The iSeries & AS/400
Programmer’s Guide to
COOLTHINGS
Mike Faust
iSeries and AS/400®
Programmer’s Guide to Cool Things
To the memory of my father; Raymond (Jim) Faust who always told me that anything you can imagine is possible.
iSeries and AS/400®
Programmer’s Guide
to Cool Things

Mike Faust

MC PRESS
Contents

Section I: Green Screen Cool Things ........................................... 1

Chapter 1: Creating Reports with Query Manager/400 .................. 3
           Query Manager Queries ......................................... 6
           Query Manager Report Forms .................................. 14
           Query Manager Tables .......................................... 25
           Query Manager Profiles ........................................ 28
           Query Manager Procedures ..................................... 29

Chapter 2: Creating Green Screen GUI Applications ................. 33
           Display File Windows ........................................... 33
           Building Menus .................................................... 41
           Additional GUI Objects .......................................... 52

Chapter 3: ILE RPG User-Defined Functions ......................... 59
           Creating the Modules ............................................ 60
           Useable Samples .................................................. 70
           Pop-Up Calendar ................................................. 75
           Message Box ....................................................... 84
           Pop-Up Calculator ................................................ 87
           List Box ............................................................ 91

Chapter 4: Advanced Function Printing ............................... 99
           The Set Up .......................................................... 100
           The Keywords ..................................................... 101
           Page Definition Keywords ..................................... 102
           Text and Font Keywords ........................................ 106
           Graphic Keywords ............................................... 109
Section I

Green Screen Cool Things

Even with the dominance of graphical Windows applications and Web-browsers, let’s face it, most of the day-to-day functions performed on the 400 are done using a dumb-terminal display or terminal emulator. That’s why this section focuses on things that you may not know about that can be done from a “green-screen.” We’ll start by examining the power of DB2 Query Manager for AS/400. We’ll show you a utility that allows you to easily create reports without writing a single line of RPG code. All without sacrificing flexibility like with Query/400. Once you’ve seen how powerful Query Manager is you won’t want to write another report program in RPG again.

Of course you still do need RPG to create interactive applications. That’s why chapter 2 covers dressing up your green-screen applications by using enhanced display functions. This set of display file functions allows you to give green screen applications the look and feel of an application with a graphical user interface (GUI). This gives you the ability to create windows, push buttons, check boxes, radio buttons and even a list box all on a green-screen display. You can
make your applications look better while making them easier to use at the same
time all with the addition of a few simple DDS keywords.

After we examine enhanced display functions, we’ll take a look at creating
user-defined functions in ILE RPG in chapter 3. User-defined functions can be
used to perform tasks from the most basic arithmetic calculation to a more com-
plex function that reads physical file data, displays it, and then returns a selected
value. We’ll look at a few useful examples that you can incorporate into your ap-
plications. In this chapter we’ll also look at using FTP to do more than simply
send and receive files.

Once you’re done with this section you’ll have added a plethora of new tools to
your programming toolbox.

To download the code for all the examples in this book, do the following:

1. Go to MCPressOnline.com
2. Click on Help on the top navigational bar
3. Click on MC-Store.com Support
4. Click on MC Product Support
5. Click on Product Updates
Creating Reports
with Query Manager/400

To start, we examine how to use Query Manager/400 as an alternative to writing RPG programs or using Query 400 to create custom reports. QM/400 gives you the ability to create printed reports with nearly all the flexibility of writing a program in RPG in only slightly more time than it takes to create a query using Query/400. The best part is that you can easily convert a Query/400 query definition into QM/400.

Let me start out by defining a few terms. Query management queries are a standard part of OS/400. These are created using source files containing an SQL statement defining the data for your query and a separate source file member containing the definition of the report layout. The term Query Manager as used in this chapter refers to DB2 UDB Query Manager for AS/400, which is part of DB2 Query Manager and SQL Development kit licensed program number 5769ST1. This product is a more user-friendly interface to Query Management queries and forms.
There are two basic components used to generate a report using QM/400. These are the Query Manager query, which is used to define what data will appear on your report, and the Query Manager report form, which is used to define the look of the report, including column and page headings and level breaks. This means that you can create one Query Manager report form that can be used by one or many Query Manager queries. This in itself can help to greatly reduce your development time for creating new reports because reports with a common look can use the same Query Manager report form but use unique Query Manager queries to define their data sets.

The flexibility in QM/400 is partially achieved through the ability to insert parameters into your Query Manager query and form. Using these parameters you can pass information such as report criteria (i.e., customer numbers or date ranges) into your report. You can also use these parameters to pass information into your Query Manager report form when you print the headings and footings, including page headings or footings. The ability to insert parameters or field values into headings is a great feature because it allows you to show these inserted values right on the report. We'll get into exactly how to do this a little later in this chapter. Let's start by going over the various ways to create Query Manager queries.

There are three ways to create a report using QM/400:

- Convert a Query/400 QUERY definition
- Use STRQM command to display the QM/400 main menu
- Create a Query Manager query, form a source member, and compile it.

Which of these routes you take greatly depends on your starting point. For example, if you had an existing query definition that was similar to the report that you want to generate with QM/400, you would convert that query definition to a Query Manager query and form. On the other hand if you were generating the report completely from scratch, you would probably want to use the QM/400 menu. In most cases you won't want to use the third option because creating this source can be tedious and doesn't have any advantages over using the QM/400 menu.
To start, let's examine converting a Query/400 query definition into QM/400. This process involves splitting the query into a Query Manager query and a Query Manager report form. Before we can do this we need to create source files to contain the Query Manager query and form source members. Use the Create Source Physical File (CRTSRCPF) command and create two source files, one named QMQRYSRC, which will be used to store the query source, and QMFORMSRC, which will be used to store the form source.

CRTSRCPF FILE(LIB/QMQRYSRC)
CRTSRCPF FILE(LIB/QMFORMSRC)

We need to create these files because to convert a Query/400 query definition into QM/400 we must first build the source for the query and form from the query definition.

To create the source for the Query Manager query we'll use the Retrieve Query Manager Query (RTVQMQRY) command by entering the following command:

RTVQMQRY QMQRY(library/queryname) SRCFILE(library/QMQRYSRC) ALWQRYDFN(*YES)

The key parameter here is the Allow Information from a Query Definition (ALWQRYDFN) parameter, which enables the command to derive the Query Manager query source from a Query/400 query definition.

To build the second part of the puzzle, the Query Manager report form, we must use the Retrieve Query Manager Report Form (RTVQMFORM) command. This command is very similar to RTVQMQRY and is executed by entering:

RTVQMFORM QMFORM(library/queryname) SRCFILE(library/QMFORMSRC) ALWQRYDFN(*YES)

Once we have retrieved this source, we must create the QM/400 objects by compiling these source members. We create the Query Manager query by using the Create Query Manager Query (CRTQMQRY) command by typing:
Finally, we use the Create Query Manager Report Form (CRTQMFORM) command to compile to form source by typing:

```
CRTQMFORM QMFORM(library/qmform) SRCFILE(library/QQMFOMSRC)
```

At this point you may be thinking that this seems like a lot of work just to convert our query definition into QM/400, but when you see what you can do with QM/400 you'll see why it really is worth the effort.

**Query Manager Queries**

To work with the QM/400 objects, we must display the QM/400 main menu. This is accessed by typing the Start Query Manager/400 (STRQM) command, which has no parameters. Figure 1.1 shows the Query Manager/400 main menu.

![Query Manager main menu](image)

*Figure 1.1: Use the Query Manager main menu to create queries and reports.*

Option 1 on this menu is used to work with Query Manager queries. Option 2 allows you to work with Query Manager report forms. Option 3 allows you to create or modify tables to be used within QM/400. Option 10 on this menu allows...
you define the level of access that individual user profiles have. Enter option 1 to display the Work with Query Manager Queries screen shown in Figure 1.2.

There are two different styles of query that can be created from this screen, prompted and SQL. A prompted query is defined using screens similar to those used to define a query in Query/400. Selecting Specify Tables from the Define Prompted Query screen allows you to select which database files will be used within your query and how they will be joined. The Define Expressions screen is used to create fields within your query whose values are calculated based on fields from your database files, for example, calculating extended value based on unit cost times units sold. Select and Sequence Columns allow you to choose which fields from your database files will be included in the query and in what order they are to appear. The Select Rows screen lets you specify which records to include in your query. Select Summary Functions is used when grouping records to define which type of summary (AVERAGE, SUM, COUNT, etc.) should be used on the columns of your query.

The next option on the Define Prompted Query screen is not an option that is available in Query/400. The Specify Duplicate Rows option allows you to tell Query Manager how to handle duplicate records within your query—that is,
those of the fields selected for your query that have identical values on two or more rows. The options are

1. Keep duplicate rows
2. Keep only the first copy of each row.

To define collating sequences for the data in your query, use the Define Sort Sequences screen. The Define Report Formatting screen allows you to state which Query Manager report form is to be used with this query. In general, prompted queries are good for someone who is not familiar with SQL programming or someone who is looking to get their feet wet in Query Manager/400. Prompted queries do not, however, allow you to use parameters.

SQL queries, on the other hand, are created using standard SQL code. For example:

```
SELECT CUSNAM, CUSAD1, CUSAD2, CUSCTY, CUSSTE, CUSZIP
FROM CUSTMAST WHERE CUSSTE='PA'
```

would display list the name and address fields for all customers in Pennsylvania. The fact that this type of query uses the power of SQL, in addition to being able to insert parameters into your SQL code, makes SQL queries much more flexible than prompted queries. The mode you choose to use to create your queries can be changed from the Work with Query Manager Queries display by pressing F19. The default mode for each user is set in the Query Manager profile, which is accessed through option 10 on the Query Manager main menu.

Table 1.1 shows a listing of each of the functions on the Work with Query Manager Queries screen and gives descriptions of their use.
Table 1.1: Work with Query Manager Queries Options

<table>
<thead>
<tr>
<th>Option #</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Create a new query using the current mode as shown on the top of the screen.</td>
</tr>
<tr>
<td>2</td>
<td>Modify an existing query. The mode used is based on the mode that was used to last save the query.</td>
</tr>
<tr>
<td>3</td>
<td>Create a copy of the selected query.</td>
</tr>
<tr>
<td>4</td>
<td>Delete an existing query.</td>
</tr>
<tr>
<td>5</td>
<td>Display the query definition using the mode in which the query was last saved.</td>
</tr>
<tr>
<td>6</td>
<td>Print a copy of the query’s SQL statement to a printer.</td>
</tr>
<tr>
<td>7</td>
<td>Rename the selected QM query object.</td>
</tr>
<tr>
<td>9</td>
<td>Run the selected query after displaying a set of options for the query run.</td>
</tr>
<tr>
<td>10</td>
<td>Convert a prompted query to an SQL query.</td>
</tr>
</tbody>
</table>

You can convert a prompted query into an SQL query from the Work with Query Manager Queries screen using option 10. It’s important to note that this process cannot be reversed and there is no way to convert an SQL query into a prompted query. Also Query Manager queries created from source retrieved from a Query/400 query definition will always be SQL queries.

Inserting parameters into your SQL statements gives you the ability to create SQL queries that are fed their selection criteria each time they are run. This is as simple as replacing a constant in the WHERE clause with a variable name preceded with an ampersand (&) sign. The above example would look like this:

```
SELECT CUSNAM, CUSAD1, CUSAD2, CUSCTY, CUSSTE, CUSZIP
FROM CUSTMAST WHERE CUSSTE=&STATE
```

When the query is run you can supply the value on the Start Query Manager Query (STRQMQRQY) command using the Set Variable (SETVAR) parameter. In our earlier example this would be:

```
STRQMQRQY CUSTLIST SETVAR(STATE 'PA')
```
If no SETVAR parameter is used, the user is prompted to enter the value.

When the variable used is being compared to an alphanumeric field, the value fed into the variable must contain the required value enclosed in single quotes. If quotes are not used, Query Manager will misinterpret the value as a field name and an error will be returned. This also means that you can insert field names or entire portions of the SQL code as variables. You could replace the entire WHERE clause above with a variable and build the WHERE clause dynamically at run time. This allows you to create generic Query Manager queries and allow a CL application to control which data is selected for your report.

To do this using our previous example, we replace the entire WHERE clause with the variable &WHERE and then use the CL code in Figure 1.3. In this program, if only one parameter is fed in, the program creates the WHERE clause only for that field. If multiple values are sent in, the program combines them within the WHERE clause using AND.

```cl
/*============================================================*/
/* To compile:                                                */
/*                                                            */
/* CRTCLPGM PGM(XXX/CLS001CL) SRCFILE(XXX/QCLSRC)             */
/*                                                            */
/*============================================================*/
CLS001CL: PGM PARM(&STATE &ZIP &GROUP)

DCL VAR(&STATE) TYPE(*CHAR) LEN(2)
DCL VAR(&ZIP) TYPE(*CHAR) LEN(5)
DCL VAR(&GROUP) TYPE(*CHAR) LEN(10)
DCL VAR(&WHEREC) TYPE(*CHAR) LEN(128)

IF COND(&STATE' ') THEN(DO)
    CHGVAR VAR(&WHEREC) VALUE('WHERE CUSSTE''' || &STATE || ''')
ENDDO

IF COND(&GROUP' ') THEN(DO)
    IF COND(&WHEREC'') THEN(DO)
        Figure 1.3: This is how you would create a query management WHERE clause in a CL Program (part 1 of 2).
    ENDDO
ENDDO
```
Finally, after building our entire WHERE clause, we run the query using the STRQMRQRY command with the SETVAR parameter setting the variable WHERE to the value of the field &WHEREC. If no parameters are populated when the program is run, the program leaves a blank WHERE clause, which causes the QM query to display all records for the file.

A standard SQL statement takes the format:

```
SELECT CUSNAM FROM CUSTMAST WHERE CUSNO=1234
```

One of the concepts behind the SQL language is to be able to get access to a database through “plain English” queries. This makes it easy to recognize that, in the example given earlier, we want to select the customer name field from the customer master table for any records where the customer number is equal to 1234. Additional optional clauses may include the GROUP BY clause, which is used to define record grouping, and the ORDER BY clause, which is used to define record sorting. Although we are not going to spend a lot of time going over SQL functions, we are going to spend a little time examining the CASE function.
The CASE function allows you to define a field's value conditionally. For example, this powerful function gives you the ability to avoid “divide-by-zero” errors on calculated fields using the CASE function as shown here:

\[
(CASE \text{ WHEN } x = 0 \text{ THEN } 0 \text{ ELSE } y/x \text{ END})
\]

This example returns 0 as the result if \( x \) is 0, otherwise the result will be \( y \) divided by \( x \). The CASE operator is also extremely useful in a situation where, for example, you want to summarize sales data into columns for a specific period and year to date. Figure 1.4 shows the SQL code that would be required to do this:

```sql
SELECT ITEMNO, SLSPER, SLSYER, SUM(CASE WHEN SLSPER = &PERD THEN SLSAMT ELSE 0 END) AS PERD_SALES, SUM(CASE WHEN (SLSPER <= &PERD AND SLSYER = &YEAR) THEN SLSAMT ELSE 0 END) AS YTD_SALES
FROM SALESFILE
WHERE SLSPER <= &PERD AND SLSYER = &YEAR
GROUP BY ITEMNO, SLSYER, SLSPER
ORDER BY ITEMNO, SLSYER, SLSPER
```

Figure 1.4: This SQL code creates columns based on data values using the CASE function.

Once again we use parameters to define the period and year for our query. Our selected period field is defined using both the SUM and CASE functions to only include data for the required period and then summarize the data by item number. Our year-to-date sales field again uses the SUM and CASE functions, however, this time we select anything up to and including the selected period. You'll also notice that we are using the &PERD and &YEAR variables in the WHERE clause, which prevents the unnecessary reading of records not required by the query.

This also prevents the query from displaying rows where both the current period and year-to-date values are zero. When we create a Query Manager report form to use along with this query, we can insert the values of the SLSPER and SLSYER values into the page headings. A sample of the page-heading screen for the Query Manager report form required to do this is shown in Figure 1.5. We have used &2
to refer to the period number, which is found in column two of our query, and &3 to refer to the year field found in column three. A sample of the resulting output can be seen in Figure 1.6.

![Figure 1.5: Values from a query can be inserted into the column heading as shown here.](image)

![Figure 1.6: When the report is printed the values are inserted into the page heading.](image)