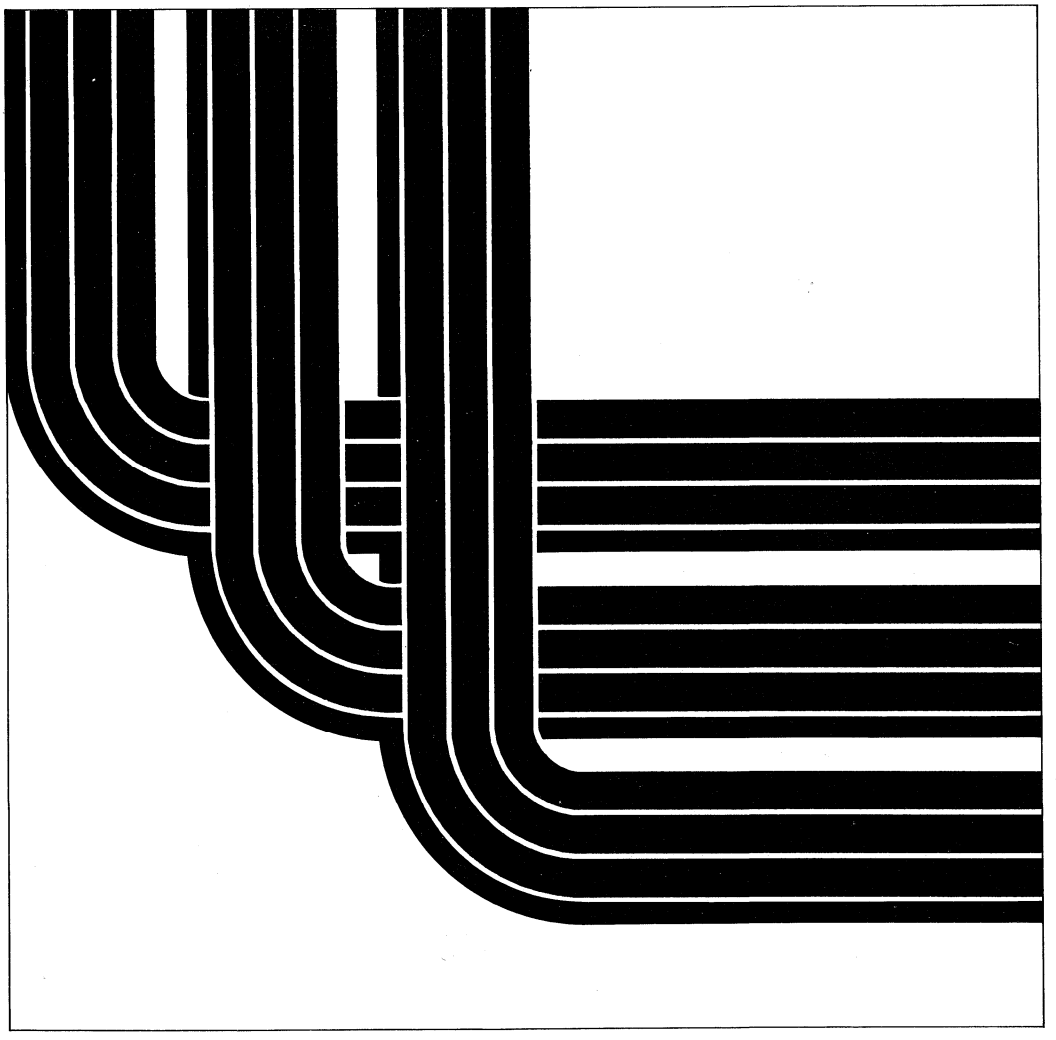
Application System/400™

SC41-8192-00

Programming: Query Management/400 Programmer's Guide

Version 2



Application Development

Take Note!

Before using this information and the product it supports, be sure to read the general information under "Notices" on page xi.

First Edition (May 1991)

This edition applies to the licensed program IBM Operating System/400 (Program 5738-SS1), Version 2 Release 1 Modification 0, and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure that you are using the proper edition for the level of the product.

Order publications through your IBM representative or the IBM branch serving your locality. Publications are not stocked at the address given below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, you may address your comments to:

Attn Department 245
IBM Corporation
3605 Highway 52 N
Rochester, MN 55901-7899

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you or restricting your use of it.

© Copyright International Business Machines Corporation 1991. All rights reserved.
Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

© Copyright IBM Corp. 1991

Figures

© Copyright IBM Corp. 1991

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Any reference to an IBM licensed program or other IBM product in this publication is not intended to state or imply that only IBM's program or other product may be used.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Commercial Relations, IBM Corporation, Purchase, NY 10577.

The following terms, denoted by an asterisk (*) in this publication, are trademarks of the IBM Corporation in the United States and/or other countries:

Application System/400	AS/400
C/440	Cobol/400
IBM	Operating System/400
OS/400	RPG/400
SAA	SQL/400
Systems Application Architecture	400

This publication could contain technical inaccuracies or typographical errors.

This manual may refer to products that are announced but are not yet available.

Information that has changed since Version 1 Release 3 Modification 0 is indicated by a vertical bar (|) to the left of the change.

This manual contains small programs that are furnished by IBM as simple examples to provide an illustration. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. All programs contained herein are provided to you "AS IS". THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE EXPRESSLY DISCLAIMED.

Programming Interface

This *Query Management/400 Programmer's Guide* is intended to help the customer use Query Management/400. It primarily contains general-use programming interfaces, which allow the customer to write programs that use the services of Query Management/400.

© Copyright IBM Corp. 1991

About This Guide

This guide describes and provides examples for using the Query Management/400 functions provided by the Operating System/400 licensed program.

The term *DATA* spelled in capital letters is used throughout this book. *DATA* or *DATA set* refers to the active information resulting from running a query.

This guide was produced in conjunction with the *Programming: Query Management/400 Reference*, SC41-8193.

You may need to refer to other IBM manuals for more specific information about a particular topic. The *Publications Guide*, GC41-9678, provides information on all the manuals in the AS/400 library.

For a list of related publications, see the "Bibliography."

Who Should Use This Guide

This guide is intended to be used by the following:

- Applications programmers familiar with the Systems Application Architecture (SAA) environment, the Common Programming Interface (CPI), Query/400, and using Query/400 definitions
- System operators who have had formal AS/400 system training and are familiar with operating the AS/400 system and Query/400 functions
- Users who are performing problem analysis
- IBM programming service personnel, who are responsible for resolving noncustomer problems with systems programs and the Query Management/400 application

Note: Before you use this guide, you should be knowledgeable in the following:

- SAA environment
- CPI
- Query/400
- Structured Query Language/400 (SQL/400) licensed program
- AS/400 system operation and functions

What the SAA Solution Is

The SAA solution is based on a set of software interfaces, conventions, and protocols that provide a framework for designing and developing applications. The SAA solution:

- Defines a common programming interface that you can use to develop applications that can be integrated with each other, and transported to run in multiple SAA environments
- Defines common communications support that you can use to connect applications, systems, networks, and devices

© Copyright IBM Corp. 1991

- Defines a common user access that you can use to achieve consistency in panel layout and user interaction techniques
- Offers some applications and application development tools written by IBM.

Chapter 1. Introduction

This chapter introduces the Operating System/400* (OS/400*) query management system (called Query Management/400) and describes some of its characteristics, specifications, and requirements, and its relationship to the Systems Application Architecture* (SAA*) environment and the Query/400 programs.

Query Management Overview

This manual covers the AS/400* implementation of SAA Query Common Programming Interface (CPI) as defined in the *SAA CPI Query Reference* manual.

Note: A working knowledge of SAA Query CPI is recommended as a prerequisite to working with this product. A knowledge of Query/400 and Structured Query Language (SQL) would also be beneficial.

This manual describes the functions and user interface for the AS/400 SAA query management CPI. The CPI functions are provided by query management. The CPI allows a user to access information in a relational database and control how this data appears when formatted into a report. The CPI provides services that fall into two major categories: querying and report writing.

Application programs can use query management services through a program-to-program call interface using Query/400 objects. The Query/400 objects are created only through the CPI using files containing externalized query, procedure, and form definitions. The externalized files can be built using an editor, built by an application program, transferred from another system (from which they were exported), or created through conversion of definition (QRYDFN) objects that were created through the Query/400 product.

Query Management Enhancements

Query Management/400 uses functions common to both Query/400 and SAA Query CPI. Query management also has sections that are enhanced versions of SAA Query CPI. This section describes some of the query management enhancements to the following SAA Query CPI elements:

- Query management concepts
- Naming conventions
- Security and authorization

AS/400 and the SAA Environment

Query management objects are created and maintained as AS/400 system objects. Figure 1-1 on page 1-2 shows the relationships among the SAA application relational database terms, AS/400 system terms, and the AS/400 SAA environment terms.

© Copyright IBM Corp. 1991

Figure 1-1. AS/400 and SAA Terminology

SAA Term	AS/400 System Term	AS/400 SAA Use
Collection — A collected group of tables.	**Library** — A library groups related objects, allowing the user to find the objects by name.	**Collection** — A collection consists of a library, a journal, a journal receiver, a data dictionary, and an SQL catalog. A collection groups related objects, allowing the user to find the objects by name.
Table — Logical structures made up of columns and rows maintained by the database manager.	**Physical file** — A collected group of records.	**Table** — A collection of columns and rows.
Row — A sequence of values such that the *n*th value is a value in the *n*th column of the table.	**Record** — A collection of fields.	**Row** — The horizontal part of a table containing a serial collection of columns.
Column — A set of values of the same type.	**Field** — One or more characters of related information of one data type.	**Column** — The vertical part of a table of one data type.
View — An alternative way of looking at the data in one or more tables.	**Logical file** — A subset of fields and records of one or more physical files.	**View** — A subset of columns and rows of one or more tables.
Authorization ID — A short identifier that designates a user.	**User profile** — A name that identifies a user and designates a set of privileges on the AS/400 system.	**Authorization ID** — A character string of not more than 10 bytes that identifies a user.

Collection Use by Query Management

Query management treats all objects as belonging to a collection. Query management is not as strict in its use of objects within collections. The following list describes query management collection conventions:

- Query management objects (queries, forms, procedures) are not part of a collection. As shown in Figure 1-1, a collection on the AS/400 system is a library. A query management object may be part of a library, but if the library is a collection, there is no entry for the query management object within the collection catalogs or journals.
- Tables manipulated by the query management commands (ERASE and SAVE) are treated as SQL tables. Manipulation of a table using the ERASE query management command requires that the table be part of a collection. Manipulation of a table using the SAVE query management command requires that the table be part of a collection only when saving to a new table. The SQL/400 rules applicable to the SQL table manipulation statements are in effect when manipulating tables through query management commands. The following list shows the correspondence between query management commands and SQL statements:

ERASE	Drop Table
SAVE (to new table)	Create Table and Insert
SAVE REPLACE	Delete and Insert
SAVE APPEND	Insert

Naming Conventions

Use the guidelines in the following sections when you create a table or view or name an object that you want to save in the database.

Query Objects

You can specify query objects on commands using either SAA names or system names. The naming convention to use is specified in the query management command procedure on the START command or the Start Query Management Query (STRQMQRY) CL command. Query management uses SAA (*SAA) naming as the default. The naming convention specified for a query management instance is used throughout the entire instance and applies only to that instance. You cannot change it after you issue the START command for the query instance.

Note: A query management instance is a collection of system resources and a set of query commands within an application program.

These naming conventions only apply to the query object specified on a command. System naming conventions always apply on the file name specified on the IMPORT and EXPORT commands.

System Naming

When a query instance uses system naming, the following rules apply for a query management object name specified in a query command:

- Specify query objects using names enclosed in quotation marks. The name is a character string with the following characteristics:
  - One to eight characters long
  - Begins and ends with quotation marks ("")
  - Contains any character except
    - A blank
    - An asterisk (*)
    - A question mark (?)
    - Apostrophes (')
    - Quotation marks ("")
    - The numbers hex 00 through 3F, or hex FF

  The name may be qualified, but the qualifier and the name must be surrounded by quotation marks separately from each other.
- Specify query objects using simple names. Simple names are character strings up to 10 characters long and must begin with an alphabetic character (A through Z, $, #, or @). Periods and blanks are not allowed in simple names.
- Qualify a query command file name by using a library name up to 10 characters long as a qualifier. A slash (/) must separate the qualifying library name and the file name. For example, MYLIB/FILE1 (file FILE1 in library MYLIB) is a qualified name. The rules applying to AS/400 names in quotation marks and simple names apply to the library name used as the qualifier.
- Give objects of the same type that are stored in the same library different names. (You cannot have two files named TEST, for example). Queries and forms are different AS/400 object types. Therefore, a query and form may have the same name. Names for procedures, tables, and views must be different, since they are all AS/400 files. A procedure, table, or view can have the same

name as a query object or form object, but cannot have the same name as another procedure, table, or view.

- You can name queries, forms, and procedures using reserved words (such as FORM, QUERY, COUNT, NULL, and so on). However, naming something with one of these SQL keywords is not recommended.
- Query management follows AS/400 search conventions if a query, form, or procedure specified in a query command is not qualified. If you specify an unqualified query management object name, query management searches the library list (*LIBL) for the query object. If the query object is being created, query management places the object in the current library (*CURLIB).
- See the *SQL/400* Reference* manual for the search conventions to follow if an unqualified table or view name is specified on a query management command.

For more information on system naming and AS/400 search conventions, see the *CL Reference* manual.

SAA Naming

When the query management instance is using SAA names, the following rules apply for a query management object name specified in a query management command. These rules are similar to the AS/400 object naming conventions. In most cases, they are an extension of the object naming conventions stated in the *SAA CPI Query Reference* manual. The deviations from the *SAA CPI Query Reference* manual are noted.

- You can specify query objects using names enclosed in quotation marks. The name is a character string with the following characteristics:
  - One to eight characters long
  - Begins and ends with quotation marks ("")
  - Contains any character except
    - A blank
    - An asterisk (*)
    - A question mark (?)
    - Apostrophes (')
    - Quotation marks ("")
    - The numbers hex 00 through 3F, or hex FF

  Note: An SAA name can be 12 characters long, including the quotation marks. The AS/400 system only allows 10 characters, including the quotation marks.

  The name may be qualified, but the qualifier and the name must be surrounded by quotation marks separately from each other.
- You can use simple names to specify query management objects. Simple names are character strings up to 10 characters long and must begin with an alphabetic character (A through Z, $, #, or @). Periods and blanks are not allowed in simple names.
- You can qualify one name by using another name (usually a user or an authorization identification) of up to 10 single-byte characters, with a period (.) separating the qualifier and the name. For example, Q.QUERY1 (the query in the Q collection) is a qualified name. Query management uses the SQL/400 convention of treating the authorization ID as a user profile. If SAA naming conventions apply, query management attempts to find the object in the library with the same

name as the authorization ID. The rules applying to names enclosed in quotation marks and simple names apply to the library name used as the qualifier.

- You must give different names to objects of the same type that are stored in the same library. (You cannot have two files named TEST, for example.) Queries and forms are different AS/400 object types. Therefore, a query and form may have the same name. Names for procedures, tables, and views must be different, since they are all AS/400 files. A procedure, table, or view can have the same name as a query object or form object , but not the same as another procedure, table, or view.
- Query management searches the library with the same name as the current user profile for the query object if an unqualified query object name is specified. If the query object is being created, query management places the object in the library with same name as the current user profile. These are the same conventions followed by the AS/400 SQL/400 product.

AS/400 Objects

The *Filename* specified on the IMPORT and EXPORT Query commands follows the AS/400 naming conventions for a source physical file.

- You can specify the file name using names enclosed in quotation marks. The name is a character string with the following characteristics:
  - One to eight characters long
  - Begins and ends with quotation marks ("")
  - Contains any character except
    - A blank
    - An asterisk (*)
    - A question mark (?)
    - Apostrophes (')
    - Quotation marks ("")
    - The numbers hex 00 through 3F, or hex FF

  The name may be qualified, but the qualifier and the name must be surrounded by quotation marks separately from each other.
- You can specify the file name using simple names. Simple names are character strings up to 10 characters long and must begin with an alphabetic character (A through Z, $, #, or @). Periods and blanks are not allowed in simple names.
- You can use a library name to qualify a *Filename* in a query command. The library name can be up to 10 characters long, with a slash (/) separating the qualifier and the name. For example, MYLIB/FILE1 (a file in library MYLIB) is a qualified name.
- Query management searches the library list (*LIBL) for a source file named *filename* if a *filename* specified in a query command is not qualified. If the file is being created, query management places the file in the current library (*CURLIB).
- Use the following rules for specifying a member of a physical file that is a multiple member source file.
  - The member name used if no member name is specified defaults to *FIRST on the IMPORT and EXPORT commands.
  - Query management processes a specified member in a physical file if a member name is given as part of the file name. The member name must follow the file name and be delimited by parentheses with no intervening

blanks. For example, you can specify the member MEMBER1 in the file FILE1 by entering a file name as follows:

```
FILE1(MEMBER1)
```

Variable Names

See the section on extended variables in the *Query Management/400 Reference* manual for the rules that apply when you use variables in SQL queries across the callable interface. AS/400-specific rules are:

- Variable names used in SQL must start with a single-byte character letter and be preceded by an ampersand (&). The ampersand delimits the beginning of a variable name and is not included as one of the 18 characters allowed for the name. You cannot have more than one ampersand in a variable name, since each ampersand delimits the beginning of a distinct variable name.
- User-defined variables may not start with the character string DSQ. An error is generated if you try to set a variable that starts with DSQ.
- Variable names within query management are case sensitive. Therefore, the variable i_owe_you, is not the same as the variable I_OWE_YOU.

The following are valid variable names:

In an SQL Query	In the GET/SET Command
&I_owe_you	I_owe_you
&MYVAR123	MYVAR123
&THIS_IS_A_BIG_NAME	THIS_IS_A_BIG_NAME

Other Query Names

The naming convention being used by the query instance also applies to the SQL statements in any SQL query run during the instance. If system names are being used in the query instance, system names apply to the SQL query. If SAA naming conventions apply, then SQL naming conventions apply to the SQL query. See the *SQL/400* Reference* manual for a description of the SQL and system naming conventions followed by the SQL/400 product.

Security and Authorization

Query management uses the AS/400 security and authorization model instead of the security and authorization model described in the *SAA CPI Query Reference* manual. See the *Security Concepts and Planning* manual for information about security concepts for the AS/400 system.

This section discusses general security authorization considerations for the following items:

- Query management objects
- AS/400 objects
- SQL

Query Management Objects

When you create query management objects through a query management command, use one of the following methods to specify the type of public authority for the query management object that you want to give to other users:

- You can specify a default public authority for all objects created during a query management instance by setting a value in the DSQOAUTH keyword in the query command procedure or on the START command. You can specify the following values:

  ***LIBCRTAUT** *LIBCRTAUT authority allows the same authority for the object as specified on the CRTAUT parameter of the library in which the object is being created. If the CRTAUT parameter is changed, the new value will not affect the authority of existing objects.

  ***CHANGE** *CHANGE authority allows other users to perform all operations on the object except those limited to the owner or controlled by object existence authority and object management authority. A user can change or use the query object in any way, except for deleting or transferring it to a new owner.

  ***ALL** *ALL authority allows other users to perform all operations on the object except those limited to the owner or controlled by authorization list management rights. A user can do anything with the query object (including erasing it) except transfer it to a new owner.

  ***EXCLUDE** *EXCLUDE authority prevents other users from doing anything with the query object. Unless given specific types of authority, no user except its owner can use the query object.

  ***USE** *USE authority allows other users to run, export, or print the query object, but prevents them from importing or saving to it.

  authorization list name
  An authorization list controls users' ability to use a query object. For more information, see the *Security Concepts and Planning* manual.

- If you do not specify an authority through the query command procedure, other users have *EXCLUDE access to the query object.

AS/400 Objects

Query management uses the same public authority for creating a nonquery object, such as the source physical file on an EXPORT command, as it does when creating query objects.

SQL

See the *SQL/400* Reference* manual for information on object authority as it applies to the SQL statement within an SQL query.

Chapter 2. Working with Commands

This chapter describes query management and control language (CL) commands and provides some examples of how you can use them to organize and process general reports.

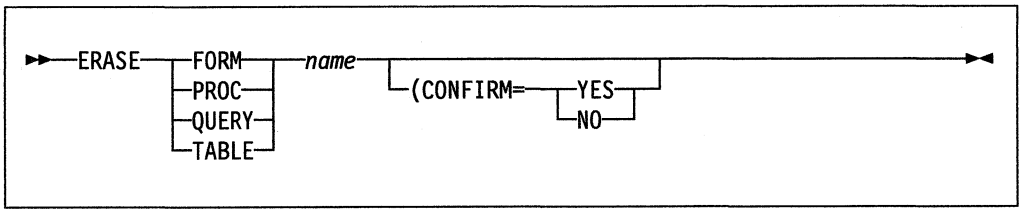
Query Management Commands

Use the following query management commands when writing applications to organize general reports from database files:

- ERASE
- EXIT
- EXPORT
- GET
- IMPORT
- PRINT
- RUN
- SAVE DATA AS
- SET
- START

ERASE

The ERASE command removes a form, procedure, query, or table object from a database file specified by a user.



name

The *name* variable specifies a form, procedure, query, or table object to be removed from the database.

This name can be a qualified name of the form library/object or database.object. Specify the naming convention you intend to use in your first query command procedure.

A user can only erase those objects to which he has been granted *ALL authority and must also have *CHANGE or *ALL authority to the library in which the object resides.

CONFIRM = YES | NO

This option provides a check before performing the ERASE request. The confirmation request occurs only when an existing object in the database is about to be erased.

If your job is running interactively, an inquiry message is sent to your display station and the job is suspended until you respond to the message. The message asks whether you want to erase the object. If your job is running as a

© Copyright IBM Corp. 1991

batch job, or DSQSMODE was set to Batch mode during START processing, CONFIRM=YES results in an error.

CONFIRM=YES forces the system to display the confirmation message. CONFIRM=NO suppresses a display of the confirmation message and erases the object. The default for this option is CONFIRM=YES unless the option is changed during START processing by setting the value of the query-defined variable DSQCONFIRM in the query command procedure.

When you issue the ERASE command, the system returns the message:

```
Object exists. Do you want to replace it?
```

This message is displayed only if you specify the CONFIRM=YES option or if you omit the CONFIRM option.

Examples of the ERASE Command

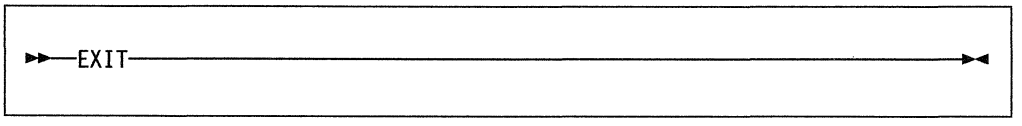
The following examples show how to use the ERASE command in query management:

```
ERASE TABLE EMP

ERASE TABLE SMITH.EMP (CONFIRM=YES

ERASE TABLE SMITH/EMP (CONFIRM=YES

ERASE PROC SMITH/MONTHEND (CONFIRM=NO
```

You can issue an ERASE TABLE command only on database physical files that are in a library that is a collection.

EXIT

The EXIT command stops your application's instance with query management and ends the associated instance of query management in your system. No parameters are allowed with this command.



An implied EXIT command is processed for all query instances when the job ends.

The EXIT command is not valid in a query procedure.

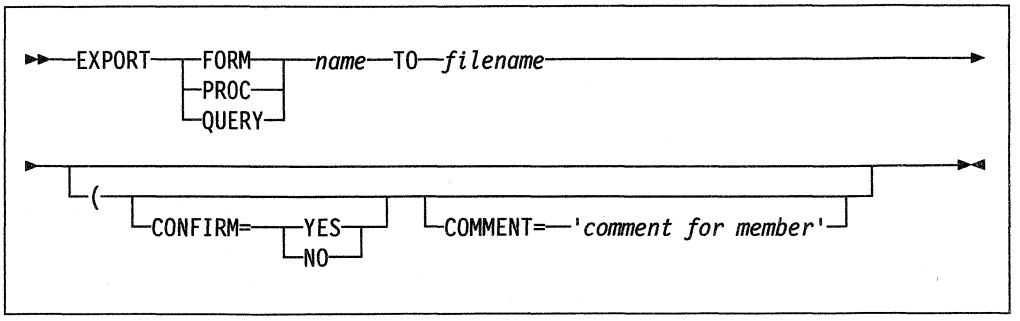
There are no authority considerations related to the EXIT command.

See "EXIT Subprogram" on page 6-13 for an example of a program that uses the EXIT command.

EXPORT

Use the EXPORT command to create a file containing the contents of certain query management objects. The following objects can be exported: form, procedure, or query.

Note: The option to import a table is not supported on the AS/400 system. To export a table to another AS/400 system, use the Save Object (SAVOBJ) and Restore Object (RSTOBJ) CL commands.



name

The *name* variable specifies a form, procedure, or query object to be exported.

This name can be a qualified name of the form library/object or database.object. Specify the naming convention you intend to use in your initial query command procedure.

You can only export forms, procedures, and queries to which you have been granted *ALL authority. You also must have *ALL authority to the library in which the object resides.

If the form or query object specified is not found, and DSQSCNVT=YES is specified on the START command, query management searches for a Query/400 definition with that name. If a Query/400 definition is found, the information is used to create a query or form that query management can use.

filename

The *filename* variable specifies the system file that is to receive the exported object.

This name can be a qualified name of the form library/object or library/object (member name). If the file name is unqualified and does not exist in the library list, the file is placed in the user's current library.

If you are exporting an object to a system other than an AS/400 system, it is recommended that the name of the file be from 1 to 8 characters long.

To be consistent when working with more than one system, do not specify more than one file name without qualifying it. This enables system defaults to take effect.

CONFIRM = YES | NO

This option provides a check before performing the EXPORT request. The confirmation request occurs only when an existing object in the database is about to be exported.

If your job is running interactively, an inquiry message is sent to your display station and the job is suspended until you respond to the message. The message asks whether you want to export the object. If your job is running as a batch job, or DSQSMODE was set to Batch mode during START processing, CONFIRM=YES results in an error.

CONFIRM=YES forces the system to display the confirmation message. CONFIRM=NO suppresses a display of the confirmation message and erases the object. The default for this option is CONFIRM=YES unless the option is

changed during START processing by setting the value of the query-defined variable DSQCONFIRM in the query command procedure.

When you issue the EXPORT command, the system returns the message:

```
Object exists. Do you want to replace it?
```

This message is displayed only if you specify the CONFIRM=YES option or if you omit the CONFIRM option.

COMMENT=Comment for member

Use the comment option to specify the member text when exporting a form, procedure, or query. Comments are useful for preserving information about the object. Because comments usually include embedded blanks, they must be enclosed in apostrophes. Apostrophes within a comment must be specified by two adjacent apostrophes.

Examples:

```
COMMENT='SALES QUERY'
COMMENT='THIS QUERY DOESN''T INCLUDE SALES'
```

Maximum comment length in Query Management/400 is 50 characters.

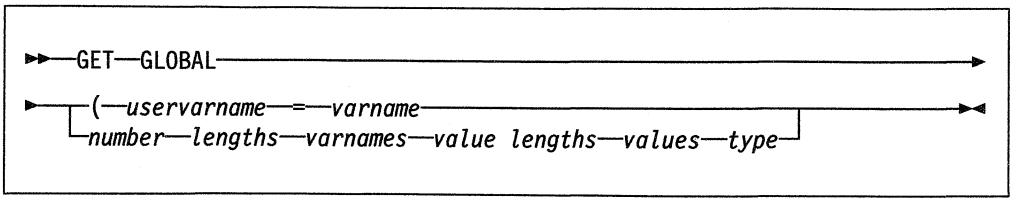
Examples of the EXPORT Command

The following examples show how to use the EXPORT command in query management:

```
EXPORT QUERY SAMP1 TO SAMP1EX

EXPORT PROC DB1.MYPROC TO DB1/MYPROCEX

EXPORT QUERY SMITH/SALARY TO SMITH/SALARY

EXPORT FORM DB1.MYFORM TO DB1/MYFORM
```

GET

Use the GET command to get the value of a query management variable and provide it to a user program or procedure. When using the GET command from a procedure, you must use the short version of the command syntax. When using the GET command from a program, use the extended version of the command syntax. If you use the short version of the command syntax from a program or procedure, the command is not functional since query management does not have access to the user's program storage area to store the variable value.



GLOBAL

The GLOBAL option specifies the *varname* variable located in the query management product global variable pool to be returned to the requester. If the variable is not found in the global pool, an error message is returned.

uservarname
The *uservarname* specifies the name of the user variable to contain the varname value.

varname
The *varname* variable specifies the name of the variable located in the query management variable pool. For rules that apply to variable names used across the callable interface, see the section on extended variable support in the *SAA CPI Query Reference* manual.

Extended Parameter List

Use the following parameters to further qualify the GET command:

number of varnames
Number of *varnames* requested for this call.

varname lengths
Length of each *varname* that is specified.

varname Name of the variable located in the query management variable pool.

user value lengths
Length of the program storage area that is to contain the *varname* value.

user value
Name of the program storage area that is to contain the *varname* value.

value type
Data type of the storage area that is to contain the *varname* value.

Examples of the GET Command

The following examples show the GET command as used in a procedure or on the query command string in a program:

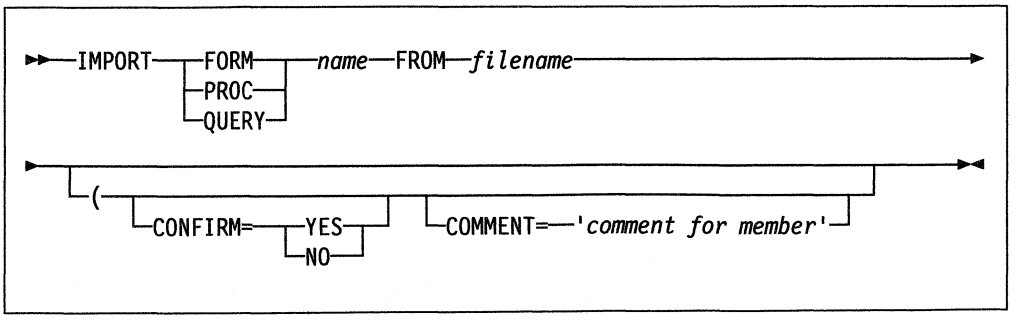
```
GET GLOBAL (sqlret=DSQSQLEC myvar=VARNAME

GET GLOBAL (prtname = DSQAPRNM
```

For more information on the extended parameter list, see "Callable Interface" on page 3-1.

IMPORT

Use the IMPORT command to copy a file containing an exported object into a form, procedure, or query object. The IMPORT command does not affect the external file.

Note: The option to import a table is not supported on the AS/400 system. To import a table from another AS/400 system, you can use the SAVOBJ and RSTOBJ CL commands.



name

The *name* variable specifies the form, procedure, or query object to be imported.

This name can be a qualified name of the form library/object or database.object. Specify the naming convention you intend to use in your initial query command procedure.

filename

The *filename* variable specifies the system file that query management is to read (the source file for the imported object). This name can be a qualified name of the form library/object or library/object(*membername*). To be consistent when working with more than one system, do not specify more than one *filename* without qualifiers or extensions.

For information on the description of the exported file, see "Exported Objects" on page 3-29.

CONFIRM = YES | NO

This option provides a check before performing the IMPORT request. The confirmation request occurs only when an existing object in the database is about to be imported.

If your job is running interactively, an inquiry message is sent to your display station, and the job is suspended until you respond to the message. The message asks whether you want to import the object. If your job is running in batch mode, or DSQSMODE was set to Batch during START processing, CONFIRM = YES results in an error.

CONFIRM = YES forces the system to display the confirmation message. CONFIRM = NO suppresses a display of the confirmation message and imports the object. The default for this option is CONFIRM = YES unless the option is changed during START processing by setting the value of the query-defined variable DSQCONFIRM in the query command procedure.

When you issue the IMPORT command, the system returns the message:

```
Object exists. Do you want to replace it?
```

This message is displayed only if you specify the CONFIRM = YES option or if you omit the CONFIRM option.

COMMENT = Comment for object

Use the comment option to specify a text description for a form, procedure, or query object. Comments are useful for preserving information about the object. Because comments usually include embedded blanks, they must be enclosed in apostrophes. Apostrophes within a comment must be specified by two adjacent apostrophes.

Examples:

```
COMMENT = 'My Form'
COMMENT = 'This form doesn''t include breaks'
```

In Query Management/400 a comment may be a maximum of 50 characters.

The IMPORT command creates a query management object from a source member into the database. For SQL queries and procedures, each record in the file becomes a separate line in the object. All files exported using the query management EXPORT command can be imported.

The AS/400 system does not support variable length records. Therefore, if you are importing an object with variable length records, the externalized object is con-

verted to fixed length during the transfer to the AS/400 system. Forms must have a fixed-length record of 150 bytes.

When importing files containing SQL queries and procedures, query management accepts records having a logical record length greater than 79 bytes, but text exceeding position 79 is truncated. If query management finds a logical record length greater than 79 bytes, it displays a warning message.

If the imported file has a fixed record format (and logical record length greater than 79 bytes), query management accepts only data in positions 1 to 79 and ignores all others.

When importing files with a logical record length less than 79 bytes, query management pads the record with blanks up to and including position 79. If the line contains an open string enclosed in quotation marks, this padding is included within the string and may cause unexpected results.

When importing query and procedure objects, query management does not perform any validation or checking on the contents of the files. Therefore, it is possible to create query and procedure objects containing characters that cannot be displayed. This could happen, for example, if a program's object file were imported as a query. Also, it is possible to import SQL statements from a query into the procedure object and vice versa.

Query management validates form objects. If some part of the file fails a validation test, the object is brought into the database, but you are sent warning messages. It is possible for the file to be in a state that passes the validation test, but produces unpredictable results when used for formatting.

You can use the extended parameter list format for this command. For more information on this format, see "GET" on page 2-4.

Examples of the IMPORT Command

The following examples show how to use the IMPORT command in query management:

```
IMPORT FORM REPORT1 FROM REPT1EX

IMPORT QUERY SALARYWK FROM JENSON

IMPORT QUERY JONES/SALARYWK FROM JENSON/PAYROLL
```

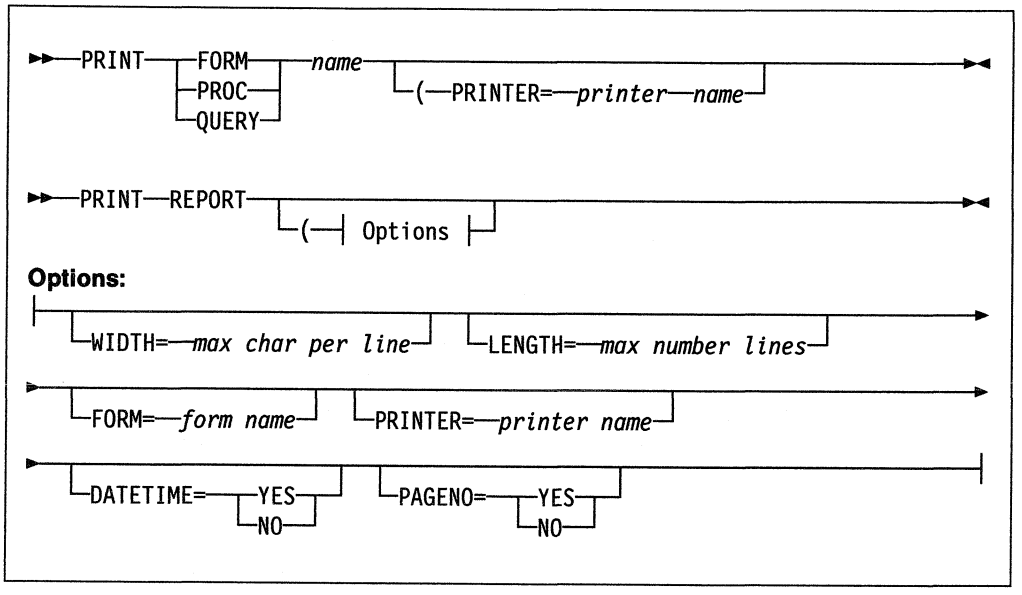
PRINT

Use the PRINT command to print the contents of a query management object or to print a report.

The PRINT command uses standard system functions for printing. Query management does not externalize the printer attributes to your application, nor does it alter these attributes' values. Instead, it uses the printer definitions currently in effect.

An object's printed appearance is very much like its displayed appearance. However, the following differences appear between the display and formats of a report:

- The title line of the report object in the display format is replaced with a page heading at the top of each page in the printed report (assuming that a page heading has been defined).
- A page footing appears at the bottom of each page of the printed report, but only once at the bottom of the displayed object.



name

The *name* variable specifies the object to be printed. The name specified may be a form, procedure, or query in the database.

This name can be a qualified name of the form library/object or database.object. Specify the naming convention you intend to use in your initial query command procedure.

You must have *ALL, *CHANGE, or *USE authority to the object you are printing, and you must have *ALL, *CHANGE, or *USE authority to the library in which the object resides.

WIDTH = *maximum characters per print line*

The value of WIDTH must be an integer between 22 and 378.

If the object is wider than the default printer width, the lines in the printout are truncated on the right. To avoid truncation, set the printer file QPQXOBJPF to FOLD = *YES. You can run a PRINT PROC command against a procedure file that has a record length greater then 79 bytes.

It is important that you ensure the compatibility of WIDTH with the printer you are using. For example, if your current printer settings identify a 10-pitch device (10 characters per inch) mounted with 8.5-inch-wide paper, then a WIDTH value of 132 results in truncated output. Because query management does not know the width of the physical printer, no message is shown when truncation occurs.

If you do not specify this option, query management uses the page width from the printer file associated with the query session. See "Printer File Use" on page 2-10 for more information on query management printer file use.

LENGTH = *maximum number of lines per page*

The value of LENGTH must be an integer ranging from 1 through 999.

When an object is to be printed and the value for LENGTH is inadequate (if the

value for LENGTH is less than the total number of lines needed for column headings, page headings and footings, plus the line needed to print the page number, date, and time), an error message is generated and the object is not printed.

For LENGTH values within the range allowed, query management performs a page eject whenever the number of lines of object data printed on a page is equal to LENGTH.

If you do not specify this option, query management uses the page length from the printer file associated with the query session. See "Printer File Use" on page 2-10 for more information on query management printer file use.

FORM = *form name*

The *form name* specifies the form you want to use to format your data.

This name can be a qualified name of the form library/object or database.object. Specify the naming convention you intend to use in your initial query command procedure.

PRINTER = *printer name*

The *printer name* specifies the printer that produces the output.

If you do not specify this option, the output is directed to the device specified in the printer file associated with the query session unless the default for the global variable DSQAPRNM is changed on the START command. See "Printer File Use" on page 2-10 for more information on query management printer file use.

DATETIME = YES | NO

This option controls the generation and display of the system date and time on the bottom of each page. When you specify DATETIME = YES, the date and time are placed on the last line of each page. When you specify DATETIME = NO, the system date and time do not print. The default for this option is YES.

PAGENO = YES | NO

This option controls the printing of page numbers on the last line of each page. The default for this option is YES.

You can use the extended parameter list format for this command. For more information on this format, see "GET" on page 2-4.

Examples of the PRINT Command

The following examples show how to use the PRINT command in query management:

```
PRINT QUERY Q1 (PRINTER=PRT1

PRINT QUERY DB1.Q1 (PRINTER=PRT1

PRINT FORM FORM1

PRINT PROC LIBA/PROCA

PRINT PROC LIBA/PROCA(MBRA)

PRINT PROC PROCA

PRINT REPORT

PRINT REPORT (WIDTH=80 LENGTH=60 DATETIME=YES PAGENO=YES
```

Printer File Use

Default printer files called QPQXOBJPF and QPQXPRTF are included as part of query management and are in the QSYS library. These printer files are used when a PRINT QUERY, PRINT PROC, or PRINT REPORT command is issued. The printer file QPQXOBJPF has page length and width defaults of 66 lines and 132 characters, respectively. The printer file QPQXPRTF has page length and width defaults of 66 lines and 80 characters, respectively. The printer device name specified by the printer file is *JOB, which lets all printer output be directed to the printer set up for the job. Unless overridden, the printer files QPQXOBJPF and QPQXPRTF are used by query management for formatting the printed objects and report.

You can direct query management to use a different printer by specifying a value on the PRINT command or by changing the default value DSQAPRNM on the START command from *SAME to a printer name or *JOB.

You can direct query management to use a different printer file by using the Override Printer File (OVRPRTF) CL command.

You can use the Change Printer File (CHGPRTF) CL command to permanently change the query management printer files QPQXOBJPF and QPQXPRTF. To use SAA defaults again, issue another CHGPRTF CL command to change the attributes back.

On every install, the printer files are created again in the QSYS library. All changes to the printer files must be applied again. To save changes to a printer file, you can create your own printer file in your library with the desired attributes and use the OVRPRTF CL command to direct query management to this printer file. You can also copy the printer files to your own library and make the changes to the copy in that library. To use a printer file with the same name as the one in the SAA Query library, your library must be in the library list before the SAA Query library.

Print Object Formatting

While processing the PRINT QUERY and PRINT PROC commands, query management formats the printed output into 132 column lines. The column lines are broken down into 123 bytes of text and 7 bytes for the line number, which is generated during the PRINT command.

The width of 132 is wide enough to handle the printing of most files with ease and is compatible with most AS/400 printers.

Directing the printer output to a printer with a line width less than 132 characters results in possible loss of data unless the printer file has *YES specified for the Fold Record parameter. The default for the Fold Record parameter in the QPQXOBJPF printer file is *NO.

Print Report Formatting

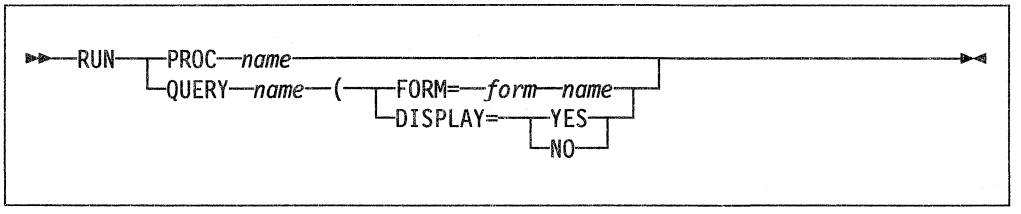
While processing the PRINT REPORT command, query management formats the printed report using the width specified on the PRINT command or the default from the printer file. If the report is wider than the print WIDTH, it is split between pages. In this case, multiple printer files are opened, and each segment of each report line is directed to the appropriate opened printer file.

If the report is directed to a printer with a width less than the WIDTH specified on the PRINT command or in the printer file, each print record is truncated. If the Fold Record option in the printer file is changed from the default to *YES, each print record is wrapped. For example, a report that formats to 200 columns when printed with a command of PRINT REPORT (WIDTH=200 PRINTER=xyz, results in line wrapping if the printer width is less than 200. The Record Wrap option on the printer file has been overridden to *YES. If the Record Wrap option is not overridden, the rightmost columns in the report are truncated.

RUN

The RUN command processes a procedure or a query. When you issue the RUN command, you must identify the procedure or query you want to run. Therefore, the procedure or query must exist prior to the start of RUN.

When used to run a query (SELECT only), the RUN command produces new data that replaces the existing data from the previous RUN QUERY.



name

The *name* variable specifies the query or procedure to be run.

This name can be a qualified name of the form library/object or database.object. Specify the naming convention you intend to use in your initial query command procedure.

You can only run a query or procedure to which you have *USE authority to the query and form. You must also have *CHANGE or *ALL authority to the library in which the objects reside.

FORM = *form name*

Use the *form name* only for queries that contain a SELECT statement.

The FORM option specifies the form to be used in formatting the report that automatically displays when running interactively. This option is ignored on the RUN command if the job is running in batch mode.

If a RUN command with a form specified is followed by a PRINT command with no form specified, the form specified on the RUN command is applied on the PRINT command.

This name can be a qualified name of the form library/object or database.object. Specify the naming convention you intend to use in your initial query command procedure.

If the form specified by the FORM option cannot be found (when the form does not exist), the RUN command is rejected with an error message. If the form does exist but does not work with the data (as when different data types for the columns are specified), query management responds with an error message.

If you omit the FORM option, a default form is used.

If the form or query specified is not found, and DSQSCNVT = YES is specified on the START command, query management searches for a Query/400 definition with that name. If a query definition is found, the information is used to create a query or form that query management can use.

DISPLAY = YES | NO

Use the DISPLAY keyword to indicate whether to display the report. This keyword defaults to Yes if you are processing interactively. If you specified Batch mode on the START command, this keyword is ignored.

You can use the extended parameter list format for this command. For more information on this format, see "GET" on page 2-4.

Examples of the RUN Command

The following examples show how to use the RUN command in query management:

```
RUN QUERY QN1

RUN PROC WEEKREPT

RUN QUERY SMITH.Q6 (FORM=SMITH.SAL_REPT

RUN QUERY SMITH/Q6 (FORM=SMITH/SAL_REPT
```

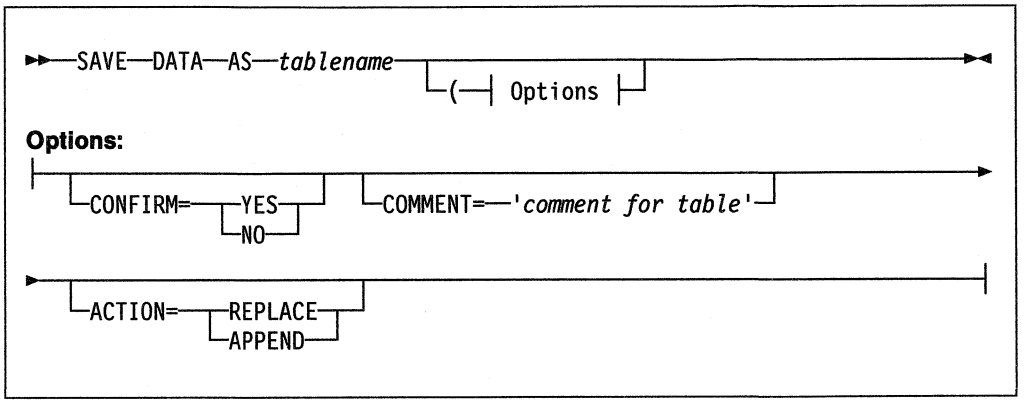
SAVE DATA AS

Use the SAVE DATA AS command to save data in a table in the database. The saved table object is named according to the name you specify with the command.

If data is saved and a table or view is being replaced, the data must be compatible with the existing definition. Compatible data has matching data types, lengths, and null attributes. The number of columns in the data must match the target. If the two objects are incompatible, query management rejects the SAVE DATA AS command and the database remains unchanged.

The column names for a saved table that does not already exist are generated by query management using the same algorithm used in generating the default column headings in the form object. You cannot change the column names.

Query management allows for the replacement of data in any file that is compatible with the data to be saved. The file does not have to be a table in a collection if the table name specified exists. However, the library in which the table is to be created must be a collection.



tablename

The *tablename* variable specifies the table or view in which the data is stored in the database.

If the table name specified does not exist in the database, a new table is created. This name can be a qualified name of the form library/object or database.object. Specify the naming convention you intend to use in your initial query command procedure.

To save the data to a table, you must have proper SQL authority to change or create a table. Refer to the *SQL/400* Reference* manual for table authorization rules.

SQL/400 conventions allow a view to be updated only if it is associated with just one table. Updating a view on multiple tables is not allowed, nor is updating a view associated with a table that is an SQL catalog.

CONFIRM = YES | NO

This option provides a check before performing your SAVE request. The confirmation request occurs only when an existing object in the database is about to be replaced.

If your job is running interactively, an inquiry message is sent to your display station, and the job is suspended until you respond to the message. The message asks whether you want to save the object. If your job is running in batch mode, or DSQSMODE was set to Batch during START processing, CONFIRM = YES results in an error.

CONFIRM = YES forces the system to display the confirmation message. CONFIRM = NO suppresses a display of the confirmation message and saves the object. The default for this option is CONFIRM = YES, unless the option is changed during START processing by setting the value of the query-defined variable DSQCONFIRM in the query command procedure.

When the SAVE DATA AS command is issued and there is already an object in the library with the same name as the object to be saved, the system returns the message:

```
Object exists. Do you want to replace it?
```

This message is displayed only if the CONFIRM=YES option is specified or if you omit the CONFIRM option.

COMMENT=*comment for table*

Use the COMMENT option to supply a comment when saving data as a table. Comments are useful for preserving descriptive information about the object.

Because comments usually consist of multiple words and embedded blanks, you must enclose them in apostrophes. You must enter apostrophes embedded within the commentary as two adjacent apostrophes.

The following examples illustrate proper quotation:

```
COMMENT='The primary EMPLOYEE table-see John (X3971)'

COMMENT='Don''t ERASE this data without telling Phil!'
```

Query management restricts object comments to a maximum of 50 characters, excluding the apostrophes.

ACTION=REPLACE | APPEND

Use the ACTION option to indicate whether an existing table or view is to be replaced or if the data is to be added to the end. The default is ACTION=REPLACE. The ACTION keyword is ignored if the table or view does not exist.

You can use the extended parameter list format for this command. For more information on this format, see "GET" on page 2-4.

Examples of the SAVE DATA AS Command

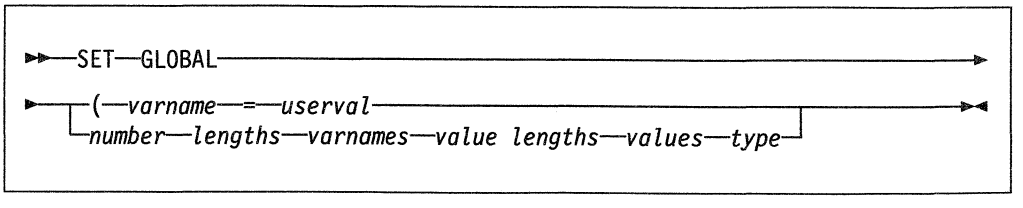
The following examples show how to use the SAVE DATA AS command in query management:

```
SAVE DATA AS EMP12

SAVE DATA AS EMP12 (COMMENT='CLASSIC TWO TABLE JOIN')

SAVE DATA AS DB1.EMP12 (COMMENT='CLASSIC TWO TABLE JOIN')

SAVE DATA AS LIB1/EMP12
```

SET

Use the SET command to set the value of a query management variable from the user program or procedure. Use the short version of the command syntax when using the SET command from a procedure. Use the short version or the extended version of the command syntax when using the SET command from a program. If you use the short version of the command syntax from a program, the SET command runs as it does from a procedure.

A maximum of 1000 variables may be set in a single query instance.



GLOBAL
The GLOBAL option specifies the *varname* variable located in the query management global variable pool to be returned to the requester. If the variable is not found in the global pool, a new variable is created. If the variable does exist, its contents are replaced.

varname
The *varname* variable specifies the name of the variable located in the query management variable pool. For rules that apply to variable names used across the callable interface, see the section on extended variable support in the *SAA CPI Query Reference* manual.

userval
The *userval* variable specifies the value to associate with the variable name specified by *varname*. If it is a constant enclosed in single quotation marks, the single quotation marks are removed.

Extended Parameter List

Use the following parameters to further qualify the SET command:

number of varnames
Number of *varnames* requested for this call.

varname lengths
Length of each *varname* that is specified.

varname Name of the variable located in the query management variable pool.

user value lengths
Length of the program storage area that is to contain the *varname* value.

user value
Name of the program storage area that is to contain the *varname* value.

value type
Data type of the storage area that is to contain the *varname* value.

Examples of the SET Command

The following are examples of how to use the SET command in a procedure or with the short command syntax through a program:

```
SET GLOBAL (CHARVAR = 'abc'

SET GLOBAL (NUMBVAR = 199

SET GLOBAL (NAMEVAR = MYTABLE

SET GLOBAL (CHARVAR='abc' NUMBVAR=199 NAMEVAR=MYTABLE
```

Quotation Marks in *varname* Values

Use two adjacent single quotation marks to represent a quotation mark in a character string *varname* value if the variable is set with the short command syntax. Use a single quotation mark in a character string *varname* value to represent a quotation mark if the variable is set through the extended parameter list format.

Programming Considerations

The SET command is useful for selecting run-time records. You do not need to save numerous QMQRY objects with different SELECT fields or WHERE conditions.

For example, you may want to run a query on a file that contains records for employees whose names start with letters in the first part of the alphabet. You may also want to run the same query with records for employees whose names start with letters in the last part of the alphabet. Your query object, named EMPREPORT, could contain the following SQL SELECT statement:

```
SELECT  NAME,DEPT,EMPNO  FROM  MASTER
        WHERE  NAME  =  &STARTAL  AND  NAME  <  &ENDAL
```

You could then set up a procedure with the statements:

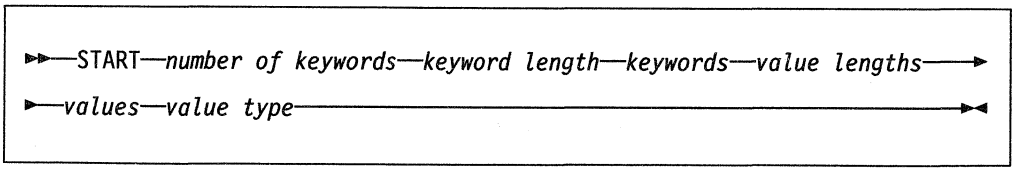
```
"SET  GLOBAL  (STARTAL='''A'''"
"SET  GLOBAL  (ENDAL='''J'''"
"RUN  QUERY  EMPREPORT"
"SET  GLOBAL   (STARTAL='''K'''"
"SET  GLOBAL   (ENDAL='''Z'''"
"RUN  QUERY  EMPREPORT"
```

You could also run the EMPREPORT query with the Start Query Management Query (STRQMQRY) CL command from an interactive mode. You are prompted for variables STARTAL and ENDAL before the SELECT statement is performed.

START

The START command provides an interface to create an instance of query management for a job. This command is only valid when issued through the callable interface. The START command allows for values to be specified that indicate how the query management instance is to be started.

If there is a conflict in job modes (interactive or batch) between a job running from an AS/400 job queue, the DSQSMODE keyword, or the DSQSMODE variable in an initial query command procedure, the job is always run in batch mode.



Extended Parameter List

Use the following parameters to further qualify the START command:

number of keywords
Number of keywords that are passed with this call.

keyword length
Length of each specified keyword.

keywords Name of the START command keyword that is being set.

The following keywords are used on the START command in query management:

DSQSMODE

This keyword indicates the mode of query management operation when subsequent commands are issued. Valid options are:

INTERACTIVE This option allows the display of reports and messages during query management processing. Any reports generated as a result of a RUN QUERY command are shown on your display. Any confirmation messages requiring a response are shown on your display, and you can then reply to the message.

BATCH No displays are shown during query management processing. Any messages requiring a response result in an error. All other messages are sent to the job log.

The keyword value set for the DSQSMODE variable on the START command overrides any keyword value set for the DSQSMODE variable by the query command procedure.

DSQSCMD

This keyword is the name of a query command procedure that is used to set up the query management instance. The SET command is the only type of statement allowed in this procedure. If the DSQSNAME keyword is not specified on the START command, *SAA conventions are used to find the procedure. Otherwise, the naming conventions set by the DSQSNAME keyword are used. If you do not specify the DSQSCMD keyword, query management searches for a default query command procedure called DSQSCMDP.

DSQSRUN

This keyword names the query management procedure to run after initialization is started.

DSQSNAME

This keyword specifies the naming convention to be used when processing query management commands and the SQL query. The keyword value set for DSQSNAME on the START command overrides any keyword value set for DSQSNAME in the query command procedure.

***SAA** Any qualified query management object name specified in commands or procedures is of the format database.object.

***SYS** Any qualified query management object name specified in commands or procedures is of the format library/object.

DSQSCNVT This keyword indicates whether query management searches for a Query/400 definition object if a query management object is not found or is not to be used.

The information contained in the query definition is used to create a temporary query management object to be used in a command.

For example, the command `RUN QUERY MYLIB/QRY1 (DSQSCNVT=YES` tells query management to search for a QMQRY object named QRY1. If that object is not found, query management searches for a QRYDFN object and uses the information contained in it to run a query.

YES Query management does search for a Query/400 definition object if a query management object is not found.

NO Query management does not search for a Query/400 definition object if a query management object is not found.

ONLY Query management searches only for a Query/400 definition object.

DSQOAUTH

This keyword indicates the authority given to any object created by query management.

You can specify a default public authority for all objects created during a query instance by setting a value in the DSQOAUTH keyword in the query command procedure or on the START command. The values you can specify are:

***LIBCRTAUT** *LIBCRTAUT authority allows the same authority for the object as specified on the CRTAUT parameter of the library in which the object is being created. If the CRTAUT parameter is changed, the new value will not affect the authority of existing objects.

***CHANGE** *CHANGE authority allows other users to perform all operations on the object except those limited to the owner or controlled by object existence authority and object management authority. A user can change or use the query object in any way, except for deleting it or transferring it to a new owner.

***ALL** *ALL authority allows other users to perform all operations on the object except those limited to the owner or controlled by authorization list management rights. A user can do anything with the query object (including erasing it) except for transferring it to a new owner.

***EXCLUDE** *EXCLUDE authority prevents other users from doing anything with the query management object. Unless given specific types of authority, no user except its owner can use the query management object.

***USE** *USE authority allows other users to run, export, or print the query management object, but prevents them from importing it or saving to it.

authorization list name

If you specify the name of an authorization list, its authority is used to control the user's ability to use a query management object. For more information, see the *Security Concepts and Planning*

If you do not specify an authority through the query command procedure, other users have *EXCLUDE access to the query object.

value lengths Length of the program storage area that is to contain the keyword value.

values Name of the program storage area that is to contain the keyword value.

value type Data type of the storage area that is to contain the keyword value.

Examples of the START Command

See "START Subprogram" on page 6-1 for an example of a program that uses the START command.

Query Management Query Command Procedure

Use the DSQSCMD keyword on the START command to specify the name of the query command procedure that is run as part of query management initialization. This procedure supplies default START parameters that are related to the environment in which query management is to run.

The query command procedure used on the DSQSCMD parameter is the only place where users can set DSQ variables. The following DSQ variables can be set using the query command procedure:

- DSQSMODE
- DSQSRUN
- DSQOAUTH
- DSQSNAME
- DSQCONFIRM
- DSQAPRNM

Any other DSQ variables set in the query command procedure are ignored. Defaults are applied to all DSQ variables that are not set in the user-supplied procedure. The possible parameters for the DSQ variables that users can set using the query command procedure are:

DSQSMODE = INTERACTIVE | BATCH

This parameter indicates the mode of query management operation when subsequent commands are issued. Valid options are:

INTERACTIVE

Allows displays to be shown during query management processing. Any reports generated as a result of a RUN QUERY command are shown on your display. Any confirmation messages requiring a response are shown on your display, and you can then reply to the messages.

BATCH

Does not show displays during query management processing. Any messages requiring responses result in errors. All other messages are sent to the job log.

DSQSRUN = *query procedure name*

The query procedure name names the query management procedure to run after initialization is started.

If DSQSRUN parameter is not specified on the START command and the DSQSRUN variable is not set in the query command procedure, no initialization procedure is run.

DSQOAUTH = *CHANGE, *EXCLUDE, *USE, *ALL, or authorization list name

If DSQOAUTH is not set by the query command procedure, it defaults to *EXCLUDE.

DSQSNAME = *SYS | *SAA

This parameter specifies the naming convention to be used when processing query management commands.

***SYS**

Use the format library/object to specify any qualified names in commands or query management procedures.

***SAA**

Use the format database.object to specify any qualified names in commands or query management procedures.

If the DSQSNAME parameter is not specified on the START command and the DSQSNAME variable is not set in the query command procedure, the naming convention defaults to *SAA.

DSQCONFIRM = YES | NO

This keyword specifies the default to be taken when CONFIRM is not specified on a command that allows for confirmation processing (IMPORT, EXPORT, and SAVE DATA). If this DSQ variable is not specified in the query command procedure, the default is DSQCONFIRM = YES.

DSQSCNVT = YES | NO | ONLY

This keyword indicates whether query and form information may be derived from a Query/400 definition (QRYDFN) if query management object information is not available. Specifying NO for this keyword causes the command to end with an error if the query or form specified on an EXPORT, PRINT, or RUN command is not found.

Specify Yes for this keyword to request query management to attempt to use Query/400 information if the query or form specified on an EXPORT, PRINT, or RUN command is not found. If this keyword is not specified on the START command, it defaults to DSQSCNVT = NO.

Example of the Query Management Command Procedure

The following is an example of the contents of the default procedure included with the product:

```
'SET GLOBAL (DSQSMODE=BATCH'
'SET GLOBAL (DSQOAUTH=*EXCLUDE'
'SET GLOBAL (DSQSNAME=*SAA'
'SET GLOBAL (DSQCONFIRM=YES'
```

CL Commands

The following control language (CL) commands are commonly used when working with query management and writing applications to create query management reports. For further information on using CL commands, see the *CL Reference* manual.

ANZQRY (Analyze Query) Command

The Analyze Query (ANZQRY) command lets you analyze a Query/400 definition (QRYDFN) object for query management conversion problems. Query management returns diagnostic messages to your display station. These messages detail potential differences between how Query/400 and query management use query and form information derived from the analyzed QRYDFN object. A completion message shows the highest severity of the problems that are found.

CRTQMFORM (Create Query Management Form) Command

The Create Query Management Form (CRTQMFORM) command lets you create a query management form from a specified source. The form defines how to format DATA (from running a query) when printing or displaying a report. Form information is encoded in source file member records.

CRTQMQRY (Create Query Management Query) Command

The Create Query Management Query (CRTQMQRY) command lets you create a query from a specified source. A query is any single SQL statement that can contain variable substitution values. The query can be spread over multiple records in a source file member.

DLTQMFORM (Delete Query Management Form) Command

The Delete Query Management Form (DLTQMFORM) command lets you delete an existing query management form from a library. Use a generic form name to delete multiple forms from a library or list of libraries.

DLTQMQRY (Delete Query Management Query) Command

The Delete Query Management Query (DLTQMQRY) command lets you delete an existing query management query from a library. Use a generic query name to delete multiple queries from a library or list of libraries.

RTVQMFORM (Retrieve Query Management Form) Command

The Retrieve Query Management Form (RTVQMFORM) command lets you retrieve encoded form source records from a query management form (QMFORM) object. The source records are placed into a source file member that can be edited.

You can also retrieve form source records from a QRYDFN object.

RTVQMQRY (Retrieve Query Management Query) Command

The Retrieve Query Management Query (RTVQMQRY) command lets you retrieve an SQL source statement from a query management query (QMQRY) object. The source records are placed into a source file member that can be edited.

You can also retrieve query source records from a QRYDFN object.

STRQMPRC (Start Query Management Procedure) Command

The Start Query Management Procedure (STRQMPRC) command lets you run a query management procedure that was saved as a member in a source file.

STRQMQRY (Start Query Management Query) Command

The Start Query Management Query (STRQMQRY) command lets you run an existing query management query. The query runs the SQL statement saved in the query management query. The DATA collected from running an SQL SELECT statement can be displayed, printed, or stored in another database file.

You can also derive the SQL statement or the information for formatting displayed or printed output from a QRYDFN object.

WRKQMFORM (Work with Query Management Forms) Command

The Work with Query Management Forms (WRKQMFORM) command shows a list of query management forms from a user-specified subset of query management form names. Several query management form-related functions are available from this list.

WRKQMQRY (Work with Query Management Queries) Command

The Work with Query Management Queries (WRKQMQRY) command shows a list of query management queries from a user-specified subset of query management query names. Several query management query-related functions are available from this list.

Chapter 3. Working with Query Management Programs

This chapter describes the functions available in query management to form and run programs that write query reports.

Callable Interface

The SAA callable interface (CI) provides the ability for application programs to run query management functions through calls to the SAA Query Interface. After completion of a query management function, return code and status information is available to the calling program. The CI is supported by query management for C, COBOL, and RPG languages:

The CI consists of the following elements:

- SAA CI macroinstructions

  The SAA CI macroinstructions are comprised of the include and macro files used when application programs that call the SAA CI modules are compiled. They contain the declarations for the communications area structure and any constants that are required to update and access the communications area structure. They also provide a standard interface from different programming languages to SAA CI modules. The interface provides common storage and access of program variables between the programming language and query management. One SAA CI macro or include is provided for each language that query management supports.

  The following macro include packages are available for query management:

  - C

    Library QCC

    File H

    Member DSQCOMMC

  - COBOL

    Library QLBL

    File QILBINC

    Member DSQCOMMB

  - RPG

    Library QRPG

    File QIRGINC

    Member DSQCOMMR

  Before compiling an application program that uses these includes or macros, copy the member to the default include file used by the compiler. This allows the include to be used by the application program without being qualified with the library or file, which maintains a higher degree of portability.

  You can code application programs to qualify the include with the library and file name. This ensures the program is compiled with the newest version of the include.

- Query management

© Copyright IBM Corp. 1991

Query management provides query and report writing services.

- SAA CI modules

  Modules are provided by the CPI to allow access to query management functions. These modules are:

  DSQCICE The C language interface module for extended parameter lists

  DSQCIC The C language interface module for nonextended parameter lists

  DSQCIB The COBOL language interface module

  DSQCIR The RPG language interface module

Callable Interface Description

The CI is an interface that programming languages use to run query management commands. All query management commands are supported through the CI. See the *SAA CPI Query Reference* manual for more details on the CI.

Interface Communications Area (DSQCOMM)

The query management CI communications area (DSQSCOMM) is required on all CI calls. The program allocates storage for the CI communications area using the query management CI. A unique communications macro is defined for each supported language.

Return Variables

For more information on return codes and variables, see the *SAA CPI Query Reference* manual for a description of the areas supported by query management.

Return Codes: Return codes are returned after each call to query management CI. Return code values are described by the data interface. For applications to be portable, values must be referred to by variable name rather than the equated value, since this value may be different on other systems. Return code values for DSQRET are:

DSQSUC Successful process of the request.

DSQWAR Normal completion with warnings.

DSQFAI Command did not process correctly.

DSQSEV Severe error; query management session ended for the applicable instance. Because the session ended, additional calls to query management cannot be made using this instance ID.

Global Variable Support

A variable is a named entity within query management that can be assigned a value. Global variable support allows applications to define global variables within query management.

You can use variables as substitution values in SQL queries and when using the CI. Once a variable is created, it is available to the query management session for the life of the session. In addition to application-defined variables, query management maintains a set of product variables. You can also use these variables in SQL queries and the CI.

Creating Variables

You must create application-defined variables using the SET GLOBAL command. An attempt to run an SQL query that refers to a variable that is not set results in an error.

Referring to Variables

Refer to variables by specifying the variable name in an SQL query or a user program through the GET GLOBAL command. When a variable name is referred to in an SQL query, you must prefix the variable name with an ampersand (&) so query management can recognize it as a variable. For example:

```
SELECT * FROM &TNAME
```

The value &TNAME is considered a query management variable. There are no specific rules regarding where a variable name may be specified in an SQL query. As an extreme case, you can do a SET GLOBAL command to set a variable to a character string containing an SQL statement and an SQL query containing nothing but the variable reference can be run. For example, QUERY1 could contain the value &sqlstmt. A program then issues the command string:

```
SET GLOBAL (sqlstmt = 'SELECT NAME, DEPT FROM MYLIB/PAYROLL'
```

Note: The program does only one pass of variable substitution. Therefore, using a variable that contains another variable substitution string may result in an SQL error when processed. If an SQL error occurs, the error generated depends on the location of the unresolved variable name in the SQL statement.

Variable Names

For more information on the rules that apply when you use variables in SQL queries across the callable interface, see the *SAA CPI Query Reference* manual. Rules specific to the AS/400 system can be found on page 1-6.

Variable Values

The following values are valid for the query management variables listed:

Character variables

- Character variable values consist of any value up to 55 characters long.
- A GET of a character variable into a smaller character field is allowed. The character string is truncated on the right after 55 characters.
- A GET of a character variable into a larger character field is allowed. The character string is left-adjusted and blank-padded. The null character at the end of a C string is not moved.
- A C null character is inserted at the end of the string if the GET was done through the C language callable interface.

Integer variables

- Integer variable values must be 4 bytes long. An attempt to SET or GET an integer with a length other then 4 bytes results in an error.
- Integer variable values are assumed to be signed.
- The value must observe SQL/400 rules when used in an SQL query.
- An integer value is converted to a character string without leading or trailing blanks prior to substitution into the SQL statement. Do not use an integer vari-

able if an implied result field width is needed or if you are using variable substitution while defining the result field in the SQL statement.

Query Management-Defined Variables

Query management provides global variables for user programs. You can retrieve these variables by using the GET GLOBAL command in your program. Use the current set of query management variables to determine the current status of query management environments and particular objects. Query management variables cannot be altered by users, programs, or procedures. Set a subset of these variables by using the query command procedure specified on the START command.

The following variables are available in query management:

DSQAAUTH — Current connect authorization ID. This name contains the name of the user profile under which the job is running.

Type Character

Length 10

Value —

DSQOAUTH — Default object public authority to be given to objects created through query commands.

Type Character

Length 10

Value One of the following:

- *LIBCRTAUT
- *EXCLUDE
- *ALL
- *USE
- *CHANGE
- An authorization list name

DSQSNAME — Naming convention be used. See "START" on page 2-16 for a description of this variable.

Type Character

Length 4

Value One of the following:

- *SAA
- *SYS

DSQAPRNM — Current default printer nickname. This name is set to the printer device name specified in the user profile of the job at the time the query management session was started.

Type Character

Length 10

Value —

DSQCATTN — Last command cancel indicator.

Type Character

Length 1

Value One of the following:

- 0 — Command completed
- 1 — Command canceled

DSQCISQL Last SQL return code.

Type Integer

Length 4

Value See the appendix on SQLCODES in the *SQL/400* Programmer's Guide*.

DSQAROWS Current number of rows fetched for data.

Type Integer

Length 4

Value 0 to maximum number of rows

DSQAROWC Current data is completed.

Type Character

Length 1

Value One of the following:

- 0 — Data is not complete
- 1 — Data is complete

DSQSMODE Current processing mode.

Type Character

Length 11

Value One of the following:

- Batch
- Interactive

DSQCONFIRM Confirm processing default.

Type Character

Length 3

Value One of the following:

- YES
- NO

DSQSCNVT Conversion processing default.

Type Character

Length 4

Value One of the following:

- YES
- NO
- ONLY

DSQCIMNO The query management message ID. It is the same value that is returned on the query management message line in the communications area.

Type Character

Length 8

Value —

DSQCIQNO — The message ID. It is the same value that is returned on the completion message line in the communications area.

Type Character

Length 8

Value —

DSQCIMSG — Contains the message text as it is displayed to the user interactively.

Type Character

Length 55

Value —

DSQCIQMG — Contains the query message text as it is displayed to the user interactively.

Type Character

Length 55

Value —

Query Capability

Query management supports queries against relational data using SQL/400 conventions. The basic statements, SELECT expressions, data definitions, and authorization statements defined in the *SQL/400* Reference* manual are specifically supported. You can manipulate queries by processing query management commands such as IMPORT, EXPORT, PRINT, RUN, and ERASE. You can also manipulate them as OS/400 objects through AS/400 CL commands.

Stored queries allow flexibility in two ways. First, you can change a query and store it independently from your application program. Second, queries can contain variables. The values assigned to the variables can be set prior to or in conjunction with your application. Both of these functions allow data query parameters to be changed without writing or compiling your application again.

Rules for Creating Queries

The only way to create a query in query management is by doing an import of an externalized query source file to create a query management query object. The query management query object is stored as a QMQRY OS/400 object. The externalized query source file must contain a text string containing an SQL statement. The SQL statement can optionally contain variables. A variable can appear in any clause of the query and can represent anything that can be written into a query, such as column names, search conditions, subselects, or specific values, as well as multiple and partial clauses.

The following is an example of a query:

```
-- This query lists the name, years of employment, and salary for
-- employees in a specified department.  The department is a variable
-- and should be set before the query is run.
   SELECT NAME, YEARS, SALARY        -- names the columns used
   FROM Q.STAFF                      -- names the table used
   WHERE DEPT=&DEPTNUM               -- variable selection condition
```

Query management strips the comments and performs variable substitution before passing the query to the database manager for processing.

The following restrictions apply to queries handled by query management:

- The externalized query source file should contain only an SQL/400 statement. An externalized prompted query would successfully import to query management but would not run.
- Substitution variable values can be up to 55 bytes long.
- Comments are preceded by a double hyphen (--). Everything between the double hyphen and the end of the line is considered a part of the comment.
- The total query cannot exceed 32KB minus 1 byte (after you remove comments and blanks and make variable substitutions).
- An externalized query may optionally contain an H record and a comment V record immediately following the H record. The comment may be used as the text description when the object is imported.

Variable Substitution

The following rules apply to variable substitution in query management queries:

- Variable substitution is not done if the variable appears within a comment.
- Variable substitution is not done if the variable appears within a constant or a delimited name.
- A variable within an SQL query is defined as a string of characters that begins with an ampersand (&) and ends with any character that is not a valid variable name character.
- Query management does not substitute extra blanks between variables. Therefore, you can use variable substitution as a concatenation device. As an example, the following query management SET commands are processed through the callable interface or within a procedure:

  ```
  SET GLOBAL (library='MYLIB'
  SET GLOBAL (table='MYTABLE'
  SET GLOBAL (dol=10
  SET GLOBAL (cnts=50
  ```

  Then running the following SQL query processes the ending SQL statement:

  ```
  SELECT * FROM &library.&table
           WHERE PRICE  EQ &dol.&cnts

  SELECT * FROM MYLIB.MYTABLE
           WHERE PRICE  EQ 10.50
  ```

Variable Prompting

If your job is running in interactive mode and the variable specified in a query is not set in the global variable pool, query management sends a message to your display station prompting you for the value to be used. Enter a valid value for the specified variable, and then the query is processed.

If your job is running in batch mode and the variable specified in a query is not set in the global variable pool, query management sends a warning message to your display station. The variable is not substituted.

Comments

Comments in query management queries are handled in the following ways:

- Comments are stripped from the SQL query prior to variable substitution. Comment delimiters within substituted variables are not stripped and may result in SQL errors or unpredictable results.
- Comment delimiters within strings enclosed in quotation marks are not treated as comments. These strings may either be delimited names, which are delimited by quotation marks (""), or constants, which are delimited by apostrophes ('').
- You cannot use intervening blanks between the two hyphens (--) that make up the comment delimiters.

Line Continuations

The following rules govern the use of line continuations in query management queries:

- Some SQL clauses may span multiple lines of the SQL query. SQL does not support a line continuation character. Therefore, for readability, start a new line at a point in the SQL statement where a blank could be inserted. If a clause spans multiple lines, the clause may be split as long as the last character in the previous line is part of the clause and the first character in the next line is part of the clause with no extra blanks. For example:

```
SELECT  NAME, DEPT, SALARY, COMM
        FROM  LIBA/PAYROLL
        WHERE NAME = 'SMITH'
```

- Constants and delimited names may span multiple lines.

  Note: Mixed single-byte character set (SBCS) and double-byte character set (DBCS) character strings may not successfully be split between multiple lines. In this case, use SQL/400 concatenation.

Report Form Definition

This section describes query management reporting capabilities. You produce reports by formatting the results of a query using the formatting information that is specified in a form.

How Applications Can Use Forms

An application can create or alter a form by directly changing or creating the exported form.

You can use an application to export an existing form from query management, change it, import the form, and then format a report. You do not have to export the

form every time. An application can access and change an existing exported form, and then import it into query management for reporting.

You can also import a form from a source that allows certain form fields to be filled by default. It is possible to use only the header (H) record followed by the T and R records for column information. The remaining form fields are filled in by query management default values.

Saving a Default Form

Create a template for generating an external form object by making a source file with a record length of 162 characters (CRTSRCPF CL command). Create a member in this source file to contain minimal information. For example, the member DEFAULT in library MYLIB1 and file TESTFORM contains the following statement:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
H QM4 01 F 01 E V W E R 01 03 90/08/23 15:30
T 1110 001 000
R
E
```

This form contains only one column.

To create a default form in a library, use the Create Query Management Form (CRTQMFORM) CL command to import the source file member you just created. Use the Retrieve Query Management Form (RTVQMFORM) CL command to export the internal form created as a result of the CRTQMFORM command. The member created from the RTVQMFORM command contains a complete form for one column of data with all default values (except those set at run time) filled in. For example, you can use the following commands to create a source template form:

```
CRTQMFORM  QMFORM(MYLIB/QMFORMT)
           SRCFILE(MYLIB1/TESTFORM)
           SRCMBR(DEFAULT)

RTVQMFORM  QMFORM(MYLIB/QMFORMT)
           SRCFILE(MYLIB1/TESTFORM)
           SRCMBR(DFTCOMP)
```

The member named DFTCOMP created by these commands contains a complete form with all other values defaulted for a one-column query. You can then edit DFTCOMP to change field attributes and add more columns.

Formatting Terminology

To understand all of the options available in the form, Figure 3-1 on page 3-10 and Figure 3-2 on page 3-10 show the effects of some of the form options on a formatted report.

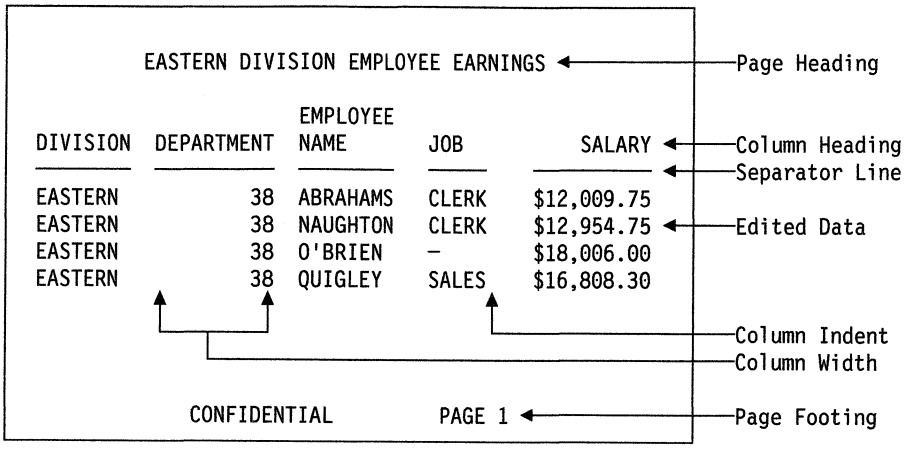


Figure 3-1. Basic Parts of a Report

Edited Data is information from the database that displays according to the relevant edit code or editing default.

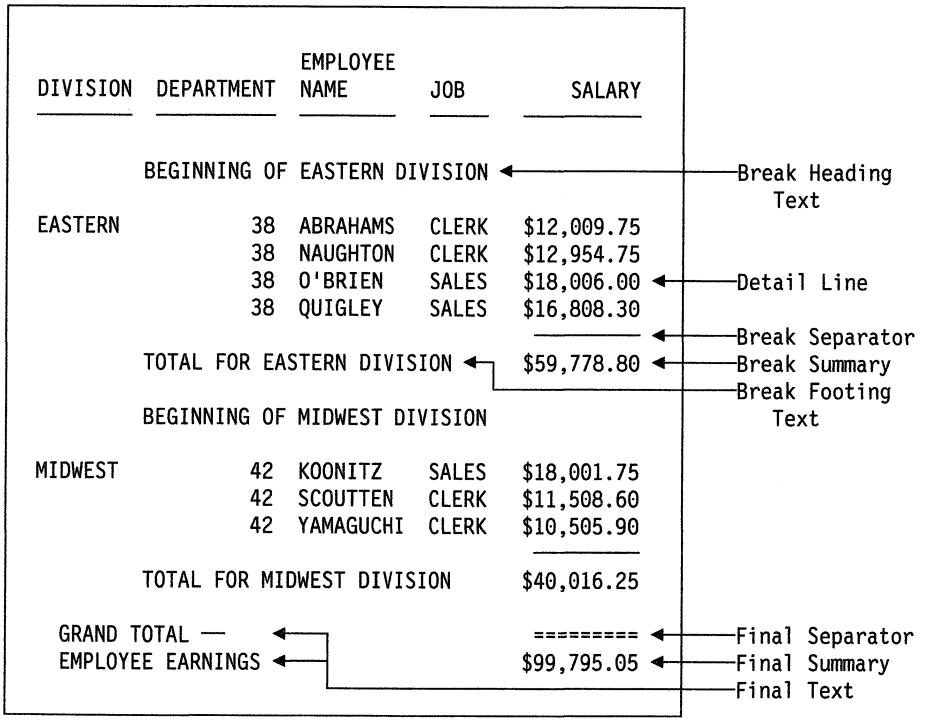


Figure 3-2. Basic Parts of a Report with One Level of Control Break

The form object contains fields that describe the report.

Defaults are provided for information that is not available. Some defaults are provided when the form is imported. Others defaults, such as Data type and Column heading, are provided at run time and depend on the resulting data of the processed query.

Keywords encoded into the form should be in uppercase English. Text fields (headings, footings, and final text) can be in upper- and lowercase letters.

The form object fields are commonly grouped by the following functional categories:

- Break
- Column
- Final
- Options
- Page

The following sections describe the form fields and how to use them to specify how results from running a query management query is formatted.

Break Fields

You can specify information for break levels one to six in the break fields. Change or specify the exported form by selecting the proper form field numbers for each of the break levels. Specify options for each break level in a similar manner. Each set of options is independent from the others.

Summary of Values for Break Information

Figure 3-3 shows the defaults and possible values for the attributes in the Break fields.

Figure 3-3. Default Values for Break Fields

Attribute	Default	Possible Values
New page for break	NO	YES, NO
New page for footing	NO	YES, NO
Repeat column heading	NO	YES, NO
Blank lines before heading	0	0 to 999
Blank lines before footing	0	0 to 999 or BOTTOM
Blank lines after heading	0	0 to 999
Blank lines after footing	1	0 to 999
Put break summary at line	1	1 to 999 or NONE
Alignment break heading text	LEFT	LEFT, CENTER, RIGHT
Alignment break footing text	RIGHT	LEFT, CENTER, RIGHT

New Page for Break and New Page for Footing

The *New page for break* and *New page for footing* fields indicate whether the subsequent part of the report begins on a new page. The default value is NO for both. When you specify YES for the *New page for break* field, the member lines for the break format on a new page. If you specify a break heading, it precedes the break member lines on the new page. When you specify YES for the *New page for footing* field, the break footing formats on the next page (if a footing exists).

Repeat Column Heading

The *Repeat column heading* field indicates whether you want the column headings to repeat above the member lines for a particular break level. No is the default value.

When paging through or printing a report, the column headings always appear at the top of the display or page. Another set of headings appears at the start of the break if YES is specified for *Repeat column headings*. This happens regardless of whether

there is any break heading text. However, if the break starts at the top of a printed page, only the set of column headings preceding the break member line format.

Blank Lines before Heading or Footing

The *Blank lines before heading* and *footing* fields indicate the number of blank lines that appear after any separator lines and before the break heading or break footing. If no break footing is specified, the value for this field is the number of blank lines before the break member lines. Query management allows any numeric value from 0 to 999. The default is 0 for both the heading and the footing.

The *Blank lines before heading* field may contain a number only.

For a break footing, you can also specify BOTTOM. Applicable only to a printed report, BOTTOM causes the break footing to position at the bottom of the current page on a printed report. BOTTOM causes insertion of blank lines to position the text immediately before the page footing text specification on the page. This also implies that a page eject occurs, since the next line must print on the next page.

Blank Lines after Heading or Footing

The *Blank lines after heading* and *footing* fields indicate the number of blank lines after the break heading or break footing. If no break heading is specified, the *Blank lines after heading* value is added to the *Blank lines before heading* value to determine the number of blank lines to use before the first or next set of detail lines. If no break footing is specified (there is no text and *Put break summary at line* is NONE), the value of this field is added to the *Blank lines before footing* value to determine the number of blank lines after the break member lines. Query management allows any numeric value ranging from 0 to 999. The default is 0 for the heading and 1 for the footing.

Put Break Summary at Line

The *Put break summary at line* field indicates whether the break summary is to format and, if it does, where to place it in relationship to the lines of break footing text. Query management allows a numeric value from 1 to 999, or NONE. The NONE value indicates that no break summary information is to display for the break. The default is 1.

If the *Put break summary at line* value is *m*, there are at least *m* break footing lines formatted in the report.

This placement is strictly vertical. For horizontal placement in the line, the break summary always formats under the columns being summarized.

Break Heading Text Line Fields

There are 999 lines available for the break heading text.

Line

The *Line* field indicates the line positioning in the heading text lines.

Align

The *Align* field controls the positioning of the break heading text within the report line. Acceptable values are:

RIGHT Right-justify the text.

LEFT Left-justify the text.

CENTER Center the text.

The default value is Left. The alignment is based on the entire width of the displayed or printed report through the end of the last column.

Break Heading Text

Query management allows 55 characters per line of break heading text. Only &*n* (where *n* is a column number) is allowed to be used as a variable.

If the text line *n* is the highest numbered line with nonblank text, then *n* indicates how many lines are to be formatted. This is true even though using *n* could result in some of the formatted lines being completely blank.

Break Footing Text Line Fields

There are 999 lines available for the break heading text.

Line

The *Line* field indicates the line positioning of the footing text lines.

Align

The *Align* field controls the positioning of the page footing text within the report line. Acceptable values are:

RIGHT Right-justify the text.

LEFT Left-justify the text.

CENTER Center the text.

The default value is Right. The alignment refers to the space between the first character position on the left and the end of the column before the first summary column (or the end of the last column in the report if there is no presentation of summary data for this break).

Break Footing Text

Only &*n* (where *n* is a column number) is allowed as a variable. Query management allows you to use up to 55 characters per line.

If the text line *n* is the highest numbered line with nonblank text, then at least *n* break footing lines are formatted. This is true even though using *n* could result in some of the formatted lines being completely blank.

Column Fields

Query management supports a maximum of 255 columns of information. The data retrieved by these columns has a limit of 32KB in the records retrieved. The following sections describe the fields available for use in defining the Column fields.

Summary of Values for Column Attributes

Figure 3-4 on page 3-14 shows the defaults and possible values for the attributes in the Column fields.

Figure 3-4. Default Values for Column Fields

Attribute	Default	Possible Values
Column heading	Run time	1 - 62 characters with 8 underscores
Usage	—	AVG, MIN, MAX, SUM, COUNT, BREAK1 - BREAK6, AVERAGE, MINIMUM, MAXIMUM, FIRST, LAST, OMIT
Indent	2	0 to 999
Width	Run time	1 to 32,767 SBCS
Datatype	Run time	CHAR, NUMERIC
Edit code — numeric	Run time	E, D, I, J, K, L, P
Edit code — character	Run time	C, CW, CT
Seq	Column number	1 to 999

Column Heading

The *Column heading* field represents the heading for a column in the report. A heading can be up to 62 characters long.

You can embed underscore characters in the heading and use them to indicate the break point for multiple line headings. Query management processes a maximum of eight underscores in a heading. Leading and trailing underscores produce blank segments before and after the column headings.

For example, a column heading of AMOUNT_LAST_INCREASE results in the following 3-line column heading:

```
AMOUNT
 LAST
INCREASE
```

Consecutive underscores (in any position) introduce blank lines.

The underscore rule prevents you from seeing an underscore character in a column heading. The only exception to this rule is when more than eight underscores appear in a column heading, in which case the extra underscores print as part of the last line of the heading.

Whenever you specify multiple-line headings, query management automatically centers the smaller lines within the space of the longest line. Headings for character data are automatically left-justified, and headings for numeric data are automatically right-justified. Data justification takes place within the width of the column. The width specification must reflect the length of the longest segment of this field.

If the number of characters in a heading line segment is greater than the number of characters specified in the *Width* field, then query management truncates the segment to the width specified for the column.

If you do not specify a column heading, query management provides a run-time default. You cannot cause a column to be shown without a heading by importing a form with a blank heading unless the database definition for the column indicates it should not have a column heading.

Usage

The value in the *Usage* field determines use of the column in the detail line of the formatted report result.

You can specify only one *Usage* value for each column in a form. If you want a column to have more than one usage, you must select the column multiple times in the query and define a usage code for each corresponding column entry in the form.

Figure 3-5 defines what each keyword describes.

Figure 3-5. Usage Code Definitions

Usage Code	Definition
AVERAGE (or AVG)	The average of the values in the column
COUNT	The count of the values in the column
FIRST	The first value in the column
LAST	The last value in the column
MAXIMUM (or MAX)	The maximum value in the column
MINIMUM (or MIN)	The minimum value in the column
SUM	The sum of the values in the column
OMIT	The column to be excluded from the report
BREAK1 – BREAK6	The value used to specify break levels

The *Usage* options include the following:

[blank]
Column to be included in the report.

OMIT
Column to be excluded from the report.

AVERAGE, COUNT, FIRST, LAST, MAXIMUM, MINIMUM, SUM
These keywords name aggregating uses that summarize the data in a column. The result of the usage is given at a break or final summary. AVERAGE and SUM work only on numeric data; COUNT, FIRST, LAST, MAXIMUM, and MINIMUM work with both numeric and character data.

When you compare characters using MAXIMUM and MINIMUM, the shorter string is padded with blanks and the strings are compared based on the internal binary codes. For example, the character string ab is greater than the character string aaa. There is no special processing on a character-by-character basis.

The following rules apply to the aggregating usages AVERAGE, COUNT, FIRST, LAST, MAXIMUM, MINIMUM, and SUM:

1. If aggregation overflow occurs, the value in the field is represented by greater-than symbols (>>>>) for the width of the column.

2. If the aggregation cannot be displayed due to the column width being too small, the value in the field is represented by asterisks (*****) for the width of the column.
3. If the data retrieved from the database for a particular column is bad, the value in the field is represented by question marks (?????) for the width of the column.

BREAK1

BREAK1 is the value used to specify a column as the first level, or highest, control break. A **control break** is the break point where a column value changes.

For example, if a set of rows of employees is ordered by department number and job title, you can use a BREAK1 to total the salaries of all the employees in the department, and a BREAK2 to total the salaries by job title within department. Each time a row with a different job title is read, a BREAK2 creates and displays the appropriate data and totals. Each time a row with a different department number is read, a BREAK2 and BREAK1 are created, and both sets of appropriate data display.

Before each break summary displays, a line is placed in the report consisting of a row of hyphens (--) under any displayed column with an aggregation usages. You can suppress this line by using an option described with the Options fields. A blank line is normally placed after each set of break data.

Use the aggregation usages (AVERAGE, COUNT, FIRST, LAST, MAXIMUM, MINIMUM, SUM) at control breaks. For example, summary data appears as subtotals of all columns with a usage of SUM, or an average of the columns with a usage of AVERAGE.

The data printed as part of the break is determined by the break definition. For more information on break definitions, see "Break Fields" on page 3-11.

The break level numbers are not required to be consecutive. In this respect, control break numbers are not absolute; you could specify control breaks 2, 4, and 6, without specifying 1, 3, and 5. However, control break text assignments continue to be absolute. Text for control break 2 is always associated with control break number 2, not with the second control break.

You can assign multiple columns to the same BREAK*n* value. When this occurs and control breaks need to be determined, query management considers all columns having the same control break level as a single concatenated column. This is particularly useful when LAST_NAME and FIRST_NAME are stored as separate columns. Likewise, it is needed when MONTH, DAY, and YEAR are each specified as separate columns.

When using a control break, the data in the column should be ordered. For the data to be in order, the SELECT that produces the report must use ORDER BY.

There is no automatic reordering of columns due to break specifications, but break text is usually displayed to the left of any summary columns. Therefore, IBM* recommends using the *Seq* value to display the break columns to the left, and the aggregated columns to the right, on the report.

BREAK2

BREAK2 is the use value that specifies a column as the second level control break. A BREAK2 automatically generates whenever the columns on which it is defined changes, or when there is a BREAK1 generated.

Note: A BREAK1 causes a BREAK2, but a BREAK2 does not cause a BREAK1.

BREAK3 through BREAK6
BREAK3 through BREAK6 are values that name control columns for breaks at levels 3 through 6.

Indent

The value in the *Indent* field represents the relative location of the column within a row. Its units are the number of blank characters between the column and:

- The right edge of the previous column
- The left edge of the display or paper

The *Indent* can be *n*, where 0 < = *n* < = 999. Query management defaults the *Indent* field to 2. Query management defines this field as numerics only.

Width

The value in the *Width* field specifies the number of spaces to reserve for displaying the column heading and data. Column heading lines that are wider than specified in the *Width* field are truncated. Query management allows only a numeric *Width* field value. The maximum width is 32,767 spaces. If the length of the value to display exceeds the width of the column, the value is replaced with a row of asterisks (****) if it is numeric data, or truncated at the right if it is character data. If your report is not displayed or printed as you specified, change the *Width* field value and display the report again. If no Width value is specified, query management provides a run time default.

Datatype

The *Datatype* field specifies the kind of data that is represented by the column.

Possible values are:

- CHAR
- NUMERIC

If no Datatype value is specified, query management provides a run-time default. If the data type is specified in the form, it must match the corresponding data type of the field in the database for a report to be generated.

Edit

Use edit codes to format character and numeric data for display.

The following edit codes are valid for *character* data:

C The C edit code makes no change in the display of a value. If the value cannot fit onto one line in the column, query management truncates the text according to the width of the column.

CW The CW edit code makes no change in the display of a value, but if the value cannot fit on one line in the column, query management wraps the text according to the width of the column. Instead of truncating the data at the end of the column, query management puts as much data as possible on one line in the column and then continues the data on the next line in the column.

Use the CW edit code on columns of mixed DBCS and single-byte character data.

CT The CT edit code makes no change in the display of a value. If the value cannot fit onto one line in the column, query management wraps the column according to the text in the column. Instead of cutting off the data at the end of the column, query management fits as much data as possible on a line, interrupts the line when it finds a blank, and continues the data on the next line. If a string of data is too long to fit into the column and does not contain a blank, query management wraps the data by width until it finds a blank and can therefore continue wrapping by text.

Use the CT edit code on columns of mixed DBCS and single-byte characters. Query management interrupts the line when it finds a single-byte blank.

The following edit codes are valid for *numeric* data:

E The E edit code displays numbers in scientific notation. For example, the number -1234.56789 displays as -1.234E+03. As many characters as can display are placed in the report, up to a maximum of 23. One space is always reserved for a leading sign, although it does not display for positive numbers. There is always a sign and three digits after the E. The system can display up to three digits after the E.

D, I, J, K, L, and P

The D, I, J, K, L, and P edit codes display numbers in decimal notation with different combinations of leading zeros, negative symbols, thousands separators, currency symbols, and percent signs. Examples of these edit codes are shown in Figure 3-6.

Each code can be followed by a number from 0 to 31 that tells how many places to allow after the decimal point. If you do not specify a number, query management assumes no places after the decimal point. Numbers that have more decimal places than fit into the allowed space are rounded; numbers with fewer decimal places are padded with zeros.

Figure 3-6 shows how the numeric edit codes format the number -1234567.885.

Edit Code	Lead Zeros	Negative Sign	Thousands Separators	Currency Symbol	Percent Sign	Display of -1234567.885
E	No	Yes	No	No	No	-1.23456789E+06
D2	No	Yes	Yes	Yes	No	-$1,234,567.89
I3	Yes	Yes	No	No	No	-0001234567.885
J2	Yes	No	No	No	No	000001234567.89
K3	No	Yes	Yes	No	No	-1,234,567.885
L2	No	Yes	No	No	No	-1234567.89
P2	No	Yes	Yes	No	Yes	-1,234,567.89%

Figure 3-6. Use of Edit Codes

If you do not specify an *Edit* value, query management provides a run-time default.

Seq

Specify Seq to order the columns in the generated report. The following rules apply to evaluating *Seq* values:

- Values default to *n* for the *n*th column in the form
- Values can be any number from 1 to 999.
- Numbers need not be consecutive.
- Columns with the same *Seq* value appear in the report in the same order that they appear in the form.

Run-Time Defaults

Query management uses system-provided default values for the *Datatype, Column heading, Edit,* and *Width* fields when columns of data extracted by running a query need to be formatted for a report, and one of the following is true:

- No form is specified or *SYSDFT is specified to refer to the default form for the extracted data.
- The specified form does not contain the information needed to format the report (the form was imported with warnings about blank values or missing column table fields).

Datatype

Figure 3-7 shows datatype defaults that are assumed from internal numeric identifiers returned with the data. For an explanation of the datatype numbers and meanings, see the *SQL/400* Reference.*

Figure 3-7. Run-Time Defaults for Columns Datatype Field

Datatype Value	Datatype Number	Meaning (SQL)
NUMERIC	496	Integer (binary)
	500	Small integer (binary)
	484	Decimal
	480	Floating point
	488	Numeric (zoned)
CHAR	452	Fixed character
	448	Varying-length character
	456	Long varying-length character

Column Heading

Establish Column heading default values for file data when defining a field to the system. Field names are used as column headings unless other text is specified when you use interactive data definition utility (IDDU).

Define files that do not cause column headings to be defaulted. Column heading default values for calculated data and for file fields without established defaults are taken from the set of unique column names manufactured at run time for the selected columns. These are the column names that are used to create a new file for a SAVE DATA request.

To manufacture these names, query management processes the selected columns in order, from first to last. The unique name for the *n*th selected column is created in the following way:

1. If it does not match any previously created name, the column name from the file (table) definition (or SEL for a calculated field) is used as the created name.
2. If the matched name is too long to add the next available number, that number is added to COL to create the unique name.
3. If the matched name is not too long to add the next available number, that number is added to the matched name (or SEL) to create the unique name.

Note: The first number added to a column name or COL to make the name unique is 1, the next number added is 2, and so on.

The following example shows the unique names created for a particular SELECT list.

```
SELECT 7*WEEKS, SALARY, SALARY, SALARY/7*WEEKS, MAXBENEFIT, MAXBENEFIT
         |       |       |          |              |          |
         |       |       |          |              |          |
        SEL    SALARY  SALARY1     SEL2        MAXBENEFIT     COL3
```

If some of the columns have previously defined column heading defaults, the column headings in a formatted report are not necessarily unique. This can be true for the *SYSDFT form as well as for a form specified by name.

Edit

Editing default values are intended to be the same as those used by other data-displaying products on the system, such as Query/400. Character data is unchanged or truncated (this is what happens for SAA edit code C). Scientific notation is used for floating data (this is what happens for SAA edit code E). Editing defaults usually have no corresponding SAA representation for other types of numeric fields, and can include edit words, edit descriptions, and RPG edit codes.

File (table) data default values are established when a field (column) is defined to the system. System-level values determine the editing for calculated data. These values come from a translatable message and are used for building a default edit description whenever one is needed.

Changeable system-level values can also affect the editing applied to numeric data. For example, the QDECFMT system value determines the decimal point used in the editing applied for RPG edit code J.

Width

Width default values are intended to provide room for everything that has to be shown in the column. For a particular column, the default is the maximum of the following:

- The length of the longest segment of the column heading
- The edited data width (after adjustment of the number of digits by 3 if SUM Usage aggregation values have to be shown in the column)
- A length of 9 if COUNT Usage aggregation values have to be shown in the column

Final Text Fields

The Final Text fields specify the characteristics of the final text that appear on a report.

Summary of Values for Final Text Information

Figure 3-8 shows the defaults and possible values for the attributes on the Final Text fields.

Figure 3-8. Default Values for Final Text Fields

Attribute	Default	Possible Values
New page	NO	YES, NO
Put final summary at line	1	1 to 999 or NONE
Blank lines before text	0	0 to 999 or BOTTOM
Alignment for final text	RIGHT	LEFT, CENTER, RIGHT

New Page for Final Text

The *New page* field indicates whether the subsequent part of the report (final text) must begin on a separate page when printed. The default is No. When you specify Yes for *New page*, the final text formats on a new page.

Put Final Summary at Line

The *Put final summary at line* field indicates whether the final summary should format, and if so, where to vertically position it in the report final text. Query management allows numeric values from 1 to 999 or the word NONE, where NONE indicates there is no presentation of final summary data. The default value is 1.

A value of 1 to 999 indicates the relative line number within the final text at which the summary data must format. This is strictly vertical placement. For horizontal placement in the line, the final summary is always formatted under the columns being summarized. If the *Put final summary at line* value is *m*, then at least *m* final lines are formatted in the report.

Blank Lines before Text

The *Blank lines before text* field indicates the number of blank lines between the body of the report and the first line of final text. Query management allows any numeric value from 0 to 999. The default value is 0. You can also specify BOTTOM, which positions the final text at the bottom of the printed page. BOTTOM causes a number of blank lines to be inserted in order to position the final text immediately before the page footing text specification on the page. If there is not enough space for the final text on the current page, it is placed at the bottom of the next page.

Final Text Line Fields

There are 999 lines available for the final text.

Line

The *Line* field indicates the line positioning in the final text lines.

Align

The *Align* field controls the position of the final text within the report line; it refers to alignment between the first character position on the left and the end of the column before the first summary column (or the end of the last column in the report if no final summary data is presented). The following shows the acceptable values:

RIGHT Right-justify the text.

LEFT Left-justify the text.

CENTER Center the text.

The default value is Right for the final text. If there is no associated final text, the *Align* value is ignored.

Final Text

Specify a maximum of 55 characters of text for each text line. Only &*n* (where *n* is a column number) is allowed as a variable.

If the text line *n* is the highest numbered line with nonblank text, then at least *n* lines are formatted. This is true even though it could result in some of the formatted lines being completely blank.

---

options fields

The Options fields allow you to specify various report formatting options.

Summary of Values for Options Attributes

Figure 3-9 shows the defaults and possible values for the attributes on the Options fields.

Figure 3-9. Default Values for Options Fields

Attribute	Default	Possible Values
Detail line spacing	1	1 to 4
Outlining for break columns	YES	YES, NO
Default break text	YES	YES, NO
Column wrapped lines kept on page	YES	YES, NO
Column heading separators	YES	YES, NO
Break summary separators	YES	YES, NO
Final summary separators	YES	YES, NO

Detail Line Spacing

The *Detail line spacing* field indicates the spacing you request between detail lines in the report. Query management only allows numeric values for this field. Acceptable values are 1, 2, 3, or 4, where 1 is single spacing, 2 is double spacing, and so on. The default value is 1.

Outlining for Break Columns

If you assign a usage code for a break to one of the columns, use the *Outlining for break columns* field to determine when the value in the break column displays in the report. YES is the default and displays the value in the break column only when the value changes. A NO in this field displays the value in the break column on every detail line in the report.

Default Break Text

The *Default break text* field indicates whether you are requesting inclusion of the default break text in the report. The default value is YES. The default break text is one or more asterisks for each break level: one asterisk for the highest numbered break level text, two asterisks for the next highest numbered break level text, and so on. For example, if the report has two control breaks, Break2 and Break4, query management generates one asterisk to mark the level-4 break and two asterisks to mark the level-2 break. Default break text is used only for levels with no break text and numeric values for the *Put summary at line* field. Default break text is right-justified on break footing line 1.

Column Wrapped Lines Kept on a Page

If you specify column wrapping for one or more columns in the report, use the *Column wrapped lines kept on page* field to determine whether the wrapped columns can be split between two pages. The default for this field is YES. It prevents splitting wrapped columns between two pages (unless the wrapped column is longer than the page depth). A NO in this field allows splitting of wrapped columns between pages.

Column Heading Separators

The *Column heading separators* field indicates whether the column heading separators (dashed lines) appear in the report. The default value is YES.

Break Summary Separators

The *Break summary separators* field indicates whether the break summary separators (dashed lines) appear in the report. The default value is YES.

Final Summary Separators

The *Final summary separators* field indicates whether the final summary separators (equal signs) appear in the report. The default value is YES.

Page Fields

The Page fields allow you to specify headings and footings on a report. Figure 3-10 shows the defaults and possible values for the attributes on the Page fields.

Figure 3-10. Default Values for Page Fields

Attribute	Default	Possible Values
Blank lines before heading	0	0 to 999
Blank lines before footing	2	0 to 999
Blank lines after heading	2	0 to 999
Blank lines after footing	0	0 to 999
Alignment on heading text	CENTER	LEFT, CENTER, RIGHT
Alignment on footing text	CENTER	LEFT, CENTER, RIGHT

Blank Lines before Heading or Footing

The *Blank lines before heading* or *footing* fields indicate the number of blank lines before the page heading or page footing and must be specified as numbers. Query management allows any numeric value from 0 to 999. The default value is 0 for the page heading and 2 for the page footing. A page eject always precedes the heading on each page. The *Blank lines before heading* field controls the number of blank lines between the heading and the top of the page. The *Blank lines before footing* field controls the number of blank lines between the report body and the first footing line.

Blank lines are included in the count of the number of lines printed on the page.

Blank Lines after Heading or Footing

The *Blank lines after heading* or *footing* fields indicate the number of blank lines after the page heading or page footing. The fields are defined in query management as numerics only. Query management allows any numeric value from 0 to 999. The default value for the page heading is 2 and for the page footing is 0. The *Blank lines after heading* field controls the number of blank lines between the last heading line and the report body. The *Blank lines after footing* controls:

- The number of blank lines between the last footing line and the end of page
- The last footing line and the line containing the date and time and page number

Blank lines are included in the count of the number of lines printed on the page.

Blank lines after footing takes precedence over *Blank lines before footing*. On a report page that has extra space left after the body of the report, extra blank lines are inserted so the *Blank lines after footing* value is the correct number of lines.

Page Heading Text Line Fields

There are 999 lines available for the page heading text.

Line

The *Line* field denotes the line positioning in the heading text lines.

Align

The *Align* field controls the position of the page heading text within the report line. Acceptable values are:

RIGHT Right-justify the text.

LEFT Left-justify the text.

CENTER Center the text.

The default value for headings is Center.

Page Heading Text

The *Page heading text* field allows you to specify text that appears as the page heading in the report. You can specify a maximum of 55 characters for each text line.

If the text line *n* is the highest numbered line with nonblank text, then *n* indicates how many page heading lines are to be formatted. This is true even though using *n* could result in some of the formatted lines being completely blank.

Page headings may contain four types of special variables. All variables must be coded with a leading ampersand (&) to identify them within the heading text.

&*n* &*n* is assigned the value from the *n*th column of the current record.

&DATE Query management replaces the value with the current date.

&TIME Query management replaces the value with the current time.

&PAGE Query management replaces the value with the current page number. The page number is a 4-digit number ranging from 1 through 9999, with leading zeros suppressed. After 9999, the counter wraps to 0 and continues to increase for subsequent pages *without* leading zero suppression.

When the report prints, the page heading appears at the top of each page, formatted according to the format specification. The variable &*n* formats according to the edit code specification, except when column wrapping is specified. If column wrapping is specified for the column, it is ignored when the data formats into the text.

All variables are resolved when the report is created.

Page Footing Text Line Fields

There are 999 lines available for page footing text.

Line

The *Line* field denotes the line positioning in the footing text lines.

Align

The *Align* field controls the positioning of the page footing text within the report line. Acceptable values are:

RIGHT Right-justify the text.

LEFT Left-justify the text.

CENTER Center the text.

The default value for footings is Center.

Page Footing Text

The *Page footing text* field allows you to specify text for the page footing that is to appear in the report. You can specify a maximum of 55 characters for each text line. You can use the following special variables:

- &*n*
- &DATE
- &TIME
- &PAGE

When the report formats, appropriate values are substituted for the variables. When the report prints, the page footing appears at the bottom of each page, formatted according to the format specification. The variable &*n* formats according to the edit code specification, except when column wrapping is specified. If you specify column wrapping, it is ignored when the data formats into the text. The formatted page footing appears once at the bottom of the displayed report.

If the text line *n* is the highest numbered line with nonblank text, then *n* indicates how many page footing lines are to be formatted. This is true even though using *n* could result in some of the formatted lines being completely blank.

Procedures

You may find yourself creating reports over and over again that use the same query management commands. If you do, consider running these steps together by creating a procedure. A **procedure** allows you to run a set of query management commands with a single RUN command.

The following example illustrates a procedure called PAYROLL:

```
'RUN QUERY PAYROLL'
'PRINT REPORT (FORM=PAYFORM'
'SAVE DATA AS MYLIB/PAYSUMM'
```

This procedure runs the PAYROLL query, prints a report, and saves the data to a file called PAYSUMM.

Procedures also allow flexibility in your application. Your application can be written to run a named procedure. At any time, the procedure can be updated or tailored to fit a new situation without requiring you to change your application program. The following sections describe how query management uses procedures to process commands.

Procedure Interaction and Rules

Procedures can make working with query management easier and faster than coding each program separately. Consider the following rules when creating procedures for use in query management:

- The width of a procedure line is limited to the source file record width.
- The width of a query management command on a procedure line after procedure parsing is done is limited to 256 characters. Procedure parsing involves stripping leading and trailing blanks and reducing the number of quotation marks.
- Each query management command must be in uppercase English letters.
- Procedures can contain only query management commands and blank lines. (Blank lines have no effect on command processing)
- Procedures can contain a RUN command that runs another procedure or query.
- Nested procedures are allowed, and each procedure uses the instance defaults and selected fields of the procedure that called the one currently running. Therefore, a PRINT command processed in a procedure that has just run a procedure with a RUN QUERY command prints the data from the RUN QUERY.
- The query command within a procedure must be delimited by quotation marks or apostrophes ("" or '').
- The GET command is not functional since query management procedures do not have high-level language constructs. A GET command within a procedure results in an informational message queued to the job queue. The message contains the variable name and the value.
- Query management treats all variable values on a SET command as character strings. Therefore, it is not possible to set an integer variable within a procedure.
- A procedure may optionally contain an H record and a comment V record. The comment record may be used for a procedure description when the procedure is imported.

Interaction with a query user depends on the interactive state of query management. This state is controlled by the start-up parameter DSQSMODE on the START command. If you allow the interactive state, there are further considerations when using a procedure.

You see a formatted report display as each query in a procedure processes. You can then page through the report. An exit from the report returns control to the procedure and causes the next statement to process. For more information on how each command is processed in a particular instance of the procedure, see "Query Management Commands" on page 2-1.

Procedure Objects as File Members

The query management procedure is a source physcial file. Query management allows a specific member to be identified on the RUN PROC, IMPORT PROC, EXPORT PROC, PRINT PROC, and ERASE PROC commands by allowing members to be given as part of the query object name. The member name must follow the query object name and be delimited by parentheses with no intervening blanks. The following example shows how each of the procedure commands can be changed to direct query management to a specific member:

```
RUN PROC MYLIB/MYPROCS(MYMEMBER)
PRINT PROC MYLIB/MYPROCS(MYMEMBER)
IMPORT PROC MYPROCS(MYMEMBER) FROM QQMPRCSRC
EXPORT PROC MYLIB.MYPROCS(MYMEMBER) TO QQMPRCSRC(MYMEMBER)
```

If a member is not specified as part of the object name, it is always assumed to be the first member of the file. If an ERASE PROC is issued and more than one member exists, only the first member is deleted. If a member is not specified and is to be created as part of the IMPORT PROC processing, it is created with the same name as the procedure file name.

Delimiters in the Procedure

All comments must be surrounded by apostrophes, for example, SAVE AS LASTWKDATA (COMMENT = 'Last weeks data'. If the comment contains an internal apostrophe, it is represented by two successive apostrophes ('Last week''s data').

Apostrophes ('') are supported as procedure string delimiters. When the procedure statement is surrounded by double quotation marks (""), internal apostrophes ('') need not be doubled, and vice versa.

Example

The following example illustrates how apostrophes are used in a procedure.

```
'SAVE DATA AS LASTWKDATA (COMMENT=''Last week''''s data'''
'IMPORT FORM REPT4 FROM MYLIB/FORMS(REPT4) (CONFIRM=YES'
'SET GLOBAL (TBLENAME=MYFILE'
'SET GLOBAL (CMPVAL2=''Joe A. Customer'''
'RUN QUERY REPT4QRY (FORM=REPT4'
'SAVE DATA AS LASTWKDATA'
'PRINT REPORT'
```

Error Handling

Whenever an error with the severity of FAILURE occurs, procedure processing stops, and the completion code of the procedure reflects the error. All messages encountered during procedure processing are queued to the job log, and a summary message is returned in the communications area.

Error Categories

The following error categories are possible in query management procedures:

- Command not allowed in query management procedure.
- Command in query management procedure not valid.
- String in query management procedure not valid.
- Recursion not allowed in query management procedure.
- Maximum procedure nesting level exceeded. A nesting level of 15 is allowed.

Exported Objects

This section covers the IMPORT requirements and the AS/400 EXPORT specifications for the source physical files used and generated during import and export of query management objects.

IMPORT and EXPORT File Considerations

Query management imports externalized query management objects from and exports them to source physical files. The following set of general rules applies when using the source physical file that contains an externalized query management object:

- An object comment is allowed in an externalized query management form, query, or procedure. The comment is specified as a V record with the field number 1001. Maximum comment length is 50 characters and a comment greater than 50 will be truncated. The comment is generated in the externalized object on export. The comment record, if present, must immediately follow the H record. The H record object type field must be either an F for form, Q for query, or P for procedure.
- On import, the following sequence is followed to determine which text description is used on the query management object.
  - If the COMMENT= option is specified, the value on this option is used.
  - If there is member text on the member, the member text is used.
  - If there is a comment in the object, this comment is used.
  - If there is no comment to use, a blank text description results.
- On export, the following sequence is followed to determine which text is written to the externalized query management object.
  - If the COMMENT= option is specified, the value on this option is used.
  - In the absence of the COMMENT= option, the text description on the Query Management object is used.
- The H and comment V records are the only parts of the encoded format that are allowed for query or procedure objects. Attempting to use other record types like T, R, E, and * will generate unpredictable results.
- An object comment in a procedure must be inside comment delimiters with no intervening blanks between the start comment and the record identifier. A comment begins with '/*' and ends with '*/'. In addition to an object comment, procedures may have user comments, which are ignored at run time. Comments may not span multiple lines and may not be used inside other comments.

The following is an example of a procedure with an object and user comments:

```
/*H QM4 01 P 01 E V W E R 01 01 90/07/24 13:30*/
/*V 1001 050 SALES PROCEDURE*/
'IMPORT QUERY SALES FROM QRYSRC'
'RUN QUERY SALES' /* Total sales query */
'PRINT REPORT' /* Management report */
```

The following is an example of an object comment in a query:

```
H QM4 01 Q 01 E V W E R 01 03 90/06/28 01:25
```

```
V 1001 050 SALES QUERY
SELECT SALARY FROM SALESFILE WHERE SALARY > 50000
```

- Query management allows import from and export to multiple member source files.
- You must create a source file with a record length that allows for 12 positions for the *Source sequence number* and *Date* fields required in each record.
- Query management truncates data and generates a warning message when exporting to an existing source file if the file does not have a sufficient data length. The following situations can result in truncation:
  - Exporting an SQL query or procedure object to a file that has a data length less than 79 bytes. Query management creates the truncation warning message whenever any part of the SQL statement or query command has been truncated.
  - Exporting a query management form object to a file that has an insufficient data length. The minimum allowed data length is 150 bytes (based on the maximum length of an exported encoded-format form record, a Columns table R record). When query management creates a file as a result of the export, it is created with a data length of 150 bytes. Therefore, a data length of 150 bytes is recommended. Query management issues the truncation warning message when any truncation of data occurs. Since parts of the form are optional, the actual data length required may be less than 150 bytes.
- The confirmation message (when CONFIRM=YES is specified on the EXPORT command) is sent when a member of a source physical file is being replaced. It is not sent if a new member of an already existing source physical file is being created.
- Query management ignores all columns in the file past column 79 during import of an SQL query or procedure object. A message is generated if columns are ignored.
- Query management ignores all columns in the file past column 150 during import of a form object. A message is generated if columns are ignored.
- The AS/400 system does not support files with varying record lengths. Therefore, prior to or during the transfer to the AS/400 system for import, the file containing the externalized form object in encoded format must be converted to fixed record format. On export of a form object, query management creates a file with fixed-length record format. The record is padded with blanks from the end of meaningful data to the end of the record. Therefore, before importing to a product that does not support fixed format externalized forms, the file must be converted to varying-length record format.
- Query management pads the internal representation of each record with blanks up to and including position 79 when importing an SQL query or procedure object, if the input file has a data length less than 79 bytes. If the line contains an open string enclosed in quotation marks, this padding is then included within this string and can cause unexpected results.
- The source physical file that contains the externalized SQL query or procedure can be any size allowed by the AS/400 system for a source physical file.

  Note: Although the source physical file can be any size allowed by the system, the results may be truncated when imported if they exceed the maximum allowed limits. For more information on import limits, see the *SQL/400* Reference* manual.

Display Format

For more information on the externalized procedure and SQL query objects that use the display format, see "Procedures" on page 3-26 and "Query Capability" on page 3-6.

Encoded Format

Query management uses the encoded format only for the externalized form object. This section covers each record type in the encoded format as used by query management. Query management tolerates fields that are not defined in this manual.

Note: Unrecognized fields are lost during import and are not displayed on subsequent exports.

Importing a Form Object

The following rules apply when importing a form object to query management in encoded format:

- The H record must be the first record in the file.
- Record types other than H, V, T, R, and * encountered within the file before the E record are ignored
- A warning message is issued if unknown record types are encountered.
- Records after the E record are ignored
- The T record of the Columns table must immediately follow the header or comment V record, and must include a numeric count of the number of rows in the table (an * row count is not allowed).
- Query management ignores the control area length (cc) field in the H record.
- Query management assumes that the control area length value on all form object records is 01.
- Query management uses the delimiter values specified on each of the form object records. Therefore, a nonblank delimiter is allowed.
- The delimiter value on the H record is ignored, since the H record is column-specific.
- The following fields must be in uppercase when a form object is imported:
  - Record identifiers for all records
  - The following in the header record:
    - Product identifier (QRW, QMF*, QM4, and so on)
    - Type of object (F)
    - Format of object (E)
    - Action (R)
  - Data type values (NUMERIC, CHAR) in the R records for the Columns table
  - All the form object keywords and substitution variables
- Duplicate occurrences of data values or tables override previous settings, with one exception. Query management does not override previous settings if the new object violates the rules established for an object. For example, the number of columns provided for a form cannot be varied after the first Columns table has been processed.
- Combining the original format and the new format for representing the break information is not allowed.

- Object values not included in the input file are set to their default values.

Columns Table Details

The form must contain all of the columns for the underlying data. Form and data mismatch are not detected until the form is applied at RUN or PRINT time.

When the entire Columns table is processed, unspecified fields result in the default values at run time. The default values are applied at run time and are those that were defined at file definition from the table you are querying. Export of a form that was imported with missing Columns table fields results in an externalized form that has the same Columns table fields missing. Query management allows and uses AS/400 edit codes when the defaults are applied at run time. A warning message is generated at import and export when these fields are not specified.

Query management allows multiple occurrences of the Columns table but does not allow subsequent occurrences of the table to alter the number of columns.

Query management supports the import of forms that specify the data types of DATE, TIME, and GRAPHIC, although SQL/400 conventions and the AS/400 database do not support these data types. Query management allows importing a form object containing these data types, but an attempt to apply the form results in data and form mismatch errors at run time. A warning message is generated if a form object is imported specifying an unsupported data type. Values in the Columns table that are not recognized or not valid are ignored, and default values are assumed.

Exporting a Form Object

Query management uses blank delimiters, regardless of the delimiter used on the imported object. The information that was defaulted at import time is exported for all report sections other than Columns. Query management exports form objects using the new format to describe the break information.

Records that Make Up the Base Encoded Format

The formats of the records that make up the base encoded format are described in the following sections.

For each record type there is a description of its purpose, contents, format, and a set of notes on its use. There are also record descriptions that provide a list of the possible values for each field in the record. Some of these fields (particularly in the header record) may contain only a single value. This is intentional and often signifies that other values for the field will be allowed in the future.

The base set of record types in the encoded format includes:

Record Type	Descriptive Name
H	Header (H) record
V	Value (V) records
T	Table description (T) records
R	Table row (R) records
E	End-of-object (E) records
*	Application data (*) records

Header (H) Record

The header (H) record identifies the contents of an externalized object (an object is an externalized object when it is being displayed either online or as a printed copy). It contains information describing the characteristics of the object and the file format.

Figure 3-11 summarizes the contents of the H record.

```
                ....+....1....+....2....+....3....+....4....+

FORMAT:         Hdppp rr t oo f u s n a cc ii yy/mm/dd hh:mm
                     |  | |  | | | | | | |  |  |        |     |
                     d  d d  d d d d d d  d  d        d     d

                    |_____________________________________|
                                       |

                                  fixed format


WHERE:          H          indicates this is an object Header record
                d          is the data field delimiter for this record
                           only -- a blank
                ppp        is the product identifier:
                             QM4
                rr         is the QM4 level in which the object
                           was produced:
                             03, 04, 05, and so on.
                t          is the type of object in this file:
                             F for form
                             Q for query
                             P for procedure
```

Figure 3-11 (Part 1 of 2). Header Record Description

```
              oo        is the QM4 object level at the time when the
                        given type of object was produced:
                          01, 02, 03, and so on.
              f         is the format of the object in this file:
                          E for encoded format
              u         indicates the status of the object:
                          E for contains Errors
                          W for contains Warnings
                          V for Valid
              s         indicates the subset of the object included:
                          W for Whole object
              n         indicates the language of the exported
                        object.
                             E          English
              a         is the action against the item:
                          R for Replace object
              cc        is the length of the control area in the
                        beginning of each following record (including
                        the 1-byte record type):
                          01 for Forms
              ii        is the length of the integer length fields
                        specified in "V" and "T" type records:
                          03 for all objects
              yy/mm/dd  date stamp
              hh:mm     time stamp

EXAMPLE:      H QM4 03 F 03 E V W E R 01 03 89/09/23 15:21

              (QM4 form file at "object level" 3, written in the
               encoded format, with no errors or warnings, in entirety
               in English, usable for complete replacement, with 1
               byte of control area, and 3 bytes for integer lengths)
```

Figure 3-11 (Part 2 of 2). Header Record Description

Figure 3-12 summarizes the location and contents of each of the fields in the H record. The field names used in Figure 3-12 are defined in Figure 3-11, which describes the entire H record. Object-specific values are noted as appropriate.

Field	Columns	Possible Values	Required on Input	Default if Blank on Input
H	01	H	yes	
d	02	(blank)	no	(not used on input)
ppp	03–05	QM4	yes	
rr	07–08	03	no	(not used on input)
t	10	F Q P	yes	
oo	12–13	03	no	current object level
f	15	E	yes	
u	17	E W V	no	(not used on input)
s	19	W	no	(not used on input)
n	21	E (English)	no	(not used on input)
a	23	R	yes	
cc	25–26	01 (Form)	no	(not used on input)
ii	28–29	03	no	(not used on input)
yy/mm/dd	31–38	(dates)	no	(not used on input)
hh:mm	40–44	(times)	no	(not used on input)

Figure 3-12. Header Record Fields

Notes to Figure 3-12:

- The H record must be the first record in the external file.
- Those fields (or portions of fields) left unspecified are assumed to be blank if the record is shorter than its fixed format length.
- The object level (oo) is used to denote a change in the externalized format of an object. When a particular level of query management changes the external format of an object, the object's level number is increased by 1. An application can use this number to determine the format of the object's records.
- The control area (with length cc) is a fixed area in the beginning of each of the encoded format records (except the H record) that contains control information pertaining to the given record; the control area contains information such as the record type and a record continuation indicator.
- The subset, format, action, control area length, and integer length fields are included in the H record for future extensions to the encoded format.
- Additional fields will be added to the end of the H record in the future.

Value (V) Records

The value (V) record provides a value for a single nontabular field in an object (such as a Form Options field). It includes the unique field number, the field's value, and its length.

Figure 3-13 summarizes the contents of the V record.

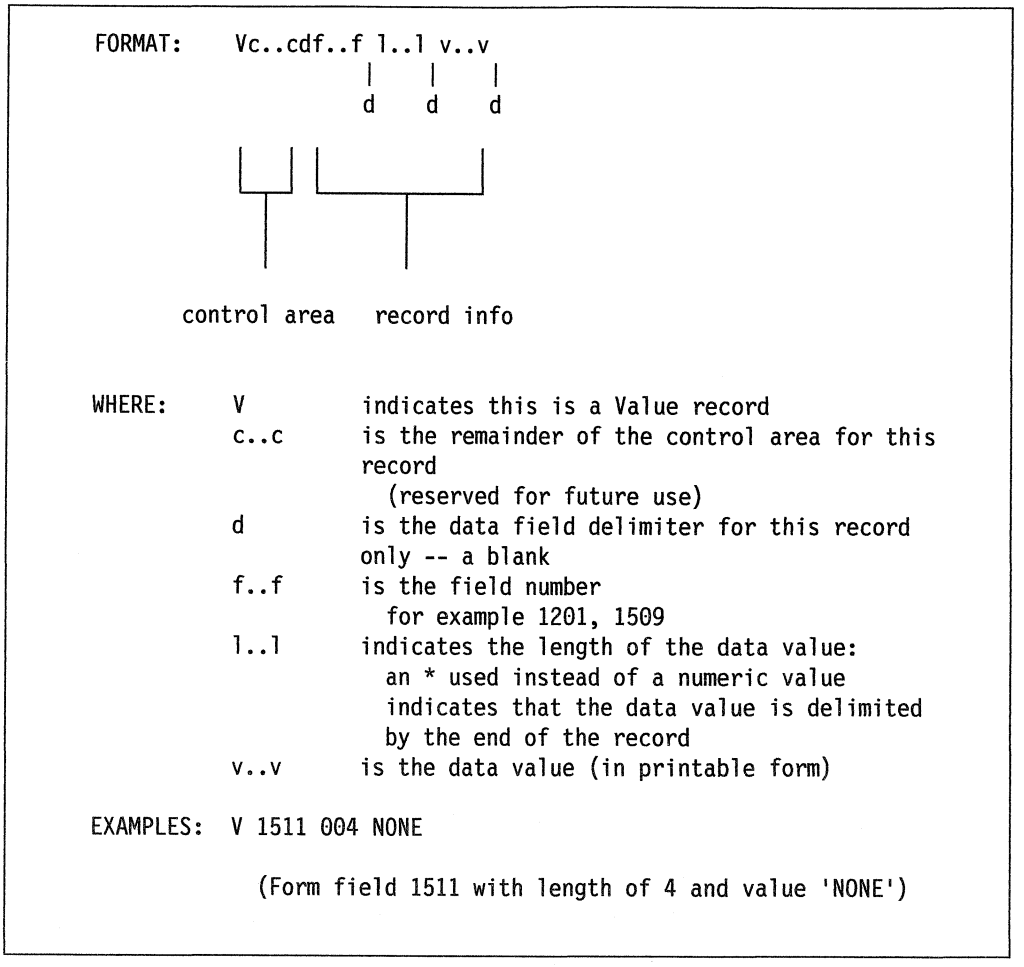


Figure 3-13. Value Record Description

Figure 3-14 on page 3-37 summarizes the location and contents of each of the fields in the V record. The field names used in Figure 3-14 on page 3-37 are defined in Figure 3-13 which describes the entire V record. Object-specific values are noted as appropriate.

Control Area				
Field	Columns	Possible Values	Required on Input	Default if Blank on Input
V	01	V	yes	
c..c	02	(x)	n/a	
x	02	(blank)	n/a	

Remainder of Record				
Field	Offsets past Control Area	Possible Values	Required on Input	Default if Blank on Input
d	+01	(blank)	no	
f..f	+02–05	1001–9999	yes	
l..l	+07–09	* 000–999	yes	
v..v	+11–end	(data)	no	(blank)

Figure 3-14. Value Record Fields

Notes to Figure 3-14:

- An omitted data value (such as end-of-record), or blanks only following the asterisk (*), indicates that a blank value is to be applied to this field.
- To set a field to blank, the field must have a specified positive length and a blank data value.
- Fields are set to their default values when the object is updated if:
  - The specified length is zero
  - No length is specified
- Query management issues a warning when it finds a field length of zero to indicate that the default value is set for this field.
- Query management uses the specified length and issues a warning message if the specified length is shorter than the supplied data value.
- Query management sets the data value without extending beyond the end of the record and issues a warning message if the specified length is longer than the supplied data value.
- IBM is keeping the length field open for future V record uses in which an explicit length may be specified (for example, to indicate significant blanks), and for the possibility of V record expansion.

Table Description (T) Records

The table (T) record describes the content and format of the table of values that follows. The contents of a T record determine the contents of all row (R) records for this table. A T record indicates which table is being described (by its unique table number), which columns are included (by their unique field numbers), in what order they appear, and the lengths of the values in these columns.

Figure 3-15 summarizes the contents of the T record.

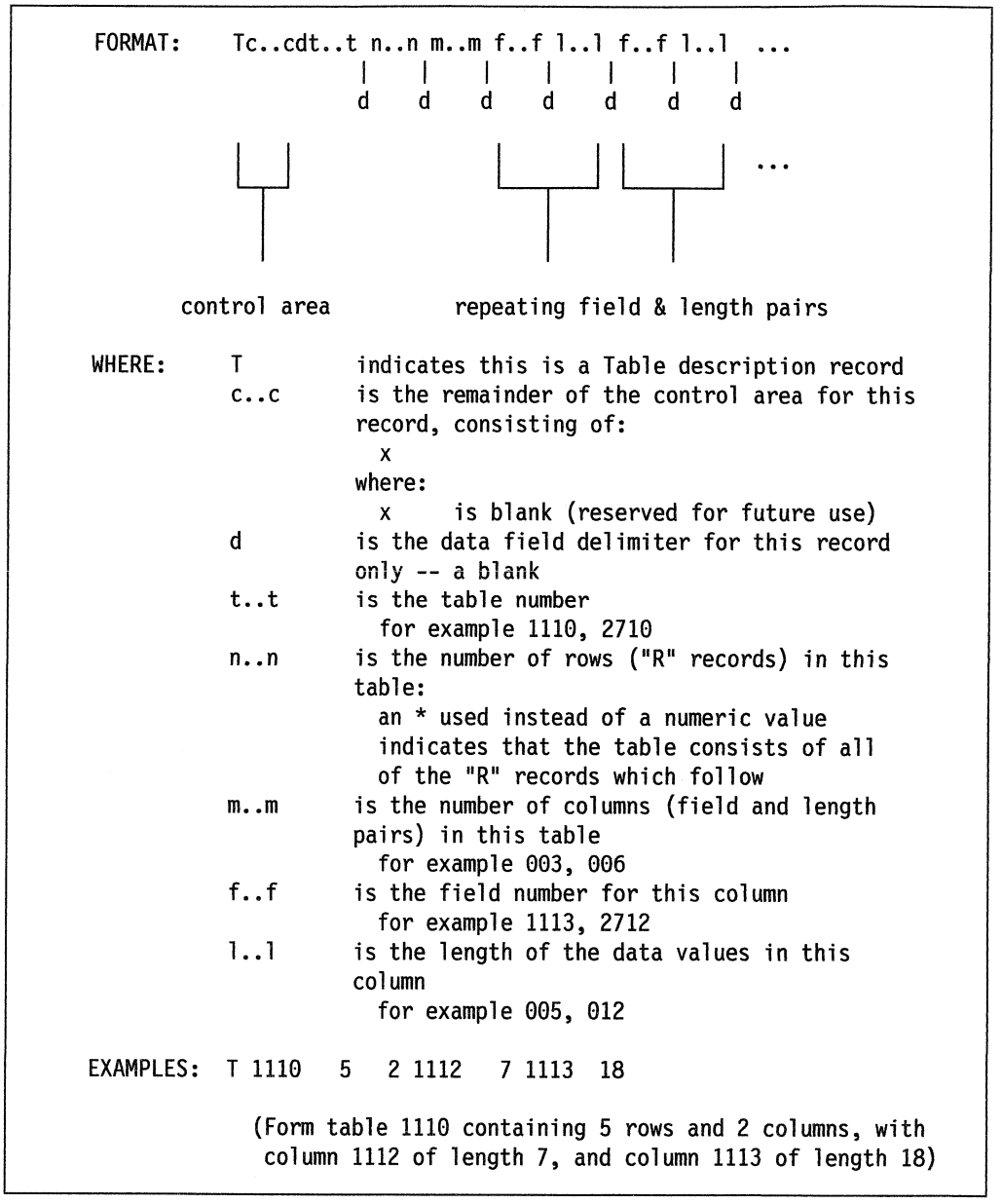


Figure 3-15. Table Record Description

Figure 3-16 on page 3-39 summarizes the location and contents of each of the fields in the T record. The field names used in Figure 3-16 on page 3-39 are defined in Figure 3-15 which describes the entire T record. Object-specific values are noted as appropriate.

Control Area				
Field	Columns	Possible Values	Required on Input	Default if Blank on Input
T	01	T	yes	
c..c	02	(x)	n/a	
x	02	(blank)	n/a	

Remainder of Record				
Field	Offsets past Control Area	Possible Values	Required on Input	Default if Blank on Input
d	+01	(blank)	no	
t..t	+02–05	1001–9999	yes	
n..n	+07–09	* 000–999	yes	
m..m	+11–13	000–999	yes	
f..f	+15–18 +24–27	1001–9999	yes	
l..l	+20–22 +29–31 and so on.	000–999	yes	

Figure 3-16. Table Record Fields

Notes to Figure 3-16:

- An error condition results if the number of R records following the T record does not exactly match the numeric row count specified in the T record.
- The number of f..f / l..l pairs is limited to the number of columns in the given table.
- The number of columns should agree with the number of column field numbers and lengths. If not, a warning message is issued, and the number of columns used is the number of field numbers and column data value lengths in the T record.
- The order of f..f and l..l pairs is arbitrary.
- All of the R records immediately following a T record (that is, those associated with a single table) must contain values of the exact lengths specified for each column in the T record. Records shorter than the implied length result in blank or blank-padded values.
- Query management sets columns with a length of zero (or not specified) to their default values when the object is updated.
- Query management issues a warning message for columns with a specified length of zero to indicate that it set the default value for this column.
- A table with zero rows in it (or not included in the file) has the same effect as applying columns of length zero to the table; query management sets all of the columns to their default values.

- To set a column field to blank, the column must have a positive length in the T record and a blank value in the R record.

Table Row (R) Records

The table row (R) record provides a set of values for a single row in the current table. It consists of an ordered list of values as described by the associated T record. An R record must exactly match the description of the positions and lengths of the data values as specified in the T record.

Figure 3-17 summarizes the contents of the R record.

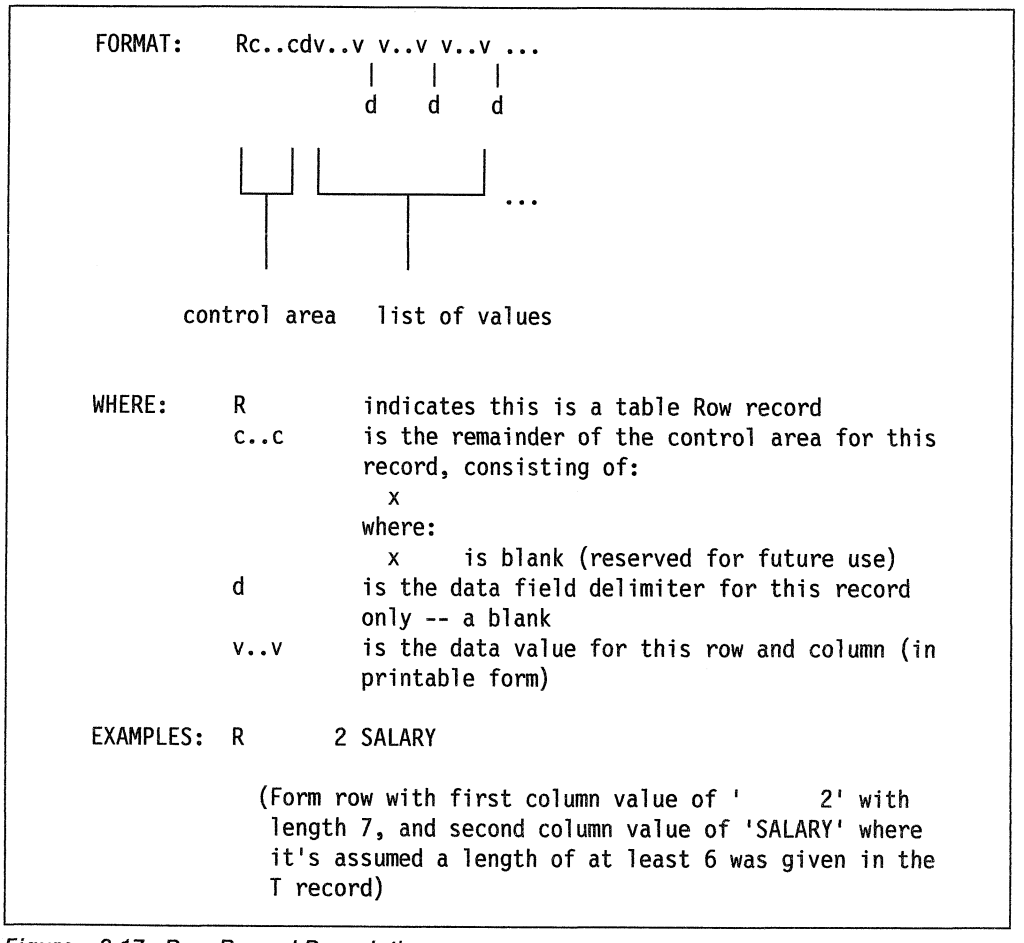


Figure 3-17. Row Record Description

Figure 3-18 on page 3-41 summarizes the location and contents of each of the fields in the R record. The field names used in Figure 3-18 on page 3-41 are defined in Figure 3-17 which describes the entire R record. Object-specific values are noted as appropriate.

Control Area				
Field	Columns	Possible Values	Required on Input	Default if Blank on Input
R	01	R	yes	
c..c	02	(x)	n/a	
x	02	(blank)	n/a	

Remainder of Record				
Field	Offsets past Control Area	Possible Values	Required on Input	Default if Blank on Input
d	+01	(blank)	no	
v..v	+02–xx +(xx+2)–yy +(yy+2)–zz and so on .	(data)	no	(blank)

Figure 3-18. Row Record Fields

Notes to Figure 3-18:

- An R record must immediately follow another R record, or a T record.
- The number of v..v values must exactly match the description in the associated T record.
- A data value length of zero in the associated T record indicates that no value is to be applied to this row and column of the object; it is set to its default value. However, the presence of the field in the T record requires that the R record contain an extra delimiter for this field; a zero-length value results in one delimiter followed by another in the R record.

End-of-Object (E) Record

The end-of-object (E) record delimits the end of the object.

Figure 3-19 summarizes the contents of the E record.

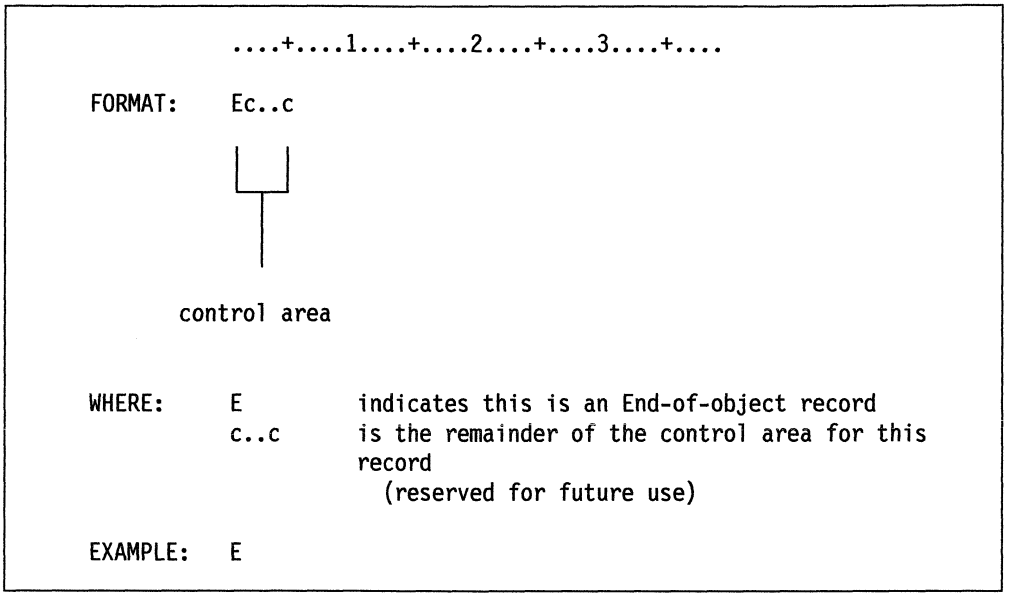


Figure 3-19. End-of-Object Record Description

Figure 3-20 summarizes the location and contents of each of the fields in the E record. The field names used in Figure 3-20 are defined in Figure 3-19 which describes the entire E record. Object-specific values are noted as appropriate.

Control Area				
Field	Columns	Possible Values	Required on Input	Default if Blank on Input
E	01	E	yes	
c..c	02	(x)	n/a	
x	02	(blank)	n/a	

Figure 3-20. End-of-Object Record Fields

Notes to Figure 3-20:

- The E record should be the last record in the external file.
- Those fields (or portions of fields) left unspecified are assumed to be blank if the record is shorter than its fixed format length.

Application Data (*) Record

The application data (*) record allows you to include your own data associated with the given object in the external file. You may choose to use these as comment records to further describe the object in the file.

Figure 3-21 summarizes the contents of the * record.

```
FORMAT:    *v..v

WHERE:     *         indicates this is an application data record
           v..v      is the data value(s) produced by an appli-
                     cation program (preferably in printable form)

EXAMPLE:   * This is the Form that groups by DEPT.

             (comment record in a Form file)
```

Figure 3-21. Application Data Record Description

Figure 3-22 summarizes the location and contents of each of the fields in the * record. The field names used in Figure 3-22 are defined in Figure 3-21 which describes the entire * record. Object-specific values are noted as appropriate.

Control Area				
Field	Columns	Possible Values	Required on Input	Default if Blank on Input
*	01	*	yes	
v..v	02–end	(data)	no	(not used on input)

Figure 3-22. Application Data Record Fields

Notes to Figure 3-22:

- Application data records may appear anywhere in the external file except ahead of the H record.
- Other than validating the format of the record, the * record is ignored and has no effect on the input process.
- Those fields (or portions of fields) left unspecified are assumed to be blank if the record is shorter than its fixed format length.

Figure 3-23 on page 3-44 is an example of an exported query management form. A form may be edited and subsequently imported to obtain the desired report formatting. See the list of record types and field numbers in Figure 3-24 on page 3-48 to match the field numbers to specific report attributes. The * records, which are valid comment records, are used to explain the meaning of certain parts of the form. The examples given are of certain T records with R records (tables) and V records. The same interpretation applies to records of the same types in the remainder of the form.

```
H QM4 01 F 01 E V W E R 01 03 90/3/19 14:27
V 1001 Sales Form
* The 'H' record must be the first record in the file as above.
* The columns table must immediately follow the 'H' record unless there
*  are comment records.
* The T record describes the information which follows in the R records
* The field number '1110' identifies the table as the columns table.
* |    The '005' means that there are 5 columns (R records)
* |    |  following the 'T' record.
* |    |   The '006' means that there are 6 field number, field
* |    |   |   pairs in the T record.
* |    |   |   Starting with '1112' are the field number, field
* |    |   |   pairs which describe the values in the R record.
* |    |   |   For example '1112' corresponds to 'Data type'
* |    |   |   |   (see the table of field numbers) and has a
* |    |   |   |   length of 8.
* |    |   |   |   If I wanted to change the indent for a column
* |    |   |   |     I would look in the table of field numbers
* |    |   |   |     and find that the indent identifier is '1115'.
* |    |   |   |                 Counting the field numbers, starting
* |    |   |   |                 |    with '1112', I find that '1115'
* |    |   |   |                 |    is 3rd in the series of
* |    |   |   |                 |    field length pairs and has a
* |___ |__ |__ |_____            |_____  field width of 6.
* |  | | | | |     |           |     |
T 1110 005 006 1112 008 1114 007 1115 006 1116 005 1117 005 1113 040
*                 The indent value is the third field over in the
*                 |    R records and contains a 3 for every column.
*                 |    The values are left justified and separated
*                 |_   by blank delimiters.
*                 | |
R CHAR             3      6     C     NAME
R CHAR     BREAK1  3      6     C     DEPARTMENT
R NUMERIC  SUM     3      6     L     YEARS
R NUMERIC  SUM     3      6     L     SALARY
R NUMERIC  SUM     3      6     L     COMMISSION
* A 'V' record describes a single attribute in the form.
* The '1201' field number is the blank lines before page
* |  heading attribute.
* |    It has a value length of 1.
* |    |  It has a value of 1.
* |__  |_ |_
* |  | | || |
V 1201 001 1
V 1202 001 2
```

Figure 3-23 (Part 1 of 4). Sample Externalized Form

```
* The following table describes the page heading text.
* The fields used are the line number, alignment, and text respectively.
T 1210 005 003 1212 004 1213 006 1214 055
R 1    CENTER **********************************************
R 2    CENTER ****                                          ****
R 3    CENTER ****           COMPANY REPORT                 ****
R 4    CENTER ****                                          ****
R 5    CENTER **********************************************
*     The '*' in place of a numeric length indicates to use the
*     |  remainder of the record for the length of the data value.
*     |_
*     | |
V 1301 *   1
V 1302 *   2
* Page footing text
T 1310 003 003 1312 004 1313 006 1314 055
R 1    CENTER XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
R 2    CENTER XXXXXXXX Internal Use Only XXXXXXXX
R 3    CENTER XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
V 1401 003 YES
V 1402 001 2
T 1410 005 003 1412 004 1413 006 1414 055
R 1    LEFT   ************************************************
R 2    LEFT   *******                                       ********
R 3    LEFT   *******          END OF REPORT                ********
R 4    LEFT   *******                                       ********
R 5    LEFT   ************************************************
V 1501 *   1
V 1502 003 YES
V 1503 003 YES
V 1505 003 YES
V 1507 003 YES
V 1508 003 YES
V 1510 003 YES
* The following section shows the break information using the
* new format.
*          The value in the '3080' V record indicates the break
*          |  level which applies to all of the break information
*          _|  until the next '3080' V record is encountered.
*         | |  The break level in this example is 1.
V 3080 001 1
V 3101 002 NO
V 3102 002 NO
V 3103 001 0
V 3104 001 0
T 3110 001 003 3112 004 3113 006 3114 055
R  1   CENTER BREAK 1 HEADING
V 3201 002 NO
V 3202 002 NO
V 3203 001 0
V 3204 001 0
T 3210 003 003 3212 004 3213 006 3214 055
```

Figure 3-23 (Part 2 of 4). Sample Externalized Form

```
R  1    CENTER ****************************
R  2    CENTER **** BREAK 1 FOOTING    ****
R  3    CENTER ****************************
* Break level 2 information
V 3080 001 2
V 3101 003 YES
V 3102 003 YES
V 3103 001 0
V 3104 001 0
T 3110 002 003 3112 004 3113 006 3114 055
R 1    CENTER BREAK 2 HEADING
R 2    CENTER ---------------
V 3201 002 NO
V 3202 002 NO
V 3203 001 0
V 3204 001 0
T 3210 003 003 3212 004 3213 006 3214 055
R 1    CENTER ****************************
R 2    CENTER **** BREAK 2 FOOTING    ****
R 3    CENTER ****************************
* Break level 3 information
V 3080 001 3
V 3101 003 YES
V 3102 003 YES
V 3103 001 0
V 3104 001 0
T 3110 002 003 3112 004 3113 006 3114 055
R 1    CENTER BREAK 3 HEADING
R 2    CENTER ---------------
V 3201 002 NO
V 3202 002 NO
V 3203 001 0
V 3204 001 0
T 3210 003 003 3212 004 3213 006 3214 055
R 1    CENTER ****************************
R 2    CENTER **** BREAK 3 FOOTING    ****
R 3    CENTER ****************************
* Break level 4 information
V 3080 001 4
V 3101 003 YES
V 3102 003 YES
V 3103 001 0
V 3104 001 0
V 3201 002 NO
V 3202 002 NO
V 3203 001 0
V 3204 001 0
T 3210 003 003 3212 004 3213 006 3214 055
```

Figure 3-23 (Part 3 of 4). Sample Externalized Form

```
R 1    CENTER ****************************
R 2    CENTER **** BREAK 4 FOOTING    ****
R 3    CENTER ****************************
* Break level 5 information
V 3080 001 5
V 3101 003 YES
V 3102 003 YES
V 3103 001 0
V 3104 001 0
V 3201 002 NO
V 3202 002 NO
V 3203 001 0
V 3204 001 0
* Break level 6 information
V 3080 001 6
V 3101 003 YES
V 3102 003 YES
V 3103 001 0
V 3104 001 0
V 3201 002 NO
V 3202 002 NO
V 3203 001 0
V 3204 001 0
T 3210 003 003 3212 004 3213 006 3214 055
R 1    CENTER ***********************
R 2    CENTER *** BREAK 6 FOOTING ***
R 3    CENTER ***********************
* The 'E' record is the last record in the file. Records after
* the 'E' record will be ignored.
E
```

Figure 3-23 (Part 4 of 4). Sample Externalized Form

You can edit an externalized form object to change your report format. The externalized form object layout is in the encoded format, which uses record types and field number identifiers to represent the form. Each field number identifier in the externalized form object represents a different attribute in the report. After making changes to the externalized form, you must import it for the changes to take effect.

Figure 3-24 on page 3-48 shows the descriptive names of the encoded form fields.

Record Type	Table number	Field number	Description
V		1001	Object Comment
			************************************
			*** Columns section of the report ***
			************************************
T	1110		Column Fields
		1112	--Column data type
		1113	--Column heading
		1114	--Column usage
		1115	--Column indent
		1116	--Column width
		1117	--Column edit
		1118	--Column sequence
			************************************
			*** Page section of the report ***
			************************************
V		1201	Blank lines before heading
V		1202	Blank lines after heading
T	1210		Page heading text table
		1212	--Page heading line number
		1213	--Page heading align
		1214	--Page heading text
V		1301	Blank lines before footing text
V		1302	Blank lines after footing text
T	1310		Page footing text table
		1312	--Page footing line number
		1313	--Page footing align
		1314	--Page footing text
			************************************
			*** Final section of the report ***
			************************************
V		1401	New page for final text
V		1402	Put final summary at line
V		1403	Skip lines before final text
T	1410		Final text table
		1412	--Final text line number
		1413	--Final text align
		1414	--Final text
			**
			** Options fields section of the report **
			**
V		1501	Detail line spacing
V		1502	Outlining for break columns
V		1503	Default break text
V		1505	Column wrapped lines kept on a page
V		1507	Column heading separators
V		1508	Break summary separators
V		1510	Final summary separators

Figure 3-24. Descriptive Names of Encoded Format Form Fields

A new format exists for the break information in the encoded object. To support the forms that use the original format, query management supports both the original format and the new format to describe the break information. An attempt to use a combination of the two formats is not allowed and the import request is ended. All form objects are exported using the new format.

Figure 3-25 is a description of the new format that provides for a break level indicator (V record with field number 3080) to indicate the break level. All of the break information that follows each break level indicator is applied to the break level value in the 3080 V record.

The new format uses a single set of field numbers to describe the break heading and footing information, which allows for more efficient future expansion of the number of break levels supported.

```
Record      Table      Field     Description
 Type       number     number
                                 ************************************
                                 * Break fields section of the report *
                                 ************************************
   V                     3080    Break level indicator
   V                     3101    New page for break heading
   V                     3102    Repeat column headings
   V                     3103    Blank lines before heading
   V                     3104    Blank lines after heading
   T         3110                Break heading table
   V                     3112    --Break heading line number
   V                     3113    --Break heading align
   V                     3114    --Break heading text
   V                     3201    New page for break footing
   V                     3202    Put break summary at line
   V                     3203    Blank lines before footing
   V                     3204    Blank lines after footing
   T         3210                Break footing table
   V                     3212    --Break footing line number
   V                     3213    --Break footing align
   V                     3214    --Break footing text
```

Figure 3-25. Preferred Format for Encoded Break Information

Figure 3-26 on page 3-50 is a description of the original format for representing the break information in the encoded object. This format uses a unique field number for each of the break attributes. This format cannot be used in combination with the new break format.

Record Type	Table number	Field number	Description
			************************************
			* Break fields section of the report *
			************************************
V		1601	Break 1: New page for heading
V		1602	Break 1: Repeat column headings
V		1603	Break 1: Blank lines before heading
V		1604	Break 1: Blank lines after heading
T	1610		Break 1: Heading table
V		1612	--Break 1: Heading line number
V		1613	--Break 1: Heading align
V		1614	--Break 1: Heading text
V		1701	Break 1: New page for break footing
V		1702	Break 1: Put break at summary line
V		1703	Break 1: Blank lines before footing
V		1704	Break 1: Blank lines after footing
T	1710		Break 1: Footing table
V		1712	--Break 1: Footing line number
V		1713	--Break 1: Footing align
V		1714	--Break 1: Footing text
V		1801	Break 2: New page for heading
V		1802	Break 2: Repeat column headings
V		1803	Break 2: Blank lines before heading
V		1804	Break 2: Blank lines after heading
T	1810		Break 2: Heading table
V		1812	--Break 2: Heading line number
V		1813	--Break 2: Heading align
V		1814	--Break 2: Heading text
V		1901	Break 2: New page for break footing
V		1902	Break 2: Put break at summary line
V		1903	Break 2: Blank lines before footing
V		1904	Break 2: Blank lines after footing
T	1910		Break 2: Footing table
V		1912	--Break 2: Footing line number
V		1913	--Break 2: Footing align
V		1914	--Break 2: Footing text
V		2001	Break 3: New page for heading
V		2002	Break 3: Repeat column headings
V		2003	Break 3: Blank lines before heading
V		2004	Break 3: Blank lines after heading
T	2010		Break 3: Heading table
V		2012	--Break 3: Heading line number
V		2013	--Break 3: Heading align
V		2014	--Break 3: Heading text

Figure 3-26 (Part 1 of 3). Original Format for Encoded Break Information.

```
V                2101     Break 3: New page for break footing
V                2102     Break 3: Put break at summary line
V                2103     Break 3: Blank lines before footing
V                2104     Break 3: Blank lines after footing
T        2110             Break 3: Footing table
V                2112     --Break 3: Footing line number
V                2113     --Break 3: Footing align
V                2114     --Break 3: Footing text

V                2201     Break 4: New page for heading
V                2202     Break 4: Repeat column headings
V                2203     Break 4: Blank lines before heading
V                2204     Break 4: Blank lines after heading
T        2210             Break 4: Heading table
V                2212     --Break 4: Heading line number
V                2213     --Break 4: Heading align
V                2214     --Break 4: Heading text

V                2301     Break 4: New page for break footing
V                2302     Break 4: Put break at summary line
V                2303     Break 4: Blank lines before footing
V                2304     Break 4: Blank lines after footing
T        2310             Break 4: Footing table
V                2312     --Break 4: Footing line number
V                2313     --Break 4: Footing align
V                2314     --Break 4: Footing text

V                2401     Break 5: New page for heading
V                2402     Break 5: Repeat column headings
V                2403     Break 5: Blank lines before heading
V                2404     Break 5: Blank lines after heading
T        2410             Break 5: Heading table
V                2412     --Break 5: Heading line number
V                2413     --Break 5: Heading align
V                2414     --Break 5: Heading text

V                2501     Break 5: New page for break footing
V                2502     Break 5: Put break at summary line
V                2503     Break 5: Blank lines before footing
V                2504     Break 5: Blank lines after footing
T        2510             Break 5: Footing table
V                2512     --Break 5: Footing line number
V                2513     --Break 5: Footing align
V                2514     --Break 5: Footing text
```

Figure 3-26 (Part 2 of 3). Original Format for Encoded Break Information.

```
V                    2601     Break 6: New page for heading
V                    2602     Break 6: Repeat column headings
V                    2603     Break 6: Blank lines before heading
V                    2604     Break 6: Blank lines after heading
T          2610               Break 6: Heading table
V                    2612     --Break 6: Heading line number
V                    2613     --Break 6: Heading align
V                    2614     --Break 6: Heading text

V                    2701     Break 6: New page for break footing
V                    2702     Break 6: Put break at summary line
V                    2703     Break 6: Blank lines before footing
V                    2704     Break 6: Blank lines after footing
T          2710               Break 6: Footing table
V                    2712     --Break 6: Footing line number
V                    2713     --Break 6: Footing align
V                    2714     --Break 6: Footing text
```

Figure 3-26 (Part 3 of 3). Original Format for Encoded Break Information.

Using DBCS Data in Query Management

The following sections explain how using double-byte character set (DBCS) data in query management functions is different from using single-byte data.

Input Fields

All query management and Systems Application Architecture (SAA) Query input fields, with the exception of names, allow DBCS data.

Queries

The following query management areas can be either DBCS or mixed single-byte and DBCS:

- Substitution values
- Strings enclosed in quotation marks in character data-type fields
- Comments
- Graphic strings to be entered or compared to graphic data type fields

All SQL keywords must be in English.

Importing DBCS Data

You can import DBCS data in queries, procedures, and forms. When importing DBCS queries and procedures in this way, be certain the record length does not exceed 79 bytes.

Printing DBCS Data

If you are using DBCS data and the page splits, printing resumes on the second and subsequent pages of the report at the fourth-byte position from the left side of the page.

Query Management Objects

The following two SAA Query object types are defined in the *SAA CPI Query Reference* manual:

- Query
- Form

This section discusses how query management maps these SAA Query-specific objects to AS/400 objects. Support for query management requires the definition of two object types: the query management query object and the query management form object. The OS/400 object type for the query management query object is QMQRY. The OS/400 object type for the query management form object is QMFORM. The query management procedure is stored as a single member source physical file.

Query Management CL Commands

The following query management CL commands support the QMQRY and the QMFORM objects:

STRQMQRY Start Query Management Query

CRTQMQRY Create Query Management Query

CRTQMFORM Create Query Management Form

DLTQMQRY Delete Query Management Query

DLTQMFORM Delete Query Management Form

RTVQMQRY Retrieve Query Management Query

RTVQMFORM Retrieve Query Management Form

WRKQMQRY Work with Query Management Query

The WRKQMQRY CL command supports the following CL commands:

DLTQMQRY Delete Query Management Query

STRQMQRY Start Query Management Query

CHGOBJD Change Object Description

WRKQMFORM Work with Query Management Form

The WRKQMFORM CL command supports the following CL commands:

DLTQMFORM Delete Query Management Form

CHGOBJD Change Object Description

Generic Commands

You can run the following generic commands against the QMQRY and QMFORM objects:

MOVOBJ Move Object

RNMOBJ Rename Object

GRTOBJAUT Grant Object Authority

RVKOBJAUT Revoke Object Authority

CHGOBJOWN	Change Object Authority
SAVOBJ	Save Object
SAVCHGOBJ	Save Changed Object
RSTOBJ	Restore Object
DSPOBJD	Display Object Description
DSPOBJAUT	Display Object Authority
CHKOBJ	Check Object
CHGOBJD	Check Object Description
EDTOBJAUT	Edit Object Authority
WRKOBJ	Work with Object
CRTDUPOBJ	Create Duplicate Object

Creating a Query Management Object

Use the following information to create query management objects:

- Query

  Follow these steps to create a query management query (QMQRY) object:

  1. Create the source as a member in a source physical file (a file record length of 91 bytes is recommended) using any of the following methods:
     - Use an editor to type the SQL statement and save it in a file.
     - Use Interactive Structured Query Language (ISQL) to develop the SQL statement interactively, save the session in a file, and then edit it to remove extraneous lines.
     - Use the RTVQMQRY CL command or EXPORT QUERY CPI command to retrieve the query source from a QRYDFN object defined using Query/400.
     - Use the RTVQMQRY CL command or EXPORT QUERY CPI command to retrieve the query source from a QMQRY object created previously.
     - Restore a file containing query source exported from another SAA system environment.
  2. Use the CRTQMQRY CL command or IMPORT QUERY CPI command to create the QMQRY object from the query source.

- Form

  Follow these steps to create a query management form (QMFORM) object:

  1. Create the source as a member in a source physical file (a file record length of 162 bytes is recommended) using any the following methods:
     - Use an editor to type the encoded form and save it in a file.
     - Use the RTVQMFORM CL command or the EXPORT FORM CPI command to retrieve the form source from a QRYDFN object defined using Query/400.
     - Use the RTVQMFORM CL command or the EXPORT FORM CPI command to retrieve the form source from a QMFORM object created previously.
     - Restore a file containing form source exported from another SAA system environment.

2. Use the CRTQMFORM CL command or IMPORT FORM CPI command to create the QMFORM object from the form source.

- Procedure

  The query management procedure object is stored as a single-member source physical file. You can create the source file through AS/400 CL commands or through the query management IMPORT command. The query management procedure can contain up to 408 lines, and it should have a record width of 79 bytes.

  Note: This format is not SAA-compatible and may not successfully export to a non-AS/400 SAA query management object.

- Tables

  Table objects are any single-format, single-member physical or logical files that you can create through SQL/400 conventions or AS/400 database management. Query management does not support manipulation of System/36 multiple format files or AS/400 multiple member files (except in cases where processing defaults to *FIRST).

Chapter 4. Instance Processing

A query management instance is a progression of steps that results in creating a displayed or printed report from the data found in a database file or Query/400 definition. Query management puts the data specified into a DATA set (the active information resulting from running a query) called a query management query (QMQRY) object, which is organized by the query management form (QMFORM) object. By changing the form, you can use the same QMQRY to create multiple reports that are organized according to your needs for a particular situation. This chapter describes how to create, change, and convert a query management instance that creates a report arranged to your needs.

Creating a Query Management Instance

Obtain access to the query management query function by beginning with a control language (CL) command or a user application. Once you access the query function, you can use query management commands to direct query management in creating an instance. An instance is a DATA set containing the data collected from the database file and the global variable pool that contains the DSQ variables used to define the query.

Using the query management instance created by the commands issued, build a printed or displayed report by creating or changing the form in a way that gives you the needed information. Figure 4-1 illustrates how a query management instance is created.

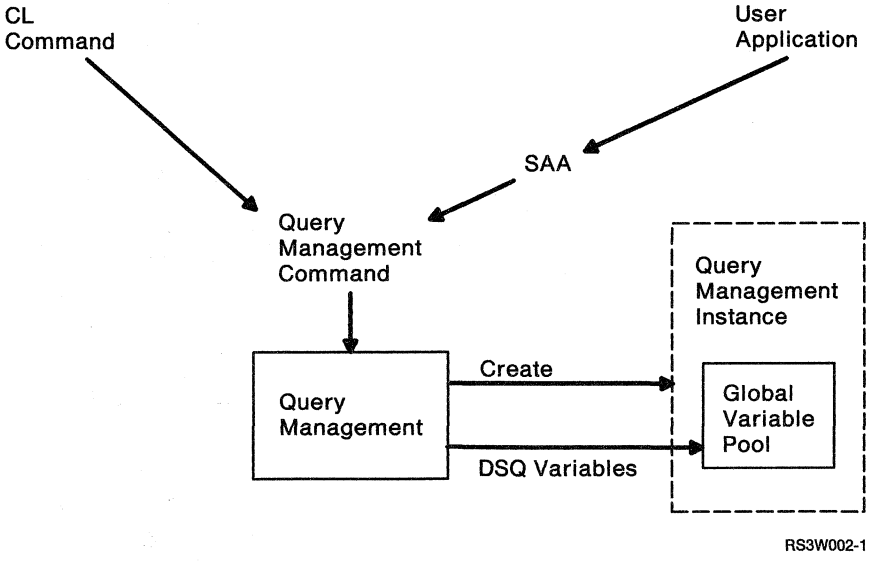


Figure 4-1. Creating a Query Management Instance

Exit the query management instance using the same procedure and the EXIT command. This destroys the instance.

© Copyright IBM Corp. 1991

Running a Query Management/400 Query

Use one of the following methods to run a query management query:

- Specify the Run Query (RUNQRY) command in a procedure.
- Issue the Start Query Management Query (STRQMQRY) CL command
- Run the query from a user application.

Issue a query management command to access the query management function. Query management then creates a QMQRY object (with the name you specified) or changes an existing QMQRY to create the new instance. By using a Structured Query Language (SQL) statement, query management accesses an OS/400 database file and puts the information requested in the QMQRY into the DATA set contained in the instance.

The DATA set created by running the query remains in existence until another query is processed or the associated instance is ended by the EXIT command. A different DATA set is created for each query management instance.

Note: The DATA set is only created if the query is a SELECT statement.

Figure 4-2 illustrates how a query is run using query management.

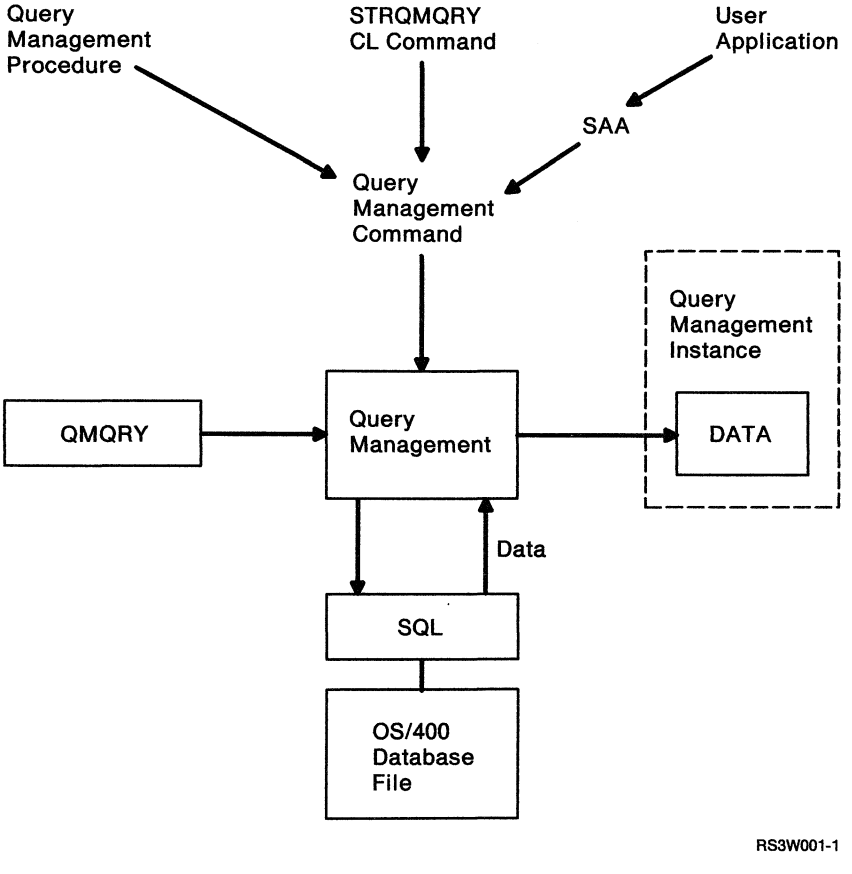


Figure 4-2. Running a Query Management Query

Global Variable Substitution

If you run a query with global variable substitution specified, query management processes the request in the same manner as described previously, except the global variable pool defined when the instance was created is searched to resolve global variables. If the variable specified in the query is not set in the global variable pool, query management sends a message to the display prompting you for the value to be used in the field specified. Enter a valid value for the prompted field and the query will be run with the variable value entered at the prompt.

Creating Query Management Reports

Once you have created the DATA set, you can request that a report be printed or shown at the display station. Use the Create Query Management Form (CRTQMFORM) command or the Work with Query Management Form (WRKQMFORM) command to create or change a QMFORM object that puts the data in the DATA set into a form specific to your needs.

Once the QMFORM and the DATA set are created, use the Display Report display to show the report on your display station, or use the PRINT command to produce a printed version of the report data.

Note: Creating a report requires that you have already run a query to create a DATA set.

Figure 4-3 illustrates how to display or print a report using query management guidelines.

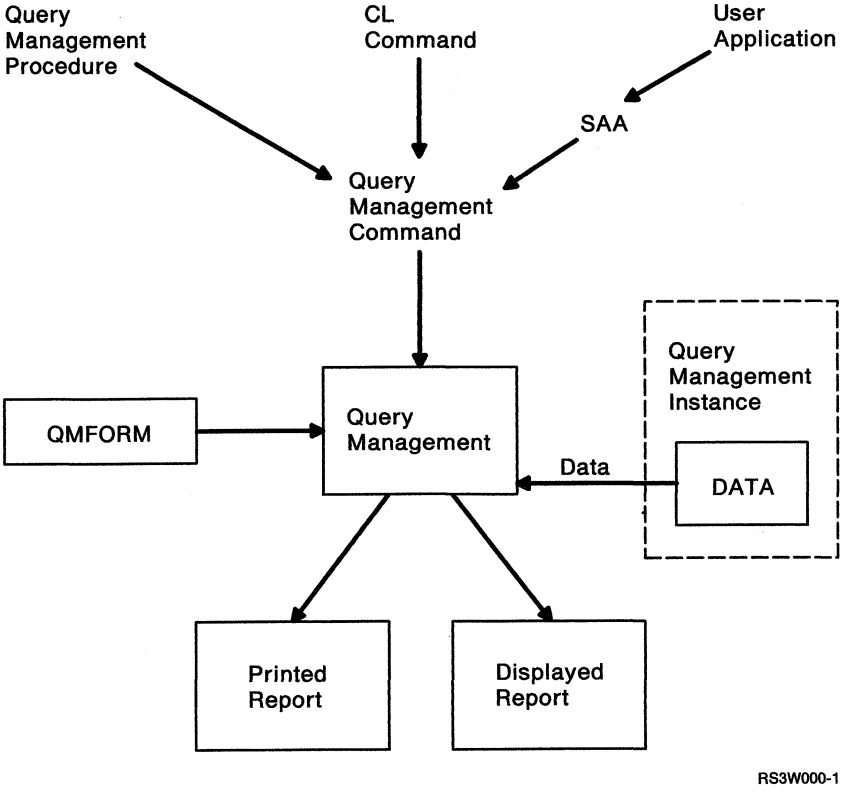


Figure 4-3. Creating a Query Management Report

Importing a Query or Form Object

Use the following methods to start the process of importing a query or form object for use in creating a query management report:

- Specify the IMPORT command in a procedure.
- Issue the Create Query Management Query (CRTQMQRY) or Create Query Management Form (CRTQMFORM) CL command.
- Import the query or form object from a user application using the IMPORT command.

Use query management commands to request the data be imported from a source file member that contains the query or form source to create the specified query or form.

Exporting a Query or Form Object

Use the following methods to start the process of exporting a query or form object for use in creating a query management report:

- Specify the EXPORT command in a procedure.
- Issue the Retrieve Query Management Query (RTVQMQRY) or Retrieve Query Management Form (RTVQMFORM) CL command.
- Export the query or form from a user application using the EXPORT command.

Use query management commands to send the data to a source file member to contain the query or form information.

Importing and Exporting a Query Management Procedure

The process for importing and exporting query management procedures is the same as for importing and exporting queries and forms with one exception. A query management procedure is a source physical file member. An import or export of a procedure copies the information from one member to another. Therefore, it is not necessary to import or export a procedure, because it is already in the format needed to transfer it to another SAA system.

Figure 4-4 on page 4-5 illustrates the process for importing and exporting queries, forms, and procedures.

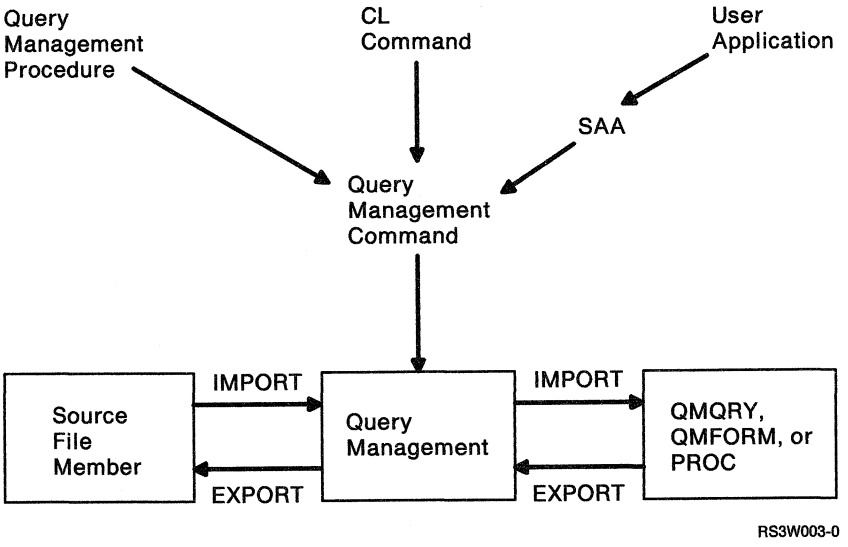


Figure 4-4. Importing and Exporting Query Management Members

Running a Query Management Procedure

Use one of the following methods to start working with procedures in the query management environment:

- Specify the procedure from inside another procedure using the RUN command.
- Issue the Start Query Management Procedure (STRQMPRC) CL command.
- Start the procedure from a user application using the RUN command.

Use query management commands to request the data from a source file member or another procedure for query management to use in creating a query management instance. The commands in the procedure are processed using the same instance as the instance associated with the RUN PROC command.

Query management also allows you to call multiple procedures when creating an instance using this process. Figure 4-5 illustrates how to run a query management procedure.

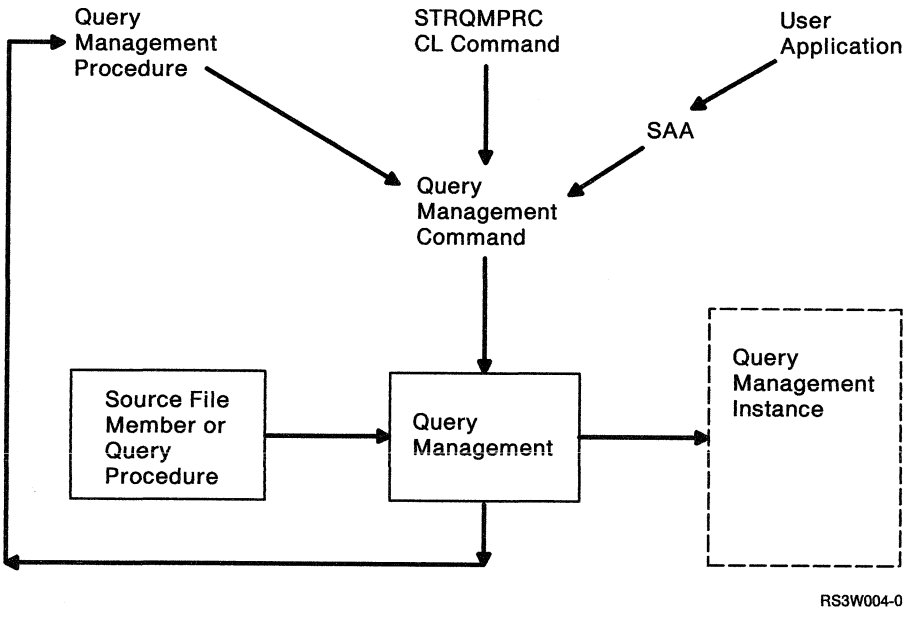


Figure 4-5. Running Query Management Procedures

Using the Save Data As Command

Query management allows you to save the data created in the DATA set of your instance to a query management table. Use the following methods to start query management processing when working with the Save Data As command:

- Specify the save operation from a procedure using the SAVE DATA AS command.
- Issue the Start Query Management Query (STRQMQRY) CL command.
- Start the command from a user application using the SAVE DATA AS command.

Use query management commands to request that the data from the DATA set in an instance created previously be used to save the data in an OS/400 database file to a query management table. You must have processed a RUN QUERY command under the same instance to create the query management DATA set.

Figure 4-6 illustrates how to save the data in a database file to a table using a query management instance.

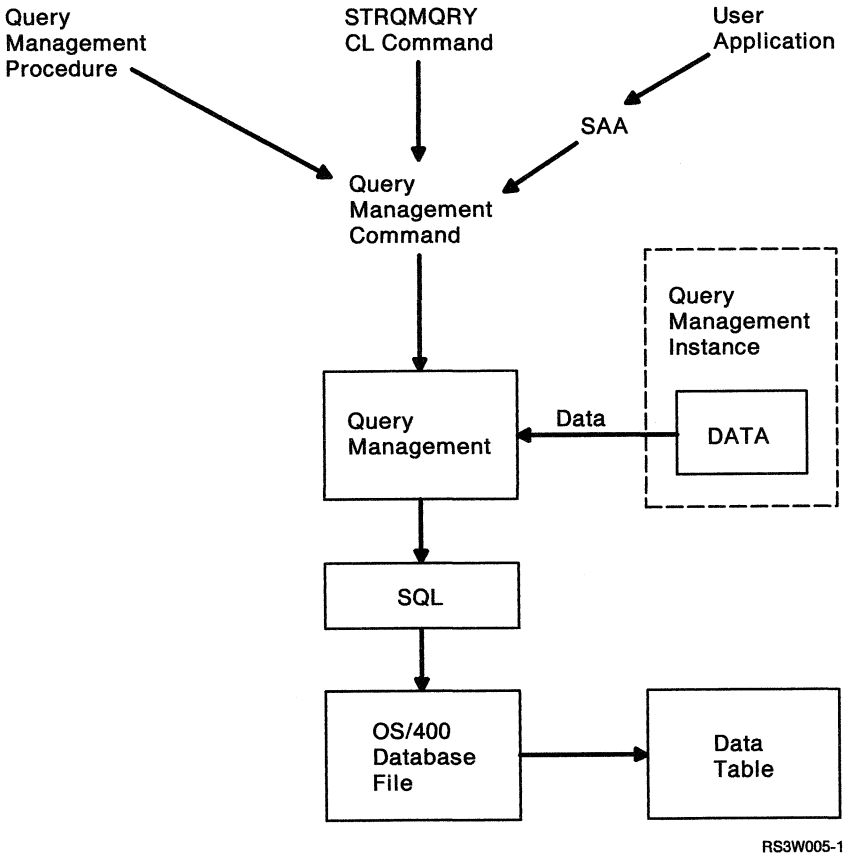


Figure 4-6. Saving Data to a Query Management Table

Using SET GLOBAL and GET GLOBAL Commands

Query management allows you to get and change variables in the global variable pool. Use the GET GLOBAL command to get the value of a query management variable in the previously created instance and provide it to a user program or procedure.

Use the SET GLOBAL command to set or change the value of a query management variable in the previously created instance from a user program or procedure. Use the following methods to start query management processing when working with the GET GLOBAL and SET GLOBAL commands:

- Specify the command from a procedure.
- Issue the Start Query Management Query (STRQMQRY) CL command.
- Start the command from a user application.

Use query management commands to request that the values from the global variable pool in an instance created previously be set from a user program or procedure. The GET GLOBAL process is the same as the SET process, except query management gets the variable values *for* a user program in the GET GLOBAL process.

Figure 4-7 illustrates how query management uses the GET GLOBAL and SET GLOBAL commands to change or retrieve query management variables.

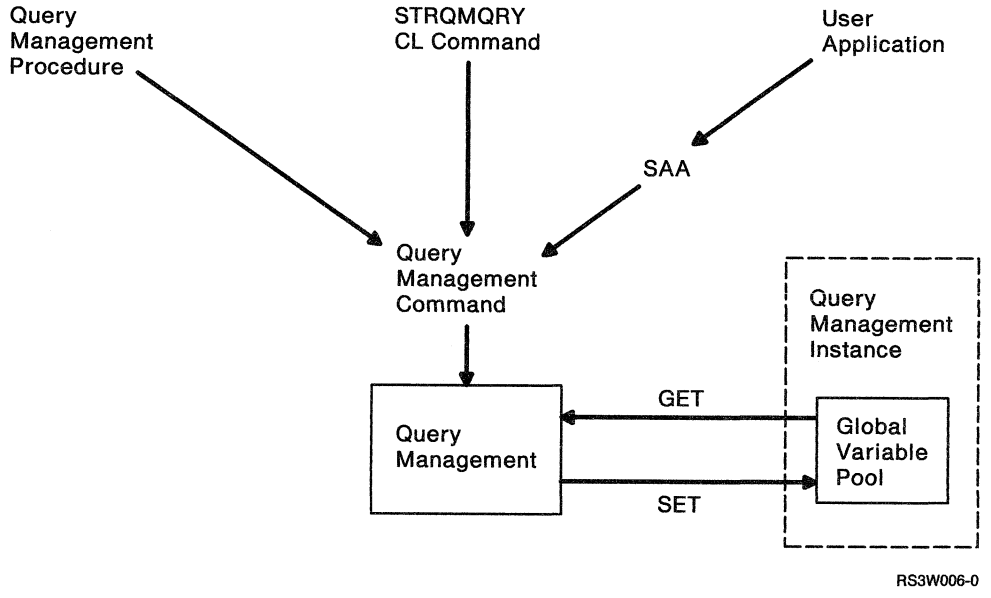


Figure 4-7. Using GET GLOBAL and SET GLOBAL Commands

Chapter 5. Using Query Management in HLL Programs

This chapter describes the high-level languages you can use to access the query management callable interface (CI). Query management supports the following languages:

- C/400*
- COBOL/400*
- RPG/400*

C Language Interface

Access the query management CI using normal C function calls. The exact description of each function call is provided in the CI C communications include file DSQCOMMC. The communications include file is unique for each operating system. Query management provides two function calls — DSQCIC and DSQCICE. Use DSQCIC to run query management commands that do not require access to program variables. Use DSQCICE to run commands that *do* require access to program variables. Use the following commands with the DSQCICE function:

- START
- SET GLOBAL
- GET GLOBAL

Specify all other query management commands using the DSQCIC function. For more information on the C language interface, see the *SAA CPI Query Reference* manual.

Example DSQCOMMC

Figure 5-1 on page 5-2 shows the AS/400 version of the query management CI C communications macro.

© Copyright IBM Corp. 1991

```
/*****************************************************************/
/*                                                               */
/*  NAME:  dsqcommc.h                                            */
/*                                                               */
/*  MODULE-TYPE: IBM C/400 Query Management Interface include file */
/*                                                               */
/*  PROCESSOR:  C                                                */
/*                                                               */
/*  DESCRIPTION:                                                 */
/*      This include file contains the declarations needed       */
/*      by a C application program for interfacing               */
/*      with the query management callable interface.            */
/*      query management is the AS/400 implementation of the     */
/*      Systems Application Architecture Query Callable          */
/*      Programming Interface.                                   */
/*                                                               */
/*  Copyright: 5728-SS1 (C) COPYRIGHT IBM CORP. 1989             */
/*                                                               */
/*****************************************************************/


/*****************************************************************/
/* Callable Interface Constants and Structures                   */
/*****************************************************************/

/* return code values for DSQ_RETURN_CODE                        */
#define DSQ_SUCCESS        0   /* successful running of the request  */
#define DSQ_WARNING        4   /* normal completion with warnings    */
#define DSQ_FAILURE        8   /* command did not process correctly  */
#define DSQ_SEVERE        16   /* severe error; SAA Query session    */
                               /* ended.                             */

/* Variable data types */
#define DSQ_VARIABLE_CHAR "CHAR"    /* unsigned character data type  */
#define DSQ_VARIABLE_FINT "FINT"    /* long integer type             */

/* Cancel indicator                                              */
#define DSQ_CANCEL_YES    "1"       /* Yes it was canceled.        */
#define DSQ_CANCEL_NO     "0"       /* No, it was not canceled.    */

/* Derived query/form indicator                                  */
#define DSQ_DERIVED_YES   "1"       /* Yes it was derived from QRYDFN*/
#define DSQ_DERIVED_NO    "0"       /* No, it was not derived      */

/* Yes/No indicator.  This indicator can be used to test the values  */
/*   returned for the following global variables:                */
/*      DSQCATTN - Last command cancel indicator.                */
/*      DSQAROWC - Current data completed indicator.             */
/*                                                               */
#define DSQ_YES           "1"       /* Yes                       */
#define DSQ_NO            "0"       /* No                        */

/* misc defines                                                  */
#define DSQ_TRUE           1        /* indicates TRUE            */
#define DSQ_FALSE          0        /* indicates FALSE           */
#define DSQ_MATCH          0        /* match indicator           */
```

Figure 5-1 (Part 1 of 2). Example DSQCOMMC

```
/* define the Communication Area structure */
struct dsqcomm
  {
  unsigned long dsq_return_code;      /* function return code         */
  unsigned long dsq_instance_id;      /* instance id for this session */
  unsigned char dsq_reserve1[44]; /* reserved space -       */
                                      /* not for application use      */
  unsigned char dsq_message_id[8];  /* completion message id*/
  unsigned char dsq_q_message_id[8];     /* query message id*/
  unsigned char dsq_start_parm_error[8];       /* start parm*/
  unsigned char dsq_cancel_ind[1];   /* command canceled by */
                                       /* Control-Break (1=yes,0=no)*/
  unsigned char dsq_reserve2[17];       /* reserved space - */
                                        /*not for application use   */
  unsigned char dsq_query_derived[1];     /* query used was */
                                       /*derived from AS/400 *QRYDFN */
  unsigned char dsq_form_derived[1];/*form used was derived */
                                        /*   from AS/400 *QRYDFN    */
  unsigned int  dsq_delete_env;         /* flag used by QM to       */
                                        /*  control the QM          */
                                        /*  environment.            */
  unsigned char dsq_reserve3[924];    /* Reserve area 3     */
                                        /* application use          */
  } ;

/*********************************************************************/
/* Callable Interface External Function/Routine Definition           */
/*********************************************************************/

/* pragma definitions */
#define dsqcice DSQCICE
#define dsqcic DSQCIC
#pragma linkage(DSQCIC, OS)
#pragma linkage(DSQCICE, OS)

/* prototype for DSQCICE  */
extern void dsqcice  (
            struct dsqcomm *,      /* Communication Area           */
            signed long  *,        /* command length               */
            char *,                /* command                      */
            signed long *,         /* number of parms              */
            signed long *,         /* keyword lengths              */
            char *,                /* keywords                     */
            signed long *,         /* data lengths                 */
            void *,                /* data                         */
            char *);               /* data value type              */

/* prototype for DSQCIC   */
extern void dsqcic   (
            struct dsqcomm *,      /* Communication Area           */
            signed long  *,        /* command length               */
            char *);               /* command                      */
```

Figure 5-1 (Part 2 of 2). Example DSQCOMMC

COBOL Language Interface

Use normal COBOL function calls to access the query management CI. The exact description of each function call is provided in the query management COBOL communications macro DSQCOMMB. The communications macro is unique for each operating system. Query management provides a function called DSQCIB, which is used to run all query management commands. The parameters passed on the call to DSQCIB determine whether program variables are being passed. Program variables must be passed on the following query management commands:

- START
- SET GLOBAL
- GET GLOBAL

No other query management commands specify program variables. For more information on the COBOL language interface, see the *SAA CPI Query Reference* manual.

DSQCIB Function Syntax

The following command string is an example of using the DSQCIB function syntax:

```
CALL DSQCIR USING DSQCOMM, CMDLTH, CMDSTR.
```

where:

- *DSQCOMM* is the structure DSQCOMM.
- *CMDLTH* is the length of the command string CMDSTR. The length is specified as an integer PIC 9(8) variable.
- *CMDSTR* is the SAA Query command to run. The command string is specified as a character string of the length specified by CMDLTH.

DSQCIB Extended Function Syntax

The following command string is an example of using the DSQCIB *extended* function syntax:

```
CALL DSQCIB USING
            DSQCOMM CMDLTH CMDSTR
            PNUM KLTH KWORD VLTH VALUE VTYPE.
```

where:

- *DSQCOMM* is the structure DSQCOMM.
- *CMDLTH* is the length of the command string CMDSTR. The length is specified as an integer PIC 9(8) variable.
- *CMDSTR* is the SAA Query command to run. The command string is specified as a character string of the length specified by CMDLTH.
- *PNUM* is the number of command keywords. PNUM is specified as an integer PIC 9(8) variable.
- *KLTH* is the length of each specified keyword. The length of the keyword or keywords is specified as an integer PIC 9(8) variable or variable array.
- *KWORD* is the query management keyword or keywords.

  The keyword string is specified as a character or a structure of characters whose length is the same as specified by KLTH. You can use an array of characters, provided all of the characters are of the same length.

- *VLTH* is the length of each value associated with the keyword.

  The length of the associated values is specified as an integer PIC 9(8) variable or variable array.

- *VALUE* is the value associated with each keyword.

  The value string is specified as a character or a structure of characters or an integer PIC 9(8) variable or variable array. The type is specified in the VTYPE parameter.

- *VTYPE* indicates the query management data type of the value string VALUE.

  The value string contains one of the following values that is provided in the query management communications macro:

  - DSQ-VARIABLE-CHAR indicates that value is character.
  - DSQ-VARIABLE-FINT indicates that value is an integer PIC 9(8).

Example DSQCOMMB

Figure 5-2 is an example of the SAA Query CI COBOL communications macro that has been tailored to fit the AS/400 environment.

```
****************************************************************
*                                                              *
*  NAME:  DSQCOMMB                                            *
*                                                              *
*  MODULE-TYPE: IBM COBOL/400 Query Management Interface       *
*               include file                                   *
*                                                              *
*  PROCESSOR: COBOL                                            *
*                                                              *
*  DESCRIPTION:                                                *
*      This include file contains the declarations needed      *
*      by a COBOL/400 application program for interfacing      *
*      with the query management callable interface.           *
*      query management is the AS/400 implementation of the    *
*      Systems Application Architecture Query Callable         *
*      Programming Interface.                                  *
*                                                              *
*  Copyright: 5728-SS1 (C) COPYRIGHT IBM CORP. 1989            *
*                                                              *
****************************************************************

* Structure declare for communications area
  01 DSQCOMM.
     03 DSQ-RETURN-CODE      PIC 9(8) USAGE IS BINARY VALUE 0.
*                             * Function return code       *
     03 DSQ-INSTANCE-ID      PIC 9(8) USAGE IS BINARY VALUE 0.
*                             * Identifier from START cmd  *
     03 DSQ-RESERVE1         PIC X(44).
*                             * Reserved area              *
     03 DSQ-MESSAGE-ID       PIC X(8).
*                             * Completion message id      *
```

Figure 5-2 (Part 1 of 3). Example DSQCOMMB

```
     03 DSQ-Q-MESSAGE-ID     PIC X(8).
*                              * Query message ID          *
     03 DSQ-START-PARM-ERROR PIC X(8).
*                              * START parameter in error  *
     03 DSQ-CANCEL-IND       PIC X(1).
*                              * 1 = Command canceled      *
*                              * 0 = Command not canceled  *
     03 DSQ-RESERVE2         PIC X(17).
*                              * Reserved space -- not for *
*                              * application use           *
     03 DSQ-QUERY-DERIVED    PIC X(1).
*                              * 1 = Query was derived from *
*                              *      AS/400 QRYDFN        *
*                              * 0 = Query was not derived *
*                              *      from AS/400 QRYDFN   *
     03 DSQ-FORM-DERIVED     PIC X(1).
*                              * 1 = Form was derived from *
*                              *      AS/400 QRYDFN        *
*                              * 0 = Form was not derived  *
*                              *      from AS/400 QRYDFN   *
     03 DSQ-DELETE-ENV       PIC 9(8) USAGE IS BINARY VALUE 0.
*                              * Flag used to delete env.  *
     03 DSQ-RESERVE3         PIC X(924).
*                              * Reserved space -- not for *
*                              * application use           *


*  Return code values for DSQ-RETURN-CODE
  01 DSQ-SUCCESS             PIC 9(8) USAGE IS BINARY VALUE 0.
  01 DSQ-WARNING             PIC 9(8) USAGE IS BINARY VALUE 4.
  01 DSQ-FAILURE             PIC 9(8) USAGE IS BINARY VALUE 8.
  01 DSQ-SEVERE              PIC 9(8) USAGE IS BINARY VALUE 16.

*  Callable Interface program name
  01 DSQCIB                  PIC X(7) VALUE "QQXMAIN".

*  Values for variable type on CALL parameter
  01 DSQ-VARIABLE-CHAR       PIC X(4) VALUE "CHAR".
  01 DSQ-VARIABLE-FINT       PIC X(4) VALUE "FINT".

*  Values for query/form derived field in communications area
  01 DSQ-DERIVED-NO          PIC X(1) VALUE "0".
  01 DSQ-DERIVED-YES         PIC X(1) VALUE "1".
```

Figure 5-2 (Part 2 of 3). Example DSQCOMMB

```
*  Values for the cancel indicator field in communications area
  01 DSQ-CANCEL-YES         PIC X(1) VALUE "1".
  01 DSQ-CANCEL-NO          PIC X(1) VALUE "0".

*  Yes/No indicator.  This indicator can be used
*   to test the values
*   returned for the following global variables:
*     DSQCATTN - Last command cancel indicator.
*     DSQAROWC - Current data completed indicator.
*
  01 DSQ-YES                PIC X(1) VALUE "1".
  01 DSQ-NO                 PIC X(1) VALUE "0".
```

Figure 5-2 (Part 3 of 3). Example DSQCOMMB

RPG Language Interface

Access the query management CI using normal RPG function calls. The exact description of each function call is provided in the query management RPG communications include member DSQCOM. Query management provides a function called DSQCIR that is used to run all query management commands. The parameters that are passed on the call to DSQCIR determine whether program variables are being passed. Program variables must be passed on the following query management commands:

- START
- SET GLOBAL
- GET GLOBAL

No other query management commands specify program variables.

DSQCIR Function Syntax

The following command string is an example of using the DSQCIR function syntax:

```
C                   CALL DSQCIR
C                   PARM           DSQCOM
C                   PARM           CMDLTH
C                   PARM           CMDSTR
```

where:

- *DSQCOM* is the structure DSQCOM.
- *CMDLTH* is the length of the command string CMDSTR. The length is specified as a 4-byte binary field.
- *CMDSTR* is the SAA Query command to process. The command string is specified as a character string of the length specified by CMDLTH.

DSQCIR Extended Function Syntax

The following command string is an example of using the DSQCIR *extended* function syntax:

```
C                   CALL DSQCIR
C                   PARM           DSQCOM
C                   PARM           CMDLTH
C                   PARM           CMDSTR
C                   PARM 1         PNUM
C                   PARM           KLTH
C                   PARM           KWORD
C                   PARM           VLTH
C                   PARM           VALUE
C                   PARM           VTYPE
```

where:

- *DSQCOM* is the structure DSQCOM.
- *CMDLTH* is the length of the command string CMDSTR. The length is specified as a 4-byte binary field.
- *CMDSTR* is the SAA Query command to run. The command string is specified as a character string of the length specified by CMDLTH.
- *PNUM* is the number of command keywords. PNUM is specified as a 4-byte binary field.
- *KLTH* is the length of each specified keyword. The length of the keyword or keywords is specified as a 4-byte binary field.
- *KWORD* is the query management keyword or keywords.

  The keyword string is specified as a character or a structure of characters whose length is the same as specified by KLTH. An array of characters may be used, provided all of the characters are of the same length.
- *VLTH* is the length of each value associated with the keyword. The length of the associated values is specified as a 4-byte binary field.
- *VALUE* is the value associated with each keyword.

  The value string is specified as a character or a structure of characters or a 4-byte binary field. The type is specified in the VTYPE parameter.
- *VTYPE* indicates the query management data type of the value string VALUE.

  The value string contains one of the following values that is provided in the query management communications include member:

  - DSQVCH indicates that value is character.
  - DSQVIN indicates that value is an integer (4-byte binary).

Interface Communications Area (DSQCOM)

The query management interface communications area is part of the communications include member DSQCOM. The interface communications area is described as a structure named DSQCOM.

The interface communications area DSQCOM contains the information shown in Figure 5-3. This information must *not* be altered by the calling program.

Figure 5-3. DSQCOM Programming Information

Variable	Type	Length (bytes)	Description
DSQRET	Binary	4 bytes	Integer that indicates the status of query management processing after a command is run.
DSQINS	Binary	4 bytes	Identifier that is established by query management when processing the START command.
DSQRES	Character	44 bytes	Reserved for future use.
DSQMSG	Character	8 bytes	Completion message ID.
DSQQMG	Character	8 bytes	Query message ID.
DSQSPE	Character	8 bytes	Parameter in error when START failed due to a parameter error.
DSQCNL	Character	1 byte	Command cancel indicator; indicates whether the user had canceled command processing while query management was running a command: DSQCLY "VALUE 1" DSQCLN "VALUE 0"
DSQRS2	Character	17 bytes	Reserved for future use.
DSQQDR	Character	1 byte	Query was derived from a Query/400 QRYDFN. DSQDRY "VALUE 1" DSQDRN "VALUE 0"
DSQFDR	Character	1 byte	Form was derived from a Query/400 QRYDFN. DSQDRY "VALUE 1" DSQDRN "VALUE 0"
DSQRS3	Character	156 bytes	Reserved for future use.
DSQRS4	Character	256 bytes	Reserved for future use.
DSQRS5	Character	256 bytes	Reserved for future use.
DSQRS6	Character	256 bytes	Reserved for future use.

Example DSQCOMMR

Figure 5-4 is an example of the query management CI RPG communications include member. This version of the communications include member has been tailored for the AS/400 system environment.

```
I*****************************************************************
I*                                                               *
I*  NAME:  DSQCOMMR                                              *
I*                                                               *
I*  MODULE-TYPE: IBM RPG/400 Query Management Include File       *
I*                                                               *
I*  PROCESSOR: RPG                                               *
I*                                                               *
I*  DESCRIPTION:                                                 *
I*      This include file contains the declarations needed       *
I*      by an RPG/400 application program for interfacing        *
I*      with the query management callable interface.            *
I*      query management is the AS/400 implementation of the     *
I*      Systems Application Architecture Query Callable          *
I*      Programming Interface.                                   *
I*                                                               *
I*  Copyright: 5728-SS1 (C) COPYRIGHT IBM CORP. 1989             *
I*                                                               *
I*****************************************************************
I*****************************************************************
I*               QUERY INTERFACE INCLUDE                         *
I*                                                               *
I* DSQCOM Definition, contains QUERY interface variables:        *
I*                                                               *
I*           DSQRET     - Status of QUERY processing            *
I*           DSQINS     - SAA QUERY identifier                  *
I*           DSQRS1     - Reserved                              *
I*           DSQMSG     - Completion message-ID                 *
I*           DSQQMG     - QUERY message ID                      *
I*           DSQSPE     - START fail parameter error            *
I*           DSQCNL     - Command cancel indicator              *
I*           DSQQDR     - Query was derived from AS/400 QRYDFN  *
I*           DSQFDR     - Form was derived from AS/400 QRYDFN   *
I*           DSQDEN     - Environment deletion indicator        *
I*           DSQRS2     - Reserved                              *
I*           DSQRS3     - Reserved                              *
I*           DSQRS4     - Reserved                              *
I*           DSQRS5     - Reserved                              *
I*           DSQRS6     - Reserved                              *
I*                                                               *
I*****************************************************************
IDSQCOM      DS
```

Figure 5-4 (Part 1 of 3). Example DSQCOMMR

```
I                                     B   1   40DSQRET
I                                     B   5   80DSQINS
I                                         9   52 DSQRS1
I                                        53   60 DSQMSG
I                                        61   68 DSQQMG
I                                        69   76 DSQSPE
I                                        77   77 DSQCNL
I                                        78   94 DSQRS2
I                                        95   95 DSQQDR
I                                        96   96 DSQFDR
I                                        97  100 DSQDEN
I                                       101  356 DSQRS3
I                                       357  612 DSQRS4
I                                       613  868 DSQRS5
I                                      8691024 DSQRS6
I*


I* DSQRET - DSQ return code meanings
I*                    SUCCESS       --     value  0
I*                    WARNING       --     value  4
I*                    FAILURE       --     value  8
I*                    SEVERE        --     value 16

I*
I             0                      C         DSQSUC
I             4                      C         DSQWAR
I             8                      C         DSQFAI
I             16                     C         DSQSEV
I*
I* DSQCNL - DSQ cancel indicator meanings
I*                    CANCEL YES    --     value '1'
I*                    CANCEL NO     --     value '0'
I*
I             '1'                    C         DSQCLY
I             '0'                    C         DSQCLN
I*
I* DSQQDR/DSQFDR - DSQ QRYDFN derivation indicator meanings
I*                    DERIVED YES   --     value '1'
I*                    DERIVED NO    --     value '0'
I*
I             '1'                    C         DSQDRY
I             '0'                    C         DSQDRN
I*
I* DSQYES/DSQNO  - DSQ constants for the values returned
I*                   for the following global variables:
I*                     DSQCATTN - Last command cancel indicator.
I*                     DSQAROWC - Current data completed indicator.
I*
I             '1'                    C         DSQYES
I             '0'                    C         DSQNO
I*
```

Figure 5-4 (Part 2 of 3). Example DSQCOMMR

```
I* DSQCIR - Interface program call name definition
I*
I              'QQXMAIN'             C         DSQCIR
I*
I* DSQVCH - contains constant value 'CHAR'
I* DSQVIN - contains constant value 'FINT'
I*
I              'CHAR'                C         DSQVCH
I              'FINT'                C         DSQVIN
I*
I*           END OF DSQCOM QUERY INCLUDE
I*************************************************************
```

Figure 5-4 (Part 3 of 3). Example DSQCOMMR

Chapter 6. Subprogram Use and CPI Handling

You may wish to use subprograms to access the query management callable interface (CI). Subprograms relieve you of most of the data manipulation necessary to access the CI. This chapter describes subprograms and how to use them in handling queries.

Subprogram Use

This section describes the listings for seven subprograms that represent the more common functions performed. Create different subprograms if you find there are other commonly used functions in your particular environment.

When using the subprograms, consider the following issues:

- If the calling program calls the subprogram only once (or infrequently), end the subprogram when returning to the calling program. Refer to *RPG/400* User's Guide* for more details on calling other programs.
- Once the CI is started, an instance identifier is allocated. Therefore, the data structure DSQCOM is passed from program to program. You must pass this instance identifier to the CI for each access under that session.
- If the application programs using these subprograms are run on an AS/400 system different from the one that created the subprograms, object code versions of these subprograms need to be on the AS/400 system running the programs.
- If the application programs using these subprograms are run on a non-AS/400 system, object code versions of programs that perform the same function must be created on that system.

  Note: The code described in this chapter is written in RPG/400 language and does not necessarily compile on non-AS/400 systems.
- The code provided in this chapter is written in RPG, but you can develop similar functions in other programming languages. You can also access the RPG/400 subprograms, once compiled, from a COBOL program.

Description of Subprograms

The following sections describe how you can use subprograms to accomplish the commonly used query management functions. Following each description is the example subprogram created to accomplish the query management task.

START Subprogram

The values for the keywords DSQSMODE, DSQSCMD, DSQSRUN, and DSQSNAME are passed to this program as a string of 132 characters. The first 33 characters represent the DSQSMODE keyword value, the next 33 characters represent the DSQSCMD keyword value, the next 33 the DSQSRUN keyword value, and the last 33 the DSQSNAME keyword value. Left-justify the keyword values you type. If a keyword value is not used, it still must be passed, but as a string of 33 blank characters.

The START subprogram reads the passed keyword values string, tests for blank values, and calculates the lengths of the values. It also strings together the start command, keywords, and keyword values with the necessary lengths and calls the

© Copyright IBM Corp. 1991

programmable interface. The interface is started and the START subprogram is ended with control returned to the calling program.

```
*****************************************************************
*                                                               *
*        START COMMAND CPI QUERY INTERFACE HANDLER              *
*        ----------------------------------------               *
*                                                               *
*  1) Include member DSQCOMMR contains the communications       *
*     area to be passed to the query management interface.      *
*  2) This program handles the START CPI QM interface           *
*     command. It reads the DSQ keywords information to be      *
*     processed, reformats it, then passes it to the interface  *
*  3) The keyword information is passed to this program in the  *
*     form of 4 values which are the 4 keyword values to be     *
*     passed to query management. This program calculates       *
*     the length of each keyword name and keyword value and     *
*     strings the necessary information into arrays for         *
*     passing to the query management callable interface.       *
*                                                               *
*****************************************************************
H
 *
E                   LTH     1   4  9 0 KEY     8   lengths of k/wds
E                   STA         4 33              k/wd vals passed
E                   KEL         4  9 0            keyword lengths
E                   KEN        30  1              keyword names
E                   VAL         4  9 0            value lengths
E                   VAV        81  1              value values
E                   TST        33  1              test value length
 *
I            DS
I                                    B   1   40BIN1
I                                    B   5   80BIN2
I                                    B   9  240KEL
I                                    B  25  400VAL
I/COPY BPLIB/QRPGSRC,DSQCOMMR
 *
 * receive the passed start command keyword values:
 *
C           *ENTRY    PLIST
C                     PARM           DSQCOM          comms area
C                     PARM           STA             keywords passed
 *
 * prepare keyword name lengths, names, value lengths, values
 *
C                     Z-ADD1         Y       20      initialize
C                     Z-ADD1         W       20      counters
C                     Z-ADD0         KEL             and numeric
C                     Z-ADD0         VAL             arrays
```

Figure 6-1 (Part 1 of 2). Example START Subprogram

```
 *
C           V         DOUEQ4                          look at each
C                     ADD  1         V       10       passed keyword
C           STA,V     COMP *BLANKS                   50value & process
C           *IN50     IFEQ '0'                        if not blank
 *
C                     ADD  1         X       10       keyword name
C                     MOVE LTH,V     KEL,X            lengths array
 *
C           ' '       LOKUPKEN,Y                     60 string keyword
C                     MOVE KEY,V     WORK1   8          name into
C                     MOVEAWORK1     KEN,Y              names array
 *
C                     MOVEASTA,V     TST                find keyword
C                     Z-ADD1         Z       20         value length
C           ' '       LOKUPTST,Z                     61 and move
C   61                SUB  1         Z                  to keyword
C  N61                Z-ADD33        Z                  value lengths
C                     Z-ADDZ         VAL,X              array
 *
C           ' '       LOKUPVAV,W                     62 string keyword
C                     MOVE STA,V     WORK2  33          value into
C                     MOVEAWORK2     VAV,W              values array
 *
C                     END
C                     END
 *
 * start the query interface session:
 *
C                     CALL DSQCIR
C                     PARM           DSQCOM             comms area
C                     PARM 5         BIN1               command length
C                     PARM 'START'   CHAR1   5          START
C                     PARM X         BIN2               # keywords
C                     PARM           KEL                keyword lengths
C                     PARM           KEN                keyword names
C                     PARM           VAL                value lengths
C                     PARM           VAV                values
C                     PARM DSQVCH    CHAR2   4          CHAR
 *
C                     MOVE '1'       *INLR
 *
**  start DSQ keyword name lengths and names loaded as compile time array
000000008DSQSMODE
000000007DSQSCMD
000000007DSQSRUN
000000008DSQSNAME
```

Figure 6-1 (Part 2 of 2). Example START Subprogram

SETC Subprogram

The SETC subprogram performs the SET GLOBAL variable function for a character value to be passed to the CI. The SETC subprogram handles one variable at a time. The variable name and value are passed to this program as two separate parameters. The name can be up to 10 characters long, and the value up to 20 characters long.

This subprogram calculates the necessary lengths, strings the information together, and calls the programmable interface. The variable is set as CHAR data type, and control then returns to the calling program.

Note: The SETC subprogram is not ended because it may be called a number of times in the session.

```
*****************************************************************
*                                                               *
*          SET GLOBAL COMMAND (CHARACTER VARIABLE)              *
*                CPI QUERY INTERFACE HANDLER                    *
*          ------------------------------------                 *
*                                                               *
*  1) Include member DSQCOMMR contains the communications       *
*     area to be passed to the Query Interface.                 *
*  2) This program handles the SET GLOBAL Query Interface       *
*     command for variable values to be passed to the           *
*     interface as CHAR type.
*  3) It reads the variable name and value, calculates the      *
*     length of each, and passes the information to Query       *
*     Management.                                               *
*  4) The program handles one variable at a time, the length    *
*     of the variable name can be a maximum of 10 characters    *
*     and the length of the variable value can be a maximum     *
*     of 20 characters.                                         *
*                                                               *
*****************************************************************
H
 *
E                   TNL        10  1              test name length
E                   TVL        20  1              test value length
 *
I              'SET GLOBAL'          C         CMD
 *
I            DS
I                                    B   1   40BIN1
I                                    B   5   80BIN2
I                                    B   9  120BIN3
I                                    B  13  160BIN4
I/COPY BPLIB/QRPGSRC,DSQCOMMR
 *
 * receive the passed variable name and value:
 *
C           *ENTRY    PLIST
C                     PARM           DSQCOM           comms area
C                     PARM           VARNAM 10        variable name
C                     PARM           VARVAL 20        variable value
 *
 * calculate the variable name length and variable value length:
 *
C                     MOVEAVARNAM    TNL
C                     Z-ADD1         X       20         X = last
C           ' '       LOKUPTNL,X                    60  non blank
C   60                SUB  1         X                  character
C  N60                Z-ADD10        X                  in name
```

Figure 6-2 (Part 1 of 2). Example SETC Subprogram

```
 *
C                   Z-ADD20       Y       20          if value
C           VARVAL  IFNE *BLANKS                      blank pass
C                   MOVEAVARVAL   TVL                 20 blanks
C           AGAIN   TAG
C           ' '     COMP TVL,Y                    61  Y = last
C   61              SUB  1        Y                   non blank
C   61              GOTO AGAIN                        character
C                   END                               in value
 *
 * set the global variables:
 *
C                   CALL DSQCIR
C                   PARM          DSQCOM              comms area
C                   PARM 10       BIN1                command length
C                   PARM CMD      CHAR1  10           SET GLOBAL
C                   PARM 1        BIN2                # variables
C                   PARM X        BIN3                var name length
C                   PARM          VARNAM              variable name
C                   PARM Y        BIN4                var value lngth
C                   PARM          VARVAL              variable value
C                   PARM DSQVCH   CHAR2   4           CHAR
 *
C                   RETRN
```

Figure 6-2 (Part 2 of 2). Example SETC Subprogram

SETA Subprogram

The SETA subprogram performs the SET GLOBAL variable function for a character value to be enclosed in apostrophes and then passed to the CI. This function is required when creating a query that compares a data item to a constant character value (DEPT = 'ACCT'). The variable name and value are passed to this program as two separate parameters. The name can be up to 10 characters long and the value up to 20 characters long.

This program encloses the value in apostrophes, calculates the necessary lengths, strings the information together, and calls the programmable interface. The variable is set as CHAR data type, and control then returns to the calling program.

Note: The SETA program is not ended because it may be called a number of times in the session.

```
*****************************************************************
*                                                               *
*       SET GLOBAL COMMAND (APOSTROPHE ENCLOSED CHARACTER       *
*            VARIABLE) CPI QUERY INTERFACE HANDLER              *
*       ------------------------------------------              *
*                                                               *
*  1) Include member DSQCOMMR contains the communications       *
*     area to be passed to the query management interface.      *
*  2) This program handles the SET GLOBAL interface             *
*     command for variable values to be enclosed in             *
*     apostrophes and passed to the interface as CHAR type.     *
*  3) It reads the variable name and value, calculates the      *
*     length of each, encloses the value in apostrophes,        *
*     and passes the information to query management.           *
*  4) The program handles one variable at a time, the length    *
*     of the variable name can be a maximum of 10 characters    *
*     and the length of the variable value can be a maximum     *
*     of 20 characters.                                         *
*                                                               *
*****************************************************************
H
 *
E                   TNL        10  1              test name length
E                   TVL        22  1              test value length
 *
I              'SET GLOBAL'          C         CMD
 *
I            DS
I                                    B   1   40BIN1
I                                    B   5   80BIN2
I                                    B   9  120BIN3
I                                    B  13  160BIN4
I/COPY BPLIB/QRPGSRC,DSQCOMMR
 *
 * receive the passed variable name and value:
 *
C           *ENTRY    PLIST
C                     PARM           DSQCOM           comms area
C                     PARM           VARNAM 10        variable name
C                     PARM           VARVAL 20        variable value
 *
 * calculate the variable name length and variable value length:
 *
C                     MOVEAVARNAM    TNL
C                     Z-ADD1         X       20         X = last
C           ' '       LOKUPTNL,X                    60  non blank
C   60                SUB  1         X                  character
C  N60                Z-ADD10        X                  in name
```

Figure 6-3 (Part 1 of 2). Example SETA Subprogram

```
 *
C                   MOVE ''''      TVL,1              set up first
C                   MOVEAVARVAL    TVL,2              apostrophe
C                   Z-ADD21        Y       20
C           AGAIN   TAG                                Y = last
C           ' '     COMP TVL,Y                    61   blank
C   61              SUB  1         Y                  character
C   61              GOTO AGAIN
C                   ADD  1         Y                  set up last
C                   MOVE ''''      TVL,Y              apostrophe
 *
 * set the global variables:
 *
C                   CALL DSQCIR
C                   PARM           DSQCOM             comms area
C                   PARM 10        BIN1               command length
C                   PARM CMD       CHAR1  10          SET GLOBAL
C                   PARM 1         BIN2               # variables
C                   PARM X         BIN3               var name length
C                   PARM           VARNAM             variable name
C                   PARM Y         BIN4               var value lngth
C                   PARM           TVL                variable value
C                   PARM DSQVCH    CHAR2   4          CHAR
 *
C                   RETRN
```

Figure 6-3 (Part 2 of 2). Example SETA Subprogram

SETN Subprogram

The SETN subprogram performs the SET GLOBAL variable function for a numeric value (nonbinary) to have a decimal point and trailing sign inserted and then passed to the CI. This function is required when creating a query that compares a numeric data item to a constant value (AMOUNT = 525.30-).

The variable name, the variable value, and the number of decimal positions are passed to this program as three separate parameters. The name can be up to 10 characters long, the value must be 15 numeric digits long, and the number of decimal places 2 numeric digits long. The value and decimal positions must be passed as standard numeric data (do not left-justify before passing). The subprogram inserts a decimal point if specified, adds a minus sign if the number is negative, calculates the necessary lengths, strings the information together, and calls the programmable interface. The variable is set as CHAR data type, and control returns to the calling program.

Note: The SETN subprogram is not ended because it may be called a number of times in the session.

```
*****************************************************************
*                                                               *
*      SET GLOBAL COMMAND (NUMERIC - NON BINARY INTEGER)        *
*                CPI QUERY INTERFACE HANDLER                    *
*      ------------------------------------------------         *
*                                                               *
*  1) Include member DSQCOMMR contains the communications       *
*     area to be passed to the query management interface.      *
*  2) This program handles the SET GLOBAL interface             *
*     command for variable values to be passed to the           *
*     interface as numeric data CHAR type.                      *
*  3) It reads the variable name and value, calculates the      *
*     length of each, inserts the decimal point and leading     *
*     negative sign (if required) and passes the information    *
*     to query management.                                      *
*  4) The program handles one variable at a time, the length    *
*     of the variable name can be a maximum of 10 characters    *
*     and the length of the variable value can be a maximum     *
*     of 15 numeric digits (plus sign and decimal point).       *
*                                                               *
*****************************************************************
H
 *
E                    TNL        10  1              test name length
E                    TVL        17  1              variable value
 *
I              'SET GLOBAL'           C         CMD
 *
I            DS
I                                     B   1   40BIN1
I                                     B   5   80BIN2
I                                     B   9  120BIN3
I                                     B  13  160BIN4
I/COPY BPLIB/QRPGSRC,DSQCOMMR
 *
 * receive the passed variable name and value:
 *
C           *ENTRY    PLIST
C                     PARM           DSQCOM           comms area
C                     PARM           VARNAM 10        variable name
C                     PARM           VARVAL 150       variable value
C                     PARM           VARDEC  20       decimal places
 *
 * calculate the variable name length:
 *
C                     MOVEAVARNAM    TNL
C                     Z-ADD1         X       20         X = last
C           ' '       LOKUPTNL,X                    60  non blank
C   60                SUB  1         X                  character
C  N60                Z-ADD10        X                  in name
```

Figure 6-4 (Part 1 of 2). Example SETN Subprogram

```
 *
 * set up the variable with decimal point and leading minus sign:
 *
C                   MOVE *BLANKS    TVL             Clear array
C                   MOVE VARVAL     VARCHA 15       Setup as alpha
C           VARVAL  COMP 0                      61  Negative value
C   61              MLLZO'8'        VARCHA          so strip sign
C*
C           VARDEC  IFEQ 0                          * Processing
C                   MOVEAVARCHA     TVL,3           * if value
C   61              MOVE '-'        TVL,2           * has no
C                   GOTO PASS                       * decimals
C                   END                             *
C*
C                   MOVEAVARCHA     TVL,2           * Processing
C   61              MOVE '-'        TVL,1           * if value
C                   Z-ADD16         Y       20      * has
C                   Z-ADD17         Z       20      * decimals
C           AGAIN   TAG                             *
C                   MOVE TVL,Y      TVL,Z           * Move each
C                   SUB  1          VARDEC          * array
C           VARDEC  IFNE 0                          * position
C                   SUB  1          Y               * over one
C                   SUB  1          Z               * place until
C                   GOTO AGAIN                      * decimal
C                   END                             * location
C                   MOVE '.'        TVL,Y           * is reached
C*
C           PASS    TAG
 *
 * set the Global Variables:
 *
C                   CALL DSQCIR
C                   PARM            DSQCOM          comms area
C                   PARM 10         BIN1            command length
C                   PARM CMD        CHAR1  10       SET GLOBAL
C                   PARM 1          BIN2            # variables
C                   PARM X          BIN3            var name length
C                   PARM            VARNAM          variable name
C                   PARM 17         BIN4            var value lngth
C                   PARM            TVL             variable value
C                   PARM DSQVCH     CHAR2   4       CHAR
 *
C                   RETRN
```

Figure 6-4 (Part 2 of 2). Example SETN Subprogram

RUNQ Subprogram

The RUNQ subprogram activates the RUN QUERY interface. The query name and form name are passed to the program as a string of 42 characters. The first 21 characters constitute the query name, and the last 21 characters are the form name.

The query name and form name must be left-justified. If the form is not being used, positions 22 to 42 of the string must still be passed, but as blank characters.

The RUNQ subprogram reads the passed query and form names, tests for blank forms, calculates lengths, formats the RUN QUERY command, and calls the pro-

grammable interface. After the query is run, the RUNQ subprogram returns control to its calling program.

Note: The RUNQ subprogram is not ended because it may be called a number of times in the session.

```
*****************************************************************
*                                                               *
*       RUN QUERY COMMAND CPI QUERY INTERFACE HANDLER           *
*       ----------------------------------------                *
*                                                               *
*  1) Include member DSQCOMMR contains the communications       *
*     area to be passed to the query management interface.      *
*  2) This program handles the RUN QUERY interface command.     *
*     It reads the passed query name and form information,      *
*     reformats it, then passes the information to query        *
*     management.                                               *
*                                                               *
*****************************************************************
H
 *
E                    VAL        59  1              value to pass
 *
IRUNQ        DS
I                                       1  21 QNAM
I                                      22  42 FNAM
I            DS
I                                    B   1   40BIN1
I/COPY BPLIB/QRPGSRC,DSQCOMMR
 *
 * receive the passed run query command information:
 *
C           *ENTRY    PLIST
C                     PARM           DSQCOM          comms area
C                     PARM           RUNQ            query and form
 *
 * prepare the run query command:
 *
C                     MOVEA'RUN'     VAL             Set up RUN
C                     MOVEA'QUERY'   VAL,5           QUERY and
C                     MOVEAQNAM      VAL,11          Query name
C                     Z-ADD12        X       20      Set array index
```

Figure 6-5 (Part 1 of 2). Example RUNQ Subprogram

```
 *
C           FNAM      IFNE *BLANKS                    Only if form
C           ' '       LOKUPVAL,X                    60Find next blank
C                     ADD  1          X               leave a space &
C                     MOVEA'(FORM='   VAL,X           insert (FORM=
C           ' '       LOKUPVAL,X                    60Find next blank
C                     MOVEAFNAM       VAL,X           form name
C                     END
 *
C           ' '       LOKUPVAL,X                    61Find last blank
C   61                SUB  1          X               Last non blank
C  N61                Z-ADD59         X               No blanks left
 *
 * process the run query command:
 *
C                     CALL DSQCIR
C                     PARM            DSQCOM          comms area
C                     PARM X          BIN1            command length
C                     PARM            VAL             command
 *
C                     RETRN
```

Figure 6-5 (Part 2 of 2). Example RUNQ Subprogram

RUNP Subprogram

The RUNP subprogram activates the RUN PROC interface. The procedure name is passed to the program as a string of 33 characters. The procedure name must be left-justified. The RUNP subprogram reads the passed procedure name, calculates lengths, formats the RUN PROC command, and calls the programmable interface. After the procedure is run, the RUNP subprogram returns control to the calling program.

Note: The RUNP subprogram is not ended because it may be called a number of times in the session.

```
*************************************************************
*                                                           *
*        RUN PROC COMMAND CPI QUERY INTERFACE HANDLER       *
*        ----------------------------------------           *
*                                                           *
*  1) Include member DSQCOMMR contains the communications   *
*     area to be passed to the query management interface.  *
*  2) This program handles the RUN PROC interface command.  *
*     It reads the passed procedure name and form information, *
*     reformats it, calculates the length, then passes the  *
*     information to query management.                      *
*                                                           *
*************************************************************
H
 *
E                    VAL        42  1              value to pass
 *
I            DS
I                                        B   1   40BIN1
I/COPY BPLIB/QRPGSRC,DSQCOMMR
 *
 * receive the passed run procedure command information:
 *
C           *ENTRY    PLIST
C                     PARM           DSQCOM           comms area
C                     PARM           RUNP   33        procedure name
 *
 * prepare the run procedure command:
 *
C                     MOVEA'RUN'     VAL              Set up RUN
C                     MOVEA'PROC'    VAL,5            PROC and
C                     MOVEARUNP      VAL,10           Procedure name
 *
C                     Z-ADD11        X       20       Set array index
C           ' '       LOKUPVAL,X                     60Find last blank
C   60                SUB  1         X                Last non blank
C  N60                Z-ADD42        X                No blanks left
 *
 * process the run procedure command:
 *
C                     CALL DSQCIR
C                     PARM           DSQCOM           comms area
C                     PARM X         BIN1             command length
C                     PARM           VAL              command
 *
C                     RETRN
```

Figure 6-6. Example RUNP Subprogram

EXIT Subprogram

The EXIT subprogram requires no additional parameters to the DSQCOM communications area. When called, it ends the query interface, ends itself, then returns to the program that called it.

```
*****************************************************************
*                                                               *
*          EXIT COMMAND CPI QUERY INTERFACE HANDLER             *
*          ---------------------------------------              *
*                                                               *
*  1) Include member DSQCOMMR contains the communications       *
*     area to be passed to the query management interface.      *
*  2) This program handles the EXIT interface command.          *
*     It passes to query management the command length          *
*     and command.                                              *
*                                                               *
*****************************************************************
H
 *
I           DS
I                                    B   1   40BIN1
I/COPY BPLIB/QRPGSRC,DSQCOMMR
 *
 * receive the passed communications area:
 *
C           *ENTRY    PLIST
C                     PARM           DSQCOM          comms area
 *
 * call the interface and end the session:
 *
C                     CALL DSQCIR
C                     PARM           DSQCOM          comms area
C                     PARM 4         BIN1            command length
C                     PARM 'EXIT'    DATA    4       command
 *
C                     MOVE '1'       *INLR           end program
```

Figure 6-7. Example EXIT Subprogram

Chapter 7. Query Management/400 Considerations

This chapter describes how query management interacts with other system functions and suggests some techniques to help you work with the product.

Override Considerations

You can use overrides specified by the Override Database File (OVRDBF) command to redirect a reference to a different file. The following sections discuss some considerations of how overrides are handled when query management processes different types of files.

Tables and Views

Override considerations for tables and views referred to in the Structured Query Language (SQL) statement during a RUN QUERY command are the same as those used in SQL. The following parameters are processed when you specify an override:

- TOFILE
- MBR
- SEQONLY
- LVLCHK
- INHWRT
- WAITRCD

SQL can process a member other than the first member in a query management query by specifying the desired member with the MBR keyword on the OVRDBF command prior to the RUN QUERY.

The query fails if it selects a member from a file that has an override of MBR(*ALL). For more information about using overrides in an SQL statement, see the *SQL/400* Programmer's Guide*.

Tables Referred to on the ERASE TABLE Command

Overrides are ignored on the ERASE TABLE command.

Tables and Views Referred to on the SAVE DATA AS Command

You can direct query management to process a file other than the table or view specified on the command by using the OVRDBF CL command. Overrides are ignored if the file specified on the TOFILE keyword of the OVRDBF command does not exist.

You can save data to a member other than the first member of the file by specifying the desired member on the MBR keyword of the OVRDBF command before issuing the SAVE DATA AS command.

If you issue a SAVE DATA AS command to a file that has an override of MBR(*ALL), the command fails.

The following parameters are processed on the SAVE DATA AS command if an override is specified:

- TOFILE
- MBR

© Copyright IBM Corp. 1991

- SEQONLY
- LVLCHK
- INHWRT
- WAITRCD

IMPORT and EXPORT Source Files

Overrides on the source files referred to by an IMPORT or EXPORT command are processed.

The following parameters are processed on an IMPORT or EXPORT command if an override is specified on the source file:

- TOFILE
- MBR

An IMPORT from a source file that has an override of MBR(*ALL) is allowed. The IMPORT processes each record of each member. The members are read in the order in which they are created. The IMPORT completion message lists only the name of the first member processed during the import.

An EXPORT to a source file fails if it has an override of MBR(*ALL).

If an EXPORT refers to a file name that has an override, and the file to which the override is directed does not exist, query management creates the file. The file is named the same as the name specified on the TOFILE keyword on the OVRDBF command. For example, the following CL command was run prior to calling query management:

```
OVRDBF FILE(XYZ) TOFILE(MYLIB/MYFILE)
```

The file MYFILE in MYLIB does not exist. The following query management command results in the creation of a source physical file named MYFILE created in the library MYLIB:.

```
EXPORT QUERY MYQUERY TO XYZ
```

A member name specified with an override takes precedence over a member name specified on the command. For example, the following CL command was run prior to calling query management:

```
OVRDBF FILE(XYZ) TOFILE(MYLIB/MYFILE) MBR(MEMBER2)
```

Issuing the following query management command results in the source for the query MYQUERY being put in member MEMBER2 of file MYFILE in library MYLIB:

```
EXPORT QUERY MYQUERY TO XYZ(MEMBER1)
```

Query Procedures

Overrides are not processed on files referred to as query procedures on RUN PROC, ERASE PROC, PRINT PROC, IMPORT PROC, or EXPORT PROC commands. Overrides of other files processed by query commands in procedures being run with the RUN PROC are processed. Overrides of the source files specified on IMPORT PROC and EXPORT PROC commands are processed.

Query management cannot process overrides on any files while processing a procedure if those files have the same name as the procedure being run. This rule applies to:

- The source file on an IMPORT PROC or EXPORT PROC command if the source file has the same name as the procedure file.
- The source files on any IMPORT or EXPORT command that is run in a procedure with the same name, or to any procedure nested in a procedure of the same name.
- The file referred to on a SAVE DATA AS command if the command is run in a procedure with the same name, or to any procedure nested in a procedure of the same name.
- A file referred to by the SQL statement in a query if the RUN query is run in a procedure with the same name, or to any procedure nested in a procedure of the same name.

The following is an example of how overrides on PROC statements are processed.

The following CL commands were run prior to calling query management:

```
OVRDBF FILE(XYZ) TOFILE(MYLIB/MYFILE)
OVRDBF FILE(ABC) TOFILE(MYLIB/MYFILE)
```

The following commands result in processing the procedure ABC in MYLIB even though the above CL command overrides file ABC to the file MYFILE.

```
RUN PROC MYLIB/ABC
PRINT PROC MYLIB/ABC
ERASE PROC MYLIB/ABC
```

The following IMPORT command imports the procedure ABC in MYLIB from the source file MYFILE in MYLIB because the override was not processed for the query procedure, but it was for the source file.

```
IMPORT PROC MYLIB/ABC FROM MYLIB/XYZ
```

The following EXPORT command imports the procedure ABC in MYLIB to the source file ABC in MYLIB because the override was not processed for the query procedure. Because the file that was specified on the source file was the same as the query procedure, overrides were not processed.

```
EXPORT PROC MYLIB/ABC(MEMBER1) TO MYLIB/ABC(MEMBER2)
```

The query called QUERY1 contains the SQL statement:

```
SELECT * FROM MYLIB/ABC A1, MYLIB/XYZ A2 WHERE A1.X=B1.X
```

and the file MYLIB/ABC contains the command `RUN QUERY QUERY1`. The following RUN PROC command runs the query procedure ABC in MYLIB. When the query is run, the data is selected from the files ABC in MYLIB and MYFILE in MYLIB. The overrides for file ABC are not processed during the processing of the query procedure ABC.

```
RUN PROC MYLIB/ABC
```

For more information on overrides, see the *Database Guide* and the *Data Management Guide*.

Miscellaneous Tips and Techniques

This section describes special applications for query management functions and suggests other ways to use other products and system functions to make working with query management easier.

Printing a Query Management Object

When printing any query management object, create a source file member and edit it. Use the following instructions to complete the printing process using the member created:

- Print a query (QMQRY) object by putting the following statement in the source file member created at the start of the session:

  ```
  'PRINT QUERY libname/queryname (PRINTER= printername'
  ```

  Then run the Start Query Management Procedure (STRQMPRC) CL command against the procedure member to print the contents of the query management query.

- Print a form (QMFORM) object by putting the following statement in the source file member created at the start of the session:

  ```
  'PRINT FORM libname/formname (PRINTER= printername'
  ```

  Then run the Start Query Management Procedure (STRQMPRC) CL command to print the contents of the query management form.

- Print a procedure (QMPROC) object by putting the following statement in the source file member created at the start of the session:

  ```
  'PRINT PROC libname/procedurename (PRINTER= printername'
  ```

  Then run the Start Query Management Procedure (STRQMPRC) CL command against the procedure member to print the contents of the query management procedure.

Changing STRQMQRY Defaults for QRYDFN Use

If you prefer to use the WRKQRY command to develop and maintain the query and form information used by query management, it may be convenient to use a copy of the STRQMQRY command that has been changed to use defaults that let you run a QRYDFN object just by naming it. For example, you could create the command STRQRYDFN in the current library for your job by entering the following commands:

```
CRTDUPOBJ OBJ(STRQMQRY) FROMLIB(QSYS) OBJTYPE(*CMD) TOLIB(*CURLIB)
          NEWOBJ(STRQRYDFN)

CHGCMDDFT CMD(*CURLIB/STRQRYDFN) NEWDFT('QMFORM(*QMQRY)')

CHGCMDDFT CMD(*CURLIB/STRQRYDFN) NEWDFT('ALWQRYDFN(*YES)')
```

To run QRY1 in the current library (*CURLIB), type `STRQRYDFN *CURLIB/QRY1`, or just `STRQRYDFN QRY1`.

Displaying Information about Using QRYDFN Objects

To read about the system actions taken when there are problems deriving information from QRYDFN objects, display the query management conversion messages using the following command string:

```
DSPMSGD RANGE(QWM2301 QWM2399) DETAIL(*BASIC)
```

The messages that are displayed may contain warnings about unexpected consequences of the system action taken, or suggest ways of avoiding or minimizing problems of the sort diagnosed by the message.

Defining Queries with Global Variables Using Query/400

Query/400 supports a data or text merge function that involves using dependent QRYDFN objects that cannot be run the same as other QRYDFN objects. These objects are different because record selection tests contain variables (dependent values). You can use query management to run these queries if you assign the correct values to these variables.

When information for a query is derived from a dependent QRYDFN object, dependent values are converted to global variables: `:T01.cusnam` becomes `&T01_CUSNAM`, for example. (You are prompted for a value for `T01_CUSNAM` if you did not specify one on the SETVAR parameter when using the STRQMQRY command to run the query.)

Use the SETVAR parameter to insert the value you want into the WHERE clause of the derived SELECT statement. You could, for example, specify the value '"Smith" and addr like "%Apt%"' for the `T01_CUSNAM` variable.

You can create a CL program and command to improve the prompting, provide possible choices, and restrict or validate the values entered.

Using RUNQRY to Process Data

The Systems Application Architecture (SAA) database does not support some querying functions available through Query/400 (such as alternative collating sequences and unmatched records joining). On the other hand, Query/400 lacks many of the formatting functions available through query management and does not produce parallel printed reports. You may be able to take advantage of the strengths of both products by defining two QRYDFN objects for report generation. For example, create the following objects for report generation:

1. Create the first object to be run by Query/400 to collect the data using one or more non-SAA querying functions, and save it in a database file.
2. Create the second object to be run by query management to produce a printed or displayed report from the saved data, or use it as the base from which to create QMQRY and QMFORM objects to run for this purpose.
3. Store the intermediate file in QTEMP for convenience and write a CL program to run the paired queries.

Using Query/400 to Create a QMFORM for an Existing QMQRY

You can use Query/400 to define form information based on the defaults if running an existing QMQRY object with the system default (*SYSDFT) form does not produce the formatting results you want. The following steps show the procedure for defining form information based on system defaults:

1. Run the QMQRY object and save the data in a table.
2. Use WRKQRY option 1 (Create) to define a QRYDFN object.
   a. Specify the table in which the data was saved as the file selection.
   b. Consider using the following functions tolerated but not supported by Query/400:
      - &field insertion variables in page text
      - &field insertion variables ended with an underscore character
      - Variables other than break field insertion variables in break or final text
      - &column# insertion variables ended with any nondigit character
   c. Save your definition work as a QRYDFN object.
3. Request query management use the form information from the new QRYDFN object when running the original query, or retrieve the form information and use it to create a QMFORM to use with the QMQRY object.

Displaying Data from a Single Oversized Record

If you have a QRYDFN object that produces a single-record, multiple-column report that is too wide to see in column-headed format, you can create a form that lets you see the whole report in captioned format. Create the new form using the following steps:

1. Use the WRKQRY command to change the QRYDFN object:
   a. Specify 0 length for all report columns.
   b. Define page heading text with appropriately arranged captions and insert variables. Up to 3 lines are available.
   c. Use page footing text as desired.
2. Retrieve the form source, and make any desired adjustments (for example, additional page heading or footing text lines, left alignment, or spacing).
3. Create the query management form object from the adjusted source.
4. Run the query using the form created to show the complete report.

The following is an example of a displayed single-record report in captioned form:

```
                              Display Report
Query . . . . .:   USRLIB/QI                Width . . .:      71
Form  . . . . .:   USRLIB/QI                Column  . .:       1
Control  . . . .
Line   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7.
000001 Attn (tele) . . . :  Howard Jones (218-485-0162)
000002 Account name  . . :  International Milling Company
000003 Address . . . . . :  4126 Kettering Memorial Parkway
000004 City, state . . . :  Fort Wayne, In.
000005 Zip . . . . . . . :  46815
000006
000007 Invoice # . . . . :  B12358-9
000008 Date shipped  . . :  03/27/90
000009 Hauler  . . . . . :  Dave (3-7809)
000010
000011
******  * * * * *  E N D  O F  D A T A  * * * * *



                                                                      Bottom
   F3=Exit      F12=Cancel      F19=Left      F20=Right      F21=Split
```

Using Query Management or CL Commands in PDM Options

You may find it convenient to use programming development manager (PDM) to work on lists of QMQRY, QMFORM, or QRYDFN objects. Define options that run CL commands (for example, STRQMQRY or ANZQRY) after substituting library and object names selected when you type the option code beside a list entry. For example:

- Define option SQ to be:

  ```
  STRQMQRY &L/&N QMFORM(*QMQRY) ALWQRYDFN(*YES).
  ```

  Then use SQ to display a report using query and form information derived from a selected QRYDFN object without having to remember to override the defaults QMFORM(*SYSDFT) and ALWQRYDFN(*NO).

- Define option Z to be:

  ```
  ANZQRY   &L/&N 99.
  ```

  Then type Z beside all the names in a list of QRYDFN objects to get completion messages. Check these messages to see which QRYDFN objects may need adjustment for satisfactory query management use.

You could define other options to run user-developed CL commands or to call user-developed CL programs that act on query management objects.

Creating a CL Program for Permanent Conversion of a QRYDFN Object

You may want to create a CL program to convert QRYDFN objects to query management objects if this operation is performed frequently. Define parameters for the program to specify object names and other variables.

Figure 7-1 is an example of the source for a program that converts QRYDFN information into query management objects. This program assumes *LIBL should be searched for the QRYDFN object, and that query management objects should be placed in *CURLIB. It shows the report produced from the QRYDFN object by Query/400, then the report produced from converted objects by query management. If the request is not canceled, the program copies the converted objects from QTEMP to *CURLIB.

```
Columns . . . :    1  68           Edit                          USRLIB/QCLSRC
SEU==>                                                              MIGRATE
FMT **  ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+...
       *************** Beginning of data ************************************
0001.00 PGM PARM(&OBJ)
0002.00 DCL VAR(&OBJ) TYPE(*CHAR) LEN(10)
0003.00 CRTSRCPF QTEMP/QQMQRYSRC 91
0004.00 MONMSG MSGID(CPF7302)
0005.00 CRTSRCPF QTEMP/QQMFORMSRC 162
0006.00 MONMSG MSGID(CPF7302)
0007.00 RUNQRY *LIBL/&OBJ OUTPUT(*)
0008.00 RTVQMQRY *LIBL/&OBJ QTEMP/QQMQRYSRC &OBJ ALWQRYDFN(*YES)
0009.00 MONMSG MSGID(QWM2701)
0010.00 CRTQMQRY QTEMP/&OBJ QTEMP/QQMQRYSRC &OBJ
0011.00 MONMSG MSGID(QWM2701)
0012.00 RTVQMFORM *LIBL/&OBJ QTEMP/QQMFORMSRC &OBJ ALWQRYDFN(*YES)
0013.00 MONMSG MSGID(QWM2701)
0014.00 CRTQMFORM QTEMP/&OBJ QTEMP/QQMFORMSRC &OBJ
0015.00 MONMSG MSGID(QWM2701)
0016.00 STRQMQRY QMQRY(&OBJ) QMFORM(*QMQRY)
0017.00 MONMSG MSGID(QWM2701 QWM2703) EXEC(RETURN)
0018.00 DLTQMQRY QMQRY(*CURLIB/&OBJ)
0015.00 MONMSG MSGID(CPF2105)
0019.00 CRTDUPOBJ OBJ(&OBJ) FROMLIB(QTEMP) OBJTYPE(*QMQRY) TOLIB(*CURLIB)
0020.00 DLTQMFORM QMFORM(*CURLIB/&OBJ)
0015.00 MONMSG MSGID(CPF2105)
0021.00 CRTDUPOBJ OBJ(&OBJ) FROMLIB(QTEMP) OBJTYPE(*QMFORM) TOLIB(*CURLIB)
0022.00 ENDPGM
       ****************** End of data ************************************
```

Figure 7-1. CL Source for Permanent Conversion Program

Querying for Field Values

You can create a generic query to display an ordered list of the values used in a field of a particular file. The library, file, and field names can be global variables to be set when the query is run. The following is an example of a SELECT statement that creates a generic query:

```
SELECT DISTINCT &FIELD FROM &LIBRARY/&FILE ORDER BY 1
```

Run the QMQRY object created from this statement to get the list specified in the following SETVAR parameter (this assumes you name the created QMQRY object qryvalues, and you have a database file named staff, in the library testdata, with a field named dept):

```
STRQMQRY qryvalues SETVAR((FIELD dept) (LIBRARY testdata) (FILE staff))
```

Get a subset of the values by adding a record selection test when you set the variables:

```
STRQMQRY qryvalues SETVAR((FIELD dept) (LIBRARY testdata)
        (FILE 'staff where dept > 50'))
```

View all the columns (with no records duplicated) by using an asterisk (*) when you set the variables:

```
STRQMQRY qryvalues SETVAR((FIELD '*' ) (LIBRARY testdata) (FILE staff))
```

Make it easier to specify values for the global variables by writing simple CL prompting programs and commands. Set it up so that you can get the list you want by typing:

```
q testdata/staff dept
```

This is a helpful command to use if you are using source entry utility (SEU) to edit query source and want to see which values could be used in tests. SEU permits you to request a window for entering system or user-defined commands.

Passing Variable Values to a Query

Global variable names are not necessarily meaningful, and a user being prompted for a value may not know what to type. You can write CL programs and commands to provide meaningful prompting and validation of typed values. Figure 7-2 and Figure 7-3 show source statements that you can use to create a program and to create a command for a query that shows an ordered list of values for a specified field in the first member of a specified file. Create a CL program from the program source, then specify it as the command processing program when the command is created from the command source.

```
0001.00 PGM PARM(&FILE &FIELD)
0002.00 DCL VAR(&FILE) TYPE(*CHAR) LEN(20)
0003.00 DCL VAR(&LIB) TYPE(*CHAR) LEN(10)
0004.00 DCL VAR(&TABLE) TYPE(*CHAR) LEN(10)
0005.00 DCL VAR(&FIELD) TYPE(*CHAR) LEN(10)
0006.00 CHGVAR &LIB %SUBSTRING(&FILE 11 10)
0007.00 CHGVAR &TABLE %SUBSTRING(&FILE 1 10)
0008.00 STRQMQRY CAJR30/QRYVALUES SETVAR((LIBRARY &LIB)(FILE &TABLE)(FIELD &FIELD))
0009.00 ENDPGM
```

Figure 7-2. CL Source for Global Variable Prompting Program

```
                                                                  ******
0001.00  Q:          CMD        PROMPT('Query Column Values(Q)')
0002.00              PARM       KWD(FILE) TYPE(Q1) MIN(1) MAX(1) +
0003.00                           PROMPT('Table name')
0004.00              PARM       KWD(FIELD) TYPE(*CHAR) LEN(10) +
0005.00                           PROMPT('Column name')
0006.00  Q1:         QUAL       TYPE(*NAME) LEN(10) MIN(1)
0007.00              QUAL       TYPE(*NAME) LEN(10) +
0008.00                         DFT(*LIBL) +
0009.00                         SPCVAL(*LIBL (*CURLIB *CURLIB)) +
0010.00                         PROMPT('Collection')
```

Figure 7-3. CL Source for Global Variable Prompting Command

The following figure is a sample of a user-developed prompting display needed for passing variable values:

```
                        Query Column Values (Q)
Type choices, press Enter.
Table name . . . . . .                    Name
  Collection . . . . . .      *LIBL       Name, *LIBL, *CURLIB
Column name  . . . . . .

                                                                  Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Defining a Column with No Column Heading

To prevent a column from having a heading, specify *NONE in the leftmost position of the top heading line shown when working on the definition under interactive data definition utility (IDDU) or the WRKQRY command. You can also specify a single underline character as the column heading in form source, but this causes the report to have at least two lines reserved for column headings. In either case, the column still has separators unless you eliminate column heading separators from the whole report. To eliminate column heading separators, retrieve and edit the appropriate field in the form source, and create the form again.

Using Query Management to Format an ISQL-Developed Query

You can use Structured Query Language (SQL) interactively to develop a query that uses any of the supported SQL database functions. By using the Interactive Structured Query Language (ISQL) product, which exists on top of SQL, you can run SQL commands interactively. These functions include subqueries, scalar functions, GROUP BY statements, and others not available through the Query/400 prompted interface. The following steps describe how to get an ISQL-developed SELECT statement into a QMQRY object, and how to use Query/400 to define information that query management can use to format the displayed or printed output from running this QMQRY object:

1. Specify the Start Structured Query Language (STRSQL) command.
   a. Use ISQL to develop the query you want.
   b. Create a database (collection) to receive the output of this query, or use a previously created collection.
   c. Change the output device for the session to be a database file in the previously created collection. Use a name (for example, QRYPURPOSE) that describes the purpose of the query.
   d. Run the query again to create the file (table) QRYPURPOSE.

e. Save the query session as member QRYPURPOSE. Remember the file and library names you specify so you can edit the session later for use as query management query source.

f. Exit the ISQL session.

2. Specify the Start System Entry Utility (STRSEU) command to edit the saved session.

   a. Remove all lines other than those containing the SELECT statement that defines your query.

   b. Add any comments that are needed.

   c. Replace appropriate elements in the SELECT statement with global variables.

   d. Exit SEU, saving the changed member.

3. Use the Create Query Management Query (CRTQMQRY) command to create the QMQRY object QRYPURPOSE from member QRYPURPOSE.

4. Specify the WRKQRY command and choose the *Create* option.

   a. Select file QRYPURPOSE created in the ISQL session.

   b. Specify report column formatting overrides. The defaults shown are the same as ISQL used to show the report, but not the same as query management would use if you ran QRYPURPOSE with the *SYSDFT form. If you want to use the defaults shown, make Query/400 treat them as overrides so that they will be saved with the QRYDFN object. (Any change to column headings causes the default to be considered overridden, even if you put back the original default value. The same is true for length, decimal positions, and numeric editing.)

   c. Use edit codes J (numeric values), J$ (currency values), and M (numeric identifiers) to define editing you can convert to SAA edit codes incorporating any decimal position overrides you specify.

   d. Add any extra formatting you think will improve your report. You can, for example, define final text and overall summaries to appear below columns defined as aggregating scalar functions in the SELECT statement saved in QRYPURPOSE.

   e. Save your formatting choices as QRYDFN object QRYPURPOSE.

5. Optionally retrieve form source from QRYDFN QRYPURPOSE and use it to create QMFORM QRYPURPOSE.

6. Use the STRQMQRY command to run query QRYPURPOSE. Use QMFORM(*QMQRY) and, if you did not create a QMFORM from QRYDFN QRYPURPOSE, specify ALWQRYDFN(*YES) to allow use of formatting information derived from the QRYDFN object.

Figure 7-4 shows an ISQL-developed query.

```
Query . . . . . :   MAXSALARY
  Library . . . . :     USRLIB
Text . . . . . :
SEQNBR *...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
000001 select dept,max(salary) from testdata/staff group by dept
                                        * * * * *   END OF SOURCE   * * * * *
```

Figure 7-4. Sample Printed ISQL-Developed QMQRY Object

The following display shows the formatted report produced from the query in Figure 7-4. This report was created by query management use of form information derived from a QRYDFN object created from the ISQL output file definition.

```
                                Display Report
Query . . . . .:   USRLIB/MAXSALARY          Width . . .:      36
Form  . . . . .:   USRLIB/MAXSALARY          Column  . .:       1
Control  . . . .
Line    |...+....1....+....2....+....3....+....4....+....5....+....6...
                                    Maximum
                          Dept       Salary
                     ------------  ------------
000001
000002
000003                       10     $22,959.20
000004                       15     $20,659.80
000005                       20     $18,357.50
000006                       38     $18,006.00
000007                       42     $18,352.80
000008                       51     $21,150.00
000009                       66     $21,000.00
000010                       84     $19,818.00
000011                             ============
000012       Overall maximum:
000013                              $22,959.20
                                                                      More...
   F3=Exit      F12=Cancel      F19=Left      F20=Right      F21=Split
```

Using Text Insertion Variables To Stack Captions on Final Summaries

The following figure shows a final level summary report with the summary values stacked and captioned. It demonstrates that the final summary values can be kept on one page and shown in any desired order instead of being spread over multiple displays or printer files in separate columns.

```
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
        (Salary analysis for 35 employees in department 10)

Minimum . . . . . : $10,506
Maximum . . . . . : $22,959
Average . . . . . : $16,676
Total . . . . . . : $583,647

/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
```

Figure 7-5. Final Level Summary Values as Cover Page and Heading Text Insertions

The report was produced by query management from a single QRYDFN object with the following characteristics:

- Summary-only output form

- No break fields selected
- Final level summaries not suppressed
- Summaries selected
- Length 0 specified for all summary fields to be used as inserts
- Cover page and page heading text containing the desired captions and headings with summary value insert placement indicated by &# references to output column numbers

Note: `All field widths set to 0` is diagnosed, and no report is produced, when an attempt is made to run the QRYDFN using Query/400.

Text insertion can be used in combination with tabular layout. To produce the example that follows, record selection tests were defined to limit the output to a particular customer (specified at run time because of the use of a global variable), and MIN and COUNT functions were defined to supply the inserts for the cover page text used to define the label and for the final text.

```
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
                    Orders Inquiry

Herman B. Wannamaker
3124 Melrose Ave - Apt 35
Gooseneck, NY  55945


          TOTAL          AVG          MAX          MIN
         Charges        Price        Price        Price
  --------------  -----------  -----------  -----------
       $3,859.72       $79.54     $1024.89        $3.50

Number of transactions: 35
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
```

Figure 7-6. Final Level Text Insertions with Summary Table

Converting a Multiple-Level Summary-Only QRYDFN

The following figures show summary-only reports that have both break summaries and final summaries. The query management report was produced with the lowest level summaries coming from SQL column functions in a query-defining QRYDFN object, and the other summaries coming from column usages in a second, form-defining QRYDFN object. Here is how it was done. A copy of the original QRYDFN was changed to collapse all levels and suppress final summaries, then saved for use as the query. Another copy was changed to suppress summaries at the lowest level, then saved for use as the form. Then STRQMQRY was applied to the query and form with the use of QRYDFN objects allowed. Note that the overall average is really an average of averages.

```
                                Salary Report Summary, 1989
                              AVERAGE      MINIMUM      MAXIMUM
Job        Years              Salary       Salary       Salary
-----  -------  -----------  -----------  -----------
CLERK          0   $12,655.98   $11,508.60   $13,504.60
               1   $10,988.00   $10,988.00   $10,988.00
               3   $12,689.78   $12,009.75   $13,369.80
               4   $12,258.50   $12,258.50   $12,258.50
               5   $12,769.35   $12,508.20   $13,030.50
               6   $12,482.95   $10,505.90   $14,460.00
               8   $14,252.75   $14,252.75   $14,252.75
                   -----------  -----------  -----------
Overall CLERK:
                   $12,585.33   $10,505.90   $14,460.00

MANAGER        5   $18,383.50   $17,506.75   $19,260.25
               6   $21,150.00   $21,150.00   $21,150.00
               7   $19,889.83   $18,352.80   $22,959.20
               9   $18,555.50   $18,555.50   $18,555.50
              10   $20,162.60   $19,818.00   $20,659.80
              12   $21,234.00   $21,234.00   $21,234.00
                   -----------  -----------  -----------
  Overall MANAGER:
                   $19,895.91   $17,506.75   $22,959.20

SALES          0   $16,808.30   $16,808.30   $16,808.30
               4   $16,858.20   $16,858.20   $16,858.20
               5   $15,454.50   $15,454.50   $15,454.50
               6   $18,488.08   $18,001.75   $19,456.50
               7   $17,333.78   $16,502.83   $17,844.00
               8   $18,171.25   $18,171.25   $18,171.25
               9   $18,674.50   $18,674.50   $18,674.50
              13   $21,000.00   $21,000.00   $21,000.00
                   -----------  -----------  -----------
Overall SALES:
                   $17,848.58   $15,454.50   $21,000.00

                   ===========  ===========  ===========
      Overall:
                   $16,679.11   $10,505.90   $22,959.20
06/18/90 09:50:21                                        1
```

Figure 7-7. Form Usages Applied to SQL Column Functions

```
06/21/90  14:23:04  Salary Report Summary, 1989  Page    1
Job    Years       Salary
CLERK      0
           AVG $12,655.98
           MIN $11,508.60
           MAX $13,504.60

CLERK      1
           AVG $10,988.00
           MIN $10,988.00
           MAX $10,988.00

CLERK      3
           AVG $12,689.78
           MIN $12,009.75
           MAX $13,369.80

CLERK      4
           AVG $12,258.50
           MIN $12,258.50
           MAX $12,258.50

CLERK      5
           AVG $12,769.35
           MIN $12,508.20
           MAX $13,030.50

CLERK      6
           AVG $12,482.95
           MIN $10,505.90
           MAX $14,460.00

CLERK      8
           AVG $14,252.75
           MIN $14,252.75
           MAX $14,252.75

CLERK
           Overall CLERK:
           AVG $12,612.61
           MIN $10,505.90
           MAX $14,460.00

MANAGER    5
           AVG $18,383.50
           MIN $17,506.75
           MAX $19,260.25

MANAGER    6
           AVG $21,150.00
           MIN $21,150.00
           MAX $21,150.00
```

Figure 7-8 (Part 1 of 3). Report with Multiple Break Levels - Query/400

```
06/21/90  14:23:04  Salary Report Summary, 1989  Page    2
Job     Years       Salary
MANAGER    7
           AVG $19,889.83
           MIN $18,352.80
           MAX $22,959.20
MANAGER    9
           AVG $18,555.50
           MIN $18,555.50
           MAX $18,555.50
MANAGER   10
           AVG $20,162.60
           MIN $19,818.00
           MAX $20,659.80

MANAGER   12
           AVG $21,234.00
           MIN $21,234.00
           MAX $21,234.00

MANAGER
           Overall MANAGER:
           AVG $19,805.80
           MIN $17,506.75
           MAX $22,959.20

SALES      0
           AVG $16,808.30
           MIN $16,808.30
           MAX $16,808.30

SALES      4
           AVG $16,858.20
           MIN $16,858.20
           MAX $16,858.20

SALES      5
           AVG $15,454.50
           MIN $15,454.50
           MAX $15,454.50

SALES      6
           AVG $18,488.08
           MIN $18,001.75
           MAX $19,456.50

SALES      7
           AVG $17,333.78
           MIN $16,502.83
           MAX $17,844.00
```

Figure 7-8 (Part 2 of 3). Report with Multiple Break Levels - Query/400

```
06/21/90  14:23:04  Salary Report Summary, 1989  Page    3
Job    Years       Salary
SALES      8
           AVG $18,171.25
           MIN $18,171.25
           MAX $18,171.25

SALES      9
           AVG $18,674.50
           MIN $18,674.50
           MAX $18,674.50

SALES     13
           AVG $21,000.00
           MIN $21,000.00
           MAX $21,000.00

SALES
           Overall SALES:
           AVG $17,869.36
           MIN $15,454.50
           MAX $21,000.00

           Overall:
           AVG $16,675.64
           MIN $10,505.90
           MAX $22,959.20
* * *  E N D  O F  R E P O R T  * * *
```

Figure 7-8 (Part 3 of 3). Report with Multiple Break Levels - Query/400

Sorting and Subsetting Break Level Summary Groups

If you have a QRYDFN or QMQRY object that produces a break level summary report (the SQL statement contains SQL column functions and a GROUP BY clause), you can create source for a QMQRY object that uses column function values to exclude unwanted groups and order the rest by editing HAVING and ORDER BY clauses in the retrieved source.

Using Information from Query/400 QRYDFN Objects

Query management is able to derive information for running queries and formatting reports from QRYDFN objects created by Query/400. Conversion to query management query (QMQRY) and form (QMFORM) objects is not required. Refer to the DSQSCNVT parameter of the CPI START command for information about how to take advantage of this capability from a user-written program. Refer to the ALWQRYDFN parameter on the following CL commands for information about how to take advantage of this capability interactively or from a CL program:

- STRQMPRC - run a procedure (a stored sequence of CPI commands)
- STRQMQRY - run a query and either save the data or format a report
- RTVQMQRY - retrieve editable query management query source
- RTVQMFORM - retrieve editable query management form source

Some of the functions that can be specified and saved in a QRYDFN object cannot be transformed into query-management-supported SAA functions, and some cannot be precisely transformed. Except for the case where a SELECT statement grows beyond 32KB in length and it is not possible to use the derived query information,

query management uses the derived information and issues no warnings about the actions taken (truncation, and so on) when a transformation problem is encountered. The ANZQRY command provides analysis in the form of messages and online help information that suggest ways of dealing with transformation problems.

Different, and possibly unacceptable, output can be produced from derived information even when there are no transformation problems. Query management may provide different defaults for functions that cannot be specified, or use successfully transformed choices differently. The *Query Management/400 Reference* manual has detailed information about the differences to expect when comparing query management output with that from Query/400, as well as detailed suggestions about what to do to get the best results from the use of information saved in a QRYDFN object.

Because the information saved in a QRYDFN object does not provide complete access to a query management function, and because it is less efficient to derive information from QRYDFN objects than from query management objects, many users will want to convert QRYDFN objects to the corresponding QMQRY and QMFORM objects. The *Query Management/400 Reference* manual explains how to retrieve (export) information from QRYDFN objects and use it to create (import) query management objects, and how to add function not available through the Query/400 prompted interface for defining QRYDFN objects.

Using the STRQMQRY Command Instead of the RUNQRY Command

Both the Start Query Management Query (STRQMQRY) and the Run Query (RUNQRY) commands can be used to produce a formatted report or database file output according to specifications in a previously created QRYDFN object. The following examples demonstrate the minimum parameter requirements for each command when the object to be run is a QRYDFN object.

```
RUNQRY mylib/myqrydfn
STRQMQRY mylib/myqrydfn  QMFORM(*QMQRY) ALWQRYDFN(*YES)
```

Some reasons for using the STRQMQRY command:

- The appearance of a report can be improved if STRQMQRY is used. Refer to figures contrasting Query/400 and query management output for a QRYDFN defined to produce summary output for a single level of break values.
- STRQMQRY also provides less restrictive use of QRYDFN information. Unlike RUNQRY, STRQMQRY can complete successfully using:
  - A QRYDFN object saved with errors
  - A dependent query in batch mode
  - Form and query information from separate objects
  - Form information derived from a QRYDFN object that includes selection of a file that is not defined on the system
  - Query information derived from a QRYDFN object that includes a numeric constant with a decimal point delimiter other than indicated by the QDECFMT system value.
- Another possible STRQMQRY advantage when requesting summary-only information from large files is a conversion mode (SQL summary) that uses an SQL GROUP BY clause or column functions in the derived query instead of summary function usages in the derived form. However, this more efficient way of producing summary values and omitting detail records can only be used for a QRYDFN with special characteristics. Refer to "Miscellaneous Considerations" on page 7-20.

Using STRQMQRY instead of RUNQRY may not achieve acceptable results, or certain actions may be necessary to ensure a successful outcome. Consider the following items first:

- If STRQMQRY is used and there is a QMQRY or QMFORM named `myqrydfn` in `mylib`, information for running the query or formatting the report will be taken from that object instead of the QRYDFN object unless *ONLY is specified as the ALWQRYDFN parameter value. If *YES has to be specified to use the QRYDFN object with a query management object, one of the objects may have to be renamed or moved.
- Because the query database does not support member specification, it may be necessary to use file overrides to cause the intended member to be used instead of the *FIRST member (refer to "Override Considerations" on page 7-1).
- Both commands have OUTFILE parameters, but STRQMQRY has no *RUNOPT default. This means that if other than displayed output is intended, OUTFILE and related parameters must be specified for STRQMQRY because default values from the QRYDFN object are not used.
- If the QRYDFN object is a dependent query, the dependent values must be set at run-time. For the RUNQRY command, the record selection (RCDSLT) parameter must be used. This requires interactive mode, and causes the *Select records* prompt to be displayed so that the dependent values can be specified. For STRQMQRY, the dependent value names are converted to global variables, which can be set using the SETVAR parameter. The query can be run in batch mode if the SETVAR parameter is used to specify values for all the global variables. In interactive mode, STRQMQRY uses INQUIRY messages to prompt for any global variables not previously set by use of the SETVAR parameter.
- RUNQRY processing makes dynamic adjustments for changes made to a file definition since the query was saved; STRQMQRY processing does not. The QRYDFN object may need to be resaved before STRQMQRY is used.
- If the acceptability of the RUNQRY result depends on any of the following, the STRQMQRY result will probably be unacceptable. Parenthesized phrases indicate the potentially unacceptable system action for each item:

  - Dynamic resolution of field selections
    (The saved field list is used)
  - Data from a file with multiple formats
    (No output is produced)
  - Unmatched join with primary file
    (A matched join is performed)
  - Matched join with primary file
    (A matched join is performed)
  - Control of expression scale and precision
    (Defaults are used for calculations)
  - More than 255 field selections and extra function selections
    (Only the first 255 selections are used)
  - LIKE test of result expression with || or SUBSTR
    (No output is produced)
  - Non-EBCDIC collating
    (No alternative collating is performed)
  - Decimal position override with default numeric editing
    (The number of decimal positions is the default value)
  - Length override n to format n digits or characters
    (The number of digits or characters is

the default value)
- Length override with default numeric editing
  (The default length is used)
- Length override with default column heading
  (The default length is used)
- Edit override with default decimal positions
  (Column values are edited with 0 decimal positions)
- Date and time numeric editing override
  (The run-time default is used)
- Non-SAA numeric editing override
  (SAA edit code K is used)
- Multiple summary functions for field in same column
  (A separate column is added for each summary function)
- Omission of break field column
  (The break field column is not omitted)
- Summary function for break field
  (A separate column is added to hold summary values)
- Break text for non-SQL summary when 1st or only column has summary
  (No break text is displayed)
- Final text for non-SQL summary when 1st or only column has summary
  (No final text is displayed)
- Summary only output with multiple-level break summaries
  (Detail records are not omitted)
- Summary only output with break and final summaries
  (Detail records are not omitted)
- Summary only output with result field used for break control
  (Detail records are not omitted)
- Column function other than COUNT for result field literal
  (No output is produced for SQL summary)
- MAX column function for character field wider than SQL allows
  (No output is produced for SQL summary)
- MIN column function for character field wider than SQL allows
  (No output is produced for SQL summary)
- Total break fields size for GROUP BY wider than SQL allows
  (No output is produced for SQL summary)
- Total break and summary columns size wider than SQL allows
  (No output is produced for SQL summary)
- Line wrapping
  (No line wrapping is used; parallel printer files are produced)
- Cover page text
  (No cover page is printed)
- Page text wider than 55 characters
  (Page text is truncated)
- Truncation of numeric overflow
  (Numeric overflow is rounded)
- Ignoring of decimal data error
  (Errors are not ignored; data is not corrected)

Miscellaneous Considerations

Knowing some things about the differences between Query/400 and query management may help you get better results from using information derived from Query/400 definition objects:

- The form retrieved from a QRYDFN object may have blank column entries causing warnings when the form is used to create a QMFORM object.

- The FROM clause in query source retrieved from a QRYDFN object uses system naming conventions.
- The default printer form width used by query management is smaller than that used by Query/400. Parallel reports may be produced for a report that Query/400 kept on one printer form width.
- Queries defined for files created by other queries may not be usable with files created by query management from information derived from these other queries. This is especially likely if result fields are included in the output.
- Dependent values in record selection tests are converted to global variables, which must be set to suitable values before a derived query is run. For example, t01.cusnam can be set using the global variable name T01_CUSNAM in the SETVAR parameter of STRQMQRY.
- Query/400 cannot run a definition if an expression or value contains a decimal point delimiter that is not recognized. Query management can run such a definition because valid decimal point delimiters are converted to the delimiter indicated by the QDECFMT system value.
- SAA database processing truncates decimal positions after division of unscaled values. The value calculated for 2/3 is 0, for example. For this reason, conversion processing makes sure each numeric constant in an expression contains a decimal point delimiter. This causes the size of the expression column to include 15 digits to the right of the decimal point. Expressions involving division of unscaled numeric fields (no constants) will probably cause unexpected results.
- SAA database processing rules for double-byte character set (DBCS) data are different from those applied for Query/400. Use of concatenated strings to test other than open strings should be avoided.
- Because form text that is too long is truncated, conversion processing compresses blank strings.
- If a QRYDFN object intended to produce summary-only output is suitably defined, query management will convert query information into the following:
  - An SQL SELECT statement with a GROUP BY clause
  - A field selection list containing grouping columns or column functions (for report break control fields and any selected summary functions).

  If no report break is defined, only column functions are used. If a QRYDFN object is not suitably defined for this mode of conversion, detail records will not be omitted. A QRYDFN is not suitable for SQL summary conversion if any of the following applies:
  - Detail output type
  - Multiple break levels
  - Single break level but final summaries are not omitted
  - Result field is used as a break control field
  - Summary function other than COUNT for the result field with no file field reference
  - MIN or MAX for a character field wider than SQL allows
  - Total break fields larger than SQL allows
  - Total break and summary columns larger than SQL allows

- Summary-only output from query management cannot be added to a file created by an earlier Query/400 running of the same QRYDFN object. This is because Query/400 creates the file format with extra fields for holding level and overflow feedback information, which query management does not provide or leave room for.

Limits to Query Management Processing

Query management may not be able to process a report in the manner that you prefer. The following sections discuss the limits to query management processing.

The Query Management Command

The command string on the callable interface is limited to 256 bytes. The command string in a procedure prior to removing the quotation marks is also limited to 256 bytes.

You can specify a limit of 1000 keywords and variables on a single command. This is a combined total of the keywords or variables specified as part of the command string and keywords or variables specified through the extended interface. Duplicate occurrences of the same keyword or variable count as part of the limit.

SQL Query

The size of the SQL query statement after blanks and comments are removed is limited to 32KB minus 1 byte.

Externalized Query

The following limits exist on the source file that makes up the externalized query:

- Data in columns past column 79 is ignored if the record width is greater than 79 bytes.
- You can specify a maximum of 211 929 lines of source text.

Externalized Form

The following limits exist on the source file that makes up the externalized form:

- Data in columns past column 150 is ignored if the record width is greater than 150 bytes.
- Since query management allows duplicate information sections in the externalized form object and allows a file to have an override of MBR(*ALL), there is almost no limit on the number of source records in an externalized form that can be handled.

Instances

You can specify a maximum of 25 query management instances per process or job that are active at any one time.

Global Variables

A maximum of 1000 unique global variables can be set for each query management instance.

Procedures

Query management supports any file width when running and printing a query procedure. When exporting and importing a query procedure, if the source file that is the target of the command is created, it is created with a width of 79 bytes. If the target of the command already exists and has a width less than the source, data may be truncated. If the target has a width greater than the source, no data is lost, and each record is padded with blanks.

Chapter 8. Using Query/400 Definition Information

Query/400 definitions contain specifications for functions that are common to both Query/400 and SAA (CPI) Query. Query management uses this information, saved in Query/400 definition (QRYDFN) objects, to produce reports. This chapter describes how to control query management's use of the information contained in QRYDFN objects, what to do if the results are unsatisfactory, and how to add functions not available through the Query/400 displays.

QRYDFN Conversion

When a query management query (QMQRY) or form (QMFORM) object is needed for command processing, query management searches the library or library list for an object with a name that matches the one specified. You can force query management to skip this search or, to search the library or library list for a Query/400 definition (QRYDFN) with the specified name. If a QRYDFN object is found to match the search, the query or form information required for processing is derived from this object, and appropriate messages and codes are returned to indicate that this has happened.

Query management uses information from a QRYDFN regardless of any problems encountered while deriving that information. No tests about possible defects or functional differences are generated when QRYDFN is used.

The derived information is discarded when the request is completed. Permanent conversion to query management objects can be done by retrieving the query or form source from a QRYDFN object, then using that source to create the query management object of that type.

Applying Query Management to QRYDFN Objects

Query management normally uses information only from query management objects. You can request that query management use Query/400 information if query management form or query information is not available. You can also prevent query management form or query information from being used.

On the START command specify either:

- DSQSCNVT=YES
- DSQSCNVT=ONLY

When using:

- STRQMPRC (Start Query Management Procedure)
- STRQMQRY (Start Query Management Query)
- RTVQMQRY (Retrieve Query Management Query)
- RTVQMFORM (Retrieve Query Management Form)

specify either ALWQRYDFN(*YES) or ALWQRYDFN(*ONLY).

Although query management processing normally uses form and query information derived from query management objects, information from Query/400 definition objects can also be used. To do this, either:

- specify DSQSCNVT = YES or DSQSCNVT = ONLY on the START command

© Copyright IBM Corp. 1991

or

- specify ALWQRYDFN(*YES) or ALWQRYDFN(*ONLY) on the STRQMPRC, STRQMQRY, RTVQMQRY, or RTVQMFORM CL command. This will cause DSQSCNVT to be set to YES or ONLY.

The CPI commands shown in the following examples can be coded in a program or procedure and applied directly to QRYDFN objects.

- RUN QUERY *myqrydfn*
- RUN QUERY *myqrydfn* (FORM=*myqrydfn*
- RUN QUERY *myqrydfn* (FORM=*myqrydfn2*
- RUN QUERY *myqmqry* (FORM=*myqrydfn*
- RUN QUERY *myqrydfn* (FORM=*myqmform*
- PRINT REPORT (FORM=*myqrydfn*
- PRINT QUERY *myqrydfn*
- PRINT FORM *myqrydfn*
- EXPORT QUERY *myqrydfn*
- EXPORT FORM *myqrydfn*

Note: *myqrydfn* and *myqrydfn2* must have unique names so that query management does not find a query or form object of the same name if the DSQSCNVT value is YES instead of ONLY when it searches for an object to use.

For the STRQMQRY, RTVQMQRY, and RTVQMFORM commands, query management will resolve names specified for any QMQRY keyword by looking only for QRYDFN objects if you specify *ONLY as the ALWSQRYDFN keyword value.

If you do not want Query/400 definitions to be used during query management processing, allow query management to default to DSQSCNVT = No on the START command or ALWQRYDFN(*NO) on the STRQMPRC, STRQMQRY, RTVQMQRY, or RTVQMFORM CL commands. Another way to stop query management from using a QRYDFN object is to exclude the library containing the Query/400 definition from the library or library list that query management searches for the information.

You can also convert queries created in a System/36 environment. Use the Convert System/36 Query (CVTS36QRY) command to convert a System/36 query to a Query/400 definition. Query management information can then be derived from the QRYDFN object converted from the System/36 query.

Using the STRQMQRY Command Instead of the RUNQRY Command

You can use both the query management STRQMQRY command and the Query/400 RUNQRY command to produce a formatted report or database file output according to specifications in a previously created QRYDFN object. The following example illustrates the minimum parameter requirements for each command when the object being run is a QRYDFN object:

```
RUNQRY mylib/myqrydfn
STRQMQRY mylib/myqrydfn  QMFORM(*QMQRY) ALWQRYDFN(*ONLY)
```

Using the STRQMQRY command can improve the appearance of a report and provide less restrictive use of QRYDFN information. Unlike the RUNQRY command, STRQMQRY can complete successfully using the following:

- A QRYDFN object saved with errors
- A dependent query in batch mode
- Form and query information from separate objects
- Form information derived from a QRYDFN object that includes selection of a file that is not defined on the system
- Query information derived from a QRYDFN object that includes a numeric constant with a decimal point delimiter not indicated by the QDECFMT system value

Consider the following before using the STRQMQRY command instead of the RUNQRY command.

Note: You may not get acceptable results or you may have to take certain actions to ensure a successful outcome when using the STRQMQRY command.

- If you use the STRQMQRY command without specifying ALWQRYDFN(*ONLY) and there is already a QMQRY or QMFORM with the same name in the specified library, information for running the query or formatting the report is taken from that object instead of the QRYDFN object.
- Since the query database does not support member specification, you may need to use file overrides to use the intended member instead of the *FIRST member (refer to the OVRDBF command in the *CL Reference*).
- Both commands have OUTFILE parameters, but the STRQMQRY command has no *RUNOPT default. If you intend to use output that is not displayed, specify OUTFILE and related parameters for STRQMQRY since values are not defaulted from the QRYDFN object.
- If the QRYDFN object is a dependent query, the dependent values must be set at run time. For RUNQRY, the Record Select (RCDSLT) parameter must be used. The RCDSLT parameter causes the *Select Records* prompt to be displayed so that you can specify the dependent values. For STRQMQRY, dependent value names are converted to global variables that can be set using the SETVAR parameter. The query can be run in batch mode if you use the SETVAR parameter to specify values for all the global variables.

  In interactive mode, the STRQMQRY command uses INQUIRY messages to prompt for any global variables not previously set using the SETVAR parameter.
- RUNQRY command processing makes dynamic adjustments for changes made to a file definition since the query was saved. STRQMQRY command processing does not. The QRYDFN object may need to be saved again before the STRQMQRY command is used.
- If the RUNQRY result depends on any of the following, the result of running the STRQMQRY command will probably be unacceptable. Figure 8-1 on page 8-4 shows the RUNQRY action and the potentially unacceptable system action taken for each item when the STRQMQRY command is run.

Figure 8-1 (Page 1 of 2). RUNQRY and STRQMQRY Actions When Converting a QRYDFN

RUNQRY Action	System Action When Using STRQMQRY
Dynamic resolution of field selections	The saved field list is used
Data from a file with multiple formats	No output is produced
Unmatched join with primary file	A matched join is performed
Matched join with primary file	A matched join is performed
Control of expression scale and precision	Defaults are used for calculations
More than 255 field and extra function selections	Only the first 255 field selections are used
LIKE test of result expression with || or SUBSTR	No output is produced
Non-EBCDIC collating	No alternative collating is performed
Decimal position override with default numeric editing	The default decimal position is used
Length override *n* to format *n* digits or characters	The default number of digits or characters is used
Length override with default numeric editing	The default length is used
Length override with default column heading	The default length is used
Edit override with default Decimal Position	No decimal positions in an edited number
Date and time numeric editing override	Run-time defaults are used
Non-SAA numeric editing override	SAA edit code K is used
Multiple summary functions for field in same column	Separate column for each field summary function is used
Omission of break field column	Break field column is not omitted
Summary function for break field	Separate column added to hold summary function values
Break text for non-SQL summary when first or only column has summary	No break text is displayed
Final text for non-SQL summary when first or only column has summary	No final text is displayed
Summary only output	Detail records are not omitted
Multiple breaks defined	Detail records are not omitted
Single break defined, but level 0 summaries not omitted	Detail records are not omitted
Result field used as break field	Detail records are not omitted
Line wrapping	Line wrapping is not used, parallel printer files are produced
Cover page text	No cover page is printed
Page text wider than 55 characters	Page text is truncated

Figure 8-1 (Page 2 of 2). RUNQRY and STRQMQRY Actions When Converting a QRYDFN

RUNQRY Action	System Action When Using STRQMQRY
Truncation of numeric overflow	Numeric overflow is rounded
Ignoring of decimal data errors	Errors are not ignored, data is not repaired

QRYDFN Conversion Considerations for Satisfactory Results

The information derived from a QRYDFN may be unacceptable for use as a QMQRY or QMFORM. The report or data record output produced from it could have obvious defects, or it could be so different from the report or data record output produced by Query/400 that it cannot be used for the same purpose. On the other hand, you may be able to eliminate unacceptable differences by working on the Query/400 definition to make simple adjustments, such as:

- Condensing text
- Removing underline characters from column headings
- Increasing the space before the value for the first report field
- Specifying length and decimal position overrides for formatting numeric result fields

Report Differences

The following figures show sample report pages contrasting a printed report produced by Query/400 with a printed report produced by query management from information derived from the same Query/400 definition. This definition was picked to show what can happen when derived information is used, and is not necessarily representative of what you can expect from most of your Query/400 definitions.

```
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
   08/11/90  12:25:21                          PAGE    1

 DEPT       SALARY       COMM   Commission
                                as percent of salary
                                ____________________
   10    19,260.25        .00                 .00
         20,010.00        .00                 .00
         21,234.00        .00                 .00
         22,959.20        .00                 .00

    Dept 10  summary information
    MIN  19,260.25        .00
    MAX                   .00

   15    12,258.50     110.10                 .01
         12,508.20     206.60                 .02
         16,502.83   1,152.00                 .07
         20,659.80        .00                 .00

    Dept 15  summary information
    MIN  12,258.50        .00
    MAX              1,152.00

    Year 1988 - salary and commission survey
    MIN  12,258.50        .00
    MAX              1,152.00

* * *  E N D  O F  R E P O R T  * * *


Page    1 **************************** IBM Confidential
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
```

Figure 8-2. Query/400 Output before Adjustment

```
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
                                                                Commission
                                                           as percent of salary
               MIN         MIN         MAX
   DEPT       SALARY       COMM        COMM                 ____________
------- ----------- --------- --------- ----------------------------------
     10   19,260.25         .00         .00          .0000000000000000000000000
          20,010.00         .00         .00          .0000000000000000000000000
          21,234.00         .00         .00          .0000000000000000000000000
          22,959.20         .00         .00          .0000000000000000000000000
        ----------- --------- ---------
Dept 10
          19,260.25         .00         .00

     15   12,258.50      110.10      110.10          .0089815230248399070033038
          12,508.20      206.60      206.60          .0165171647399306055227770
          16,502.83    1,152.00    1,152.00          .0698062089956692276415620
          20,659.80         .00         .00          .0000000000000000000000000
        ----------- --------- ---------
Dept 15
          12,258.50         .00    1,152.00

        =========== ========= =========
Year 19
          12,258.50         .00    1,152.00


               Page        1          **************************** IBM
08/11/90 11:58:42                                                           1
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
```

Figure 8-3. Query Management Output before Adjustment

The following figures show sample report pages contrasting the printed output from query management with that from Query/400 after the following minor changes were made to the QRYDFN object:

- Space before first column overridden to 3
- Length 2 and dec pos 2 specified for the result field size
- Underscores removed from column heading text
- Break and final text condensed
- Page footing text condensed and &PAGE variable removed

```
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
  08/11/90  12:25:21                                    PAGE    1

   DEPT        SALARY       COMM   Commission
                                   as percent of salary
     10     19,260.25        .00               .00
            20,010.00        .00               .00
            21,234.00        .00               .00
            22,959.20        .00               .00

      Dept  10 :
      MIN   19,260.25        .00
      MAX                    .00

     15     12,258.50     110.10               .01
            12,508.20     206.60               .02
            16,502.83   1,152.00               .07
            20,659.80        .00               .00

      Dept  15 :
      MIN   12,258.50        .00
      MAX               1,152.00

      Year 1988:
      MIN   12,258.50        .00
      MAX               1,152.00

  * * *  E N D  O F  R E P O R T  * * *


  ***************** IBM Confidential *****************
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
```

Figure 8-4. Query/400 Output after Adjustment

```
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/

                MIN          MIN         MAX         Commission
   DEPT        SALARY        COMM        COMM   as percent of salary
 -------   -----------   ---------   ---------   --------------------
     10      19,260.25         .00         .00                    .00
             20,010.00         .00         .00                    .00
             21,234.00         .00         .00                    .00
             22,959.20         .00         .00                    .00
           -----------   ---------   ---------
 Dept 10:
             19,260.25         .00         .00

     15      12,258.50      110.10      110.10                    .01
             12,508.20      206.60      206.60                    .02
             16,502.83    1,152.00    1,152.00                    .07
             20,659.80         .00         .00                    .00
           -----------   ---------   ---------
 Dept 15:
             12,258.50         .00    1,152.00

           ===========   =========   =========
Year 1988:
             12,258.50         .00    1,152.00

          ***************** IBM Confidential *****************
08/11/90 12:25:04                                                    1
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
```

Figure 8-5. Query Management Output after Adjustment

Figure 8-6 on page 8-10 and Figure 8-7 on page 8-11 show sample report pages contrasting a printed report produced by Query/400 with a printed report produced by query management from information derived from the same Query/400 definition. This definition was picked to show what can happen when derived information is used, and is not necessarily representative of what you can expect from most of your Query/400 definitions.

```
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
 DEPT        SALARY       COMM   Commission
                                 as percent of salary
                                 --------------------
   10     19,260.25        .00                 .00
          20,010.00        .00                 .00
          21,234.00        .00                 .00
          22,959.20        .00                 .00

    Dept 10  summary information
    MIN  19,260.25
    MAX                    .00                 .00

   15     12,258.50     110.10                 .01
          12,508.20     206.60                 .02
          16,502.83   1,152.00                 .07
          20,659.80        .00                 .00

    Dept 15  summary information
    MIN  12,258.50
    MAX               1,152.00                 .07

    Year 1988 - salary and commission survey
    MIN  12,258.50
    MAX               1,152.00                 .07

* * *  E N D  O F  R E P O R T  * * *


Page    1
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
```

Figure 8-6. Query/400 Output before Adjustment

```
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
                                                        Commission
                                                     as percent of salary



  DEPT       SALARY       COMM                          ______________
------  -----------  ---------  ---------------------------------


    10    19,260.25        .00          .0000000000000000000000000
          20,010.00        .00          .0000000000000000000000000
          21,234.00        .00          .0000000000000000000000000
          22,959.20        .00          .0000000000000000000000000
        -----------  ---------  ---------------------------------
Dept 1
          19,260.25        .00          .0000000000000000000000000

    15    12,258.50     110.10          .0089815230248399070033038
          12,508.20     206.60          .0165171647399306055227770
          16,502.83   1,152.00          .0698062089956692276415620
          20,659.80        .00          .0000000000000000000000000
        -----------  ---------  ---------------------------------
Dept 1
          12,258.50   1,152.00          .0698062089956692276415620
        ===========  =========  =================================
Year 1
          12,258.50   1,152.00          .0698062089956692276415620


               Page        1    **************************** IBM
Page      1                     Date: 1990-01-29
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
```

Figure 8-7. Query Management Output before Adjustment

Figure 8-8 on page 8-12 and Figure 8-9 on page 8-13 show sample report pages contrasting the printed output from query management with that from Query/400 after the following minor changes were made to the QRYDFN object.

- Space 4 before first column instead of 0.
- Specify length 2 and decimal position 2 for the result field size.
- Remove underscores from column heading text.
- Condense break and final text.
- Condense page footing text and remove &PAGE variable.

```
03/18/90  10:01:34                                        PAGE    1

    DEPT   SALARY           COMM             Commission
           (and minimum)    (and maximum)    as percent of salary
                                             (and maximum)
     10         19,260.25              .00                .00
                20,010.00              .00                .00
                21,234.00              .00                .00
                22,959.20              .00                .00

         Dept 10 :
         MIN    19,260.25
         MAX                           .00                .00

     15         12,258.50           110.10                .01
                12,508.20           206.60                .02
                16,502.83         1,152.00                .07
                20,659.80              .00                .00

         Dept 15 :
         MIN    12,258.50
         MAX                      1,152.00                .07

         Year 1988:
         MIN    12,258.50
         MAX                      1,152.00                .07

  * * *  E N D   O F   R E P O R T  * * *
```

Figure 8-8. Query/400 Output after Adjustment

```
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/

                                          Commission
           SALARY          COMM        as percent of salary
    DEPT  (and minimum)  (and maximum)     (and maximum)
---------- ------------- ------------- --------------------


      10      19,260.25           .00                  .00
              20,010.00           .00                  .00
              21,234.00           .00                  .00
              22,959.20           .00                  .00
           ------------- ------------- --------------------
  Dept 10:
              19,260.25           .00                  .00

      15      12,258.50        110.10                  .00
              12,508.20        206.60                  .01
              16,502.83      1,152.00                  .06
              20,659.80           .00                  .00
           ------------- ------------- --------------------
  Dept 15:
              12,258.50      1,152.00                  .06
           ============= ============= ====================
Year 1988:
              12,258.50      1,152.00                  .06


01/01/90 00:00:00                                                      1
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
```

Figure 8-9. Query Management Output after Adjustment

If you intend to start using query management regularly to run a particular Query/400 definition, you should do the following to ensure satisfactory results:

1. Use the Analyze Query (ANZQRY) command and read any diagnostic messages produced, including message help.
2. Use the STRQMQRY command and inspect the output carefully. You may catch undiagnosed defects, or you may discover that some ANZQRY diagnostic messages can be disregarded.
3. Use the WRKQRY command to display the QRYDFN object, and check the definition displays for deviations from the compatibility guidelines. Use the Change option to make any adjustments you think will improve the outcome.
4. Take appropriate action regarding ANZQRY diagnostic messages that do not involve changing the QRYDFN object itself.

The following sections describe in more detail how defects and unacceptable differences can be detected and avoided.

Analyzing a QRYDFN

The Analyze Query (ANZQRY) command is used to inspect a Query/400 definition to find processing problems. This command returns diagnostic messages that detail the potential differences between Query/400 and query management's use of information derived from the analyzed QRYDFN object. A completion message shows the highest severity of the potential differences diagnosed. If this code is greater than 10, check the message help for warnings about easily overlooked and possibly

serious effects of the system action stated in the message. For example, ignoring a collating sequence choice could change the outcome of a character comparison used for controlling record selection or sorting.

Using the ANZQRY command has the following limitations:

- It checks some information about the specified files but ignores any database file override in effect (see the OVRDBF command), and performs no level check. There is no verification that the information saved in the QRYDFN object is still comparable to that found in the file definitions.
- It does not predict errors that could occur during conversion, such as a SELECT statement that grows too large.
- It does not predict errors that could occur during run time, such as the following:
  - SQL syntax errors
  - SQL data errors (missing fields, mismatched type comparisons)
  - Excessive formatted report width
  - Inappropriate summary function or editing for field data type
  - It ignores the following differences between Query/400 and query management:
    - Loss of translatable FINAL TOTALS heading text
    - Loss of leading blanks in column heading lines
    - Different alignment of oversized column headings (centered on axis)
    - Underscores treated as line break characters
    - Break or final text kept from extending into the space before the summary
    - Break or final text kept from extending past last data column
    - Different summary types on the same line
    - Different defaults for result field size
    - Different rules for calculation scale and precision
    - Different values from calculations involving certain data types

      (ANZQRY returned a code of 0 for the QRYDFN object from which query management derived the query and form information for producing the sample report shown on page 8-7.)
    - Severity codes do not take into account the number of messages generated by running the ANZQRY command and may be misleading because a diagnosed condition might not be that severe a problem for the particular QRYDFN analyzed. For example, the fact that decimal data errors will no longer be ignored may or may not be of consequence for a particular query, but is diagnosed as a severity 30 problem even for a query with no numeric fields.

Inspecting the Output

In some cases, the only way of detecting defects and unacceptable differences is to inspect the output. This may also be the only way to evaluate the actual severity of a diagnosed problem using the ANZQRY command. The Start Query Management Query (STRQMQRY) Query Management/400 command and the Run Query (RUNQRY) Query/400 command can be used to get comparable output.

Look for the following differences when running a derived query:

- Records not coming from the *LAST member or the member specified by name
- Omission of unmatched records
- Loss of columns or different column order
- Columns added for extra summary functions
- Different record order or selection set

- Different length or precision of result field data columns
- Different values or unexpected numeric overflow in result fields
- Different values or unexpected numeric overflow of sums or averages
- Divide by zero errors for numeric calculations

Look for the following differences when displaying or printing a report using a derived form:

- Text truncation
- Unresolved text insertion variables
- Editing changes
- Report column width changed
- Heading or footing alignment changed
- No line wrapping
- No cover page
- No page number or date and time information in page headings

Applying QRYDFN Option Guidelines

Many potential problems can be avoided by following guidelines when creating or changing a Query/400 definition intended for query management use. Use the Work with Query (WRKQRY) command to create a new QRYDFN object or change an existing object, and make sure the option fields contain values recommended by the following query management compatibility guidelines:

- Specify file selections
  - Select only files with single formats
  - Select only the *FIRST member
- Specify type of join
  - Use only matched record joining (no primary file)
- Define result fields
  - Specify the value SUBSTR to refer to the result field SUBSTR in an expression
  - Use size overrides only for numeric column formatting control
  - Use a size override if the expression involves division
  - Use '_' in a column heading only where you want the line to break text
  - Do not use column headings with data alignment dependencies
  - Do not use multiple-line figures in column headings
  - Do not use a line for column heading separator characters
  - Use SUBSTR (not a formatting override) to reduce character field size
- Select and sequence fields
  - Make specific field selections
  - Do not select more than 255 fields
- Select records
  - Specify the value SUBSTR to refer to the result field SUBSTR as a test value
  - Do not use LIKE to test a result field defined using || or SUBSTR
  - Use dependent value syntax where a global variable is desired
- Select sort fields
  - Do not sort on a character field unless EBCDIC sequence is acceptable
- Select collating sequence
  - Select EBCDIC sequencing

- Specify report column formatting
  - Allow space for break or final text to the left of summaries
  - If overriding the length of a file field, override the column heading also
  - Use '_' in a column heading only where you want the line to break
  - Do not use column headings with data alignment dependencies
  - Do not use multiple-line figures in column headings
  - Do not use a line for column heading separator characters
  - Do not omit break fields
  - Override editing when overriding numeric file field size
- Define numeric field editing
  - Use only the edit code or numeric editing choices
- Specify edit code
  - Use only the J and M edit codes
  - Do not use the modifier for the asterisk fill option
  - Do not use the modifier for a floating currency symbol with edit code M
- Describe numeric field editing
  - Use only a description that can be converted to an SAA edit code
  - Do not request suppression of zero values
  - Do not request asterisk fill
  - Do not request a single leading zero
- Select report summary functions
  - Do not request more than one summary function per field unless you want columns added for the extra functions
  - Do not request a summary function for a break field unless you want a column added for the summary function
  - Do not request a summary function other than COUNT for a result field that is not calculated from a file field
  - Do not request MIN or MAX for a character field wider than 255
- Define report breaks
  - Do not break on a result field if you want detail records to be omitted
  - Define only one break level if you want detail records to be omitted
  - Do not break on a combination of fields wider than 255 if you want detail records to be omitted
- Format report break
  - Omit level 0 summaries if some other level is defined and you want detail records to be omitted
  - Do not use break text if the first report field has a summary function unless detail records are to be omitted
  - Use any report field name for a break text variable
  - Define final text to replace the FINAL TOTALS default
- Select output type and output form
  - Do not define a query for producing summary-only database file output
  - Put run-time device options in CL commands or procedures
  - Use line spacing if desired
  - Do not define cover page text unless the output will be final summaries only
  - Do not use line wrapping
  - Do not use more than 55 characters for page heading or footing text
  - Make sure no text extends beyond the last data column
  - Use any report field name for a page text variable

- Specify processing options
  - Do not request truncation of numeric overflow
  - Do not request ignoring decimal data errors

Conversion Details

The following figures show the relationship between the WRKQRY displays used to prompt for query definition specifications and the various sections of query management query and form objects. In the figures, dotted lines connect display prompts with object sections built from prompt responses. Dotted lines extending to the right margin represent WRKQRY display choices that are ignored. Dotted lines extending from the left margin represent query management functions that cannot be defined through the prompted interface.

Some SQL functions that query management supports and that are not represented in the figures, such as sublists and certain scalar functions, are not available through the prompted interface. Query management uses one of two conversion modes, depending on whether or not the QRYDFN object is suitably defined for using SQL GROUP BY or column functions to omit detail records. Connections that apply only to SQL summary mode are marked with an asterisk at the left.

```
Specify File Selections
 - File, Library..................... SELECT   FROM
 - Member.....................................................
 - Format.....................................................
 - File ID........................... SELECT   FROM
                                     SELECT   list, ORDER BY, GROUP BY
                                              (used as Field qualifier)
Specify Type of Join
 - Type of join...............................................
Specify How to Join Files
 - Field__Test__Field................ SELECT   WHERE
```

```
Define Result Fields
 - Field__....(if no column heading).... Column:  Column heading, Width
 - Expression__...................... SELECT   list, WHERE
                                              (substituted for Field)
 - Column Heading__.(if no override).... Column:  Column heading, Width
 - Len__............................. Column:  Edit, Width
 - Dec............................... Column:  Edit, Width
```

```
Select and Sequence Fields
 - Seq__Field........................ SELECT   list
```

```
Select Records
 - AND/OR__Field__Test__Value........ SELECT   WHERE
```

```
Select Sort Fields
 - Sort Prty__A/D__Field............. SELECT   ORDER BY
```

```
Select Collating Sequence
 - Collating sequence option..................................
 - Table, Library.............................................
Define Collating Sequence
 - Sequence__Char.............................................
```

```
Specify Report Column Formatting
 - Field__....................................................
..................................... Column:  Seq
..................................... Column:  Datatype
 - Column Spacing__.................. Column:  Indent
 - Column Heading__..(override only).... Column:  Column heading, Width
 - Len__.............(override only).... Column:  Usage, Edit, Width
    (0)                                         (OMIT)
 - Dec__.............(override only).... Column:  Edit, Width
    (#)                                         ( #)
 - Edit..............(override only).... Column:  Edit, Width
    (Untransformable numeric editing)           (K )
Define Numeric Field Editing
 - Edit option................................................
Describe Numeric Field Editing
 - Decimal point, and so on.................. Column:  Edit, Width
    (Transformable description)                 (matched SAA code)
```

Figure 8-10. Correlation between WRKQRY Displays and Query Management Objects

```
Describe Date/Time Field Editing
 - Date/time separator.....................................................
Specify Edit Code
 - Edit code, Optional modifier..... Column:  Edit, Width
    (J )                                       (K )
    (J$)                                       (D )
    (M )                                       (L )
Specify Edit Word
 - Edit word...............................................................
 - Edit word for summary total.............................................
```

```
Select Report Summary Functions
 - Options_Field..................... Column:  Usage, Width
    (1=Total  )                                (SUM    )
    (2=Average)                                (AVERAGE)
    (3=Minimum)                                (MINIMUM)
    (4=Maximum)                                (MAXIMUM)
    (5=Count  )                                (COUNT  )
.................................................(FIRST  )
.................................................(LAST   )
                                          or
 * Options_Field..................... SELECT   list column function
    (1=Total  )                                (SUM(Field))
    (2=Average)                                (AVG(Field))
    (3=Minimum)                                (MIN(Field))
    (4=Maximum)                                (MAX(Field))
    (5=Count  )                                (COUNT(*))
```

```
Define Report Breaks
 - Break level_Sort Prty_Field..... Column:  Usage
    (1-6)                                      (BREAK1-BREAK6)
                                        and
 * Break level_Sort Prty_Field..... SELECT   list, GROUP BY
    (1,2,3,4,5, or 6)
..................................... SELECT   HAVING
Format Report Break (level 0)
..................................... Final:   New page for final text
 - Suppress summaries................ Final:   Put final summary at line
    (Y=Yes, N=No (if text present))            (NONE, 2)
..................................... Final:   Blank lines before footing
 - Break text........................ Final:   Final footing text
    (1st and only line)                        (Line 1)
.................................................(Lines 2-12)
Format Report Break (level 1-6)
 - Skip to new page.................. Break:   New page for heading
..................................... Break:   Repeat column heading
..................................... Break:   Blank lines before heading
..................................... Break:   Blank lines after heading
..................................... Break:   New page for footing
..................................... Break:   Blank lines before footing
```

Figure 8-11. Correlation between WRKQRY Displays and Query Management Objects

```
..................................... Break:   Blank lines after footing
                                        or
 *.....(if no text present).............Break:   Break lines after footing
                                                  (0)
 - Suppress summaries.................. Break:   Put summary at line
   (Y=Yes, N=No (if text present))                (NONE, 2)
 *..................................... Break:   Put summary at line
                                                  (NONE)
.....................................Break:   Break heading text
                                        or
 * Break text.......................... Break:   Break heading text
   (1st and only line)                            (Line 1)
.................................................(Lines 2-5)
 - Break text.......................... Break:   Break footing text
   (1st and only line)                            (Line 1)
.................................................(Lines 2-5)
                                        or
 *.....................................Break:   Break footing text
```

```
Select Output Type and Output Form
 - Output type.....................................................
 - Form of output..................................................
 - Line wrapping...................................................
Define Printer Output
 - Printer.........................................................
 - Form size.......................................................
 - Start line......................................................
 - End line........................................................
 - Line spacing ...................... Options: Detail line spacing
   (1-3)                                          (1-3)
.................................................(4  )
..................................... Options: Outlining for break columns
                                        or
 *..................................... Options: Outlining for break columns
                                                  (N)
..................................... Options: Default break text
                                        or
 *..................................... Options: Default break text
                                                  (N)
..................................... Options: Column wrapped lines kept
..................................... Options: Column heading separators
..................................... Options: Break summary separators
                                        or
 *..................................... Options: Break summary separators
                                                  (N)
..................................... Options: Final summary separators
                                        or
 *..................................... Options: Final summary separators
                                                  (N)
 - Print definition................................................
Define Spooled Output
 - Spool the output................................................
 - Form type.......................................................
 - Copies..........................................................
 - Hold............................................................
Specify Cover Page
 - Print cover page................................................
 - Cover page text (up to 4 lines).................................
                                        or
 * Cover page text (for final summary).. Page:    Page heading text
   (lines 1-4)                                    (Lines n to n+3)
```

Figure 8-12. Correlation between WRKQRY Displays and Query Management Objects

```
Specify Page Headings and Footings
 - Print standard page headings.....................................
..................................... Page:    Blank lines before heading
..................................... Page:    Blank lines after heading
 - Page heading..................... Page:    Page heading text
    (lines 1-3)                                  (Lines 1-3)
.................................................(Lines 4-5)
..................................... Page:    Blank lines before footing
..................................... Page:    Blank lines after footing
 - Page footing..................... Page:    Page footing text
    (1st and only line)                          (Line 1)
.................................................(Lines 2-5)
Define Database File Output
 - File, Library, Member.....................................................
 - Data in file..............................................................
 - Authority (for new file)..................................................
 - Text (for new file).......................................................
 - Print definition..........................................................
```

```
Specify Processing Options
 - Use rounding..............................................................
 - Ignore decimal data errors................................................
```

Figure 8-13. Correlation between WRKQRY Displays and Query Management Objects

Most of the transformations represented in the previous figures are straightforward. The following actions may not be expected, and may need to be understood to get the best results:

- Columns are added to the SELECT list as needed so that each summary function selected for a field has a separate column (unless summaries are completely omitted from the report). Attributes from the original field are carried over to each added field.
- Detail columns are omitted from the SELECT list if the output form is summary-only.
- SQL summary mode is used for detail record omission if the output form is summary-only unless one of the following applies:
  - More than one break level is defined
  - A break level is defined and final summaries are not omitted
  - A result field is used as a break control field
- Every character in an expression or test value is transformed to uppercase unless it is in a string constant. This includes the characters in a name enclosed in quotation marks.
- SQL reserved words used as field names are placed in apostrophes in a derived SELECT statement.
- The corresponding expression is substituted for each result field name which would otherwise appear in a SELECT list or record selection test. Substitution is recursive, since result field names can be used in the expressions for other result fields. Column numbers are substituted for result field names in the ORDER BY clause.
- If there are any join tests, these are used to start the WHERE clause and any record selection tests are separated by the AND keyword.

- Dependent values in record selection tests are converted to global variables. For example: The dependent value :t01."collection" is converted to &T01_COLLECTION.
- Valid decimal point delimiters are converted to the delimiter indicated by the QDECFMT system value in numeric constants in a derived SELECT statement.
- SAA database processing truncates decimal positions after division of unscaled values. The value calculated for 2/3 is 0, for example. For this reason, the decimal point delimiter indicated by the QDECFMT system value is put with every number without a decimal point delimiter in expressions in a derived SELECT statement.
- The column heading for a column is left blank unless there is a column heading override for it; the column will hold expression values (the result field name is the default if no heading is defined), or the column will hold summary values (the program will create a heading).

  A column heading override or default is handled in the following way:

  - *NONE, meaning no column heading, is transformed to a single underline character, or, for a column that will hold summary values, to the caption Query/400 uses for the values.
  - Otherwise, a column heading string is built by stripping leading and trailing blanks from each line up to and including the last non-blank line, then connecting the resulting segments with single underline characters. Underline characters in the heading text are not replaced with substitute characters. A single line heading such as 1_9_9_0 is transformed with no changes and causes 4 lines of heading when the derived form information is used.
  - The heading for a column intended to hold summary values is built by connecting the column heading string to the caption Query/400 uses for the values. If this results in a string longer than 62 characters, the string is truncated.
  - If there is no heading string to connect to the summary function caption, the field name is used. Any heading defined for the field as part of a file definition is ignored.
- The character part of the Edit value is left blank if any of the following occur:
  - The column will hold only COUNT summary function values.
  - There is no size override unless the column is result field for which a size has been specified.
  - There is no override editing defined, the column is not a result field, and there is a numeric decimal positions override.

  The value C is used if the decimal positions override indicates a character field. The value K is used for a numeric field if no closer match to an SAA edit code can be determined for the edit override, or if there is no override and the column is for a result field.

  The following rules apply when trying to determine an SAA edit code match:

  - Only the type of override editing indicated by the Edit option choice is considered.
  - The effect of system defaults on RPG edit code editing is disregarded; J, J with currency symbol, and M are always converted to K, D, and L.

- Edit description choices not involved in the actual editing (a left or right currency symbol when no currency symbols are to be used, for example) are ignored when comparisons are made.

- The numeric part of the Edit value is left blank if the character part is blank or C, or if the number of decimal positions to be used cannot be determined from a decimal positions value saved as an override (with a non-zero length override) or as part of the definition of a result field.
- The Width value for a column is left blank unless the information saved in the QRYDFN object completely specifies the width requirements for everything that has to appear in the column. A column formatting length override can influence the Width, but determines the number of digits or decimal positions formatted only when nothing else in the column (such as a heading segment or count summary value 9,999,999) is wider than the edited data and the override length is smaller than the data width assumed by database processing.
- Cover page text specified for an SQL summary report and defined to show only final summaries is used as extra page heading text. Left alignment is forced and line numbers are assigned starting from 1 if there is no page heading text specified, otherwise starting from 2 plus the number of page heading text lines.
- Final text for an SQL summary report is left-aligned and appears below any summary values.
- Page, break, and final text are adjusted in the following ways:
  - Strings of consecutive blanks are compressed to single blanks.
  - Field referencing insert variables are converted to column referencing variables, and special variables (&time, &date, &page) are converted to all uppercase characters. Strings starting with & are recognized as text insert variables only if the character after the & is not a blank or a numeric digit, and only if ended by the end of text or by one of the following: blank, slash, colon, dash, ampersand, underscore, or DBCS shift out. All selected fields and extra summary function selections are considered (in report order) when converting a field reference to a column reference.
  - If the resolved text is longer than 55 characters, it is truncated at the blank or ampersand in the highest of the first 56 positions. If no blank or ampersand is found, the text is not used.
  - DBCS strings left open due to truncation are closed.

Creating Query Management Objects from QRYDFN Objects

If you want to add query- or form-related function that is not available through the Query/400 prompted interface, you have to use information derived from the QRYDFN object to create a query management object of that type.

Converting QRYDFN Objects

The following steps represent a typical way that a QRYDFN could be permanently converted to a query management form and a query management query:

1. Create separate source file members, one for externalized queries and one for externalized forms.
2. Use the Work with Query (WRKQRY) command to change and save the resolved QRYDFN object if there have been any changes to the file definitions referred to. You could also use this command to change the object to improve the conversion or to add function permitted but not supported by Query/400.

3. Use query management CL or CPI commands (RTVQMQRY or EXPORT QUERY, for example) to extract the usable information.
4. Edit the externalized objects if you want to add functions not available through the Query/400 prompted interface.
5. Use query management CL or CPI commands (CRTQMQRY or IMPORT QUERY, for example) to create query management objects.

The following figure illustrates how a Query/400 QRYDFN is converted to a query management query and a query management form.

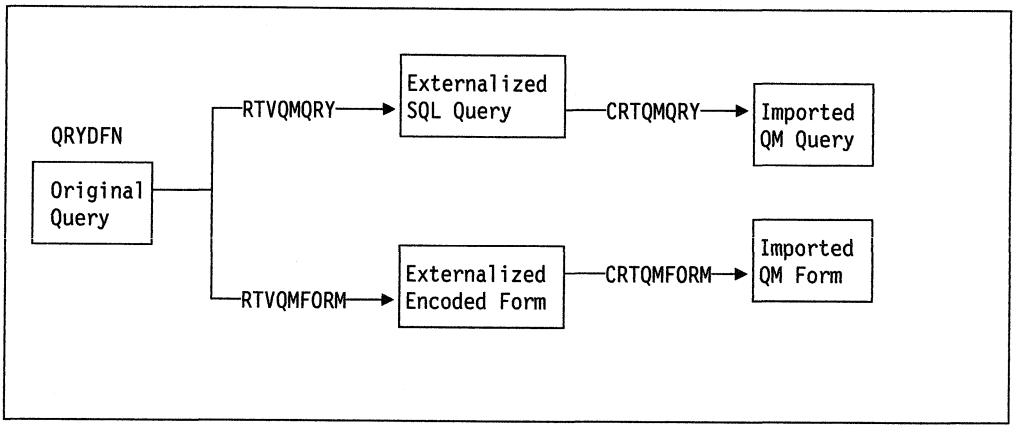


Figure 8-14. Conversion Data Flow

The following figure is an example of how CL commands can be used to convert a Query/400 QRYDFN object to query management objects.

```
                                                Representative work...
                                                 QRY1 in MYLIB saved as
                                                 QRY1 in QTEMP on exit
                                                 from option 2 (Change)
                                                                      |
Type command, press Enter.                                            |
   ► crtsrcpf qtemp/qqmqrysrc 91                                      |
   ► crtsrcpf qtemp/qqmformsrc 162                                    |
   ► wrkqry ------------------------------------------------------------
   ► rtvqmqry qry1qry qtemp/qqmqrysrc qry1
   ► RTVQMFORM command completed using derived information.
   ► strseu qtemp/qqmqrysrc qry1 ---------------------------------------
   ► crtqmqry qry1qry qtemp/qqmqrysrc qry1                              |
   ► rtvqmform qry1frm qtemp/qqmformsrc qry1                            |
   ► RTVQMFORM command completed using derived information.             |
   ► crtqmform qry1frm qtemp/qqmformsrc qry1                            |
===►                                                                    |
                                                Representative editing...
                                                 library/object names in
                                                 FROM clause changed to
                                                 database.object names
```

Figure 8-15. Sample CL Command Sequence for QRYDFN Conversion

Adding SAA Function

The WRKQRY command can be used to define a function that is permitted but not supported by Query/400, for example:

- &field insertion variables in page text
- &field insertion variables ended with '_'
- Other than break field insertion variables in break or final text
- &column# insertion variables, ended with any nondigit character

The Start Source Entry Utility (STRSEU) command can be used to edit retrieved source to add a function that cannot be defined using WRKQRY:

1. Retrieve the source from a QRYDFN object or a query management query (QMQRY) or form (QMFORM) object.
2. Edit the source to add the desired function.
3. Then use the edited source to create a QMQRY or QMFORM object that can be referred to in subsequent query management requests.

The following functions can be added to the type of source indicated:

Query source (a Structured Query Language (SQL) SELECT statement)

- DISTINCT records instead of ALL records
- Selection of all fields using *
- SAA naming conventions in the FROM clause
- Column or scalar function as SQL expression or predicate test value
- NOT as search condition qualifier
- NOT IN and NOT LIKE predicate tests
- GROUP BY and HAVING clauses
- Use of parentheses with connectors, for example (... OR ...) AND (... OR ...)

Form source (encoded records)

- Character field editing (column wrapping)
- Different SAA edit codes for numeric editing
- Explicit column width control
- FIRST and LAST uses
- More than 9 break columns
- Resequencing in form definition
- Report area spacing (blank lines before and after)
- Summary line placement (on other than second break or final footing line)
- Break heading text
- Additional text lines (more than the AS/400 system allows)
- Text line alignment
- Control of framing (separators, default break text, outlining, and so forth)

Refer to the *SAA CPI Database Reference* for detailed information about Structured Query Language (SQL) syntax, and to the *SAA CPI Query Reference* for detailed information about the encoded form layout.

Miscellaneous Considerations

Knowing some things about the differences between Query/400 and query management may help you get better results from using information derived from Query/400 definition objects:

- The form retrieved from a QRYDFN object may have blank column entries causing errors when the form is used to create a QMFORM object.
- The FROM clause in query source retrieved from a QRYDFN object uses system naming conventions.
- The default printer form width used by query management is smaller than that used by Query/400. Similar reports may be produced for a report that Query/400 kept on one printer form width.
- Queries defined for files created by other queries may not be usable with files created by query management from information derived from these other queries. This is especially likely if result fields are included in the output.
- Dependent values in record selection tests are converted to global variables which must be set to suitable values before a derived query is run. For example, t01.cusnam can be set using the global variable name T01_CUSNAM in the SETVAR parameter of STRQMQRY.
- Query/400 cannot run a definition if an expression or value contains a decimal point delimiter that is not recognized. Query management can run such a definition because valid decimal point delimiters are converted to the delimiter indicated by the QDECFMT system value.
- SAA database processing truncates decimal positions after division of unscaled values. For example, the value calculated for 2/3 is 0. For this reason, conversion processing makes sure each numeric constant in an expression contains a decimal point delimiter. This causes the size of the expression column to include 15 digits to the right of the decimal point. Expressions involving division of unscaled numeric fields (no constants) will probably cause unexpected results.
- SAA database processing rules for double-byte character set (DBCS) data are different from those applied for Query/400. Use of concatenated strings to test other than open strings should be avoided.
- Because form text that is too long is truncated, conversion processing compresses blank strings to allow as much space as possible for text.

Appendix A. Message Descriptions

The following error messages are possible when working with query management commands and functions:

FAILURE The query management command issued failed, and processing stops. A message is returned to the display station that details the reason for the command failure and gives some possible solutions for correcting the error.

SEVERE A severe error occurred when query management attempted to process the command issued, and all system function stops. A message is returned to the display station that details the reason for the error and gives some possible solutions for correcting the error.

SUCCESS The query management command issued processed successfully, and data is available for use.

WARNING Query management encountered an error in the command or procedure issued, but processing continues. The error is ignored, or system defaults are used to correct the error. Check the error messages for an explanation of the warning and what, if anything, you can do to correct the error.

© Copyright IBM Corp. 1991

Appendix B. Query Management Interface Example

This appendix shows an example of how you can use query management to create a report. Sample RPG and COBOL programs are provided to show how to create a procedure or program.

Producing a Report

Figure B-1 illustrates using query management to select data and produce a report using predefined query management form (QMFORM) and query management query (QMQRY) objects. Run the SAMP1 program to produce the report.

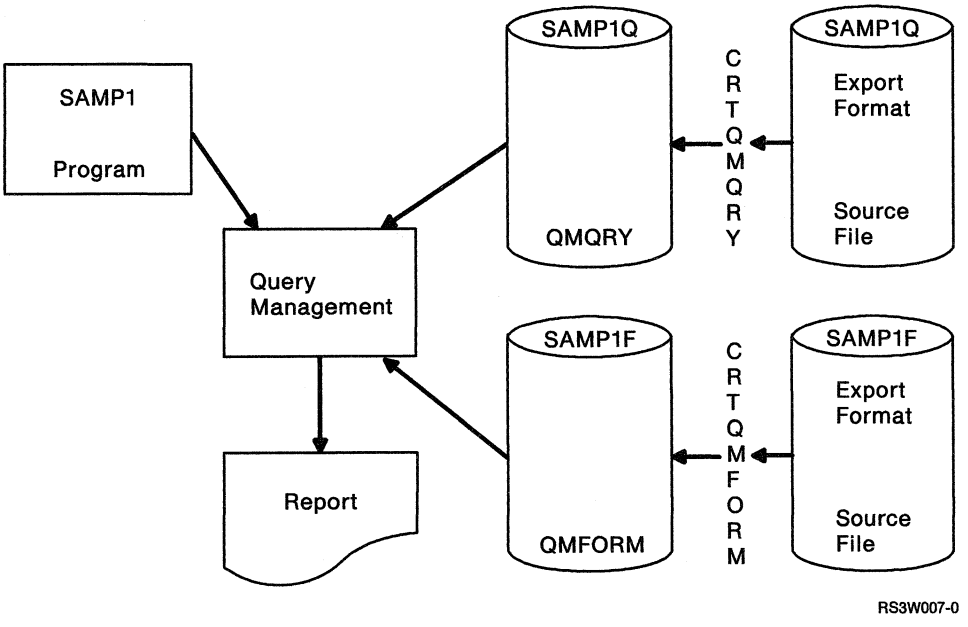


Figure B-1. Overview of Using Query Management to Produce a Report

In the example shown in this appendix, the database file WKPAY exists on the system and contains the following information: employee name, employee number, weekly hours worked, and the hourly rate of pay. Before running the SAMP1 program, create a QMQRY object and a QMFORM object on the system. Do this by first creating the source for the two objects in Query Management Query Source (QMQRYSRC) and Query Management Form Source (QMFORMSRC) files respectively. Then use the Create Query Management Query (CRTQMQRY) and Create Query Management Form (CRTQMFORM) commands to import them to QMQRY and QMFORM formats. Figure B-2 on page B-2 shows the data description specifications (DDS) for the WKPAY file.

© Copyright IBM Corp. 1991

```
     *
     * weekly payroll details
     *
    A                                      UNIQUE
    A          R PAYR                      TEXT('Weekly Pay Record')
    A            EMPNO          5          COLHDG('Employee' 'Number')
    A            NAME          20          COLHDG('Employee' 'Name')
    A            HOURS          5S 2       COLHDG('Hours' 'Worked')
    A            RATE           5S 2       COLHDG('Hourly' 'Rate')
    A            WKAMT          5S 2       COLHDG('Weekly' 'Pay')
    A          K EMPNO
```

Figure B-2. Data Description Specifications for WKPAY File

Figure B-3 shows the query used in the example:

```
SELECT NAME, HOURS, RATE, WKAMT FROM BPLIB/WKPAY
ORDER BY NAME
```

Figure B-3. Query Source Select Statement

Note: The maximum length of this source file is 91 characters (with line number equal to 6 characters, date equal to 6 characters, and data equal to 79 characters). The SQL statement is organized to conform to AS/400 standards, since the option to use *SYS is specified in the START query command issued by the sample program.

Figure B-4 shows the results of running the SAMP1 program.

```
DATE: 11/21/89          WEEKLY PAY REPORT          PAGE:    1

                          HOURS          HOURLY          PAY
EMPLOYEE NAME             WORKED         RATE          AMOUNT

ANDERSON, W                38.50          6.23          100.00
COLLINS, R                 41.50          5.40          400.00
GREEN, C                   40.00          7.10          300.00
SMITH, T                   42.00          5.30          200.00

                                          TOTAL       1,000.00
```

Figure B-4. Report Results for SAMP1

Sample Programs

The sample programs provided in RPG (Figure B-5 on page B-3) and COBOL (Figure B-6 on page B-5) perform the same functions. The programs process the WKPAY file by reading one record at a time, calculating the weekly pay amount, and then updating the file with the calculated information. When all the records are updated, query management is started with a call to the interface program QQXMAIN and passes the START command and the communications area. The START command specifies that the interface session uses system naming conventions (*SYS) and not the default (*SAA).

Perform the query by calling the callable interface and passing the RUN command. Print the results using the PRINT command. Use the EXIT command to end the interface, and end the program with control returning to the calling program.

When passing query management data to the interface, it is necessary to pass the lengths of commands and parameters in integer (binary) format. Structures have been set up in the program to allow data to be passed in this format.

A module containing the communications area is included in the program during compilation. Use this to communicate the status of operations between query management and the user program. The interface program name and standard field names for the query status are also defined in this include module. Use these names whenever required in the application program to allow for the transfer of query applications between SAA systems.

Sample RPG Program

Figure B-5 is a sample RPG program to process the WKPAY file.

```
*****************************************************************
*                                                               *
*        SAMPLE 1 RPG PROGRAM USING QUERY INTERFACE             *
*        ------------------------------------------             *
*                                                               *
*  1) Include member DSQCOMMR contains the communications       *
*     area to be passed to the query management interface.      *
*  2) The WKPAY weekly payroll details are read and the hours   *
*     worked are multiplied by the hourly rate to calculate     *
*     the weeks pay. The file is then updated with the weekly   *
*     pay amount.                                               *
*  3) Once all the records in the WKPAY file are updated then   *
*     the interface is started and a query report               *
*     printed using the just updated file.                      *
*                                                               *
*****************************************************************
H
FWKPAY   UF  E                    DISK
 *
E                   COM     1   4 25              interface cmds
 *
I            DS
I                                      B   1   40BIN1
I                                      B   5   80BIN2
I                                      B   9  120BIN3
I                                      B  13  160BIN4
I/COPY QRPG/QIRGINC,DSQCOMMR
 *
 * Update the Weekly Pay file with weekly earnings:
 *
C           *IN50     DOUEQ'1'
C                     READ WKPAY                    50 EOF
C  N50      HOURS     MULT RATE      WKAMT      H      calculate pay
C  N50                UPDATPAYR                        update pay file
C                     END
 *                                                     ensure all
C                     FEOD WKPAY                       changes done
```

Figure B-5 (Part 1 of 2). Sample RPG Program

```
     *                                                        in storage
     * Start the query interface session:
     *
    C                   CALL DSQCIR
    C                   PARM           DSQCOM          comms area
    C                   PARM 5         BIN1            command length
    C                   PARM           COM,1           START
    C                   PARM 1         BIN2            # keywords
    C                   PARM 8         BIN3            keyword length
    C                   PARM 'DSQSNAME'DATA8   8       keyword
    C                   PARM 4         BIN4            value length
    C                   PARM '*SYS'    DATA4   4       value
    C                   PARM DSQVCH    TYPE    4       CHAR
     *
     * Run the query:
     *
    C                   CALL DSQCIR
    C                   PARM           DSQCOM          comms area
    C                   PARM 16        BIN1            command length
    C                   PARM           COM,2           RUN QUERY
     *                                                    SAMP1Q
     * Print the results of the query:
     *
    C                   CALL DSQCIR
    C                   PARM           DSQCOM          comms area
    C                   PARM 25        BIN1            command length
    C                   PARM           COM,3           PRINT REPORT
     *                                                   (FORM=SAMP1F
     * End the query interface session:
     *
    C                   CALL DSQCIR
    C                   PARM           DSQCOM          comms area
    C                   PARM 4         BIN1            command length
    C                   PARM           COM,4           EXIT
     *
    C                   MOVE '1'       *INLR           end the program
     *
**    commands loaded as compile time array
START
RUN QUERY SAMP1Q
PRINT REPORT (FORM=SAMP1F
EXIT
```

Figure B-5 (Part 2 of 2). Sample RPG Program

Sample COBOL Program

Figure B-6 is a sample COBOL program to process the WKPAY file.

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID.     SAMP1.
       DATE-COMPILED.
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION
       SOURCE-COMPUTER.    IBM-AS400.
       OBJECT-COMPUTER.    IBM-AS400.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
           SELECT PAY-FILE
               ASSIGN TO DISK-WKPAY
               FILE STATUS IS PAY-FILE-STATUS.
       DATA DIVISION.
       FILE SECTION.
       FD  PAY-FILE LABEL RECORDS STANDARD.
       01  PAY-REC.
           COPY DDS-PAYR OF WKPAY.
       WORKING-STORAGE SECTION.
      ****************************************************************
      *                                                              *
      *         SAMPLE 1 COBOL PROGRAM USING QUERY INTERFACE         *
      *         ----------------------------------------             *
      *                                                              *
      *  1) Include member DSQCOMMB contains the communications      *
      *     area to be passed to the query management interface.     *
      *                                                              *
      *  2) The WKPAY weekly payroll details are read and the hours  *
      *     worked are multiplied by the hourly rate to calculate    *
      *     the weeks pay. The file is then updated with the weekly  *
      *     pay amount.                                              *
      *                                                              *
      *  3) Once all the records in the WKPAY file are updated,      *
      *     the interface is started and a query report              *
      *     printed using the file just updated.                     *
      *                                                              *
      ****************************************************************

     * Include the Communications area

        COPY DSQCOMMB OF QLBL-QILBINC.

      * Query Interface Commands

       77   START-CMD           PIC X(5)   VALUE "START".
       77   KEYWORD-NAME        PIC X(8)   VALUE "DSQSNAME".
       77   NAME-VALUE          PIC X(4)   VALUE "*SYS".
       77   RUN-QUERY-CMD       PIC X(16)  VALUE "RUN QUERY SAMP1Q".
       77   PRINT-CMD           PIC X(25)
                                 VALUE "PRINT REPORT (FORM=SAMP1F".
       77   EXIT-CMD            PIC X(4)   VALUE "EXIT".
```

Figure B-6 (Part 1 of 3). Sample COBOL Program

```
77   ONE                 PIC 9(8) USAGE IS BINARY VALUE 1.
77   FOUR                PIC 9(8) USAGE IS BINARY VALUE 4.
77   FIVE                PIC 9(8) USAGE IS BINARY VALUE 5.
77   EIGHT               PIC 9(8) USAGE IS BINARY VALUE 8.
77   SIXTEEN             PIC 9(8) USAGE IS BINARY VALUE 16.
77   TWENTY-FIVE         PIC 9(8) USAGE IS BINARY VALUE 25.

77   PAY-FILE-STATUS     PIC XX.

01   FILE-END            PIC X VALUE SPACE.
     88 END-OF-FILE      VALUE "E".

01   FILE-ERROR-INFO.
     05  OP-NAME         PIC X(7).
     05  FILLER          PIC X(20) VALUE " ERROR ON FILE WKPAY".
     05  FILLER          PIC X(18) VALUE " - FILE STATUS IS ".
     05  STATUS-VALUE    PIC XX.

PROCEDURE DIVISION.
DECLARATIVES.
FILE-ERROR SECTION.
    USE AFTER STANDARD ERROR PROCEDURE ON PAY-FILE.
FILE-ERROR-PARA.
    MOVE PAY-FILE-STATUS TO STATUS-VALUE.
    DISPLAY "FILE PROCESSING ERROR".
    DISPLAY FILE-ERROR-INFO.
    DISPLAY "PROCESSING ENDED DUE TO FILE ERROR".
    STOP RUN.
END DECLARATIVES.

FILE-UPDATE SECTION.
OPEN-FILE.
    MOVE "OPEN" TO OP-NAME.
    OPEN I-O PAY-FILE.
    PERFORM READ-PAY-FILE THRU UPDATE-PAY-FILE
        UNTIL END-OF-FILE.
READ-PAY-FILE.
    MOVE "READ" TO OP-NAME.
    READ PAY-FILE
        AT END SET END-OF-FILE TO TRUE
        MOVE "CLOSE" TO OP-NAME
        CLOSE PAY-FILE
        PERFORM PROCESS-QUERY.
UPDATE-PAY-FILE.
    MULTIPLE HOURS BY RATE GIVING WKAMT ROUNDED.
    MOVE "UPDATE" TO OP-NAME.
    REWRITE PAY-REC.

* Query Interface command and parameter lengths

PROCESS-QUERY SECTION.
START-INTERFACE.
    CALL DSQCIB USING DSQCOMM, FIVE, START-CMD,
                      ONE, EIGHT, KEYWORD-NAME,
                      FOUR, NAME-VALUE, DSQ-VARIABLE-CHAR.
```

Figure B-6 (Part 2 of 3). Sample COBOL Program

```
RUN-QUERY.
    CALL DSQCIB USING DSQCOMM, SIXTEEN, RUN-QUERY-CMD.
PRINT-REPORT.
    CALL DSQCIB USING DSQCOMM, TWENTY-FIVE, PRINT-CMD.
EXIT-INTERFACE.
    CALL DSQCIB USING DSQCOMM, FOUR, EXIT-CMD.
    STOP RUN.
```

Figure B-6 (Part 3 of 3). Sample COBOL Program

Query and Form Source

The sample RPG and COBOL programs in Figure B-5 on page B-3 and Figure B-6 on page B-5 refer to query and form source files to produce a report. Figure B-7 is the query source file (SAMP1Q) referred to in the sample programs.

```
SELECT NAME, HOURS, RATE, WKAMT FROM BPLIB/WKPAY
ORDER BY NAME
```

Figure B-7. Sample Query Source

Figure B-8 is the form source (SAMP1F) referred to in the sample programs.

```
H QM4 01 F 01 E E   E R 01 03 89/11/20 15:51
T 1110 004 005 1112 007 1115 006 1116 005 1118 003 1113 015
R CHAR    2      30    1   Employee_Name
R NUMERIC 2      8     2   Hours_Worked
R NUMERIC 2      8     3   Hourly_Rate
R NUMERIC 2      8     4   Weekly_Pay
E
```

Figure B-8. Sample Form Source

Query and Form Printed Output

Figure B-9 is the printed output resulting from running the command PRINT QUERY SAMP1Q on the query source (SAMP1Q) referred to in the sample programs.

```
                                IBM Query Management/400

Query  . . . . . :   SAMP1Q
  Library  . . . . :     BPLIB
Text . . . . . :   SAA Query
SEQNBR *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8...
000001 SELECT NAME, HOURS, RATE, WKAMT FROM BPLIB/WKPAY
000002 ORDER BY NAME
                                          * * * * *   END OF SOURCE   * * * * *
```

Figure B-9. Printed Output of Query Source

Figure B-10 is the printed output resulting from running the command PRINT FORM SAMP1F on the form source (SAMP1F) referred to in the sample programs.

```
Form . . . . . :   SAMP1F
  Library  . . . :     BPLIB
Text . . . . . :   Form layout for sample 1 program
                                                    Column Information
Nbr     Heading                 Usage      Type       Indent     Width     Edit      Seq
  1     Employee_Name                      CHAR            2        30                1
  2     Hours_Worked                       NUMERIC         2         8                2
  3     Hourly_Rate                        NUMERIC         2         8                3
  4     Weekly_Pay                         NUMERIC         2         8                4
                                                     Page Information
Heading text  . . . . . . . . . . . . . . . :  NO
Blank lines before heading  . . . . . . . . :  0
Blank lines after heading . . . . . . . . . :  2
Footing text  . . . . . . . . . . . . . . . :  NO
Blank lines before footing  . . . . . . . . :  2
Blank lines after footing . . . . . . . . . :  0
                                                     Final Information
Final text  . . . . . . . . . . . . . . . . :  NO
New page for final text . . . . . . . . . . :  NO
Put final summary at line . . . . . . . . . :  1
Blank lines before text . . . . . . . . . . :  0
                                                     Break Information
Break number  . . . . . . . . . . . . . . . :  1
Columns with this break number  . . . . . . :  NONE
Heading text  . . . . . . . . . . . . . . . :  NO
New page for heading  . . . . . . . . . . . :  NO
Blank lines before heading  . . . . . . . . :  0
Blank lines after heading . . . . . . . . . :  0
Repeat column headings  . . . . . . . . . . :  NO
Footing text  . . . . . . . . . . . . . . . :  NO
New page for footing  . . . . . . . . . . . :  NO
Blank lines before footing  . . . . . . . . :  0
Blank lines after footing . . . . . . . . . :  1
Put break summary at line . . . . . . . . . :  1
                                                     Break Information
Break number  . . . . . . . . . . . . . . . :  2
Columns with this break number  . . . . . . :  NONE
Heading text  . . . . . . . . . . . . . . . :  NO
New page for heading  . . . . . . . . . . . :  NO
Blank lines before heading  . . . . . . . . :  0
Blank lines after heading . . . . . . . . . :  0
Repeat column headings  . . . . . . . . . . :  NO
```

Figure B-10 (Part 1 of 3). Printed Output of Form Source

```
Form . . . . . :   SAMP1F
  Library  . . . :     BPLIB
Text . . . . . :   Form layout for sample 1 program
Footing text  . . . . . . . . . . . . . . . :   NO
New page for footing  . . . . . . . . . . . :   NO
Blank lines before footing  . . . . . . . . :   0
Blank lines after footing . . . . . . . . . :   1
Put break summary at line . . . . . . . . . :   1
                                                        Break Information
Break number  . . . . . . . . . . . . . . . :   3
Columns with this break number  . . . . . . :   NONE
Heading text  . . . . . . . . . . . . . . . :   NO
New page for heading  . . . . . . . . . . . :   NO
Blank lines before heading  . . . . . . . . :   0
Blank lines after heading . . . . . . . . . :   0
Repeat column headings  . . . . . . . . . . :   NO
Footing text  . . . . . . . . . . . . . . . :   NO
New page for footing  . . . . . . . . . . . :   NO
Blank lines before footing  . . . . . . . . :   0
Blank lines after footing . . . . . . . . . :   1
Put break summary at line . . . . . . . . . :   1
                                                        Break Information
Break number  . . . . . . . . . . . . . . . :   4
Columns with this break number  . . . . . . :   NONE
Heading text  . . . . . . . . . . . . . . . :   NO
New page for heading  . . . . . . . . . . . :   NO
Blank lines before heading  . . . . . . . . :   0
Blank lines after heading . . . . . . . . . :   0
Repeat column headings  . . . . . . . . . . :   NO
Footing text  . . . . . . . . . . . . . . . :   NO
New page for footing  . . . . . . . . . . . :   NO
Blank lines before footing  . . . . . . . . :   0
Blank lines after footing . . . . . . . . . :   1
Put break summary at line . . . . . . . . . :   1
                                                        Break Information
Break number  . . . . . . . . . . . . . . . :   5
Columns with this break number  . . . . . . :   NONE
Heading text  . . . . . . . . . . . . . . . :   NO
New page for heading  . . . . . . . . . . . :   NO
Blank lines before heading  . . . . . . . . :   0
Blank lines after heading . . . . . . . . . :   0
Repeat column headings  . . . . . . . . . . :   NO
Footing text  . . . . . . . . . . . . . . . :   NO
New page for footing  . . . . . . . . . . . :   NO
```

Figure B-10 (Part 2 of 3). Printed Output of Form Source

```
                                IBM Query Management/400

Form . . . . . :   SAMP1F
  Library  . . . :     BPLIB
Text . . . . . :   Form layout for sample 1 program
Blank lines before footing  . . . . . :   0
Blank lines after footing . . . . . :   1
Put break summary at line . . . . . :   1
                                                  Break Information
Break number  . . . . . :   6
Columns with this break number  . . . . . :   NONE
Heading text  . . . . . :   NO
New page for heading  . . . . . :   NO
Blank lines before heading  . . . . . :   0
Blank lines after heading . . . . . :   0
Repeat column headings  . . . . . :   NO
Footing text  . . . . . :   NO
New page for footing  . . . . . :   NO
Blank lines before footing  . . . . . :   0
Blank lines after footing . . . . . :   1
Put break summary at line . . . . . :   1
                                                  Option Information
Detail line spacing . . . . . :   1
Outlining for break columns . . . . . :   YES
Default break text  . . . . . :   YES
Column wrapped lines kept on page . . . . . :   YES
Column heading separators . . . . . :   YES
Break summary separators  . . . . . :   YES
Final summary separators  . . . . . :   YES
              * * * * *  E N D  O F  C O M P U T E R  P R I N T O U T  * * * * *
```

Figure B-10 (Part 3 of 3). Printed Output of Form Source

Control Language Interface

You can use query management to create simple applications for report generation. By defining a command, a control language (CL) program, a query, and a form, you can prompt the user for any amount of information required to generate a meaningful report.

Creating a QMQRY Object

You can define a query (QMQRY object) using an SQL statement. Figure B-11 is an example of how to create the QMQRY object SALARYQ2 using an SQL statement.

```
                                IBM Query Management/400

Query  . . . . . :   SALARYQ2
  Library  . . . :     EXAMPLE
Text . . . . . :   TEST QUERY
SEQNBR |...+....1....+....2....+....3....+....4....+....5....+....6....
000001 SELECT DEPT,NAME,ID,JOB,YEARS,SALARY,COMM
000002        FROM TESTDATA/STAFF
000003        WHERE DEPT = &COND1 AND SALARY < &COND2
000004        ORDER BY DEPT,SALARY
                                  * * * * *   END OF SOURCE   * * *
```

Figure B-11. Test Query SELECT Statement

For more information about creating QMQRY objects, see "Creating a Query Management Object" on page 3-54.

Creating a QMFORM Object

You can define a form (QMFORM object) to specify the information to be included in the report generated. Figure B-12 is an example of how to create the QMFORM object SALARYF2 that contains the following information:

```
                                IBM Query Management/400

Form . . . . . :   SALARYF2
  Library  . . . :     EXAMPLE
Text . . . . . :   TEST FORM
                                                    Column Information
Nbr     Heading                 Usage       Type      Indent     Width     Edit      Seq
  1                             BREAK1                  0                              1
  2                                                     2                              2
  3                                                     2                              3
  4                                                     2                              4
  5                             AVERAGE                 2                              5
  6                             AVERAGE                 2                              6
  7                                                     2                              7
                                                   Page Information
Heading text  . . . . . . . . . . . . . . . :   NO
Blank lines before heading  . . . . . . . . :   0
Blank lines after heading . . . . . . . . . :   2
Footing text  . . . . . . . . . . . . . . . :   NO
Blank lines before footing  . . . . . . . . :   2
Blank lines after footing . . . . . . . . . :   0

Break number  . . . . . . . . . . . . . . . :   1
Columns with this break number  . . . . . . :   1
Heading text  . . . . . . . . . . . . . . . :   NO
New page for heading  . . . . . . . . . . . :   NO
Blank lines before heading  . . . . . . . . :   0
Blank lines after heading . . . . . . . . . :   0
Repeat column headings  . . . . . . . . . . :   NO
Footing text  . . . . . . . . . . . . . . . :   YES
New page for footing  . . . . . . . . . . . :   NO
Blank lines before footing  . . . . . . . . :   0
Blank lines after footing . . . . . . . . . :   1
Put break summary at line . . . . . . . . . :   1
Line  Align   Footing Text
  1   RIGHT   Dept Avg:
```

Figure B-12. Test Form Statement

For more information about creating QMFORM objects, see "Creating a Query Management Object" on page 3-54.

Sample CL Program

The CL program in Figure B-13 uses the query shown in Figure B-11 and the form shown in Figure B-12.

```
                                    SEU SOURCE LISTING
SOURCE FILE . . . . . .  EXAMPLE/SOURCE
MEMBER  . . . . . . . .  CLPGM
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...
  100 PGM PARM(&VAR1 &VAR2 &VAR3 &VAR4)
  200 DCL &VAR1 *CHAR LEN(6)
  300 DCL &VAR2 *CHAR LEN(6)
  400 DCL &VAR3 *CHAR LEN(6)
  500 DCL &VAR4 *CHAR LEN(10)
  600 STRQMQRY QMQRY(EXAMPLE/SALARYQ2) QMFORM(EXAMPLE/&VAR4) +
  700          OUTPUT(&VAR3)                                 +
  800          SETVAR((COND1 &VAR1) (COND2 &VAR2))
  900 ENDPGM
                          * * * *  E N D  O F  S O U R C E  * * * *
```

Figure B-13. CL Program Source File

Figure B-14 shows the command to run this example CL program.

```
                                    SEU SOURCE LISTING
SOURCE FILE . . . . . .  EXAMPLE/SOURCE
MEMBER  . . . . . . . .  DEPTREP1
SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...
  200 CMD   PROMPT('Department Report')
  300             PARM      KWD(VAR1) TYPE(*CHAR) LEN(6) RSTD(*YES) +
  301                         DFT(10) VALUES(10 20 30 40 50) +
  400                         PROMPT('Department to report on')
  500             PARM      KWD(VAR2) TYPE(*CHAR) LEN(6) DFT(100000) +
  600                         PROMPT('With salary less than')
  700             PARM      KWD(VAR3) TYPE(*CHAR) LEN(6) RSTD(*YES) +
  701                         DFT(*) VALUES(* *PRINT) PROMPT('Output +
  800                         Loc. (*/*PRINT)')
  900             PARM      KWD(VAR4) TYPE(*CHAR) LEN(10) RSTD(*YES) +
  901                         DFT(SALARYF2) VALUES(BYDEPT BYDIVISION +
 1000                         SALARYF2) PROMPT('Report name')
                            * * * *  E N D  O F  S O U R C E  * * * *
```

Figure B-14. CL Command Source File

When you enter the command EXAMPLE/DEPTREP and press F4 to prompt, the following display appears:

```
                        Department Report (DEPTREP)

Type choices, press Enter.
Department to report on . . . .  10           10, 20, 30, 40, 50
With salary less than . . . . .  100000       Character value
Output Loc. (*/*PRINT) . . . . .  *PRINT      *, *PRINT
Report name . . . . . . . . . .  SALARYF2     BYDEPT, BYDIVISION, SALARYF2

                                                                      Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Press the Enter key to use the default values for this job.

The report shown in Figure B-15 is displayed with the department averages for salary and years of experience for department 10.

DEPT	NAME	ID	JOB	YEARS	SALARY	COMM
10	DANIELS	240	MGR	5	19,260.25	.00
	LU	210	MGR	10	20,010.00	.00
	JONES	260	MGR	12	21,234.00	.00
	MOLINARE	160	MGR	7	22,959.20	.00
			Dept Avg:	9	20,865.86	

Figure B-15. CL Program Report Example

Bibliography

The following manuals contain information about topics described or referred to in this guide. They are listed with their full title and base order number. When these manuals are referred to in text, a shortened version of the title is used.

- The *Database Guide*, SC41-9659, provides a detailed discussion of the AS/400 database organization, including information on how to create, describe, and manipulate database files on the AS/400 system.

  Short title: *Database Guide*

- The *Data Management Guide*, SC41-9658, provides information about the fundamental structure and concepts of data management support of OS/400 files.

  Short title: *Data Management Guide*

- *Languages: Systems Application Architecture* AD/Cycle* RPG/400* User's Guide*, SC09-1348, provides information needed to write, test, and maintain RPG/400 programs on the AS/400 system.

  Short title: *RPG/400* User's Guide*

- The *Programming: Control Language Reference*, SC41-0030, provides information about using AS/400 control language (CL) and its commands when working with licensed programs.

  Short title: *CL Reference*

- *Programming: Query Management/400 Reference*, SC41-8193, contains information about how to write applications that use Query Management/400.

  Short title: *Query Management/400 Reference*

- *Query/400 User's Guide*, SC41-9614, provides information on working with Query/400 and how to use Query/400 to get data from any database file.

  Short title: *Query/400 User's Guide*

- *SAA CPI Database Reference*, SC26-4348, provides information pertaining to SAA CPI database.

- *SAA CPI Query Reference*, SC26-4349, provides information pertaining to SAA Query.

- *Security Concepts and Planning*, SC41-8083, provides information about system security concepts, planning for security, and setting up security on the AS/400 system.

  Short title: *Security Concepts and Planning*

- *Systems Application Architecture: An Overview*, GC26-4341, provides general information on SAA functions.

- *Systems Application Architecture* Structured Query Language/400 Programmer's Guide*, SC41-9609, provides information on how to design, write, run, and test SQL/400 statements. It also describes Interactive Structured Query Language (ISQL).

  Short title: *SQL/400* Programmer's Guide*

© Copyright IBM Corp. 1991

Index

© Copyright IBM Corp. 1991

E

F

Q

R

S

Readers' Comments

Application System/400™
Programming:
Query Management/400
Programmer's Guide
Version 2

Publication No. SC41-8192-00

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

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.

Be sure to print your name and address below if you would like a reply.

Name _____ Address _____

Company or Organization _____ _____

Phone No. _____ _____

IBM®

Cut o
Along

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

ATTN DEPT 245
IBM CORPORATION
3605 HWY 52 N
ROCHESTER MN 55901-7899

Fold and Tape **Please do not staple** Fold and Tape

Cut o
Along

Program Number: 5738-SS1

Printed in Denmark by Interprint

SC41-8192-00