Craig S. Mullins

Return to Home Page

October 1995

DB2 Version 4 Outer Join Basics
by Craig S. Mullins

DB2 Version 4 provides a multitude of new features designed to ease the process of application design and programming. Perhaps the most desired new feature though is explicit outer join support. Outer joins will significantly impact the manner in which DB2 applications are coded. This article will discuss the manner in which outer joins are supported by DB2 V4 along with examples and suggestions for usage guidelines.

Various sample SQL statements will be presented in this article. Variations on the classic employee and department data structures will be used in these statements. Refer to the following tables to clarify SQL statements and result:

The EMP Table (Employees)

EMPNO	LASTNAME	FIRSTNAME	WORKDEPT
100	Tater	Paul	INT
200	Workerbee	Fred	HR
300	Merlin	Joe	DBA
400	Gump	George	DIS

The DEPT Table (Departments)

DEPTNO	DEPTNAME
INT	International
HR	Human Resources
DBA	Database Administration
FIN	Finance

The Basics of Joining

The capability to query data from multiple tables using a single SQL statement is known as a join. Join capability has existed in DB2 from the very beginning (from Version 1 through Version 3). The technical term for this type of join is inner join. Consider two tables, EMP and DEPT, containing information on employees and the departments in which they work. The following SQL statement will produce a list of employees and the name of the department in which each works:

```
SELECT  E.EMPNO, E.LASTNAME, D.DEPTNAME
FROM       EMP      E,
           DEPT    D
WHERE    E.WORKDEPT = D.DEPTNO;
```

This is an example of an inner join. An inner join will match the data based on the values of one or more columns in each table. In the example, the WORKDEPT column in the EMP table is compared to the DEPTNO column in the DEPT table. Only rows in which the values match are combined, creating a result row that is a concatenation of the columns from each table. If the value of the data in the columns

being matched is not unique, multiple matches might be found for each row in the table. Conversely, rows that do not have columns that match will not be retrieved.

The result of running this query will produce the following output:

EMPNO	LASTNAME	DEPTNAME
100	Tater	International
200	Workerbee	Human Resources
300	Merlin	Database Administration

Notice that the non-matched rows are not returned. The row for employee 400, Gump, is not returned because there is no match for the WORKDEPT code of DIS in the DEPT table. Likewise, the row for the "Finance" department name, FIN, is not returned because there are no employees that work in that department in the EMP table. What is an Outer Join? Because of the inherent functionality of the inner join, it can pose problems when users require not only rows that match based on the predicate, but also those rows that do not match. For example, if an employee is temporarily not assigned to a department, we still might want that information to appear in the result set of the previous query. As it is presently written, this information will not be returned when the query is executed.

This is where the outer join is useful. An outer join causes rows in one of the tables having no match in the other to appear in the result set. The missing column positions in the result set will contain nulls. This is necessary because sometimes the results will contain a value (when there is a match) and sometimes the results will not contain a value

(when there is no match).

Prior to DB2 V4, there was no explicit SQL operator to perform an outer join. Instead, a programmer would have to code a relatively complex SQL statement consisting of a simple SELECT, a UNION, and a correlated subselect.

The following SQL statement is an example of the outer join code that was necessary prior to DB2 Version 4:

```
SELECT  E.EMPNO, E.LASTNAME, D.DEPTNAME
FROM       EMP      E,
           DEPT     D
WHERE     E.WORKDEPT = D.DEPTNO
UNION
SELECT  E.EMPNO, E.LASTNAME, '** NO DEPT NAME **'
FROM       EMP      E
WHERE   NOT EXISTS
          (SELECT       D.DEPTNAME
           FROM    DEPT      D
           WHERE E.WORKDEPT = D.DEPTNO);
```

The first part of this monstrosity (above the UNION) will return all of the matching rows. The second part, all of the non-matching rows. This statement will return all employees in the EMP table. Where there is a match in the DEPT table the department name will be displayed. Where there is no match in the DEPT table, place holder text stating "** NO DEPT NAME **" is returned instead.

DB2 Version 4 simplifies this query significantly:

```
SELECT  EMP.EMPNO, EMP.LASTNAME, DEPT.DEPTNAME
FROM       EMP LEFT OUTER JOIN DEPT
ON     EMP.WORKDEPT = DEPT.DEPTNO;
```

The keywords LEFT OUTER JOIN cause DB2 to invoke an outer join returning rows that have matching values in the predicate columns but also return unmatched rows from the table on the left side of the join. In the case of the left outer join example shown, this would be the EMP table, because it is on the left side of the join clause.

Note that the WHERE keyword is replaced with the ON keyword for the outer join statement. Additionally, the missing values in the result set are filled with nulls (not a sample default as shown in the previous example).

There are three types of outer joins supported by DB2 V4: left outer join, right outer join, and full outer join. The syntax for the SQL SELECT statement has been enhanced enabling users to explicitly specify LEFT OUTER JOIN, RIGHT OUTER JOIN and FULL OUTER JOIN.

As you might guess, the keywords RIGHT OUTER JOIN cause DB2 to return rows that have matching values in the predicate columns but also return unmatched rows from the table on the right side of the join.

So the following query is 100% equivalent to the previous query:

```
SELECT  EMP.EMPNO, EMP.LASTNAME, DEPT.DEPTNAME
FROM       DEPT RIGHT OUTER JOIN EMP
ON     EMP.WORKDEPT = DEPT.DEPTNO;
```

The only difference is that the positioning of the DEPT and EMP tables is switched and a RIGHT OUTER JOIN is employed instead of a LEFT OUTER JOIN. Additionally, consider the following query:

```
SELECT  EMP.EMPNO, EMP.LASTNAME, DEPT.DEPTNAME
FROM       EMP RIGHT OUTER JOIN DEPT
ON     EMP.WORKDEPT = DEPT.DEPTNO;
```

In this example, the rows where there is a department number match are returned, but unmatched department names are also returned. This is useful if there are department names in the DEPT table with no employees assigned to the department.

The remaining outer join option is the FULL OUTER JOIN. It, like all previous outer joins, returns matching rows from both tables, but it also returns non-matching rows from both tables; left and right. A FULL OUTER JOIN can use only the equal (=) comparison operator. Left and right outer joins are able to use all the comparison operators. An example of the FULL OUTER JOIN will be shown later in the article.

Basic Outer Join Rules of Thumb

Do not move any explicit outer join statements into a production environment until you are sure that you will not need to fall back to DB2 V3. Outer join syntax is supported by DB2 V4 only. If you must fall back to DB2 V3, any outer join statements will cease to function.

Never code the old style of outer join requiring a simple SELECT, UNION, and correlated subselect once you have migrated to DB2 V4 and are sure that you will not fall back to a previous release. The new outer join syntax is easier to code, easier to maintain, and should be more efficient to execute. The outer join capability of DB2 V4 should reduce the number of bugs and speed application development time due solely to the significant reduction in lines of code required. Furthermore, as IBM improves the optimizer over time, techniques designed to make outer join more efficient will most likely focus only on the new, explicit outer join syntax and not on the old, complex SQL formulation.

Favor coding LEFT OUTER JOIN over RIGHT OUTER JOIN. The choice is truly arbitrary, but the manner in which DB2 V4 shows EXPLAIN information makes left outer joins easier to tune. A new, optional column has been added to the PLAN_TABLE to describe the type of outer join method being used by a particular SQL join statement. The JOIN_TYPE column is defined as CHAR(1) NOT NULL WITH DEFAULT. This column will contain the value "F" for a FULL OUTER JOIN, "L" for a LEFT OUTER JOIN or RIGHT OUTER JOIN, or a blank for INNER JOIN or no join. Right outer joins will be converted to left outer joins in the PLAN_TABLE. So, deciphering the PLAN_TABLE data is more difficult for a RIGHT OUTER JOIN than for a LEFT OUTER JOIN.

Outer joins can be combined with another join condition only by AND. Neither OR nor NOT can be specified!

The keyword OUTER can be removed from any of the outer joins. The following table provides a listing of equivalent syntax statements:

Outer Join Syntax

Syntax	Synonym
LEFT OUTER JOIN	LEFT JOIN
RIGHT OUTER JOIN	RIGHT JOIN
FULL OUTER JOIN	FULL JOIN

When specifying outer joins, remember that the WHERE keyword must be replaced with the ON keyword.

The COALESCE Function

At times, the COALESCE function may be needed to avoid nulls in the result columns of OUTER JOIN statements. The COALESCE function is a synonym for the VALUE function. How can it be useful in an outer join? Consider the following query:

```
SELECT  EMP.EMPNO, EMP.WORKDEPT, DEPT.DEPTNAME
FROM       EMP FULL OUTER JOIN DEPT
ON     EMP.WORKDEPT = DEPT.DEPTNO;
```

This query will return the following results:

EMPNO	WORKDEPT	DEPTNAME
100	INT	International
200	HR	Human Resources

300	DBA	Database Administration
400	DIS	---
---	---	Finance

Note that the department code for Finance is not displayed, even though we know by simple browsing of the DEPT table that the code is FIN. Well, the query requests that the WORKDEPT column from EMP be returned, not the DEPTNO column from DEPT. This can be rectified using the COALESCE function. The COALESCE function notifies DB2 to look for a value in both of the listed columns, one from each table in the outer join (in this case, EMP and DEPT). If a value is found in either table, it can be returned in the result set. Consider the following example:

```
SELECT  EMP.EMPNO,
        COALESCE(EMP.WORKDEPT, DEPT.DEPTNO)
                    AS DEPTNUM, DEPT.DEPTNAME
FROM       EMP FULL OUTER JOIN DEPT
ON     EMP.WORKDEPT = DEPT.DEPTNO;
```

This query will return the following results:

EMPNO	DEPTNUM	DEPTNAME
100	INT	International
200	HR	Human Resources
300	DBA	Database Administration
400	DIS	---

---	FIN	Finance

In this case, the last row of the result set contains the correct department code. The COALESCE function determined that the department code was stored in the DEPT.DEPTNO column and returns the value, instead of the null returned because there was no corresponding WORKDEPT number.

Column Renaming
The SQL statement we just examined also used another new feature of DB2 V4 known as column renaming. Using the AS clause, columns can renamed in a SELECT statement. This is useful for columns containing data derived from an expression or function. Of course, it can also be used to simply rename any column, not just those that are derived.

New Inner Join Syntax
As mentioned earlier, inner join capability has been supported in DB2 since its inception. DB2 Version 4 provides a new, optional syntax for specifying an inner join. This syntax has been added to provide compatibility with the new outer join syntax.

Consider, once again, our sample inner join statement:

```
SELECT  E.EMPNO, E.LASTNAME, D.DEPTNAME
FROM       EMP      E,
         DEPT    D
WHERE    E.WORKDEPT = D.DEPTNO;
```

This is the manner in which joins have been specified in all prior versions and releases of DB2. The tables being joined are separated

in the FROM clause by commas. The WHERE clause indicates the join predicates. DB2 V4 enables the comma to be replaced by the keywords INNER JOIN, and the WHERE clause to be replaced with ON. Our sample inner join thus becomes:

```
SELECT  EMP.EMPNO, EMP.LASTNAME, DEPT.DEPTNAME
FROM       EMP INNER JOIN DEPT
ON     EMP.WORKDEPT = DEPT.DEPTNO;
```

This new syntax is optional. The comma and WHERE clause are still functional for inner joins. However, when INNER JOIN is specified, the WHERE keyword must be replaced with the ON keyword.

The new inner join syntax may help when training new programmers in SQL. It was always confusing for neophytes to understand the concept of a join, when there was no explicit join keyword. Well, now there is!

Synopsis

DB2 Version 4 is the richest new release of DB2 in terms of programming enhancements in years. And outer join is one of the biggest impacts to SQL query syntax since DB2's inception. It is wise to learn these new techniques before your shop migrates to DB2 V4. If you do, you will be able to take advantage of these new features immediately, and save yourself and your company many coding hassles!

From DB2 Update (Xephon), October 1995.

Home. Phone: 281-494-6153 Fax: 281-491-0637