# Using SQL views and stored procedures with DB2 Web Query

*Extend your reporting capabilities and secure your database by using these DB2 for i5/OS features*

*Gene Cobb*
*ISV Business Strategy and Enablement*
*January 2008*

## Table of contents

# Abstract

*This white paper explains how IBM System i developers and database administrators can use IBM DB2 for i5/OS features (views and stored procedures) to provide database security at the row and column level through the use of the IBM DB2 Web Query tool. This product was announced in July 2007 as the strategic query and reporting tool for the IBM System i platform. It provides modernized development interfaces and output formats for reports and graphs and leverages the strengths of the DB2 for i5/OS database. Specifically, you will learn how DB2 Web Query can use views and stored procedures to enhance and extend its reporting and graphing capabilities.*


# Introduction

In July 2007, IBM® announced a new licensed program named *IBM DB2® Web Query for System i*. This product is the result of a product alliance with Information Builders, Inc. and provides the IBM System i™ and IBM i5/OS® community with a Web-based query and reporting tool for DB2 for i5/OS data that is the strategic replacement for the IBM Query/400 product. DB2 Web Query surpasses Query/400 in many ways, but the major advantages it has over its predecessor are that it provides:

- A graphical report-development interface
- An SQL-based product that leverages the superior performance of the new SQL query engine (SQE)
- Modern report-output options, such as HTML, PDF and Microsoft® Excel spreadsheets

Its simple and intuitive interface allows users who possess a variety of skill levels to quickly produce meaningful reports.

The intent of this white paper is not to introduce you to DB2 Web Query, but rather to focus on some more advanced topics and techniques. Therefore, it assumes that you have already learned the basics of setting up and using the product. If you are new to the tool and need to learn about its fundamentals, it is suggested that you start with one of the IBM Redbooks™, *Getting Started with DB2 Web Query for System i*. This Redbooks title accompanied the product announcement and is available at the following Web site: www.redbooks.ibm.com/abstracts/sg247214.html?Open.

If you have had the opportunity to install and begin using DB2 Web Query, you probably went through the following report-development process:

1. Create a synonym (also referred to as *metadata*) against a table (*physical file*)
2. Create your report or graph based on that synonym
3. Run and test the report

Creating reports against table-based synonyms is the most commonly used method, especially when you are still getting familiar with the tool. However, as you can see in Figure 1, there are several types of sources on which to base your synonym. One of the goals of this white paper is to get you thinking about using a couple of these alternate synonym sources: SQL views and stored procedures. When in place, these synonym sources can provide additional reporting flexibility and functionality, improve report runtime performance, allow you to salvage and repurpose existing business logic, and provide you with more control over what actually appears in your DB2 Web Query reports.

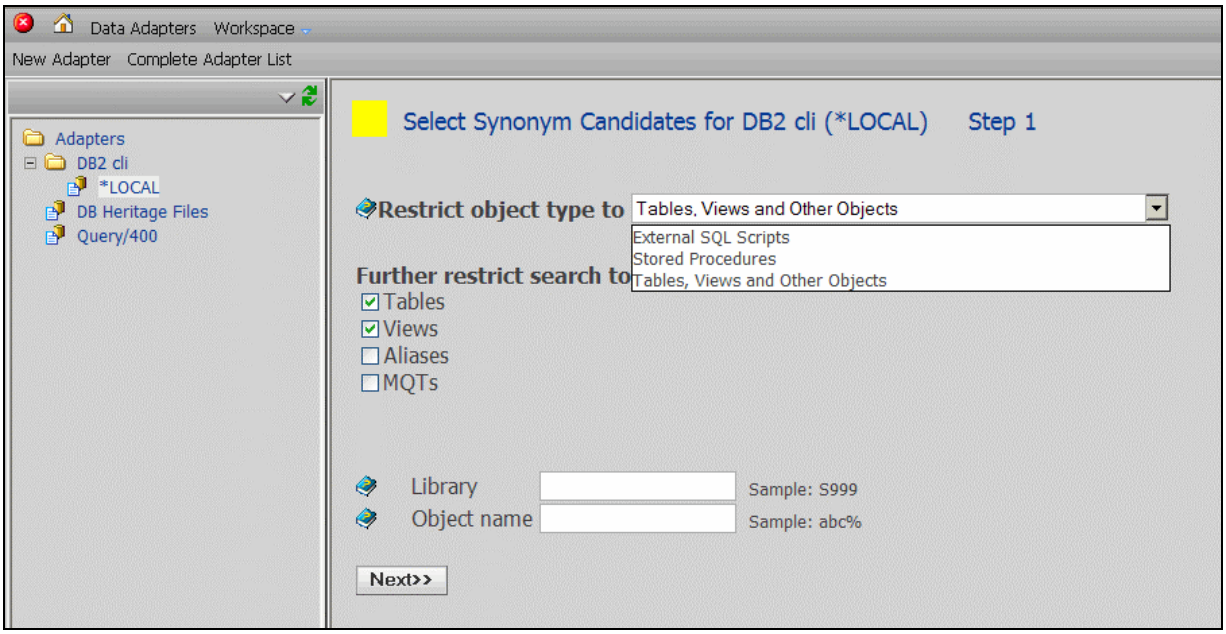


*Figure 1. DB2 Web Query synonym options*

## SQL views

An SQL view is a virtual table whose definition is based on a SELECT statement. In traditional i5/OS terms, you can think of them as nonkeyed logical files. Because no keys are associated with views, there is no access-path maintenance. This means you can create as many views as you like with no worry about performance repercussions that are associated with access-plan maintenance.

To understand the importance of using views, consider what occurs during the report-development and runtime processes:

- When you create a new report DB2 Web Query attempts to generate an optimal SQL statement to satisfy the request. That SQL statement is stored in the report's definition.
- When a user or requestor runs the report you have created, DB2 Web Query submits the SQL statement to the database engine, which processes that SQL request and returns a result set.

In some complex reporting cases, DB2 Web Query might have a difficult time formulating the optimal SQL statement, and the subsequent poor runtime performance of the report can reflect this. (**Note:** For an example of this, see the section "Performance Case Study" in Chapter 14 of the *Getting Started with DB2 Web Query for System i* Redbook mentioned earlier in this white paper. In these cases, you can use SQL views to help influence the actual SQL statement that is submitted to the database engine for processing.

When a report is created against a view-based synonym, DB2 Web Query uses the view when constructing its SQL statement. Based on the report definition, it might add other clauses to the statement (such as those for local selection, grouping and ordering). However, the view is still the referenced database object instead of the underlying table. When the statement is submitted to the database engine, the optimizer component combines the SQL select statement that was generated by DB2 Web Query with the select statement of the view's definition to generate a new query. (**Note:** The result is referred to as the *composite statement*). Consequently, reports that are based on SQL-view synonyms give you more (and sometimes complete) control over the statement that is ultimately submitted to the engine. In those cases where DB2 Web Query is not able to generate the optimal SQL statement, a report based on a view can result in superior runtime performance.

From a DB2 Web Query perspective, the benefits do not end there. Consider some other things that views can offer which tables cannot:

- They provide the ability to include join types (such as EXCEPTION and RIGHT OUTER) that are not supported directly by DB2 Web Query Web-based reporting tools. (**Note:** You can create RIGHT OUTER join types using the IBM add-on Developer Workbench client-development tool.)
- They give you the ability to create reports with complex business logic that uses CASE expressions, built-in functions and user-defined functions (UDFs).
- They can simplify DB2 Web Query report definitions by hiding this complex business logic.
- They can be used to implement security at both the row and column level.

To illuminate that last point, consider an example scenario that assumes you have been asked to produce a new DB2 Web Query report for use by department managers in your organization. The requirements for the report are as follows:

- Show all employees, as well as only those employees in the manager's division. (**Note:** This requires row-level security.)
- Sort and group the employees by division and department name.
- Include other columns (employee name [first and last], job title and salary).
- Display salaries only for employees who directly report to the manager (that is, they are in the same department as the manager who runs the report). Otherwise, show blanks for this column. (**Note:** This requires column-level security)

The new report runs against the tables shown in Figure 2. These tables are logically joined by the DeptNo (department number) column.

ORG table

| Division | DeptNo | DeptName | MgrID |
|---|---|---|---|
| Corporate | 10 | Head Office | 160 |
| Eastern | 15 | New England | 50 |
| Eastern | 20 | Mid Atlantic | 10 |
| Eastern | 38 | South Atlantic | 30 |
| Midwest | 42 | Great Lakes | 100 |
| Midwest | 51 | Plains | 140 |
| Western | 66 | Pacific | 270 |
| Western | 84 | Mountain | 290 |

STAFF table

| EmpID | UsrPrf | Lname | Fname | DeptNo | Job | Salary |
|---|---|---|---|---|---|---|
| 10 | PSANDERS | Sanders | Peggy | 20 | Mgr | 78357.5 |
| 20 | FPERNAL | Pernal | Frank | 20 | Sales | 78171.25 |
| 30 | JMARENGHI | Marenghi | Jill | 38 | Mgr | 77506.75 |
| 40 | PO'BRIEN | O'Brien | Peter | 38 | Sales | 78006 |
| 50 | EHANES | Hanes | Emily | 15 | Mgr | 80659.8 |
| 60 | BQUIGLEY | Quigley | Barry | 38 | Sales | 76808.3 |
| 70 | SROTHMAN | Rothman | Susanne | 15 | Sales | 76502.83 |
| 80 | MJAMES | James | Mel | 20 | Clerk | 73504.6 |
| 90 | KKOONITZ | Koonitz | Kerry | 42 | Sales | 78001.75 |
| 100 | BPLOTZ | Plotz | Bob | 42 | Mgr | 78352.8 |
| 110 | JNGAN | Ngan | Jane | 15 | Clerk | 72508.2 |
| 120 | SNAUGHTON | Naughton | Sam | 38 | Clerk | 72954.75 |
| 130 | TYAMAGUCHI | Yamaguchi | Tom | 42 | Clerk | 70505.9 |
| 140 | RFRAYE | Fraye | Roger | 51 | Mgr | 81150 |
| 150 | CWILLIAMS | Williams | Charlie | 51 | Sales | 79456.5 |
| 160 | JMOLINARE | Molinare | Jackie | 10 | Mgr | 82959.2 |
| 170 | AKERMISCH | Kermisch | Ann | 15 | Clerk | 72258.5 |

*Figure 2. Tables used to run example employee reports*

Using table-based synonyms, you would have a hard time creating a report with these requirements. Within the DB2 Web Query designer tool, you could probably create some defined fields and joins to satisfy the requirement for row-level security. However, the requirement for column-level security would be extremely difficult to fulfill.

One possibility is to create separate versions of the report: one with hard-coded selection criteria for each manager. In some manner, you then need to merge the results into a final report. This also requires more system administration because of the creation of i5/OS group profiles (one group profile for each manager) and DB2 Web Query domains that restrict which reports appear for each group profile. This solution is not highly desirable, particularly if there are many distinct departments and variations of the organizational reports to produce.

However, you can create this type of report without such a complex infrastructure. Creating an SQL view and a UDF allows you to easily build a single report that satisfies all the requirements.

*Figure 3 and Figure 4. SQL view definition*

show the UDF and view definitions.

```
CREATE FUNCTION protectsalary (
indept SMALLINT ,
insalary DECIMAL(7, 2) )
RETURNS DECIMAL(7, 2)
LANGUAGE SQL
NOT FENCED
DETERMINISTIC
BEGIN
  DECLARE mymgrflag CHAR(1)


  SELECT '1' INTO mymgrflag
  FROM staff
  WHERE usrprf = SESSION_USER AND job = 'Mgr' AND dept = indept ;
  IF mymgrflag = '1' THEN
    RETURN insalary ;
  ELSE
    RETURN NULL ;
  END IF ;
END  ;
```

*Figure 3. UDF definition*

```
CREATE VIEW secureDept
(division ,deptname ,lname ,fname ,salary )
AS
(
SELECT division, deptname, lname, fname,
       protectsalary(deptno, salary) AS salary
FROM staff a
INNER JOIN org b ON a.deptno = b.deptno
WHERE division IN
  (SELECT division FROM org c
   INNER JOIN staff d ON c.deptno = d.deptno
   WHERE usrprf = SESSION_USER)
)
```

*Figure 4. SQL view definition*

The view restricts the information that the user sees in the DB2 Web Query report. Managers who run the report based on this view only see data for the employees in their particular division. Notice the use of SESSION_USER in the WHERE clause of the view definition. This special register is new in i5/OS Version 5 Release 4 and contains the user profile of the user who runs the report. Because it effectively provides a way to filter table data based on the user profile, it is the key element in implementing row-level security.

The UDF is used to enforce the requirement for column-level security: the salary value from the table is only returned if the employee reports directly to the manager who runs the report. Otherwise, NULL is returned for this column.

A DB2 Web Query report that is created against this view and run by user PSANDERS (Peggy Sanders, the manager for department number 20) looks similar to the sample report shown in Figure 5:

PAGE 1

| DIVISION | DEPTNAME | LNAME | FNAME | SALARY |
|---|---|---|---|---|
| Eastern | Mid Atlantic | James | Mel | $73,504.60 |
| | | Pernal | Frank | $78,171.25 |
| | | Sanders | Peggy | $78,357.50 |
| | | Sneider | Tom | $74,252.75 |
| | New England | Hanes | Emily | . |
| | | Kermisch | Ann | . |
| | | Ngan | Jane | . |
| | | Rothman | Susanne | . |
| | South Atlantic | Abrahams | Jason | . |
| | | Marenghi | Jill | . |
| | | Naughton | Sam | . |
| | | O'Brien | Peter | . |
| | | Quigley | Barry | . |

*Figure 5. Sample employee report*

Notice that all employees in the Eastern Division (Peggy's division) appear in the report, but the only salaries shown are those for members of the Mid-Atlantic department, which Peggy manages.

Using these techniques, only one report is required for all managers — separate group profiles and DB2 Web Query domains are not necessary. Further, you have effectively pushed the business rules down to the database. This not only greatly simplifies the configuration and maintenance of a secure reporting environment, but also provides this business logic to all other interfaces and applications that access the database.

## Additional database security measures

To further secure your database, you can consider implementing a couple of additional techniques: *view-only access* and *data encryption*.

### View-only access

When accessing data in the database, view-only access forces all users and interfaces to go through the views rather than to access the tables directly. This measure prevents users from circumventing the business logic that is built into the views. When all your SQL views are in place, you can do this by taking the following steps:

1. Prevent users from opening the tables directly by removing all authorities or, at the very least, remove the *OBJOPR authority.

   ~~2.~~Give users the ability to access and read the view by granting *OBJOBR and *READ authorities to the view.

   ~~3.~~Allow users to read the table data through the view by granting *READ access to the underlying tables.

For the sample table and views described previously, you can use the CL program example shown in Figure 6 to implement view-only access.

```
PGM

/* Prevent users from opening the tables directly by revoking all authority */
RVKOBJAUT  OBJ(STAFF) OBJTYPE(*FILE) USER(*PUBLIC) AUT(*ALL)
RVKOBJAUT  OBJ(ORG) OBJTYPE(*FILE) USER(*PUBLIC) AUT(*ALL)

/* Give users ability to access and read the view by                    */
/*   granting *OBJOBR and *READ     authorities to the view             */
GRTOBJAUT OBJ(SECUR00001) OBJTYPE(*FILE) USER(*PUBLIC) AUT(*OBJOPR *READ)

/* Grant *READ access to the underlying tables */
GRTOBJAUT OBJ(STAFF) OBJTYPE(*FILE) USER(*PUBLIC) AUT(*READ)
GRTOBJAUT OBJ(ORG) OBJTYPE(*FILE) USER(*PUBLIC) AUT(*READ)

ENDPGM
```

*Figure 6. CL program example that implements view-only access*

### Data encryption

For an even more secure environment, you can consider the data encryption and decryption functions that are available with SQL. These functions allow you to store and retrieve sensitive data in an encrypted format in the database. For more information, see Kent Milligan's white paper *Protecting i5/OS data with encryption* at **ibm.com**/servers/enable/site/education/abstracts/efbe_abs.html.

## Stored procedures

Another type of synonym that you can create is based on a stored procedure. A *stored procedure* is a program or procedure in a service program that an application can call by using the SQL CALL statement. You can write stored procedures in the SQL procedural language, or you can use existing programs or service-program procedures (written in RPG, COBOL, Java™ or other languages) and register them to the database as stored procedures. If a stored procedure returns a result set, DB2 Web Query can capture that result set and use it as the source data for a report. Because returning a result set is often done by using a cursor in a stored procedure, you might be thinking that a cursor can be turned into an SQL view, and that views are simpler to implement. Although this might be true in many cases, stored procedures do offer a number of distinct advantages as compared to views. Consider that, with stored procedures, you can take advantage of the following benefits:

- Reuse and repurpose the business logic in your existing programs. You can modify an existing RPG program, (or one that is written in any language that the System i platform supports) to return a result set; then you can register that program to the database as a stored procedure. This means that you can convert many existing reports that are generated by RPG programs so that they work with DB2 Web Query. You can comment out the header lines and change the details line to write to a result set, temporary file or array — rather than to a print file record. By combining DB2 Web Query with existing report programs, you can salvage proven business logic and provide a modernized output format for those existing reports.
- Use native record-level access (RLA), instead of SQL, if you prefer that method of data access.
- Provide the ability to call to other programs, commands and system APIs, as well as to pull data from other system objects (such as messages queues, data queues or data areas).
    - **Adopt the authority of the user profile that owns the underlying program or service program.** This gives you the ability to restrict access to the database objects by allowing users to access the objects only through programs (stored procedures) with adopted authority.
    - **Provide auditing capability.** Your stored procedure can include logic to insert a row in an audit-log table to record any report request. With so much attention given to security and auditing these days, this can be an important consideration — especially for users who have access to sensitive information.
    - **Provide the ability to programmatically change attributes of the querying job.** The following examples show what you can specify in a stored procedure to change the environment:
        - SET CURRENT DEGREE to enable symmetric multiprocessing (SMP) and boost query performance of a long-running report if the requestor is a manager or other high-profile user.
        - SET OPTION SRTSEQ to change the collating sequence of the report.

To illustrate how you might use a stored procedure, assume that your report now has one more requirement: an auditing feature. That is, each time the report runs, it must log information (such as the timestamp, name of the report, and requesting user profile) to an audit table.

To satisfy this new requirement, you can take the following steps:

1. Create the audit table, as shown in Figure 7:

```
CREATE TABLE rptaudlog (
rpttimst TIMESTAMP,
rptname CHAR(25),
rptusrprf CHAR(10) )
```

*Figure 7. Creating the audit table*

2. Create the stored procedure, as shown in Figure 8:

```
CREATE PROCEDURE secure_departments_proc
DYNAMIC RESULT SETS 1
LANGUAGE SQL
NOT DETERMINISTIC
MODIFIES SQL DATA
P1 : BEGIN
  DECLARE c1 CURSOR WITH RETURN TO CLIENT FOR     1
  SELECT division, deptname, lname, fname, salary
  FROM secure_departments;   2
  INSERT INTO rptaudlog VALUES(CURRENT TIMESTAMP,
        'SECURE_DEPARTMENTS', SESSION_USER);      3
  OPEN c1 ;   4
END P1  ;
```

*Figure 8. Creating the stored procedure*

Several things are important to understand in this stored procedure. Refer to the lines that are annotated in Figure 8 as you read the corresponding explanations that follow:

[1] The cursor is declared with the WITH RETURN TO CLIENT clause specified. It is a good idea to specify this clause to ensure that the result set is returned to the client application. The WITH RETURN TO CALLER clause is the default, which causes problems in the event that a nested stored procedure (that is, a stored procedure called by another stored procedure) returns the result set.

[2] The view defined earlier is specified. There is no need to duplicate the business logic.

[3] The INSERT statement is specified to log the request. This satisfies the auditing requirement.

[4] The cursor is opened (and left open). This returns a result set to the client application (which is DB2 Web Query).

## Summary

Although DB2 Web Query is still new, SQL has been available on the System i platform for several years. Both are powerful tools that give you the ability to extract data from your database in an efficient and meaningful way. Use them together to create a reporting environment that is flexible, functional, secure and easy to maintain.

# Resources

These Web sites provide useful references to supplement the information contained in this document:

- IBM System i Information Center
  http://publib.boulder.ibm.com/iseries
- i5/OS on IBM PartnerWorld®
  **ibm.com**/partnerworld/i5os
- IBM Publications Center
  www.elink.ibmlink.ibm.com/public/applications/publications/cgibin/pbi.cgi?CTY=US
- IBM Redbooks® Web site
  **ibm.com**/redbooks
  - *Getting Started with DB2 Web Query for System i*
    www.redbooks.ibm.com/abstracts/sg247214.html?Open
- Online educational course: SQL Performance Basics
  **ibm.com**/servers/enable/site/education/ibo/record.html?4fa6
- The new SQL query engine (SQE)
  **ibm.com**/iseries/db2/sqe.html
- *Protecting i5/OS data with encryption* white paper by Kent Milligan
  **ibm.com**/servers/enable/site/education/abstracts/efbe_abs.html

# About the author

**Gene Cobb** is a DB2 for i5/OS technology specialist within the IBM ISV Enablement for System i organization. He has worked on IBM midrange systems since 1988, with 10 years in the IBM System i Lab Services group – formerly the Client Technology Center (CTC) in Rochester, Minnesota. When he was with the CTC, he assisted IBM customers with application design and development using RPG, DB2 for i5/OS, IBM CallPath/400 and IBM Lotus® Domino®. His current responsibilities include providing consulting services to System i developers, with special emphasis in application and database modernization.

# Acknowledgements

Thanks to the following people who reviewed and contributed to this paper:

- Dan Cruikshank
- Kent Milligan
- Michael Cain

The material contained in this white paper was originally printed in the November 2007 issue of *System iNews* and is reprinted here with their permission.

# Trademarks and special notices



References in this document to IBM products or services do not imply that IBM intends to make them available in every country.

DB2, Domino, i5/OS, IBM, the IBM logo, Lotus, PartnerWorld, Redbooks and System i are trademarks of International Business Machines Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Information is provided "AS IS" without warranty of any kind.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.