Maximize SQL Query Engine (SQE) Usage of Your DB2 Web Query Reports

Published Tuesday, 08 January 2008 18:00 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

Ensure top-notch DB2 Web Query reports by using QAQQINI, exit points, database monitors, and Visual Explain.

In July of 2007, IBM announced DB2 Web Query for System i, a product positioned as a query and reporting tool for the System i and the strategic enhancement to the Query/400 tool. DB2 Web Query has many advantages over Query/400. Most notably, it provides modernized development and end-user interfaces via a Web browser as well as modern report output formats such as HTML to the browser, Excel spreadsheets, and PDF. For more information on DB2 Web Query, download the IBM Redbook titled Getting Started with DB2 Web Query for System i.

In this article, I explain how DB2 Web Query accesses data from the DB2 for i5/OS database and how you can use various tools available on the System i to influence certain optimization behaviors. The ultimate objective is to ensure that DB2 Web Query reports are using the latest IBM technologies and obtaining optimal data access performance.

When accessing data from DB2 for i5/OS, DB2 Web Query uses one of three data adapters:

- DB2 CLI—Generates an SQL statement and submits a Call Level Interface (CLI) API request to the database engine.
- DB Heritage File—Generates an OPNQRYF CL command to submit to the database engine. Used to access multi-member files and multi-member logical files.
- Query/400—Uses the RUNQRY CL command to run an existing Query/400 query (*QRYDFN object).

Of the three adapters, only DB2 CLI can leverage the database's new SQL Query Engine (SQE). SQE is a totally redesigned query engine and was first introduced with V5R2 of i5/OS. It employs efficient, advanced techniques when optimizing queries, creating access plans, and accessing data. Prior to SQE, all query requests were processed by the Classic Query Engine (CQE).

DB2 for i5/OS actually uses both engines to handle all query requests. As of V5R4, the most common items and environments that prevent SQE from executing an SQL statement have been these:

- Non-SQL interfaces (Query/400, OPNQRYF, QQQQry API)
- Upper, Lower, or Translate function

- National language sort sequence
- User-defined table function (UDTF)
- DDS LOGICAL file references on FROM clause
- Select-Omit or derived logical files (non-standard indexes) defined on underlying table

This means that even though DB2 Web Query generates and submits an SQL statement to the database, there may still be cases in which the request is processed by CQE. For most of the restrictions listed above, the only way to obtain SQE processing is to completely remove the offending condition. However, if you are one of many shops that uses Select-Omit logical files in your applications, read on, as there is a method to allow peaceful Select-Omit logical file and SQE coexistence.

Avoiding CQE

Many legacy System i applications rely heavily on the use of Select-Omit logical files; therefore, the underlying database tables, when referenced by an SQL statement, are ineligible for SQE processing. It is important to clarify that the mere existence of a Select-Omit logical over a table will cause the query to go down CQE, even when it's the table, not the logical file itself, that is referenced in the statement. Consequently, when any SQL interface (including DB2 Web Query via the DB2 CLI adapter) references such a table, the query will be optimized and processed by CQE. In most cases, this is not the desired behavior.

In a DB2 Web Query environment, you can influence the optimizer to ignore the fact that these Select-Omit logical files exist, thereby clearing the path for SQE processing. While it is true that a possible remedy would be to remove those Select-Omit logical files, it is a solution fraught with risk because existing applications likely rely on the business logic contained in these files. It is better to craft a solution that will not impact the behavior and environment of other applications.

To minimize disruption while accomplishing the stated objective, two System i technologies can be used together:

- QAQQINI table
- CLI connect exit point

QAQQINI

QAQQINI is known as the query options table. It is a special table that DB2 for i5/OS uses to change/override the default behavior of the query engines (both CQE and SQE). Each system is shipped with a template QAQQINI file in the QSYS library. If this file is copied to the QUSRSYS library, its options are in effect for all users on the system. However, if you do not want its settings activated globally, you can copy the file to another library and use the CHGQRYA command to instruct the database to only use the settings for the file in that specific library and only for the job that issued the command.

One of the QAQQINI settings is IGNORE_DERIVED_INDEX. The default value of this setting is currently *NO, which means that the optimizer will not ignore derived indexes, so impacted queries must be

processed by CQE. Changing this value to *YES allows SQE to process the query even when a derived key or select/omit index exists over a table in the query. When the query is run with this setting in place, SQE ignores the derived indexes over the referenced table and is allowed to process the request (barring any other SQE inhibitors).

Normally, you do not have to use QAQQINI as a database-tuning mechanism. Running without a QAQQINI file (which would be the same as running with the file using its default settings) provides good performance to the majority of System i customers. In fact, an important change planned for V6R1 is to change the default of the IGNORE_DERIVED_INDEX to *YES. As a result, all derived indexes and Select-Omit logical files will be ignored by default, and SQE processing can occur without intervention. But in the interim, you can use the technique described in this article.

CLI Connect Exit Point

DB2 Web Query has three adapters that you can use to process a query request. When creating a new report, unless you need to access a multi-member file, a multi-format logical file, or an existing Query/400 object, the adapter you will be using is the DB2 CLI adapter. As described previously, when this adapter is used, the reporting server component of DB2 Web Query will generate an SQL statement to satisfy the request and submit the statement via a CLI request. These requests are handled by the QSQSRVR (pre-start) jobs on the System i. This is known as "server mode," and each time one of these jobs is used in this mode, a connection event occurs. The CLI connect exit point allows control to be passed to a user-written program each time one of these connection event occurs. By now, you probably see where this is going: Use this exit point to invoke a program each time a DB2 Web Query report is run. This program will direct the database to use a version of QAQQINI that instructs the database to ignore derived logical files, clearing the path for SQE processing.

By marrying these two technologies, we can allow other existing applications to run and continue to use Select-Omit logical files, while enabling DB2 Web Query requests to use SQE.

Determining Whether a Solution Is Needed

Before implementing the solution proposed in this article, you may want to first determine whether the described Select-Omit logical file and CQE scenario is occurring in your DB2 Web Query environment. Note: These same detection steps could be used for any SQL workload.

To do this, take the following steps:

1. Start an SQL performance monitor.

Sign on to a 5250 session and start a database monitor, using some of the new filtering options available in V5R4. In the following example, I use the FTRFILE and FTRUSER parameter to filter what data is collected. For DB2 Web Query report testing, database monitor data is only collected for all files queried in library QZRE for queries requested by user profile COBBG.

```
STRDBMON OUTFILE(QGPL/CQE_TEST1) JOB(*ALL) FTRFILE(QZRE/*ALL)
FTRUSER(COBB)
```

Note: You can also use iSeries Navigator to start a database monitor. This tool provides a graphical, wizard-like interface to start the monitor. For more information, consult the IBM Redbook OnDemand SQL Performance Analysis Simplified on DB2 for i5/OS in V5R4.

2. Run your DB2 Web Query report(s).
3. Analyze results.

From an SQL interface (STRSQL or the Run SQL Scripts window in iSeries Navigator), analyze the data collected in the database monitor collection. The following SQL statement can be used for this type of analysis:

```
WITH STATEMENTS AS (
  SELECT DISTINCT QQUCNT, QQJFLD, QQ1000
  FROM cqe_test1
  WHERE QQRID=1000 AND  QQC21 IN ('OP','SI','UP','IN','DL')  )
SELECT
   a.qvc43,
   COUNT(*) AS COUNT,
   B.QQ1000 AS STATEMENT
FROM cqe_test1  A, STATEMENTS B
   WHERE QQRID = 3014
   AND QQc16 ='N'  AND A.QQJFLD=B.QQJFLD
GROUP BY a.qvc43, B.QQ1000
ORDER BY 2 DESC
```

After running this statement against the collected monitor data, look for rows where the QVC43 column equals DK. This value indicates that CQE processing occurred because "An index with derived key or select/omit was found over a queried table." Here is an example of how this would look:

QVC43	COUNT	STATEMENT
DK	1	SELECT T1."C2CID1",T1."C2CID2",T1."C2APIM" FROM QZRE/QAZRECEV T1 ORDER BY T1."C2CID1",T1."C2CID2" FOR FETCH ONLY

For more information on the QVC43 column and the other types of information captured in the SQL performance monitor collection, see the online IBM publication DB2 Universal Database for iSeries Database Performance and Query Optimization, which can be downloaded from the i5/OS Information Center Web site.

In addition, you can use the iSeries Navigator tools to find and analyze the data collected in the database monitor and run the Visual Explain tool to see the optimization details of the captured query. In the example shown in Figure 1, notice that the reason SQE was not used is "Derived Key of Select/Omit Index Exists."

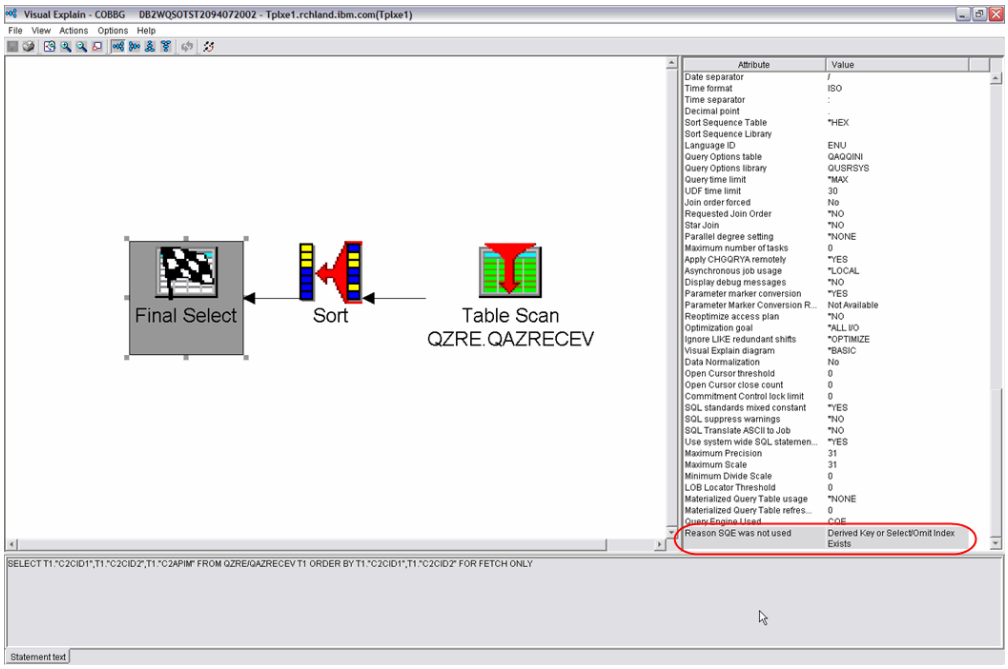

Figure 1: The Visual Explain tool allows you to see the optimization details of the captured query.

Note: The Visual Explain query engine feedback indicators require that you install the latest iSeries Access fixpack and Database Group PTFs.

Setting Up the Environment

Once you have determined that the proposed solution is needed in your environment and you're armed with the knowledge of the technologies to be used, the next step is to perform the tasks necessary to implement it:

1. Create and update a new version of QAQQINI.
2. Create the program to be called by exit point.
3. Activate the exit point.

Create and Update a New Version of QAQQINI

Since you want to ignore only derived indexes for DB2 Web Query requests, a global version of QAQQINI in the QUSRSYS library is undesirable. You want to activate this setting only for DB2 Web Query requests made through the CLI interface. To do that, you first need to create an "isolated" version of the query options file by following these steps:

1. Create a new copy of the file. From the command line, issue this command: `CRTDUPOBJ OBJ(QAQQINI) FROMLIB(QSYS) OBJTYPE(*FILE) TOLIB(COBBG) DATA(*NO) TRG(*YES)`

2. Populate the new file with the IGNORE_DERIVED_INDEX setting. From an SQL interface, issue this statement: `INSERT INTO COBBG/QAQQINI VALUES('IGNORE_DERIVED_INDEX', '*YES', NULL)`

Create the Program to Be Called by Exit Point

The next step is to create a CL program that will be called by the exit point. When creating the program for this exit point, the only requirement is that you must specify an input parameter for the user profile that is running the job (opening the connection). The example program shown below uses the CHGQRYA command to instruct the database to use the specified query options file and honor its settings for the current connection:

```
PGM         PARM(&CURUSER)

DCL         VAR(&CURUSER) TYPE(*CHAR) LEN(10)
DCL         VAR(&FORUSER) TYPE(*CHAR) LEN(10)

RTVDTAARA   DTAARA(COBBG/CLIEXITDA (1 10)) RTNVAR(&FORUSER)
IF          COND((&FORUSER = '*ALL') *OR (&FORUSER = +
              &CURUSER)) THEN(CHGQRYA QRYOPTLIB(COBBG))

ENDPGM
```

Notice in the example program that a data area object is being used as a "switch" to effectively turn off/on the activation of the query options file. If the value in the data area is *ALL or equals the name of the current user profile, the CHGQRYA command is executed to instruct the database optimizer to use this query options file for the current connection. If the data area contains any other value, the switch is turned off and the CHGQRYA command is not executed. This method makes it easy to enable and disable the functionality.

Note: The example program uses the data area simply to demonstrate this switching technique. It is not required for the exit program and could be eliminated if desired.

To create this supporting data area, issue the following command:

```
CRTDTAARA DTAARA(CLIEXITDA) TYPE(*CHAR) LEN(10) VALUE(*ALL)
```

When you're finished writing the program, issue the CRTCLPGM command against the CL source member to create the new program object.

Activate the Exit Point

The final step is actually activating the exit point so that the program is called each time a CLI connect event occurs. To do this, issue the following i5/OS command (specifying the exit program name and library created previously):

```
ADDEXITPGM EXITPNT(QIBM_QSQ_CLI_CONNECT) FORMAT(CLIC0100) PGMNBR(1)
PGM(COBBG/CLIEXIT)
```

Verifying the Results

Once everything is in place, the verification process described previously can be repeated to determine whether the solution had the desired effect:

1. Start another database monitor.
2. Run the DB2 Web Query report(s) again.
3. Analyze/compare results using SQL analysis statement and Visual Explain.

After running Visual Explain against the same queries in the captured database monitor, you should see some differences. For those same queries, the monitor data should no longer contain rows in which the value of the QVC43 column is DK. The Visual Explain graphical rendering should also look different. An example is shown in Figure 2.

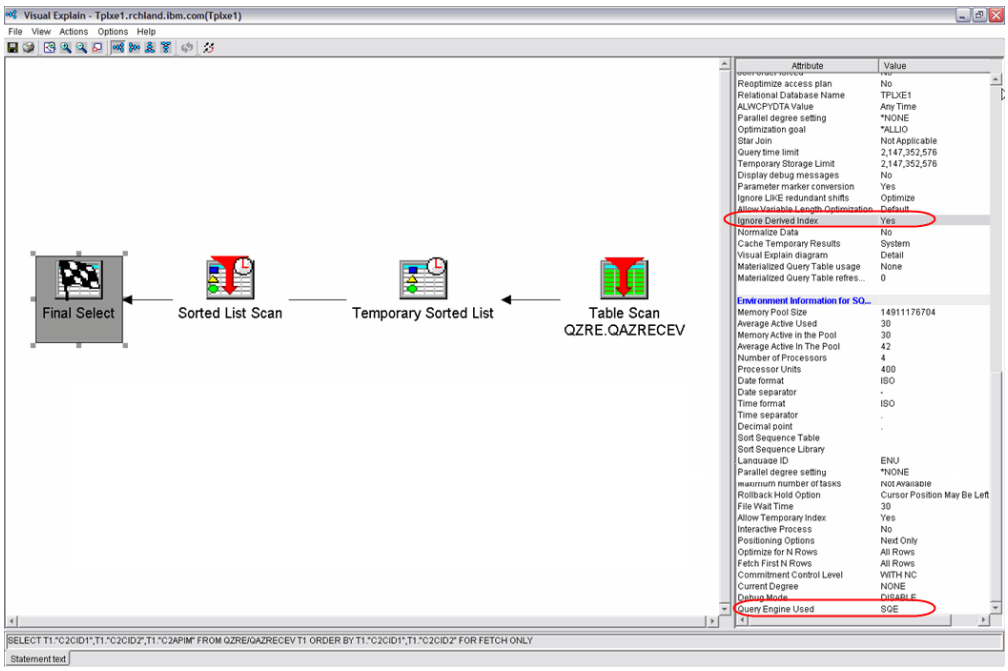

Figure 2: The Visual Explain graphics of the SQE processed query should look different now.

Several interesting things should be pointed out from this Visual Explain example:

- The Ignore Derived Index setting was implemented.
- The Query Engine used was SQE.
- The access plan is different from the one generated by the CQE optimizer. As mentioned previously, the SQE optimizer was redesigned with the intent of formulating more-optimal access plans. Though your results may vary, a better plan should result in a better-performing query.

Other Considerations

You should consider several other factors when putting this solution into practice.

Exit Program Already Defined

It is quite possible that you have other applications that are using CLI to access information in the database, and as such, you already have a program defined for the QIBM_QSQ_CLI_CONNECT exit point. Prior to implementing the solution, this can be verified by taking these 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 there is another exit program defined and you don't wish to disrupt this process, implementing this solution will require program consolidation. This means you will have to incorporate the CHGQRYA command into the existing exit point program.

Indexing Strategies

Because the IGNORE_DERIVED_INDEX setting does exactly that (forces SQE to ignore all non-standard indexes), the new optimizer cannot use this index during query implementation. In some cases, this may in fact mean that the optimal index exists but cannot be used by SQE because it is being ignored. Consequently, it is suggested that you do some additional analysis to determine if the key fields of those indexes should be used to create new standard indexes or DDS logical files (without the Select-Omit or derivation fields). Doing so would mean that SQE could recognize and use these indexes.

Here are some ideas for this kind of analysis:

- Determine whether the CQE access plan used any non-standard indexes.
- Run the query and look for index advisories issued by the SQE optimizer. This can be done using Visual Explain or the new V5R4 Index Advisor interface.
- Experiment! Create the non-standard indexes as standard ones and see if SQE uses them. If it does not, simply delete them.

For more information on indexing strategies, download IBM's white paper "Indexing and statistics strategies for DB2 for i5/OS."

Other Ideas for Exit Point Integration

The CLI exit point can be used in a DB2 Web Query environment for more than just influencing SQE usage. You could also use it to accomplish other objectives:

- Activate the Symmetric Multiprocessing (SMP) feature so that the database engine can use more than one processor to handle your DB2 Web Query requests.

- Activate the optimizer's inclusion of Materialized Query Tables (MQTs) when considering what resources to use in its access plan.
- Activate the QAQQINI QUERY_TIME_LIMIT option. This setting specifies a time limit that queries are allowed to run. If activated, the optimizer compares its time-to-run estimate with the specified setting. If the estimate exceeds the setting, the query is not allowed to run.

Getting the Best Results

In this article, I have introduced you to several tools available in the System i toolset: the QAQQINI query options file to influence optimizer behavior, exit points to provide customized end-user control at key system intervals, database monitors to capture optimizer decisions, and Visual Explain to drill down to see the details of those decisions. When used together, these tools can help ensure that your DB2 Web Query reports and graphs are using the latest and greatest IBM technologies.