DB2 10 and 11 for z/OS – Implementing and Using Autonomic Statistics

John Iczkovits
IBM

Session Code: A14
May 16, 2014 9:15 am | Platform: DB2 for z/OS

- Title: DB2 10 and 11 for z/OS - Implementing and Using Autonomic Statistics
- Abstract: DB2 10 and 11 have a neat feature, automatically executing RUNSTATS on required objects. By the way, you get to choose when and how to execute RUNSTATS. The actual implementation however is not straight forward. Come to this presentation and learn how to successfully implement and review the output for this new feature.
- Special thanks to IBM's Bjoern Broll
- **Objective 1:** Learn what autonomic statistics is
  Objective 2: Learn how to setup autonomic statistics
  Objective 3: Learn how the entire process executes
  Objective 4: Learn how to determine success or failure of the executions
  Objective 5: Learn how to view all of the results from the executions

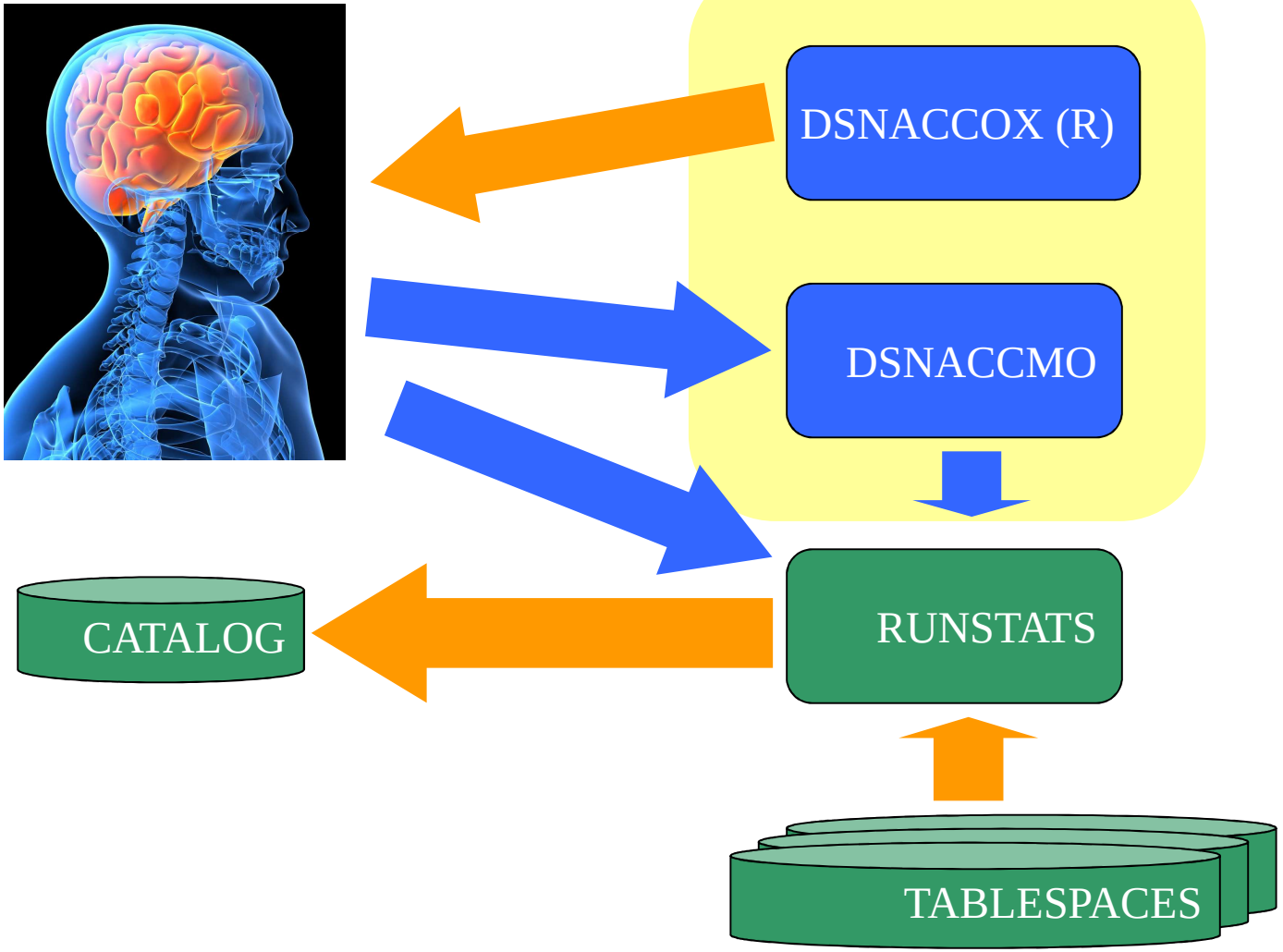
Agenda

- Introduction
- New RUNSTATS features
- AutoStats
  - Design
  - Interfaces
  - Installation
  - Example Scenario
- Considerations
- Test and results

What autonomic statistics offers

- Collecting stats is a difficult and time consuming manual process
  - Need to look at the queries to figure out what stats are needed
  - Need to repeatedly look at the RTS tables to figure out when to recollect
- Inadequate stats collection leads to poor query performance or inconsistent query performance (sometimes the query runs well and sometimes it runs poorly)
- Solution is to automate the process
  - More efficient
  - More accurate
  - More stable

DB2 V8-9 (state-of-the-art)



Autonomic Statistics Management

- Reduces much of the work currently done by DBA's regarding database statistics management by adding functions to DB2 which will perform those tasks without the need for DBA involvement. These tasks include:
  - Identifying what stats to collect (minimal)
  - Storage of RUNSTATS profiles
  - Identifying when stats need to be collected or re-collected
  - Invoking the stats collection service at the correct time with the proper stats collection criteria

New RUNSTATS features

New RUNSTATS features

- SET PROFILE
  - Allows RUNSTATS to generate a statistics profile from the options specified in the current RUNSTATS invocation, store this profile in the system catalog table SYSIBM.SYSTABLES_PROFILES
- USE PROFILE
  - Allows RUNSTATS to use a previously stored statistics profile to gather statistics for a table
  - The statistics profile is created using the SET PROFILE option and is updated using the UPDATE PROFILE option
  - The column, column group, and index specifications stored in the statistics profile are used. These may not be specified as part of the control statement.
- DELETE PROFILE
  - This option will cause RUNSTATS to delete the stored statistics profile from table SYSIBM.SYSTABLES_PROFILES
  - Column, column group, and index specifications are not allowed as part of the control statement when DELETE PROFILE is used
- UPDATE PROFILE
  - Allows RUNSTATS to update an existing statistics profile in the system catalog tables with the options specified in the current RUNSTATS invocation
  - If the column or column group specification already exists in the profile, the new specification will replace the existing one

New catalog table SYSIBM.SYSTABLES_PROFILES

- RUNSTATS profiles are stored in the new SYSIBM.SYSTABLES_PROFILES table
- The profile is table based
- The associated RUNSTATS options are stored in the PROFILE_TEXT column
- These options have the same meaning as they do when specified directly in the RUNSTATS statement
- Any profile modifications done through SQL statements must follow the same restriction, or error messages will result when the profile is used.
- The PROFILE functions cannot be executed when there are syntax errors in the statistics profile. Syntax errors may be corrected using RUNSTATS UPDATE PROFILE or SQL statements, or by deleting the profile with RUNSTATS DELETE PROFILE or SQL DELETE.

Previewing a stats profile

- Stats profiles can be previewed using the PREVIEW option.
- When executing RUNSTATS with the PREVIEW option, DB2 only prints the stats profile for each table to SYSPRINT and normal utility execution does not take place.
- An alternative is executing SELECT * FROM SYSIBM.SYSTABLES_PROFILES

New RUNSTATS feature

- TABLESAMPLE SYSTEM
  - This option allows RUNSTATS to collect statistics on a sample of the data pages from the table
  - System sampling considers each page individually, including that page with probability P/100 (where P is the value of numeric-literal) and excluding it with probability 1-P/100
  - The size of the sample is controlled by the integer parameter in parentheses, representing an approximate percentage P of the table to be returned. Only a percentage of the data pages as specified through the numeric-literal parameter will be retrieved and used for the statistics collection.
  - Only valid on single-table table spaces.

Design of AutoStats

Key Stored Procedures used with autonomic statistics – created in installation job DSNTIJRT

- ADMIN_UTL_MONITOR (Administration Statistics Monitor)
  - Determines which stats should be collected / recollected
- ADMIN_UTL_EXECUTE (Administration Alert Execution)
  - Used for solving alerts written by ADMIN_UTL_MONITOR within timewindows defined in SYSIBM.SYSAUTOTIMEWINDOWS
- ADMIN_UTL_MODIFY
  - Maintains the history table SYSIBM.SYSAUTORUNS_HIST and the alert table SYSIBM.SYSAUTOALERTS
- Authorization - Only DB2 Administrators having system DBADM authority or higher are authorized to manage and run AUTOSTATS procedures.

UDFs required for autonomic statistics created in installation job DSNTIJRT

- ADMIN_TASK_LIST
- ADMIN_TASK_STATUS
- Authorized IDs must have call privileges

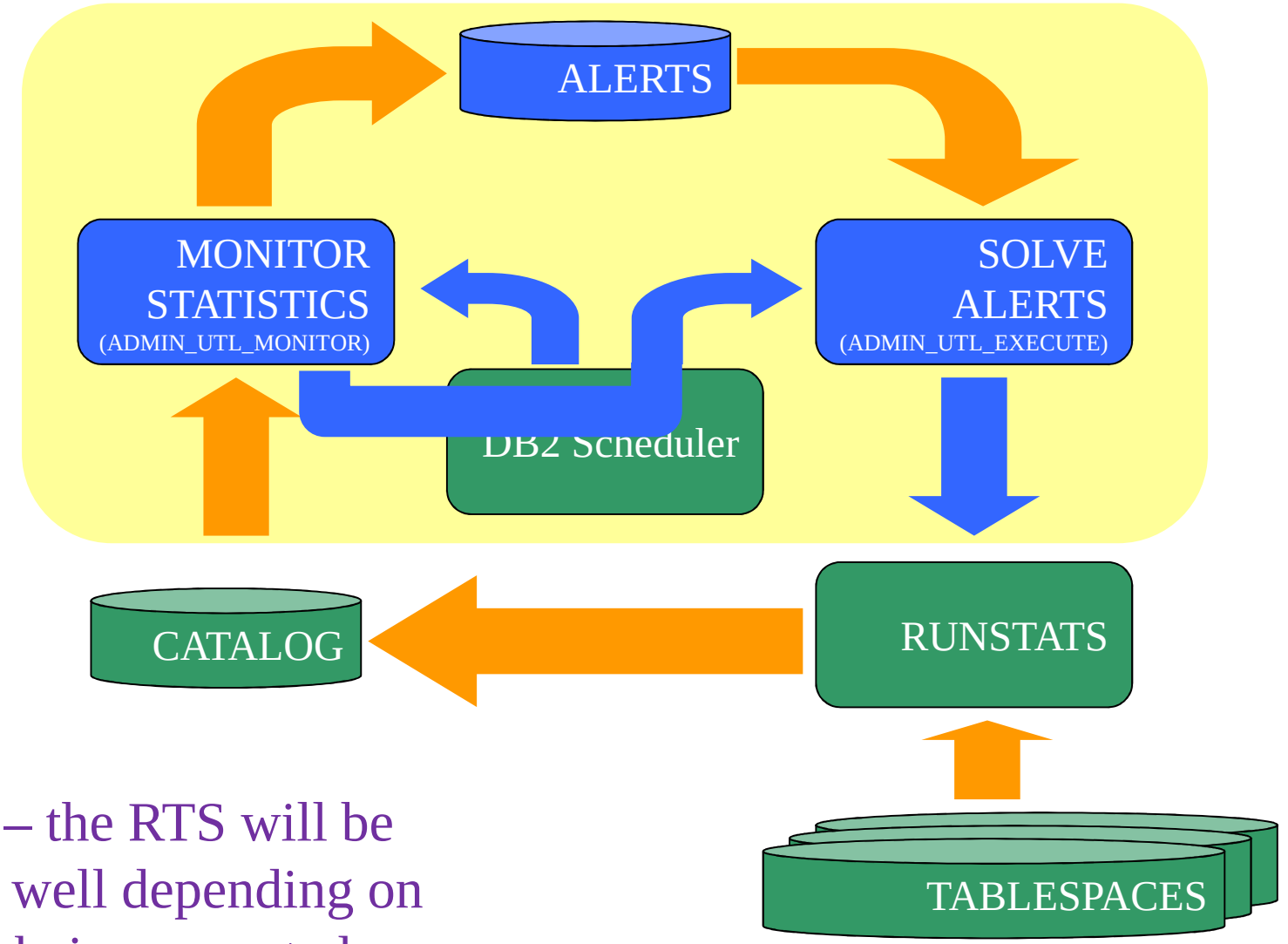
Catalog tables required for auto statistics for SELECT and UPDATE

- SYSIBM.SYSAUTOALERTS
- SYSIBM.SYSAUTORUNS_HIST
- SYSIBM.SYSAUTOTIMEWINDOWS
- SYSIBM.SYSTABLES_PROFILES
- Authorized IDs must have the privilege to select and modify these tables

Catalog tables required for read – authorized IDs must have read access

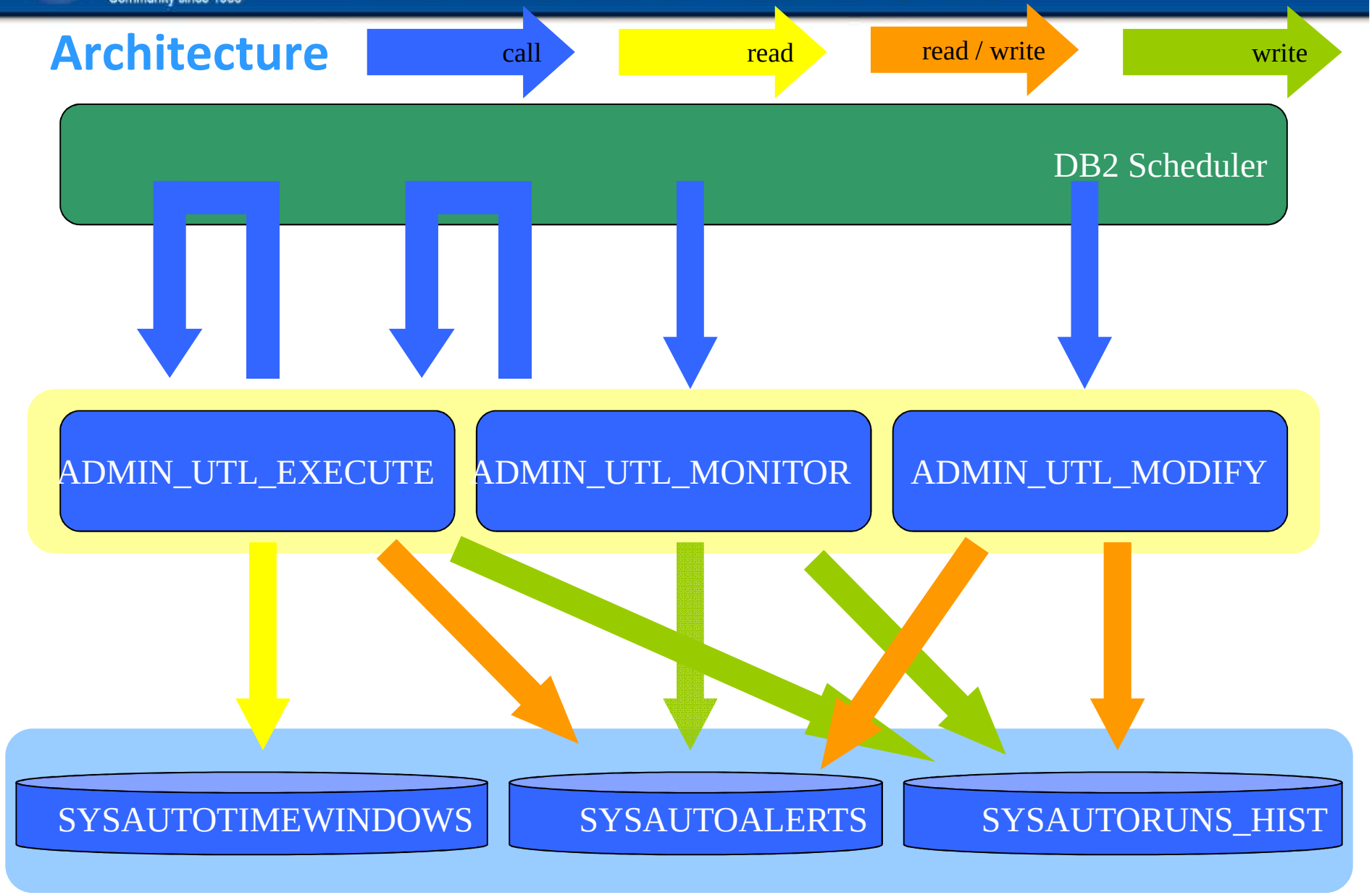
- SYSIBM.SYSTABLESPACESTATS (note, auto stats does not use SYSINDEXSPACESTATS)
- SYSIBM.SYSTABLESPACE
- SYSIBM.SYSDATABASE
- SYSIBM.SYSTABLES
- SYSIBM.SYSINDEXES
- SYSIBM.SYSKEYS
- SYSIBM.SYSCOLUMNS
- SYSIBM.SYSCOLDIST
- SYSIBM.SYSDUMMY1
- SYSIBM.UTILITY_OBJECTS

AutoStats Idea

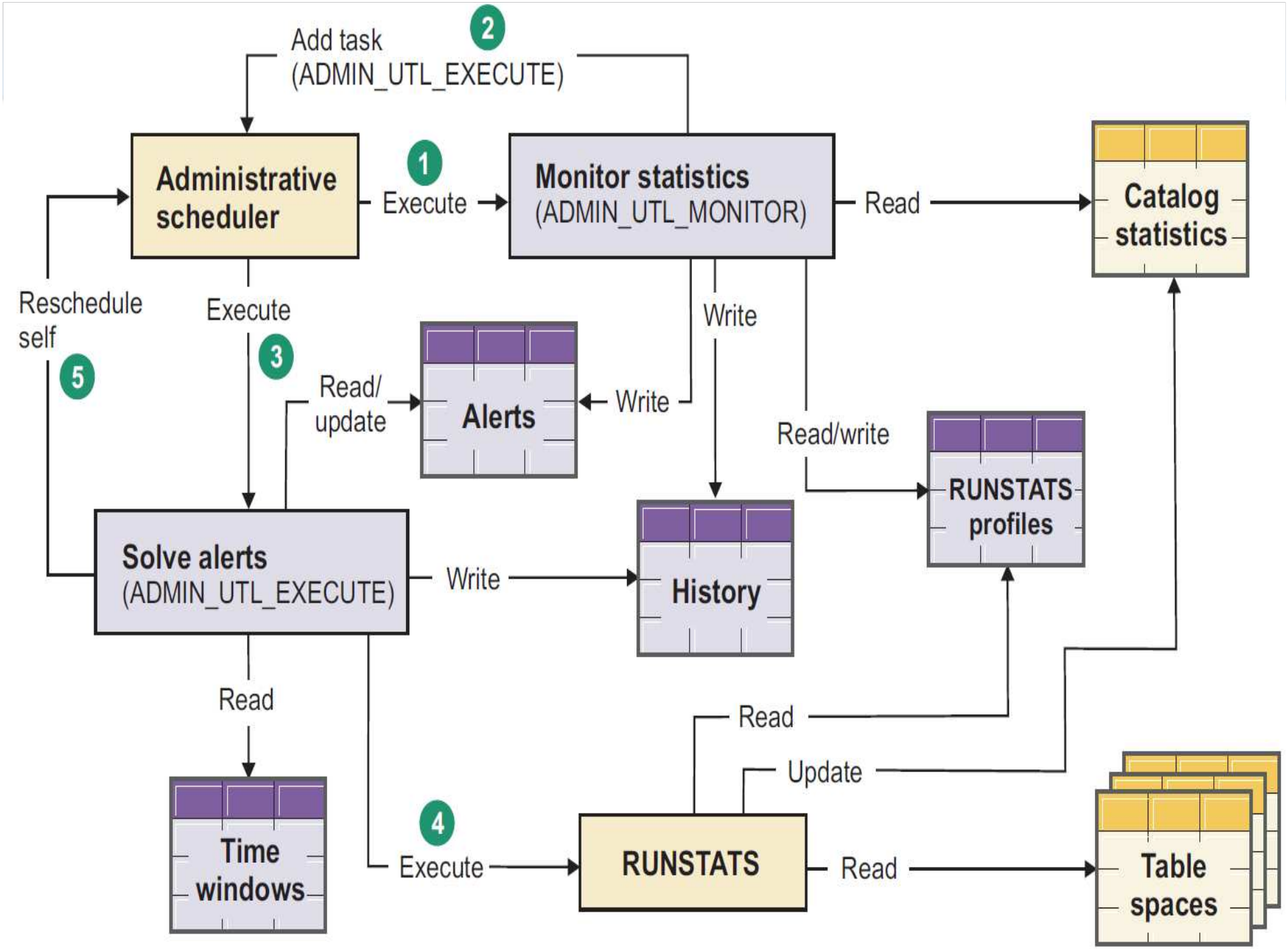


NOTE – the RTS will be read as well depending on what is being executed.

Architecture



Note - table SYSTABLES_PROFILES is not listed on this slide


Add task
(ADMIN_UTL_EXECUTE)
2
Administrative scheduler
1
Execute
Monitor statistics
(ADMIN_UTL_MONITOR)
Read
Catalog statistics
Reschedule self
5
Execute
3
Read/ update
Alerts
Write
Write
Read/write
RUNSTATS profiles
Solve alerts
(ADMIN_UTL_EXECUTE)
Write
History
Read
Read
Update
Time windows
4
Execute
RUNSTATS
Read
Table spaces

Architecture explained from previous page

1. ADMIN_UTL_MONITOR is called by ‘DB2 scheduler for administrative tasks' regularly specified by the customer's setup.
2. ADMIN_UTL_MONITOR checks the table spaces and tables for old, missing, and inconsistent statistics and writes an alert for every table space (-partition) / table which needs a RUNSTATS. Afterwards ADMIN_UTL_MONITOR schedules ADMIN_UTL_EXECUTE in DB2 for administrative tasks for direct execution.
3. ADMIN_UTL_EXECUTE is called by ‘DB2 scheduler for administrative tasks’. If at the current point-in-time a time window exists ADMIN_UTL_EXECUTE starts to run RUNSTATS for the table spaces listed in SYSAUTOALERTS and updates the status of the alerts, otherwise ADMIN_UTL_EXECUTE reschedules itself for the next maintenance window.

Technical details – Interfaces to AutoStats
New tables
New stored procedures

SYSIBM.SYSAUTOTIMEWINDOWS


```
SYSIBM.SYSAUTOTIMEWINDOWS (
WINDOW_ID       BIGINT     NOT NULL GENERATED ALWAYS AS IDENTITY,
DB2_SSID        CHAR(4),
MONTH_WEEK      CHAR(1)    NOT NULL,
MONTH           INTEGER,
DAY             INTEGER,
FROM_TIME       TIME,
TO_TIME         TIME,
ACTION          VARCHAR(256),
MAX_TASKS       INTEGER,
PRIMARY KEY(WINDOW_ID));
```

SYSIBM.SYSAUTOALERTS


```
SYSIBM.SYSAUTOALERTS (
ALERT_ID              BIGINT      NOT NULL GENERATED ALWAYS AS IDENTITY,
HISTORY_ENTRY_ID      BIGINT      NOT NULL,
ACTION                VARCHAR(32)  NOT NULL,
TARGET_QUALIFIER      VARCHAR(128) NOT NULL,
TARGET_OBJECT         VARCHAR(128) NOT NULL,
TARGET_PARTITION      SMALLINT     NOT NULL,
OPTIONS               VARCHAR(4000),
CREATEDTS             TIMESTAMP NOT NULL WITH DEFAULT,
DURATION              INTEGER,
STATUS                VARCHAR(32),
STARTTS               TIMESTAMP,
ENDTS                 TIMESTAMP,
OUTPUT                CLOB(2M),
RETURN_CODE           INTEGER,
ERROR_MESSAGE         VARCHAR(1331),
ROWID                 ROWID NOT NULL GENERATED ALWAYS);
```

SYSIBM.SYSAUTORUNS_HIST


```
SYSIBM.SYSAUTORUNS_HIST (
HISTORY_ENTRY_ID    BIGINT      NOT NULL GENERATED ALWAYS AS IDENTITY,
PROC_NAME           VARCHAR(128) NOT NULL,
STARTTS             TIMESTAMP,
ENDTS               TIMESTAMP,
OUTPUT              CLOB(2M),
ERROR_MESSAGE       VARCHAR(1331),
RETURN_CODE         INTEGER,
ROWID               ROWID       NOT NULL GENERATED ALWAYS);
```

SYSPROC.ADMIN_UTL_MONITOR

ADMIN_UTL_MONITOR

Tasks:

- Identifies out-of-date / missing / inconsistent statistics
- Writes alerts for out-of-date / missing / inconsistent tablespaces to SYSIBM.SYSAUTOALERTS
- Schedules ADMIN_UTL_EXECUTE in „DB2 scheduler for administrative tasks“ (immediate execution)

```
SYSPROC.ADMIN_UTL_MONITOR (
IN     options              VARCHAR(30000)
OUT history-entry-id        BIGINT
OUT return-code             INT
OUT message                 VARCHAR(1331)
);
```

SYSPROC.ADMIN_UTL_MONITOR

ADMIN_UTL_MONITOR

Options

- restrict-ts
  - WHERE CLAUSE ON SYSIBM.SYSTABLESPACESTATS
  - For example, exclude all objects with the database name of DSNDB01 and DSNDB06. Another example, only include objects with a database name of JOHNICZ and table spaces starting with ‘JOHN’.
- statistics-scope:
  - BASIC - Out-of-date statistics are checked, such as whether RUNSTATS has been run since the last LOAD or REORG operation or whether the number and percentage of changes in a table space are greater that a defined threshold. BASIC is the default value.
  - PROFILE - Out-of-date statistics and the completeness of statistics are checked, including whether all statistics in the table profile have been collected.
  - PROFILE-CONSISTENCY - Out-of-date statistics, the completeness of statistics, and the consistency of statistics are checked.
- stand-alone
  - Prohibits interaction with „DB2 scheduler for administrative tasks“
  - This option is needed when another scheduler than the DB2 scheduler is used
  - This option can also be used to review what objects are set for alerts, but not execute the RUNSTATS
  - Also review running in a Data Sharing environment

SYSPROC.ADMIN_UTL_MONITOR

ADMIN_UTL_MONITOR

Options

- Runstats
  - SAMPLING-THRESHOLD (TABLE CARDF)
  - SAMPLING-RATE (1-100)
- Thresholds
  - RTS
    - PCT-CHANGES
    - NUM-CHANGES
    - NUM-MASS-DELETES

SYSPROC.ADMIN_UTL_MONITOR

ADMIN_UTL_MONITOR

Options

- Thresholds
  - Inconsistencies
    - TABCARD-LESS-THAN-COLCARD
    - TABCARD-LESS-THAN-COLGROUPCARD
    - SUM-OF-FREQUENCY-GREAT-THAN-ONE
    - FREQUENCY-OUT-OF-RANGE
    - NUMBER-OF-FREQUENCY-RECORDS-GREATER-THAN-COLGROUP-CARD
    - MAXIMUM-FREQUENCY-LESS-THAN-RECIPROCAL-OF-COLGROUP-CARD
    - COLGROUP-CARD-GREATER-THAN-SUPERSET-COLGROUP-CARD
    - PRODUCT-OF-COLCARD-LESS-THAN-COLGROUP-CARD
    - QUANTILE-CARD-GREATER-THAN-COLCARD
    - QUANTILE-CARD-GREATER-THAN-COLGROUP-CARD
    - SUM-OF-HISTOGRAM-GREATER-THAN-COLCARD
    - SUM-OF-HISTOGRAM-GREATER-THAN-COLGROUP-CARD
    - SUM-OF-HISTOGRAM-FREQUENCY-GREATER-THAN-ONE
    - QUANTILE-FREQUENCY-OUT-OF-RANGE
    - TABCARD-LESS-THAN-INDEX-KEYCARD
    - TABCARD-NOT-EQUAL-UNIQUE-INDEX-FULLKEYCARD
    - INDEX-FULLKEYCARD-LESS-THAN-FIRSTKEYCARD
    - INDEX-FULLKEYCARD-LESS-THAN-ANY-KEY-CARD
    - SINGLE-COL-INDEX-FULLKEYCARD-NOT-EQUAL-FIRSTKEYCARD
    - DIFFERENT-COLGROUP-CARD-FROM-INDEXES
    - DIFFERENT-COLGROUP-CARD-FROM-COLDIST-AND-INDEX
    - DIFFERENT-SINGLE-COLGROUP-CARD-FROM-INDEXES
    - DIFFERENT-SINGLE-COLGROUP-CARD-FROM-COLDIST-AND-INDEX
    - DRF-LESS-THAN-NPAGES
    - DRF-GREATER-THAN-TABCARD

SYSPROC.ADMIN_UTL_EXECUTE

ADMIN_UTL_EXECUTE

Tasks

- Solve alerts listed in SYSIBM.SYSAUTOALERTS within timewindows defined in SYSIBM.SYSAUTOTIMEWINDOWS
  - Prioritization of alerts
  - Parallel execution of RUNSTATS
- Reschedule itself

```
SYSPROC.ADMIN_UTL_EXECUTE (
IN     options              VARCHAR(30000)
OUT history-entry-id        BIGINT
OUT return-code             INT
OUT message                 VARCHAR(1331)
);
```

SYSPROC.ADMIN_UTL_EXECUTE

ADMIN_UTL_EXECUTE

Options

- stand-alone
  - Prohibits interaction with „DB2 scheduler for administrative tasks“. This option is needed when another scheduler than the DB2 scheduler (ADMT STC) is used or when running on a specific Data Sharing member.

SYSPROC.ADMIN_UTL_MODIFY

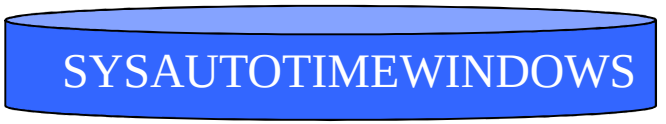
ADMIN_UTL_MODIFY

- Maintenance: Removes old entries in SYSIBM.SYSAUTORUNS_HIST & SYSIBM.SYSAUTOALERTS
- Options
  - history-days: Number of days after which solved alerts and log entries should be removed

```
SYSPROC.ADMIN_UTL_MODIFY (
IN     options              VARCHAR(30000)
OUT history-entry-id        BIGINT
OUT return-code             INT
OUT message                 VARCHAR(1331)
);
```

Installation process

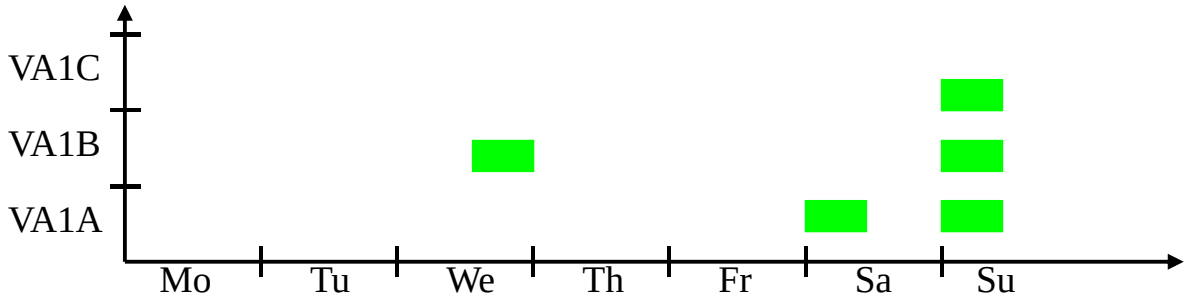
Installation



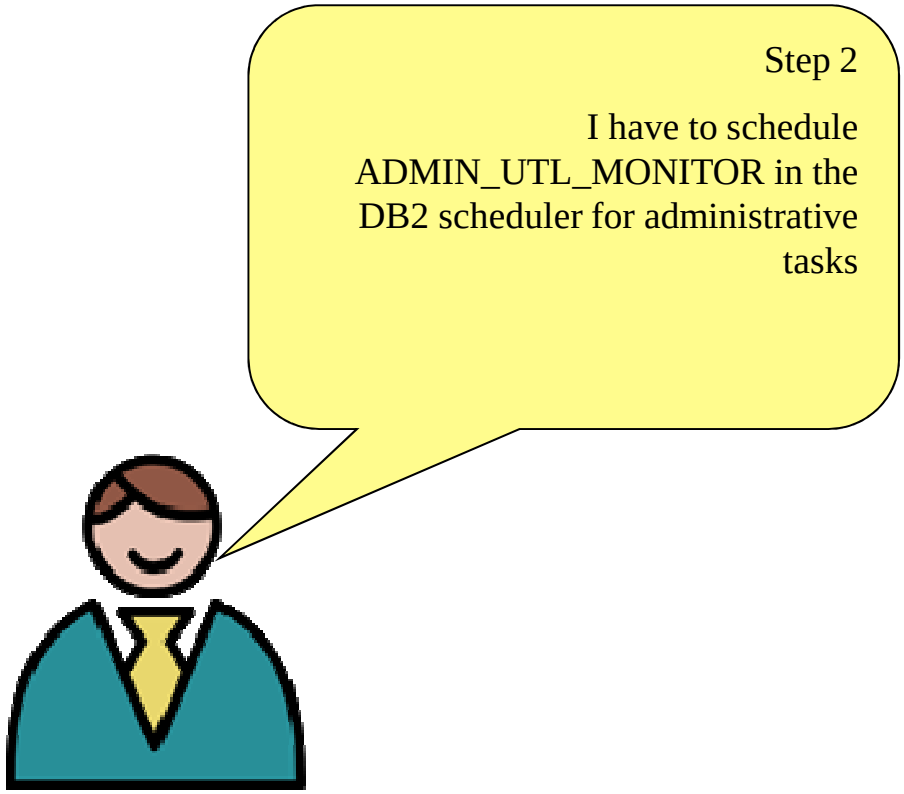
Add RUNSTATS maintenance window (Sunday 0-12)

Add RUNSTATS maintenance window (Saturday 0-12), SSID = VA1A

Add RUNSTATS maintenance window (Wednesday 12-24), SSID = VA1B



Installation

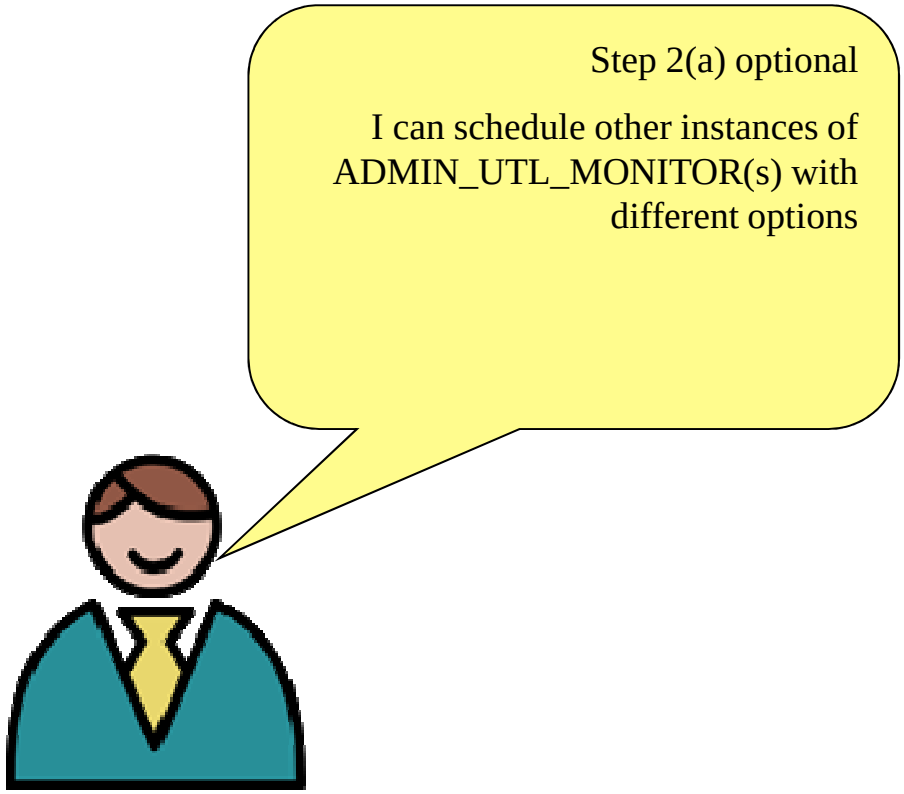


DB2 Scheduler

```
Call ADMIN_TASK_ADD(
   „SYSPROC“,
   „ADMIN_UTL_MONITOR“,  // Proc
   ....
   „0 22 * * *“,                    // Interval
   „DBNAME not in (...)“
   ....)
```

Now, everyday at 10 pm ADMIN_UTL_MONITOR is called (options = DBNAME not in (...)).

Installation

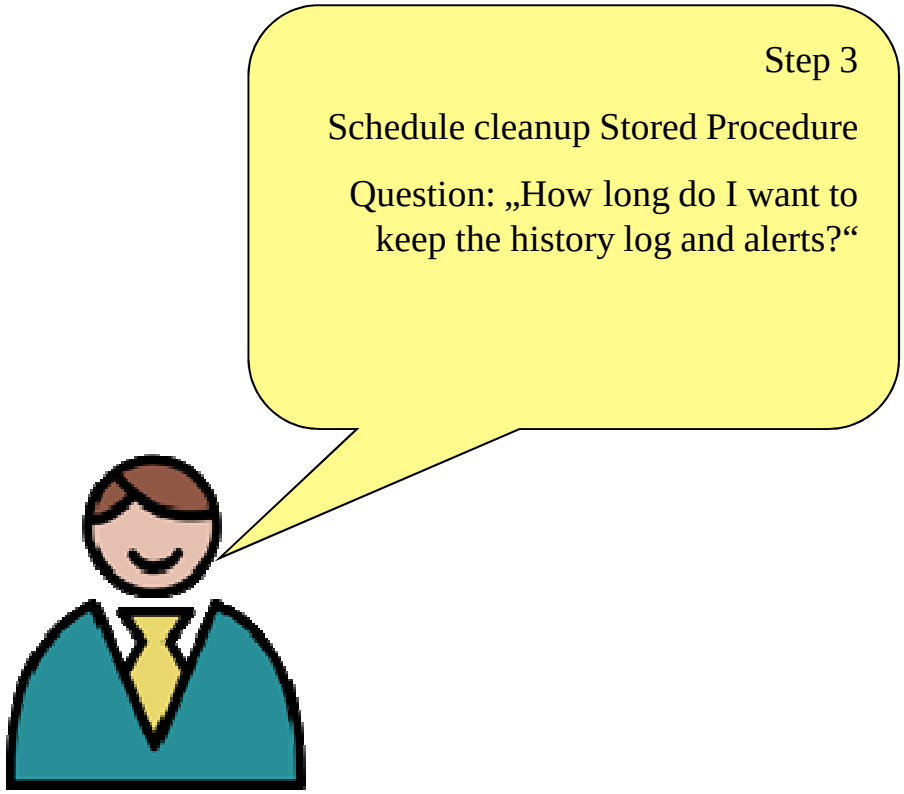


DB2 Scheduler

```
Call ADMIN_TASK_ADD(
   „SYSPROC“,
   „ADMIN_UTL_MONITOR“,  // Proc
   ....
   „0 22 1 * *“,                    // Interval
   „DBNAME in (...), SCOPE=DETIAL“
   ....)
```

Now another instance of ADMIN_UTL_MONITOR is called with options (DBNAME in (...) and SCOPE=DETAIL) on every 1st day of the month.

Installation



DB2 Scheduler

```
Call ADMIN_TASK_ADD(
   „SYSPROC“,
   „ADMIN_UTL_MODIFY“,    // Proc
   ....
   „0 22 1 * *“,                  // Interval
   „history=30d“
   ....)
```

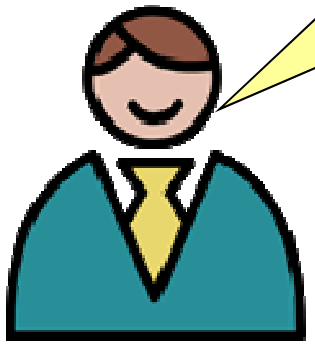
Now ADMIN_UTL_MODIFY is called on every 1st day of month.

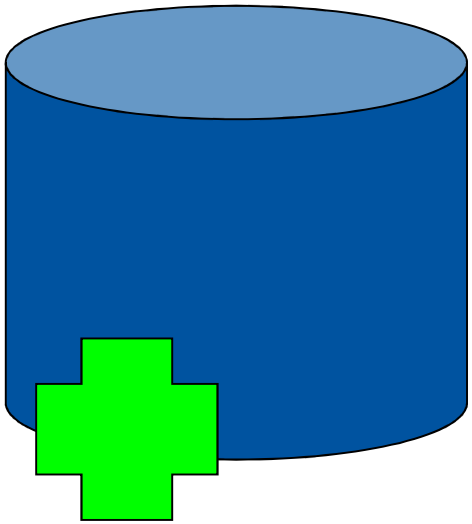
Installation

Step 4

Get some coffee and watch DB2 maintaining itself ;-)

History log can be found in SYSIBM.SYSAUTORUNS_HIST & SYSIBM.SYSAUTOALERTS





SYSAUTOALERTS

SYSAUTORUNS_HIST

Example of how to configure and monitor AutoStats via SQL

STEP 1: Define the maintenance windows in which RUNSTATS is allowed to run autonomously

```
Insert into SYSIBM.SYSAUTOTIMEWINDOWS(DB2_SSID,MONTH_WEEK,MONTH,DAY,FROM_TIME,TO_TIME,ACTION,MAX_TASKS)
       values(NULL,'W',NULL,1,'00:00','23:59:59','RUNSTATS',1);

Insert into SYSIBM.SYSAUTOTIMEWINDOWS(DB2_SSID,MONTH_WEEK,MONTH,DAY,FROM_TIME,TO_TIME,ACTION,MAX_TASKS)
       values(NULL,'W',NULL,2,'00:00','23:59:59','RUNSTATS',1);

Insert into SYSIBM.SYSAUTOTIMEWINDOWS(DB2_SSID,MONTH_WEEK,MONTH,DAY,FROM_TIME,TO_TIME,ACTION,MAX_TASKS)
       values(NULL,'W',NULL,3,'00:00','23:59:59','RUNSTATS',1);

Insert into SYSIBM.SYSAUTOTIMEWINDOWS(DB2_SSID,MONTH_WEEK,MONTH,DAY,FROM_TIME,TO_TIME,ACTION,MAX_TASKS)
       values(NULL,'W',NULL,4,'00:00','23:59:59','RUNSTATS',1);

Insert into SYSIBM.SYSAUTOTIMEWINDOWS(DB2_SSID,MONTH_WEEK,MONTH,DAY,FROM_TIME,TO_TIME,ACTION,MAX_TASKS)
       values(NULL,'W',NULL,5,'00:00','23:59:59','RUNSTATS',1);

Insert into SYSIBM.SYSAUTOTIMEWINDOWS(DB2_SSID,MONTH_WEEK,MONTH,DAY,FROM_TIME,TO_TIME,ACTION,MAX_TASKS)
       values(NULL,'W',NULL,6,'00:00','23:59:59','RUNSTATS',1);

Insert into SYSIBM.SYSAUTOTIMEWINDOWS(DB2_SSID,MONTH_WEEK,MONTH,DAY,FROM_TIME,TO_TIME,ACTION,MAX_TASKS)
       values(NULL,'W',NULL,7,'00:00','23:59:59','RUNSTATS',1);
```

This example allows RUNSTATS to be executed every day of the week at any time. The default for MAX_TASKS when set to NULL is run as many parallel tasks as possible.

For DB2_SSID, it refers to the DB2 member name on which the planned tasks are to be run. If this column contains NULL in a Data Sharing environment, the tasks during this time window can be run on any DB2 member. Refer to “FAQ – How do I use the scheduler in a Data Sharing environment?” later in this presentation.

For week, day 1 is Monday, not Sunday, same as with SYSIBM.SYSINDEXCLEANUP

select * from SYSIBM.SYSAUTOTIMEWINDOWS


```
select * from SYSIBM.SYSAUTOTIMEWINDOWS
```

WINDOW_ID	DB2_SSID	MONTH_WEEK	MONTH	DAY	FROM_TIME	TO_TIME	ACTION	MAX_TASKS
28	DBP1	W	NULL	1	00:00:00	23:59:59	RUNSTATS	1
29	DBP1	W	NULL	2	00:00:00	23:59:59	RUNSTATS	1
30	DBP1	W	NULL	3	00:00:00	23:59:59	RUNSTATS	1
31	DBP1	W	NULL	4	00:00:00	23:59:59	RUNSTATS	1
32	DBP1	W	NULL	5	00:00:00	23:59:59	RUNSTATS	1
33	DBP1	W	NULL	6	00:00:00	23:59:59	RUNSTATS	1
34	DBP1	W	NULL	7	00:00:00	23:59:59	RUNSTATS	1

In this example (slightly different than the previous page) we only want executions on DBP1

Note – If you are installing DB2 10 or a subsequent release, DSNTIJTC creates SYSAUTOTIMEWINDOWS. If you are migrating to DB2 10, DSNTIJEN (ENFM) creates it in step ENFM0001 – ENFM start . It is part of table space SYSTSATW with index DSNTWX01.

If DB2 is running V10 NFM or higher and SYSIBM.SYSAUTOTIMEWINDOWS is empty, DSNTRIN in job DSNTIJRT will initialize it with 7 rows of data as shown above. Execute RUNSTATS daily at anytime. The WINDOW_ID is 1-7. If the rows exist, there is no reason to manually add the rows – you are ready to run unless you want to modify the schedule.

FAQ, I want to schedule runs daily except for the first day on the month – do not run at all, and on the 15th of the month only run from 1 to 3 am.

- There is no way of telling the scheduler not to run on a specific day. An alternative:

```
Insert into SYSIBM.SYSAUTOTIMEWINDOWS(DB2_SSID,MONTH_WEEK,MONTH,DAY,FROM_TIME,TO_TIME,ACTION,MAX_TASKS)
      values(NULL,'M',NULL,2,'00:00','23:59:59','RUNSTATS',1);
Insert into SYSIBM.SYSAUTOTIMEWINDOWS(DB2_SSID,MONTH_WEEK,MONTH,DAY,FROM_TIME,TO_TIME,ACTION,MAX_TASKS)
      values(NULL,'M',NULL,3,'00:00','23:59:59','RUNSTATS',1);
```

etc.

```
Insert into SYSIBM.SYSAUTOTIMEWINDOWS(DB2_SSID,MONTH_WEEK,MONTH,DAY,FROM_TIME,TO_TIME,ACTION,MAX_TASKS)
      values(NULL,'M',NULL,15,'01:00','02:59:59','RUNSTATS',1);
Insert into SYSIBM.SYSAUTOTIMEWINDOWS(DB2_SSID,MONTH_WEEK,MONTH,DAY,FROM_TIME,TO_TIME,ACTION,MAX_TASKS)
      values(NULL,'M',NULL,16,'00:00','23:59:59','RUNSTATS',1);
```

etc.

```
Insert into SYSIBM.SYSAUTOTIMEWINDOWS(DB2_SSID,MONTH_WEEK,MONTH,DAY,FROM_TIME,TO_TIME,ACTION,MAX_TASKS)
       values(NULL,'M',NULL,31,'00:00','23:59:59','RUNSTATS',1);
```

- In this case as an alternative - first SELECT * FROM SYSIBM.SYSAUTOTIMEWINDOWS and verify only the seven rows inserted during the install/migration process exist to execute daily. Delete all of the rows (DELETE FROM SYSIBM.SYSAUTOTIMEWINDOWS). Insert one row for every day of the month (instead of week) except for day 1, and on day 15 change the times. Make sure you change MONTH_WEEK from 'W' weekly to 'M' monthly.

Step 2: Schedule ADMIN_UTL_MONITOR in the DB2 scheduler for administrative tasks

```
Call sysproc.admin_task_add(user-id,password,NULL,NULL,NULL,NULL,'*/30 * *
* *',NULL,NULL,NULL,NULL,'SYSPROC','ADMIN_UTL_MONITOR','select ''statistics-
scope=BASIC'',0,0,'''' from
sysibm.sysdummy1',NULL,NULL,NULL,'ADMIN_UTL_MONITOR_1','ADMIN_UTL_MONITOR
scheduled every 30 minutes, statistics-scope=BASIC',0,NULL);
```

```
Call sysproc.admin_task_add(user-id,password,NULL,NULL,NULL,NULL,'* 1 * *
*',NULL,NULL,NULL,NULL,'SYSPROC','ADMIN_UTL_MONITOR','select ''statistics-
scope=PROFILE-CONSISTENCY'',0,0,'''' from
sysibm.sysdummy1',NULL,NULL,NULL,'ADMIN_UTL_MONITOR_2','ADMIN_UTL_MONITOR
scheduled once a day at 1 am, statistics-scope=PROFILE-CONSISTENCY', 0,NULL);
```

In this case we have enabled two instances of ADMIN_UTL_MONITOR:

1. One instance which is scheduled every 30 minutes (on the hour) on a low detail level
2. One instance which is scheduled every day at 1 am on a high detail level

Date and time differences between SYSIBM.SYSAUTOTIMEWINDOWS and ADMIN_UTL_MONITOR

- SYSIBM.SYSAUTOTIMEWINDOWS is a DB2 table that works in the same date and time format you are used to
- The Stored Procedures used for autonomic statistics use the UNIX cron date and time format
- Both the CRON and usual time format can be run at different non–consecutive times of the day. For example, of the 15th of the month, execute RUNSTATS at 1 to 3 am and at 7 to 9 pm:

```
Insert into
SYSIBM.SYSAUTOTIMEWINDOWS(DB2_SSID,MONTH_WEEK,MONTH,DAY,FROM_TIME,TO_TIME,ACT
ION,MAX_TASKS)
      values(NULL,‘M',NULL,15,'01:00',‘02:59:59','RUNSTATS',1);
Insert into
SYSIBM.SYSAUTOTIMEWINDOWS(DB2_SSID,MONTH_WEEK,MONTH,DAY,FROM_TIME,TO_TIME,ACT
ION,MAX_TASKS)
      values(NULL,‘M',NULL,15,‘19:00',‘20:59:59','RUNSTATS',1);
```

Changing parameters for ADMIN_UTL_MONITOR

- You have everything in place, but you now want to change ADMIN_UTL_MONITOR2 to execute at 6 am instead of 1 am
  - Use Stored Procedure ADMIN_TASK_UPDATE
  - Unlike ADMIN_UTL_ADD, ADMIN_UTL_UPDATE does not have a parameter for the userid or password.
  - The admin scheduler checks the password at ADMIN_TASK_ADD time. If it is valid it stores the task in the task list (without the password). Internally a different approach is used to switch to the context of the user (it runs APF authorized and in key 0), so there is no need to know the password of the user. Scheduled jobs should not be affected when a user changes their password.

Removing ADMIN_UTL_MONITOR – you no longer want to run ADMIN_UTL_MONITOR_1 every 30 minutes

- CALL sysproc.admin_task_remove('ADMIN_UTL_MONITOR_1',0,'')
- Before executing admin_task_remove:

```
SELECT * FROM TABLE (DSNADM.ADMIN_TASK_LIST()) AS T;
 USERID  POINT_IN_TIME PROCEDURE_SCHEMA PROCEDURE_NAME    TASK_NAME
 ------- ------------ ---------------- ----------------- -------------------------
 JOHNICZ */30 * * * *  SYSPROC          ADMIN_UTL_MONITOR ADMIN_UTL_MONITOR_1
 JOHNICZ SYSPROC       NULL             ADMIN_UTL_EXECUTE DB2 AUTO PROCEDURE EXECUTE
```

- After executing admin_task_remove:

```
SELECT * FROM TABLE (DSNADM.ADMIN_TASK_LIST()) AS T;
 USERID  POINT_IN_TIME PROCEDURE_SCHEMA PROCEDURE_NAME    TASK_NAME
 ------- ------------ ---------------- ----------------- -------------------------
 JOHNICZ SYSPROC       NULL             ADMIN_UTL_EXECUTE DB2 AUTO PROCEDURE EXECUTE
```

- Even though the SELECT still returns a row for ADMIN_UTL_EXECUTE, it is no longer executed
- Both SELECT * FROM TABLE (DSNADM.ADMIN_TASK_LIST()) AS T and SELECT * FROM TABLE (DSNADM.ADMIN_TASK_STATUS()) AS T show that the last execution of ADMIN_UTL_EXECUTE was in the past.

FAQ – RUNSTATS is eligible to run 24 hours a day, 7 days a week, how do I make sure specific objects only execute during a specific period?

- I want to make sure that eligible RUNSTATS for all objects can execute all day, except database ABC can only execute from 1 – 3 am, but database XYZ can only execute from 9 – 11 pm.
  - Set up at least two monitor Stored Procedures using specific cron times and the restrict-ts parameter.
  - As an alternative, you can have one monitor Stored Procedure that executes at periodic intervals for all objects excluding databases ABC and XYZ. You have a second monitor Stored Procedure used for database ABC only with a specific time frame. You have a third monitor Stored Procedure used for database XYZ only with a specific time frame.

Step 3 (OPTIONAL): Schedule ADMIN_UTL_MODIFY in 'DB2 scheduler for administrative tasks' to delete old alerts and history logs regularly

```
Call sysproc.admin_task_add(user-
    id,password,NULL,NULL,NULL,NULL,'* * 1 *
    *',NULL,NULL,NULL,NULL,'SYSPROC','ADMIN_UTL_MODIFY','sel
    ect ''history-days=30'',0,0,'''' from
    sysibm.sysdummy1',NULL,NULL,NULL,'ADMIN_UTL_MODIFY','
    ADMIN_UTL_MODIFY executed on every first day of the
    month',0,NULL);
```

Step 4: We use Data Studio to monitor the autonomic stored procedures

select * from sysibm.sysautoalerts

(Alerts found by ADMIN_UTL_MONITOR + status of the alerts)

after ADMIN_UTL_MONITOR was executed.

ALERT_ID	HISTORY_...	ACTION	TARGET_Q...	TARGET_O...	TARGET_P...	OPTIONS	CREATEDTS	DURATION	STATUS	STARTTS	ENDTS	RETURN_C...	ERROR_M...	OUTPUT	ROWID
1534	863	RUNSTATS	DSN8DA1A	DSN8SA1D	0	TABLE(DSN...	21.01.10 0...	1							5479879f9...
1535	863	RUNSTATS	DSN8DA1A	DSN8SA1E	0	TABLE(DSN...	21.01.10 0...	2							3b0f879f9d...
1536	863	RUNSTATS	DSN8DA1A	DSN8SA1I	0	TABLE(DSN...	21.01.10 0...	1							92e8479f9...
1537	863	RUNSTATS	DSN8DA1A	DSN8SA1P	0	TABLE(DSN...	21.01.10 0...	1							ff9e479f9d...
1538	863	RUNSTATS	DSN8DA1A	DSN8SA1X	0	TABLE(DSN...	21.01.10 0...	1							0684279f9...
1539	863	RUNSTATS	DSN8DA1L	DSN8SA1B	0	TABLE(DSN...	21.01.10 0...	1							e886279f9...
1540	863	RUNSTATS	DSN8DA1X	DSN8SA1X	0	TABLE(DSN...	21.01.10 0...	1							a73ea79f9...
1541	863	RUNSTATS	DSN8DA1Y	DSN8SA1Y	0	TABLE(DSN...	21.01.10 0...	1							87d9e79f9...
1542	863	RUNSTATS	DSNADMDB	DSNADMTS	0	TABLE(ALL)...	21.01.10 0...	2							2d4be79f9...
1543	863	RUNSTATS	DSNMQDB	DSNMQTS	0	TABLE(SYSI...	21.01.10 0...	1							5bbc179f9...

after ADMIN_UTL_EXECUTE finished executing

ALERT_ID	HISTORY_...	ACTION	TARGET_Q...	TARGET_O...	TARGET_P...	OPTIONS	CREATEDTS	DURATION	STATUS	STARTTS	ENDTS	RETURN_C...	ERROR_M...	OUTPUT	ROWID
1534	863	RUNSTATS	DSN8DA1A	DSN8SA1D	0	TABLE(DSN...	21.01.10 0...	1	COMPLETED	21.01.10 0...	21.01.10 0...	4	DSNU010I...	2010-01-2...	5479879f9...
1535	863	RUNSTATS	DSN8DA1A	DSN8SA1E	0	TABLE(DSN...	21.01.10 0...	2	COMPLETED	21.01.10 0...	21.01.10 0...	0		2010-01-2...	3b0f879f9d...
1536	863	RUNSTATS	DSN8DA1A	DSN8SA1I	0	TABLE(DSN...	21.01.10 0...	1	COMPLETED	21.01.10 0...	21.01.10 0...	4	DSNU010I...	2010-01-2...	92e8479f9...
1537	863	RUNSTATS	DSN8DA1A	DSN8SA1P	0	TABLE(DSN...	21.01.10 0...	1	COMPLETED	21.01.10 0...	21.01.10 0...	0		2010-01-2...	ff9e479f9d...
1538	863	RUNSTATS	DSN8DA1A	DSN8SA1X	0	TABLE(DSN...	21.01.10 0...	1	COMPLETED	21.01.10 0...	21.01.10 0...	0		2010-01-2...	0684279f9...
1539	863	RUNSTATS	DSN8DA1L	DSN8SA1B	0	TABLE(DSN...	21.01.10 0...	1	COMPLETED	21.01.10 0...	21.01.10 0...	4	DSNU010I...	2010-01-2...	e886279f9...
1540	863	RUNSTATS	DSN8DA1X	DSN8SA1X	0	TABLE(DSN...	21.01.10 0...	1	COMPLETED	21.01.10 0...	21.01.10 0...	0		2010-01-2...	a73ea79f9...
1541	863	RUNSTATS	DSN8DA1Y	DSN8SA1Y	0	TABLE(DSN...	21.01.10 0...	1	COMPLETED	21.01.10 0...	21.01.10 0...	0		2010-01-2...	87d9e79f9...
1542	863	RUNSTATS	DSNADMDB	DSNADMTS	0	TABLE(ALL)...	21.01.10 0...	2	COMPLETED	21.01.10 0...	21.01.10 0...	0		2010-01-2...	2d4be79f9...
1543	863	RUNSTATS	DSNMQDB	DSNMQTS	0	TABLE(SYSI...	21.01.10 0...	1	COMPLETED	21.01.10 0...	21.01.10 0...	0		2010-01-2...	5bbc179f9...

Step 5: Output from second row (previous page)

OUTPUT (1,14)

```
2010-01-21-01:55:02.718371> 1DSNU000I    021 01:54:56.43 DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = X71X036003000002
2010-01-21-01:55:02.718422> DSNU1045I    021 01:54:56.47 DSNUGTIS - PROCESSING SYSIN AS UNICODE UTF-8
2010-01-21-01:55:02.718483> 0DSNU050I    021 01:54:56.47 DSNUGUTC -  RUNSTATS TABLESPACE DSN8DA1A.DSN8SA1E TABLE(DSN8A10.EMP) TABLESAMPLE SYSTEM
2010-01-21-01:55:02.718532>  AUTO USE PROFILE
2010-01-21-01:55:02.718573> DSNU1361I @ 021 01:54:56.48 DSNUGPRF - THE STATS PROFILE WITH STATSTIME = 2010-01-21-01.54.41.799597  FOR TABLE EMP HAS
2010-01-21-01:55:02.718723> BEEN USED
2010-01-21-01:55:02.718766> DSNU1368I    021 01:54:56.48 DSNUGPRB - PARSING STATS PROFILE FOR TABLE EMP
2010-01-21-01:55:02.718810> DSNU1369I    021 01:54:56.49 DSNUGPRB - PARSING STATS PROFILE FOR TABLE EMP COMPLETED
2010-01-21-01:55:02.718854> DSNU042I    021 01:54:59.31 DSNUGLSR - SORT PHASE STATISTICS -
2010-01-21-01:55:02.718897>                NUMBER OF RECORDS=588
2010-01-21-01:55:02.718939>                ELAPSED TIME=00:00:00
2010-01-21-01:55:02.718980> DSNU610I  @ 021 01:55:00.17 DSNUSUTP - SYSTABLEPART CATALOG UPDATE FOR DSN8DA1A.DSN8SA1E SUCCESSFUL
2010-01-21-01:55:02.719027> DSNU610I  @ 021 01:55:00.17 DSNUSUPT - SYSTABSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.719076> DSNU610I  @ 021 01:55:00.19 DSNUSUPC - SYSCOLSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.719124> DSNU610I  @ 021 01:55:00.20 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.719205> DSNU610I  @ 021 01:55:00.20 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.719254> DSNU610I  @ 021 01:55:00.20 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.719513> DSNU610I  @ 021 01:55:00.21 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.719570> DSNU610I  @ 021 01:55:00.21 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.719617> DSNU610I  @ 021 01:55:00.21 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.719663> DSNU610I  @ 021 01:55:00.22 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.719901> DSNU610I  @ 021 01:55:00.22 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.719970> DSNU610I  @ 021 01:55:00.23 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.720018> DSNU610I  @ 021 01:55:00.23 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.720066> DSNU610I  @ 021 01:55:00.23 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.720113> DSNU610I  @ 021 01:55:00.24 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.720163> DSNU610I  @ 021 01:55:00.24 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.720254> DSNU610I  @ 021 01:55:00.24 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.720305> DSNU610I  @ 021 01:55:00.24 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.720352> DSNU610I  @ 021 01:55:00.24 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.729593> DSNU610I  @ 021 01:55:00.25 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.729696> DSNU610I  @ 021 01:55:00.27 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.729744> DSNU610I  @ 021 01:55:00.27 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.729814> DSNU610I  @ 021 01:55:00.28 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.729867> DSNU610I  @ 021 01:55:00.28 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.729915> DSNU610I  @ 021 01:55:00.28 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.729961> DSNU610I  @ 021 01:55:00.29 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.730008> DSNU610I  @ 021 01:55:00.29 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.730055> DSNU610I  @ 021 01:55:00.29 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.730102> DSNU610I  @ 021 01:55:00.29 DSNUSUTB - SYSTABLES CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.730146> DSNU610I  @ 021 01:55:00.33 DSNUSUCO - SYSCOLUMNS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.730190> DSNU610I  @ 021 01:55:00.40 DSNUSUTS - SYSTABLESPACE CATALOG UPDATE FOR DSN8DA1A.DSN8SA1E SUCCESSFUL
2010-01-21-01:55:02.730262> DSNU610I  @ 021 01:55:00.41 DSNUSUCD - SYSCOLDIST CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.730308> DSNU610I  @ 021 01:55:00.41 DSNUSUCD - SYSCOLDIST CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.730351> DSNU610I  @ 021 01:55:00.42 DSNUSUCD - SYSCOLDIST CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.730395> DSNU610I  @ 021 01:55:00.42 DSNUSUCD - SYSCOLDIST CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.730438> DSNU610I  @ 021 01:55:00.44 DSNUSUCD - SYSCOLDIST CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.730484> DSNU610I  @ 021 01:55:00.44 DSNUSUCD - SYSCOLDIST CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.730527> DSNU610I  @ 021 01:55:00.45 DSNUSUCD - SYSCOLDIST CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.730571> DSNU610I  @ 021 01:55:00.46 DSNUSUCD - SYSCOLDIST CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.730616> DSNU610I  @ 021 01:55:00.47 DSNUSUCD - SYSCOLDIST CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.730660> DSNU610I  @ 021 01:55:00.48 DSNUSUCD - SYSCOLDIST CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.730705> DSNU610I  @ 021 01:55:00.49 DSNUSUCD - SYSCOLDIST CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.730749> DSNU610I  @ 021 01:55:00.49 DSNUSUCD - SYSCOLDIST CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.730842> DSNU610I  @ 021 01:55:00.52 DSNUSUCD - SYSCOLDIST CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.730888> DSNU610I  @ 021 01:55:00.54 DSNUSUCD - SYSCOLDIST CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.730983> DSNU610I  @ 021 01:55:00.54 DSNUSUCD - SYSCOLDIST CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.731028> DSNU610I  @ 021 01:55:00.55 DSNUSUCD - SYSCOLDIST CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.731072> DSNU610I  @ 021 01:55:00.55 DSNUSUCD - SYSCOLDIST CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.731118> DSNU610I  @ 021 01:55:00.56 DSNUSUCD - SYSCOLDIST CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.731162> DSNU610I  @ 021 01:55:00.56 DSNUSUCD - SYSCOLDIST CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.731206> DSNU610I  @ 021 01:55:00.57 DSNUSUCD - SYSCOLDIST CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.731250> DSNU610I  @ 021 01:55:00.58 DSNUSUCD - SYSCOLDIST CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.731294> DSNU610I  @ 021 01:55:00.58 DSNUSUCD - SYSCOLDIST CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.731339> DSNU610I  @ 021 01:55:00.59 DSNUSUCD - SYSCOLDIST CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.731383> DSNU1374I @ 021 01:55:00.59 DSNUSEF2 - SAMPLE SIZE USED :
2010-01-21-01:55:02.731426>    ROWS SAMPLED    = 42
2010-01-21-01:55:02.731469>    PAGES SAMPLED   = 2
2010-01-21-01:55:02.731569>    SAMPLE RATE USED =  100.00
2010-01-21-01:55:02.731612>    REPEATABLE      = 0
2010-01-21-01:55:02.731654> DSNU610I  @ 021 01:55:00.62 DSNUSUIP - SYSINDEXPART CATALOG UPDATE FOR DSN8A10.XEMP1 SUCCESSFUL
2010-01-21-01:55:02.731719> DSNU610I  @ 021 01:55:00.62 DSNUSUPI - SYSINDEXSTATS CATALOG UPDATE FOR DSN8A10.XEMP1 SUCCESSFUL
2010-01-21-01:55:02.731767> DSNU610I  @ 021 01:55:00.63 DSNUSUPC - SYSCOLSTATS CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.731814> DSNU610I  @ 021 01:55:00.63 DSNUSUIP - SYSINDEXPART CATALOG UPDATE FOR DSN8A10.XEMP2 SUCCESSFUL
2010-01-21-01:55:02.731861> DSNU610I  @ 021 01:55:00.63 DSNUSUCO - SYSCOLUMNS CATALOG UPDATE FOR DSN8A10.XEMP1 SUCCESSFUL
2010-01-21-01:55:02.731907> DSNU610I  @ 021 01:55:00.64 DSNUSUIX - SYSINDEXES CATALOG UPDATE FOR DSN8A10.XEMP1 SUCCESSFUL
2010-01-21-01:55:02.731954> DSNU610I  @ 021 01:55:00.64 DSNUSUCO - SYSCOLUMNS CATALOG UPDATE FOR DSN8A10.XEMP2 SUCCESSFUL
2010-01-21-01:55:02.732032> DSNU610I  @ 021 01:55:00.64 DSNUSUIX - SYSINDEXES CATALOG UPDATE FOR DSN8A10.XEMP2 SUCCESSFUL
2010-01-21-01:55:02.732079> DSNU610I  @ 021 01:55:00.64 DSNUSUCD - SYSCOLDIST CATALOG UPDATE FOR DSN8A10.EMP SUCCESSFUL
2010-01-21-01:55:02.732123> DSNU620I  @ 021 01:55:00.64 DSNUSEOF - RUNSTATS CATALOG TIMESTAMP = 2010-01-21-01.54.56.534912
2010-01-21-01:55:02.732369> DSNU010I    021 01:55:00.68 DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```

Step 6: select * from sysibm.sysautoruns_hist (History for each autonomic stored procedure call, Start time, End time, what was done, etc)

HISTORY_...	PROC_NAME	STARTTS	ENDTS	OUTPUT	ERROR_M...	RETURN_C...	ROWID
863	SYSPROC.ADMIN_UTL_MONITOR	21.01.10 0...	21.01.10 0...	2010-01-2(...)		0	7e4a079f9...
864	ADMIN_UTL_EXECUTE	21.01.10 0...	21.01.10 0...	2010-01-2(...)	DSNT408I(...)	4	4037ef9f9d...

Step 7: some more details from the ADMIN_UTL_MONITOR call

OUTPUT (0,4)

```
2010-01-20-17:54:41.717833> Executing DSNX7MON
2010-01-20-17:54:41.717916> with options: statistics-scope=profile-consistency,restrict-ts="dbname <> 'DSNDB06'"
2010-01-20-17:54:41.717944> stored procedure begins at 2010-01-20-17:54:41.693357
2010-01-20-17:54:41.717963> ----------->>parsing parameters<<-----------
2010-01-20-17:54:41.717999> parsed option statistics-scope=profile-consistency
2010-01-20-17:54:41.718020> parsed option restrict-ts=dbname <> 'DSNDB06'
2010-01-20-17:54:41.718064> string option STATISTICS-SCOPE found
2010-01-20-17:54:41.718087> string option STAND-ALONE not found
2010-01-20-17:54:41.718114> string option RESTRICT-TS found
2010-01-20-17:54:41.718139> double option PCT-CHANGES not found
2010-01-20-17:54:41.718158> integer option NUM-CHANGES not found
2010-01-20-17:54:41.718177> integer option NUM-MASS-DELETES not found
2010-01-20-17:54:41.718195> integer option SAMPLING-THRESHOLD not found
2010-01-20-17:54:41.718214> integer option SAMPLING-RATE not found
2010-01-20-17:54:41.718987> double option TABCARD-LESS-THAN-COLCARD not found
2010-01-20-17:54:41.719025> double option TABCARD-LESS-THAN-COLGROUPCARD not found
2010-01-20-17:54:41.719044> double option SUM-OF-FREQUENCY-GREAT-THAN-ONE not found
2010-01-20-17:54:41.719063> double option FREQUENCY-OUT-OF-RANGE not found
2010-01-20-17:54:41.719082> double option NUMBER-OF-FREQUENCY-RECORDS-GREATER-THAN-COLGROUP-CARD not found
2010-01-20-17:54:41.719101> double option MAXIMUM-FREQUENCY-LESS-THAN-RECIPROCAL-OF-COLGROUP-CARD not found
2010-01-20-17:54:41.719122> double option COLGROUP-CARD-GREATER-THAN-SUPERSET-COLGROUP-CARD not found
2010-01-20-17:54:41.719142> double option PRODUCT-OF-COLCARD-LESS-THAN-COLGROUP-CARD not found
2010-01-20-17:54:41.719161> double option QUANTILE-CARD-GREATER-THAN-COLCARD not found
2010-01-20-17:54:41.719181> double option QUANTILE-CARD-GREATER-THAN-COLGROUP-CARD not found
2010-01-20-17:54:41.719200> double option SUM-OF-HISTOGRAM-GREATER-THAN-COLCARD not found
2010-01-20-17:54:41.719220> double option SUM-OF-HISTOGRAM-GREATER-THAN-COLGROUP-CARD not found
2010-01-20-17:54:41.719239> double option SUM-OF-HISTOGRAM-FREQUENCY-GREATER-THAN-ONE not found
2010-01-20-17:54:41.719259> double option QUANTILE-FREQUENCY-OUT-OF-RANGE not found
2010-01-20-17:54:41.719278> double option TABCARD-LESS-THAN-INDEX-KEYCARD not found
2010-01-20-17:54:41.719297> double option TABCARD-NOT-EQUAL-UNIQUE-INDEX-FULLKEYCARD not found
2010-01-20-17:54:41.719389> double option INDEX-FULLKEYCARD-LESS-THAN-FIRSTKEYCARD not found
2010-01-20-17:54:41.719411> double option INDEX-FULLKEYCARD-LESS-THAN-ANY-KEY-CARD not found
2010-01-20-17:54:41.719432> double option SINGLE-COL-INDEX-FULLKEYCARD-NOT-EQUAL-FIRSTKEYCARD not found
2010-01-20-17:54:41.719460> double option DIFFERENT-COLGROUP-CARD-FROM-INDEXES not found
2010-01-20-17:54:41.719480> double option DIFFERENT-COLGROUP-CARD-FROM-COLDIST-AND-INDEX not found
2010-01-20-17:54:41.719500> double option DIFFERENT-SINGLE-COLGROUP-CARD-FROM-INDEXES not found
2010-01-20-17:54:41.719519> double option DIFFERENT-SINGLE-COLGROUP-CARD-FROM-COLDIST-AND-INDEX not found
2010-01-20-17:54:41.719539> double option DRF-LESS-THAN-NPAGES not found
2010-01-20-17:54:41.719558> double option DRF-GREATER-THAN-TABCARD not found
2010-01-20-17:54:41.719579> ----------->>check statistics<<-----------
2010-01-20-17:54:41.737128> ALERT written for TABLE DSN8A10.DEPT REASON()
2010-01-20-17:54:41.782835> alert written for DSN8DA1A.DSN8SA1D partition 0 with option TABLE(DSN8A10.DEPT) TABLESAMPLE SYSTEM AUTO USE PROFILE
2010-01-20-17:54:41.791428> ALERT written for TABLE DSN8A10.EMP REASON()
2010-01-20-17:54:41.800665> alert written for DSN8DA1A.DSN8SA1E partition 0 with option TABLE(DSN8A10.EMP) TABLESAMPLE SYSTEM AUTO USE PROFILE
2010-01-20-17:54:41.809488> ALERT written for TABLE DSN8A10.DSN_QUERY_TABLE REASON()
2010-01-20-17:54:41.810484> alert written for DSN8DA1A.DSN8SA1I partition 0 with option TABLE(DSN8A10.DSN_QUERY_TABLE) TABLESAMPLE SYSTEM AUTO USE PROFILE
2010-01-20-17:54:41.816265> ALERT written for TABLE DSN8A10.EMPPROJACT REASON()
2010-01-20-17:54:41.835789> alert written for DSN8DA1A.DSN8SA1P partition 0 with option TABLE(DSN8A10.EMPPROJACT) USE PROFILE
2010-01-20-17:54:41.864793> ALERT written for TABLE DSN8A10.DSN_SORTKEY_TABLE REASON()
2010-01-20-17:54:41.944193> alert written for DSN8DA1A.DSN8SA1X partition 0 with option TABLE(DSN8A10.DSN_SORTKEY_TABLE) USE PROFILE
2010-01-20-17:54:41.959149> ALERT written for TABLE DSN8A10.EMP_PHOTO_RESUME REASON()
2010-01-20-17:54:41.960463> alert written for DSN8DA1L.DSN8SA1B partition 0 with option TABLE(DSN8A10.EMP_PHOTO_RESUME) TABLESAMPLE SYSTEM AUTO USE PROFILE
2010-01-20-17:54:42.032014> ALERT written for TABLE DSN8A10.PRODUCT REASON()
2010-01-20-17:54:42.033272> alert written for DSN8DA1X.DSN8SA1X partition 0 with option TABLE(DSN8A10.PRODUCT) USE PROFILE
2010-01-20-17:54:42.036028> DSN8DA1X.XPRO0000 is a LOB table space. We skip this table space.
2010-01-20-17:54:42.040624> ALERT written for TABLE DSN8A10.SALESFACT REASON()
2010-01-20-17:54:42.172392> alert written for DSN8DA1Y.DSN8SA1Y partition 0 with option TABLE(DSN8A10.SALESFACT) USE PROFILE
2010-01-20-17:54:42.174383> percentage and number of changes above threshold. Mark tablespace for an alert
2010-01-20-17:54:42.179651> ALERT written for TABLE SYSIBM.ADMIN_TASKS REASON()
2010-01-20-17:54:42.186315> alert written for DSNADMDB.DSNADMTS partition 0 with option TABLE(ALL) USE PROFILE
2010-01-20-17:54:42.207700> ALERT written for TABLE SYSIBM.MQSERVICE_TABLE REASON()
2010-01-20-17:54:42.213614> alert written for DSNMQDB .DSNMQTS  partition 0 with option TABLE(SYSIBM.MQSERVICE_TABLE) USE PROFILE
2010-01-20-17:54:42.234699> CHECK FINISHED!
2010-01-20-17:54:42.234767> ----------->>execution summary<<-----------
2010-01-20-17:54:42.234797> Number of table spaces checked: 17
2010-01-20-17:54:42.234821> Number of partitions checked (having DATASIZE > 0): 18
2010-01-20-17:54:42.234847> Number of tables checked: 71
2010-01-20-17:54:42.234866> Number of table spaces that need a RUNSTATS: 1
2010-01-20-17:54:42.234884> Number of partitions that need a RUNSTATS: 0
2010-01-20-17:54:42.234903> Number of tables that need a RUNSTATS: 9
2010-01-20-17:54:42.237343> ----------->>trigger ADMIN_UTL_EXECUTE<<-----------
2010-01-20-17:54:42.648012> ADMIN_UTL_EXECUTE successfully scheduled...
2010-01-20-17:54:42.648370> stored procedure ends at 2010-01-20-17:54:42.648353
2010-01-20-17:54:42.648390> Exiting...
```

Considerations

What to think about before setting up autonomic statistics

- Get used to running and understanding Stored Procedures
  - How do I run a Stored Procedure and how do I monitor the results?
  - Setting up the monitor requires one Stored Procedure to add another Stored Procedure that will invoke another Stored Procedure.
- Autonomic statistics first made its debut in DB2 LUW and was then ported to DB2 for z/OS
  - There is very limited information regarding how to setup and use autonomic statistics, some of the better material resides in the LUW manuals. For example, review: http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp?topic=%2Fcom.ibm.db2.luw.sql.rtn.doc%2Fdoc%2Fr0054371.html
  - The date and time coded in the Stored Procedures are in UNIX Cron format. Review the “UNIX cron Format” in the above link.

What to think about before setting up autonomic statistics

- On what schedule should RUNSTATS potentially execute?
- What happens if a RUNSTATS starts near the end of the time schedule and now has to run for an extensive period?
- How many RUNSTATS should run simultaneously?
- If running in a Data Sharing environment, which member should RUNSTATS and the monitor execute on?
- How often should the monitor Stored Procedure run?
- Are there objects that require special statistics?
- Which objects should be included or excluded for analysis?
- Should objects recommended for RUNSTATS execute or just be reported on?
- Which statistics scope should be run on which schedule?
- Will the ADMT scheduler be used or an operating system one?
- How often should the log file and alert history be cleaned up?
- How will the log file and alert history be reviewed for unsuccessful executions? What action should be taken?
- RUNSTATS of objects helped (or hurt) dynamic SQL, but what will you do about static and REBIND?
- Is DSNACCOX also run? If it is, how will it be used along side auto stats?
- How will I deal with auto stats running at the same time as my maintenance window for REORG then inline stats? How about scheduled RUNSTATS?

Running Stored Procedures

- You can write your own code for the Stored Procedures or use a tool such as Data Studio or Optim Query Tuner. My tests used OWQT 4.1.0.1.
- Use the Data perspective to run Stored Procedures in Data Studio or OWQT.
  - After connecting to a DB2, setup a SQL script by clicking on “Run SQL”
  - You may run one or more SQL statements and/or Stored Procedures
  - Results will be on the bottom of the screen
  - You may save your SQL scripts and results
- When coding Stored Procedures, be very careful of using ‘ vs ‘’ vs “ (single quote vs. double single quote vs. double quote). Incorrect usage will cause many frustrating failures.
- When Stored Procedures require a userid and password such as for admin_task_add you must use an authorized id and password used to sign onto TSO.

Scheduling autonomic statistics

- STC ADMT can be used to schedule the Stored Procedures required for autonomic statistics
  - STC ADMT running does not mean you are ready to go. You will require additional RACF authorization is receiving the following:

```
ICH408I USER(STCRACF ) GROUP(SYS1    ) NAME(RACF STC USERID     )
  IRRPTAUTH.DBP1ADMT.JOHNICZ CL(PTKTDATA)
  INSUFFICIENT ACCESS AUTHORITY
  FROM IRRPTAUTH.*.* (G)
  ACCESS INTENT(UPDATE )  ACCESS ALLOWED(NONE   )
```

Scheduling autonomic statistics

- Verify that optional install job DSNTIJRA was executed before scheduling any required Stored Procedures.
- If scheduled Stored Procedures do not execute during the required timeframe, execute the following (partial results):

```
SELECT * FROM TABLE (DSNADM.ADMIN_TASK_LIST()) AS T;
 USERID  POINT_IN_TIME PROCEDURE_SCHEMA PROCEDURE_NAME
 ------- ------------ ---------------- ------------------
 JOHNICZ */30 * * * *  SYSPROC          ADMIN_UTL_MONITOR
```

From the above SELECT we find that ADMIN_UTL_MONITOR is called every 30 minutes

```
SELECT * FROM TABLE (DSNADM.ADMIN_TASK_STATUS()) AS T;
 TASK_NAME          STATUS MSG
 ------------------ ------ ------------------------------------------------------------------------------------
 ADMIN_UTL_MONITOR_1 NOTRUN DSNA691I  DSNA6THD THE ADMIN SCHEDULER DBP1ADMT CANNOT GENERATE A PASSTICKET FOR TASK
```

From the above SELECT we find that the ADMIN_UTL_MONITOR Stored Procedure was not run because DSNTIJRA was not executed enabling pass tickets

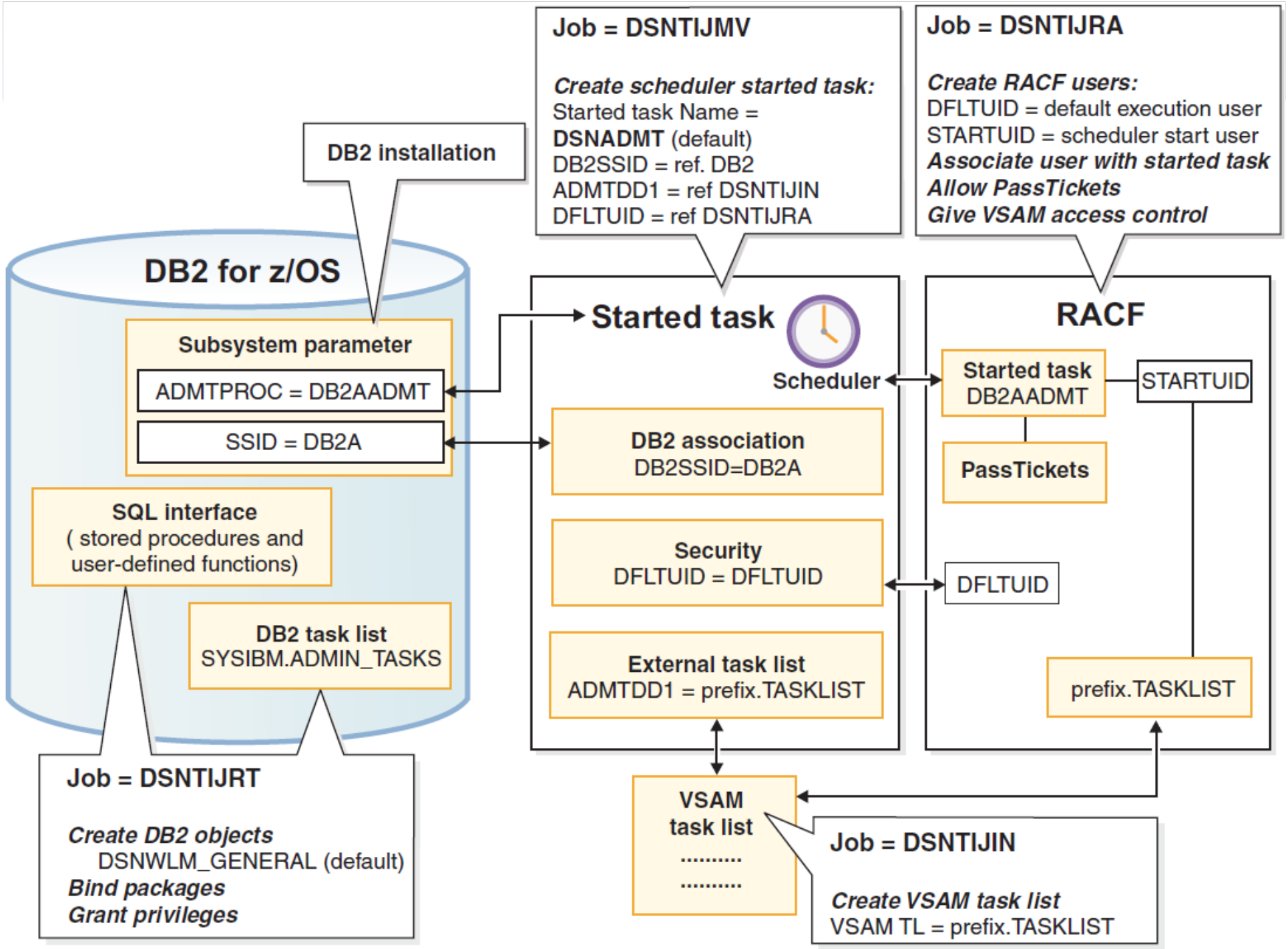
Scheduling autonomic statistics

- Output from DB2 command DISPLAY PROCEDURE shows that Stored Procedure ADMIN_UTL_MONITOR was not executed:

```
DSNX940I  -DBP1 DSNX9DIS DISPLAY PROCEDURE REPORT FOLLOWS -
------- SCHEMA=SYSIBM
PROCEDURE      STATUS ACTIVE QUED MAXQ TIMEOUT FAIL WLM_ENV
SQLCAMESSAGE
               STARTED     0    0    1       0    0 DBP1WLM_GENERAL
------- SCHEMA=SYSPROC
PROCEDURE      STATUS ACTIVE QUED MAXQ TIMEOUT FAIL WLM_ENV
DSNWZP
               STARTED     0    0    1       0    0 DBP1WLM_NUMTCB1
ADMIN_TASK_ADD
               STARTED     0    0    1       0    0 DBP1WLM_GENERAL
ADMIN_INFO_SSID
               STARTED     0    0    1       0    0 DBP1WLM_GENERAL
ADMIN_COMMAND_DB2
               STARTED     0    0    1       0    0 DBP1WLM_GENERAL
ADMIN_COMMAND_DSN
               STARTED     0    0    1       0    0 DBP1WLM_REXX
ADMIN_INFO_SYSPARM
               STARTED     0    0    1       0    0 DBP1WLM_NUMTCB1
DSNX9DIS DISPLAY PROCEDURE REPORT COMPLETE
DSN9022I  -DBP1 DSNX9COM '-DISPLAY PROC' NORMAL COMPLETION
```

DB2 output from command DISPLAY PROCEDURE when the monitor and execute Stored Procedures were run


```
DSNX940I  -DBP1 DSNX9DIS DISPLAY PROCEDURE REPORT FOLLOWS -
------- SCHEMA=SYSIBM
PROCEDURE      STATUS ACTIVE QUED MAXQ TIMEOUT FAIL WLM_ENV
SQLCAMESSAGE
               STARTED     0    0    1       0    0 DBP1WLM_GENERAL
------- SCHEMA=SYSPROC
PROCEDURE      STATUS ACTIVE QUED MAXQ TIMEOUT FAIL WLM_ENV
DSNWZP
               STARTED     0    0    1       0    0 DBP1WLM_NUMTCB1
DSNUTILU
               STARTED     0    0    1       0    0 DBP1WLM_UTILS
ADMIN_TASK_ADD
               STARTED     0    0    1       0    0 DBP1WLM_GENERAL
ADMIN_UTL_SORT
               STARTED     0    0    1       0    0 DBP1WLM_GENERAL
ADMIN_INFO_SSID
               STARTED     0    0    1       0    0 DBP1WLM_GENERAL
ADMIN_COMMAND_DB2
               STARTED     0    0    1       0    0 DBP1WLM_GENERAL
ADMIN_COMMAND_DSN
               STARTED     0    0    1       0    0 DBP1WLM_REXX
ADMIN_TASK_UPDATE
               STARTED     0    0    1       0    0 DBP1WLM_GENERAL
ADMIN_UTL_EXECUTE
               STARTED     0    0    1       0    0 DBP1WLM_GENERAL
ADMIN_UTL_MONITOR
               STARTED     0    0    1       0    0 DBP1WLM_PGM_CONTRO
                                                    L
ADMIN_INFO_SYSPARM
               STARTED     0    0    1       0    0 DBP1WLM_NUMTCB1
ADMIN_UTL_SCHEDULE
               STARTED     0    0    1       0    0 DBP1WLM_GENERAL
DSNX9DIS DISPLAY PROCEDURE REPORT COMPLETE
```


Job = DSNTIJMV
Create scheduler started task:
Started task Name =
DSNADMT (default)
DB2SSID = ref. DB2
ADMTDD1 = ref DSNTIJIN
DFLTUID = ref DSNTIJRA
Job = DSNTIJRA
Create RACF users:
DFLTUID = default execution user
STARTUID = scheduler start user
Associate user with started task
Allow PassTickets
Give VSAM access control
DB2 installation
DB2 for z/OS
Subsystem parameter
ADMTPROC = DB2AADMT
SSID = DB2A
SQL interface
(stored procedures and
user-defined functions)
DB2 task list
SYSIBM.ADMIN_TASKS
Started task
Scheduler
DB2 association
DB2SSID=DB2A
Security
DFLTUID = DFLTUID
External task list
ADMTDD1 = prefix.TASKLIST
RACF
Started task
DB2AADMT
STARTUID
PassTickets
DFLTUID
prefix.TASKLIST
Job = DSNTIJRT
Create DB2 objects
DSNWLM_GENERAL (default)
Bind packages
Grant privileges
VSAM
task list
.....
.....
Job = DSNTIJIN
Create VSAM task list
VSAM TL = prefix.TASKLIST

FAQ – Can AUTOSTATS and Manual RUNSTATS Coexist?

- When the DB2 subsystem is configured with AUTOSTATS enabled, would it still allow users (or tools) to retain the freedom to run RUNSTATS manually?
- Can these two modes of operation coexist?
- The answer is YES. AUTOSTATS and manual RUNSTATS run do not interfere with one another.
- Manual stats collection can be run from time to time, with or without AUTOSTATS enabled. There is no required changes from the usual practice or procedure.
- Be careful that manual and auto stats do not update the same objects.
- Review DB2 Utilities manual – see section “Combining autonomic and manual statistics maintenance”

FAQ – How do I use the scheduler in a Data Sharing environment?

- I have a three member Data Sharing environment DBP1, DBP2, and DBP3 – some options:
- Run ADMIN_UTL_MONITOR and ADMIN_UTL_EXECUTE on any member
  - For the sysautotimewindows table, specifying NULL for DB2_SSID would allow Stored Procedures ADMIN_UTL_MONITOR and ADMIN_UTL_EXECUTE to run on any member. Each Stored Procedure can run on different members.
  - Keep in mind, ADMT's TASKLIST data set and database DSNADMDB are shared by all three members, therefore all three members know that one of the members ran the task.
- Run ADMIN_UTL_MONITOR on member DBP2 only
  - For the sysautotimewindows table, specifying DBP2 for DB2_SSID would allow Stored Procedure ADMIN_UTL_MONITOR to run on member DBP2 only, but ADMIN_UTL_EXECUTE will run on any member
  - If you want both Stored Procedures ADMIN_UTL_MONITOR and ADMIN_UTL_EXECUTE to run on DBP2 only, invoke both with stand-alone=yes and use ADMIN_UTL_ADD to manually run ADMIN_UTL_EXECUTE. In this case ADMIN_UTL_EXECUTE will require very low overhead to periodically check the window set in sysautotimewindows.

IBM Automation Tool for autonomic statistics

- If you want to make your life much easier and not go through the manual steps to setup autonomic statistics which can be very error prone, use IBM's Automation tool. See:

http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=%2Fcom.ibm.db2tools.haa41.doc.ug%2Ftopics%2Fhaauc on_autostats_overview.htm

Using your own operating system scheduler instead of DB2's STC ADMT

- If you choose your own operating system scheduler instead of DB2's STC ADMT:
  - Pass the required parameters to your operating system scheduler
  - Schedule the ADMIN_UTL_MONITOR and the ADMIN_UTL_EXECUTE Stored Procedures with the stand-alone parameter.
  - Be aware that in this case ADMIN_UTL_EXECUTE cannot be scheduled by ADMIN_UTL_MONITOR or ADMIN_UTL_EXECUTE to resolve the alerts.
  - Schedule ADMIN_UTL_EXECUTE to run to resolve alerts

Manuals and sites for autonomic statistics

- DB2 Installation and Migration Guide – review sections regarding the auto stats Stored Procedures
- DB2 Managing Performance – see section “Automating statistics maintenance”
- DB2 SQL Reference – see section “DB2 catalog tables”. Review SYSIBM.SYSAUTOALERTS, SYSIBM.SYSAUTORUNS_HIST, and SYSIBM.SYSAUTOTIMEWINDOWS. Table functions ADMIN_TASK_LIST and ADMIN_TASK_STATUS
- DB2 Utilities manual – see section “Combining autonomic and manual statistics maintenance”
- DB2 10 for z/OS Technical Overview Redbook – see sections DB2-supplied stored procedures, Administrative task scheduler, Administration enablement, DB2 statistics routines, Autonomic statistics, Using RUNSTATS profiles, Updating RUNSTATS profiles, Deleting RUNSTATS profiles, Combining autonomic and manual statistics maintenance
- http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp?topic=%2Fcom.ibm.db2.luw.sql.rtn.doc%2Fdoc%2Fr0054371.html

Test and results

Test with five variations – all table spaces, no indexes

1. JOHNITS1 – inserted one row followed by regular RUNSTATS TABLE(ALL)
2. JOHNITS2 – inserted many identical rows without any RUNSTATS
3. JOHNITS3 – inserted many rows with a variation, 99% zipcode 23190, 1 % with 11234 followed by regular RUNSTATS TABLE(ALL)
4. JOHNITS4 – inserted many rows with a variation, 99% zipcode 23190, 1 % with 11234 followed by special RUNSTATS with COLGROUP for the zip code and SET PROFILE, then double the size of the table with no RUNSTATS
5. JOHNITS5 – inserted many rows followed by regular RUNSTATS TABLE(ALL), then DELETE from table (mass delete) with no RUNSTATS

Results after autonomic statistics run with BASIC, partial output from sysibm.sysautoalerts


```
ACTION   TARGET_QUALIFIER TARGET_OBJECT OPTIONS
-------- --------------- ------------- -------------------------------------
RUNSTATS DSNDB04         JOHNITS2      TABLE("JOHNICZ"."JOHNITB2") USE PROFILE
RUNSTATS DSNDB04         JOHNITS4      TABLE("JOHNICZ"."JOHNITB4") USE PROFILE
```

1. JOHNITS1 – inserted one row followed by regular RUNSTATS TABLE(ALL) – auto RUNSTATS not run
2. JOHNITS2 – inserted many identical rows without any RUNSTATS – auto RUNSTATS executed
3. JOHNITS3 – inserted many rows with a variation, 99% zipcode 23190, 1 % with 11234 followed by regular RUNSTATS TABLE(ALL) – auto RUNSTATS not run
4. JOHNITS4 – inserted many rows with a variation, 99% zipcode 23190, 1 % with 11234 followed by special RUNSTATS with COLGROUP for the zip code and SET PROFILE, then double the size of the table with no RUNSTATS – auto RUNSTATS executed
5. JOHNITS5 – inserted many rows followed by regular RUNSTATS TABLE(ALL), then DELETE from table (mass delete) with no RUNSTATS – auto RUNSTATS not run. ***WARNING – RUNSTATS should have run because RTS shows mass deletes>0. See APAR PM95437. Without this fix autostats is ignoring table spaces with a RTS datasize of zero***

Results after autonomic statistics run with BASIC, partial output from sysibm.sysautoalerts and sysibm.sysautoruns_hist


```
sysibm.sysautoalerts
ACTION   TARGET_QUALIFIER TARGET_OBJECT OPTIONS
-------- --------------- ------------- ------------------------------------
RUNSTATS DSNDB04         JOHNITS2      TABLE("JOHNICZ"."JOHNITB2") USE PROFILE
RUNSTATS DSNDB04         JOHNITS4      TABLE("JOHNICZ"."JOHNITB4") USE PROFILE

sysibm.sysautoruns_hist
TABLESPACE DSNDB04.JOHNITS2 REASON(no recent statistics found) ALERT written on
DSNDB04.JOHNITS2 with options (TABLE("JOHNICZ"."JOHNITB2") USE PROFILE )
TABLESPACE DSNDB04.JOHNITS4 REASON(no recent statistics found) ALERT written on
DSNDB04.JOHNITS4 with options (TABLE("JOHNICZ"."JOHNITB4") USE PROFILE )
```

1. JOHNITS2 – inserted many identical rows without any RUNSTATS – auto RUNSTATS executed
2. JOHNITS4 – inserted many rows with a variation, 99% zipcode 23190, 1 % with 11234 followed by special RUNSTATS with COLGROUP for the zip code and SET PROFILE, then double the size of the table with no RUNSTATS – auto RUNSTATS executed

Notes regarding this test

- Could not export nor save the results correctly in xls or notepad correctly. The xls version did not save all of the data, some was truncated, the notepad and wordpad versions did not process the title breaks appropriately. Needed to manually save the data to a txt file and then use a third party tool to open the data correctly.
- Autonomic statistics does not replace the Data Studio or OWQT Stats Advisor. In the case of JOHNITS3 and JOHNITS4 auto statistics did not use a COLGROUP even though there was a distribution issue. You must still periodically run Stats Advisor. Auto stats will use the RUNSTATS PROFILE once set.
- Autonomic statistics does not integrate with DB2 11 SYSIBM.SYSSTATFEEDBACK and DSN_STAT_FEEDBACK.

Autonomic statistics use of SYSIBM.SYSTABLES_PROFILES

```
SCHEMA   TBNAME   PROFILE_TYPE PROFILE_TEXT                                          PROFILE_UPDATE             PROFILE_USED
-------- -------- ------------ ------------------------------------------------------ -------------------------- ----------------
JOHNICZ  JOHNITB4 RUNSTATS     COLUMN (ZIPC) COLGROUP (ZIPC) FREQVAL COUNT 15 MOST 2014-02-13 15:20:35.443766 2014-02-13
15:20:35.443766
JOHNICZ  JOHNITB1 RUNSTATS     COLUMN("FNAME","LNAME",“ZIPC") INDEX(ALL)          2014-02-17 17:00:05.499402 NULL
JOHNICZ  JOHNITB2 RUNSTATS     INDEX(ALL)                                         2014-02-17 17:00:05.500781 NULL
JOHNICZ  JOHNITB3 RUNSTATS     COLUMN("FNAME","LNAME",“ZIPC") INDEX(ALL)          2014-02-17 17:00:05.502202 NULL
JOHNICZ  JOHNITB5 RUNSTATS     COLUMN("FNAME","LNAME",“ZIPC") INDEX(ALL)          2014-02-17 17:00:05.514561 NULL
```

- JOHNITB4 was manually added via RUNSTATS SET PROFILE without INDEX(ALL), all others added automatically added by auto stats.
- INDEX(ALL) is added to the ones auto stats added even though none of the tables have indexes. This is done in case indexes are added in the future.
- Output from `sysibm.sysautoalerts`

```
DSNUGPRF - THE STATS PROFILE WITH STATSTIME = 0000-00-00-00.00.00.000000  FOR TABLE
2014-02-17 17:00:12.145416>  JOHNITB2 HAS BEEN USED
DSNUGPRF - THE STATS PROFILE WITH STATSTIME = 2014-02-13-15.20.35.443766  FOR TABLE
2014-02-17 17:00:12.147256>  JOHNITB4 HAS BEEN USED
```

Additional thoughts

- Use the Query Tuner tool to interface with DB2 11 SYSIBM.SYSSTATFEEDBACK and DSN_STAT_FEEDBACK using special Stored Procedures.
- Consider using SELECT with SELECTIVITY in DB2 11 to override filter factors instead of executing special RUNSTATS when they apply to only a small fraction the SQL run.
  - Also consider the use of DSN_PREDICATE_SELECTIVITY outlined in the Managing Performance Guide

John Iczkovits
IBM
iczkovit@us.ibm.com

Session: A14

Title: DB2 10 and 11 for z/OS – Implementing and Using Autonomic Statistics

Please fill out your session evaluation before leaving!