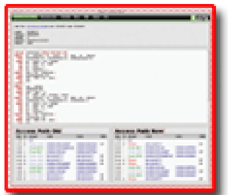

Craig S. Mullins

Return to Home Page

October / November 2006

The Resource for Users of IBM zSeries & S/390 Systems
z/JOURNAL

zData Perspectives

by Craig S. Mullins

To Rebind or Not To Rebind, is That a *Question*?

One of the most important contributors to the on-going efficiency and health of your DB2 environment is proper management of DB2 access path changes. A thorough REBIND management process is a requirement for healthy DB2 applications.

But many shops do not do everything possible to keep access paths up-to-date with the current state of their data. Approaches vary, such as rebinding only when a new version of DB2 is installed, whenever PTFs are applied to DB2, or to rebind automatically after a regular period of time. Although these methods are workable, they are less than optimal.

The worst approach though is the “if it ain’t broke don’t fix it” mentality. In other words, never REBIND unless you have to. The biggest problem it creates is that it penalizes every program in your subsystem for fear of a few degraded access paths. This results in potentially many programs having sub-optimal performance because the optimizer never gets a chance to create better access paths as the data changes. Of course, the possibility of degraded performance after a REBIND is real – and that is why some sites have adopted this approach.

Even so, the best approach is to perform regular REBINDs as your data changes. To do so, you should follow the Three R’s. Regularly **r**eorganizing to ensure optimal structure; followed by **R**UNSTATS to ensure that the reorganized state of the data is reflected in the DB2 Catalog; and finally, **r**ebinding all of programs that access the reorganized structures. This technique can improve application performance because access paths will be better designed based on an accurate view of your data.

Of course, adopting the Three R’s approach raises questions, such as “When should you reorganize?” To properly determine when to reorganize you’ll have to examine statistics. This means looking at either RUNSTATS in the catalog or Real-Time Statistics (RTS). So, the Three R's become the Four 4 R's – RUNSTATS (or RTS), REORG, RUNSTATS, then REBIND.

Some organizations do not rely on statistics to schedule REORGs. Instead, they build reorganization JCL as they create each object – that is, create a table space, build and schedule a REORG job, and run it monthly or quarterly. This is better than no REORG at all, but it is not ideal because you are likely to be reorganizing too soon (wasting CPU cycles) or too late (causing performance degradation until REORG).

It is better to base your REORGs off of thresholds on catalog or real-time statistics. Statistics are the fuel that makes the optimizer function properly. Without accurate statistics the optimizer cannot formulate the best access path to retrieve your data because it does not know how your data is currently structured. So when should you run RUNSTATS? One answer is "as frequently as possible based on how often your data changes." To succeed you need an understanding of data growth patterns – and these patterns will differ for every table space and index.

The looming question is this: why are we running all of these RUNSTATS and REORGs? To improve performance, right? But only with regular REBINDs will your programs take advantage of the new statistics to build more efficient access paths.

Without an automated method of comparing and contrasting access paths, DB2 program change management can be time-consuming and error-prone – especially when we deal with thousands of programs. And we always have to be alert for a rogue access path – that is, when the optimizer formulates a new access path that performs worse than the previous access path.

Regular rebinding means that you must regularly review access paths and correct any "potential" problems. Indeed, the Four R's become the Five R's because we need to ***review*** the access paths after rebinding to make sure that there are no problems. So, we should begin with RUNSTATS (or use RTS) to determine when to REORG. After reorganizing we should run RUNSTATS again, followed by a REBIND. Then we need that fifth R – which is to review the access paths generated by the REBIND.

The review process involves finding which statements might perform worse than before. Ideally, the DBAs would review all access path changes to determine if they are better or worse. But DB2 does not provide any systematic means of doing that. There are tools that can help you achieve this though.

The bottom line is that DB2 shops should implement best practices whereby access paths are tested to compare the before and after impact of the optimizer's choices. Only then can the question posed in the title of this column go away…

From zJournal, Oct / Nov 2006

.

Home.