Sinking in a Query/400 Quagmire?

Use Print_Query_Definition to assist in your database modernization efforts

As published June 2011 by IBM Systems Magazine

Written by Gene Cobb – cobbg@us.ibm.com

If you are like many customers, you may have accumulated hundreds, perhaps thousands of Query/400 objects over the years. While Query/400 has been a useful and popular tool, its use does seem to result in a large of number reports that tend to linger around and often require maintenance. I recently met a customer at COMMON who claimed to have over 20,000 such objects! There are various reasons for this:

- Many customers create new, ad-hoc queries on the fly to perform quick analysis of their data, then save that query "just in case" they might need it later. Often they forget about it and create an identical or similar one a short time later.
- Often the accumulation occurs as a result of satisfying various, yet similar end user reporting requirements. Based on the user's specifications, the query developer creates a report that is quite satisfactory – but days later the user requests another one that has a slightly different twist to it (such as sorted by a different column or filtered to only show data for a specific column value). The next week a user in a different department requests the same query but one that accesses a file in a differently library. And on and on it goes…

Whatever the reasons, over time you suddenly realize that your collection of Query/400 objects has become a sizable one that is difficult to manage!

In addition to its propensity to result in redundant objects, Query/400 has some major drawbacks from a technological standpoint:

- For database access, it uses the Query API interface to DB2 for i and not SQL. This means that all Query/400 requests are processed by the Classic Query Engine (CQE) and not the engine designed specifically for SQL access – the SQL Query Engine (SQE). This almost always equates to inferior performance because CQE does not have the optimization technology built into SQE. It also means less sophisticated database collection and analysis tools.
- It does not offer modern interfaces (such as web browser) for development activities and actually running the queries
- It does not offer modern output formats such as PDF, Excel, and HTML.
- Because it does not use SQL, it cannot take advantage of the seemingly endless number of features built into this industry database language. Common Table Expressions, stored procedures, sub-selects, unions, intersections, system and user defined functions, OmniFind text search server, grouping sets and cube/rollup functions are all examples of the power that is built into SQL – none of which can be directly leveraged by Query/400. This is why you see so many examples of "query chaining" – where the report developer must write multiple Query/400 objects and chain them together in a CL program. In most cases, the same request could be satisfied by a single SQL statement! Moreover, since it would be processed by SQE, it will likely run much faster!

All of these are reasons to consider utilizing a different way to analyze your data. What are your options here? Well, you could use SQL if you are proficient or comfortable using this language. This would allow you to leverage SQE but you would still need to incorporate some additional process to get the data into the desired format such as PDF document or spreadsheet. Or you could employ tools (such as DB2 Web Query) that generate SQL for you AND have built-in features to send the data directly into a spreadsheet.

But with so much business logic and years of effort built into these Query/400 objects, customers are justifiably reluctant to throw this investment out the window and start over. So how does one leverage this investment and move to an SQL based solution? And where does one start with this venture? The obvious first step is a thorough understanding of what your Query/400 objects are doing: information such as what files they are accessing, what columns are defined in the reports, what the sort fields are, and what are the break levels are key to gaining this understanding. But how does one do this in an efficient manner without manually opening and interrogating each and every individual query?

Enter the Print_Query_Definition stored procedure. Introduced in V5R3, this little gem has flown under the radar and most customers are not even aware of it. If you have ever used WRKQRY option 6, the stored procedure effectively does the same thing – it prints all of the known information about the query to a spooled file. The advantage of having it a stored procedure form is that it can be called programmatically. This means you can build some logic around it so that it can be called in a loop for every Query/400 object in a library. You can then use the CPYSPLF command to copy the data from each spooled file to a database file (from which you can do your own extraction and analysis)

This stored procedure has 3 parameters:

- Query_library - The name of the library that contains the query
- Query_name - The name of the query
- Language_option
    - 0 – The printed results are translated based on the first MRI library in the library list.
    - 1 – The results are printed in English

Considerations:

- Print_Query_Definition is a stored procedure and as such must be called from SQL interfaces such as STRSQL, Run SQL Scripts window in System i Navigator, or embedded SQL in RPG program.
- The names must be specified in the correct case. The names must be passed in upper case unless the names are delimited.

Here is an example invocation of the stored procedure that prints the definition for the query object named REVGPFTQRY in library QWQCENT:

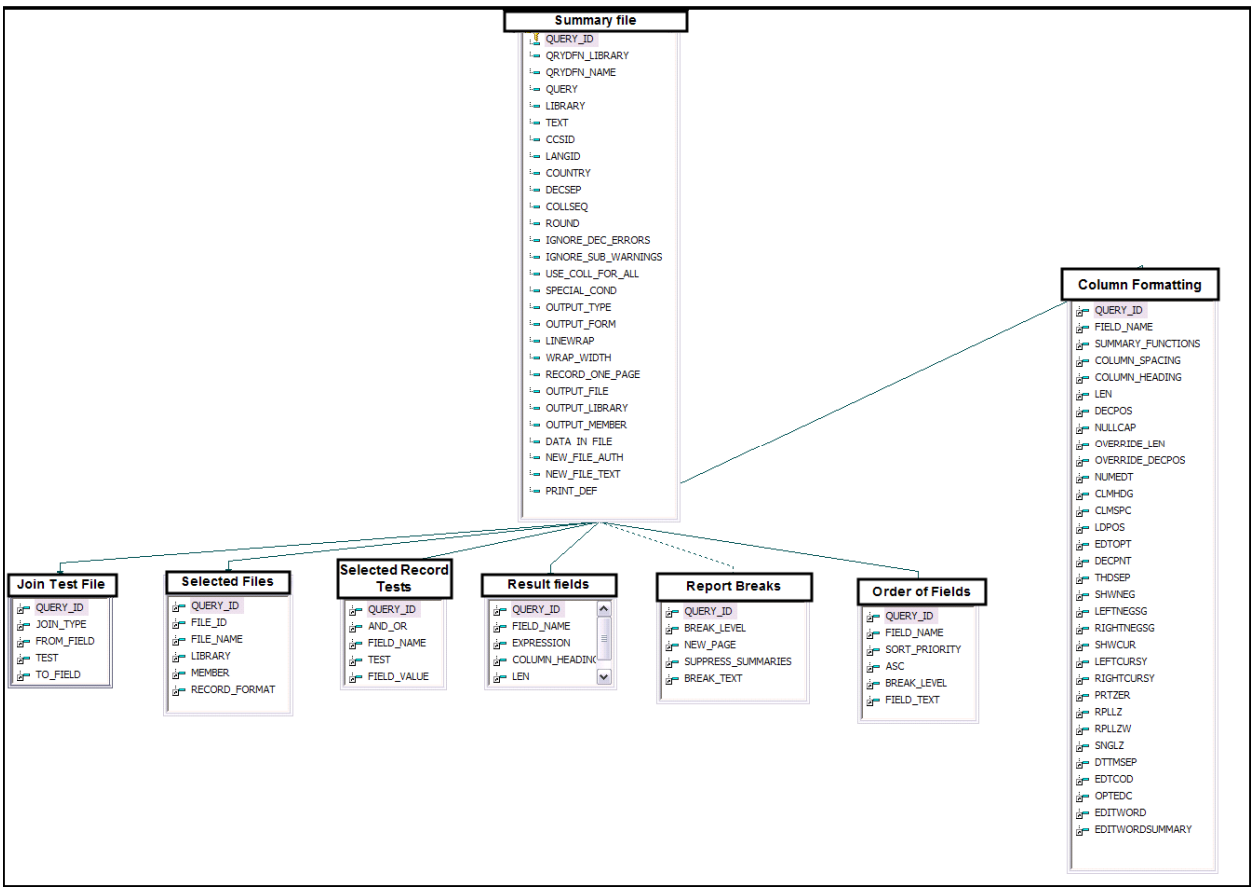
```
CALL QSYS2/Print_Query_Definition('QWQCENT','REVGPFTQRY',1)
```

The result of this invocation is the spooled file shown below.

```
       V6R1M0  080215         IBM Query for iSeries        LP12UT21  5/11/11   10:36:52        Page    1
  Query . . . . . . . . . . . . . . . . REVGPFTQRY
    Library . . . . . . . . . . . . . . QWQCENT
  Query text  . . . . . . . . . . . . .
  Query CCSID . . . . . . . . . . . . .    37
  Query language id . . . . . . . . . . ENU
  Query country or region id  . . . . . US
  Collating sequence  . . . . . . . . . Hexadecimal
  Processing options
    Use rounding  . . . . . . . . . . . Yes (default)
    Ignore decimal data errors  . . . . . No (default)
    Ignore substitution warnings  . . . . Yes
    Use collating for all compares  . . . Yes
  Special conditions
    *** All records selected by default ***
Selected files
  ID     File          Library       Member        Record Format
  T01    ORDERS        QWQCENT       *FIRST        ORDERS
  T02    INVENTORY     QWQCENT       *FIRST        INVENTORY
Join tests
  Type of join  . . . . . . . . . . . . Matched records
  Field              Test               Field
  T01.PROD_NUM       EQ                 T02.PROD_NUM
Result fields
  Name        Expression                        Column Heading          Len  Dec
  GROSSPROFT  lineprice - line_cogs             Gross Profit
  REVENUE     lineprice                         Revenue
Ordering of selected fields
  Field           Sort      Ascending/  Break  Field
  Name            Priority  Descending  Level  Text
  T02.PRODCAT     10        A           1
  T02.PRODTYPE    20        A           1
  REVENUE
  GROSSPROFT
Report column formatting and summary functions
  Summary functions:  1-Total, 2-Average, 3-Minimum, 4-Maximum, 5-Count                  Overrides
  Field           Summary   Column                      Dec  Null       Dec  Numeric  Clm  Clm  L-D  Edt  Dec  Thd  Shw  Left    Right   Shw  Left    Right   Prt  Rpl  Rpl  Sng  DtTm  Edt  Opt  Edit
  Name            Functions  Spacing  Column Headings         Len  Pos  Cap   Len  Pos  Editing  Hdg  Spc  Pos  Opt  Pnt  Sep  Neg  Neg_Sg  Neg_Sg  Cur  Cur_Sy  Cur_Sy  Zer  LZ   LZW  LZ   Sep   Cod  EdC  Word
  T02.PRODCAT                0        Product Category        30       Y     15                          Yes
  T02.PRODTYPE               2        Product Type            15       Y
  REVENUE         1          2        Revenue                 20    2  Y     15    2                     Yes
  GROSSPROFT      1          2        Gross Profit            21    2  Y     12    2                     Yes
Report breaks
  Break  New  Suppress   Break
  Level  Page  Summaries  Text
  0      No    Yes        FINAL TOTALS
  1      No    No
Selected output attributes
  Output type . . . . . . . . . . . . . Display
  Form of output  . . . . . . . . . . . Summary only
  Line wrapping . . . . . . . . . . . . No
              * * * * *   E N D   O F   Q U E R Y   P R I N T   * * * * *
```

While the information in the spooled file may be exactly what you are looking for, chances are you would prefer to have it stored in a structured set of database tables where the information could be more easily harvested and analyzed. As mentioned, you could do this yourself by creating the table and using the CPYSPLF command. But since the data in the spooled file has various bits of information in different formats, it is not structured. Consequently you would need to create various tables to hold the

different formats and a program to extract and parse the information and direct it to the appropriate structured table.. No time or inclination to do this? Never fear – IBM STG Lab Services is here! This branch of the IBM development lab has created a utility known as the "Query/400 Discovery Tool." This utility accepts a library name as an input parameter (including an option for all libraries), utilizes Print_Query_Definition to print information about all Query/400 objects in the specified library, parses the output, and sends the extracted data to one of 8 structured tables. The tables are joined together by a QUERY_ID column – which is a unique identifier for each Query/400 object. The data model for this tool is shown below.



Once the tool is run and tables populated, you can use your favorite query tool for analyzing the data (and please resist the temptation to use Query/400 for this!!)

For example the following SQL statement could be used to analyze all of the files that are referenced in your Query/400 definitions:

```
SELECT
      T1.QRYDFN_LIBRARY,
      T1.QRYDFN_NAME,
      T3.FILE_NAME,
      T3.LIBRARY,
      T2.JOIN_TYPE,
```

```
FROM
      QZRDQRYSPF/SUMMARY T1,
      QZRDQRYSPF/JOIN_TESTS T2,
      QZRDQRYSPF/SELECTED_FILES T3
WHERE
      (T2.QUERY_ID = T1.QUERY_ID) AND
      (T3.QUERY_ID = T1.QUERY_ID)
ORDER BY
      T1.QRYDFN_LIBRARY,
      T1.QRYDFN_NAME,
      T3.FILE_NAME,
      T3.LIBRARY
```

An example result of this SQL statement is shown below:

QRYDFN_LIBRARY	QRYDFN_NAME	FILE_NAME	LIBRARY	JOIN_TYPE
DBMSTF	RICHGOLPAK	ALITMMST	DBMSTF	Matched records
		DMCUSMST	DBMSTF	Matched records
		SALESCI	SLSHST	Matched records
JDEQRY	PRUEBAINF	FECHAVEN	PRBQRY	Matched records
		FECJUL	PRBQRY	Matched records
		FSA58069	PRBQRY	Matched records
		FSA5816	PRBQRY	Matched records
		F41002	PRBQRY	Matched records
		F4101	PRBQRY	Matched records
		F41021	PRBQRY	Matched records
		F4108	PRBQRY	Matched records
		RANGOS	PRBQRY	Matched records
	SYLQRY	FECHAVEN	PRBQRY	Matched records
		FECJUL	PRBQRY	Matched records
		FSA58069	PRBQRY	Matched records
		FSA5816	PRBQRY	Matched records
		F41002	PRBQRY	Matched records
		F4101	PRBQRY	Matched records
		F41021	PRBQRY	Matched records
		F4108	PRBQRY	Matched records
		RANGOS	PRBQRY	Matched records

IBM Lab Services developed this offering to help clients assess their Query/400 environment with the idea of MODERNIZING those reports. The value in modernization of your Query/400 reports could be quite significant. For example it could:

- Reduce the number of Query/400 definitions to maintain through reduction of those redundant Query/400 definitions and use of modernized reporting approaches like parameterized reports
- Result in better performance (as mentioned previously Query/400 cannot use many of the advanced query optimization technologies added to DB2 for i over the last 10 years)
- Simplify delivery of reports to end users in the formats that they want most (spreadsheets, graphs/charts, graphical exception reporting, et.al.)

- Automat report distribution
- Establish consistent definition of data meaning across the enterprise
- Result in better application integration

The Lab Services offering combines Print_Query_Definition with the discovery tool which feeds into a recommendation (a roadmap) to get you on your way to the benefits of Query/400 modernization!

If you are interested in learning more about Query/400 modernization offerings from Lab Services, contact Doug Mack at mackd@us.ibm.com.