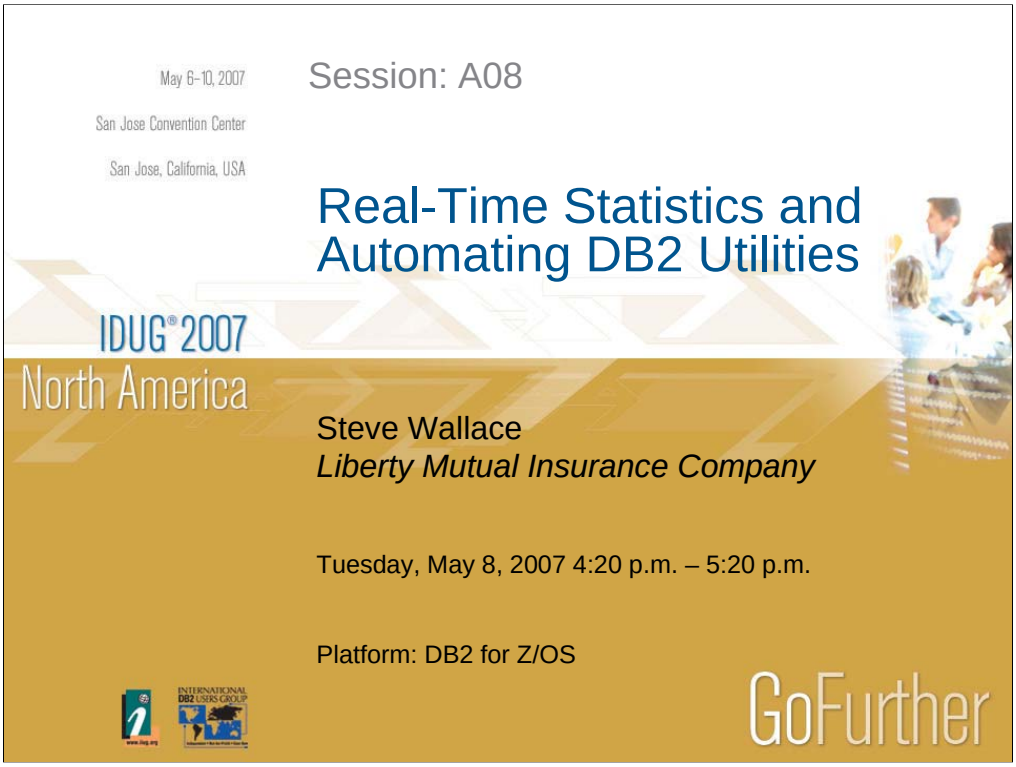


We will discuss briefly what real-time statistics (RTS) are, how the statistics are externalized, and certain benefits a DBA can realize by using the real-time statistics information. My presentation will also discuss how to utilize the information contained in the real-time statistics' tables in order to automatically generate DB2 utilities for the DBA (no manual intervention!). The utilities that we will discuss automating are full image copies, runstats, and reorgs. IBM has included a stored procedure called DSNACCOR that analyzes the information contained in the real-time statistics' tables, and recommends what utility operations should be run on DB2 objects based on that information. I will discuss how to call DSNACCOR using a REXX program, generate reports based on DSNACCOR information, identify objects in a restricted status, and how to automatically create the necessary utility control cards. I will discuss how to override default DSNACCOR formulas to tailor DSNACCOR recommendations to company specifications.

Main Presentation Topics

- A. What are real-time statistics?
- B. DSNACCOR Stored Procedure.
- C. REXX Program Specifics.
- D. Benefits of real-time statistics.
- E. RTS Experiences.

A. What are real-time statistics?
1. DB2 Tables.
2 .Externalization of RTS.
3. RTS Data Interpretation.
4. Counter initialization - why?

B. DSNACCOR Stored Procedure.
1.Formulas.
2. Authorizations.

C. REXX Program Specifics.
1. Calling DSNACCOR in REXX.
2. Override DSNACCOR Default Formulas.
3. Tailor DSNACCOR for specific utilities.
4. Accessing DB2 Catalog for other information.
5. Automatically build image copy GDG bases.
6. Reporting - Why are Objects flagged for a utility?
7. Discuss differences in generating the 3 utilities.

D. Benefits of real-time statistics.
1. Accurate DB2 maintenance.
2. Automation of utilities.
3. Reduction of waste.
4. Improved application performance.

E. RTS Experiences.
1. Significant CPU savings related to "old" processes.
2. Utilities processed based on real-time need.
3. Objects in restricted status .

What are Real-time statistics (RTS)

- Statistics automatically generated.
- Existence of two DB2 tables allows for easy interpretation.
- Statistics externalized based on certain events.
- Counters! Very Important!

Introduced in DB2 Version 7, DB2 generates statistics in memory for each DB2 object.

The statistics are located in 2 different DB2 tables and are externalized to the DB2 tables every 30 minutes (default), or during certain DB2 database operations.

The real-time statistics' tables are Sysibm.Tablespacestats and Sysibm.Indexspacestats.

In memory statistics are still kept, even if the real-time statistics' tables do not exist. There is virtually no overhead associated with having the statistics written out to the DB2 tables.

Certain "counters" reflect amount of object level activity that has occurred since the last DB2 utility operation has been performed (copy, runstat, reorg).

Upon implementing real-time statistics, counters need to be initialized, or the information contained in Tablespacestats and/or Indexspacestats could become distorted and/or inaccurate.

Real-time statistics – DB2 Objects

Object name	Description
DSNRTSDB	Database for real-time statistics objects
DSNRTSTS	Tablespace for real-time statistics objects
SYSIBM.TABLESPACESTATS	Table for statistics on tablespaces and tablespace partitions
SYSIBM.INDEXSPACESTATS	Table for statistics on indexspaces and indexspace partitions
SYSIBM.TABLESPACESTATS_IX	Unique index on SYSIBM.TABLESPACESTATS (columns DBID, PSID, and PARTITION)
SYSIBM.INDEXSPACESTATS_IX	Unique index on SYSIBM.INDEXSPACESTATS (columns DBID, ISOBID, and PARTITION)

Source: appendix 1.7.1.1 "DB2 UDB for Z/OS V8 Administration Guide

This chart explains the DB2 objects that are necessary to have in place so that real-time statistics can be externalized, and easily interpreted.

Externalization of real-time statistics

- When you issue STOP DATABASE(DSNRTSDB).
- When you issue STOP DATABASE(*database-name*) SPACENAM(*space-name*).
- At the end of the time interval that you specify during installation ("Statsint" parameter).
- When you issue STOP DB2 MODE(QUIESCE).
- During utility operations.

Source: Appendix 1.5.3.1 "DB2 UDB for Z/OS V8 Administration Guide

Externalization refers to DB2 actually writing the in memory statistics out to the Sysibm.Tablespacestats and Sysibm.Indexspacestats.

The default for the Statsint parameter is 30 minutes. The default can be modified if necessary.

RTS Data Interpretation

- Counters reflect amount of inserts, updates, or deletes since the last utility run (either image copy, reorg, or runstat).
- Other information located within the real-time statistics tables include database, object name, partition, space, totalrows/totalentries, space, and extents.
- Updatestatstime reflects the time a row was inserted or was last updated.

SYSIBM.TABLESPACESTATS DATA EXAMPLES:

```
DBNAME   NAME    UPDATESTATSTIME             TOTALROWS             SPACE   EXTENTS
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+
TESTDB   TESTTS   2007-01-09-10.24.58.535725  +0.2905700000000000E+05    5760     12

DBNAME   NAME    STATSLASTTIME          STATSINSERTS  STATSDELETES  STATSUPDATES
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+
TESTDB   TESTTS   2006-08-23-01.14.03.900053       3889          378             1338

DBNAME   NAME    COPYLASTTIME           COPYUPDATEDPAGES  COPYCHANGES
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+
TESTDB    TESTTS   2007-01-02-04.32.29.604342           85                      261

DBNAME   NAME    REORGLASTTIME          REORGINSERTS    REORGDELETES  REORGUPDATES
-------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+
TESTDB   TESTTS   2006-06-18-15.31.45.468624       6922                  1670              1961
```

This is not an all inclusive list of the columns contained on the Tablespacestats table. For a full definition of both the Tablespacestats and Indexspacestats tables, please refer to Appendix 1.5.2 in the DB2 UDB for Z/OS V8 Utility Guide and Reference manual.

Sysibm.Indexspacestats shows similar information for indexspaces.

RTS Counter Initialization

- Upon implementing real-time statistics, it is critical that certain values be initialized on both Tablespacestats and Indexspacestats.
- Initialization allows precise interpretation of how much activity is occurring your DB2 objects.
- Inaccurate initialization, or failing to initialize, could lead to wastefulness in executing DB2 utilities, or not processing DB2 utilities when utilities would be necessary.
- Initialization of Indexspacestats would be similar to the Tablespacestats initialization show in the notes page.


```
UPDATE SYSIBM.TABLESPACESTATS
SET REORGLASTTIME = '0001-01-01-00.00.00.000000‘
  ,REORGINSERTS = 0
  ,REORGDELETES = 0
  ,REORGUPDATES = 0
   ,REORGMASSDELETE = 0
 WHERE REORGLASTTIME IS NULL;
UPDATE SYSIBM.TABLESPACESTATS
 SET STATSLASTTIME = '0001-01-01-00.00.00.000000'
   ,STATSINSERTS = 0
  ,STATSDELETES = 0
   ,STATSUPDATES = 0
   ,STATSMASSDELETE = 0
 WHERE STATSLASTTIME IS NULL;
UPDATE SYSIBM.TABLESPACESTATS
 SET COPYLASTTIME = '0001-01-01-00.00.00.000000'
   ,COPYUPDATEDPAGES = 0
   ,COPYCHANGES = 0
 WHERE COPYLASTTIME IS NULL;
UPDATE SYSIBM.TABLESPACESTATS A
   SET TOTALROWS =
     (SELECT B.CARDF FROM SYSIBM.SYSTABLEPART B
      WHERE A.DBNAME=B.DBNAME
      AND   A.NAME = B.TSNAME
      AND   A.PARTITION=B.PARTITION)
  WHERE  A.TOTALROWS IS NULL;
```

DSNACCOR Stored Procedure

- IBM Supplied Stored Procedure.
- DSNACCOR recommendations
- Sysibm.Tablespacestats / Sysibm.Indexspacestats
- Recommends when you should reorganize, image copy, or update statistics for tablespaces or indexspaces.
- Indicates whether objects are in a restricted status.
- Indicates tablespaces or indexspaces that have exceeded their data set limits (extents).
- Recommendations based on general formulas.

Source: IBM DB2 UDB for Z/OS
Version 8 Administration Guide

IBM supplies a Stored Procedure called DSNACCOR.

This Stored Procedure aids the DBA in helping to maintain DB2 databases by suggesting certain DB2 utility actions.

Recommendations include when to reorganize, image copy, update statistics, indicate objects that are in a restrictive status, or warn the DBA if a DB2 dataset is in a high number of dataset extents.

DSNACCOR utilizes the information from Sysibm.Tablespacestats and Sysibm.Indexspacestats in order to make its recommendations.

DSNACCOR recommendations are based on certain “canned” formulas that can be customized in order to suit application needs.

All objects in a DB2 subsystem are analyzed by DSNACCOR, but this can be overridden.

DSNACCOR Formulas

- Full Image Copy on tablespace.
- Full Image Copy on indexspace.
- Incremental Image Copy on tablespace.
- Reorg on tablespace.
- Reorg on indexspace.
- Runstats on tablespace.
- Runstats on indexspace.
- Warning that dataset extent threshold exceeded.
- DSNACCOR formulas:
  - Appendix 1.2.5.5 of the DB2 UDB for Z/OS Utility Guide and Reference Manual.

Full Image Copy on Tablespace Formula:
((*QueryType*='COPY' OR *QueryType*='ALL') AND (*ObjectType*='TS' OR *ObjectType*='ALL') AND *ICType*='F') AND (COPYLASTTIME IS NULL OR REORGLASTTIME>COPYLASTTIME OR LOADRLASTTIME>COPYLASTTIME OR (CURRENT DATE-COPYLASTTIME)>*CRDaySncLastCopy*
OR (COPYUPDATEDPAGES*100)/NACTIVE>*CRUpdatedPagesPct*
OR (COPYCHANGES*100)/TOTALROWS>*CRChangesPct*)

Reorg on Tablespace Formula:
((*QueryType*='REORG' OR *QueryType*='ALL') AND (*ObjectType*='TS' OR *ObjectType*='ALL')) AND (REORGLASTTIME IS NULL OR ((REORGINSERTS+REORGDELETES+REORGUPDATES)*100)/TOTALROWS >*RRTInsDelUpdPct*
OR (REORGUNCLUSTINS*100)/TOTALROWS>*RRTUnclustInsPct*
OR (REORGDISORGLOB*100)/TOTALROWS>*RRTDisorgLOBPct*
OR ((REORGNEARINDREF+REORGFARINDREF)*100)/TOTALROWS>*RRT IndRefLimit* OR REORGMASSDELETE>*RRTMassDelLimit*
OR EXTENTS>*ExtentLimit*)

Runstats on Tablespace Formula:
((*QueryType*='RUNSTATS' OR *QueryType*='ALL') AND (*ObjectType*='TS' OR *ObjectType*='ALL')) AND (STATSLASTTIME IS NULL OR (((STATSINSERTS+STATSDELETES+STATSUPDATES)*100)/TOTALROWS> *SRTInsDelUpdPct*
AND (STATSINSERTS+STATSDELETES+STATSUPDATES)>*SRTInsDelUpd Abs*) OR STATSMASSDELETE>*SRTMassDeleteLimit*)

DSNACCOR Authorizations

- To call DSNACCOR, "caller" must have one or more of the following:
  - The EXECUTE privilege on the package for DSNACCOR.
  - Ownership of the package.
  - PACKADM authority for the package collection.
  - SYSADM authority.

as well as the following:

1. SELECT authority on the real-time statistics tables.
2. The DISPLAY system privilege.

Source: Appendix 1.2.5.2 "DB2 UDB for Z/OS V8 Utility Guide and Reference

This slide covers what types of authorizations need to exist in order to use DSNACCOR.

One of the four bulleted items need to be in place, as well as both:

1) Select authority on the RTS tables.

2) Display system privilege.

Determine DSNACCOR Actions

- Determine Query Type. What utility recommendations do you want DSNACCOR to make?
- Determine Object Type.
- If image copy, determine ICTYPE.
- Determine Criteria – Narrow list of objects down to what you what recommendations for.

The valid query types are: All, Copy, Runstats, Reorg, Extents, or Restrict.

The valid object types are: All, TS, or IX.

For image copies, the valid values are: F(ull),I(incremental),B(oth).

You can include, or exclude whatever tablespaces, or databases, you want. For example, for full image copies, we have 10 different jobs that run the REXX program that generates full image copies. Each of these jobs either includes, or excludes, certain databases. In all cases, the system level databases are excluded, as those are copied and maintained by another department.

Call DSNACCOR in REXX – Example 1

DSNACCOR Recommendations for Full Image Copy on Tablespace:

```
L1   = 'COPY' 'RESTRICT'   /* QUERY TYPE */
L1I  = 0
L2   = TS                  /* OBJECT TYPE */
L2I  = 0
L3   = F                   /* IC TYPE */
.
.
L8   = "DBNAME IN('TESTDB1','TESTDB2')"
L8I  = 0
.
.
L10  = 15                  /* Percentage of Updated Pages (Default is 20)    */
L10I = 0
L11  = 15                  /* Percentage of Total Changes  (Default is 10)    */
L11I = 0
L12  = 10                  /* Number of days since last copy (Default is 7)   */
L12I = 0
```


12

In this REXX, we are telling DSNACCOR that we want to identify tablespace objects in either the TESTDB1 or TESTDB2 databases that are in need of a full image copy.

In addition, we want to identify any object in either of those two databases that are in a restricted status. The program takes the tablespace objects that are in a restrictive state (unless those objects are in "RW,COPY" status), and writes those objects out to another dataset. This dataset gets emailed to the DBA group, identifying restricted status objects. These restricted status objects are not copied.

Objects that exist in a "RW,COPY" status are automatically copied to remove the copy pending flag.

The REXX program is coded in such a manner that any tablespace object that has a percentage of updated pages that are 15% or more of the total number of object pages since the last image copy to be flagged for a full image copy.

In addition, any tablespace object where the number of inserts, updates, and deletes are greater than 15% of the total number of rows that exist for the object since the last image copy is identified to be full image copied.

Lastly, if the last full image copy performed on an object was 10 days ago, or later, that tablespace object is flagged for a full image copy.

Example 1 - Continued

```
DO UNTIL(SQLCODE ¬= 0)
  ADDRESS DSNREXX "EXECSQL FETCH CRS102 INTO " X
  IF SQLCODE = 0 THEN
    DO
     DBNAME              = O01
     NAME                = O02
     PARTITION           = O03
     OBJECTTYPE          = O04
     OBJECTSTATUS        = O05
     IMAGECOPY           = O06
     COPYLASTTIME        = O12
     CRUPDPGSPCT         = O15
     CRCPYCHGPCT         = O16
     CRDAYSCELSTCPY = O17
    X1 = LEFT(DBNAME,8) LEFT(NAME,8) right(PARTITION,4) "    "
    X2 = LEFT(OBJECTTYPE,2) " " LEFT(OBJECTSTATUS,35) " "
    X3 = LEFT(IMAGECOPY,3) RIGHT(COPYLASTTIME,26) "  "
    X4 =  RIGHT(CRUPDPGSPCT,7) "              "
    X5 = RIGHT(CRCPYCHGPCT,7) "              " RIGHT(CRDAYSCELSTCPY,3)
```


DSNACCOR is called by the REXX, passing all 39 parameters, including the parameters that were specified on the previous slide.

The output of the DSNACCOR result set is stored in the “O” variables.

The “X” variables are used for reporting purposes, and are written to a separate report file.

These report files allow us to determine exactly why a tablespace object has been flagged as a full image copy candidate.

Example 1 - Continued

```
IF LEFT(O05,1) = '00'x|(POS('TS=RW,COPY',O05) > 0 & POS('-',O12) = 0 )
  THEN DO
    CNTR = CNTR + 1
    STEPCNTR = STEPCNTR + 1
    LDBNAME.CNTR = WORD(L1,1)
    LTSNAME.CNTR = WORD(L1,2)
    LPART.CNTR   = WORD(L1,3)
    LINEO.1 = X1 || X2 || X3 || X4 || X5
    CALL SET_DSN
    CALL COPY_JCL
    "EXECIO 1 DISKW OUT1 (STEM LINEO."
   END
ELSE DO
   RETC = 2
   LINEO.1 = X1 || X2 || X3 || X4 || X5
   "EXECIO 1 DISKW OUT3 (STEM LINEO."
END
```


If the O05 variable (status), is clear, OR if the tablespace object is in “RW,COPY”, processing continues. Counters are incremented, a line is written out to report showing what objects will be full image copied. A call is made to 2 different sub-routines. Subroutine SET_DSN sets the image copy dataset names, and subroutine COPY_JCL writes out the actual JCL to another file. The COPY_JCL subroutine also calls the CREATE_GDG subroutine to check for the existence of the proper GDG base.

If the tablespace object is in a restricted state, a line is written to a restricted object report, and forces the batch job to receive a return code of “2” in the JCL step. The return code of 2 signifies that object(s) are in a restricted status, and forces another JCL step to run emails the DBA group a restricted objects report.

Call DSNACCOR in REXX – Example 2

DSNACCOR Recommendations for Runstats on Tablespace:

```
L1   = 'RUNSTATS' 'RESTRICT'   /* QUERY TYPE */
L1I  = 0
L2   = TS                      /* OBJECT TYPE */
L2I
.
L8   = "DBNAME IN('TESTDB1','TESTDB2')"
L8I  = 0
.
IF SSID = "PRODUCTION" THEN
     P27  = 10   /* Percentage of Total Changes      - DEFAULT 20 */
  ELSE
     P27  = 20   /* Percentage of Total Changes      - DEFAULT 20 */
P27I =  0
```


15

In this REXX, we are telling DSNACCOR that we want to identify tablespace objects that are in need of a runstat on either the TESTDB1 or TESTDB2 databases.

In addition, we want to identify any object on either of those two databases that are in a restricted status. The program takes identified tablespace objects that are in a restrictive state and writes those objects out to another dataset. This dataset gets emailed to the group, identifying restricted status objects. These objects are omitted from runstat processing.

In production, a tablespace object is runstatted if the percentage of total changes [(inserts+updates+deletes) / totalrows] is 10% or greater since the last time runstats were performed on that tablespace object. A runstat occurs on a tablespace object if this value is 20% or higher since the last time runstats were performed in the test regions.

Except for percentage of total changes, all other default formulas are used by DSNACCOR when making runstat utility recommendations.

Access DB2 Catalog

- Requirement for latest full image copy to match our DBA naming standard.
- After DSNACCOR makes initial recommendations, access DB2 Catalog.
- Run SQL and write information to reports and write out additional full image copy JCL.

```
sql =  "SELECT A.DBNAME, A.TSNAME, A.DSNUM, C.PARTITIONS ",
       "FROM SYSIBM.SYSCOPY A, SYSIBM.SYSTABLESPACE C",
       "WHERE A.ICTYPE = 'F'                          ",
       "AND A.DSNAME NOT LIKE 'DB2IC%' AND            ",
       ""||L8||"                                      ",
       "AND A.TSNAME = C.NAME                         ",
       "AND A.DBNAME = C.DBNAME                       ",
       "AND A.TIMESTAMP =                             ",
       "(SELECT MAX(B.TIMESTAMP) FROM SYSIBM.SYSCOPY B ",
       "WHERE A.TSNAME=B.TSNAME AND A.DBNAME=B.DBNAME  ",
       "AND A.DSNUM=B.DSNUM)                          "
```


This query is done because of our automated disaster recovery process.

The disaster recovery process only looks for full image copies with a certain high-level naming convention.

Due to this requirement, every night this query looks for the existence of a full image copy that is the most recent full image copy that does not conform with the DBA group high-level naming convention for full image copies, and identifies that tablespace object as a full image copy candidate (with our DBA standard high-level naming convention).

This query is run after DSNACCOR makes its initial recommendations on which tablespace objects need a full image copy, and is one of the last steps of the full image copy REXX program.

Automatically Build New GDG Bases

- Use REXX to call IGGCSIRX (catalog search interface), to check for the existence of GDG base.
- If GDG base exists, no action is taken.
- If GDG base does not exist, record gets written out to a file that is later executed via IDCAMS to build new the GDG base.
- No manual intervention required for new tablespaces.

```
CREATE_GDG:
$DSN = WORD(x.5,3)
$DSN = SUBSTR($DSN,5)
$DSN = LEFT($DSN,LENGTH($DSN)-5)
     if IGGCSIRX($DSN) = 16 then do
      PCNTR = PCNTR + 1
       PREFIX.PCNTR = " DEFINE GDG (NAME("||$DSN||") LIMIT(15)
            SCRATCH)"
     END
```


No manual intervention is required for new tablespace objects.

There is no dataset to “maintain” that includes a running “list” of image copy GDG bases.

How many times have we implemented a new tablespace object, and forgot to build the GDG base to store the full image copy file?

DSNACCOR Reporting

- &hilevel.&subsystem.copy.rtsjcl.
  - JCL of Copied Objects.
  - Copied every night to production JCLLIB.
- &hilevel.&subsystem.copy.rtslist.
  - List of Copied Objects.
- &hilevel.&subsystem.restrict.rtslist.
  - Objects in restricted status/not copied (unless object in "RW,COPY").
- &hilevel.&subsystem.runst.rtslist.
  - List of Runstatted Objects.
- &hilevel.&subsystem.restruns.rtslist.
  - Objects in restricted status/not runstatted.

DB2 SUBSYSTEM: DB2P - 01/16/07 21:02:52 - RESTRICTED STATUS - NO COPY! (UNLESS TABLESPACE IN 'RW,COPY')

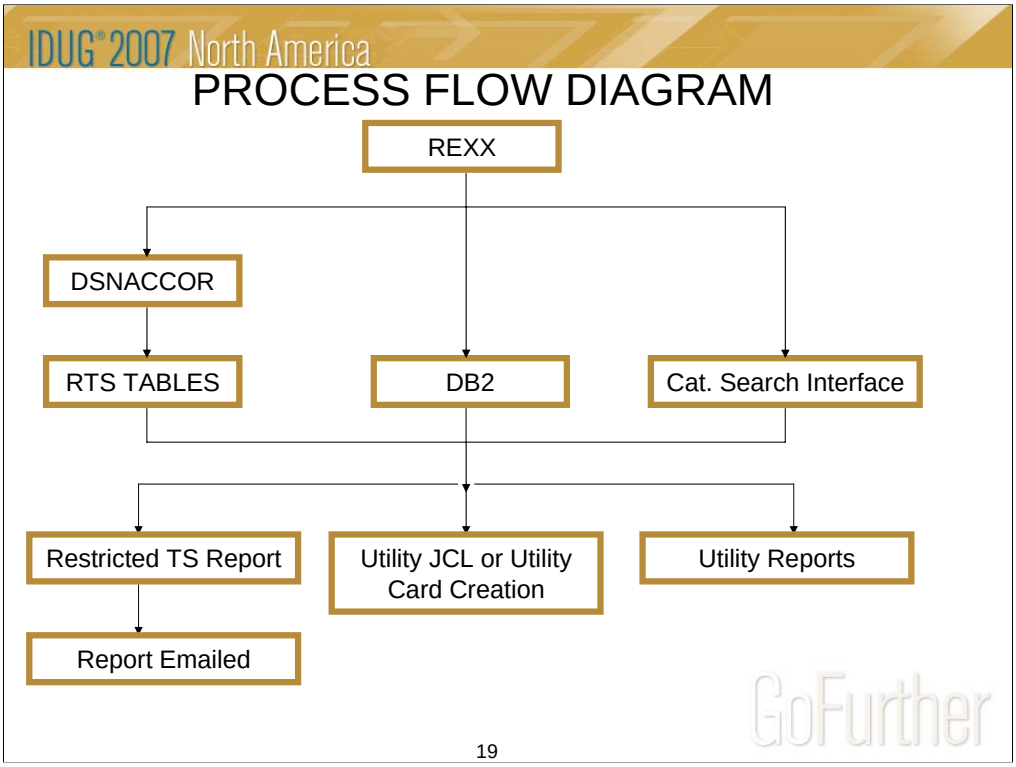
DBNAME	NAME	PART	TYPE	STATUS	IC	COPYLASTTIME	UPDPGSPCT	CPYCHGPCT
TBDB01	TBPOTS	0	TS	DB=RW ;TS=RW,COPY	FUL	2007-01-15-21.23.49.903434	59	2
TBDB01	TBINTS	0	TS	DB=RW ;TS=RW,COPY	FUL	2007-01-15-21.23.44.650618	0	0

DB2 SUBSYSTEM: DB2Q - 01/17/07 04:30:23 - DSNACCOR IMAGE COPY RECOMMENDATIONS

DBNAME	NAME	PART	TYPE	STATUS	IC	COPYLASTTIME	UPDPGSPCT	CPYCHGPCT
EPDB01	EPPLTS	1	TS		FUL	2007-01-12-04.31.25.879842	18	22
LRDB01	LRFATS	0	TS		FUL	2007-01-04-04.31.29.017595	8	3

DB2 SUBSYSTEM: DB2Q - 01/16/07 22:00:18 - DSNACCOR RUNSTAT RECOMMENDATIONS

DBNAME	NAME	PART	TYPE	STATSLASTTIME	SRTINSDELUPDPCT	STATSMASSDELETE
EPDB01	EPPLTS	1	TS	2007-01-02-22.15.46.128376	38	0
EPDB01	EPPLTS	2	TS	2007-01-02-22.15.45.822708	25	0
ABDB01	ABIMTS	0	TS	2006-12-31-15.30.33.644209	18	1
ABDB01	ABUMTS	0	TS	2007-01-07-15.30.28.551754	255	1
ACDB01	ACCHTS	0	TS	2007-01-02-22.04.34.655550	100	1
ACDB01	ACCTTS	0	TS	2007-01-09-22.00.35.283803	672300	0
ACDB01	ACSUTS	0	TS	2007-01-07-15.30.42.471797	212882	0
BIDB01	BIISTS	0	TS		0	0



This slide illustrates the flow of our automatic utility generation by using real-time statistics, and DSNACCOR.

The catalog search interface is only called during full image copy utility generation.

The “DB2” box refers to:

1) Full Image Copy – Queries the system catalog for non-standard DBA high-level image copies.
2) Runstats – Queries the system catalog to gather a list of all associated indexes for a tablespace object, so all NPI’s and non-clustering indexes can also be runstatted.
3) Reorg – Queries the system catalog to gather space information for alter of primary and secondary quantities. Also queries the system catalog to ensure NPI’s of partitioned tablespaces are reorged.

Utility Generation Differences

- The L1 Variable in all 3 REXX programs.
- Depending on utility, default formulas overridden to meet application needs.
- Image copy REXX queries sysibm.syscopy.
- All indexes runstatted for a tablespace flagged for runstat.
- Copy and Runstat – only tablespaces; Reorg all objects.
- Reorg queries catalog for space information. Tablespaces and indexes primary and secondary quantities altered before reorg.
- Reorg REXX – alter and reorg NPI's for partitioned tablespaces.

The L1 variable in the 3 REXX programs differs:

RTSCOPY - *L1 = 'COPY' 'RESTRICT' /* QUERY TYPE */*
RTSRUNST - *L1 = 'RUNSTATS' 'RESTRICT' /* QUERY TYPE */*
RTSREORG - *L1 = 'REORG' 'RESTRICT' /* QUERY TYPE */*

The L1 variable in the REXX controls what utility type you want DSNACCOR to make (copy,runstats, or reorg).

The L8 (Criteria) variable overridden to determine which objects are eligible for DSNACCOR recommendations. Without overriding DSNACCOR criteria, all objects in the DB2 subsystem would be eligible to utility operations.

Criteria parameter is heavily modified to either include, or exclude, certain databases and tablespaces.

RTSCOPY REXX queries syscopy for "non" standard DBA full image copy. If the latest full image copy has an application high-level, DBA takes another full image copy (Disaster Recovery purposes) with the DBA standard high-level qualifier.

A tablespace in list is built during the RTSRUNST REXX execution. This in list is used to query sysindexes and sysindexpart. We runstat all indexes for a tablespace, not just the clustering or partitioning index.

RTSCOPY and RTSRUNST only look for the "TS" object type. RTSREORG looks for all objects. Indexes in RTSREORG usually just look for high number of extents.

The alters that occur before the actual reorg utility runs allows us to realize the benefits of compression, and to reclaim unused space. An upper and lower limit with respect to primary and secondary quantities are established for the alters.

Benefits of Real-time statistics

- Accurate reflection of objects in need of maintenance.
- Reduce waste.
- Improved Application Performance.
- DSNACCOR – easy to automate utilities.
- DSNACCOR – built in formulas.
- Ease of implementation.
- Tablespacestats and Indexspacestats (Ease of interpretation).

Allows DBA to more accurately reflect what DB2 objects need maintenance Are your daily or weekly utilities really accurate?

Reduce waste by only running DB2 utilities on objects that need the maintenance. Why run expensive utilities on DB2 objects that do not need the maintenance?

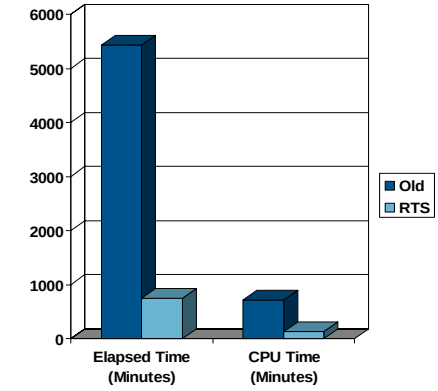
Improved Application Performance. No “lag” time between application runstats and reorgs. Reorgs run based on real-time needs. Previously, runstat would run, and then reorgs would run days later. The lag time between runstats and reorgs is eliminated.

Automation of DB2 utilities becomes easy with the use of the IBM supplied stored procedure DSNACCOR. The REXX programs take work and customization based on your specific needs, but once you have one program complete (started with image copy), it is easy to translate that over to the other utilities.

DBA can access DSNACCOR, and let the stored procedure recommend utility operations, rather than having the DBA come up with their own formulas for DB2 utility operations. DSNACCOR already has built in formulas that can be over-written, if necessary, to tailor the recommendations to application specifications.

If you are using DB2 Version 7 (or later), real-time statistics are already running. Just create the Tablespacestats and Indexspacestats tables, initialize the tables, and you’re ready to use the data, which is easy to interpret.

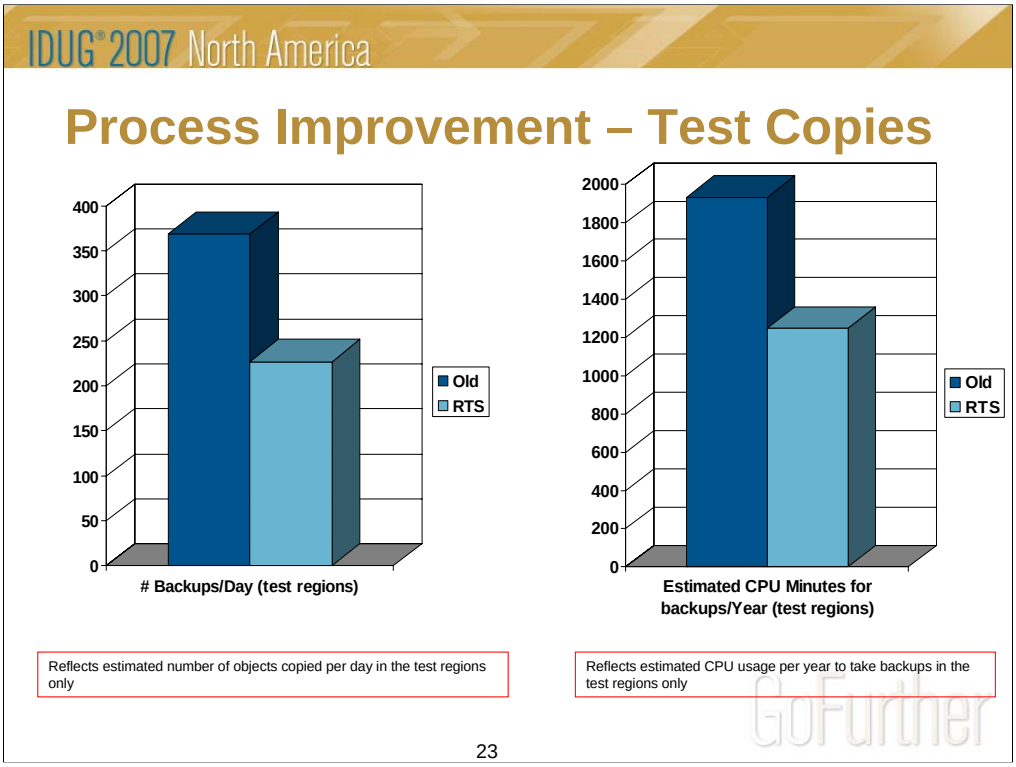
Significant Process Improvement During a Calendar Year



Reflects estimated elapsed and CPU time spent to generate backup jobs

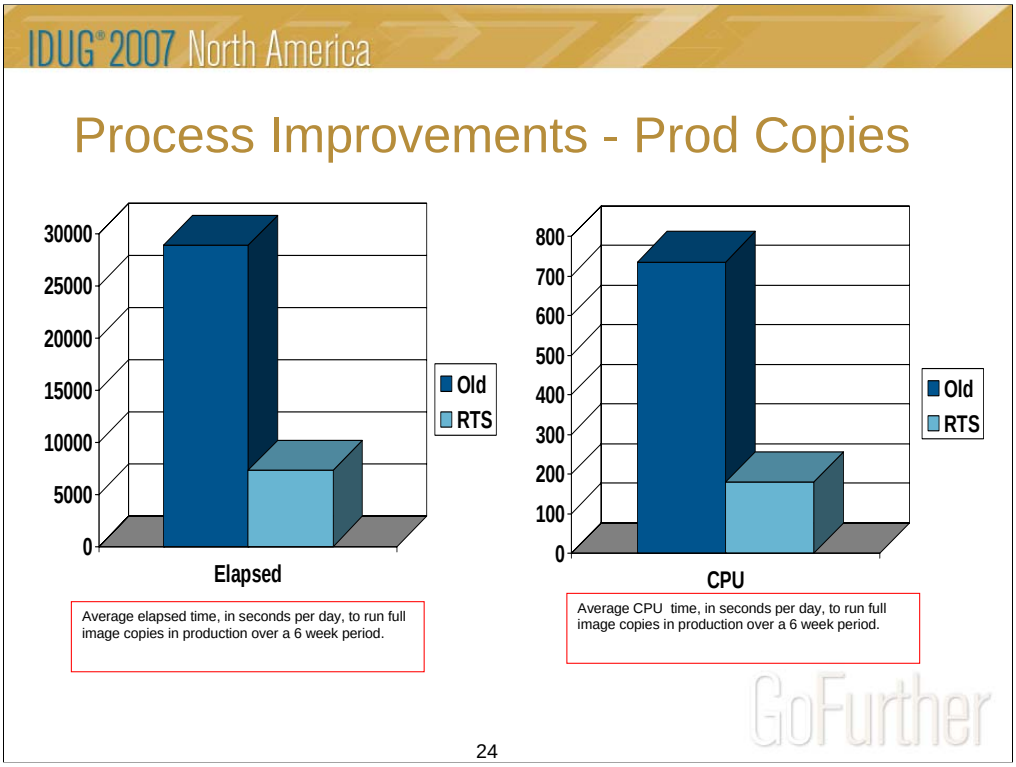
This slide illustrates a pronounced performance improvement (approximately 85%) with respect to how we previously automatically generated full image copies for application tablespace objects.

By using real-time statistics to generate the full image copies, we are realizing a significant reduction in both elapsed time and CPU time spent automatically generating our full image copy jobs.



This reduction is mainly the result of only taking a full image copy of tablespace objects in the test regions if that object changes by 20% or more (either updated pages or total changes) since the last time a full image copy on that object was taken.

Previously, for test environments, a full image copy was taken if a tablespace object changed at all (even .01%). Due to the fact that database recoveries in our test regions are rare, the 20% threshold set reduces the number of test backups taken on a daily basis, but still ensures application recoverability if the need arises.



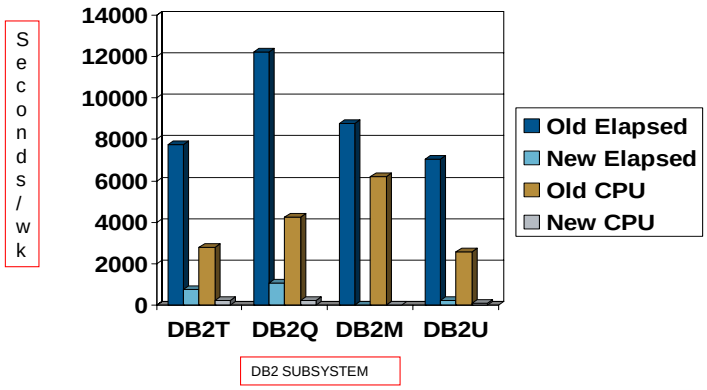
The amount of CPU usage, and elapsed time, used to take full image copies in production have both been drastically reduced due to the implementation of generating full image copies based on real-time statistics data.

Elapsed time has decreased by approximately 77%, which CPU usage has been reduced by approximately 74%.

The old process would take a full image copy of any tablespace object that changed, even if the change was ever so slight (even at .01%).

In our production environment, full image copies are only taken if the changed pages exceed 10%, if the total changes exceed 10%, or if the last image copy taken on the object is over 12 days old.

Runstat Improvements - Test



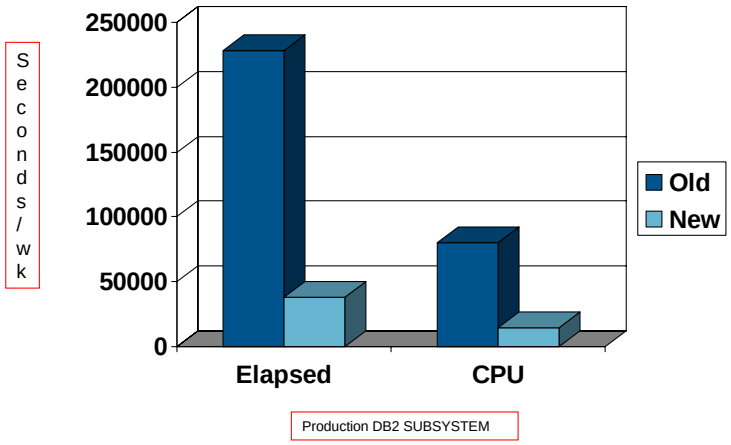
This chart represents the amount of seconds taken each week for runstats in the test regions under the old process and new (RTS) process. Reflected are both elapsed time and CPU consumption time. About 95% improvement in elapsed time and CPU time have been realized in test regions by converting our runstats to be based on real-time statistics.

The data used to compile the information was a three month period before the implementation of runstat generation based on real-time statistics, and then a three month period of time after the implementation of runstat generation based on real-time statistics.

The 95% improvement realized for test environment runstats relates to two key factors.

1) Previously, runstats were processed on any object that was copied during the week (and remember, we were copying just about everything!).
2) Our test environments now only get runstatted if the percentage of total changes exceeds 20%.

Runstat Improvements - Production



26

This chart represents amount of seconds taken each week for runstats in the production region under the old process and new (RTS) process. Reflected are both Elapsed time and CPU consumption time. About 83% improvement in elapsed time and CPU time have been realized in production. New Elapsed and CPU time savings compiled over an 16 week time period (11/12/06 – 02/27/07).

The approximate 83% improvement realized for the production environment runstats relates to two key points.

1) Previously, runstats were processed on any object that was copied during the week (and remember, we were copying just about everything!).
2) Our production environment now only get runstatted if the percentage of total changes exceeds 10%.

Processing Savings

- If you runstat virtually every DB2 object that you have in your subsystems, consider:
  - What is your cost per CPU minute for batch?
  - How long does your current runstat processing take?
  - Runstats = Expensive utility
  - How much could you really save if you experienced an 85% reduction in runstat processing time? =➔ Probably A LOT!

GoFurther

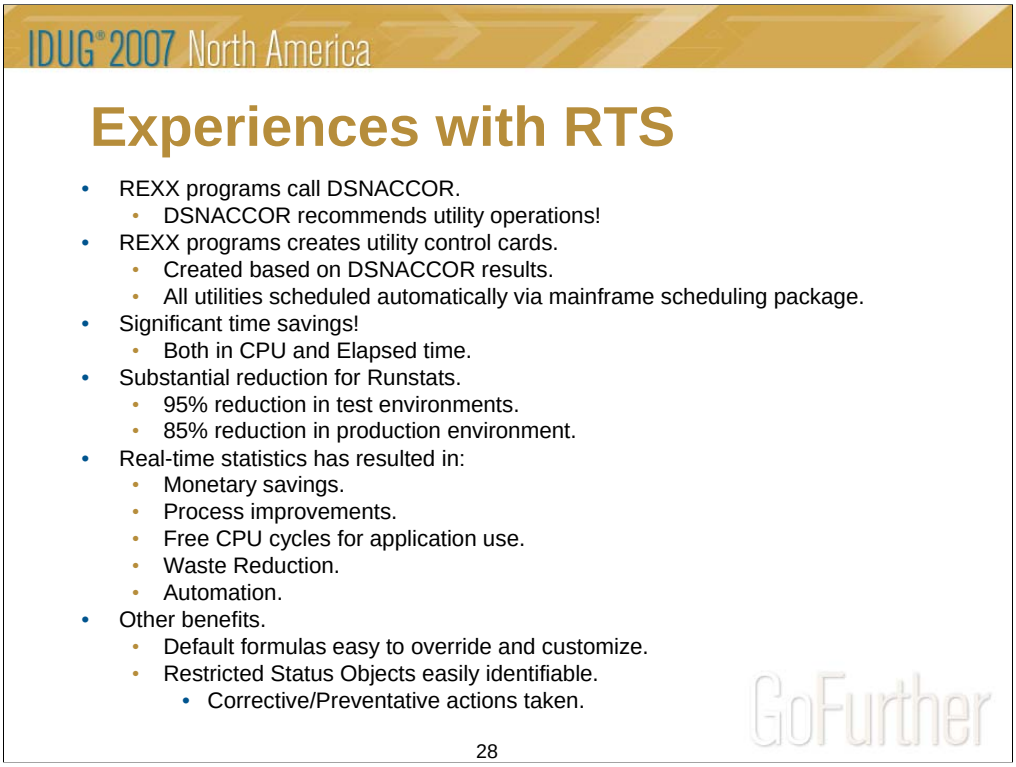
27

When I took all of the factors listed above into account, and realized how much extraneous processing was occurring with respects to the runstat utility, it was obvious we would realize substantial improvements by utilizing real-time statistics to generate our runstat utility cards.

Runstats are an expensive utility to run, and are CPU intensive, and should only be processed on objects that are dynamically changing.

We have realized significant reductions in time to process runstats, but not at the expense of application performance. Objects in need of updated statistics are still runstatted as necessary.

If you do realize these improvements by using real-time statistics to generate your runstats utility jobs, be sure to document the savings as accurately as possible.



REXX programs written that call IBM stored procedure DSNACCOR, which recommends which DB2 objects are in need of maintenance.

REXX programs create backup or runstat utility cards (utilizing the information provided by DSNACCOR) that are then processed automatically by the mainframe CA7 scheduling package.

Significant time savings over our old backup and runstat job generation process (over 80% improvement!).

Substantial reduction (about 90%) in elapsed time and CPU consumption with respects to runstat and full image copy utility processing.

Sizeable CPU reductions results in monetary savings, process improvements, guaranteed application recoverability, and more available CPU cycles for application processes.

Backup or Runstat only what objects are in need. Can increase percentage parameters in test environments, so utilities are only processed when needed (less CPU consumption, tape usage, etc.).

Objects in restricted status are written to a separate report. The report is emailed to our group so corrective actions can be taken on the objects.

RTS and the Future

- Reorg REXX currently in test.
- Lag time between runstats and Reorgs – currently.
- RTS reorgs will be based on real time need.
- RTS Reorg Rexx.
  - Same basic syntax as image copy and runstat.
  - Some Customization.
  - Reorg indexes in high extents.

Currently using for backup job generation and runstat utility card generation in all DB2 regions (test and production). Reorgs are being created based on real-time statistics in two test regions.

Currently, we reorg objects based on runstat values that were compiled a few days earlier.

Objects reorged will be based on real-time need, rather than the 3-5 day lag time after runstats are processed, which should lead to better application performance.

Will use DSNACCOR stored procedure to generate reorg utilities that will reorg objects based on real-time need.

Indexes (or tablespaces) in a high number of extents will have primary and secondary quantities altered, and then reorged to reduce extents automatically.

Session: A08
Real-Time Statistics and
Automating DB2 Utilities

Steve Wallace
Liberty Mutual Insurance Company
Steven.Wallace@LibertyMutual.com

GoFurther