Craig S. Mullins

Return to Home Page

October / November 2005

zData Perspectives

by Craig S. Mullins

SQL Performance Tuning Basics

When asked what is the single most important or stressful aspect of their job, DBAs typically respond “assuring optimal performance.” Indeed, a recent Forrester Research survey indicates that performance and troubleshooting tops the list of most challenging DBA tasks. With this in mind, let’s take a moment to outline the basic factors that influence the performance of DB2 applications.

Even though a proper investigation of DB2 performance issues should probably begin with database design, let’s start off with a discussion of SQL because it impacts more users. Everyone who needs to access or modify data in a DB2 database will use SQL to accomplish the task.

As you write SQL statements to access DB2 data, there are certain very simple, yet important rules to follow to encourage efficient SQL. Of course,

SQL performance is a complex topic and to understand every nuance of how SQL performs can take a lifetime to master. That being said, adhering to the following simple rules puts you on the right track to achieving high-performing DB2 applications.

1) The first rule is to always provide ***only the exact columns*** that you need to retrieve in the SELECT-list of each SQL SELECT statement. Another way of stating this is “do not use SELECT *”. The shorthand SELECT * means retrieve all columns from the table(s) being accessed. This is fine for quick and dirty queries but is bad practice for inclusion in application programs because:

   - DB2 tables may need to be changed in the future to include additional columns. SELECT * will retrieve those new columns, too, and your program may not be capable of handling the additional data without requiring time-consuming changes.

   - DB2 will consume additional resources for every column that requested to be returned. If the program does not need the data, it should not ask for it. Even if the program needs every column, it is better to explicitly ask for each column by name in the SQL statement for clarity and to avoid the previous pitfall.

2) ***Do not ask for what you already know***. This may sound simplistic, but most programmers violate this rule at one time or another. For a typical example, consider what is wrong with the following SQL statement:

```
SELECT   EMPNO, LASTNAME, SALARY
FROM     EMP
WHERE    EMPNO = '000010';
```

   Give up? The problem is that EMPNO is included in the SELECT-list. You already know that EMPNO will be equal to the value '000010' because that is what the WHERE clause tells DB2 to do. But with EMPNO listed in the WHERE clause DB2 will dutifully retrieve that column too. This causes additional overhead to be incurred thereby degrading performance. The overhead may be minimal, but if the same SQL statement is run hundreds, or perhaps

thousands, of times a day then that minimal performance impact can add up to a significant impact.

3) ***Use the WHERE clause to filter data*** in the SQL instead of bringing it all into your program to filter. This too is a common rookie mistake. It is much better for DB2 to filter the data before returning it to your program. This is so because DB2 uses additional I/O and CPU resources to obtain each row of data. The fewer rows passed to your program, the more efficient your SQL will be. So, the following SQL

```
SELECT   EMPNO, LASTNAME, SALARY
FROM     EMP
WHERE    SALARY > 50000.00;
```

Is better than simply reading all of the data without the WHERE clause and then checking each row to see if the SALARY is greater than 50000.00 in your program.

4) ***Avoid writing SQL to access DB2 tables like flat files.*** Programmers often fall prey to repeating tactics that worked previously. In general, this can be a useful strategy because it reduces your coding effort. But be careful to avoid mis-using DB2 by accessing it like non-database data.

DB2 is not designed to mimic the old master file processing tactics of QSAM files. By this I mean reading a record from a file and then using a value from that record to drive reads from an existing file. DB2 programmers try to mimic this processing using two cursors: one to read a row and the other using a value to drive the next cursor. This is a recipe for poor performance. Instead, code the SQL as a join and let DB2 do the work for you.

5) Finally, ***put as much work as possible into the SQL*** and let DB2 optimize the access paths for you. With appropriate statistics and proper SQL coding, DB2 almost always will formulate more efficient access paths to access the data than you can code into your programs.

These rules, though, are not the be-all, end-all of SQL performance tuning – not by a long shot. Additional, in-depth tuning may be required. But following

the above rules will ensure that you are not making “rookie” mistakes that can kill application performance.

From zJournal, October / November 2005

.

Home.