TechTip: Members Only? No Problem with DB2 Web Query

Use DB2 Web Query to access your multiple member files, let your users dynamically select the member they want to query at run time, and create a report that uses the latest and greatest database engine technology

Published Friday, 12 March 2010 01:00 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

Over the last several years, I have written several articles about DB2 for i and DB2 Web Query: some of which emphasized the importance of employing techniques to ensure that your queries are using the SQL Query Engine (SQE), the strategic database engine for DB2 for i. One of the requirements for SQE processing is that the database access method must be SQL. However, one of the very few drawbacks with SQL access on our beloved system is that it does not natively support the concept of physical file members. This means you cannot write an SQL select statement that directly references a specific member. When the file is queried, SQL can only access the first member in the file. This in turn means that the DB2 Web Query CLI adapter (the one that generates SQL to satisfy the query request) can not directly access a member either. If you have DB2 Web Query this can be frustrating because you may have hundreds or even thousands of files that have valuable data tucked away in individual members.

Fortunately, all is not lost. DB2 Web Query does provide another adapter that lets you access specific members: the DB Heritage File. You can use this adapter to configure a DB2 Web Query synonym that references a specific physical file member. Any report that is based on such a synonym will generate a database request using the OPNQRYF command (which does support the direct access of a physical file member). Problem solved right?

Well yes and no. Yes, because this will work. You can create snazzy reports, graphs, and dashboards using DB Heritage File synonyms to effectively reach the data in members. But remember what I said about SQE? It will only be used to process a query if that query is based on an SQL statement. OPNQRYF is not SQL, thus the older Classic Query Engine (CQE) must be used to perform the query workload. While CQE is no slouch when it comes to handling database requests, in most cases, this means a slower running query. It also means you can't take advantage of all the "goodies" that SQE provides such as the SQE plan cache, Maintained Temporary Indexes, and Materialized Query Tables. I could go on and on about the benefits of SQE – but just know this: IBM has not enhanced CQE in a long time and will not be in the future because SQE is the strategic engine.

OK, cheer up – I didn't write this article to put you in a surly mood. Even though SQL does not support the direct access of a specific member, there is a way to "trick" it into doing just that. Ever heard of an SQL Alias? An alias is an alternate name for a table or view. If you were to look up the CREATE ALIAS statement in the SQL reference manual, you would find this:

> *The CREATE ALIAS statement defines an alias on a table, partition of a table, view, or member of a database file at the current server.*

Wait -did you see that word? *Member*? Yes, you actually can create an SQL alias that points to a specific physical file member. You can then reference the alias in an SQL Select statement to gain access to the data in the specific member. To give you an example, let's say you have a multi-member physical file named ORDERS that resides in the QGPL library. ORDERS has several members – one for each year between 2001 and the current year. To create an alias that points to a specific member you would execute an SQL statement like the following:

```
CREATE ALIAS QGPL/ORDERS_MBR FOR QGPL/ORDERS (ORDERS2005);
```

In this example, you created an SQL Alias named ORDERS_MBR that is based on the specific member named ORDERS2005. This means any SQL statement that references that alias (like the one below) will only query the data in the ORDERS2005 member.

```
SELECT * FROM QGPL/ORDERS_MBR
```

Since SQL is used for data access, SQE can be used to handle the request.

In DB2 Web Query you can also use the CLI adapter to create a synonym based on an SQL Alias. This means you can create a DB2 Web Query report that is based on a specific member AND uses SQE! Woohoo!

But wait, we're not done yet. A question that I am often asked is how a DB2 Web Query report developer can provide an interface that allows the end user to select only a specific member to query at run time. While the list of members for a physical file can be obtained from the system catalog SYSPARTITIONSTAT in the QSYS2 schema, the development tools in DB2 Web Query do not provide a facility that gives the end user this kind of interface. But my in my experience with DB2 Web Query, whenever I can't do something with the tools, I often find I can accomplish what I want by writing my own code and integrating it into DB2 Web Query using views and stored procedures. This is another such case. Instead of basing my synonym on the SQL alias, I instead create one that references a stored procedure. The procedure does the underlying "dirty work": it accepts a member name as an input parameter, dynamically creates an alias (in QTEMP) that references the passed in member name, declares a cursor that (when opened) will return all the rows and columns in the alias, and finally, opens the cursor to return the result to DB2 Web Query.

I can also create an SQL view over that SYSPARTITIONSTAT catalog to return the list of members for my physical file. Once a synonym has been created over this view, it can be used as the data source to

populate my report's auto prompt drop-down list. To illustrate this process, let's work through an example.

An example

In the following example, I take you through all the steps required for creating a DB2 Web Query report that prompts the end user for the desired member to query, executes the query, and returns the results to the browser. Oh, and this query uses SQE too. An example of this report is shown in Figure 1.

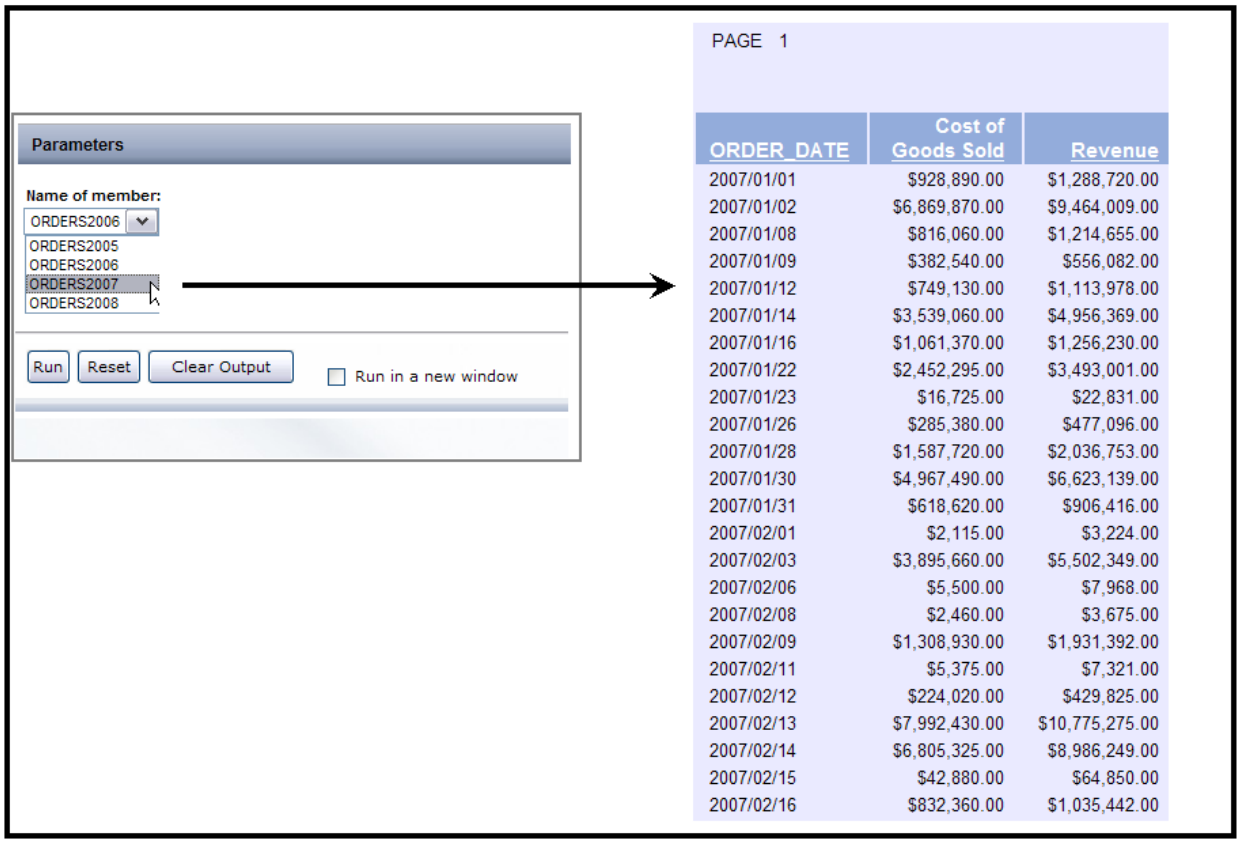


Figure 1

1. First, open an SQL interface (such as Run SQL Scripts from System i Navigator) and create an SQL stored procedure that accepts a member name as an input parameter. This procedure will create an SQL alias that points to the member value passed in. It then returns all rows and columns in the alias (because it opens a cursor based on the SELECT * FROM ORDERS_MBR statement). An example is shown below.

```
CREATE PROCEDURE DYNAMIC_ORDERS_MEMBER

                (IN MEMBERNAME CHAR(10) )
                DYNAMIC RESULT SETS 1
                LANGUAGE SQL
```

```
        MODIFIES SQL DATA
BEGIN
        DECLARE CREATE_ALIAS VARCHAR ( 200 ) ;
        DECLARE C1 CURSOR WITH RETURN TO CALLER FOR
          SELECT *
          FROM QTEMP / ORDERS_MBR ;

        --Take the inputted member name and create an SQL alias
        SET CREATE_ALIAS = 'CREATE ALIAS QTEMP/ORDERS_MBR FOR QGPL/ORDERS (' || MEMBERNAME || ')' ;
        EXECUTE IMMEDIATE CREATE_ALIAS ;
        OPEN C1 ;
END ;
```

2. To test the stored procedure, call it from the SQL interface and make sure it returns the appropriate data.

   CALL QGPL/DYNAMIC_ORDERS_MEMBER ('ORDERS2008');

3. Create a DB2 Web Query synonym over the stored procedure as shown in Figure 2.

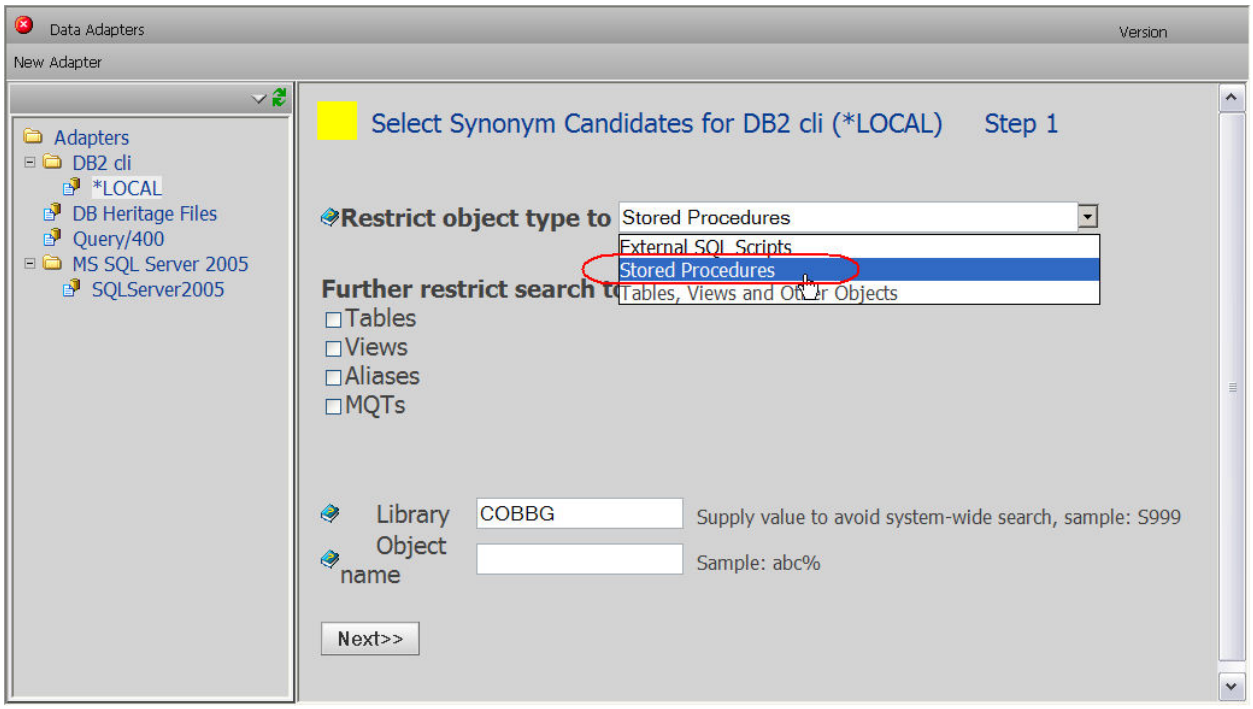


Figure 2

4. Click the Next button to display the Step 2 screen. From here, find and select the stored procedure from the presented list. An example is shown in Figure 3.

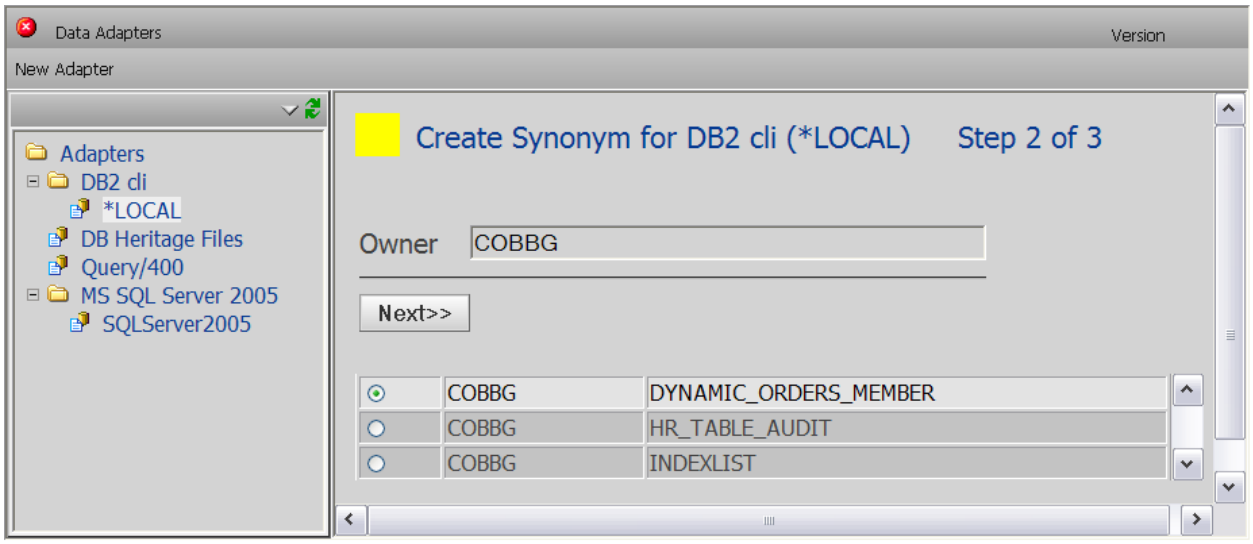

Data Adapters
Version
New Adapter
Adapters
DB2 cli
*LOCAL
DB Heritage Files
Query/400
MS SQL Server 2005
SQLServer2005
Create Synonym for DB2 cli (*LOCAL) Step 2 of 3
Owner
COBBG
Next>>
COBBG
DYNAMIC_ORDERS_MEMBER
COBBG
HR_TABLE_AUDIT
COBBG
INDEXLIST

Figure 3

5. Click the Next button to display the Step 3 screen.

   As shown in Figure 4, specify a valid (member name) value for the input parameter. This is because DB2 Web Query actually runs the stored procedure during this step so that it can generate a sample result set necessary for generating the format (columns) of the synonym. It is also recommended that you supply a suffix of "_SP" to indicate that this is a stored procedure data source. This is not required, but may be useful for your report developers. This will tell them what kind of data source that the synonym is based on.

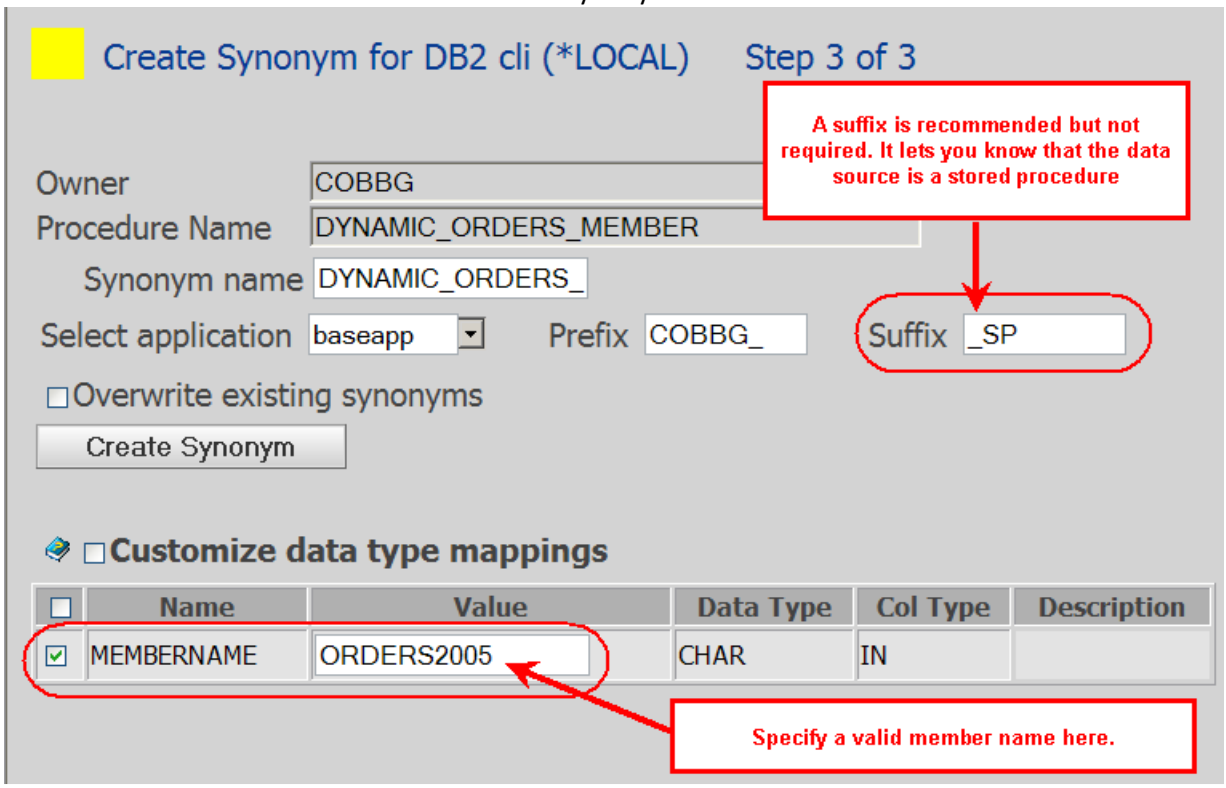


Figure 4

6. Click the Create Synonym button to finish this task and close the browser window.

7. Return to your SQL interface and create an SQL view that returns all the member names for the multi-member file using the system catalog QSYS2/SYSPARTITIONSTAT. This view will be used to populate the drop down list so the user can select which member to query at run time.

```
CREATE VIEW QGPL/SELECT_QAPMDISK_MEMBER AS
SELECT TABLE_PARTITION FROM
QSYS2/SYSPARTITIONSTAT WHERE TABLE_SCHEMA = 'QGPL' AND
TABLE_NAME = 'ORDERS'
```

8. Create a DB2 Web Query synonym over this new SQL view. An example is provided in Figure 5.

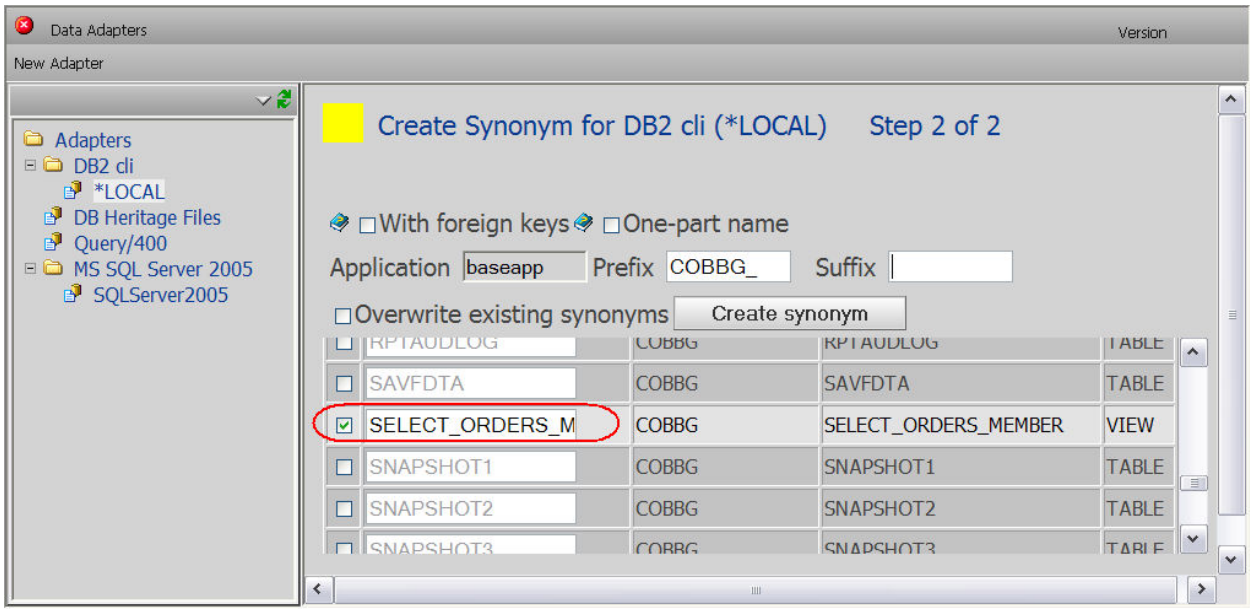


Figure 5

9. Now you are ready to create your report. Open Report Assistant and select the new stored procedure synonym as shown in Figure 6.

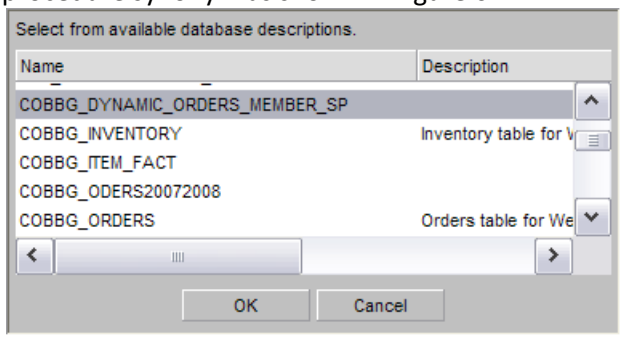


Figure 6

10. Recall that we want to dynamically prompt the users for the member name by allowing them to select from a list of members in a drop down list. To set this up, go to the **Selection Criteria** tab, drag MEMBERNAME from list of **Available fields** to **Screening conditions**, and click the **<Select values>** link. See Figure 7.

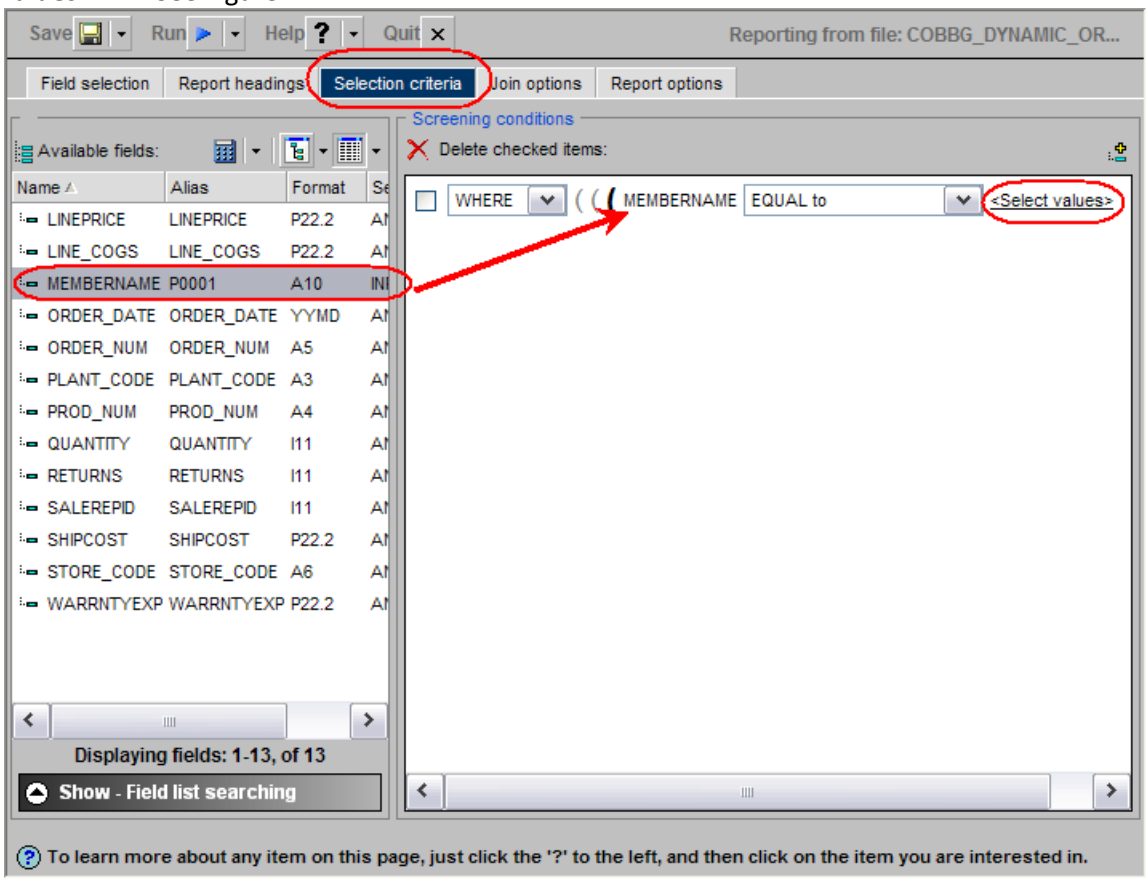


Figure 7

11. As shown in Figure 8, select **Parameter** and click on **Auto Prompt** icon

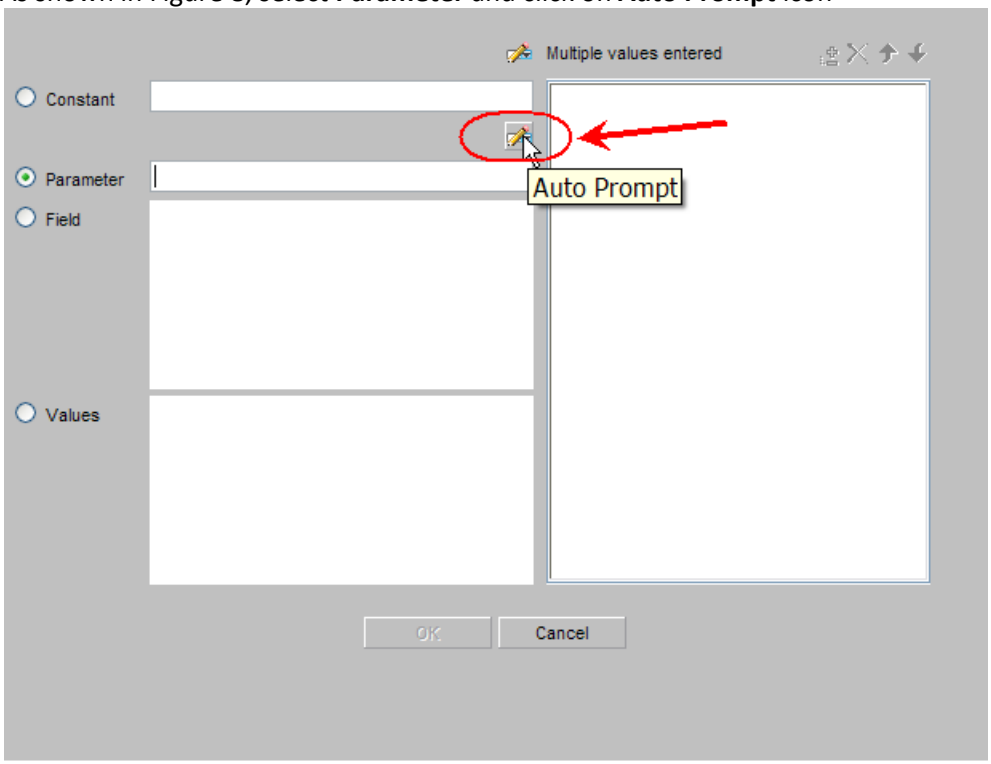


Figure 8

12. From the **Data Source** drop down list, select the synonym of the SQL view created in step 8 and select the **Field** TABLE_PARTITION. This will populate the drop down list with all the member names for the ORDERS file. Figure 9 shows an example of this.

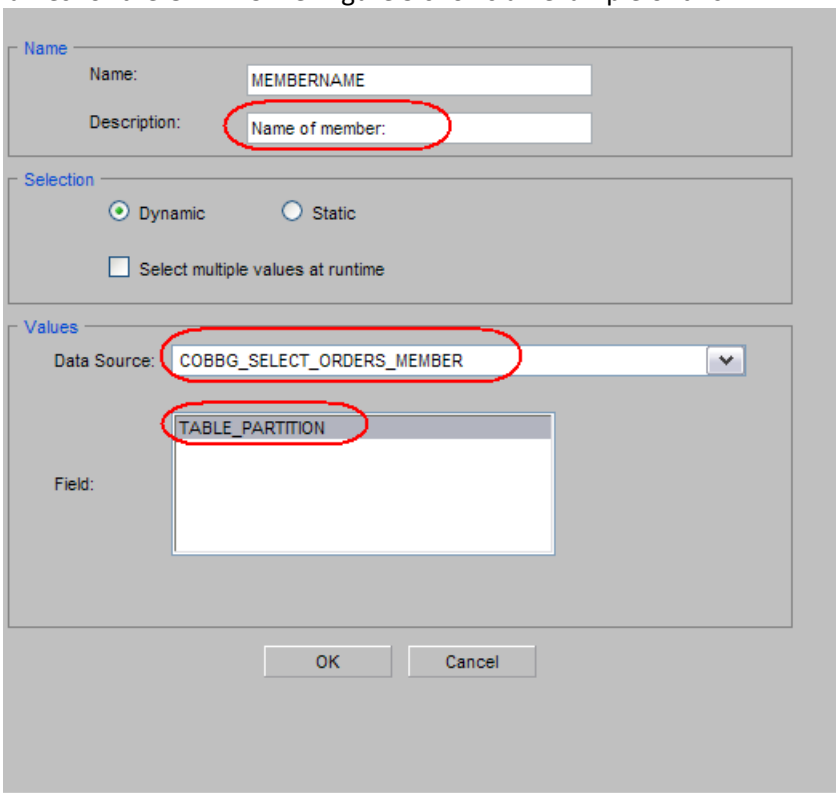


Figure 9

13. Click OK twice.

14. Finish you report definition by specifying Field selection, Report headings, and Report options.

15. Save your report

16. Run your report. As shown in Figure 10, a drop down list is presented to prompt you for the member to query.

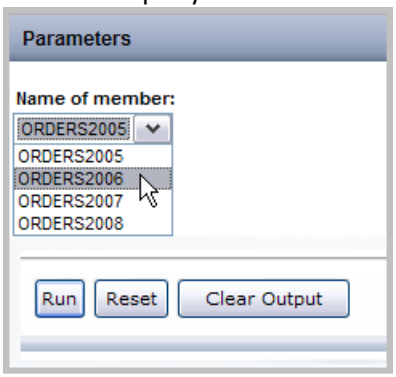
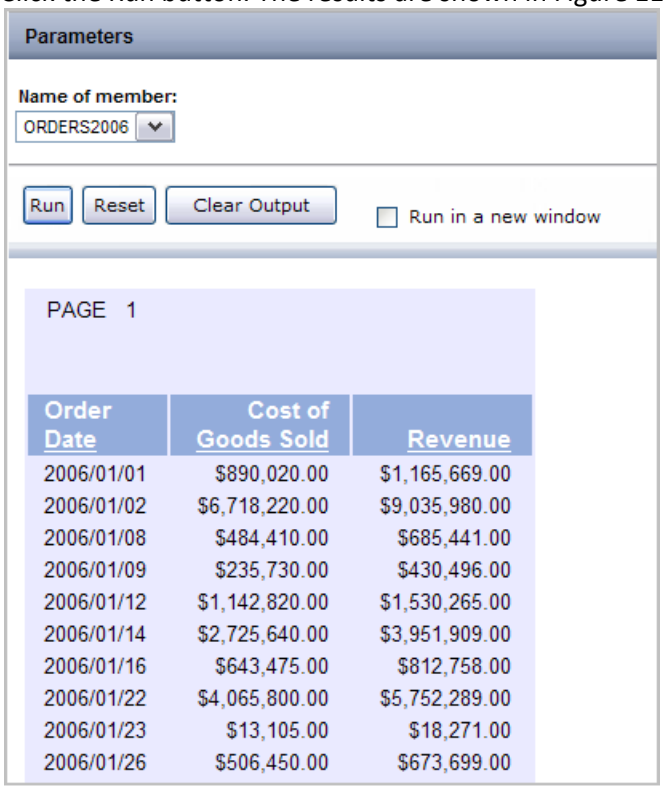


Figure 100

17. Click the Run button. The results are shown in Figure 11.



Order Date	Cost of Goods Sold	Revenue
2006/01/01	$890,020.00	$1,165,669.00
2006/01/02	$6,718,220.00	$9,035,980.00
2006/01/08	$484,410.00	$685,441.00
2006/01/09	$235,730.00	$430,496.00
2006/01/12	$1,142,820.00	$1,530,265.00
2006/01/14	$2,725,640.00	$3,951,909.00
2006/01/16	$643,475.00	$812,758.00
2006/01/22	$4,065,800.00	$5,752,289.00
2006/01/23	$13,105.00	$18,271.00
2006/01/26	$506,450.00	$673,699.00

Figure 111

Note: If you did not want a drop down list for the member names, you would simply specify a parameter name in step 12 and skips step 7,8 and 12 . This would present a text box from which you could type in the member name.

What about joins?

In the above example, all of the columns in my report definition were taken from the single physical file member (or segment). These columns were actually returned from the result set of the stored procedure that was called during report execution. For many reports, you may need to join to other related files in order to get all of the fields needed to satisfy the report requirements. To do this, you could certainly use the **Join Options** tab in Report Assistant to join the stored procedure segment to another file (thereby creating a second segment). The problem with this approach will be report performance. Because the data retrieval for the first join segment is buried in (and performed by) the stored procedure, there is no way for DB2 Web Query to generate a single SQL statement to perform the join. This means that the merging of the two join segments must be performed by the DB2 Web Query Reporting Server. While the performance of this may not be bad for smaller files, you are going to experience a longer response time for larger data sets.

In these situations, I strongly recommend that you specify all of the join syntax in the stored procedure itself. This will result in a single SQL statement being submitted to the DB2 engine. An example of the modified DECLARE CURSOR statement (from the stored procedure) is shown below.

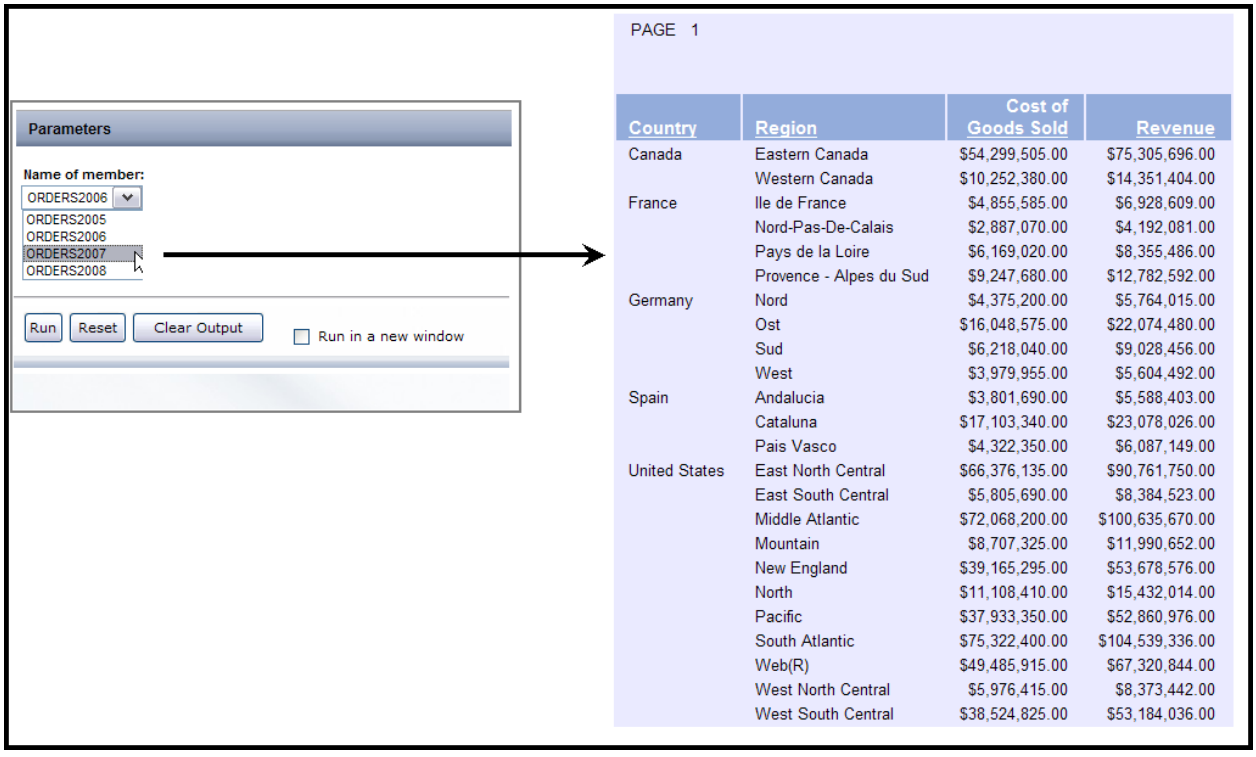
```
DECLARE C1 CURSOR WITH RETURN TO CALLER FOR  SELECT A.*, B.*, C.*, D.*
FROM QTEMP/ORDERS_MBR A
INNER JOIN QWQCENT/STORES B ON A.STORE_CODE = B.STORECODE
INNER JOIN QWQCENT/PLANT C ON A.PLANT_CODE = C.PLANTCODE
INNER JOIN QWQCENT/INVENTORY D ON A.PROD_NUM = D.PRODUCTNUMBER;
```

Specifying the join in the stored procedure has several desirable effects. Here are a couple of them:

1. The report developer does not need to intimate with the data model and know what files to join and what the correct join columns are.

2. The workload of processing the join logic will be pushed down to the database engine and will result in a faster running report.

An example report that is based on columns from multiple join segments (returned from a stored procedure) is shown in Figure 12.



Country	Region	Cost of Goods Sold	Revenue
Canada	Eastern Canada	$54,299,505.00	$75,305,696.00
	Western Canada	$10,252,380.00	$14,351,404.00
France	Ile de France	$4,855,585.00	$6,928,609.00
	Nord-Pas-De-Calais	$2,887,070.00	$4,192,081.00
	Pays de la Loire	$6,169,020.00	$8,355,486.00
	Provence - Alpes du Sud	$9,247,680.00	$12,782,592.00
Germany	Nord	$4,375,200.00	$5,764,015.00
	Ost	$16,048,575.00	$22,074,480.00
	Sud	$6,218,040.00	$9,028,456.00
	West	$3,979,955.00	$5,604,492.00
Spain	Andalucia	$3,801,690.00	$5,588,403.00
	Cataluna	$17,103,340.00	$23,078,026.00
	Pais Vasco	$4,322,350.00	$6,087,149.00
United States	East North Central	$66,376,135.00	$90,761,750.00
	East South Central	$5,805,690.00	$8,384,523.00
	Middle Atlantic	$72,068,200.00	$100,635,670.00
	Mountain	$8,707,325.00	$11,990,652.00
	New England	$39,165,295.00	$53,678,576.00
	North	$11,108,410.00	$15,432,014.00
	Pacific	$37,933,350.00	$52,860,976.00
	South Atlantic	$75,322,400.00	$104,539,336.00
	Web(R)	$49,485,915.00	$67,320,844.00
	West North Central	$5,976,415.00	$8,373,442.00
	West South Central	$38,524,825.00	$53,184,036.00

Figure 122

Membership need not be exclusive!

If your company has been running on the IBM i platform for awhile, chances are good that you have some multi-member physical files in your database. It is a very useful and common implementation – in fact almost every customer I talk to has them in their shop. As such, there are many that absolutely need to pull valuable business data from this type of data source. Armed with this technique, you can now use SQL and DB2 Web Query to extract this information, provide an interface that let's your end users access the data they need, AND enjoy all the benefits of SQE processing. Membership does have its privileges!