSO, the elapsed CPU time is shown in seconds.

You can cancel or continue with or without prompting. However, if you continue, the query or procedure can still be canceled by the QMF governor.

The Governor Interrupt display comes from the QMF governor. If your installation has its own governor, your choices might be different. Your information center can provide more information on the limits set by your installation.

Appendix A. QMF Sample Tables

This appendix contains the following tables:

- Q.APPLICANT
- Q.INTERVIEW
- Q.ORG
- Q.PARTS
- Q.PRODUCTS
- Q.PROJECT
- Q.SALES
- Q.STAFF
- Q.SUPPLIER

These tables contain data about fictional applicants, interviews, parts, products, employees, and suppliers of a fictional company.

Q.APPLICANT

This table provides information about people who have applied for jobs with the company. Each row represents an applicant. The columns are as follows:

TEMPID
Temporary identification of the applicant

NAME
Last name of the applicant

ADDRESS
City and state in which the applicant lives

EDLEVEL
Education level of the applicant

COMMENTS
Notes made by the interviewer

TEMPID	NAME	ADDRESS	EDLEVEL	COMMENTS
400	FROMMHERZ	SAN JOSE,CA	12	NO SALES EXPERIENCE
410	JACOBS	POUGHKEEPSIE,NY	16	GOOD CANDIDATE FOR WASHINGTON
420	MONTEZ	DALLAS,TX	13	OFFER SALES POSITION

TEMPID	NAME	ADDRESS	EDLEVEL	COMMENTS
430	RICHOWSKI	TUCSON,AZ	14	CAN'T START WORK UNTIL 12/92
440	REID	ENDICOTT,NY	14	1 YEAR SALES EXPERIENCE
450	JEFFREYS	PHILADELPHIA,PA	12	GOOD CLERICAL BACKGROUND
460	STANLEY	CHICAGO,IL	11	WANTS PART-TIME JOB
470	CASALS	PALO ALTO,CA	14	EXPERIENCED SALESMAN
480	LEEDS	EAST FISHKILL,NY	12	NEEDS INTERVIEW WITH BROWN
490	GASPARD	PARIS,TX	16	WORKED HERE FROM 1/90 TO 6/90

Q.INTERVIEW

This table is for installations that support date/time data. It shows dates and times in ISO format. The format of DATE, TIME, and TIMESTAMP data in your reports depends on the format chosen as your installation's default. It can be modified with the DATE, TIME, and TIMESTAMP edit codes. The columns are as follows:

TEMPID
Temporary identification of the applicant

INTDATE
Date of interview

STARTTIME
Time the interview started

ENDTIME
Time the interview ended

MANAGER
Employee number of the manager who interviewed the applicant

DISP Whether or not the applicant will be hired

LASTNAME
Last name of the applicant

FIRSTNAME
First name of the applicant

TEMPID	INTDATE	STARTTIME	ENDTIME	MANAGER	DISP	LASTNAME	FIRSTNAME
400	1990-02-05	13.30.00	15.12.00	270	NOHIRE	FROMMHERZ	RICHARD
410	1990-02-11	15.00.00	16.18.00	10	HIRE	JACOBS	SUSAN
420	1990-04-07	09.00.00	09.58.00	140	HIRE	MONTEZ	RITA

TEMPID	INTDATE	STARTTIME	ENDTIME	MANAGER	DISP	LASTNAME	FIRSTNAME
430	1990-04-24	10.30.00	11.30.00	290	NOHIRE	RICHOWSKI	JOHN
440	1990-03-13	10.15.00	11.23.00	160	HIRE	REID	CATHY
450	1990-09-19	09.45.00	11.00.00	50	HIRE	JEFFREYS	PAUL
460	1990-10-06	14.45.00	16.22.00	100	HIRE	STANLEY	JOHN
470	1990-02-05	16.30.00	18.00.00	270	HIRE	CASALS	DAVID
480	1990-03-13	13.30.00	14.45.00	160	NOHIRE	LEEDS	DIANE
490	1990-09-30	15.00.00	15.44.00	140	NOHIRE	GASPARD	PIERRE

Q.ORG

This table provides information on the organization of the company. Each row represents a department. The columns are as follows:

DEPTNUMB
Number of the department (must be unique)

DEPTNAME
Descriptive name of the department

MANAGER
Employee number of the manager of the department

DIVISION
Division to which the department belongs

LOCATION
Name of the city in which the department is located

DEPTNUMB	DEPTNAME	MANAGER	DIVISION	LOCATION
10	HEAD OFFICE	160	CORPORATE	NEW YORK
15	NEW ENGLAND	50	EASTERN	BOSTON
20	MID ATLANTIC	10	EASTERN	WASHINGTON
38	SOUTH ATLANTIC	30	EASTERN	ATLANTA
42	GREAT LAKES	100	MIDWEST	CHICAGO
51	PLAINS	140	MIDWEST	DALLAS
66	PACIFIC	270	WESTERN	SAN FRANCISCO
84	MOUNTAIN	290	WESTERN	DENVER

Q.PARTS

This table provides information about parts. The columns are as follows:

SUPPNO
Number of the supplier

PARTNAME
Name of the part

PRODUCT
Product for which the part is needed

PRODNO
Number of the product

PROJNO
Number of the project

SUPPNO	PARTNAME	PRODUCT	PRODNO	PROJNO
1100P	PLASTIC	RELAY	30	1501
1100P	STEEL	WRENCHSET	509	1520
1200S	WIRE	GENERATOR	10	1401
1200S	BEARINGS	MOTOR	50	1402
1300S	COPPER	RELAY	30	1501
1300S	BLADES	SAW	205	1510
1400P	MAGNETS	GENERATOR	10	1409
1400P	VALVES	MOTOR	50	1407
1400P	OIL	GEAR	160	1405

Q.PRODUCTS

This table provides information about a few products and their prices. The columns are as follows:

PRODNUM
Number of the product

PRODNAME
Descriptive name of the product

PRODGRP
General type of product

PRODPRICE
Price of the product

PRODNUM	PRODNAME	PRODGRP	PRODPRICE
10	GENERATOR	ELECTRICAL	45.75
505	SCREWDRIVER	TOOL	3.70
101	SHAFT	MECHANICAL	8.65
20	SWITCH	ELECTRICAL	2.60
30	RELAY	ELECTRICAL	7.55
40	SOCKET	ELECTRICAL	1.40
50	MOTOR	ELECTRICAL	35.80
150	CAM	MECHANICAL	1.15
160	GEAR	MECHANICAL	9.65
190	BUSHING	MECHANICAL	5.90
205	SAW	TOOL	18.90
330	HAMMER	TOOL	9.35
450	CHISEL	TOOL	7.75
509	WRENCHSET	TOOL	25.90

Q.PROJECT

This table provides information about project schedules. The columns are as follows:

PROJNO
Number of the project (must be unique)

PRODNUM
Number of the product

DEPT Number of the department responsible for the project

STARTD
Date the project is to start

ENDD
Date the project is to end

TIMESTAMP
Year, month, day, and time of the report

This table is for installations that support date/time data. It shows dates and times in ISO format. This format is an arbitrary choice. The table you see depends on the choice made by your installation.

PROJNO	PRODNUM	DEPT	STARTD	ENDD	TIMESTAMP
1401	10	20	1996-01-01	1998-03-31	1994-12-18-10.14.44.000001
1402	50	66	1996-01-30	1997-06-30	1994-12-18-10.15.01.999998
1403	150	51	1996-02-02	1999-05-29	1994-12-18-10.22.23.000001
1404	190	38	1997-01-04	1999-06-30	1994-12-18-10.25.43.999999
1405	160	15	1997-04-29	1999-10-30	1995-12-31-14.23.00.999999
1406	20	20	1997-07-11	1998-12-31	1996-01-05-13.31.18.009999
1407	50	42	1997-12-12	2000-06-15	1996-01-05-13.42.27.000000
1408	30	42	1999-03-13	2000-09-30	1996-01-05-13.44.16.999999
1409	10	66	1998-06-15	1999-12-31	1996-03-13-09.12.57.149572
1410	190	10	1998-09-29	2000-03-31	1996-03-13-12.18.23.402917
1501	30	51	1999-01-04	1999-12-31	1996-03-13-12.22.14.201966
1502	150	38	1999-03-01	2000-07-17	1996-03-13-13.17.48.948276

Q.STAFF

This table provides data on the employees. The columns are as follows:

ID Employee serial number (must be unique)

NAME
Name of the employee

DEPT Department number of the employee

JOB Classification of the employee's job

YEARS
Number of years the employee has worked for the company

SALARY
Employee's annual salary in dollars and cents

COMM
Employee's commission in dollars and cents

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
10	SANDERS	20	MGR	7	18357.50	-
20	PERNAL	20	SALES	8	18171.25	612.45
30	MARENGHI	38	MGR	5	17506.75	-
40	O'BRIEN	38	SALES	6	18006.00	846.55
50	HANES	15	MGR	10	20659.80	-
60	QUIGLEY	38	SALES	-	16808.30	650.25
70	ROTHMAN	15	SALES	7	16502.83	1152.00
80	JAMES	20	CLERK	-	13504.60	128.20
90	KOONITZ	42	SALES	6	18001.75	1386.70

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
100	PLOTZ	42	MGR	7	18352.80	-
110	NGAN	15	CLERK	5	12508.20	206.60
120	NAUGHTON	38	CLERK	-	12954.75	180.00
130	YAMAGUCHI	42	CLERK	6	10505.90	75.60
140	FRAYE	51	MGR	6	21150.00	-
150	WILLIAMS	51	SALES	6	19456.50	637.65
160	MOLINARE	10	MGR	7	22959.20	-
170	KERMISCH	15	CLERK	4	12258.50	110.10
180	ABRAHAMS	38	CLERK	3	12009.75	236.50
190	SNEIDER	20	CLERK	8	14252.75	126.50
200	SCOUTTEN	42	CLERK	-	11508.60	84.20
210	LU	10	MGR	10	20010.00	-
220	SMITH	51	SALES	7	17654.50	992.80
230	LUNDQUIST	51	CLERK	3	13369.80	189.65
240	DANIELS	10	MGR	5	19260.25	-
250	WHEELER	51	CLERK	6	14460.00	513.30
260	JONES	10	MGR	12	21234.00	-
270	LEA	66	MGR	9	18555.50	-
280	WILSON	66	SALES	9	18674.50	811.50
290	QUILL	84	MGR	10	19818.00	-
300	DAVIS	84	SALES	5	15454.50	806.10
310	GRAHAM	66	SALES	13	21000.00	200.30
320	GONZALES	66	SALES	4	16858.20	844.00
330	BURKE	66	CLERK	1	10988.00	55.50
340	EDWARDS	84	SALES	7	17844.00	1285.00
350	GAFNEY	84	CLERK	5	13030.50	188.00

Q.SUPPLIER

This table provides data on the suppliers of a company. The columns are as follows:

ACCTNO
The account number of the company

COMPANY
The name of the company

STREET
The street address of the company

CITY The city in which the company is located

STATE
The state in which the company is located

ZIP The company's zip code

NOTES
Information about the company

The form for this table specifies a width of 30 and an edit code of CT for the NOTES column.

ACCTNO	COMPANY	STREET	CITY	STATE	ZIP	NOTES
1100P	WESTCO, INC.	1900 115TH ST.	EMERYVILLE	CA	16600	THIS COMPANY HAS A STRONG HISTORY OF ON-TIME DELIVERY. WESTCO IS GROWING QUICKLY.
1200S	MAJOR ELECTRICS	4250 BENSON ST.	DALLAS	TX	87050	MAJOR ELECTRICS DECLARED BANKRUPTCY IN 1987, BUT HAS RECOVERED. FORESEE NO FURTHER PROBLEMS.
1300S	FRANKLIN, INC.	40025 EASTLAND	DOVER	DE	99000	DUE TO ITS LOCATION ON EASTERN SEABOARD, FRANKLIN HAS EXCELLENT TRANSPORTATION FACILITIES.
1400P	MOTORWORKS, INC.	19503 BESWICK	JOLIET	IL	12000	PROXIMITY TO CHICAGO ENSURES GOOD TRANSPORTATION, BOTH BY RAIL AND TRUCK. A RELIABLE SUPPLIER.

Appendix B. QMF Global Variable Tables

QMF provides many variables for use in your applications. In Version 3, QMF introduced the current naming convention for the callable interface. The corresponding command interface variable names are still valid.

The callable interface global variable names can be up to 18 characters long. Callable interface users can use either the old (eight character) names or the new (18 character) names; however, using the new names is recommended. Command interface users *must* use the old names.

The new naming convention is **DSQcc_xxxxxxxxxxxx**

cc Can be any one of the following category identifiers:

- **AP** Profile-related state information
- **AO** Other (not profile-related) state information
- **CM** Information about the message produced by the previous command
- **CP** Information about the Table Editor
- **DC** Controls how QMF displays information on the screen
- **EC** Controls how QMF executes commands and procedures
- **QC** Variables produced by a CONVERT QUERY option
- **QM** RUN QUERY error message information
- **QW** Variables unique to QMF for Windows

_ An underscore character

xxxxxxxxxxx
A descriptive name up to 12 characters long

Beginning with Version 3.3, QMF provides a special procedure named Q.SYSTEM_INI that allows you to customize global variables at initialization. See the QMF *Installing and Managing* book for your operating system for more information.

DSQ Global Variables for Profile-Related State Information

None of these global variables can be modified by the SET GLOBAL command.

Callable Interface Variable Name	Command Interface Variable Name	Length	Description
DSQAP_CASE	DSQAPCAS	01	CASE parameter. Values can be: **1** for UPPER **2** for MIXED **3** for STRING
DSQAP_CONFIRM	DSQAPRMP	01	CONFIRM parameter. Values can be: **0** for NO **1** for YES
DSQAP_DECIMAL	DSQAPDEC	01	DECIMAL parameter. Values can be: **1** for PERIOD **2** for COMMA **3** for FRENCH
DSQAP_LENGTH	DSQAPLEN	18	LENGTH parameter. Its value is that of the parameter. ('1' through '999' or 'CONT')
DSQAP_PFKEY_TABLE	DSQAPPFK	31	Name of the function keys table
DSQAP_PRINTER	DSQAPPRT	08	PRINTER parameter. Values can be: A nickname for a GDDM printer. Blanks for the printer associated with DSQPRINT.

Callable Interface Variable Name	Command Interface Variable Name	Length	Description
DSQAP_QUERY_LANG	DSQAPLNG	01	LANGUAGE parameter. Values can be: **1** for SQL **2** for QBE **3** for PROMPTED
DSQAP_QUERY_MODEL	DSQAMODP	01	MODEL parameter. Value can be '1' for RELATIONAL
DSQAP_RESOURC_GRP	DSQAPGRP	16	RESOURCE GROUP parameter.
DSQAP_SPACE	DSQAPSPC	50	SPACE parameter. Its value is that of the parameter.
DSQAP_SYNONYM_TBL	DSQAPSYN	31	SYNONYMS parameter.
DSQAP_TRACE	DSQAPTRC	18	TRACE parameter. Values can be: **ALL** (maximum tracing) **NONE** (minimum tracing) Specifications for individual QMF components (Example: A2L2C1)
DSQAP_WIDTH	DSQAPWID	18	WIDTH parameter. Its value is that of the parameter. ('22' through '999')

DSQ Global Variables for State Information Not Related to the Profile

None of these global variables can be modified by the SET GLOBAL command.

Callable Interface Variable Name	Command Interface Variable Name	Length	Description
DSQAO_APPL_TRACE	DSQATRAC	01	Application trace level. Values can be: **0** for level A0 **1** for level A1 **2** for level A2
DSQAO_ATTENTION	DSQCATTN	01	User attention flag.

QMF Global Variables

Callable Interface Variable Name	Command Interface Variable Name	Length	Description
DSQAO_BATCH	DSQABATC	01	Batch or interactive mode. Value will be: **1** for an interactive session. **2** for a batch-mode session.
DSQAO_CONNECT_ID	DSQAAUTH	08	The user ID used to connect to the database. (This is the user ID under which work is done.)
DSQAO_CONNECT_LOC	none	18	The location name of the database to which the user is currently connected. The name is 18 characters (padded to the right with blanks, if necessary).
DSQAO_CURSOR_OPEN	DSQACRSR	01	Database cursor status. Values can be: **1** if the cursor is open. **2** if the cursor is closed.
DSQAO_DB_MANAGER	DSQADBMG	01	Database manager. Values can be: **1** for DB2 for VM/ESA or VSE/ESA **2** for DB2 for MVS/ESA **3** for workstation database servers
DSQAO_DBCS	DSQADBCS	01	DBCS support status. Values can be: **1** for DBCS support. **2** for no DBCS support.

Callable Interface Variable Name	Command Interface Variable Name	Length	Description
DSQAO_FORM_PANEL	DSQASUBP	02	Current form panel. Values can be: **1** for FORM.MAIN **2** for FORM.COLUMN **3** for FORM.PAGE **4** for FORM.FINAL **5** for FORM.BREAK1 **6** for FORM.BREAK2 **7** for FORM.BREAK3 **8** for FORM.BREAK4 **9** for FORM.BREAK5 **10** for FORM.BREAK6 **11** for FORM.OPTIONS **12** for FORM.CALC **13** for FORM.DETAIL **14** for FORM.CONDITIONS A blank value means the form does not exist in QMF temporary storage.
DSQAO_INTERACT	DSQAIACT	01	Setting of interact flag. Values can be: **0** for no interactive execution. **1** for interactive execution allowed.
DSQAO_LOCAL_DB2	none	18	The location name of the local DB2 database. This is the location name for the subsystem named in the variable DSQAO_SUBSYS_ID. In a remote unit of work environment, DSQ_LOCAL_DB2 is the name of the application requester. The name is 16 characters (padded to the right with blanks, if necessary). This field is blank if QMF is running in the VM or VSE environment.

QMF Global Variables

Callable Interface Variable Name	Command Interface Variable Name	Length	Description
DSQAO_LOCATION	DSQAITLO	16	Location name of the current object, if any. This value is applicable only if a three-part name was used.
DSQAO_NLF_LANG	DSQALANG	01	National language of user. For the English language environment, this is 'E'.
DSQAO_NUM_FETCHED	DSQAROWS	16	Fetched data rows. Contains '0' when the DATA object is empty.
DSQAO_OBJ_NAME	DSQAITMN	18	The name of the table (contained in a report), query, procedure, or form shown on the currently displayed panel. If the current panel does not display an object, or if the displayed object has no name, this variable contains blanks.
DSQAO_OBJ_OWNER	DSQAITMO	08	The owner of the table (contained in a report), query, procedure, or form shown on the currently displayed panel. If the current panel does not display an object, or if the displayed object has no owner, this variable contains blanks.
DSQAO_PANEL_TYPE	DSQAITEM	01	Type of current panel. Values can be: **1** for HOME **2** for QUERY **3** for REPORT **4** for FORM **5** for PROC **6** for PROFILE **7** for CHART **8** for LIST **9** for Table Editor **A** for GLOBALS
DSQAO_QMF_RELEASE	DSQAREVN	02	Numeric release number of QMF. For QMF Version 6, this is '11'.

Callable Interface Variable Name	Command Interface Variable Name	Length	Description
DSQAO_QMF_VER_RLS	DSQAQMF	10	Version and release of QMF. For QMF Version 6 this is 'QMF V6'.
DSQAO_QRY_SUBTYPE	DSQASUBI	01	Query subtype. Values can be: **1** for a subtype of SQL **2** for a subtype of QBE **3** for a subtype of PROMPTED Blank means the current panel is not QUERY.
DSQAO_QUERY_MODEL	DSQAMODL	01	Model of current query. Value can be '1' for RELATIONAL
DSQAO_SAME_CMD	DSQACMDM	01	Values can be: **0** if the two commands aren't the same. **1** if the two commands are the same.
DSQAO_SUBSYS_ID	none	04	If QMF is running in TSO, this is the ID of the local DB2 subsystem to which QMF is attached. If you specify a value for the DSQSUBS program parameter from CMS or CICS, this global variable contains that value. This happens because the parameter is tolerated and the value is not processed; that is, the value is placed in the global variable field and nothing is done with it. This logic permits the same EXEC to be used in multiple environments.

QMF Global Variables

Callable Interface Variable Name	Command Interface Variable Name	Length	Description
DSQAO_SYSTEM_ID	DSQASYST	01	Current operating system. Values can be: **1** for VM/SP **2** for MVS/SP **3** for MVS/XA or MVS/ESA **4** for VM/XA or VM/ESA **5** for CICS
DSQAO_TERMINATE	DSQCSESC	01	QMF termination flag. Values can be: **0** if the session was not marked. **1** if the session was marked.
DSQAO_VARIATION	DSQAVARN	02	Form panel variation number. Blank means FORM.DETAIL is not the current panel.

DSQ Global Variables Associated with CICS

Of the variables in this table, only DSQAP_CICS_PQNAME and DSQAP_CICS_PQTYPE can be modified by the SET GLOBAL command.

When the queue type is TD, the maximum length of the corresponding queue name is 4. For example, if DSQAO_CICS_SQTYPE is TD, the maximum length of DSQAO_CICS_SQNAME is 4.

Callable Interface Variable Name	Command Interface Variable Name	Length	Description
DSQAP_CICS_PQNAME	none	08	Names the CICS data queue to contain the QMF print.

Callable Interface Variable Name	Command Interface Variable Name	Length	Description
DSQAP_CICS_PQTYPE	none	02	Type of CICS storage used to contain the QMF print. **TS** writes the QMF print to a CICS temporary storage queue on an "auxiliary" storage device. This is the default. **TD** writes the QMF print to a CICS transient data queue.
DSQAO_CICS_SQNAME	none	08	Names the CICS data queue to be used as the spill file.
DSQAO_CICS_SQTYPE	none	02	Type of CICS storage used to contain the QMF spill file. **TS** writes the QMF spill file to a CICS temporary storage queue on an "auxiliary" storage device. This is the default. **TD** writes the QMF spill file to a CICS transient data queue.
DSQAO_CICS_TQNAME	none	08	Names the CICS data queue to contain the QMF trace.
DSQAO_CICS_TQTYPE	none	02	Type of CICS storage used to contain the QMF trace. **TS** writes the QMF trace to a CICS temporary storage queue on an "auxiliary" storage device. **TD** writes the QMF trace to a CICS transient data queue. This is the default.

DSQ Global Variables Related to a Message Produced by the Previous Command

None of these global variables can be modified by the SET GLOBAL command.

Callable Interface Variable Name	Command Interface Variable Name	Length	Description
DSQCM_MESSAGE	DSQCIMSG	80	Message text
DSQCM_MSG_HELP	DSQCIMID	08	ID of message help panel
DSQCM_MSG_NUMBER	DSQCIMNO	08	Message number
DSQCM_SUB_TXT_*nn*	DSQCIM*nn*	20	Substitution value *nn*
DSQCM_SUBST_VARS	DSQCIM00	04	Number of substitution variables in the message

DSQ Global Variables Associated with Table Editor

All of these global variables can be modified by the SET GLOBAL command.

If the CONFIRM option of the EDIT TABLE command is NO, the Table Editor suppresses the display of all confirmation panels. If the CONFIRM option is YES, the Table Editor determines which categories of confirmation are enabled by checking the values of the global variables shown in this table.

The Table Editor defaults depend on the SAVE keyword from the EDIT TABLE command:

- When SAVE=IMMEDIATE, the default for each category is to enable.
- When SAVE=END, the default for the DELETE, MODIFY, and END/CANCEL categories is to enable; the default for the ADD and CHANGE categories is to disable.

Callable Interface Variable Name	Command Interface Variable Name	Length	Description
DSQCP_TEADD	none	01	Displays a confirmation panel after an ADD subcommand. Values can be: **0** panel is disabled. **1** panel is enabled. **2** panel is enabled or disabled depending on the Table Editor defaults. This is the default.
DSQCP_TECHG	none	01	Displays a confirmation panel after a CHANGE subcommand. Values can be: **0** panel is disabled. **1** panel is enabled. **2** panel is enabled or disabled depending on the Table Editor defaults. This is the default.
DSQCP_TEDEL	none	01	Displays a confirmation panel after a DELETE subcommand. Values can be: **0** panel is disabled. **1** panel is enabled. **2** panel is enabled or disabled depending on the Table Editor defaults. This is the default.

Callable Interface Variable Name	Command Interface Variable Name	Length	Description
DSQCP_TEDEL	none	01	Displays a confirmation panel after a DELETE subcommand. Values can be: **0** panel is disabled. **1** panel is enabled. **2** panel is enabled or disabled depending on the Table Editor defaults. This is the default.
DSQCP_TEDFLT	none	01	The reserved character used to indicate the default value for a column in the Table Editor. Initially set to a plus sign (+) character.
DSQCP_TEDFLT_DBCS	none	04	The reserved DBCS character used to indicate the default value for a graphic string column in the Table Editor. The value must be a four-byte, mixed string, composed of one DBCS character, preceded by the shift-out character, and followed by the shift-in character. Initially set to a DBCS plus sign (+) character. Note that this global variable is used only in a DBCS environment.

Callable Interface Variable Name	Command Interface Variable Name	Length	Description
DSQCP_TEMOD	none	01	Displays a confirmation panel when the user issues an END subcommand or a CANCEL subcommand to terminate a Table Editor subsession. The panel can appear in several variations, depending on whether or not END or CANCEL was issued, whether modifications were made to the database, and whether the screen contained modified data when END or CANCEL was issued. Values can be: **0** panel is disabled. **1** panel is enabled. **2** panel is enabled or disabled depending on the Table Editor defaults. This is the default.
DSQCP_TENULL	none	01	The reserved character used to indicate the null value for a column in the Table Editor. Initially set to a hyphen (-) character.
DSQCP_TENULL_DBCS	none	04	The reserved DBCS character used to indicate the null value (or, in the context of search criteria, to indicate ignore) for a graphic string column in the Table Editor. The value must be a four-byte, mixed string, composed of one DBCS character, preceded by the shift-out character, and followed by the shift-in character. Initially set to a DBCS hyphen (-) character. Note that this global variable is used only in a DBCS environment.

DSQ Global Variables That Control How Information is Displayed on the Screen

All of these global variables can be modified by the SET GLOBAL command.

Callable Interface Variable Name	Command Interface Variable Name	Length	Description
DSQDC_COST_EST	none	01	Optionally suppress database cost estimate. Values can be: **0** = no—Do not display the cost estimate. **1** = yes—Display the cost estimate. This is the default.
DSQDC_CURRENCY	none	18	The currency symbol used when the DC edit code is specified. The value can be a string with a length from 1 to 18 bytes. For English, the default is the euro currency symbol. The default varies for other languages. In a DBCS environment, this value can be a mixed string of SBCS and DBCS characters. The total length of the mixed string, including the shift-out and shift-in characters, cannot exceed 18 bytes.

Callable Interface Variable Name	Command Interface Variable Name	Length	Description
DSQDC_DISPLAY_RPT	DSQADPAN	01	Display report after RUN QUERY. Values can be: **0** if you don't want QMF to display the resulting report from a RUN query command. This is the default if QMF is started interactively with DSQQMFE or in BATCH mode. Changing this variable when QMF is started in BATCH mode will not cause any QMF screen to display. **1** if you want QMF to automatically display the report. This is the default if QMF is started with the callable interface. This can be overridden with the DSQADPAN program parameter on the START command. This global variable is for applications only. It has no effect when the RUN QUERY command is entered on the command line.

Callable Interface Variable Name	Command Interface Variable Name	Length	Description
DSQDC_LIST_ORDER	none	02	Sets the default sort order for objects in a list of database objects. Values for the first character can be: **1** The list will use the default order **2** The list will be sorted by object owner. **3** The list will be sorted by object name. **4** The list will be sorted by object type. **5** The list will be sorted by date modified. **6** The list will be sorted by date last used. Values for the second character can be: **A** The list will be sorted in ascending order. **D** The list will be sorted in descending order. This variable applies only to objects that are listed as a result of the LIST command. It does not apply to lists produced in other contexts, such as from a Display prompt panel, and it does not apply to lists of tables.

Callable Interface Variable Name	Command Interface Variable Name	Length	Description
DSQDC_SCROLL_AMT	none	04	Sets the scroll amount for QMF panels. Values can be: **Csr** Sets scroll amount to cursor. Depending on whether the user scrolls backward, forward, left, or right, QMF scrolls the line or column where the cursor is positioned to the bottom, top, far left, or far right of the scrollable area. **Half** Sets scroll amount to half the scrollable area. **Page** Sets scroll amount to a full page. This is the default. **n** Sets scroll amount to *n* number of lines or columns. *n* can be any number from 1 to 9999.
DSQDC_SHOW_PANID	DSQCPDSP	01	Display panel IDs on CUA-like panels. Values can be: **0** Suppress panel identifiers. This is the default. **1** Display panel identifiers.

DSQ Global Variables That Control How Commands and Procedures Are Executed

All of these global variables can be modified by the SET GLOBAL command.

Callable Interface Variable Name	Command Interface Variable Name	Length	Description
DSQEC_ALIASES	none	31	View for retrieving lists of table and view aliases when the user requests a list of tables from a DB2 for MVS/ESA location or if the current server is DB2 for MVS/ESA, or a workstation database server.
DSQEC_COLS_LDB2	none	31	View for retrieving column information for a table at the current location, if that location is DB2.
DSQEC_COLS_RDB2	none	31	View for retrieving column information for a table at a remote DB2 location (if it is not the current location).
DSQEC_COLS_SQL	none	31	View for retrieving column information for a table in a DB2 for VM/ESA or VSE/ESA database.
DSQEC_FORM_LANG	none	01	Establishes the default NLF language in a saved or exported form. Values can be: **0** The form will use the presiding NLF language. **1** The form will use English. This is the default.

Callable Interface Variable Name	Command Interface Variable Name	Length	Description
DSQEC_ISOLATION	none	01	Default Query isolation level. Values can be: **0** Isolation level UR, Uncommitted Read. **1** Isolation level CS, Cursor Stability. This is the default. **Attention:** Setting the value to '0' can introduce non-existent data into a QMF report. Do not set the value to '0' if your QMF reports must be free of non-existent data. **Limited support:** For QMF 6 the use of the value '0' is only effective with the following database servers (those supporting the SQL WITH clause): • DB2 for MVS V4 or higher • DB2 for VM/VSE V4 or higher
DSQEC_NLFCMD_LANG	none	01	Set expected NLF language for commands. Values can be: **0** Commands must be in the presiding NLF language. This is the default. **1** Commands must be in English.

Callable Interface Variable Name	Command Interface Variable Name	Length	Description
DSQEC_RERUN_IPROC	none	01	Rerun invocation procedure after the END command. Values can be: **0** Suppress rerun of invocation procedure after the END command. **1** Rerun the invocation procedure after the END command. This is the default. If you start QMF with an invocation procedure, then set this variable to '0', QMF terminates instead of rerunning the procedure.
DSQEC_RESET_RPT	none	31	Determines whether or not QMF prompts the user when an incomplete DATA object in temporary storage appears to be affecting performance. Possible values are: **0** Reset Report Prompt Panel is not displayed and QMF completes the running report. This is the default value. **1** Reset Report Prompt Panel is displayed. This panel prompts the user to complete or reset the currently running report before starting the new command. **2** Reset Report Prompt Panel is not displayed and QMF resets the currently running report.

Callable Interface Variable Name	Command Interface Variable Name	Length	Description
DSQEC_SHARE	none	31	Specifies the default value for the SHARE parameter. The possible values are: **0** Do not share data with other users. **1** Do share data with other users.
DSQEC_TABS_LDB2	none	31	View for retrieving lists of tables and views at the current server, if it is DB2 for MVS/ESA, or a workstation database server.
DSQEC_TABS_RDB2	none	31	View for retrieving lists of tables and views at remote DB2 subsystems.
DSQEC_TABS_SQL	none	31	View for retrieving lists of tables and views for a DB2 for VM/ESA or VSE/ESA database.

DSQ Global Variables That Show Results of CONVERT QUERY

None of these global variables can be modified by the SET GLOBAL command.

Callable Interface Variable Name	Command Interface Variable Name	Length	Description
DSQQC_LENGTH_*nnn*	DSQCL*nnn*	05	Length of converted result *nnn*
DSQQC_QRY_COUNT	DSQCQCNT	03	Number of queries in converted result. Value must always be '1' unless the original query is a QBE I. or U. query.
DSQQC_QRY_LANG	DSQCQLNG	01	Language of converted query. Values can be: **1** for SQL **2** for QBE **3** for prompted

Callable Interface Variable Name	Command Interface Variable Name	Length	Description
DSQQC_QRY_TYPE	DSQCQTYP	not specified	First word in converted results
DSQQC_RESULT_*nnn*	DSQCQ*nnn*	not specified	Converted result *nnn*

DSQ Global Variables That Show RUN QUERY Error Message Information

None of these global variables can be modified by the SET GLOBAL command.

Callable Interface Variable Name	Command Interface Variable Name	Length	Description
DSQQM_MESSAGE	DSQCIQMG	80	Text of query message
DSQQM_MSG_HELP	DSQCIQID	08	ID of message help panel
DSQQM_MSG_NUMBER	DSQCIQNO	08	Message number
DSQQM_SQL_RC	DSQCISQL	16	The SQLCODE from the last command or query.
DSQQM_SQL_STATE	none	05	The SQLSTATE associated with the SQLCODE in DSQQM_SQL_RC, if SQLSTATE is returned by the database manager.
DSQQM_SUB_TXT_*nn*	DSQCIQ*nn*	20	Substitution value *nn*
DSQQM_SUBST_VARS	DSQCIQ00	04	Number of substitution variables

Appendix C. QMF Functions that Require Specific Support

Table 22. These functions require the support of specific database management systems.

Function Supported	DB2 for OS/390	workstation database servers	SQL/DS
Length of query statement	32,765	32,765	8,192
Number of columns in SELECT statement	750	255	255
Import single-precision floating point numbers	X		X
Long fields with LIKE statement	X		X
Database synonyms	X		X
Database aliases for tables or views	X	X	
SAVE=IMMEDIATE option available in Table Editor (Supports CURSOR HOLD	X	X	
Distributed Unit of Work (three-part names)	X		
Remote Unit of Work	X	X	on VSE, requires Version 3 Release 4

QMF Functions Not Available in CICS

The following QMF and QMF-related functions are not available in the CICS/ESA or CICS/MVS environment.

- Command interface
- EDIT PROC
- EDIT QUERY
- Document interface
- BATCH application
- Canceling transactions
- EXTRACT
- ISPF
- DPRE
- Report calculations
- External variables
- LAYOUT application

QMF Functions that Require Specific Support

- Conditional formatting
- Column definition
- Procedures with logic

Appendix D. 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 10594-1785
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.

Any references in this publication to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is as your own risk.

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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

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.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

Advanced Peer-to-Peer Networking
AIX
AIX/6000
CICS
CICS/MVS
CICS/VSE
DATABASE 2
DB2
Distributed Relational Database Architecture
DRDA
DXT
GDDM
IBM
MVS/ESA
MVS/XA
OfficeVision/VM
OS/2
PL/I
PROFS
QMF
SQL/DS
Virtual Machine/Enterprise Systems Architecture
VM/XA
VSE/ESA

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.

Glossary of Terms and Acronyms

This glossary defines terms as they are used throughout the QMF library. If you do not find the term you are looking for, refer to the index in this book, or to the *IBM Dictionary of Computing*.

abend. The abnormal termination of a task.

ABENDx. The keyword for an abend problem.

Advanced Peer-to-Peer Networking. A distributed network and session control architecture that allows networked computers to communicate dynamically as equals. Compare with Advanced Program-to-Program Communication (APPC). An implementation of the SNA synchronous data link control LU 6.2 protocol that allows interconnected systems to communicate and share the processing of programs.

Advanced Program-to-Program Communication (APPC). An implementation of the SNA synchronous data link control LU 6.2 protocol that allows interconnected systems to communicate and share the processing of programs.

aggregation function. Any of a group of functions that summarizes data in a column. They are requested with these usage codes on the form panels: AVERAGE, CALC, COUNT, FIRST, LAST, MAXIMUM, MINIMUM, STDEV, SUM, CSUM, PCT, CPCT, TPCT, TCPCT.

aggregation variable. An aggregation function that is placed in a report using either the FORM.BREAK, FORM.CALC, FORM.DETAIL, or FORM.FINAL panels. Its value appears as part of the break footing, detail block text, or final text when the report is produced.

alias. In DB2 for OS/390, an alternate name that can be used in SQL statements to refer to a table or view in the same or a remote DB2 for OS/390 subsystem. In OS/2, an alternate name used to identify a object, a database, or a network resource such as an LU. In QMF, a locally defined name used to access a QMF table or view stored on a local or remote DB2 for OS/390 subsystem.

APAR. Authorized Program Analysis Report.

APPC. Advanced Program-to-Program Communication

application. A program written by QMF users that extends the capabilities of QMF without modifying the QMF licensed program. Started from a QMF session by issuing a RUN command for a QMF procedure, an installation-defined command, or a CMS or TSO command that invokes an EXEC or CLIST, respectively.

application requester. (1) A facility that accepts a database request from an application process and passes it to an application server. (2) In DRDA, the source of a request to a remote relational database management system.

The application requester is the DBMS code that handles the QMF end of the distributed connection. The local DB2 for OS/390 subsystem to which QMF attaches is known as the application requester for QMF, because DB2 for OS/390's application requester is installed within the local database manager.

Therefore, an entire DB2 for OS/390 subsystem (including data) is associated with the application requester, but the SQL statements are processed at the current location. This subsystem is called the "local DB2 for OS/390".

With DB2 for VM and VSE the application requester runs in the same virtual machine as QMF; that is, no database is inherently associated with the DB2 for VM and VSE application requester.

application server. The target of a request from an application requester. (1) The local or remote database manager to which the application process is connected. The application server executes at the system containing the desired data. (2) In DRDA, the target of a request from an application requester. With DB2 for OS/390, the application server is part of a full DB2 for OS/390 subsystem.

With DB2 for VM and VSE, the application server is part of a DB2 for VM and VSE database machine.

application-support command. A QMF command that can be used within an application program to exchange information between the application program and QMF. These commands include INTERACT, MESSAGE, STATE, and QMF.

area separator. The barrier that separates the fixed area of a displayed report from the remainder of the report.

argument. An independent variable.

base QMF environment. The English-language environment of QMF, established when QMF is installed. Any other language environment is established after installation.

batch QMF session. A QMF session running in the background. Begins when a specified QMF procedure is invoked and ends when the procedure ends. During a background QMF session, no user interaction and panel display interaction are allowed.

bind. In DRDA, the process by which the SQL statements in an application program are made known to a database management system over application support protocol (and database support protocol) flows. During a bind, output from a precompiler or preprocessor is converted to a control structure called a package. In addition, access paths to the referenced data are selected and some authorization checking is performed. (Optionally in DB2 for OS/390, the output may be an application plan.)

built-in function. Generic term for scalar function or column function. Can also be "function."

calculation variable. CALCid is a special variable for forms that contains a user-defined calculated value. CALCid is defined on the FORM.CALC panel.

callable interface. A programming interface that provides access to QMF services. An application can access these services even when the application is running outside of a QMF session. Contrast with command interface.

chart. A graphic display of information in a report.

CICS. Customer Information Control System.

client. A functional unit that receives shared services from a server.

CMS. Conversational Monitor System.

column. A vertical set of tabular data. It has a particular data type (for example, character or numeric) and a name. The values in a column all have the same data characteristics.

column function. An operation that is applied once to all values in a column, returns a single value as a result, and is expressed in the form of a function name followed by one or more arguments enclosed in parentheses.

column heading. An alternative to the column name that a user can specify on a form. Not saved in the database, as are the column name and label.

column label. An alternative descriptor for a column of data that is saved in the database. When used, column labels appear by default on the form, but they can be changed by users.

column wrapping. Formatting values in a report so that they occupy several lines within a column. Often used when a column contains values whose length exceeds the column width.

command interface. An interface for running QMF commands. The QMF commands can only be issued from within an active QMF session. Contrast with callable interface.

command synonym. The verb or verb/object part of an installation-defined command. Users enter this for the command, followed by whatever other information is needed.

command synonym table. A table each of whose rows describes an installation-defined command. Each user can be assigned one of these tables.

commit. The process that makes a data change permanent. When a commit occurs, data locks are freed enabling other applications to reference the just-committed data. See also “rollback”.

concatenation. The combination of two strings into a single string by appending the second to the first.

connectivity. The enabling of different systems to communicate with each other. For example, connectivity between a DB2 for OS/390 application requester and a DB2 for VM and VSE application server enables a DB2 for OS/390 user to request data from a DB2 for VM and VSE database.

conversation. A logical connection between two programs over an LU 6.2 session that allows them to communicate with each other while processing a transaction.

correlation name. An alias for a table name, specified in the FROM clause of a SELECT query. When concatenated with a column name, it identifies the table to which the column belongs.

CP. The Control Program for VM.

CSECT. Control section.

current location. The application server to which the QMF session is currently connected. Except for connection-type statements, such as CONNECT (which are handled by the application requester), this server processes all the SQL statements. When initializing QMF, the current location is indicated by the DSQSDBNM startup program parameter. (If that parameter is not specified, the local DB2 for OS/390 subsystem

current object. An object in temporary storage currently displayed. Contrast with saved object.

Customer Information Control System (CICS). An IBM licensed program that enables transactions entered at remote terminals to be processed concurrently by user-written application programs. It includes facilities for building, using, and maintaining databases.

DATA. An object in temporary storage that contains the information returned by a retrieval query. Information represented by alphanumeric characters contained in tables and formatted in reports.

database. A collection of data with a given structure for accepting, storing, and providing on demand data for multiple users. In DB2 for OS/390, a created object that contains table spaces and index spaces. In DB2 for VM and VSE, a collection of tables, indexes, and supporting information (such as control information and data recovery information) maintained by the system. In OS/2, a collection of information, such as tables, views, and indexes.

database administrator. The person who controls the content of and access to a database.

database management system. A computer-based system for defining, creating, manipulating, controlling, managing, and using databases. The database management system also has transaction management and data recovery facilities to protect data integrity.

database manager. A program used to create and maintain a database and to communicate with programs requiring access to the database.

database server. (1) In DRDA, the target of a request received from an application server (2) In OS/2, a workstations that provides database services for its local database to database clients.

date. Designates a day, month, and year (a three-part value).

date/time default formats. Date and time formats specified by a database manager installation option. They can be the EUR, ISO, JIS, USA, or LOC (LOCAL) formats.

date/time data. The data in a table column with a DATE, TIME, or TIMESTAMP data type.

DB2 for OS/390. DATABASE 2 for OS/390 (an IBM relational database management system).

DB2 for AIX. DATABASE2 for AIX. The database manager for QMF's relational data.

DBCS. Double-byte character set.

DBMS. Database management system.

default form. The form created by QMF when a query is run. The default form is not created if a saved form is run with the query.

destination control table (DCT). In CICS, a table containing a definition for each transient data queue.

detail block text. The text in the body of the report associated with a particular row of data.

detail heading text. The text in the heading of a report. Whether or not headings will be printed is specified in FORM.DETAIL.

dialog panel. A panel that overlays part of a Prompted Query primary panel and extends the dialog that helps build a query.

distributed data. Data that is stored in more than one system in a network, and is available to remote users and application programs.

distributed database. A database that appears to users as a logical whole, locally accessible, but is comprised of databases in multiple locations.

distributed relational database. A distributed database where all data is stored according to the relational model.

Distributed Relational Database Architecture. A connection protocol for distributed relational database processing that is used by IBM and vendor relational database products.

distributed unit of work. A method of accessing distributed relational data in which users or applications can, within a single unit of work, submit SQL statements to multiple relational database management systems, but no more than one RDBMS per SQL statement.

DB2 for OS/390 introduced a limited form of distributed unit of work support in its V2R2 called system-directed access, which QMF supports.

DOC. The keyword for a document problem.

double-byte character. An entity that requires two character bytes.

double-byte character set (DBCS). A set of characters in which each character is represented by two bytes. Languages such as Japanese, Chinese, and Korean, which contain more symbols that can be represented by 256 code points, require double-byte character sets. Because each character requires two bytes, the typing, display, and printing of DBCS characters requires hardware and programs that support DBCS. Contrast with single-byte character set.

DRDA. Distributed Relational Database Architecture.

duration. An amount of time expressed as a number followed by one of seven keywords: YEARS, MONTHS, DAYS, HOURS, MINUTES, SECONDS, MICROSECONDS.

EBCDIC. Extended Binary-Coded Decimal Interchange Code.

echo area. The part of the Prompted Query primary panel in which a prompted query is built.

EUR (European) format. A format that represents date and time values as follows:

- Date: dd.mm.yyyy
- Time: hh.mm.ss

extended syntax. QMF command syntax that is used by the QMF callable interface; this syntax defines variables that are stored in the storage acquired by the callable interface application and shared with QMF

example element. A symbol for a value to be used in a calculation or a condition in a QBE query.

example table. The framework of a QBE query.

fixed area. That part of a report that contains fixed columns.

fixed columns. The columns of a report that remain in place when the user scrolls horizontally. On multiple-page, printed reports, these columns are repeated on the left side of each page.

form. An object that contains the specifications for printing or displaying a report or chart. A form in temporary storage has the name of FORM.

function key table. A table containing function key definitions for one or more QMF panels, along with text describing the keys. Each user can be assigned one of these tables.

gateway. A functional unit that connects two computer networks of different network architectures. A gateway connects networks or systems of different architectures, as opposed to a bridge, which connects networks or systems with the same or similar architectures.

GDDM. Graphical Data Display Manager.

global variable. A variable that, once set, can be used for an entire QMF session. A global variable can be used in a procedure, query, or form. Contrast with run-time variable.

Graphical Data Display Manager. A group of routines that allows pictures to be defined and displayed procedurally through function routines that correspond to graphic primitives.

grouped row. A row of data in a QBE target or example table that is summarized either by a G. or a built-in function.

HELP. Additional information about an error message, a QMF panel, or a QMF command and its options.

host. A mainframe or mid-size processor that provides services in a network to a workstation.

HTML. Hypertext Markup Language. A standardized markup language for documents displayed on the World Wide Web.

ICU. Interactive Chart Utility.

INCORROUT. The keyword for incorrect output.

index. A collection of data about the locations of records in a table, allowing rapid access to a record with a given key.

initial procedure. A QMF procedure specified by the DSQSRUN parameter on the QMF start command which is executed immediately after QMF is invoked.

initialization program. A program that sets QMF program parameters. This program is specified by DSQSCMD in the callable interface. The default program for interactive QMF is DSQSCMD*n*, where *n* is the qualifier for the presiding language ('E' for English).

installation-defined command. A command created by an installation. QMF will process it as one of its own commands or as a combination of its commands.

installation-defined format. Date and time formats, also referred to as LOCAL formats, that are defined (or built) by the installation.

interactive execution. Execution of a QMF command in which any dialog that should take place between the user and QMF during the command's execution actually does take place.

interactive session. Any QMF session in which the user and QMF can interact. Could be started by another interactive session by using the QMF INTERACT command.

interactive switch. A conceptual switch which, when on, enables an application program to run QMF commands interactively.

invocation CLIST or EXEC. A program that invokes (starts) QMF.

ISO (International Standards Organization) format. A format that represents date and time values as follows:

- Date: yyyy-mm-dd
- Time: hh.mm.ss

ISPF. Interactive System Productivity Facility.

IXF. Integration Exchange Format: A protocol for transferring tabular data among various software products.

JCL. Job control language for OS/390.

job control. In VSE, a program called into storage to prepare each job or job step to be run. Some of its functions are to assign I/O devices to symbolic names, set switches for program use, log (or print) job control statements, and fetch the first phase of each job step.

JIS (Japanese Industrial Standard) format. A format that represents date and time values as follows:

- Date: yyyy-mm-dd
- Time: hh:mm:ss

join. A relational operation that allows retrieval of data from two or more tables based on matching columns that contain values of the same data type.

keyword parameter. An element of a QMF command consisting of a keyword and an assigned value.

like. Pertaining to two or more similar or identical IBM operating environments. For example, like distribution is distribution between two DB2 for OS/390's with compatible server attribute levels. Contrast with "unlike".

literal. In programming languages, a lexical unit that directly represents a value. A character string whose value is given by the characters themselves.

linear procedure. Any procedure *not* beginning with a REXX comment. A linear procedure can contain QMF commands, comments, blank lines, RUN commands, and substitution variables. See also "procedure with logic."

linear syntax. QMF command syntax that is entered in one statement of a program or procedure, or that can be entered on the QMF command line.

line wrapping. Formatting table rows in a report so they occupy several lines. The row of column names and each row of column values are split into as many lines as are required by the line length of the report.

local. Pertaining to the relational database, data, or file that resides in the user's processor. See also "local DB2 for OS/390", and contrast with *remote*.

local area network (LAN). (1) Two or more processors connected for local resource sharing (2) A network within a limited geographic area, such as a single office building, warehouse, or campus.

local data. Data that is maintained by the subsystem that is attempting to access the data. Contrast with remote data.

local DB2 for OS/390. With DB2 for OS/390, the application requester is part of a DB2 for OS/390 subsystem that is running in the same MVS system as QMF. Therefore, an entire DB2 for OS/390 subsystem (including data) is associated with the application requester, but the SQL statements are processed at the current location. This subsystem is where the QMF plan is bound.

When QMF runs in TSO, this subsystem is specified using DSQSSUBS startup program parameter. When QMF runs in CICS, this subsystem is identified in the Resource Control Table (RCT). The local DB2 for OS/390 is the subsystem ID of the DB2 for OS/390 that was started in the CICS region.

location. A specific relational database management system in a distributed relational database system. Each DB2 for OS/390 subsystem is considered to be a location.

logical unit (LU). A port through which an end user accesses the SNA network to communicate with another end user and through which the end user accesses the functions provided by system services control points.

Logical Unit type 6.2 (LU 6.2). The SNA logical unit type that supports general communication between programs in a distributed processing environment.

LU. Logical unit.

LU 6.2. Logical Unit type 6.2.

LOOP. The keyword for an endless-loop problem.

MSGx. The keyword for a message problem.

Multiple Virtual Storage. Implies the MVS/ESA product

MVS/ESA. Multiple Virtual Storage/Enterprise System Architecture (IBM operating system).

NCP. Network Control Program.

Network Control Program (NCP). An IBM licensed program that provides communication controller support for single-domain, multiple-domain, and interconnected network capability.

NLF. National Language Feature. Any of several optional features available with QMF that lets the user select a language other than US English.

NLS. National Language Support.

node. In SNA, an end point of a link or a junction common to two or more links in a network. Nodes can be distributed to host processors, communication controllers, cluster controllers, or terminals. Nodes can vary in routing and other functional capabilities.

null. A special value used when there is no value for a given column in a row. *Null* is not the same as zero.

null value. See *null*.

object. A QMF query, form, procedure, profile, report, chart, data, or table. The report, chart, and data objects exist only in temporary storage; they cannot be saved in a database. The table object exists only in a database.

object name. A character string that identifies an object owned by a QMF user. The character string can be a maximum of 18 bytes long and must begin with an alphabetic character. The term "object name" does not include the "owner name" prefix. Users can access other user's objects only if authorized.

object panel. A QMF panel that can appear online after the execution of one QMF command and before the execution of another. Such panels include the home, report, and chart panels, and all the panels that display a QMF object. They do not include the list, help, prompt, and status panels.

online execution. The execution of a command from an object panel or by pressing a function key.

owner name. The authorization id of the user who creates a given object.

package. The control structure produced when the SQL statements in an application program are bound to a relational database management system. The database management system uses the control structure to process SQL statements encountered during statement execution.

panel. A particular arrangement of information, grouped together for presentation in a window. A panel can contain informational text, entry fields, options the user can choose from, or a mixture of these.

parameter. An element of a QMF command. This term is used generically in QMF documentation to reference a *keyword parameter* or a *positional parameter*.

partner logical unit. In SNA, the remote system in a session.

PERFM. The keyword for a performance problem.

permanent storage. The database where all tables and QMF objects are stored.

plan. A form of package where the SQL statements of several programs are collected together during bind to create a plan.

positional parameter. An element of a QMF command that must be placed in a certain position within the command.

primary panel. The main Prompted Query panel containing your query.

primary QMF session. An interactive session begun from outside QMF Within this session, other sessions can be started by using the INTERACT command.

procedure. An object that contains QMF commands. It can be run with a single RUN command. A procedure in temporary storage has the name of PROC. See also “linear procedure” and “procedure with logic.”

procedure termination switch. A conceptual switch that a QMF MESSAGE command can turn on. While on, every QMF procedure to which control returns terminates immediately.

procedure with logic. Any QMF procedure beginning with a REXX comment. In a procedure with logic, you can perform conditional logic, make calculations, build strings, and pass commands back to the host environment. See also “linear procedure.”

profile. An object that contains information about the characteristics of the user’s session. A stored profile is a profile that has been saved in permanent storage. A profile in temporary storage has the name PROFILE. There can be only one profile for each user.

prompt panel. A panel that is displayed after an incomplete or incorrect QMF command has been issued.

Prompted Query. A query built in accordance with the user’s responses to a set of dialog panels.

protocol. The rules governing the functions of a communication system that must be followed if communication is to be achieved.

PSW. Program status word.

PTF. Program temporary fix.

QBE (Query-By-Example). A language used to write queries graphically. For more information see *Using QMF*

QMF administrative authority. At minimum, insert or delete priviledge for the Q.PROFILES control table.

QMF administrator. A QMF user with QMF administrative authority.

QMF command. Refers to any command that is part of the QMF language. Does **not** include installation-defined commands.

QMF session. All interactions between the user and QMF from the time the user invokes QMF until the EXIT command is issued.

qualifier. When referring to a QMF object, the part of the name that identifies the owner. When referring to a TSO data set, any part of the name that is separated from the rest of the name by periods. For example, ‘TCK’, ‘XYZ’, and ‘QUERY’ are all qualifiers in the data set name ‘TCK.XYZ.QUERY’.

query. An SQL or QBE statement, or a statement built from prompting, that performs data inquiries or manipulations. A saved query is an SQL query, QBE query, or Prompted Query that has been saved in a database. A query in temporary storage, has the name QUERY.

RDBMS. Relational database management system

relational database. A database perceived by its users as a collection of tables.

relational database management system (RDBMS). A computer-based system for defining, creating, manipulating, controlling, managing, and using relational databases.

remote. Pertaining to a relational DBMS other than the local relational DBMS.

remote data. Data that is maintained by a subsystem other than the subsystem that is attempting to access the data. Contrast with local data.

remote data access. Methods of retrieving data from remote locations. The two remote data access functions used by QMF are *remote unit of work* and DB2 for OS/390-only distributed unit of work, which is called *system-directed access*.

remote unit of work. (1) The form of SQL distributed processing where the application is on a system different from the relational database and a single application server services all remote unit of work requests within a single logical unit of work. (2) A unit of work that allows for the remote preparation and execution of SQL statements.

report. The formatted data produced when a query is issued to retrieve data or a DISPLAY command is entered for a table or view.

REXX. Restructured extended executor.

rollback. The process that removes uncommitted database changes made by one application or user. When a rollback occurs, locks are freed and the state of the resource being changed is returned to its state at the last commit, rollback, or initiation. See also *commit*.

row. A horizontal set of tabular data.

row operator area. The leftmost column of a QBE target or example table.

run-time variable. A variable in a procedure or query whose value is specified by the user when the procedure or query is run. The value of a run-time variable is only available in the current procedure or query. Contrast with global variable.

sample tables. The tables that are shipped with QMF. Data in the sample tables is used to help new QMF users learn the product.

saved object. An object that has been saved in the database. Contrast with current object.

SBCS. Single-byte character set.

scalar. A value in a column or the value of a literal or an expression involving other scalars.

scalar function. An operation that produces a single value from another value and is expressed in the form of a function name followed by a list of arguments enclosed in parentheses.

screen. The physical surface of a display device upon which information is presented to the user.

scrollable area. The view of a displayed object that can be moved up, down, left, and right.

server. A functional unit that provides shared services to workstations over a network.

Server-Requester Programming Interface (SRPI). An application programming interface (API) used by requester and server programs to communicate with the personal computer or host routers.

session. All interactions between the user and QMF from the time the user logs on until the user logs off.

single-byte character. A character whose internal representation consists of one byte. The letters of the Latin alphabet are examples of single-byte characters.

SNA. Systems Network Architecture.

SNAP dump. A dynamic dump of the contents of one or more storage areas that QMF generates during an abend.

sort priority. A specification in a retrieval query that causes the sorted values in one retrieved column to determine the sorting of values in another retrieved column.

SQL. Structured Query Language.

SQLCA. Structured Query Language Communication Area.

SRPI. Server-Requester Programming Interface.

SSF. Software Support Facility. An IBM online database that allows for storage and retrieval of information about all current APARs and PTFs.

stored object. An object that has been saved in permanent storage. Contrast with current object.

string. A set of consecutive items of a similar type; for example, a character string.

Structured Query Language. A language used to communicate with DB2 for OS/390 and SQL/DS. Used to write queries in descriptive phrases.

subquery. A complete SQL query that appears in a WHERE or HAVING clause of another query (the main query or a higher-level subquery).

substitution variable. (1) A variable in a procedure or query whose value is specified either by a global variable or by a run-time variable. (2) A variable in a form whose value is specified by a global variable.

substring. The part of a string whose beginning and length are specified in the SUBSTR function.

System Log (SYSLOG). A data set or file in which job-related information, operational data, descriptions of unusual occurrences, commands, and messages to and from the operator may be stored.

Systems Network Architecture. The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through and controlling the configuration and operation of networks.

table. A named collection of data under the control of the relational database manager. A table consists of a fixed number of rows and columns.

Table Editor. The QMF interactive editor that lets authorized users make changes to a database without having to write a query.

table name area. The leftmost column of a QBE example table.

tabular data. The data in columns. The content and the form of the data is specified on FORM.MAIN and FORM.COLUMNS.

target table. An empty table in which example elements are used to combine columns, combine rows, or include constant values in a report.

temporary storage. An area where the query, form, procedure, profile, report, chart, and data objects in current use are stored. All but the data object can be displayed.

temporary storage queue. In CICS, a temporary storage area used for transfer of objects between QMF and an application or a system service.

time. Designates a time of day in hours and minutes and possibly seconds (a two- or three-part value).

thread. The DB2 for OS/390 structure that describes an application's connection, traces its progress, provides resource function processing capability, and delimits its accessibility to DB2 for OS/390 resources and services. Most DB2 for OS/390 functions execute under a thread structure.

three-part name. A fully-qualified name of a table or view, consisting of a location name, owner ID, and object name. When supported by the application server (that is, DB2 for OS/390), a three-part name can be used in an SQL statement to retrieve or update the specified table or view at the specified location.

timestamp. A date and a time, and possibly a number of microseconds (a six- or seven-part value).

TP. Transaction Program

TPN. Transaction program name

transaction. The work that occurs between 'Begin Unit of Work' and 'Commit' or 'Rollback'.

transaction program. A program that processes transactions in an SNA network. There are two kinds of transactions programs: application transaction programs and service transaction programs.

transaction program name. The name by which each program participating in an LU 6.2 conversation is known. Normally, the initiator of a connection identifies the name of the program it wants to connect to at the other LU. When used in conjunction with an LU name, it identifies a specific transaction program in the network.

transient data queue. In CICS, a storage area, whose name is defined in the Destination Control Table (DCT), where objects are stored for subsequent internal or external processing.

TSO. Time Sharing Option.

two-phase commit. A protocol used in distributed unit of work to ensure that participating relational database management systems commit or roll back a unit of work consistently.

unit of work. (1) A recoverable sequence of operations within an application process. At any time, an application process is a single unit of work, but the life of an application process may involve many units of work as a result of commit or rollback operations. (2) In DRDA, a sequence of SQL commands that the database manager treats as a single entity. The database manager ensures the consistency of data by verifying that either all the data changes made during a unit of work are performed or none of them are performed.

unlike. Refers to two or more different IBM operating environments. For example, unlike distribution is distribution between DB2 for VM and VSE and DB2 for OS/390. Contrast with *like*.

unnamed column. An empty column added to an example table. Like a target table, it is used to combine columns, combine rows, or include constant values in a report.

USA (United States of America) format. A format that represents date and time values as follows:

- Date: mm/dd/yyyy
- Time: hh:mm xM

value. A data element with an assigned row and column in a table.

variation. A data formatting definition specified on a FORM.DETAIL panel that conditionally can be used to format a report or part of a report.

view. An alternative representation of data from one or more tables. It can include all or some of the columns contained in the table or tables on which it is defined. (2) The entity or entities that define the scope of the data to be searched for a query.

Virtual Storage Extended. An operating system that is an extension of Disk Operating System/ Virtual Storage. A VSE consists of (1) VSE/Advanced Functions support and (2) any IBM-supplied and user-written programs that are required to meet the data processing needs of a user. VSE and the hardware it controls form a complete computing system.

VM. Virtual Machine (IBM operating system). The generic term for the VM/ESA environment.

VSE. Virtual Storage Extended (IBM operating system). The generic term for the VSE/ESA environment.

WAIT. The keyword for an endless-wait-state problem.

window. A rectangular portion of the screen in which all or a portion of a panel is displayed. A window can be smaller than or equal to the size of the screen.

Workstation Database Server. The IBM family of DRDA database products on the UNIX and Intel platforms (such as DB2 Universal Database (UDB), DB2 Common Server, DB2 Parallel Edition, and DataJoiner.)

wrapping. See “column wrapping” and “line wrapping”.

Bibliography

The following lists do not include all the books for a particular library. To order copies of the books listed here, or to get more information about a particular library, see your IBM representative.

For a list of QMF publications, see "The QMF Library" on page vii.

CICS Publications

CICS Application Programmer's Reference

GDDM Publications

Interactive Chart Utility User's Guide

IBM DATABASE 2 Publications

IBM DATABASE 2 SQL Reference

REXX Publications

TSO/E Procedures Language MVS/REXX Reference

Virtual Machine/Enterprise Systems Architecture REXX/VM Reference

VM/XA System Product Interpreter Reference

SQL/Data System Publications

Version 2 Release 2

SQL/Data System: SQL Reference for VM/System Product and VM/Extended Architecture System Product

Version 3 Release 1 and later

SQL/Data System: SQL Reference for IBM VM Systems and VSE

SQL/Data System: System Administration for IBM VM Systems

Index

Special Characters

A

B

C

D

E

F

G

H

I

J

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

Readers' Comments — We'd Like to Hear from You

Query Management Facility
QMF Reference
Version 6

Publication No. SC26-9577-00

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name _____ Address _____

Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SC26-9577-00

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

IBM CORPORATION
Department BWE/H3
P.O. Box 49023
San Jose, CA
U.S.A>
95161-9023

Fold and Tape **Please do not staple** Fold and Tape

SC26-9577-00

Cut or Fold
Along Line

Program Number: 5645-DB2
5648-A70

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC26-9577-00