Automatic Maintenance: Discover the brain behind the brawn

By Subho Chatterjee, Volker Markl, Ivan Popivanov, and Belal Tassi

You've read about how using the Configure Automatic Maintenance wizard can reduce the amount of time you need to spend concerned with backup images of the database, keeping statistics up-to-date, and keeping the table data and indexes organized. You know that you can have DB2 Universal Database (DB2 UDB) schedule these maintenance tasks when necessary. But you don't know all of the technical details behind these tasks. For example, you might want to know more about the automatic statistics collection algorithm and how it works. The DB2 UDB developers who implemented these features wrote this “Brain behind the brawn” article.

Automatic database backup in action

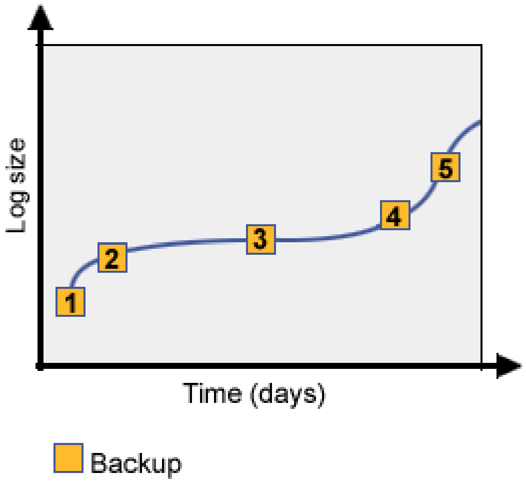
Regular backups are important to any recovery strategy for a database. And you are responsible for the creation of that recovery strategy. The automatic database backup feature provides a low overhead solution for situations that either don't have the resource and skill to effectively implement and manage a database-centric backup and recovery methodology, or for databases that don't require a formal recovery strategy. This is a "nice to have" feature in situations like non-critical development or test databases. To simplify the manageability of recovery in these environments for you, the automatic database backup feature performs full database backups only.

You should be aware that autonomic features are designed to generally defer to user workloads and activity on the system and this feature should not be used to replace backup procedures that are already in place.

The automatic database backup feature is designed to cover aspects of both time and data growth. Generally, you only need to backup your database only if there has been any change in the data since the last backup. Otherwise, the new backup would be identical to the previous one, and therefore not necessary. You certainly do not want to waste time doing an unneeded backup. In real life, the decisions you make to create a backup are based on many considerations: how much change has occurred, how many backup images and log files need to be archived, and how much time is planned or possible for a full physical restore and roll forward of the database.

Automatic database backup is adaptive in nature and reacts to the activity in the database by monitoring the amount of transaction log growth since the previous backup. The threshold at which it determines a new backup is required is configurable by you using the Configure Automatic Maintenance wizard in the DB2 UDB Control Center.

At the same time, automatic database backup allows for the ability to follow standard operating procedures that require regular scheduled backups by allowing you to configure a maximum time between backups. This ensures that, in the event you have little or no activity on your database, a regular scheduled backup is still performed by the automatic database backup feature, which you can configure. Because the feature will look after this for you, you save some of your time.



The figure illustrates the adaptive behavior of the automatic backup feature. The graph shows data change (represented by log growth) over time. Backups triggered by the automatic backup feature are represented by the numbered red ovals on the plot. At time 0 (representing a new database), if the automatic backup feature is enabled, a backup is triggered (to be run in the next available maintenance window). The triggered backup is represented by backup number 1 and indicates that the automatic backup tries to ensure that at least one backup has been performed on the database. This typically applies to databases left in the default circular logging mode.

Backup numbers 2 and 5 highlight the adaptive nature of the system as it reacts to the sharp increases in database activity and performs backups more frequently.

Backup numbers 3 and 4 indicate periods of low activity on the database where backups are performed when the maximum time allowed since the previous backup is reached.

When databases are configured in archive logging mode, it is possible to perform either online or offline backups on the database using the automatic database backup feature. By default, the feature performs online backups if the database has archive logging enabled.

In the case of offline backups, the QUIESCE DATABASE DEFER command will be issued prior to issuing the BACKUP DATABASE command. The QUIESCE command is used to put the database in ADMIN MODE. This command blocks new users from using the system and safely removes current users from the system in order to allow the offline backup to proceed. The

QUIESCE DATABASE command runs in DEFER mode to ensure that customer transactions fully complete before they are removed from the system.

Automatic reorganization in action

Normal database activity, such as repeated inserts, deletes, and updates, can affect the organization of the data in tables and indexes over time and adversely influence database performance. Rearranging a table in physical storage eliminates fragmentation and ensures that the table is stored efficiently in the database. Reorganization can also be used to control the order in which the rows of a table are stored, which is usually according to an index.

You should be aware of the following factors, which might indicate that a table needs reorganization:

- High volumes of insert, update, and delete activity on tables accessed by queries.
- Significant changes in the performance of queries that use an index with a high cluster ratio.
- Performance is not improved by running RUNSTATS to refresh statistical information.
- The REORGCHK command indicates a need to reorganize your table.

Even for experienced DBAs, it's a demanding task to analyze the trade-off between the cost of increasing degradation of query performance and the cost of reorganizing the table. The analysis includes factors like CPU time, elapsed time, and the reduced concurrency resulting from the REORG utility locking the table until the reorganization is complete.

As your business grows, the time that you can define for maintenance windows contracts. In such an environment, automatic reorganization leverages the benefits of automatic statistics collection to provide you a way to manage that trade-off and effectively reorganize tables that have significant activity.

The process flow outlined below details the process of identifying a table for automatic reorganization. In a DPF environment, this process flow is run on the database catalog partition. The evaluation of the need for reorganization is based on system table statistics that are not partition specific. As such, the running of the reorganization command is partition insensitive and the option of specifying a subset of partitions is not used.

The start of the process is to identify tables that have updated statistics. The STATS_TIME column in SYSCAT.TABLES tracks the last time the statistics were updated for a table. If the timestamp in the STATS_TIME column is more recent than the last time the process flow was run for a given table, it is identified as requiring evaluation.

The identified set of tables is then evaluated to determine which tables require reorganization. Two criteria are used to determine whether a table requires reorganization. The first criterion is the level of activity on the table. Only tables with an adequate level of activity qualify for table reorganization. If a table has sufficient activity to warrant further investigation, the statistics are then checked to determine if reorganization is required. Any table that is flagged is added to the list of tables requiring reorganization.

Table reorganization is recommended if the total number of overflow rows is more than 5 percent of the total number of rows in the table or the table size is less than 68 percent of the total space allocated for the table or the number of pages that contain no rows at all is more than 20 percent of the total number of pages.

Index reorganization is recommended if the clustering ratio of an index is less than 80 percent or more than 50 percent of the space reserved for index entries is empty or if recreating the index results in a tree having fewer levels with PCTFREE available or the number of pseudo-deleted RIDs on non-pseudo-empty pages is more than 20 percent or the number of pseudo-empty leaf pages is more than 20 percent of the total number of leaf pages.

At this point in the process flow, there is a decision point to check whether automation of table reorganization is enabled for the database. In Version 8.2, only classic reorganization actions that fit in the offline maintenance window are run automatically.

When statistics collection and reorganization are used in tandem, they leverage each other very effectively. Because statistics collection learns from and focuses more frequently on tables with greater change activity, those table statistics are updated more frequently. As a result, automatic reorganization looks at those tables more often than at those used less frequently. Such a focused approach reduces the list of objects to maintain, and expensive operations, such as reorganization, can therefore be performed more effectively.

Automatic statistics collection in action

Maintaining up-to-date statistics for tables is important because the DB2 UDB optimizer relies on statistical information to select the appropriate execution plan for a given SQL statement. Statistics collected by DB2 UDB reflect both logical and physical aspects of data stored in a table, and are also used to decide whether the table and its associated indexes need reorganization.

Consider the following key issues that are relevant to the collection of table statistics:

- Determining when to collect statistics
- Planning how to collect statistics
- Monitoring the execution (impact and progress) of the collection utility

- Monitoring the change impact of updating statistics

When to collect statistics is a superficially simple question to answer: when there's a critical mass of activity on the table (in terms of inserting, updating, and deleting data). Other activities, such as a data load or reorganization, might also require statistics being updated after they complete.

Typically, there are far too many tables for you to track the ones that exhibit enough activity to warrant statistics being collected. As a result, like many DBAs, you probably collect statistics on all tables on a daily or weekly basis. This can cost you a lot of time. Collecting statistics contributes a significant overhead in terms of CPU consumption and maintenance windows being used for those tables that don't require statistics collection.

Subtleties that contribute to the complexity of the actual task include the need to figure out and specify the relevant set of statistics to collect for a given table. Another challenge is the need to monitor the behavior of the optimizer after updating the statistics, to ensure that they do indeed contribute positively to query execution plan selection. This task is essential to avoid collecting the wrong set of statistics and to identify when data distribution and statistics remain unchanged even though there's activity on the table.

Automatic statistics profiling and statistics collection addresses these challenges and adds significant value in terms of your time saved over brute-force scheduling of RUNSTATS.

The automatic statistics collection algorithm uses a combination of heuristics and quantitative measures to determine the need to collect statistics on a particular table.

The first metric used is whether the table has had any activity at all. This is like a binary switch; if there has been no activity on the table, there's no point in collecting statistics for it.

DB2 UDB internally keeps track of the number of single row inserts, updates, and deletes done to a table; call it the UDI (update, delete, insert) counter. If there has been activity on the table, the UDI counter in combination with the table cardinality (number of rows) is used as the next metric. If more than 50% of the table has changed, RUNSTATS is scheduled. If less than 10% of the table has changed, it is not considered significant, so the table is discarded. Otherwise, the decision-making process continues its examination.

Since the data changed in the table might not impact the statistics significantly, the algorithm tries to minimize the execution of unnecessary RUNSTATS on large tables where there might be a significant impact due to the utility's execution. Large is defined as over 4 000 pages.

A small portion of data from each large table is sampled, and statistics are collected over this sample. This is accomplished using RUNSTATS sampling. The results from a quantitative comparison between the sampled statistics and the table statistics are used to decide whether to collect statistics on that table.

The efficiency of the sampling is determined by the sample size. However, sampling can introduce extra errors; that is, the statistics might look different just because they were obtained over a sample without using all data. To minimize the likelihood of such mistakes, the decision-making process compares only the distributions, which are relatively stable, and discards cardinalities that are more likely to be incorrect.

If the table requires statistics collection, a RUNSTATS is scheduled to be run at the next available maintenance window. After being run, the old and the new statistics are compared for changes. If there are indeed significant changes, then the table is examined more frequently in future iterations.

The execution model is designed to minimize impact to your user's workloads. It implicitly applies a throttling of 7% if a utility throttling limit is not explicitly specified. This is designed for highly available systems that have little or no explicit downtimes. As a result, it is geared to run in a 24x7 system and it implicitly takes advantage of periods of low activity on the system. While this is the default and recommended mode of operation, your control over actual execution times is allowed using the specification of online maintenance windows in the Configure Automatic Maintenance wizard.

This highly available model of operation is further enhanced with low priority locks that are forced in the case of any lock contentions caused by you or your user's activity on the database.

Automatic statistics profiling in action

Automatic statistics profiling is the first step towards LEO, a learning optimizer for DB2 UDB that automatically self-validates its execution model without requiring any user interaction to repair incorrect statistics or cardinality estimates using query feedback (QF). Automatic statistics profiling monitors the query activity on the database to automatically determine which statistics to collect, when to collect them, and what statistics configuration parameters to use. This feature automatically collects more detailed statistical information on areas of the database that are frequently queried.

The novel features of automatic statistics profiling include a "QF-driven" feedback loop that monitors the following activity:

- Estimated and actual results of running queries
- Methods for evaluating whether the data in a table has changed sufficiently to require a refresh of the statistics
- Methods for deciding which statistics to collect and at what level of detail to collect them, based on monitored query results

- Methods for scheduling statistics collection that combine and prioritize the recommendations from the UDI-driven and QF-driven analyses

The QF-driven autonomic process observes query activity by using a plan monitor (PM), which stores the best plan together with the optimizer's cardinality estimate for each intermediate result. During plan execution, a run-time monitor (RM) observes the actual cardinalities. All of this compile-time and run-time information is stored in a query feedback warehouse (QFW) in the form of relational tables.

A query feedback analyzer (QFA) periodically reviews these tables in order to generate modifications to the RUNSTATS profiles. The QFA bases these modifications on the discrepancy between actual and estimated cardinalities.

In addition to modifying RUNSTATS profiles, QFA communicates with the automatic statistics collection process about tables with modified RUNSTATS profiles and tables with outdated statistics. This ensures that the QFA can properly prioritize the automatic execution of statistics collection.

The QFA analyzes the data in the QFW to determine which tables have outdated statistics, whether and how the frequent values for columns on a particular table should be reconfigured, and which (intra-table) correlation statistics should be created in order to reduce estimation errors in the future. The QFA comprises three components. The table cardinality analyzer (TCA) detects whether statistics are outdated by comparing the estimated size and the actual size of a table.

The simple-predicate analyzer (SPA) uses estimated and actual cardinalities of simple equality predicates to determine the number of frequent values that should be used when creating the statistics for a particular column.

The correlation analyzer (COA) uses cardinality information about tables, simple equality predicates, and conjunctive predicates to determine the set of column-group statistics to recommend to the scheduler.

The QFA determines the cause of each estimation error by sequentially executing the table cardinality analyzer, the simple-predicate analyzer, and then the correlation analyzer. The processing is grouped either by column name or, for records involving column pairs, by column-group identifier, where a column-group identifier comprises the pair of column names enumerated in lexicographic order.

The QFA then sums up the absolute errors for each column and column group, and records the column-wise or group-wise error in the appropriate recommendation table.

Next, the QFA identifies those columns and column groups that are responsible for the most severe errors. QFA modifies the RUNSTATS profiles to ensure that RUNSTATS will increase the number of frequent-value statistics for each identified column and create joint statistics for each identified column group when it is next run on the table that contains the column or column group.

Finally, the QFA computes the total error for each table by combining the errors for table cardinality, cardinality of simple predicates, and cardinality of pairwise conjunctive predicates, weighing each error by its frequency (number of queries experiencing this error as stored in the QFW).

The output of the QFA is a prioritized list of tables that require statistics collection, along with the configuration parameter changes for the statistics of each table. The list is sent to the automatic statistics collection process, and the configuration changes are stored in the RUNSTATS profiles.

Monitoring automatic maintenance features

The automatic maintenance features are implemented in two distinct phases. The first phase evaluates the need for performing the maintenance action and schedules it internally, and the second phase executes the corresponding utility.

The evaluation phase can be performed even if the automatic maintenance database configuration parameters are disabled because the information generated can be used to report on the need to perform maintenance activities manually. The DB2 UDB Health Monitor defines a new set of maintenance health indicators that are used to alert you when manual intervention is required. Notification can occur because the automation is disabled, defined maintenance windows are not sufficient, or due to unexpected failures.

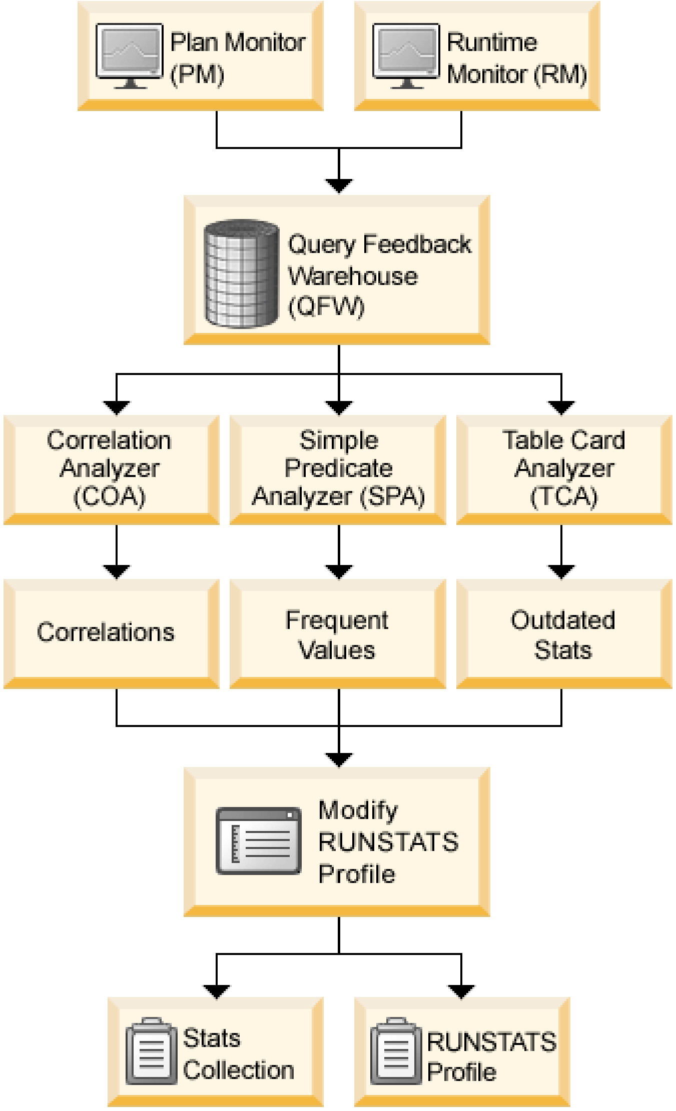
If both the health indicators and the automatic maintenance configuration parameters are disabled, then the evaluation phase will not be performed.

The execution phase is controlled through the database configuration parameters for automatic maintenance, as well as through the specification of the online and offline maintenance windows. Monitoring the execution of the RUNSTATS, REORG, and BACKUP utilities is available through the DB2 UDB System Monitor.

Witness the difference

You've read about how using the Configure Automatic Maintenance wizard can reduce the amount of time you need to spend concerned with backup images of the database, keeping statistics up-to-date, and keeping the table data and indexes organized. You know that you can have DB2 UDB schedule these maintenance tasks when necessary. Now you have the technical details behind these tasks. These details have provided you with insights into how the

Configure Automatic Maintenance wizard works so that you can use the automatic maintenance features more effectively and with greater ease.



Conclusion

Using the DB2 UDB Configure Automatic Maintenance wizard will save you time and stress. Some key maintenance tasks, with some guidance from you, can now be done by your new DBA sidekick, DB2 UDB. Go ahead and try it.

Reference

Ashraf Aboulnaga, Peter J. Haas, Sam Lightstone, Guy M. Lohman, Volker Markl, Ivan Popivanov, Vijayshankar Raman: Automated Statistics Collection in DB2 UDB. VLDB 2004As

you are working on putting together a recovery plan, you should read the first chapter in the *Data Recovery and High Availability Guide and Reference*, SC09-4831-01. The first chapter covers all aspects of developing a good backup and recovery strategy.

Information on table statistics and how they are used can be found in the "System catalog statistics" chapter in the *Administration Guide: Performance*, SC09-4821-01. Additional information on the RUNSTATS command is found in the *Command Reference*, SC09-4828-01.

The REORGCHK and REORG commands are also found in the *Command Reference*, SC09-4828-01.