DB2 update

June 2004

# Using Real Time Statistics (RTS)

***By Craig S. Mullins***

*This article is adapted from the latest version of Craig's book, DB2 Developer's Guide (5th edition).*

To maintain efficient production DB2-based systems, you must periodically monitor the DB2 objects that make up those systems. This type of monitoring is an essential component of post-implementation duties because the production environment is dynamic. Fluctuations in business activity, changes in data access patterns, or lack of attention to administrative needs can cause a system to perform inadequately. An effective strategy for monitoring DB2 objects in the production environment will catch and forestall problems before they affect performance.

One type of DB2 database object monitoring is to query the DB2 Catalog tables. However, a new feature of DB2 delivers real time statistics providing up-to-date information about DB2 database objects.

## Autonomic Statistics

Real Time Statistics (RTS) is the first step in IBM’s grand plans to automate parts of DB2 database administration. Introduced after the general availability of Version 7, but before Version 8, RTS provides functionality that maintains statistics about DB2 databases “on the fly,” without having to run a utility program.

Prior to the introduction of RTS, the only way to gather statistics about DB2 database structures was by running the RUNSTATS utility. RUNSTATS collects statistical information about DB2 database objects and stores this data in the DB2 Catalog. RTS, on the other hand, runs in the background and automatically updates statistics in two special tables as the data in DB2 databases is modified.

Where RUNSTATS is a hands-on administrative process, RTS is hands-off.

Real Time Statistics was announced with APARs PQ48447, PQ48448, PQ46859, and PQ56256.

## The RTS Tables

Although DB2 is always collecting RTS data, nothing is externalized until you set up the RTS database and tables to store the real time statistics. The RTS database is named DSNRTSDB and there is one table space (DSNRTSTS) with two tables:

SYSIBM.TABLESPACESTATS:--Contains statistics on table spaces and table space partitions

SYSIBM.INDEXSPACESTATS:--Contains statistics on index spaces and index space partitions

The columns in the SYSIBM.**TABLESPACESTATS** table are as follows:

| Column | Description |
|---|---|
| DBNAME | Database name |
| NAME | Table space name |
| PARTITION | The data set number within the table space. For partitioned table spaces, contains the partition number for a single partition. For non-partitioned table spaces, contains 0. |
| DBID | Internal database identifier. |
| PSID | Internal page set identifier (for the table space) |
| UPDATESTATSTIME | The timestamp when this statistics row was inserted or last updated. |
| TOTALROWS | The total number of rows or LOBs in the table space or partition. Indicates the number of rows in all tables for multi-table table spaces. |
| NACTIVE | |
| SPACE | |
| EXTENTS | |

| | |
|---|---|
| LOADRLASTTIME | The number of active pages in the table space or partition. Indicates the total number of preformatted pages in all data sets for multi-piece table spaces. |
| REORGINSERTS | The amount of space (in kilobytes) that is allocated to the table space or partition. Indicates the amount of space in all data sets for multi-piece linear page sets. |
| REORGDELETES | |
| REORGUPDATES | The number of extents used by the table space or partition. Indicates the number of extents for the last data set for multi-piece table spaces. For a data set that is striped across multiple volumes, the value is the number of logical extents. |
| REORGDISORGLOB | |
| REORGUNCLUSTINS | The timestamp when the last LOAD REPLACE was run for the table space or partition. |

| | |
|---|---|
| REORGMASSDELETE | The number of records or LOBs that have been inserted since the last REORG or LOAD REPLACE was run on the table space or partition. |
| REORGNEARINDREF | The number of records or LOBs that have been deleted since the last REORG or LOAD REPLACE on the table space or partition. |
| REORGFARINDREF | The number of rows that have been updated since the last REORG or LOAD REPLACE was run on the table space or partition. Does not include LOB updates because they are implemented as deletions followed by insertions. |
| STATSLASTTIME | The number of LOBs that were inserted since the last REORG or LOAD REPLACE that are not perfectly chunked. A LOB is perfectly chunked if the allocated pages are in the |
| STATSINSERTS | |
| STATSDELETES | |
| STATSUPDATES | |

STATSMASSDELETE

COPYLASTTIME

COPYUPDATEDPAGES

COPYCHANGES

COPYUPDATELRSN

COPYUPDATETIME

minimum number of chunks.

The number of records that were inserted since the last REORG or LOAD REPLACE that are not well-clustered with respect to the clustering index. A record is well-clustered if the record is inserted into a page that is within 16 pages of the ideal candidate page.

The number of mass deletes from a segmented or LOB table space, or the number of dropped tables from a segmented table space, since the last REORG or LOAD REPLACE was run.

The number of overflow records created and relocated near the pointer record since the last REORG or LOAD REPLACE was run. For non-segmented table spaces, a page is near the present page if the two page numbers differ by 16 or less. For segmented table spaces, a page is near the present page if the two page numbers differ by SEGSIZE*2 or less.

The number of overflow records created and relocated far from the pointer record since the last REORG or LOAD REPLACE was run. For non-segmented table spaces, a page is far the present page if the two page numbers differ by more than 16. For segmented table spaces, a page is far from the present page if the two page numbers differ by more than SEGSIZE*2.

The timestamp when RUNSTATS was last run on this table space or partition.

The number of records or LOBs that have been inserted/deleted/updated since the last RUNSTATS was executed on this table space or partition.

The number of mass deletes from a segmented or LOB table space, or the number of dropped tables from a segmented table space, since the last RUNSTATS was run.

The timestamp of the last full or incremental image copy on the table space or partition.

The number of distinct pages that have been updated since the last COPY was run.

The number of INSERT, UPDATE, and DELETE operations since the last COPY was run.

The LRSN or RBA of the first update after the last

| | |
|---|---|
| | COPY was run. |
| | Specifies the timestamp of the first UPDATE made after the last COPY was run. |

And the columns in the SYSIBM.**INDEXSPACESTATS** table are as follows:

| **Column** | **Description** |
|---|---|
| DBNAME | Database name |
| NAME | Index space name |
| PARTITION | The data set number within the index space. For partitioned table spaces, contains the partition number for a single partition. For non-partitioned table spaces, contains 0. |
| DBID | Internal database identifier. |
| ISOBID | |
| PSID | Internal identifier of the index space page set descriptor |

UPDATESTATSTIME

TOTALENTRIES

NLEVELS

NACTIVE

SPACE

EXTENTS

LOADRLASTTIME

REBUILDLASTTIME

REORGLASTTIME

Internal page set identifier (for the table space holding the table on which this index was created).

The timestamp when this statistics row was inserted or last updated.

The number of entries, including duplicates, in the index space or partition.

The number of levels in the index tree.

The number of active pages in the index space or partition.

The amount of space (in kilobytes) that is allocated to the index space or partition. Indicates the amount of space in all data sets for multi-piece linear page sets.

The number of extents used by the index space or partition. Indicates the number of extents for the

| | |
|---|---|
| REORGINSERTS | last data set for multi-piece table spaces. For a data set that is striped across multiple volumes, the value is the number of logical extents. |
| REORGDELETES | The timestamp when the last LOAD REPLACE was run for the table space or partition. |
| REORGAPPENDINSERT | Timestamp of the last REBUILD INDEX on the index space or partition. |
| REORGPSEUDODELETES | Timestamp of the last REORG INDEX on the index space or partition. |
| REORGMASSDELETE | The number of index entries that have been inserted since the last REORG, REBUILD INDEX or LOAD REPLACE on the index space or partition. |
| REORGLEAFNEAR | The number of index entries that have been deleted since the last REORG, REBUILD INDEX, or LOAD |

REORGLEAFFAR

REORGNUMLEVELS

STATSLASTTIME

STATSINSERTS
STATSDELETES

STATSMASSDELETE

COPYLASTTIME

REPLACE on the index space or partition.

The number of index entries that have been inserted since the last REORG, REBUILD INDEX or LOAD REPLACE on the index space or partition that have a key value that is greater than the maximum key value in the index or partition.

The number of index entries that have been pseudo-deleted since the last REORG, REBUILD INDEX, or LOAD REPLACE on the index space or partition.

The number of times that an index or index space partition was mass deleted since the last REORG, REBUILD INDEX, or LOAD REPLACE.

The number of index page splits that occurred since the last REORG, REBUILD INDEX, or LOAD REPLACE

COPYUPDATEDPAGES

COPYCHANGES

COPYUPDATELRSN

COPYUPDATETIME

in which the higher part of the split page was near the location of the original page. The higher part is near the original page if the two page numbers differ by 16 or less.

The number of index page splits that occurred since the last REORG, REBUILD INDEX, or LOAD REPLACE in which the higher part of the split page was far from the location of the original page. The higher part is far from the original page if the two page numbers differ by more than 16.

The number of levels in the index tree that were added or removed since the last REORG, REBUILD INDEX, or LOAD REPLACE.

The timestamp when RUNSTATS was last run on this table space or partition.

The number of records or LOBs that have been inserted/deleted since the last RUNSTATS was executed on this index space or partition.

The number of times that the index or index space partition was mass deleted since the last RUNSTATS was run.

The timestamp of the last full image copy on the index space or partition.

The number of distinct pages that have been updated since the last COPY was run.

The number of INSERT and DELETE operations since the last COPY was run.

The LRSN or RBA of the first update after the last COPY was run.

Specifies the timestamp of the first UPDATE made after the last COPY was run.

Each table has a unique index defined on it. Both are defined on the DBID, PSID, and PARTITION columns. The indexes names are:

- SYSIBM.TABLESPACESTATS_IX
- SYSIBM.INDEXSPACESTATS_IX

**When are Real Time Stats Externalized?**

As soon as RTS is applied (by running the proper version or maintenance level of DB2), DB2 begins to gather real time statistics. However, the RTS tables must exist in order for DB2 to externalize the real time statistics that it gathers.

Once the RTS tables have been created and started, DB2 externalizes real-time statistics to the tables at the following times:

- When the RTS database is stopped, DB2 first externalizes all RTS values from memory into the RTS tables before stopping the database.

- When an individual RTS table space is stopped, DB2 first externalizes all RTS values for that particular table space from memory into the RTS tables before stopping the database. Keep in mind, though, that the default installation uses only a single table space to store both RTS tables.

- When you issue -STOP DB2 MODE(QUIESCE), DB2 first externalizes all RTS values. Of course, if you stop using MODE(FORCE) no RTS values are externalized; instead, they are lost when DB2 comes down.

- As specified by the DSNZPARM STATSINT value. The default is every 30 minutes.

- During REORG, REBUILD INDEX, COPY, and LOAD REPLACE utility operations DB2 externalizes the appropriate RTS values impacted by running that utility.

## RTS Accuracy

In certain situations, the RTS values may not be 100% accurate. Situations that can cause the real time statistics to be off include:

- Sometimes a restarted utility can cause the RTS values to be wrong

- Utility operations that leave indexes in a restrictive state, such as RECOVER pending (RECP) will cause stats to be inaccurate.

- A DB2 subsystem failure

- A notify failure in a data sharing environment

To fix RTS statistics that are inaccurate, run a REORG, RUNSTATS, or COPY on the objects for which that stats are suspect. Furthermore, if you are using DB2 utilities from a third party vendor other than IBM, be sure that those utilities work with RTS. The third party utilities should be able both to reset the RTS values and use the RTS stats for recommending when to run utilities.

## DSNACCOR: The RTS Stored Procedure

IBM supplies a sample stored procedure called DSNACCOR that can be used to query the RTS tables and make recommendations based on the statistics. You can use DSNACCOR to recommend when to run a REORG, take an image copy, or run RUNSTATS. Additionally, DSNACCOR can report on the data set extents of table spaces and index spaces as well as on objects in a restricted state.

You can specify parameters to indicate to DSNACCOR which table spaces and indexes to analyze, or just run it without parameters to evaluates all table spaces and index spaces in the subsystem.

Keep in mind, though, that if the RTS values are inaccurate, the recommendations made by DSNACCOR will not be correct. Also, DSNACCOR makes recommendations based on general formulas requiring user input about your maintenance policies. These recommendations might not be accurate for every installation or subsystem.

You should consider using DSNACCOR in conjunction with DB2 Control Center. Control Center provides a nice GUI interface to the parameters of DSNACCOR making it easier to use than directly calling the procedure would be.

**Using the Real Time Statistics**

The following RTS guidelines and queries can be used against to help you identify maintenance and administration that needs to be carried out for database objects in your DB2 subsystems.

**Checking for Activity**

Because real time statistics are updated in an ongoing manner as DB2 operates, you can use them to see if any activity has occurred during a specific timeframe. To determine whether any activity has happened in the past several days for a particular table space or index, use the UPDATESTATSTIME column. Here is an example checking whether any activity has occurred in the past ten days for a table space (just supply the table space name):

```
SELECT   DBNAME, NAME, PARTITION,
         UPDATESTATSTIME
FROM     SYSIBM.TABLESPACESTATS
WHERE    (JULIAN_DAY(CURRENT DATE) -
          JULIAN_DAY(UPDATESTATSTIME))
<= 10
AND      NAME = ?;
```

### Basic Table Space Information

The RTS tables contain some good basic information about table spaces. The following query can be run to report on the number of rows, active pages, space used, number of extents, and when the COPY, REORG, LOAD REPLACE, and RUNSTATS were last run:

```
SELECT   DBNAME, NAME, PARTITION,
TOTALROWS,
         NACTIVE, SPACE, EXTENTS,
         UPDATESTATSTIME,
STATSLASTTIME,
         LOADRLASTTIME, REORGLASTTIME,
         COPYLASTTIME
FROM     SYSIBM.TABLESPACESTATS
ORDER BY DBNAME, NAME, PARTITION;
```

You can add a WHERE clause to this query to limit the output to only a certain database or for specific table spaces.

Pay particular attention to the timestamps indicating the last time that COPY, REORG, and RUNSTATS were run. If the date is sufficiently old, consider further investigating whether you should take an image copy, reorganize the table space, or run RUNSTATS.

Keep in mind though, that the span of time between utility runs is not the only indicator for when to copy, reorganize, or capture statistics. For example, RUNSTATS may need to be run only once on static data; similar caveats apply to COPY and REORG when data does not change.

## Reorganizing Table Spaces

Statistics that can help determine when to reorganize a table space include: space allocated, extents, number of INSERTs, UPDATEs, and DELETEs since the last REORG or LOAD REPLACE, number of unclustered INSERTs, number of disorganized LOBs, and number of near and far indirect references created since the last REORG.

```
SELECT   DBNAME, NAME, PARTITION,
SPACE,
         EXTENTS, REORGLASTTIME,
REORGINSERTS,
         REORGDELETES, REORGUPDATES,

REORGINSERTS+REORGDELETES+REORGUPDATES
              AS TOTAL_CHANGES,
         REORGDISORGLOB,
REORGUNCLUSTINS,
         REORGMASSDELETE,
REORGNEARINDREF,
         REORGFARINDREF
FROM     SYSIBM.TABLESPACESTATS
ORDER BY DBNAME, NAME, PARTITION;
```

You might want to add a WHERE clause that limits the table spaces returned to just those that exceed a particular limit. For example:

| **Specify** | **Description** |
|---|---|
| WHERE EXTENTS > 20 | Table spaces having more than 20 extents |
| WHERE TOT_CHANGES > 100000 | Table spaces with more than 100K changes |

WHERE REORGFARINDREF > 50 Table spaces with more than 50 far indirect references

Another way to get more creative with your RTS queries is to build formulas into them to retrieve only those table spaces that need to be reorganized. For example, the following query will return only those table spaces having more than 10% of their rows as near or far indirect references:

```
SELECT   DBNAME, NAME, PARTITION,
SPACE,
         EXTENTS
FROM     SYSIBM.TABLESPACESTATS
WHERE    (((REORGNEARINDREF +
REORGFARINDREF)
             *100
           )/TOTALROWS
          ) > 10
ORDER BY DBNAME, NAME, PARTITION;
```

Of course, you can change the percentage as you wish. After running the query you have a list of table spaces meeting your criteria for reorganization.

## Examining the Impact of a Program

You can use the TOTALROWS column of SYSIBM.TABLESPACESTATS to determine how many rows were impacted by a particular program or process. Simply check TOTALROWS for the table space both before and after the process; the difference between the values is the number of rows impacted.

## When to Run RUNSTATS for a Table Space

There are also statistics to help in determining when RUNSTATS should be executed. Run the following query to show the number of INSERTs, UPDATEs, and DELETEs since the last RUNSTATS execution:

```
SELECT   DBNAME, NAME, PARTITION,
         STATSLASTTIME, STATSINSERTS,
         STATSDELETES, STATSUPDATES,

STATSINSERTS+STATSDELETES+STATSUPDATES
               AS TOTAL_CHANGES,
         STATSMASSDELETE
FROM     SYSIBM.TABLESPACESTATS
ORDER BY DBNAME, NAME, PARTITION;
```

## When to Take an Image Copy for a Table Space

You can issue the following query to report on statistics that will help you to determine whether a COPY is required:

```
SELECT   DBNAME, NAME, PARTITION,
COPYLASTTIME,
         COPYUPDATEDPAGES,
COPYCHANGES,
         COPYUPDATELRSN,
COPYUPDATETIME
FROM     SYSIBM.TABLESPACESTATS
ORDER BY DBNAME, NAME, PARTITION;
```

Basically, as the number of distinct updated pages and changes since the last COPY execution increase, the need to take an image copy increases. A good rule of thumb to follow is when the percentage of updated pages since the last COPY is more than 25% of the active pages, then it is time to COPY the table space. You can add the following WHERE clause to the above query to limit the output to only these table spaces:

```
WHERE ((COPYUPDATEDPAGES*100) /
NACTIVE) > 25
```

## Basic Index Space Information

Do not forget that there are also RTS statistics gathered on indexes. The following query can be run to report on the number of rows, active pages, space used, number of extents, and when the COPY, REORG, LOAD REPLACE, and RUNSTATS were last run:

```
SELECT   DBNAME, INDEXSPACE,
PARTITION,
         TOTALENTRIES, NLEVELS,
NACTIVE,
         SPACE, EXTENTS,
UPDATESTATSTIME,
         LOADRLASTTIME,
REBUILDLASTTIME,
         REORGLASTTIME, STATSLASTTIME,
         COPYLASTTIME
FROM     SYSIBM.INDEXPACESTATS
ORDER BY DBNAME, NAME, PARTITION;
```

## Reorganizing Index Spaces

Just like the table space stats, there are index space statistics that can be used to determine when to reorganize indexes. These statistics include the last time REBUILD, REORG or LOAD REPLACE occurred, as well as statistics showing the number of INSERTs and DELETEs since the last REORG or REBUILD. And RTS does not skimp in the details. You get both real and pseudo DELETEs, as well as both singleton and mass DELETE information. RTS also tracks both the number of index levels and index page split information resulting in near and far indirect references since the last REORG, REBUILD INDEX, or LOAD REPLACE. The following query can be used to return this information:

```
SELECT   DBNAME, NAME, PARTITION,
         REORGLASTTIME, LOADRLASTTIME,
         REBUILDLASTTIME,
TOTALENTRIES,
         NACTIVE, SPACE, EXTENTS,
NLEVELS,
         REORGNUMLEVELS, REORGINSERTS,
         REORGAPPENDINSERT,
REORGDELETES,
         REORGPSEUDODELETES,
REORGMASSDELETE,
         REORGLEAFNAR, REORGLEAFFAR
FROM     SYSIBM.INDEXPACESTATS
ORDER BY DBNAME, NAME, PARTITION;
```

These statistics can be examined after running jobs or processes that cause heavy data modification.

Pay particular attention to the REORGAPPENDINSERT column. It contains the number of inserts into an index since the last REORG for which the index key was higher than any existing key value. If this column consistently grows, you have identified an object where data is inserted using an ascending key sequence. Think about lowering the free space for such objects because the free space is wasted space if inserts are always done in ascending key sequence.

### When to Run RUNSTATS for an Index Space

RTS provides index space statistics to help determine when to run RUNSTATS similar to the table space statistics. Run the following query to show the number of INSERTs, UPDATEs, and DELETEs since the last RUNSTATS execution:

```
SELECT   DBNAME, NAME, PARTITION,
         STATSLASTTIME, STATSINSERTS,
         STATSDELETES, STATSMASSDELETE
FROM     SYSIBM.TABLESPACESTATS
ORDER BY DBNAME, NAME, PARTITION;
```

### Summary

Real time statistics can be used to augment your DB2 object monitoring process. Be sure to take advantage of the continuously updated RTS values to improve the administration and performance of your DB2 databases.

From DB2 Update, June 2004.