Craig S. Mullins

Return to Home Page

October / November 2008

zData Perspectives

by Craig S. Mullins

Reoptimizing SQL

Every good DB2 developer knows that DB2 offers two types of SQL, static and dynamic. **Static SQL** is hard-coded and embedded into an application program and then bound into a package, which determines the access path that DB2 will use when the program is run. **Dynamic SQL** is built within the program "on the fly" as it executes. Once built, the dynamic SQL statement must be compiled using the PREPARE statement; or, alternately, an implicit PREPARE is issued behind the scenes when implementing the EXECUTE IMMEDIATE flavor of dynamic SQL.

But are you aware of the REOPT parameter and how you can use it to reoptimize the static and dynamic SQL in your application programs?

Effects of REOPT on Dynamic SQL

You can gain additional optimization for dynamic SQL using the REOPT parameter of BIND. REOPT specifies whether to have DB2 determine an access path at run time by using the values of host variables, parameter markers, and special

registers. There are four options from which to choose when specifying REOPT:

- **REOPT(NONE) -** DB2 will not reoptimize SQL at run time to factor in the values of host variables, parameter markers, and special registers. The access paths determined at BIND will apply.
- **REOPT(ALWAYS)** - DB2 will re-prepare every time the statement is executed. This means that statements containing host variables, parameter markers, or special registers will be prepared using actual values, which can improve the optimization. Subsequent OPEN or EXECUTE requests for the same statement will reprepare the statement, reoptimize using the current set of values for the variables, and execute the newly generated query plan. Statements in plans or packages that are bound with REOPT(ALWAYS) cannot be saved in the cache. Additionally, KEEPDYNAMIC(YES) is not compatible with REOPT(ALWAYS).
- **REOPT(ONCE) -** DB2 will re-prepare SQL statements only once, using the first set of host variable values, no matter how many times the statement is executed by the program. The access path is stored in the dynamic statement cache and will be used for all subsequent executions of the same SQL statement. This option was introduced in DB2 V8.
- **REOPT(AUTO) -** This option directs DB2 to attempt to formulate the optimal access path in the minimum number of prepares. The basic premise of REOPT(AUTO) is to re-optimize only when host variable values change significantly enough to make reoptimization worthwhile. Using this option, DB2 will examine the host variable values and will generate new access paths only when host variable values change and DB2 has not already generated an access path for those values. This option was introduced in DB2 9.

After migrating to DB2 9, consider specifying REOPT(AUTO) for SQL statements that at times can take a relatively long time to execute, depending on the values of parameter markers. In particular, especially consider REOPT(AUTO) when parameter markers refer to non-uniform data that is joined to other tables.

Also, consider re-evaluating programs bound specifying REOPT(ONCE). In some cases, switching to REOPT(AUTO) from REOPT(ONCE) can produce performance improvement by reoptimizing when it makes sense, instead of just sticking with a single access path based on the first values supplied to the parameter markers.

Effects of REOPT on Static SQL

REOPT(ALWAYS) and REOPT(NONE) apply to static SQL as well as dynamic. REOPT(ONCE) and REOPT(AUTO) are not valid for static SQL because DB2 does not cache static plans. The REOPT parameters and to which types of SQL they

apply is summarized in the following table:

REOPT Parameter	Dynamic SQL	Static SQL
NONE	YES	YES
ALWAYS	YES	YES

ONCE	YES	NO
AUTO	YES	NO

Consider binding static SQL with REOPT(ALWAYS) when the values for your program's host variables or special registers are volatile and make a significant difference for access paths. This means that these statements get compiled at the time of EXECUTE or OPEN instead of at BIND time. During this compilation, the access plan is chosen, based on actual values.

Be sure to factor in the overhead to prepare the access plan for all the SQL in the program at run time -- the more complex the SQL the greater the overhead will be. If you have only one or two static SQL statements that would benefit from reoptimization at runtime consider creating a separate program. Put the statements that can benefit from reoptimization into a program that can be bound REOPT(ALWAYS) or REOPT(AUTO), and put the remaining statements into a program that can be bound with REOPT(NONE). Doing so will cause your application to incur the cost of reoptimization only for those statements that may benefit.

And remember to keep the REOPT parameter in mind as you develop your DB2 applications.

From zJournal, Oct / Nov 2008

.

Home.