

z/Data Perspectives



CRAIG S. MULLINS

The Rise of Dynamic SQL

Have you noticed that dynamic SQL is more popular today than ever before?

There are a number of factors contributing to the success of dynamic SQL today. Commercial off-the-shelf applications, such as SAP, Siebel and Peoplesoft, utilize dynamic SQL when DB2 for z/OS is used as the database server. In many cases, too, dynamic SQL is the default choice for in-house application development. Development of Java applications using an IDE that interacts with Java Database Connectivity (JDBC) can be simpler for programmers than building static applications from scratch. Applications that use JDBC and Open Database Connectivity (ODBC) will result in dynamic SQL. Furthermore, dynamic SQL is prevalent in most Web-based development projects.

This trend to embrace dynamic SQL is in stark contrast to the mindset of a decade or so ago when the prevailing rule was to avoid it at all costs. Of course, back then, dynamic SQL in a DB2 for z/OS environment typically meant COBOL programs engineered to use dynamic SQL, which isn't as easy to code.

It was the DBA group that created those rules about avoiding dynamic SQL, and usually for good reason. Dynamic SQL was unpredictable because it was built "on the fly" in the program, and it was more difficult to monitor and tune because it's optimized at run-time, instead of beforehand during the BIND process.

Even today, the performance of dynamic SQL is one of the most widely debated DB2 issues. Some shops still try to avoid it, while many more place controls on its use. But most of the past concerns can be relegated to the dustbin of history. In this day and age, though, a strictly enforced rule of "no dynamic SQL" is unwarranted due to performance improvements made by IBM as well as the various flavors of static and dynamic SQL.

The differences between static and dynamic SQL have lessened over time. IBM has added various options and features that blur the differences between the two. Today, there are fewer hard and fast lines between what can be accomplished with dynamic SQL. There are various options now at your disposal to make static act more like dynamic, as well as to make dynamic act more like static.

Dynamic SQL even can offer advantages over static SQL when it's appropriately used. With dynamic SQL, access paths aren't pre-determined at BIND-time, so DB2 can take advantage of the most up-to-date statistics to build more optimal query execution plans. And for queries having predicates written on columns with non-uniformly distributed data, DB2 can factor the host variable values into

the access path criterion. This can produce performance improvements over static SQL, which has no knowledge of the host variable values.

Dynamic Statement Caching (DSC) removes yet another impediment to dynamic SQL performance. With DSC, DB2 saves prepared dynamic statements in a cache. After a dynamic SQL statement has been prepared and is automatically saved in the cache, subsequent prepare requests for that same SQL statement can avoid the costly preparation process by using the statement that's in the cache.

Of course, there are disadvantages to dynamic SQL, too. Whenever dynamic SQL statements are prepared and not in the DSC, total statement execution time increases. This is because the time to prepare the dynamic statement must be added to the overall execution time. And, keep in mind, in order for the dynamic prepare to be reused, the dynamic SQL statement must be exactly the same as the one that caused the prepared statement to be cached—even down to the same number of spaces in the text of the statement.

From the perspective of performance monitoring and tuning, dynamic SQL still can be more difficult to manage. For static SQL, the statement text is available in the DB2 catalog and the access path information is available in the associated plan tables after binding with EXPLAIN(YES). There is no PLAN_TABLE that contains the access paths for dynamic SQL, nor is the SQL available in the DB2 catalog. For this reason, some DBAs view dynamic SQL as a performance black hole. Of course, you can EXPLAIN the statements in the DSC, and there are tools that allow you to shine a light into this darkness by capturing statements from the DSC and explaining them.

Error detection also can be problematic because dynamic SQL is compiled only at run-time; errors in the SQL statement may not be detected until it's run. And, of course, authorization and security are more burdensome for dynamic SQL applications. For example, using dynamic SQL puts more responsibility on programmers to avoid security exposures, such as SQL injection attacks. But the bottom line is that dynamic SQL is a viable option for application development in the 21st century. It's wise to avoid stringent rules that prohibit its use in your shop. **Z**

About the Author

CRAIG S. MULLINS is a data management strategist with NEON Enterprise Software. He has worked as an application developer, a DBA, and an instructor with multiple database systems, including working with DB2 for z/OS since Version 1. He also is an IBM gold consultant and author of the *DB2 Developer's Guide and Database Administration: Practices and Procedures*. Website: www.craigsmullins.com