



ACTIVATE BUSINESS WITH THE POWER OF I.T.™



The Fastest Reorganization is No Reorganization

By Peter Plevka, BMC Software

Table of Contents

Reorganization Automation and Avoidance	1
DBA Trends	1
Tuning Potential with Reorganizations	1
Reasons to Reorganize	1
To Reorg or Not to Reorg, That's the Question	2
> Step 1: Collect Statistics	3
> Step 2: Object Selection/Exclusion	3
> Step 3: Threshold Analysis	3
> Step 4: JCL Generation	4
> Step 5: Execution	4
Summary/Conclusion	5

Reorganization Automation and Avoidance

With the number and size of DB2 database objects constantly growing, and requirements on availability getting tougher every month, it is imperative to plan reorganizations of tables and indexes well. This paper discusses different availability options to consider for reorganization and automation, and it discusses online reorganization and reorganization avoidance.

Whether you generate and execute your reorganizations manually or you have a user-written automation or a tool in place, you still need to decide which reorganization automation strategy fits your needs. Reorganization is not an art. Automate it once, get it off your desk, and spend your time on the more complex and difficult tasks.

While this paper discusses reasons to reorganize and automation processes, it will not discuss topics such as:

- > Reorg JCL or SYSIN cards
- > Error conditions and restart
- > Vendor-specific features

Rather, it will focus on how automation can help you avoid unnecessary reorganizations.

After you have read this paper, you should understand the automation approach in general, and you should be able to apply this approach in your shop. You should not have to sit in your office or at home and submit reorganizations manually, even if you have a broad maintenance window with which to work. After reading this paper, reorganizations should be simpler.

DBA Trends

Before we discuss reorganization automation and avoidance, let's take a look at a few DBA trends. Everyone is talking about IT as a service for the business. More and more people in management positions ask what IT delivers and how much it costs. Also, more and more businesses outsource their IT and pay for the service of their provider. These contracts contain service level agreements (SLAs) for deliverables such as an average response time per transaction.

You as a DBA are a part of IT, which delivers the service to the business. Directly affecting this service is the amount of data within the database, the availability of the database, and the performance of SQL accessing the database. Performance and availability actually are directly connected. It is safe to say that Performance = Availability, because a poorly performing SQL still returns a valid result, but directly influences the availability of the overall business process.

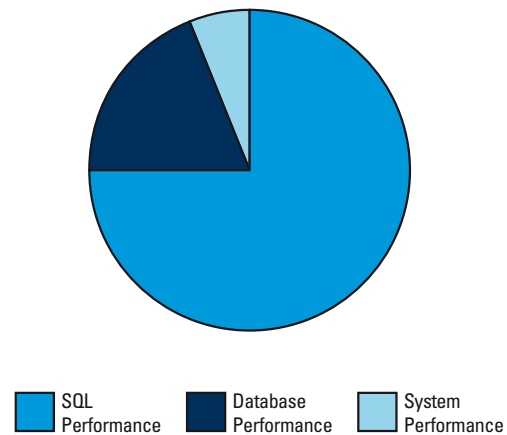
It's not black and white anymore (where black = database not available, white = database available). Rather, a poorly performing batch job may cause interruption or delay of another important business process.

Tuning Potential with Reorganizations

Consider the chart below as an application, a collection or package, or even a single SQL statement. The distribution of tuning potential is always the same. The biggest potential for tuning — 75 percent — lies within the application logic regarding its DB2 activities and the quality of the SQL statements, including index design.

The second largest part is the physical health of the database objects, the focus of this paper. You can achieve 10 percent to 12 percent tuning potential by having database objects well reorganized and sized. The smallest part — 3 percent to 5 percent — lies in the area of DB2 system resources ("System performance" in the chart). This includes tuning the buffer pool and other storage areas, as well as ZPARM settings.

You should consider all of these areas together to see the complete picture of tuning potential for a particular application, package, or single SQL statement within DB2.



Reasons to Reorganize

Running reorganization jobs should not be a time-consuming task. You have much more important things to do, many of which cannot be automated by any kind of software, such as migrating to the new version of DB2, implementing JAVA stored procedures, and so on. On the other hand, reorganizations are still important because they help to minimize I/O and increase availability of the application.

Take a look at your indexes. Healthy indexes are much more important than tablespaces. Indexes need to be reorganized to perform well for your application SQL. It is important to reorganize indexes. Consider the following:

- > You'll want to keep enough free space for inserts or updates for index spaces, because the index is always kept in perfect sort order by DB2.
- > Leaf pages should be physically in order within your index space data set. The LEAFDIST value from SYSIBM.SYSINDEXPART provides an average number of leaf pages not in perfect order. LEAFNEAR and LEAFFAR values from SYSIBM.SYSINDEXPART indicate how many leaf pages are either near or far from successive leaf pages. The logical order between leaf pages is always maintained through pointers. Whenever your application SQL needs to scan multiple leaf pages (that would either be a matching or non-matching index scan in your explain output) you want to have a LEAFDIST, LEAFNEAR, and LEAFFAR at zero — then DB2 will do sequential prefetch or list prefetch, which optimizes physical I/O.

It is also important to watch the index levels. An increase in index levels usually has a negative impact on the access path after a static or dynamic bind. But, you should only ALTER and reorganize the index if the number of levels will be decreased.

On the other hand, tablespaces still need to be watched. One of the main reasons to reorganize a tablespace with its associated indexes is to restore the clustering order of your table data, which resides in the tablespace. Before you start reorganizing tablespaces because of bad cluster ratio, ask if the clustering index really is used by application SQL.

If the SQL EXPLAIN shows the usage of the clustering index and sequential or dynamic prefetch is used by DB2 to retrieve the table data, the clustering index makes sense, and a REORG TABLESPACE to restore a bad cluster ratio would be needed. But if sequential prefetch is not used, you should question the need for this particular clustering index. Only range processing within your SQL introduces a possible need for a clustering index. A clustering index that is not used only causes unnecessary maintenance work on your side. Also the FREESPACE and PCTFREE parameters of your tablespace are directly connected to the need of a clustering index. If your SQL does not need a clustering index to do sequential prefetch, it doesn't matter in which order your table data is stored physically and, therefore, you don't need free space on your data pages. This saves space, I/O, and the need to reorganize this object because of low free space.

The reduction or elimination of secondary extents is a valid reorganization reason for both indexes and table spaces. Even though modern disk systems can retrieve data from disk very quickly, secondary extents cause an increase in CPU path length for each time the data or index page that resides on such a secondary extent needs to be retrieved from disk. These additional CPU resources might not matter for a single SQL, certainly not in terms of end-to-end transaction response time, but it adds up to a lot of CPU time for millions of SQL statements over time, thus increasing your overall CPU consumption for your DB2 workload, and possibly decreasing performance.

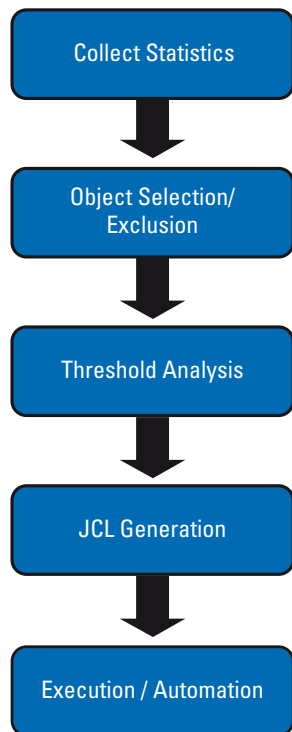
DB2 UDB for z/OS V8 provided a cure for the symptom — a sliding scale algorithm that automatically increases the amount of secondary extent allocation after a certain amount of secondary extents have been reached. The first 127 extents are allocated in increasing size, and the remaining extents are allocated based on the initial size of the data set. It allows DB2 to always reach the maximum data set size of a given table space or index. This is good, because it helps with availability and you will never run out of secondary extents anymore, but it cures the symptom and not the cause for secondary extents. It also forces you to alter your existing objects to either define no SECQTY or a value of -1 to make use of that new feature.

Other reasons to reorganize a tablespace or index are pending conditions like REORGP or AREO. When pending conditions exist on tablespaces, table space partitions, or index spaces you need to reorganize the object to avoid access failures for DML.

To Reorg or Not to Reorg, That's the Question

The graphic below shows a general step-by-step approach to automate the reorganization of DB2 table spaces and index spaces. This is an abstract view, and it does not represent the workflow of any particular product. This process is the result of many customer requirements, experiences, and implementations of reorganization automation processes around the world.

Because DB2 itself does not yet support a self-reorganization of its tables and indexes, customers around the world have devised slightly different flavors of reorganization automation and avoidance, but it all comes down to five main steps. It is not necessary that you implement all five steps at once. You can achieve a great deal of automation just by automating the threshold analysis and the JCL generation for example.



> Step 1: Collect Statistics

Before you can submit your reorganization jobs, you will need to collect statistical data on your DB2 objects. This data will be used to filter out those objects that need to be reorganized. The goal of this process is to decrease the number of reorganization jobs you must execute, to only those that need to be done, because of bad statistics.

Several utilities on the market collect different statistical information on DB2 objects and the underlying VSAM data sets. If you use IBM RUNSTATS, consider using the HISTORY option to collect statistics over time. Other vendors provide their own statistics database to hold statistical data over time and provide canned reports to help you understand trends and to do capacity planning.

If you update the DB2 catalog as part of statistic collection, you should consider existing REBIND/BIND processes. The collection of statistics to be used for access path selection has priority over your reorganization automation process. That means statistic collection at the wrong time might cause performance problems as access paths could change. If such performance considerations inhibit the use of RUNSTATS for your reorganization automation, that's when you want to start using Real-Time-Stats (RTS) for your reorganization automation. Consider using RUNSTATS primarily for your optimizer needs — thus supporting the quality of your SQL access paths.

RTS tables contain “since” statistical values, which represent a certain number of activities on a particular table space or index space since the last REORG/COPY/RUNSTATS. Remember, no history data is stored in RTS tables. Running one of the above utilities will reset the values in the RTS tables for that particular object.

Another alternative to gathering statistics is using the output of the VSAM LISTCAT command. You can use VSAM LISTCAT if you need to analyze data that is not available through RUNSTATS and/or RTS. The main advantage of using LISTCAT is that it always provides up-to-date information on the data set. But it is also more difficult to read LISTCAT output, as it cannot be analyzed by SQL, such as DB2 catalog tables. You need to parse the output to find the relevant information. The main use of the LISTCAT output lies in the analysis of space-related statistics such as allocation and extents.

> Step 2: Object Selection/Exclusion

An important part of the reorganization automation process is the concept of object selection and exclusion. You might house thousands of table spaces and index spaces in your DB2 catalog. Not all of them qualify for reorganization automation. Different applications may be using different tables with different needs for reorganization. You may need to reorganize some objects on a weekly basis, and others twice a week or only once a month.

The simplest way to select objects for reorganization is to use the IBM LISTDEF utility command. It provides a basic way to select and exclude objects based on their names for a certain reorganization run. Your own programs or vendor tools can supply more sophisticated methods for selecting and excluding objects. Consider excluding very large or small objects, LOBs, or hot tables, which have high I/O volume and need special treatment anyway.

Additional considerations are related to certain exceptions such as DEFINE NO objects, which exist in the DB2 catalog as an entry, but have not yet materialized into an existing VSAM data set (this will happen at first INSERT or LOAD).

Another requirement is to reorganize all objects related to a particular application or part of an application. You might not know which objects are used by certain packages or plans. With the information in SYSIBM.SYSPACKDEP or SYSIBM.SYSPLANDEP, you can find all objects that are used by certain plans or packages. Vendor tools do this automatically.

> Step 3: Threshold Analysis

This is the main part of the process — evaluating each selected object against a certain threshold of one or more particular statistical values. When the statistical value of the selected object breaches a predefined value, the condition is true and the object is eligible for reorganization. Certain situations require a combination of conditions with AND/OR, and if you have collected statistical data over time, you can compare current data with older values to qualify objects for reorganizations. For example, you can reorganize when a certain number of secondary extents have been reached and the increase in extents between statistical runs is greater than 25 percent.

If you have never applied conditions against your objects, it may be difficult to set the specific threshold values. A certain value could trigger all objects or none. Test this and, if necessary, change the value. For example, if you set the cluster ratio threshold to 99 percent (reorganize everything that has a cluster ratio lower than 99), you will probably get too many objects qualified for reorganization. This is not the idea of reorganization avoidance. If you set it to 50 percent, you might miss objects that need to be reorganized. Sample values should be provided by vendor solutions.

> Step 4: JCL Generation

Once you have a list of eligible objects, it's time to generate some JCL and SYSIN statements for your reorganization utility.

Avoid reorganizing the index, which qualified for reorganization in Step 3, and the parent table space, which houses the table for which this index, was created. When you reorganize a table space, all indexes defined on tables within that table space also are reorganized.

Remember to adjust the physical allocation of data sets through an ALTER before a reorganization if you find that the object has many secondary extents, or the growth rates (cardinality) increase over time. Depending on how you code your reorganization JCL and SYSIN, you may need additional work data sets (SYSREC, SYSUT1, SORTWK); size these according to the actual size of the object that is being reorganized.

IBM Online Reorganization — SHRLEVEL CHANGE

IBM Online Reorg uses mapping tables and indexes to track database changes during the reorganization phase. Actually it uses only the mapping index. Other reorganization products do not use DB2 tables to track those changes, but rather sequential files or other methods. When you use IBM Online Reorg, you will need those mapping tables. The mapping

table and index all look the same, regardless of the object that gets reorganized, but you should size it according to the size of the reorganized object. Some users have a predefined set of mapping tables and indexes. This is a valid solution, but remember that you cannot run two reorganizations of two different objects against the same set of mapping table and index at the same time. This prohibits parallelism (see Step 5).

Another solution is to create mapping table and index with the reorganization syntax in the same job step and drop those objects after the reorganization completed, all in the same SYSIN DD. When you use this solution, you should run a MODIFY RECOVERY utility periodically on the mapping table space to clean up the DBD, as it increases in size each time you create and drop a mapping table.

If you have to deal with many objects to reorganize, and must finish those reorganizations within a particular time frame/window, consider prioritizing the jobs. One option is to generate and submit the largest and worst reorganized object first. Another option is to balance objects over multiple jobs. Put a few large objects into one job and many small objects into another. If your reorganization product is able to store execution information over time, you can use this information to understand how long the reorganization for that particular object needed to execute in the past. This information can be used to prioritize and balance your reorganization work and should avoid the situation that reorganization jobs run into high transaction business hours.

> Step 5: Execution

At this point, you probably have a PDS with one or many members that contain reorganization JCL for one or many steps. If you want to automate the execution of your reorganizations, talk to your job scheduling person. Provide information on how often, on which day of the week, at which time, and how long you want to have reorganizations done in general. This is called "defining a maintenance window." To have a consistent end result, run Steps 1-5 together. Don't run Step 1 on Wednesday and then the rest of the steps on Sunday. The statistics would be old, and you might reorganize the wrong objects. You can delay Steps 4 and 5, but plan it according to the input of your job scheduling person. The more you know more about your applications and how (read/write) and when (time of day) they use DB2 data, the better, faster, and less intrusive your reorganizations will execute.

IBM Online Reorganization — Execution

If you really got a hot table to deal with (permanent transactions changing data in a CICS thread reuse environment, for

example) you may not be able to finish an online reorganization. Here, synchronization with the transactions is the issue.

Most online reorganization products on the market today need to synchronize (basically DRAIN all updaters) the reorganized table space or index space to switch over from the old (not reorganized) data set to the reorganized one. This needs to happen for all data sets that participate in the reorganization. For a table space reorganization, this also includes the index space data sets for all indexes defined on all tables within the table space that gets reorganized.

Ask yourself which process should “win” in a synchronization situation. If you want your reorganization job to end successfully, you can set SYSIN parameters in a way that will cause some end user transactions to fail. If you cannot afford that situation, use the regular maintenance window available for system maintenance, IPLs, and so on, for those objects. Even if this is only once a month, it is still better than no reorganization at all.

Conclusion

With many new technologies being developed by IBM for the upcoming releases of DB2, you need to use automation for your day-to-day tasks, to make room for the real complex work that cannot be automated anyway.

- > Reorganizations are not an art.
- > Automate as much as possible.
- > Get it off your desk.
- > And don't forget: the fastest reorganization is no reorganization.



ACTIVATE BUSINESS WITH THE POWER OF I.T.™

About BMC Software

BMC Software delivers the solutions IT needs to increase business value through better management of technology and IT processes. Our industry-leading Business Service Management solutions help you reduce cost, lower risk of business disruption, and benefit from an IT infrastructure built to support business growth and flexibility. Only BMC provides best practice IT processes, automated technology management, and award-winning BMC® Atrium™ technologies that offer a shared view into how IT services support business priorities. Known for enterprise solutions that span mainframe, distributed systems, and end-user devices, BMC also delivers solutions that address the unique challenges of the mid-sized business. Founded in 1980, BMC has offices worldwide and fiscal 2006 revenues of more than \$1.49 billion. Activate your business with the power of IT. www.bmc.com.

