RTS is your friend – Optimize DB2 housekeeping

Click to edit Master subtitle style
Rich Wolfson – Principal Consultant
Agenda

- An overview of RTS - what does it buy you
- RTS parameters worth a closer look
- RTS integration with automation products - one step further towards 100% automation
- RTS usage to become even more proactive
- Best practices combining the power of RTS with CA Database Analyzer
RTS Overview - a historical view looking at the past

- One big issue from day one of DB2
  - When to execute a REORG, COPY, Space Administration, 

- RUNSTATS used to always update DB2 catalog statistics
  - If bad statistics -> OPTIMIZER decisions could be a disaster
  - Especially for incoming BIND or dynamic REBINDS
  - Dynamic SQL was pretty much “abandoned”

- Huge demand for automation tools
  - Gather VSAM level statistics w/o updating DB2 catalog statistics
  - Storing detailed statistics in separate repository
  - Intelligent decision making what to do based on statistics
  - Allowed for history reporting and trend analysis
rts
overview
What does it buy you
RTS Overview - a historical view looking at the past

- RUNSTATS introduced REPORT ONLY / UPDATE NONE
  - Allowed for homegrown applications to analyze the status for objects (tablespaces / indexes / partitions) and schedule needed REORG’s and/or space administration
  - DB2 Catalog not updated – performance issues eliminated
  - Still a huge need for automation tools to administer REORG’s, space administration, Image Copy processes etc.
RTS Overview - a historical view looking at the past

- DB2 RTS was “born” during the DB2 V7 lifecycle
  - Optional to enable / disable
- Enhanced for DB2 V8 – still optional
  - Performance issues a huge concern
  - Overhead of activating RTS kept many sites from enabling
- DB2 9 – not really an option anymore
  - RTS tables integrated with DB2 catalog
- Is there still a need for an automation tool?
  - Some will claim the RTS statistics are sufficient due to DSNACCOX/R
  - Still many reasons exist to REORG / COPY / RUNSTAT which cannot be determined by RTS thresholds
  - Sophisticated users / demands needs more (covered later)
RTS - let’s look at the DB2 RTS objects

- Very few objects make up the RTS feature
  - Amazing how much benefit you can get from just two tables
  - (please see next page)
CREATE TABLE "SYSIBM".TABLESPACE_STATS

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBNAME</td>
<td>CHARACTER(8)</td>
<td>NOT NULL</td>
</tr>
<tr>
<td>&quot;NAME&quot;</td>
<td>CHARACTER(8)</td>
<td>NOT NULL</td>
</tr>
<tr>
<td>&quot;PARTITION&quot;</td>
<td>SMALLINT</td>
<td>NOT NULL</td>
</tr>
<tr>
<td>DBID</td>
<td>SMALLINT</td>
<td>NOT NULL</td>
</tr>
<tr>
<td>&quot;PSID&quot;</td>
<td>SMALLINT</td>
<td>NOT NULL</td>
</tr>
<tr>
<td>UPDATESTATETIME</td>
<td>TIMESTAMP</td>
<td>NOT NULL</td>
</tr>
<tr>
<td>TOTALROWS</td>
<td>FLOAT(53)</td>
<td></td>
</tr>
<tr>
<td>NACTIVE</td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>SPACE</td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>EXTENTS</td>
<td>SMALLINT</td>
<td></td>
</tr>
<tr>
<td>LOADRLASTTIME</td>
<td>TIMESTAMP</td>
<td></td>
</tr>
<tr>
<td>REORGLASTTIME</td>
<td>TIMESTAMP</td>
<td></td>
</tr>
<tr>
<td>REORGINSECTS</td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>REORGDELETES</td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>REORGUPDATESECONDS</td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>REORGCOUNTS</td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>REORGUNCLUSTINS</td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>REORGDISORGLOB</td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>REORGMASSDELETE</td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>REORGNEARINDREF</td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>REORGFARINDREF</td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>STATSLASTTIME</td>
<td>TIMESTAMP</td>
<td></td>
</tr>
<tr>
<td>STATSINSERTS</td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>STATSDELETES</td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>STATSHISTORY</td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>STATSMASSDELETE</td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>COPYLASTTIME</td>
<td>TIMESTAMP</td>
<td></td>
</tr>
<tr>
<td>COPYUPDATETIME</td>
<td>TIMESTAMP</td>
<td></td>
</tr>
<tr>
<td>COPYUPDATEDPAGES</td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>COPYCHANGES</td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>COPYUPDATELRSN</td>
<td>CHARACTER(6) FOR BIT DATA</td>
<td></td>
</tr>
</tbody>
</table>

) IN DSNRTSDB.DSNRTSTS ;
CREATE TABLE "SYSIBM".INDEXSPACESTATS

(DBNAME CHARACTER(8) NOT NULL,
INDEXSPACE CHARACTER(8) NOT NULL,
"PARTITION" SMALLINT NOT NULL,
, DBID SMALLINT NOT NULL,
,"ISOBID" SMALLINT NOT NULL,
,"PSID" SMALLINT NOT NULL,
,UPDATESTATSTIME TIMESTAMP,
,TOTALENTRIES FLOAT(53),
,NLEVELS SMALLINT,
,NACTIVE INTEGER,
,SPACE INTEGER,
,EXTENTS SMALLINT,
,LOADRLASTTIME TIMESTAMP,
,REBUILDLASTTIME TIMESTAMP,
,REORGLASTTIME TIMESTAMP,
,REORGINSERTS INTEGER,
,REORGDELETES INTEGER

,REORGAPPENDINSERT INTEGER
,REORGPSUEDODELETES INTEGER
,REORGMASSDELETE INTEGER
,REORGLEAFNEAR INTEGER
,REORGLEAFFAR INTEGER
,REORGNUMLEVELS INTEGER
,STATSLASTTIME TIMESTAMP
,STATSINSERTS INTEGER
,STATSDELETES INTEGER
,STATSMASSDELETE INTEGER
,COPYLASTTIME TIMESTAMP
,COPYUPDATEDPAGES INTEGER
,COPYCHANGES INTEGER
,COPYUPDATELRSN CHARACTER(6) FOR BIT DATA
,COPYUPDATETIME TIMESTAMP
)

IN DSNRTSDB.DSNRTSTS;
DB2 9 added columns

- For SYSTABLESPACE STATS
  - INSTANCE
  - NPAGES  < - - - - Can help with work dataset sizing
  - DATASIZE
  - UNCOMPRESSEDDATASIZE
  - IBMREQD  < - - - - Good proof that RTS integrated into the catalog

- For SYSINDEXSPACE STATS
  - INSTANCE
  - LASTUSED  !!!!!!!!!!!!!!!!
  - NPAGES
  - NLEAF
  - NAME  < - - - - Used to only have INDEXSPACE
  - CREATOR
  - IBMREQD
What is the RTS “Magic”

- A “hidden secret” for many years
- Control blocks in memory gathering statistics of SQL processing
- Even though RTS disabled – control blocks still updated “on the fly”
  - DB2 V7 and V8 offered the option to enable
  - DB2 9 not an option anymore - - - - > RTS is THE future
- When RTS enabled -> control blocks externalized to DB2-tables mentioned previously
  - Externalization controlled by a number of possible events
RTS Externalization

- **-STOP DB(userdb) SPACE(space) PART(x)**
  - The control block statistics for that specific object will be externalized
  - STOP will succeed even if externalization fails (e.g. SQL-904)

- **-STOP DB(DSNDB06) SPACE(xxxx)**
  - The entire subsystem’s control blocks externalized

- **-STOP DB2 MODE(QUIESCE)**
  - MODE(FORCE) doesn’t externalize collected statistics and the control block content is gone

For Data Sharing, each member will update the RTS tables in a serial process since each member has it’s “private” interval timer.
RTS - what is maintained / NOT maintained

- DSNDB06 (catalog) statistics maintained
- DSNDB01, SYSLGRNX, SYSUTILX only maintained for some operations (like COPY)
- TRACKER site statistics not maintained
- LISTDEF used with RTS objects included - - > nothing is externalized
- Certain counters maintained for work files / DSNDB04
  - EXTENTS, NACTIVE, SPACE
- Objects not modified - > EXTENTS, NACTIVE, SPACE are not externalized

A DB2® tracker site is a separate DB2 subsystem or data sharing group that exists solely for the purpose of keeping shadow copies of your primary site's data. No independent work can be run on the tracker site.
RTS - what is maintained / NOT maintained

- Some utilities update RTS tables
  - Copy, Runstats, Reorg, Load, Rebuild

- Other utilities can “fool” you
  - RECOVER and DSN1COPY for example

- IBM recommended process:
  - STOP the pageset being touched ☐ current statistics will be externalized from control blocks
  - Counters in the control block will be set to ZERO
  - Upon completion of the utility
    - Use SQL statements to update the appropriate counters in the RTS tables (NACTIVE, PAGES, NLEAF etc.)
    - Use SQL to update the TIMESTAMP columns
    - Use SQL to set the counters for SQL processing to ZERO
    - Alternatively execute REORG, COPY and RUNSTATS
- DROP TABLESPACE / INDEX

- The related RTS statistics are gone (unless the RTS objects are unavailable)
  - If another object with SAME DBID and PSID created, the rows are re-initialized
  - Use SQL DELETE to manually clean up if needed

- The previous topics raises the question:

  What about HISTORY information

  - We will cover this topic later.
RTS Parameters
Worth a Closer Look
Looking at what kind of RTS information is stored in the two tables – and the DB2 housekeeping processes we need to maintain, these are the most interesting tasks:

1) Image Copy
2) Reorg
3) Runstats
4) Space Administration (aka. ALTER PRIQTY / SECQTY)

The RTS tables also hold information related to REBUILD and LOAD.
RTS parameters worth a closer look - COPY

- Image Copy processing – really depends on the backup strategy in place and execution method
  - If full image copies executed on a frequent basis (every day, every week, .... ) - why bother about RTS.
  - Dynamic decision to do INCREMENTAL / FULL can be satisfied via COPY syntax.
  - Image copies based on size can be handled using TEMPLATE switching
  - If COPY parameters, skip image copy, sophisticated IF-ELSE logic or the like needed - then the RTS parameters can be useful
RTS parameters worth a closer look - COPY

- COPYLASTTIME: last IC timestamp
- COPYUPDATEDPAGES: # pages updated since last IC
- COPYCHANGES: # rows manipulated since last IC
- COPYUPDATELRSN: first U/I/D after last IC
- COPYUPDATETIME: first U/I/D after last IC
- SPACE
- NPAGES
- DATASIZE

If table space size needs to be considered for media
RTS parameters worth a closer look - Runstats

- Runstats (and especially associated potential REBIND) is always a good / HOT discussion topic.

- After LOAD REPLACE?
  - If amount of data and cardinality hasn’t changed dramatically – and access path used are OK – why?
- Every week / month / three months?
- Never?
- After REORG?
  - If Optimizer statistics were all right prior to reorg – why?
  - Did cardinality change a lot?
  - Even if data doubled since last RUNSTATS – does 120M make a difference compared to 240M – maybe!!
- When “things” have changed dramatically – probably!!
RTS parameters worth a closer look - Runstats

- **STATSINSERTS**  
  #rows inserted since last

- **STATSUPDATES**  
  #rows updated since last (perhaps)

- **STATSDELETES**  
  #rows deleted since last (perhaps)

- **STATSMASSDELETES**  
  #mass deletes executed (not rows)

Note that MASSDELETES is a counter for the number of executions – not if anything was deleted or not (or rolled back)

- **DSNACCOX** has a number of recommended thresholds

- Prefer to look at other parameters?
  - History / delta changes for NLEAF, NPAGES, NACTIVE, TOTALROWS over a period of time

  *(we will get back to the HISTORY issue later)*
RTS parameters worth a closer look - REORG

- Reorganizations are still performed in a number of different ways
  - Fixed reorgs on a periodic basis – RTS not needed for these.
    - Performed weekly, monthly, yearly – many reasons and these unconditional reorgs “might be all right”
    - Some see these as “waste of resources” - or even insufficient !
  - Based on predefined thresholds
    - IBM reorg has had a few for a while
    - Automation products have hundreds of customizable conditions (used to be based on VSAM level dataset collected statistics only)
    - RTS parameters can really assist in finding the best candidates where reorg is needed - or getting close to be needed (another scenario where history information can be very helpful)
RTS parameters worth a closer look - REORG

- **EXTENTS**  #extents for object / partition !
- **REORGINSERTS**  #rows inserted since last ?
- **REORGUPDATES**  #rows updated since last ?
- **REORGDELETES**  #rows deleted since last ?
- **REORGUNCLUSTINS**  #rows inserted out-of-sequence
- **REORGMASSDELETES**  #unqualified delete statements ?
- **REORGFARINDREF**  #rows inserted more than 16 pages away
- **REORGNEARINDREF**  #rows inserted less than 16 pages away

For the INDREF counters, special calculation for segmented tablespaces: NEAR/FAR limit = 2 times SEGSIZE
RTS parameters worth a closer look - REORG

- INDEX reorgs have a few additional parameters

- REORGAPPENDINSERT  #entries inserted with higher key ?
- REORGPSEUDODELETES  #entries pseudo-deleted !
- REORGNUMLEVELS  #removed/added levels
RTS parameters worth a closer look - ALTER

- RTS might provide sufficient information – some procedure still needed to:
  - Complete automate space administration
  - Provide flexibility to handle site specific standards
    - No pageset over xxx cylinders
    - Combining PRIQTY with SEGSIZE / PIECESIZE
  - Take current size into consideration when decision made to reclaim over-allocation
  - Change allocation up/down – or don’t change
  - When to use sliding scheme (negative 1)
RTS parameters worth a closer look

- Some parameters very interesting – despite not being related to previous housekeeping processes

- **LASTUSED** – most recent date an index was referenced
  - Access Path used for SQL SELECT or FETCH
  - Used for DELETE / UPDATE WHERE CURRENT OF
  - Enforcing RI
  - This is cool news – easier in “The Good Ole Days” when we could use SYSPACKDEP to some degree
    - Even this wasn’t sufficient due to old statements never executed

- **DATASIZE / UNCOMPRESSEDDATASIZE**
  - RTS can be used to calculate space needed for UNLOAD jobs
RTS integration with automation products - one step further towards 100% automation
Automation tools help towards better and more automation

- **DSNACCO(R/X)**
  - Relatively easy to get started and automate many housekeeping procedures.
  - Based solely on RTS statistics.
  - Thresholds customizable to meet your standards

- **Vendor products – are they still relevant?**
  - Used to be very CPU intensive collecting detailed statistics from VSAM datasets
  - RTS is now an integrated piece – combination of old/new method pretty powerful due to a number of reasons
  - DSNACCO(R/X) doesn’t automate everything
  - Flexibility – sophisticated housekeeping processes and control
Where doesn’t RTS provide all the help I need

- Which indexes can benefit from decreased NLEVELS if reorganized / rebuilt
- Which objects are over-allocated by a certain percentage
- Which objects will have changed cardinality if RUNSTATS executed (when is RUNSTATS really needed)
  - This always opens up another “can of worms”
- Space automation / alteration during reorg where “one size fits all” isn’t good enough.
- Dynamic decision of utility parameters based on object characteristics
- RTS doesn’t provide history information “out-of-the-box”
RTS usage to become even more proactive
Proactive approach - ideas

- Become PROACTIVE - reason why REORG / ALTER - the "WHY's" and "WHEN's"

  - Capture the thresholds violating the conditions used to "trigger" the reorganizations
  - Keep these in a DB2 table
    - Object name, partition, timestamp, condition, threshold and actual value which caused the reorg to happen
  - Periodically analyze the content to see which objects are triggered the most
    - Analyze the condition violated - often the same for a specific object
    - Analyze if the object's physics can be modified to eliminate some reorgs - perhaps modify the condition to trigger reorg more frequently
  - The BEST reorg is the reorg NEVER executed
Proactive approach - ideas

- One method is to simply keep a copy of the RTS tables with one additional column, “TIMESTAMP NNWD”

  INSERT INTO MY.SYSTABLESPACESTATS (naming all columns except TS)
  SELECT all-columns FROM SYSIBM.SYSTABLESPACESTATS;

  INSERT INTO MY.SYSINDEXSPACESTATS (naming all columns except TS)
  SELECT all-columns FROM SYSIBM.SYSINDEXSPACESTATS;

- Works well when all reorgs generated and executed within a small window (e.g. every weekend)

- A challenge when reorgs are scattered all over
Proactive approach - ideas

- What and when to keep RTS history?
- Insert everything:
  - Tables will grow dramatically – consider purge process
  - More difficult to see what has been triggered IF and WHEN
  - Easy to implement
- Selective approach:
  - As part of the scheduled/executed reorg – insert ONLY those value(s) which violated the conditions
  - Only executed reorgs will be monitored
  - The “DBA table” could have object, partition, timestamp, reorg reason (RTS column) and the threshold violated
  - One reorg could result in multiple rows when +1 condition violated
Proactive approach - ideas

- So far only REORG covered
  - No need to capture reason for COPY

- Having periodic “snap shot” of some RTS values make sense to monitor RUNSTATS
  - One method to do trend analysis
  - Might be used to evaluate when to schedule Runstats
  - Might be useful to monitor when REBIND’s should be evaluated and monitored

- Keeping history of space allocation
  - When trend analysis is needed
  - If “sliding scheme” isn’t used for everything – but used selectively, keeping history helps automate space administration
Best practices combining the power of RTS with CA Database Analyzer
Past Philosophy

- Automation tools like CA Database Analyzer (PDA) used to collect VSAM level statistics and store these in a repository
  - Statistics well beyond what Runstats and RTS provides
- Current – or delta statistics analyzed to determine which pre-defined actions should be generated
  - Based on pre-defined conditions
  - Provides ability to automatically do detailed analysis of statistics and generate the housekeeping jobs needed
- One down-side collecting these detailed statistics - CPU intensive (like Runstats but less)
  - As DB2 environments grow in terms of number of objects as well as object size – this process has become less attractive
RTS changed the picture

- Less need for frequent statistics collection
  - RTS has most of the details needed
  - DSNACCOX has quite a few capabilities to automate
- Automation tools still have a lot of benefits
  - Flexibility
  - Customizability
  - Benefit from “best of two worlds” – RTS and “good ole VSAM stats”
- We’ll have a look at how customers are squeezing the most out of RTS still maintaining the flexibility automating housekeeping processes using an automation tool like PDA.
CA Database Analyzer exploiting RTS

- **Weekly / bi-monthly usage of RTS thresholds** scheduling housekeeping jobs (*mostly REORG and ALTER*)
  - not collecting VSAM level statistics
  - Will solve most of reasons “why to reorg”

- **Monthly (or every second month) execute the old VSAM level statistics collection**
  - Allows for history reporting
  - Allows for more sophisticated decisions to reorg
    - Decrease number of index levels
    - Space savings (cosmic reorg factor)
    - Any homegrown conditions / site specifics

- Let’s look at the most common usage and why . . . . . .
Define scope and thresholds

- Filter as much as possible as early as possible
Define scope and thresholds

Define thresholds to be checked – only objects violating thresholds will be included in Extract Proc.

No need for stats collection

If the supplied RTS conditions or PDA conditions are insufficient – define your own using native SQL.
Homemade conditions

- Select from anything, join, subselect, combine AND/OR, “not in select from DBA table”
Define the actions / utilities needed

- Image Copy, Space Alteration, Reorg and User Defined action (which could be RTS history)
Potentially define ACTION thresholds

- These conditions will provide additional filtering to the Extract Conditions.
- %USERRx variables will transfer values to Utility Models
Example of ALTER model

- More sophisticated space administration where “one size doesn’t fit all”.
  - Implement site specifics
  - 100% space automation – no need for manual processes

```plaintext
//BPIIPT DD *
#if(%REORGP,GT,250000)
  #if(%PQTY,LT,250000)
    ALTER %OBJTYPE %CREATOR..%OBJECT %PARTLIT %PART  PRIQTY 250000 SECQTY -1
  #endif
#if(%PQTY,GT,250000)
  #if(%SQTY,LT,131068)
    ALTER %OBJTYPE %CREATOR..%OBJECT %PARTLIT %PART  SECQTY 131068
  #endif
#if(%REORGP,LT,250000)
  #if(%REORGP,GT,12)
    ALTER %OBJTYPE %CREATOR..%OBJECT %PARTLIT %PART  PRIQTY %CALC1 SECQTY %CALC2
  #endif
#if(%REORGP,LT,250000)
  #if(%REORGP,GT,12)
    ALTER %OBJTYPE %CREATOR..%OBJECT %PARTLIT %PART  PRIQTY %CALC1 SECQTY %CALC2
  #endif
#endif
//BPIOPT DD *
.list sysout(*)
.connect %sysid
```
Customize the model to make dynamic decisions

All selected partitions included in one reorg – based on the ratio between total partitions and partitions needing reorg, a decision is made whether to REBUILD or UPDATE the NPI’s. Also – dynamically decide whether to do online reorg or not. All your needs and site specifics can be implemented and automated.
User Defined model – inserting history to eliminate reorg’s (being proactive)

- Tracking reasons for reorgs provides the ability to change the object attributes – or schedule more reorgs

```c
/* UNICENTER BATCH PROCESSOR USED TO INSERT INTO THE DBA PROACTIVE TABLE.*/
/*%STEPNAME EXEC PGM=PTLDRIVM,REGION=5M,PARM='%SUFFIX.EP=BPLBCTL'*/

#include steplib //STEPLIB DD DSN=CAI.LOADLIB(S)
#include ptilib //PTILIB DD DSN=CAI.LOADLIB(S)
#include ptiparm //PTIPARM DD DSN=CAI.PARMLIB
//SYSOUT DD SYSOUT=*
//UTPRINT DD SYSOUT=*
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSREC DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SORTOUT DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//BPIIPT DD *
#if (%VALTRIG, EQ, ?)
  #SET %O = ‘TS’
#else
  #SET %O = ‘IX’
#endif

INSERT INTO RASST02.PDA_PROACTIVE
  VALUES (’%DBNAME’, ’%OBJECT’, ’%O’, 0%PARTPHY, 0%PARTLOG, ’%CONDTXT’, %VALTRIG, ’%DATECLTS’);
//BPIOPT DD *
.list sysout(*)
.connect %sysid
```
- QUESTIONS?
A special thanks to Steen Rasmussen for this presentation.

Thank you

richard.wolfson@ca.com