Leverage DB2 Web Query and SQL Special Registers

Learn how a recent enhancement to DB2 Web Query can improve your query and reporting environment and help you isolate and analyze those queries

Published Friday, 08 April 2011 00:00 by MC Press On-line [Reprinted with permission from iTechnology Manager, published by MC Press, LP; http://www.mcpressonline.com.]

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

In V6R1 of the IBM i operating system, support was added for the specification (and subsequent retrieval) of several new special registers that applications can use to identify what jobs, users, programs, and applications are issuing SQL statements to the DB2 for i engine. Currently, there are five such registers you can use:

- CLIENT_APPLNAME—Client Application Name
- CLIENT_ACCTNG—Client accounting
- CLIENT_PROGRAMID—Client program ID
- CLIENT_USERID—Client User ID
- CLIENT_WRKSTNNAME—Client Workstation

The good news is that you can use these special registers for your own applications. The better news for customers using IBM DB2 Web Query for i is that this product now uses these registers to identify all SQL requests it submits to the database engine. The values specified by DB2 Web Query for each of these special registers are provided in the table below.

SQL Special Register Name	DB2 Web Query Value
CLIENT_USERID	User profile logged into DB2 Web Query that ran the report. For example: COBBG
CLIENT_WRKSTNNAME	Job name, user profile, and job number that handled the DB2 Web Query request. For example: 678182/QWEBQRYADM/TSCOM3
CLIENT_APPLNAME	DB2 WebQuery
CLIENT_ACCTNG	User profile logged into DB2 Web Query that ran the report. For example: COBBBG
CLIENT_PROGRAMID	DB2WBQRY

Note: Special register support in DB2 Web Query requires version 1.1.2 of the 5733QU2 product id and level 2 of the DB2 Web Query group PTF. More information is provided in the Prerequisites section below.

If you are using DB2 Web Query today, you may be thinking, "OK, sounds great. But how can I use this feature?" A couple of ideas jump to the forefront:

- Selective preprocessing of DB2 Web Query requests using QIBM_QSQ_CLI_CONNECT, the CLI connect exit point
- Database analysis of requests coming from DB2 Web Query

Selective Preprocessing of DB2 Web Query Requests

Once the registers are specified by DB2 Web Query, this information can be retrieved in your own programs to determine where the request came from. If you've read my previous articles, you may remember a couple that discuss the use of the QIBM_QSQ_CLI_CONNECT, the CLI connect exit point:

- "Making Better Use of DB2 Web Query with *LIBL"
- "Maximize SQL Query Engine (SQE) Usage of Your DB2 Web Query Reports"

Registration of this exit point means that you can call a user-written program whenever a CLI connect event occurs in server mode (which happens to be every time DB2 Web Query submits an SQL request to DB2). This preprocessing gives you the ability to do some useful things before the report runs, such as dynamically changing the library list or adding a row to an audit table. One of the drawbacks of this exit point is that you can register only one program to it. This means that if you have multiple CLI applications, the exit point will be activated for *all* SQL requests coming from CLI applications running in server mode (not just DB2 Web Query). Consequently, it can be quite difficult to determine where each request is originating from. If you want to change the library list *only* for DB2 Web Query requests (not for an order entry application using CLI), how can you do this? Well, now you can use the information in the special registers! Here's an example snippet from an RPGLE program that uses embedded SQL to make this determination:

```
// First make sure that this is a Web DB2 Query request. If not, exit program
exec sql
   SELECT CURRENT CLIENT_PROGRAMID INTO :pgmID FROM sysibm/sysdummy1;
   if (pgmID <> 'DB2WBQRY');
     return;
   endif;
```

In this particular example, if the request did not originate from DB2 Web Query, the program ends and no further processing occurs. When retrieving the value of the registers, it does not matter what table you query; therefore, it is recommended that you specify one with a single row, such as the SYSIBM/SYSDUMMY1 table (which should exist on all IBM i systems). Simply add this type of conditional logic at the beginning of the exit point program and take the appropriate action: exit or branch off to the necessary program, subprocedure, subroutine, or section of code.

Analyzing Requests from DB2 Web Query

If you want to determine what applications are making SQL requests or even isolate those requests coming from DB2 Web Query to a gauge and/or analyze their impact on database resources, these special registers make it a breeze! You have a couple of options here:

- Plan Cache Snapshot—If you're confident that your requests are being processed by the SQL Query Engine (SQE)—not the Classic Query Engine (CQE)—simply create an SQL plan cache snapshot from System i Navigator. This action will materialize all of the database request information currently in the plan cache and place it into a DB2 table. Once it's in a table, you can query it!

- SQL Performance Monitor—Alternatively, start an SQL performance monitor trace (prior to the execution of the DB2 Web Query reports). This will capture all requests—SQE and CQE alike. So if you are not sure if all your requests are being processed by SQE, the performance monitor would be all encompassing and the safer way to go. Just make sure you turn it off once you have completed your analysis as it can be resource-intensive. While it's running, the monitor will dump all of the collected information into a DB2 table, so just as with the plan cache snapshot, you query the table.

Helpful hint: To reduce the strain on system resources, specify a pre-filter for the performance monitor so that it ONLY collects information from a specific program id. Depending on the IBM i OS level, use one of these commands to specify a pre-filter and only collect DB2 Web Query SQL requests.

For V6R1:

```
STRDBMON OUTFILE(QGPL/DB2WQFILE) OUTMBR(*FIRST *REPLACE) JOB(*ALL/*ALL/*ALL) TYPE(*DETAIL)
COMMENT('FTRCLTPGM(DB2WBQRY)')
```

For V7R1:

```
STRDBMON OUTFILE(QGPL/DB2WQFILE) OUTMBR(*FIRST *REPLACE)JOB(*ALL/*ALL/*ALL) TYPE(*DETAIL)FTRCLTPGM(DB2WBQRY)
```

Regardless of the method chosen, the SQL statement to query the results is basically the same. The following shows each register and corresponding column name of the table containing the collected data.

SQL Special Register Name	Database Monitor/Snapshot Column Name
CLIENT_USERID	QVC3002
CLIENT_WRKSTNNAME	QVC3003
CLIENT_APPLNAME	QVC3001
CLIENT_ACCTNG	QVC3005
CLIENT_PROGRAMID	QVC3006

Use the following statement and specify the name of the table that contains the collected data:

```
SELECT
    qqtime  AS time ,
    qvc3002 AS client_userid ,
    qvc3003 AS client_wkstnname ,
    qvc3001 AS client_applname ,
    qvc3005 AS client_acctng,
    qvc3006 AS client_programid,
    qq1000  AS statement
FROM db2wqfile
WHERE qqrid = 1000
    AND qvc3006 = 'DB2WBQRY'
ORDER BY qqtime;
```

Note: At the risk of stating the obvious, if you started an SQL performance monitor using the pre-filter described previously, it would not be necessary to include this portion in the WHERE clause:
AND qvc3006 = 'DB2WBQRY'
This is because the pre-filter would have only collected information from that program id.

Using the above statement, here is an example result set:

TIME	CLIENT_USERID	CLIENT_WKSTNNAME	CLIENT_APPLNAME	CLIENT_ACCTNG	CLIENT_PROGRAMID	STATEMENT
2011-01-24 16:31:23.308171	COBBG	192665/QWEBQRYADM/TSCOM3	DB2 WebQuery	COBBG	DB2WBQRY	SELECT T3."COUNTRY",T2."PRODUCTCATEGORY", SUM(T1."LINET
2011-01-24 20:16:56.676093	ZHUYU	192675/QWEBQRYADM/TSCOM3	DB2 WebQuery	ZHUYU	DB2WBQRY	SELECT TABLE_SCHEMA,NAME,TYPE,SUBSTR(COALESCE(REMARKS
2011-01-24 20:17:12.525777	ZHUYU	192675/QWEBQRYADM/TSCOM3	DB2 WebQuery	ZHUYU	DB2WBQRY	SELECT A.TABLE_SCHEMA,A.TBNAME,A.NAME,A.NAME,A.COLTYPE
2011-01-24 20:17:13.052713	ZHUYU	192675/QWEBQRYADM/TSCOM3	DB2 WebQuery	ZHUYU	DB2WBQRY	SELECT A.NAME, B.COLNAME, B.COLSEQ, A.UNIQUERULE ,CASE A.U
2011-01-24 20:17:14.142742	ZHUYU	192675/QWEBQRYADM/TSCOM3	DB2 WebQuery	ZHUYU	DB2WBQRY	SELECT A.TDBNAME, A.TBNAME, SUBSTR(A.COLNAME, 1 ,67), B.T
2011-01-24 20:17:15.746692	ZHUYU	192675/QWEBQRYADM/TSCOM3	DB2 WebQuery	ZHUYU	DB2WBQRY	SELECT COALESCE(REMARKS, LABEL) FROM QSYS2/SYSTABLES WH
2011-01-24 20:17:15.746692	ZHUYU	192675/QWEBQRYADM/TSCOM3	DB2 WebQuery	ZHUYU	DB2WBQRY	SELECT A.TABLE_SCHEMA,A.TBNAME,A.NAME,A.NAME,A.COLTYPE
2011-01-24 20:17:15.746692	ZHUYU	192675/QWEBQRYADM/TSCOM3	DB2 WebQuery	ZHUYU	DB2WBQRY	SELECT A.NAME, B.COLNAME, B.COLSEQ, A.UNIQUERULE ,CASE A.U
2011-01-24 20:17:15.746692	ZHUYU	192675/QWEBQRYADM/TSCOM3	DB2 WebQuery	ZHUYU	DB2WBQRY	SELECT A.TDBNAME, A.TBNAME, SUBSTR(A.COLNAME, 1 ,67), B.T
2011-01-24 20:56:38.659902	ZHUYU	192665/QWEBQRYADM/TSCOM3	DB2 WebQuery	ZHUYU	DB2WBQRY	SELECT T4."PRODUCTTYPE", SUM(T1."LINETOTAL") FROM QWQCE
2011-01-25 02:29:58.011843	ZHUYU	192665/QWEBQRYADM/TSCOM3	DB2 WebQuery	ZHUYU	DB2WBQRY	SELECT T4."PRODUCTTYPE", COUNT(*), SUM(T1."LINETOTAL") FR
2011-01-25 06:39:57.218530	ZHUYU	192665/QWEBQRYADM/TSCOM3	DB2 WebQuery	ZHUYU	DB2WBQRY	SELECT T4."PRODUCTTYPE", MAX(T4."PRODUCTTYPE") FROM QW
2011-01-25 06:47:43.117730	ZHUYU	192665/QWEBQRYADM/TSCOM3	DB2 WebQuery	ZHUYU	DB2WBQRY	SELECT T1."PRODUCTNUMBER",T1."ORDERDATE",T1."LINETOTAL"
2011-01-25 06:57:58.181013	ZHUYU	193520/QWEBQRYADM/TSCOM3	DB2 WebQuery	ZHUYU	DB2WBQRY	SELECT T1."PRODUCTNUMBER",T1."ORDERDATE",T1."LINETOTAL"
2011-01-25 07:00:59.724280	ZHUYU	192665/QWEBQRYADM/TSCOM3	DB2 WebQuery	ZHUYU	DB2WBQRY	SELECT T4."PRODUCTTYPE",T3."COUNTRY", SUM(T1."LINETOTAL"
2011-01-25 07:01:00.256057	ZHUYU	193520/QWEBQRYADM/TSCOM3	DB2 WebQuery	ZHUYU	DB2WBQRY	SELECT T4."PRODUCTTYPE",T3."COUNTRY", SUM(T1."LINETOTAL"
2011-01-25 07:03:49.759901	ZHUYU	193520/QWEBQRYADM/TSCOM3	DB2 WebQuery	ZHUYU	DB2WBQRY	SELECT T4."PRODUCTTYPE",T4."PRODUCTCATEGORY", SUM(T1."L

Figure 1: This is the type of result set you can expect.

If you really wanted to analyze a particular query by using the Visual Explain tool, simply show the statements in the snapshot or the SQL performance monitor, find the statement, and select Visual Explain. This process is shown in the examples below.

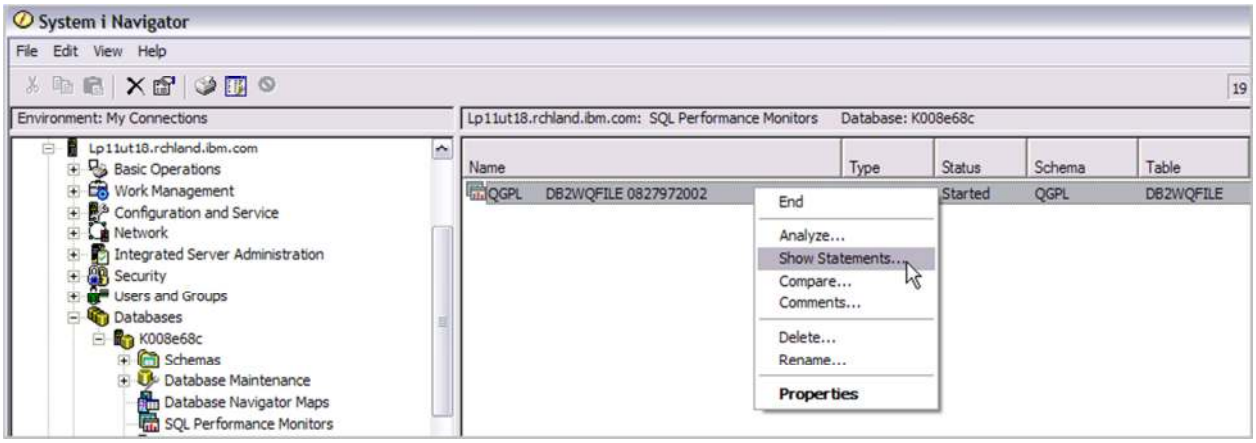

Figure 2: System i Navigator's Visual Explain tool helps you analyze queries.

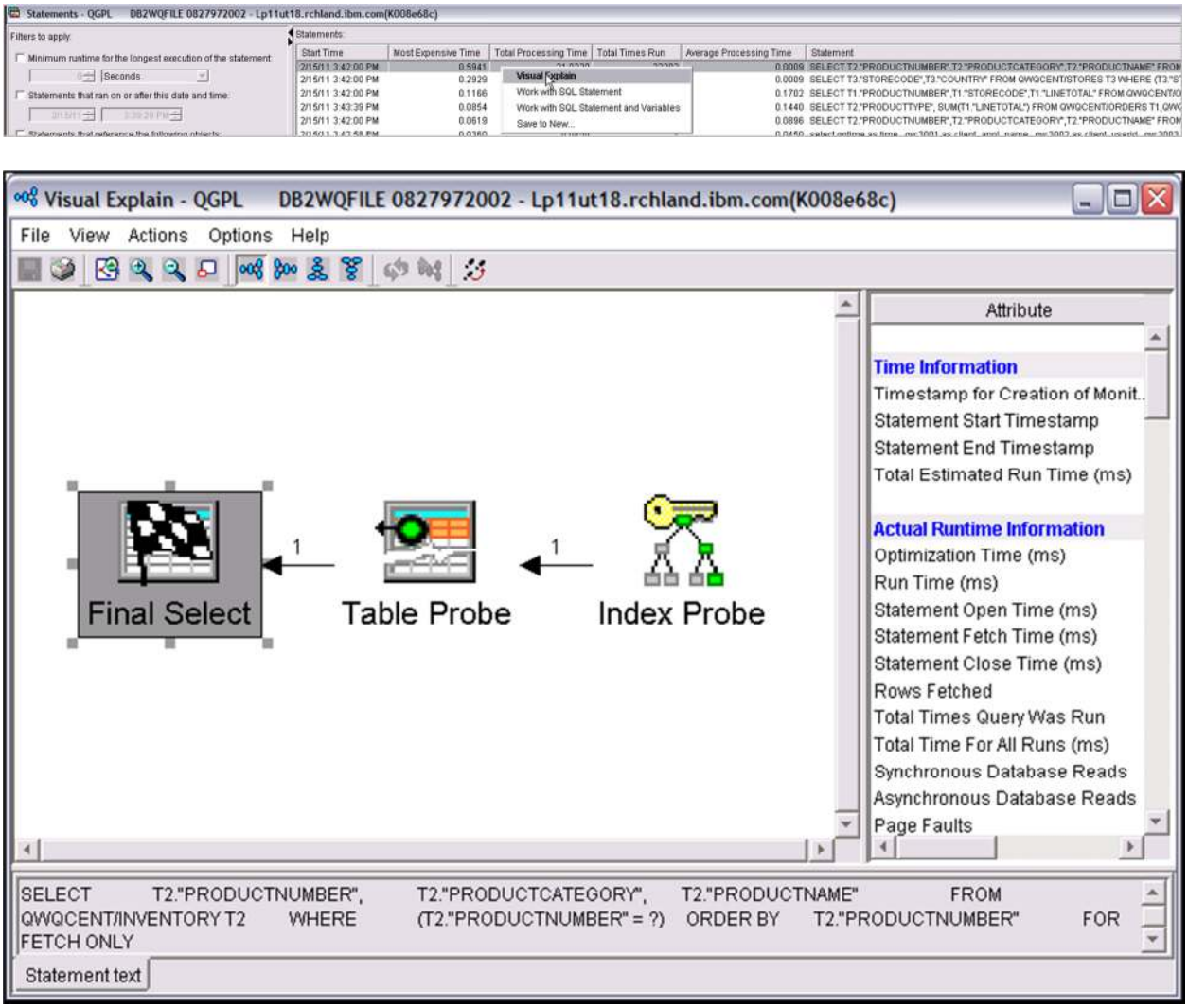

Figure 3: You can drill down for details.

One last idea for querying this data: use DB2 Web Query and create a Web Query report of Web Query requests! An example report is shown below.

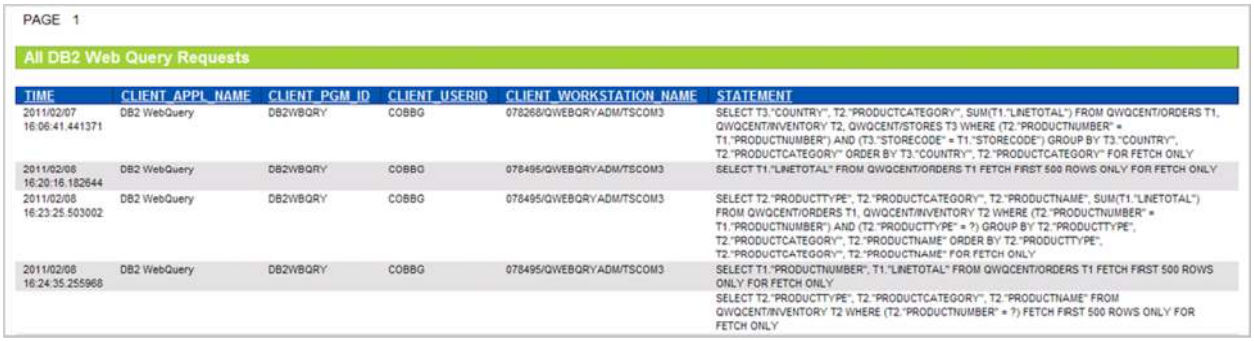

Figure 4: Get reports about queries that have been made.

Prerequisites

As mentioned previously, support for the SQL special registers started with V6R1 of the IBM i operating system. The DB2 Web Query use of these special registers is supported only for version 1.1.2 of 5733QU2 and you must have level 2 or higher of the DB2 Web Query group PTF installed. The group PTFs are different for each IBM i operating system release. The specific information for which PTF you'll need is provided below.

- V6R1 - SF99636
- V7R1 - SF99637

Register Your Own Applications!

Keep in mind that there is no magic going on here: the use of these special registers is not exclusive to DB2 Web Query, IBM i products, or internal processes! If your applications use SQL to access the data and are running on V6R1 of the IBM i operating system, they too can use these special registers to identify where the SQL requests originated from! All you have to do is call the WLM_SET_CLIENT_INFO stored procedure (specifying the values of the registers as input parameters) in your application prior to the execution of your SQL statements (within the same job, of course). Simply follow the format provided in the following example:

```
CALL  SYSPROC.WLM_SET_CLIENT_INFO (
      your_client_userid,
      your_client_wrkstnname,
      your_client_applname,
      your_client_acctng,
      your_client_programid);
```

Note: When using the WLM_SET_CLIENT_INFO stored procedure, you can allow any of the client registers to remain unchanged by specifying NULL for the appropriate parameter.

If you would like to learn more about SQL special registers, refer to these links:

http://www.ibmsystemsmag.com/ibmi/april09/technicalcorner/24804p1.aspx

http://ibmsystemsmag.blogs.com/i_can/database/