DB2 10 FOR Z/OS AND REBIND

Guest post by Jim Kurtz, Advisory Software Consultant

DB2 10 for z/OS yields out-of-the-box DB2 CPU savings of up to 10 percent for traditional workloads when compared to running the same workloads on DB2 9. That's attention-getting, and it's possible without a REBIND. But to obtain the best performance and memory improvements, it will be necessary to REBIND all PACKAGEs containing static SQL – an idea that may cause fear and angst to most DBAs.

To understand why, let's review. For static SQL, it's during the BIND/REBIND process that the DB2 optimizer determines the optimal access path to the data for each SQL statement using various inputs such as the SQL statement text, the schema definition, and the current object statistics from the DB2 catalog. The run-time performance of an SQL statement is directly related to the path the optimizer chooses at BIND/REBIND time. As long as a statement is performing well, especially in production, DBAs hesitate to REBIND unless they absolutely must (for example, the PACKAGE is marked as invalid or inoperative) for fear the DB2 optimizer will choose a less efficient, poorer performing access path.

Is REBIND Required for Migration?

REBIND isn't required for migration to DB2 10, but it's strongly recommended to achieve the

significant performance improvements. However, all PLANs containing DBRMs and all PACKAGEs that were last bound on DB2 V5 or lower must be rebound. So, as Roger Miller said in a recent IDUG Solutions Journal article, "Please be kind and REBIND."

Wouldn't it be nice to know before you migrate to DB2 10 what access path changes to expect? This post discusses a process (homegrown or vendor-purchased) that can analyze and identify access path changes between two DB2 subsystems, one at DB2 9 and one at DB2 10, before migrating. Your actual "code" for this process will vary, but here are the basic steps.

Using the PLAN_TABLE and DSN_DETCOST_TABLE

This process relies on data that's externalized to the userid.PLAN_TABLE and the userid.DSN_DETCOST_TABLE when executing a BIND/REBIND…EXPLAIN(YES) command. It presumes baseline information exists in those tables in the DB2 9 subsystem for the desired PACKAGE. In most shops, this is a byproduct of the standard for production migrations.

1. In the DB2 10 subsystem, create the DB2 objects associated with the PACKAGE exactly the same way as they're created in the DB2 9 subsystem but with DEFINE NO on the CREATE TABLESPACE and CREATE INDEX statements.
2. In the DB2 10 subsystem, create a userid.PLAN_TABLE* and a userid.DSN_DETCOST_TABLE* if they don't already exist.
3. In the DB2 9 subsystem, create a userid.PLAN_TABLE_V10* and a userid.DSN_DETCOST_TABLE_V10* (using the DB210 format).
4. For the objects created in step 1, copy all DB2 catalog statistics associated with access path selection from the DB2 9 subsystem to the DB2 10 subsystem. In DB2 9 and 10, there are 17 DB2 catalog tables containing user-updateable columns that the DB2 optimizer uses for access path determination. Be sure to get them all or it may skew what the optimizer chooses for an access path in the DB2 10 subsystem.
5. In the DB2 10 subsystem, BIND the PACKAGE with EXPLAIN(YES) and OWNER(userid) to externalize access path information to the userid.PLAN_TABLE and userid.DSN_DETCOST_TABLE.
6. In the DB2 10 subsystem, unload the data from the userid.PLAN_TABLE and userid.DSN_DETCOST_TABLE.
7. In the DB2 9 subsystem, load the data from step 6 into the userid.PLAN_TABLE_V10 and userid.DSN_DETCOST_TABLE_V10.
8. In the DB2 9 subsystem, run a query that will combine the data from the DB2 9 and DB2 10 tables in a way that visually "stacks" the DB2 9 results over the DB2 10 results while grouping it by QUERYNO, QBLOCKNO, and PLANNO.

* For steps 2 and 3, the DB2 10 format for these tables is slightly different from the DB2 9 format:

- userid.PLAN_TABLE: 59 columns in DB2 9, 64 columns in DB2 10
- userid.DSN_DETCOST_TABLE: 118 columns in DB2 9, 129 columns in DB2 10

Next week, we'll look at the results of the query and an easier way to compare access paths.

> *The postings in this blog are my own and don't necessarily represent BMC's opinion or position.*