Making Better Use of DB2 Web Query with *LIBL

Published Tuesday, 12 August 2008 19:00 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

Search the library list when accessing and querying a referenced database object such as a physical file or SQL view.

In a previous article titled "Maximize SQL Query Engine (SQE) Usage of Your DB2 Web Query Reports," I explained how you could utilize various techniques available on the System i to influence database optimization behavior for DB2 Web Query requests. That article relied upon the use of the QIBM_QSQ_CLI_CONNECT exit program to take a specific (program-defined) action. In this article, I'll show you another way to take advantage of this exit point when using DB2 Web Query.

I've given numerous DB2 Web Query presentations, workshops, and Webinars since the product was first announced early in 2007. A question that is commonly asked is whether DB2 Web Query has the ability to search the library list when accessing and querying a referenced database object such as a physical file or SQL view. Many IBM i shops have identically named and formatted files spread throughout different libraries on the system. Two examples of this could be a state agency that has separate data libraries for each county that they service or a company with different copies of application files in development, quality assurance (QA), and production libraries.

In both of these examples, files with the same name, record format name, and fields exist in each of the libraries. The different versions simply hold data relevant to the library in which they exist (examples of this include an agency in the state of Iowa that has files in a library named WAYNE that contain only data for the residents of Wayne County and a company that has a library named TESTLIB that holds files containing only test data).

User access to the files is controlled by manipulating the library list when the user signs on to the system or any time prior to accessing the files. When a called program or command references a non-qualified file name (one in which the library is not hard-coded), the system will search the library list for the first occurrence of the file with the specified name. The first file that is found in this search is the one that the command or program uses.

This technique, commonly referred to as *LIBL ("star lib list" or "star libl"), is a very popular way for System i shops to customize their application environments so that the correct data is accessed. Consequently, the question that is often asked is whether this library list searching behavior can be

implemented with DB2 Web Query. The answer to this question is "Yes," but it is not the default behavior, so you must take steps to obtain this behavior.

Note: The techniques described in this article work only for DB2 Web Query reports/graphs that access synonyms created against the DB2 CLI adapter. They will not work for imported Query/400 reports or the DB Heritage File adapter.

Creating the Synonym

Before you can report against a table (physical file) or SQL view in DB2 Web Query, you must define this data source to the tool by creating a synonym. These synonyms provide an abstraction layer that contains metadata (information about the file itself). Synonyms are stored in two files located in the IFS directory:

- ACX (access file) contains object-level information about the table/view, such as the file name and library name.
- MAS (master file) contains information about the fields/columns of the file. Column names and attributes can be found in this file.

By default, when you create a synonym, the specified file's library is stored in the ACX file. This tells the tool where to locate the file during report execution. An example of a "Two-part name" is shown in Figure 1.

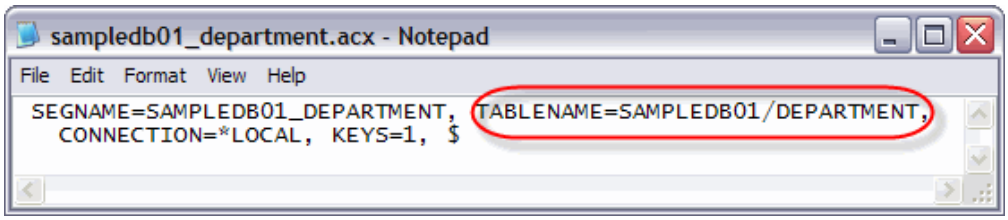

Figure 1: Here's an example of two-part name. (Click images to enlarge.)

However, the specification of the library (SAMPLEDB01 in this example) is not required. If omitted, the ACX file contains a "one-part name" for the underlying table. A one-part name instructs DB2 Web Query to search the library list for the table, just like good old *LIBL!

So how do you do this? By simply selecting the "One-part name" setting on the Create Synonym screen (Figure 2).

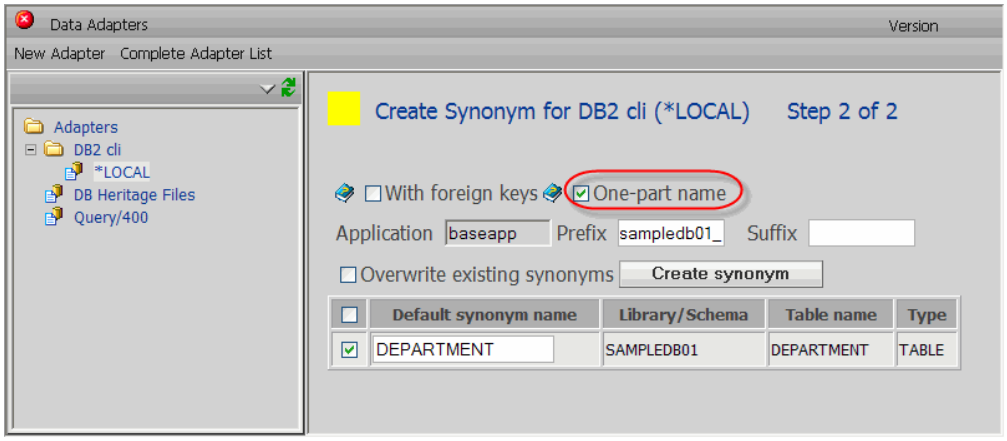

Figure 2: Specify a one-part name.

To change existing synonyms, you could also edit the ACX file and simply remove the library name. An example ACX file with a one-part name is shown in Figure 3.

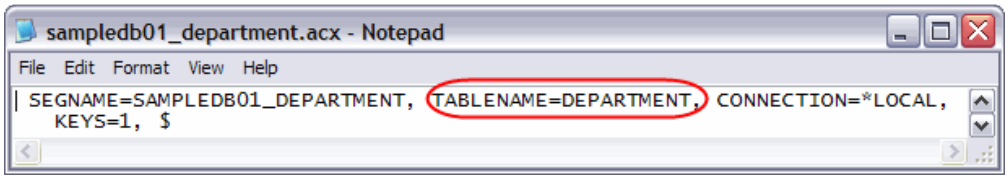

Figure 3: Edit the ACX file to create a one-part name.

Controlling the Library List

So that was the easy part. The more challenging task is actually manipulating the library list of the job that is executing the SQL statement. How do you know which job this is? And how do you access that job and change its runtime attributes (such as its library list)? To fully understand how to do these things, it would help to have some background knowledge of the database access environment employed the DB2 CLI adapter of DB2 Web Query: SQL Server Mode via Call Level Interface (CLI).

When any of the graphical development tools (Report Assistant, Graph Assistant, and Power Painter) are used to create a report or graph, DB2 Web Query generates source code and saves it into a file on the IFS. This file is called a FOCEXEC procedure and contains code written in the WebFOCUS language. It has the file extension of .FEX and is stored in the IFS directory. When you run the report or graph, the following process is carried out:

1. The DB2 Web Query Reporting Server component receives the request and does three things: reads and parses the report's .FEX file and referenced synonym, creates structures necessary for holding the result set that will eventually be returned, and passes this information to the DB2 CLI data adapter for further processing.
2. The DB2 CLI data adapter component analyzes the DB2 Web Query synonym files to obtain information such as table name to access, column names to be used, and connection to be used.

It then optimizes and translates the DB2 Web Query request to the appropriate SQL statement or statements, issues an API to set up SQL Server mode, and issues a CLI API.

3. In SQL Server mode, a pre-start job named QSQSRVR processes the CLI request and submits the specified SQL statement to the DB2 for IBM i engine. These QSQSRVR jobs are the ones that require library list manipulation.

For more information on SQL Server mode, see Scott Forstie's Redbook Technote.

It may seem like there is a lot going on here (and there is), but don't feel overwhelmed; this all happens behind the scenes, and you can tailor the environment to satisfy your requirements. The system provides three ways of changing the attributes of the appropriate QSQSRVR job so that the desired library list can be set up before the database object is accessed. These methods are listed in order of implementation simplicity:

- QUSRLIBL (user library list system value)
- Job descriptions
- Programs

QUSRLIBL

If you do nothing at all, the library list specified in the system value's QUSRLIBL will be used. This is the default library list setting for the job description QGPL/QDFTJOBD (which is the default job description for all newly created user profiles). Because this is the default behavior, it is obviously the easiest to implement. However, it is not very flexible. If you wanted to change the library list, you could do so by issuing this command:

```
CHGSYSVAL SYSVAL(QUSRLIBL) VALUE('LIB1 LIB2. . .')
```

This will change the user portion of the library list for all users set up in this manner. Because it is a global setting, if different users require different library lists, this is not a viable option.

Job Descriptions

By creating different job descriptions with different library lists and changing user profiles to use those job descriptions, you can quickly and easily set up an environment that is flexible and easy to maintain. The QSQSRVR job that processes the request will inherit the library list setting of the requesting user's job description. Consider the following examples:

Example 1:

```
CRTJOBD JOBD(QGPL/COBBG) INLLIBL(STAR1G COBBGLIB QGPL QTEMP)

CHGUSRPRF USRPRF(COBBG) JOBD(QGPL/COBBG)
```

Example 2:

```
CRTJOBD JOBD(QGPL/TEVEN) INLLIBL(STAR100M QTEMP QGPL TEVENLIB)
```

```
CHGUSRPRF USRPRF(TEVEN) JOBD(QGPL/TEVEN)
```

Using the configuration in example 1, when user profile COBBG logs into DB2 Web Query and runs a report that uses a one-part name synonym, the product will look for the underlying table in the libraries in the following order: STAR1G, COBBGLIB, QGPL, and QTEMP.

User profile TEVEN in example 2, on the other hand, can log in and run the same report, but the tool will instead search for the table in the libraries in the following order: STAR100M, QTEMP, QGPL, and TEVENLIB.

Use of the job description object is a very common way to control user library lists. Therefore, you may very well already be using this technique in your application environment today. If so, simply creating your synonyms with the one-part name setting is all you need to do to obtain DB2 Web Query *LIBL behavior.

Programs

In some cases, you may need the ability to manipulate the user's library list based on more complex business requirements, such as performing a series of database lookups or perhaps checking environmental conditions such as the time of day. In such cases, use of the job description is not sufficient; more programmatic control is needed to perform these tasks and update the library list accordingly. Fortunately, there is a way to call a program before the report is actually run. At the beginning of this article, I mentioned use of the CLI connect exit point. Here is where you would use it.

Let's say you have a program named CHGLIBLPGM that performs multiple database lookups and checks conditions such as the fiscal year and quarter to determine what libraries to set up in the current user's library list. It then issues the CHGLIBL CL command to actually change the library list based on these factors. The CLI exit point gives DB2 Web Query the ability to call this program (and use the existing business logic) to update the library list. To do so, issue the following command:

```
ADDEXITPGM EXITPNT(QIBM_QSQ_CLI_CONNECT) FORMAT(CLIC0100) PGMNBR(1)
PGM(QGPL/CHGLIBLPGM)
```

As described earlier, the database access mechanism used by the DB2 CLI adapter is CLI. Every time you run a report that uses a synonym based on this adapter, the configured exit program is called in the QSQSRVR job that is running the SQL request.

In our example, this means that the program named CHGLIBLPGM in library QGPL will be called every time a CLI connect event occurs, which happens every time you run a DB2 Web Query report that uses this adapter. Moreover, this all occurs before the database request actually happens; therefore, CHGLIBLPGM is called and manipulates the library list before any files are accessed. If the synonym that the report is using also has one-part naming in place, the database engine will then search the library list for the first occurrence of the file name specified in the synonym. *LIBL behavior has been obtained!

A Technique for Using the User Profile's Initial Program

So if the QSQSRVR jobs inherit the requesting user's library list, can you use the user profile initial program (INLPGM parameter of CRTUSRPRF command) to set up the library list? Unfortunately, no. When you sign into the system using a 5250 emulation session (green-screen), the program specified in your user profile's INLPGM parameter is executed. Many IBM i shops use the user profile initial program to set up their users' library lists. Unfortunately, initial program execution does not occur in users' DB2 Web Query environment. Even though users log in to DB2 Web Query, the actual initial program in the user profile setting is not executed; thus, the library list cannot be changed this way.

However, some customers have hundreds if not thousands of users whose library lists are manipulated at sign-on by the initial programs specified in their user profiles. Forcing these customers to create job descriptions for each of these users may be too much to ask. Moreover, there may be many different versions of the initial program created and customized for various groups of users. Such an environment would eliminate the use of the CLI connect exit point, since it can only be configured to call a single program. Fortunately, there may be a way you can utilize these existing initial programs to set up the library list in a DB2 Web Query environment. You can write a CL program that retrieves the current user's initial program and dynamically calls that program. This new CL program can then be configured as the CL connect exit point program.

Let's assume one of your existing initial programs looks like this:

```
PGM

ADDLIBLE LIB(COBBG)
ADDLIBLE LIB(DATALIB08)
ADDLIBLE LIB(APPLIB08)
ADDLIBLE LIB(STAR1G)

ENDPGM
```

Note: An initial program like this one is a good candidate for this technique, because it is only setting up the environment (changing the library list), not displaying anything to the screen (or calling other programs that are opening display files). More on this later.

To leverage such an existing initial program, follow these steps:

1. Enter the following source code into a new source file member. It is named CLI_EXIT.

```
PGM         PARM(&USER)

DCL         VAR(&USER) TYPE(*CHAR) LEN(10)
DCL         VAR(&INLPGM) TYPE(*CHAR) LEN(10)
DCL         VAR(&INLPGMLIB) TYPE(*CHAR) LEN(10)

RTVUSRPRF   USRPRF(&USER) INLPGM(&INLPGM) +
              INLPGMLIB(&INLPGMLIB)
IF          COND(&INLPGM *NE *NONE) THEN(DO)
```

```
CALL &INLPGMLIB/&INLPGM
ENDDO

ENDPGM
```

2. Compile this new program into QGPL. You should have a program object named QGPL/CLI_EXIT.
3. Configure the CLI connect exit point to call this program:

```
ADDEXITPGM EXITPNT(QIBM_QSQ_CLI_CONNECT) FORMAT(CLIC0100)
PGMNBR(1) PGM(QGPL/CLI_EXIT)
```

Additional Considerations

Depending on what your initial program actually does, it may require some modification or you may be forced to create a copy of the program that simply performs the library list manipulation commands. For example, if your initial program is interactive in nature and uses screen I/O (display files) to collect information and interact with the user, this technique will not work, because the program is actually called by the QSQSRVR (batch) job, and display files cannot be displayed in a batch mode. When this happens, the diagnostic message CPF4103 (Device &4 not found while opening file &2 in library &3) appears in the QSYSOPR message queue, and the report execution waits until you respond to the inquiry message in QSYSOPR. When you do respond to the inquiry message, the report execution fails and the messages "(FOC1400) Error occurred in SQL Call Level Interface" and "(FOC1406) SQL OPEN CURSOR ERROR" appear where the report normally would. If such a program was configured for the CLI connect exit point, every report that uses a synonym based on the DB2 CLI adapter would fail in this manner. To remedy this, you would need to eliminate the code or called program(s) that perform the I/O operations.

If you do experience problems when implementing the CLI exit point technique, be sure to check for error messages in the QSYSOPR message queue and the job logs of the active QSQSRVR jobs.

As mentioned in the previous exit point article, you may already have a program defined for QIBM_QSQ_CLI_CONNECT. Therefore, you will want to check this before implementing this solution by taking the following steps:

1. Issue the command `WRKREGINF EXITPNT(QIBM_QSQ_CLI_CONNECT)`.
2. From the Work with Registration Information screen, select option 8 to see if a program is already defined for this exit point.

If another program has already been defined, you may have to consolidate your programs to perform all of the desired tasks.

If you implement this technique but later decide you would like to disable it, you can remove the exit point program by issuing this command:

```
RMVEXITPGM EXITPNT(QIBM_QSQ_CLI_CONNECT) FORMAT(CLIC0100) PGMNBR(1)
```

Long Live the Library List

Library lists are a very popular way to allow customers to tailor applications and reporting environments to satisfy their specific business requirements. Using the techniques described here, you can continue to use the library list concept in your DB2 Web Query reports.

For more information on DB2 Web Query, download the IBM Redbook titled Getting Started with DB2 Web Query for System i.