# DB2 update

# 47

*September 1996*

## In this issue

```
address attchmvs "wtohi wtomsg"
signal qkexit
QKEXIT:
senq = 0
address ispexec "vput (senq)"
exit
```

**Notes**

Two routines invoked in the REXX routine DB2AUTO are not included in this article (most installations will have their own equivalents):

- WTOHI issues highlighted messages to the master console
- REXXWAIT puts the process to sleep for a period of time.

CONCLUSION

After implementation, an entire outage (from DB2 abend to DB2 restart) has been reduced to approximately 1 minute. CICS connections can also be automated if necessary: this will be discussed in a future article.

*N L Nguyen*
*Systems Consultant (Australia)*


## Optimizing DB2 outer joins

BACKGROUND

As organizations migrate to DB2 Version 4, application developers will be coding complex queries using the new outer join mechanism. Although the new outer join syntax is arguably quite simple to learn, there are nuances that, on initial inspection, may escape the casual programmer. In short, coding outer joins that perform in an optimal manner may take some experimentation.

The article *DB2 Version 4 outer join basics* in the October 1995 issue of *DB2 Update* discusses the basic features of DB2 Version 4 outer

 

joins. This short article will provide a brief overview of outer joins, highlight a potential outer join performance problem, and suggest a method to achieve performance improvement.

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. Consult Figures 1 and 2 for the table structure and sample data.

| EMPNO | LASTNAME | FIRSTNAME | WORKDEPT | STATE |
|---|---|---|---|---|
| 100 | Tater | Paul | INT | IL |
| 200 | Workerbee | Fred | HR | PA |
| 300 | Merlin | Joe | DBA | IL |
| 400 | Gump | George | DIS | FL |
| 500 | Shue | Joyce | DIS | IL |
| 600 | Love | Buddy | HR | NJ |
| 700 | Jones | Becky | INT | IL |

*Figure 1: Sample EMP data*

| DEPTNO | DEPTNAME | RESPONSIBILITY |
|---|---|---|
| INT | International | International |
| HR | Human resources | Domestic |
| DBA | Database administration | Domestic |
| FIN | Finance | Domestic |
| MKT | Marketing | Domestic |

*Figure 2: Sample DEPT data*

## OUTER JOIN REVIEW

Loosely speaking, the standard relational join, also known as an inner join, combines rows from two tables where values in one table match values in the other table. An outer join returns not only the matching rows, but also causes rows from at least 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.

Prior to DB2 Version 4, outer joins were written by coding a program with multiple cursors to return the matching rows and also the non-matching rows or by coding a lengthy SQL statement that combined SELECT, 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, ' - '
FROM   EMP    E
WHERE  NOT EXISTS
       (SELECT  D.DEPTNAME
       FROM     DEPT    D
       WHERE    E.WORKDEPT = D.DEPTNO);
```

In DB2 Version 4 this query becomes much simpler:

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

## A POTENTIAL PERFORMANCE PROBLEM

In both of these cases, the desired result was all of the rows that matched plus all of the EMP rows which had no DEPT match. The results produced would be as follows:

| EMPNO | LASTNAME | DEPTNAME |
|---|---|---|
| 100 | Tater | International |
| 200 | Workerbee | Human Resources |
| 300 | Merlin | Database Administration |
| 600 | Love | Human Resources |
| 700 | Jones | International |
| 400 | Gump | - |
| 500 | Shue | - |

In each case, the entire result set can be described by the join conditions. There are no local predicates that need to be applied during the outer join.

Consider, however, how the query would need to be changed if additional row qualification were required. For example, what if we only wanted to return rows for employees who work in Illinois? This requires an additional predicate, which may look like this:

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

Initially, this looks correct and returns the correct rows:

| EMPNO | LASTNAME | DEPTNAME |
|---|---|---|
| 100 | Tater | International |
| 300 | Merlin | Database Administration |
| 700 | Jones | International |
| 500 | Shue | - |

In fact, the inherent problem with this outer join formulation may never be encountered in a test environment. However, when thousands or millions of rows are filtered out by the additional predicate(s), this method of coding outer joins will perform quite poorly.

### WHAT'S THE BIG PROBLEM?

In this case, the outer join is performed first, before any rows are filtered out. Our sample returns seven rows initially, and then filters out three of them from the result set. However, what if the result set in a production environment were several thousand rows without the local predicate? If we could add the local predicate before the outer join takes place, however, the result set turns into just a few rows.

To resolve this potential problem, we can use another DB2 Version 4 feature: in-line views. The basic features of in-line views were covered in the September 1995 issue of *DB2 Update*. To summarize, in-line views (sometimes referred to as nested tables) enable developers to specify a SELECT statement in the FROM clause of another SELECT statement. Any table expression can be written in the FROM clause.

How do in-line views help to resolve our potential problem? Consider the following re-write of our sample query:

```
SELECT E.EMPNO, E.LASTNAME, DEPT.DEPTNAME
FROM   (SELECT EMPNO, LASTNAME
```

```
        FROM EMP
        WHERE STATE = "IL") AS E

      LEFT OUTER JOIN DEPT

ON    E.WORKDEPT = DEPT.DEPTNO;
```

By moving the local predicate into the FROM clause as an in-line view, we can ensure that the local predicate will be evaluated before the outer join. This reduces the number of rows to be joined and can greatly enhance performance. Of course, the resultant SQL is a bit more complex and difficult to code, explain, and maintain, but the performance gains should far outweigh these legitimate concerns.

If additional local predicates are required, additional in-line views can be specified. If we only wanted to return rows for which a domestic resource had responsibility, we could change our sample query as follows:

```
SELECT E.EMPNO, E.LASTNAME, D.DEPTNAME
FROM   (SELECT EMPNO, LASTNAME, WORKDEPT
        FROM EMP
        WHERE STATE = "IL") AS E

       LEFT OUTER JOIN

       (SELECT DEPTNO, DEPTNAME
        FROM DEPT
        WHERE RESPONSIBILITY = "Domestic") AS D

ON     E.WORKDEPT = D.DEPTNO;
```

## SYNOPSIS

The new Version 4 outer join syntax is rich in features but can be confusing to learn and implement properly. It is wise to learn these new techniques as your shop begins to fully exploit DB2 Version 4. Wise use of outer join can bring about, not only a reduction in the amount of code you must write, but also better performing queries.

*Craig S Mullins*
*Senior Technical Advisor*
*Platinum Technology Inc (USA)*