REORGING DB2 - MUNDANE HOUSEKEEPING OR A HIGH PERFORMANCE MOVE?”

Every Db2 DBA knows that there are regular “housekeeping” tasks that you need to do to keep your Db2 databases running well – and one of those tasks is to run the Reorg utility. Today, there are three major goals relating to Db2 housekeeping which are addressed by running a reorganisation:

- Tidying up data that's out of place - rebuilding index structures and generally organizing data in tablespaces
- Consolidating extents and reclaiming wasted space
- Instantiating those on-line schema changes that allow a change to be specified with an ALTER statement, but where the change is actually made by a reorg at a later date

The second goal noted above is a kind of special case of the first, and the third should really be scheduled as part of a change delivery process as there could be other actions that need to be performed (such as running RUNSTATS, rebinding affected packages etc).

Which leads us to the realization that we are actually reorging our Db2 data, tables, and indexes to maintain a high level of performance for our APPLICATIONS. This really should be the primary focus of running any reorgs – to ensure that the data is maintained in an ideal state so that application SQL is executed as efficiently as possible. The other reasons for running a reorg tend to be more ad-hoc in nature – whereas, reorging to maintain high levels of application performance should be performed as often as is necessary.

Modern Day Challenges to Performing Reorgs

If we cast our minds back to our early experiences with Db2 (personally, I have to cast my mind WAY back), we probably scheduled reorgs in a quiet part of the week. For example, I was fortunate enough to have the entire weekend to myself, when I could effectively reorganise everything in sight. In today's world, we are hit by two conflicting situations:

- Firstly, we have far less down time than we used to so these reorgs have to fit into ever shrinking periods of time.
- Secondly, even though our down time window has diminished, the amount of data and the number of tables to process seems to have grown exponentially. Most people today do not have the time or the luxury of being able to reorganize everything every week. So we started to make compromises.

The first compromise most people look at is whether they need to reorganise every table or not – and make use of Db2 catalog statistics to determine which objects to reorg. In reality, what's happening is that the selection is determining which tables/indexes do NOT look like they need reorganising right now. The process usually generates JCL for submission at a later time.

This is a good start, but the reorg or don't reorg decisions are being based on statistics that were collected some time prior and they may not be up to date. Of course, ISV customers could be using tools provided by their vendor to look at statistics outside the catalog. This was typically a solution to earlier problems with the IBMs RUNSTATS utility in that it was not possible to collect data disorganization statistics without also providing that information to the Db2 optimizer. This would risk catastrophic degradations in performance if plans/packages were to rebound when these statistics indicated severe levels of data disorganization. This challenge has been mitigated by changes to RUNSTATS and is solved completely with the more widespread use of real time statistics.

The second compromise is to decide whether really big partitioned tablespaces need reorging in their entirety. A lot of time can be saved by reorging these at a partition level, perhaps only processing a subset of the total number of partitions at a time, and creating a schedule that ensures that all partitions are reorganized - eventually. The problem with this approach is that the Non Partitioning Indexes (NPIs) will tend to get more and more disorganised, because they are not themselves reorged or rebuilt as part of this process. They are just updated as rows are moved around in the partitions that are being reorged.

This can cause a gradual decrease in performance of applications using these indexes. Running a reorg of these NPIs can yield surprising results – I've seen up to 90% elapsed and CPU improvements in the worst cases. The challenge here though is that these NPIs can be of a significant size and it may not be possible to schedule a reorg in a usual housekeeping window.

Solution Options

So what can we do to improve our Db2 reorganisation strategy yet stay within the bounds of these conflicting demands and constraints forced upon us with digital business in the 21st century?

Let's look at the options:

- We should be looking at Real Time Statistics to determine which objects to reorg and which to skip. We should also be making the decision in real time, rather than generating JCL at one point in time, to be scheduled later when time permits. In the gap between generation and execution, some other table or index may need reorganising – and perhaps this other table should be a higher priority.

- We should be looking at table and index criteria separately. In many cases, it is actually a reorg of the index(es) that can have the biggest benefit to application performance
- Be careful with partition level reorgs. Sometimes these may be a necessary evil, but where you do use them, don't forget the implications for the NPIs. Make sure these are considered for reorgs regularly.
- Don't assume once a month (or once a week even) is often enough to reorganise your Db2 data. After a reorg, monitor the real time statistics and see how quickly the data is becoming disorganised. You will probably find, that you are reorging some tables or indexes too often. Equally, you may find that for some highly active tables, reorging once a week is not often enough!
- Look at the tooling you are using – is it fit for purpose? At the very least it should be scalable to the limits you need. It should also be minimally intrusive to your applications when it runs. This does not just mean making sure the switch phase doesn't cause problems, but includes considerations about CPU consumption and workfile usage. It's not good if your reorg costs more to execute than what you will save, but I have seen examples where using modern reorg tools can even reverse the year on year growth in application cpu consumption – and, of course, these savings are immediate. People have demonstrated benefits by reorging MORE often rather than less often.

Lastly, design yourself a strategy that you can put on auto pilot and leave alone. In my experience, DBAs have far too much work to do to be manually managing Db2 housekeeping. Set your reorg thresholds or policies and create JCL that will run regardless of any changes in the application. If the data grows, or if the rate of disorganization changes, or even if new tables are created, you can relax knowing that all your Db2 data is being kept optimally organized and that your applications are operating at their peak efficiency. In other words, you have completed your housekeeping and automated it for the best possible Db2 performance.