Craig S. Mullins

Return to Home Page

Vol. 13, No. 3 (February 2007)

From IDUG Solutions Journal...

The Buffer Pool

Why Haven't You Started Using Real-Time Statistics?
By Craig S. Mullins

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. This approach requires regular RUNSTATS executions to keep your statistics up-to-date in the catalog. If you don't do that, then your catalog queries will be returning outdated information about your database objects.

There is, however, a relatively newer feature of DB2 that delivers real time statistics providing up-to-date information about DB2 database objects. This feature is called, appropriately enough, Real Time Statistics, or RTS for short.

There is a general wariness "out there" precluding widespread adoption and implementation of RTS. My very unscientific polling of DB2 user group attendees indicates that only a smattering of DB2 shops are using RTS. This is a shame because RTS delivers autonomic capabilities that help to reduce CPU (by eliminating some RUNSTATS jobs) and more up-to-date administration statistics for scheduling your DB2 utilities.

Perhaps after reading this column you will re-think your stance if yours is one of the shops that has yet to embrace RTS.

Autonomic Statistics

Real Time Statistics (RTS) is one of the first steps 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. As I already briefly mentioned, 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.

Now don't misunderstand; the real time statistics cannot entirely replace RUNSTATS. Although several of the statistics are similar, RTS is **never** used by the optimizer to determine access paths. But RTS can be used by DBAs to better administer DB2 databases.

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 outlined in Table 1. And the columns in the SYSIBM.**INDEXSPACESTATS** table are shown in Table 2.

Table 1. RTS Columns in SYSIBM.TABLESPACESTATS:

Column	Description
DBNAME	The name of the database.
NAME	The name of the table space.
PARTITION	The data set number within the table space.
DBID	The internal identifier of the database.
PSID	The internal identifier of the table space page set descriptor.
UPDATESTATSTIME	The timestamp when the row was inserted or last updated. in the TABLESPACESTATS table
TOTALROWS	The number of rows or LOBs in the table space or partition.
NACTIVE	The number of active pages in the table space or partition.
SPACE	The amount of space, in KB, that is allocated to the table space or partition.

EXTENTS	The number of extents in the table space or partition. Multi-piece table spaces = extents for the last data set.
LOADRLASTTIME	The timestamp of the last LOAD REPLACE on the table space or partition.
REORGLASTTIME	The timestamp of the last REORG on the table space or partition.
REORGINSERTS	The number of records or LOBs that have been inserted since the last REORG or LOAD REPLACE
REORGDELETES	The number of records or LOBs that have been deleted since the last REORG or LOAD REPLACE
REORGUPDATES	The number of rows that have been updated since the last REORG or LOAD REPLACE
REORGDISORGLOB	The number of LOBs that were inserted since the last REORG or LOAD REPLACE that are not perfectly chunked.
REORGUNCLUSTINS	The number of records that were inserted since the last

	REORG or LOAD REPLACE that are not well-clustered
REORGMASSDELETE	The number of mass deletes or dropped tables from a segmented table space, since the last REORG or LOAD REPLACE.
REORGNEARINDREF	The number of overflow records created since the last REORG or LOAD REPLACE relocated near the pointer record.
REORGFARINDEF	The number of overflow records created since the last REORG or LOAD REPLACE relocated far from the pointer record.
STATSLASTTIME	The timestamp of the last RUNSTATS on the table space or partition.
STATSINSERTS	The number of records or LOBs that have been inserted since the last RUNSTATS on the table space or partition.
STATSDELETES	The number of records or LOBs that have been deleted since the last RUNSTATS on the table space or partition.

STATSUPDATES	The number of rows that have been updated since the last RUNSTATS on the table space or partition.
STATSMASSDELETE	The number of mass deletes , or the number of dropped tables from a segmented TS, since the last RUNSTATS.
COPYLASTTIME	The timestamp of the last full or incremental image copy on the table space or partition.
COPYUPDATEDPAGES	The number of distinct pages that have been updated since the last COPY.
COPYCHANGES	The number of insert, delete, and update operations since the last COPY.
COPYUPDATELRSN	The LRSN or RBA of the first update after the last COPY.
COPYUPDATETIME	The timestamp of the first update after the last COPY.

Table 2. RTS Columns in SYSIBM.INDEXSPACESTATS:

Column	Description
DBNAME	The name of the database.
NAME	The name of the index space.
PARTITION	This column is used to map a data set number in an index space to its statistics.
DBID	The internal identifier of the database.
ISOBID	The internal identifier of the index space page set descriptor.
PSID	The internal identifier of the table space page set descriptor for the table space associated with the index
UPDATESTATSTIME	The timestamp when the row was inserted or last updated.
TOTALENTRIES	The number of entries, including duplicate entries, in the index space or partition.
NLEVELS	The number of levels in the index tree.
NACTIVE	The number of active pages in the index space or partition. This value is equivalent to the number of preformatted pages.
SPACE	The amount of space, in KB, that is allocated to the index space or

	partition.
EXTENTS	The number of extents in the index space or partition. On multi-piece index spaces=extents for the last data set.
LOADRLASTTIME	The timestamp of the last LOAD REPLACE on the index space or partition.
REBUILDLASTTIME	The timestamp of the last REBUILD INDEX on the index space or partition.
REORGLASTTIME	The timestamp of the last REORG INDEX on the index space or partition.
REORGINSERTS	The number of index entries that have been inserted since the last REORG, REBUILD INDEX, or LOAD REPLACE
REORGDELETES	The number of index entries that have been deleted since the last REORG, REBUILD INDEX, or LOAD REPLACE
REORGAPPENDINSERT	The number of index entries that have been inserted since the last REORG, REBUILD INDEX, or LOAD REPLACE
REORGPSEUDODELETES	The number of index entries that

	have been pseudo-deleted since the last REORG, REBUILD INDEX, or LOAD REP
REORGMASSDELETE	The number of times that an IX space/partition was mass deleted since the last REORG, REBUILD INDEX, or LOAD REP
REORGLEAFNEAR	The number near of index page splits that occurred since the last REORG, REBUILD INDEX, or LOAD REPLACE
REORGLEAFFAR	The number of far index page splits that occurred since the last REORG, REBUILD INDEX, or LOAD REPLACE
REORGNUMLEVELS	The number of levels in the index tree that were added or removed since the last REORG, REBUILD INDEX, or LOAD REP
STATSLASTTIME	The timestamp of the last RUNSTATS on the index space or partition.
STATSINSERTS	The number of index entries that have been inserted since the last RUNSTATS
STATSDELETES	The number of index entries that have been deleted since the last

	RUNSTATS
STATSMASSDELETE	The number of times that the index or index space partition was mass deleted since the last RUNSTATS.
COPYLASTTIME	The timestamp of the last full image copy on the index space or partition.
COPYUPDATEDPAGES	The number of distinct pages that have been updated since the last COPY.
COPYCHANGES	The number of insert or delete operations since the last COPY.
COPYUPDATELRSN	The LRSN or RBA of the first update after the last COPY.
COPYUPDATETIME	The timestamp of the first update after the last COPY.

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

One of the “problems” that you may encounter when embarking on your RTS implementation is the initialization of the columns in the RTS tables. All of the counters and timestamp columns are set to null, so they are basically useless until you initialize them. Oh, DB2 will try to externalize the RTS values based on STATSINT, but consider what will happen.

Let’s use the REORGINSERTS column as an example. It should contain the number of INSERTs since the last REORG. So say we have inserted 15 new rows. The STATSINT interval is hit and DB2 goes to externalize the data. It tries to add 15 to REORGINSERTS. But REORGINSERTS is currently NULL, so NULL plus 15 is NULL. Problem, right?

Therefore, for each object for which you want real-time statistics, the IBM DB2 Administration Guide manual directs that we need to run the appropriate utility (REORG, RUNSTATS, LOAD REPLACE, REBUILD INDEX, or COPY) to establish a base value from which

the delta value can be calculated. Of course, this is time-consuming and potentially disruptive. Another approach is to run RUNSTATS and then use the values from the system catalog to initialize the RTS values. To do this, you would:

- Set all counters to zero
- Set all dates to 0001-01-01-00.00.00.000000
- Set TOTALROWS and TOTALENTRIES to CARDF

Additionally, 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.

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 *IDUG Solutions Journal*, February 2007.

Home.