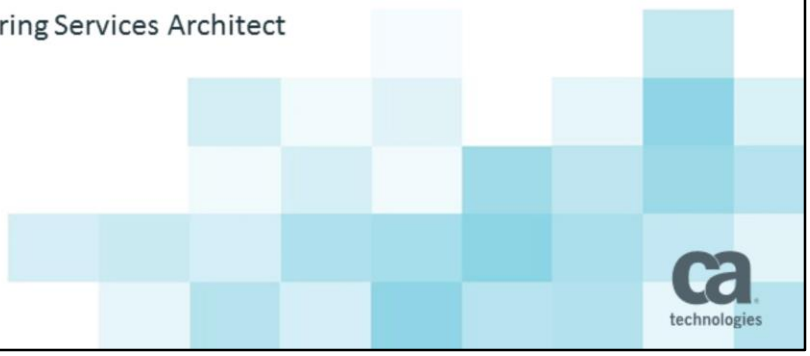
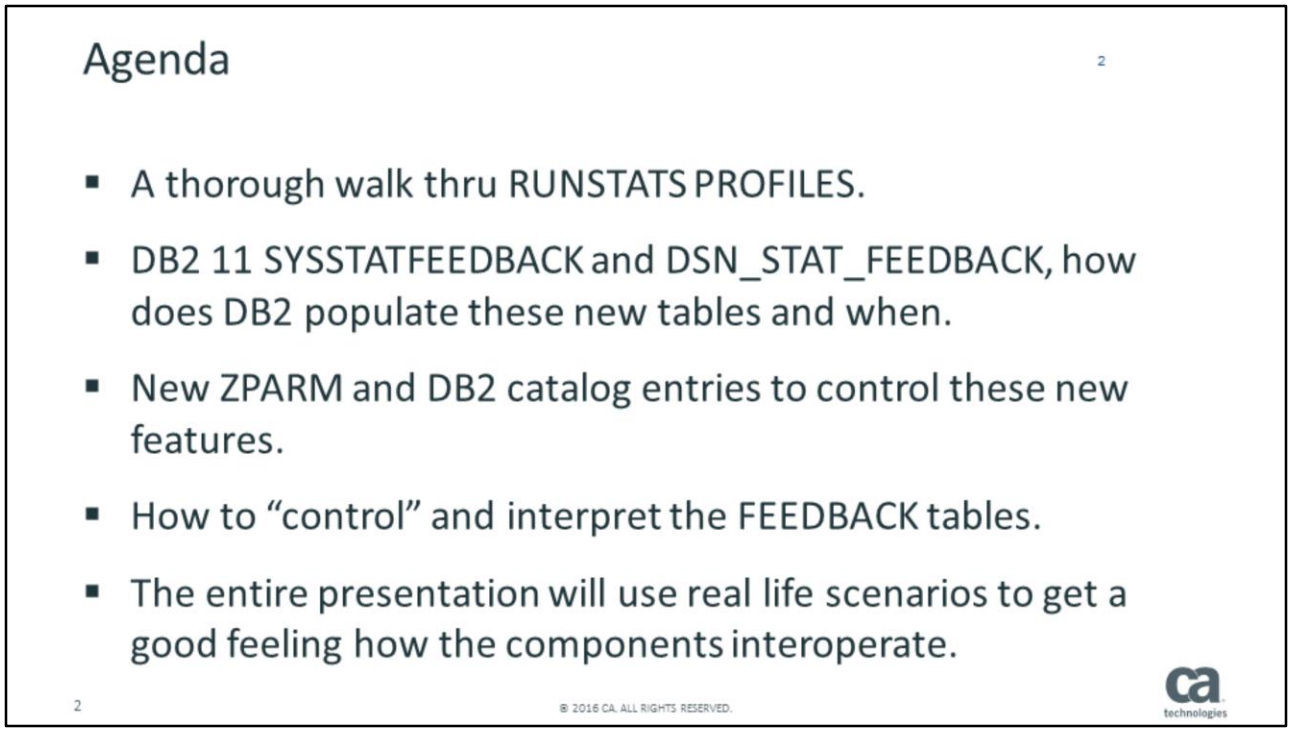
Too Many Fights in Your Marriage with Runstats ?
Try Profiles and Feedback Tables.

Steen Rasmussen, Sr. Engineering Services Architect
CA technologies

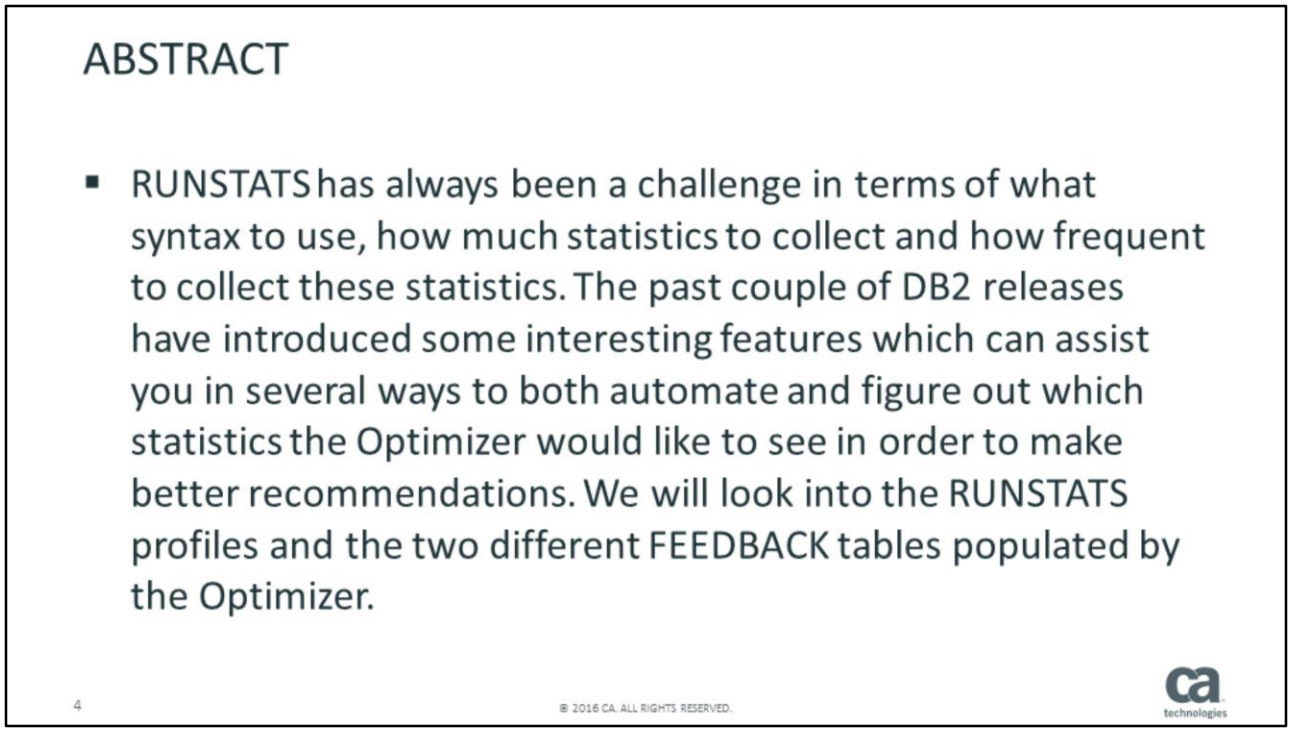




Runstats has always been a challenge in terms of what syntax to use, how much statistics to collect and how frequent to collect these statistics. The past couple of DB2 releases have introduced some interesting features which can assist you in several ways to both automate and figure out which statistics the Optimizer would like to see in order to make better recommendations. We will look into the RUNSTATS profiles and the two different FEEDBACK tables populated by the Optimizer.

DISCLAIMER

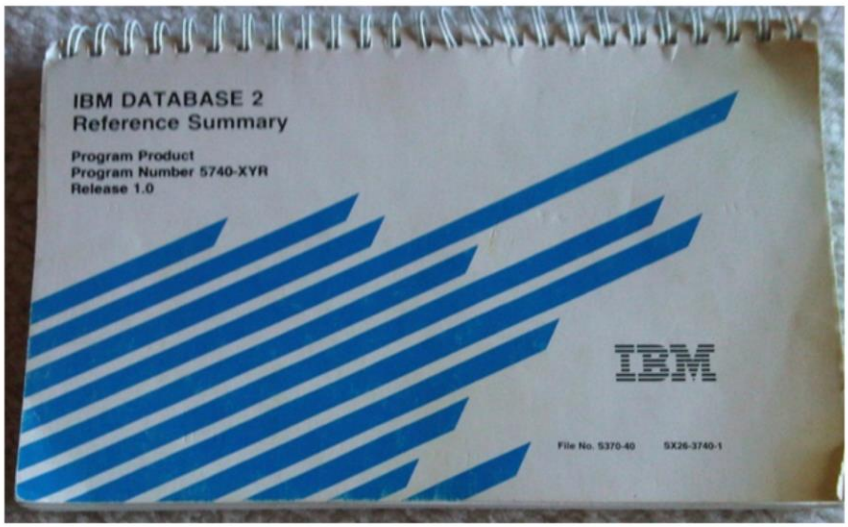
- Expressions are purely my own – not CA technologies.
- This presentation is based on using a real life DB2 11 system with maintenance as of July 2015 – discrepancies might exist.
- Your experiences may vary.



RUNSTATS has always been a challenge in terms of what syntax to use, how much statistics to collect and how frequent to collect these statistics. The past couple of DB2 releases have introduced some interesting features which can assist you in several ways to both automate and figure out which statistics the Optimizer would like to see in order to make better recommendations. We will look into the RUNSTATS profiles and the two different FEEDBACK tables populated by the Optimizer.

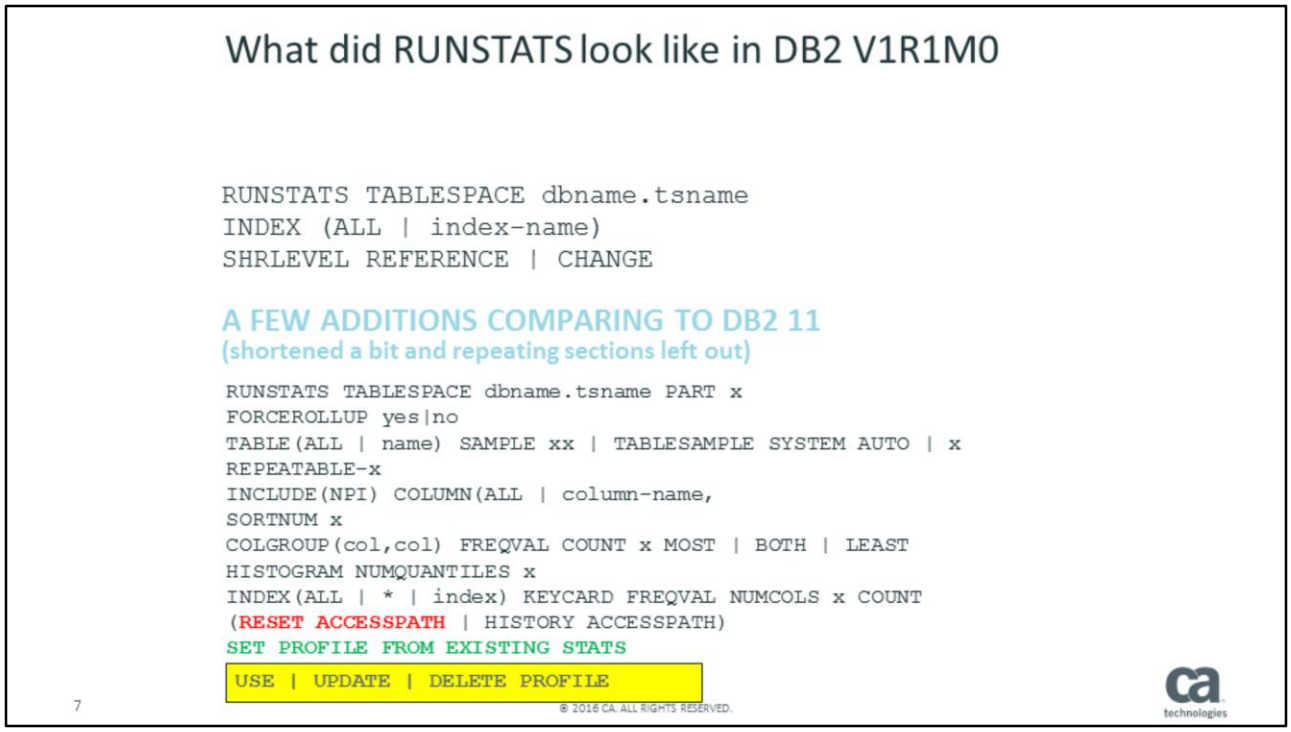
RUNSTATS PROFILES

RUNSTATS EVOLUTION over 30 years



6

RUNSTATS has indeed evolved over the past +30 years – lets look at what you could do in DB2 V1 compared to DB2 11



The very first release of DB2 provided very simple RUNSTATS syntax – not much to consider.
Looking at all the new features/keywords introduced over the next +10 releases leaves a lot to consider and think about. The task of deciding WHAT to RUNSTATS and which parameters to use has NOT gotten any easier – but it is even more crucial than ever to provide the Optimizer with the “correct” keywords in order to get the best possible performance.
DB2 10 introduced PROFILES making it easier to “memorize” what to RUNSTATS on the TABLE level, and we will look into the details how to exploit this.
Still – we need to consider WHICH parameters to use depending on the static SQL being executed as well as the more and more frequent dynamic SQL – DB2 11 introduced some cool new features to assist in this space and we will cover all of these topics in the next 45 minutes.

The newer parameters are highlighted in RED / GREEN and YELLOW.

RUNSTATS PROFILES

- Finding the “correct” RUNSTATS parameters isn’t easy
- Once you have the “correct” syntax:
  - Where do you save the parameters
  - How do you re-use the syntax
  - PROFILES can be the rescue
- However – Runstats Profiles don’t protect you 100%
  - Can still be overridden
  - Can still be modified
  - Not mandatory to use profile once defined
  - Maybe automation is a great idea ?

8

Based on all the changes to the RUNSTATS syntax – and the Optimizer being more and more sensible/sensitive to the statistics provided, finding the best RUNSTATS syntax is not an easy task.
Once you have identified the “perfect” statements – how do you keep track of these in order to be able to use these next time. Runstats PROFILES can be your lifeline here.

However – once you have created a profile, this is not a guarantee that someone won’t mess it up. These can still be overridden and modified and even deleted – and there is no way to make sure the profile is always used (and there are some limitations too).

Considering the advantages of profiles (we will see these) maybe this is a great time to install a process to always use profile if available. This requires standards and discipline !!!

RUNSTATS PROFILES

- The components in DB2 10 and DB2 11

```
CREATE TABLE "SYSIBM".SYSTABLES_PROFILES
  ("SCHEMA" VARCHAR(128)                        FOR MIXED DATA NOT NULL
   ,TBNAME VARCHAR(128)                         FOR MIXED DATA NOT NULL
   ,PROFILE_TYPE VARCHAR(32)                    FOR MIXED DATA NOT NULL
   ,PROFILE_MODE VARCHAR(32)          FOR MIXED DATA WITH DEFAULT NULL
   ,PROFILE_TEXT CLOB(1M)             FOR MIXED DATA WITH DEFAULT NULL
   ,ROWID ROWID                                NOT NULL GENERATED ALWAYS
   ,PROFILE_UPDATE TIMESTAMP (6) WITHOUT TIME ZONE           NOT NULL
   ,PROFILE_USED TIMESTAMP (6) WITHOUT TIME ZONE    WITH DEFAULT NULL
   ,CONSTRAINT "SCHEMA" PRIMARY KEY    ("SCHEMA",TBNAME,PROFILE_TYPE)
   )     IN DSNDB06.SYSTSTPF
   CCSID UNICODE  PARTITION BY SIZE EVERY 64G;
```

9

DB2 10 introduced a new catalog table : SYSIBM.SYSTABLES_PROFILES.

There are a number of important “pieces” to pay attention to once you start to exploit this feature:

1) It is TABLE based – not tablespace or index
2) No indexes are provided out-of-the-box, so if you really start to exploit, additional indexes might be useful depending on how you are exploiting this feature.
3) The most important column is PROFILE_TEXT which is a 1M CLOB column making some tasks a challenge which we will cover later.

Create a profile

- Has to be table specific – even though the tablespace only has/can hold one table
- This syntax is commonly used – but does not work with PROFILE

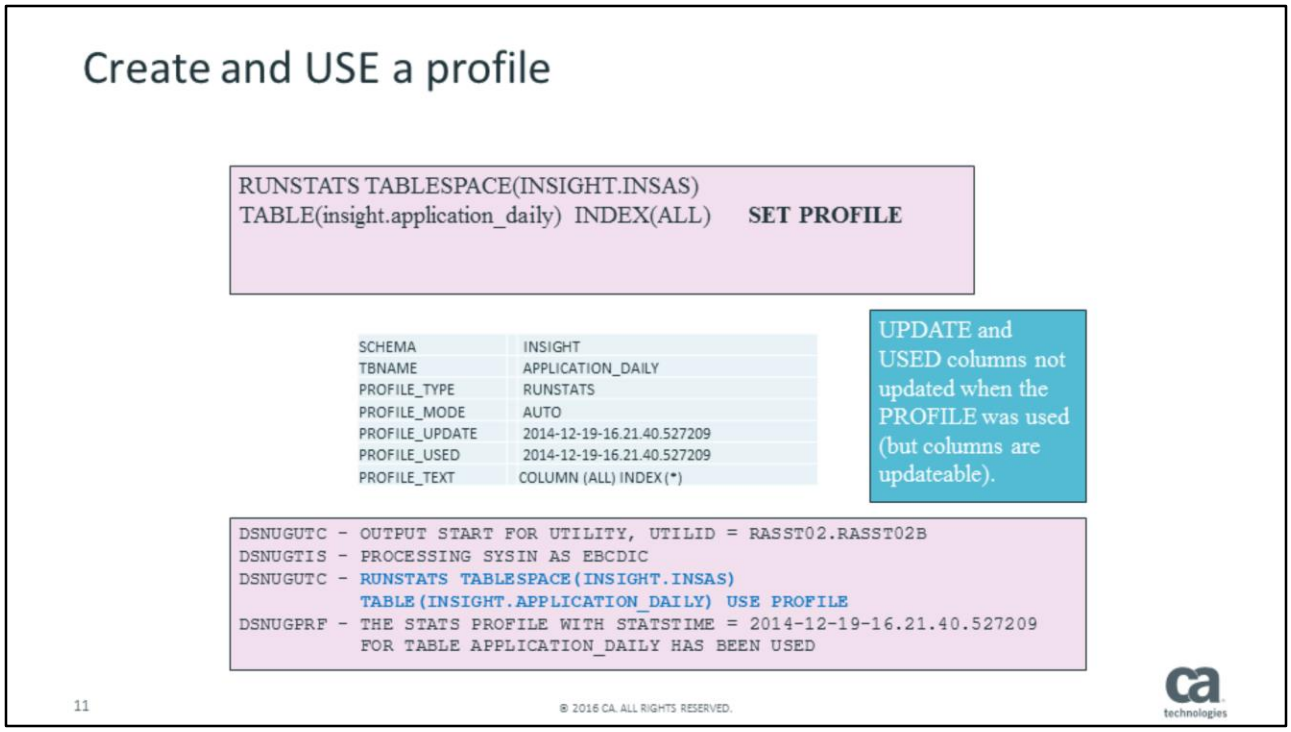
```
RUNSTATS TABLESPACE(INSIGHT.INSAS) TABLE(ALL)
INDEX(ALL)
SET PROFILE

KEYWORD OR OPERAND 'SET PROFILE' INVALID WITH
'TABLE ALL'
```

10

When you look at the table DDL – you can see the tablespace and database names are not recorded in the RUNSTATS PROFILE table – only the table creator and table name, so a PROFILE is table specific.

Some of the RUNSTATS syntax you have been using for years might not work when creating and updating profiles. Since the PROFILE is table specific and you potentially could have multiple tables in one tablespace, when a PROFILE is created (using SET PROFILE command) you will have to specify the table(s).

Note: the SET PROFILE does NOT execute RUNSTATS – all that happens is the profile is created in SYSIBM.SYSTABLES_PROFILES.



Once the SET PROFILE syntax has been corrected, a row is inserted into the PROFILE table. The PROFILE_TEXT (CLOB column) will hold everything but the TABLESPACE and TABLE parameters from the syntax.

Two columns are not really used (but you can update if needed):

PROFILE_UPDATE describes when the PROFILE row was updated / inserted. PROFILE_USED unfortunately doesn't tell you when this profile was used last time which would have been nice.

In order for RUNSTATS to use the profile you have created, simply execute the RUNSTATS with the tablespace name and table name ONLY – and specify USE PROFILE.

Use profile with additional keywords

- Not possible to specify USE PROFILE and additional keyword(s) – you will have to UPDATE the PROFILE and then USE the PROFILE with the RUNSTATS job

```
RUNSTATS  TABLESPACE(INSIGHT.INSAS)
TABLE(insight.application_daily)  INDEX(ALL)

COLGROUP  (BEGIN_DATE_TIME,  SYSTEM_ID)

USE PROFILE

KEYWORD OR OPERAND 'USE PROFILE' INVALID WITH  'COLGROUP'
```

12

If you want to add additional keywords to RUNSTATS and at the same time use PROFILE – this is not supported.
You will have to first UPDATE the profile and then USE the profile – so it is a two-step process, but then the profile is changed for good which might defeat the purpose.
Of course it is possible to do RUNSTATS without specifying a profile at all – even though a profile does exist.

Use profile with additional keywords

- Once the PROFILE has been updated with additional keywords/parameters – then USE PROFILE can be used to execute modified runstats *(the CLOB is modified accordingly)*

```
RUNSTATS TABLESPACE(INSIGHT.INSAS)
TABLE(insight.application_daily) INDEX(ALL)
COLGROUP (BEGIN_DATE_TIME, SYSTEM_ID)
UPDATE PROFILE
```

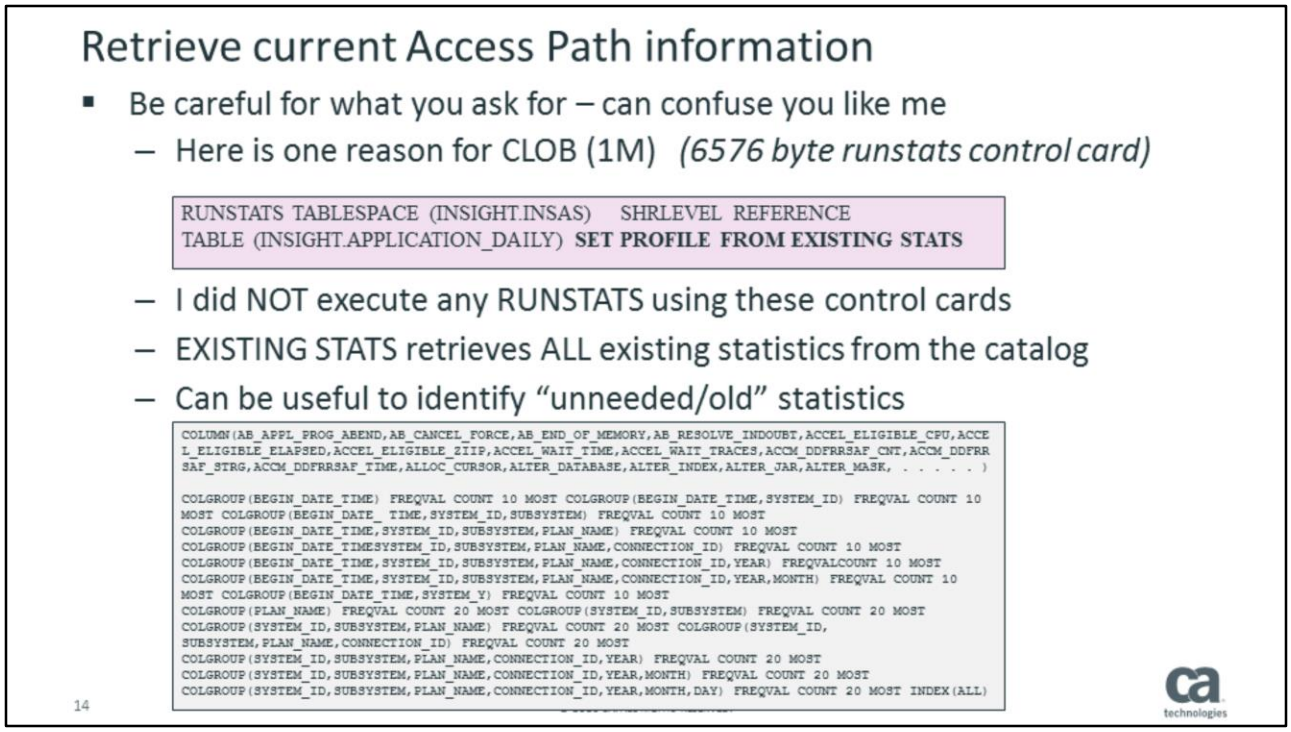
SCHEMA	INSIGHT
TBNAME	APPLICATION_DAILY
PROFILE_TYPE	RUNSTATS
PROFILE_MODE	AUTO
PROFILE_UPDATE	2014-12-19-17.17.38.500090
PROFILE_USED	2014-12-19-17.17.38.500090
PROFILE_TEXT	COLUMN (ALL) COLGROUP (BEGIN_DATE_TIME, SYSTEM_ID) INDEX (*)

13

Once you update your RUNSTATS statement and use UPDATE PROFILE, you will be able to use RUNSTATS and USE PROFILE with the additional keywords.

As you can tell from the PROFILE_TEXT, it will keep on being appended with the keywords you add using UPDATE PROFILE.

Again – please remember that RUNSTATS is NOT executed when using UPDATE PROFILE – it is still a two-step procedure to get the job done.



In case you don't know WHAT kind of access path is present in the DB2 catalog – you can retrieve the RUNSTATS KEYWORDS needed to produce what is present:

Using the SET PROFILE FROM EXISTING STATS

This kind of tells you why the PROFILE_TEXT is a 1M CLOB column – there can be a LOT of information in the catalog assisting the DB2 Optimizer. This will also provide you information about what OTHERs have RUNSTAT'ed – and it can be very helpful to identify what kind of statistics you don't want to be collected.

In this case, I never created the control cards for RUNSTATS to collect all these COLGROUPs – someone else must have requested these, but this syntax does retrieve what's present in the catalog.

Be careful however – since now the PROFILE will hold statistics to collect which you might not care about – but it is a great help and you can save these control cards (somehow) and RESET the current statistics (more about this later).

In this case – the RUNSTATS syntax being recorded in the PROFILE is almost 7000 byte . . . . . !!!!!!!!!!!!!!!!

Good luck formatting and changing – we will look at this later since there are some challenges.

Reset current Access Path information

- This can be very useful to "clean up" all existing access path information from the catalog and create a new beginning

```
RUNSTATS TABLESPACE (INSIGHT.INSAS)          RESET ACCESSPATH
```

- If you re-execute the RESET, RUNSTATS does verify that the statistics are gone

```
DSNUGUTC - RUNSTATS TABLESPACE(INSIGHT.INSAS) SHRLEVEL REFERENCE
TABLE (INSIGHT.APPLICATION_DAILY)   RESET PROFILE FROM EXISTING STATS

DSNUSITS - NO EXISTING STATISTICS AVAILABLE FOR TABLE APPLICATION_DAILY
DSNUGBAC - UTILITY EXECUTION TERMINATED, HIGHEST RETURN CODE=8 p
```

15

ca technologies

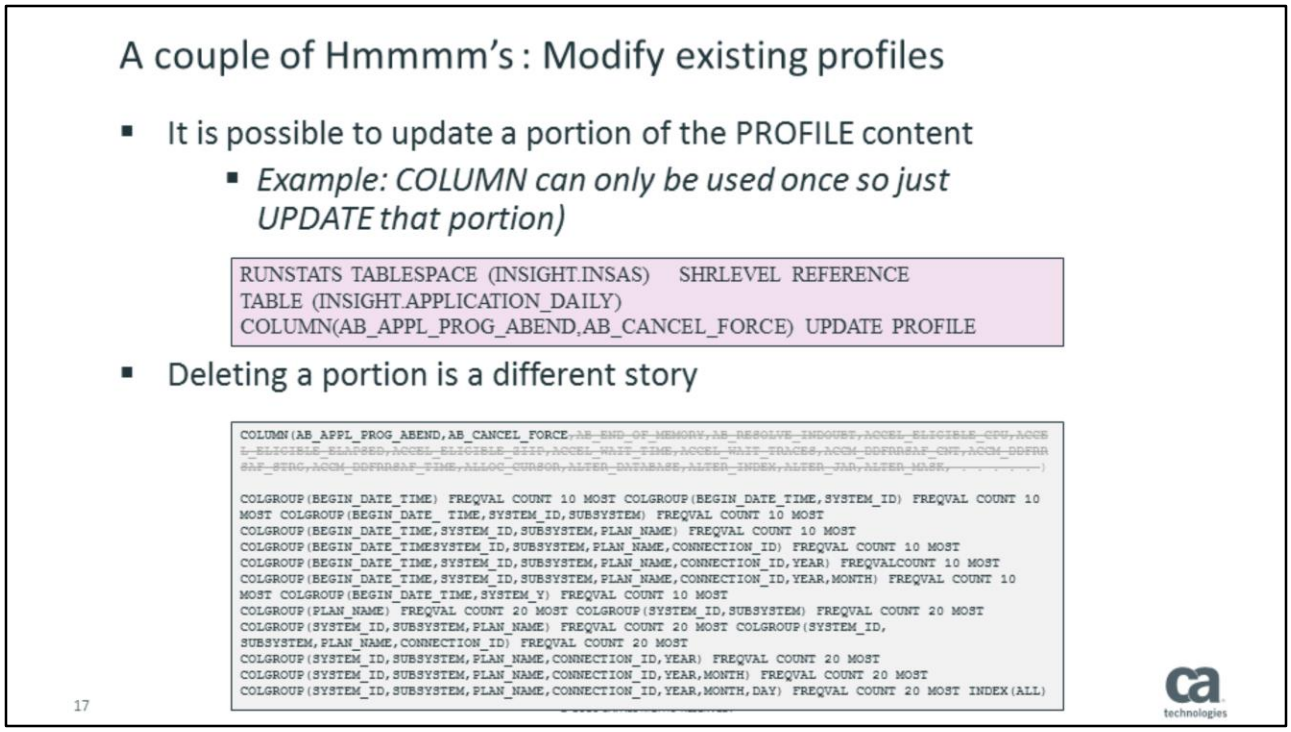
Looking at the example on the previous slide – you might find you have way too much or too outdated statistics in the catalog for the Optimizer to consider – you can very easily eliminate ALL existing statistics (or stale statistics) by using the RESET ACCESSPATH keywords with RUNSTATS (without using PROFILE).

Executing the RESET ACCESSPATH a second time without any RUNSTATS in-between illustrates the statistics used by the optimizer is really gone. The catalog will look like the object was just created – having negative one all over the place.

Reset current Access Path information

- Partition level statistics not cleaned up
  - SYSINDEXPART and SYSTABLEPART not reset during RESET ACCESSPATH execution
  - Be aware of stale statistics

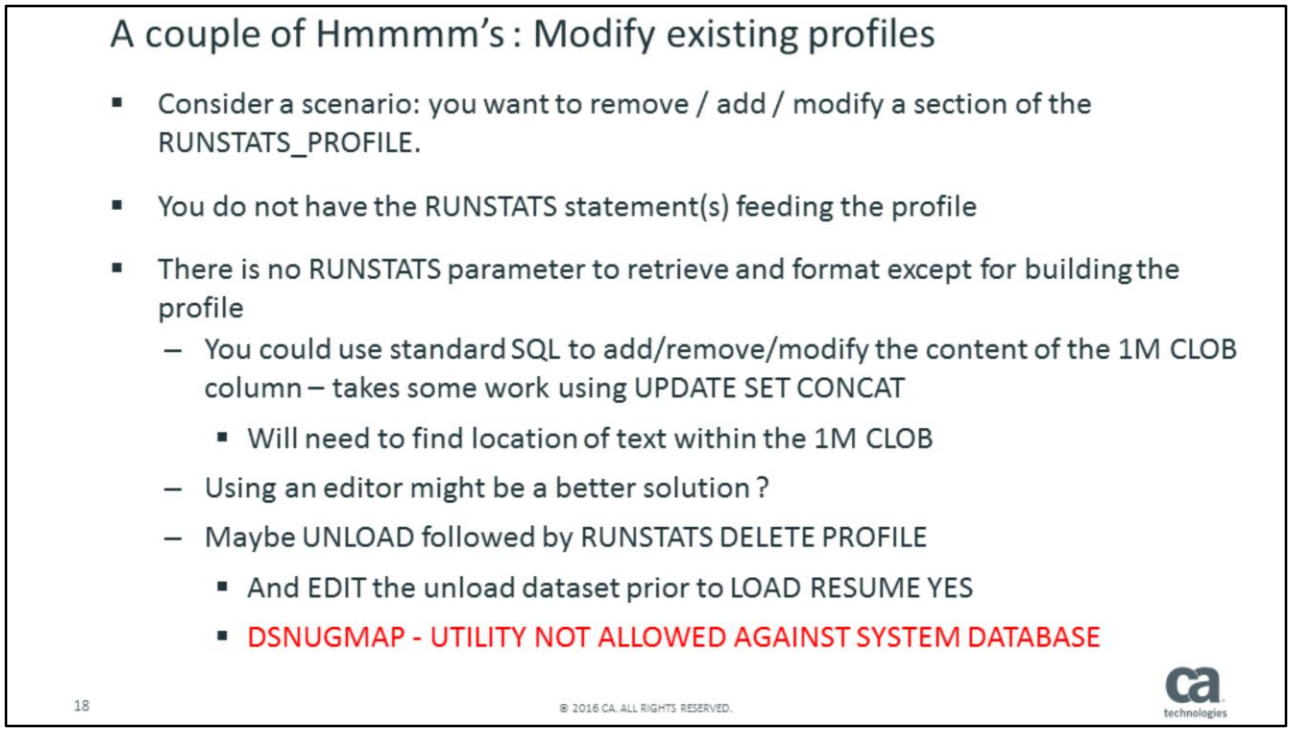
```
DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = RASST02.RASST02B
DSNUGTIS - PROCESSING SYSIN AS EBCDIC
DSNUGUTC -  RUNSTATS TABLESPACE(IDUG2015.SYSTSTAB) RESET ACCESSPATH
DSNUSRST - SYSCOLSTATS CATALOG ACCESSPATH STATISTICS RESET SUCCESSFUL
DSNUSRST - SYSTABSTATS CATALOG ACCESSPATH STATISTICS RESET SUCCESSFUL
DSNUSRST - SYSCOLDISTSTATS CATALOG ACCESSPATH STATISTICS RESET SUCCESSFUL
DSNUSRST - SYSCOLUMNS CATALOG ACCESSPATH STATISTICS RESET SUCCESSFUL
DSNUSRST - SYSTABLES CATALOG ACCESSPATH STATISTICS RESET SUCCESSFUL
DSNUSRST - SYSCOLDIST CATALOG ACCESSPATH STATISTICS RESET SUCCESSFUL
DSNUSRST - SYSTABLESPACE CATALOG ACCESSPATH STATISTICS RESET SUCCESSFUL
DSNUSRST - SYSINDEXES CATALOG ACCESSPATH STATISTICS RESET SUCCESSFUL
DSNUSRST - RUNSTATS CATALOG TIMESTAMP = 2015-07-13-15.37.44.491430
DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0


RUNSTATS TABLESPACE(PTDB.PTG600T2 PART 4) RESET ACCESSPATH
KEYWORD OR OPERAND 'PART' INVALID WITH 'RESET ACCESSPATH'
```



Let's have a closer look at how you can modify the content of a PROFILE.

Executing RUNSTATS with a KEYWORD (here COLUMN) will replace that specific part of the profile.
You can also MODIFY an existing COLGROUP the same way.

If you want to DELETE a part of the profile – that is a complete different story which we will cover later.



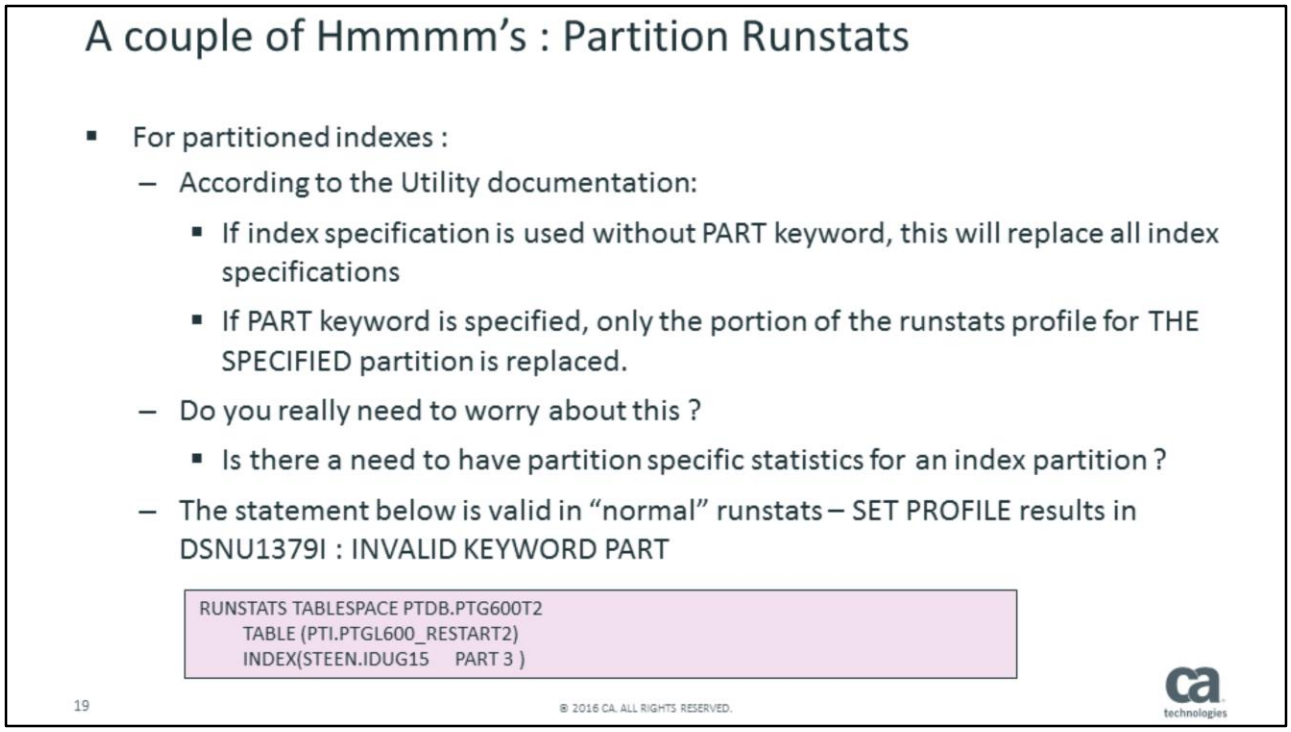
We just covered how to modify one or more parts of the profile – simply using the KEYWORD with the new content.

Here is the issue: there is no way that I know of to retrieve the PROFILE content – modify it and then reload the profile.

What I have done successfully is – one of two methods:

1) Use SQL UPDATE statement with CONCAT to modify the content. After all, it's a CLOB column so anything you can think of using SQL applies.
2) You could also UNLOAD the content – then modify it – execute runstats with DELETE PROFILE and do a load resume yes.
3) Finally – if you have an editor in place, simply edit the content.

However – be careful since RUNSTATS with syntax USE PROFILE is picky and will tell you if "he/she" doesn't like what you have done.



Another interesting story – unfortunately I have NOT been able to verify the documentation.
The IBM documentation says that if you currently have partition specific keywords in the profile, updating the profile with non-specific partition, the existing profile content for the index partitions will be wiped out.

From my perspective – I really don't worry about this since I don't see a major need for collecting specific stats for specific index partitions.

The RUNSTATS statement for partition 3 in the box executes fine without specifying profile but it fails if you attempt to set / update the profile.

Why do we care so much about RUNSTATS ?
. . . and REBIND for static SQL
It all boils down to “The nightmare of AP Changes”

A topic which often turns into a very heated discussion . . . . . When to RUNSTATS and REBIND.

Different RUNSTATS and REBIND strategies from around the world

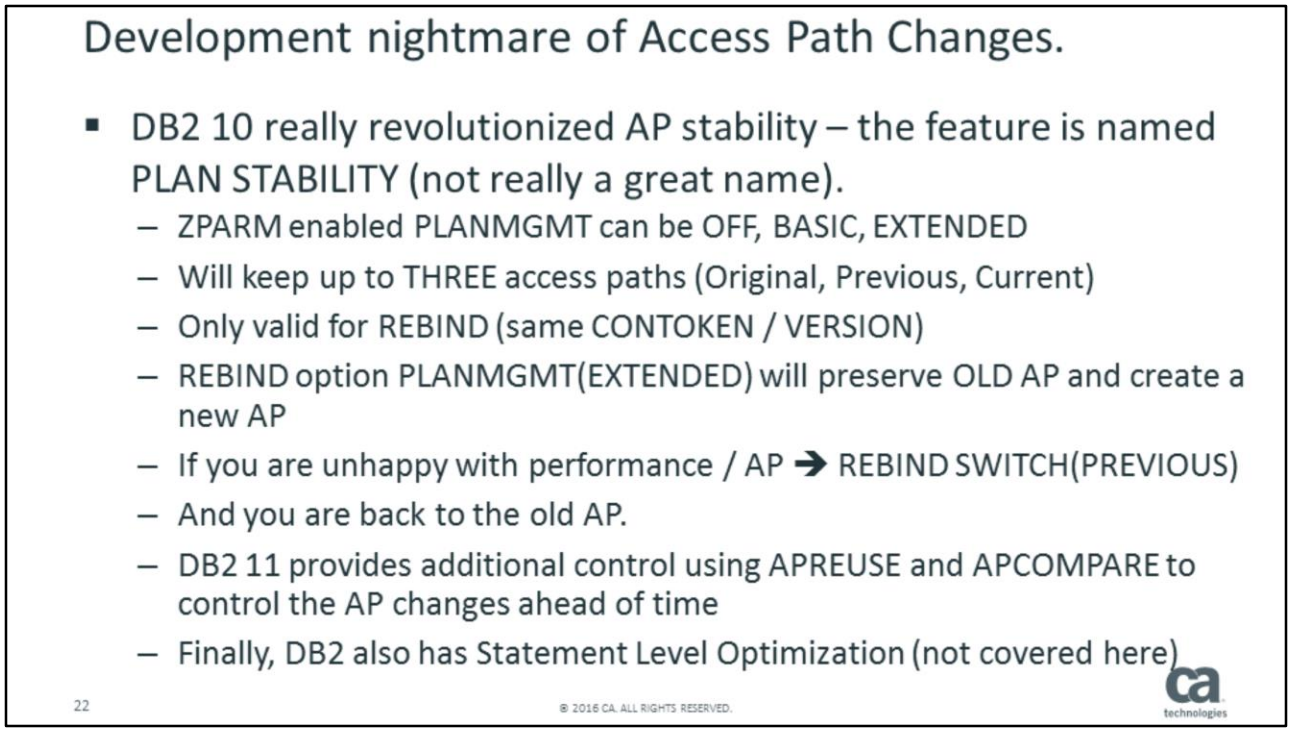
- When to RUNSTATS and when to REBIND ? ? ? ? ?
- Some do it after every REORG – but why ?
  - Did the column cardinality change since last reorg ?
  - Did the clusterratio change since last reorg ?
  - Maybe the number of rows changed – but everything else is pretty much the same so why bother – and why jeopardize stability ?
- I have a different religion:
  - Use RTS to determine when RUNSTATS might be needed
  - Keep history of RTS since history isn't supported out-of-the-box
- How to RUNSTATS isn't easy – but getting better with new autonomics

21

When to RUNSTATS and especially when to REBIND always turn into a heated discussion.

There are many ideas / philosophies . . . . . .

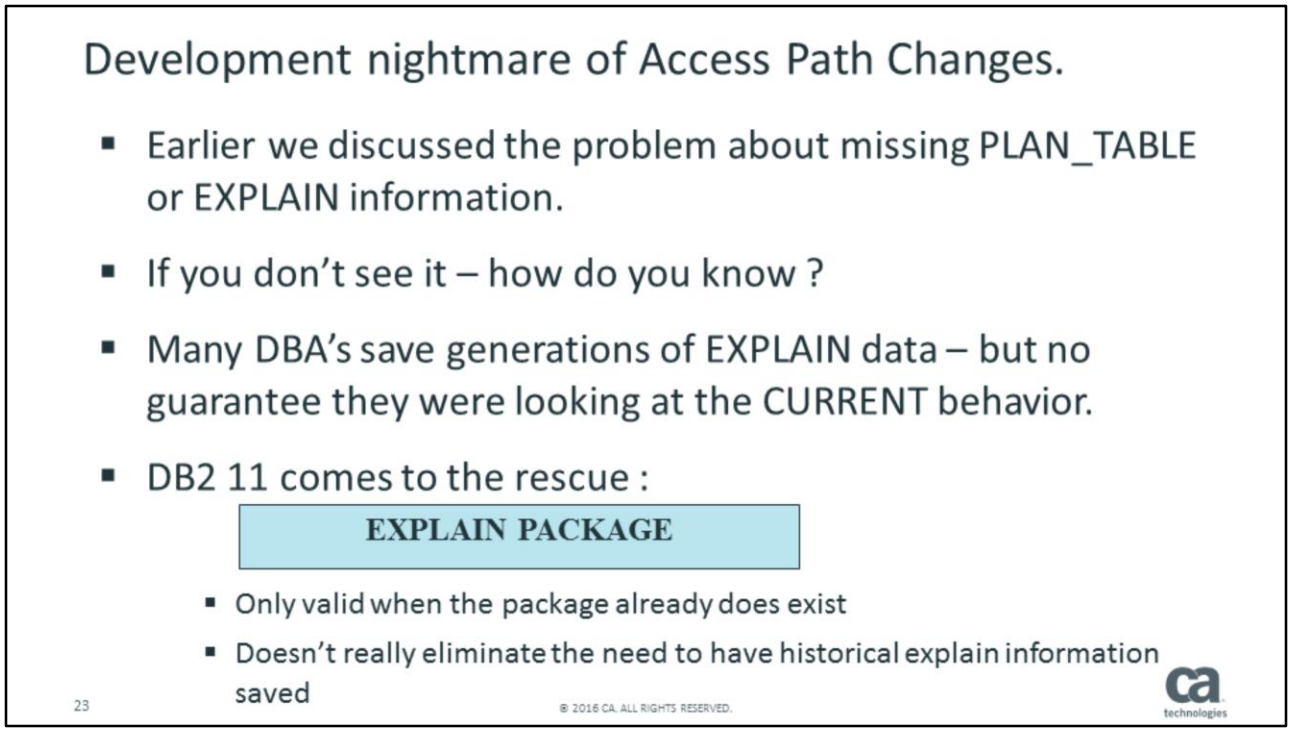
I prefer to only RUNSTATS and REBIND when I know for sure I can benefit.



This was how you can be prepared and mimic the production environment – but no matter how much you prepare, something will go wrong sooner or later – the access path will change for the worse – and this is where you really can prepare yourself and "get back to normal" as quickly as possible.

When you bind a PACKAGE, a new parameter named PLANMGMT can be used to mitigate the nightmare of the favorite access path to be gone for good. The value OFF is what we have had since packages were introduced.
The value EXTENDED will save the original access path, the current access path prior to the REBIND will become the PREVIOUS and a new access path is generated. If you are dissatisfied with the new access path, all you need to do is another rebind specifying SWITCH(PREVIOUS) and you're back to your previous access path (the great one) before the rebind which changed the access path for the worse.

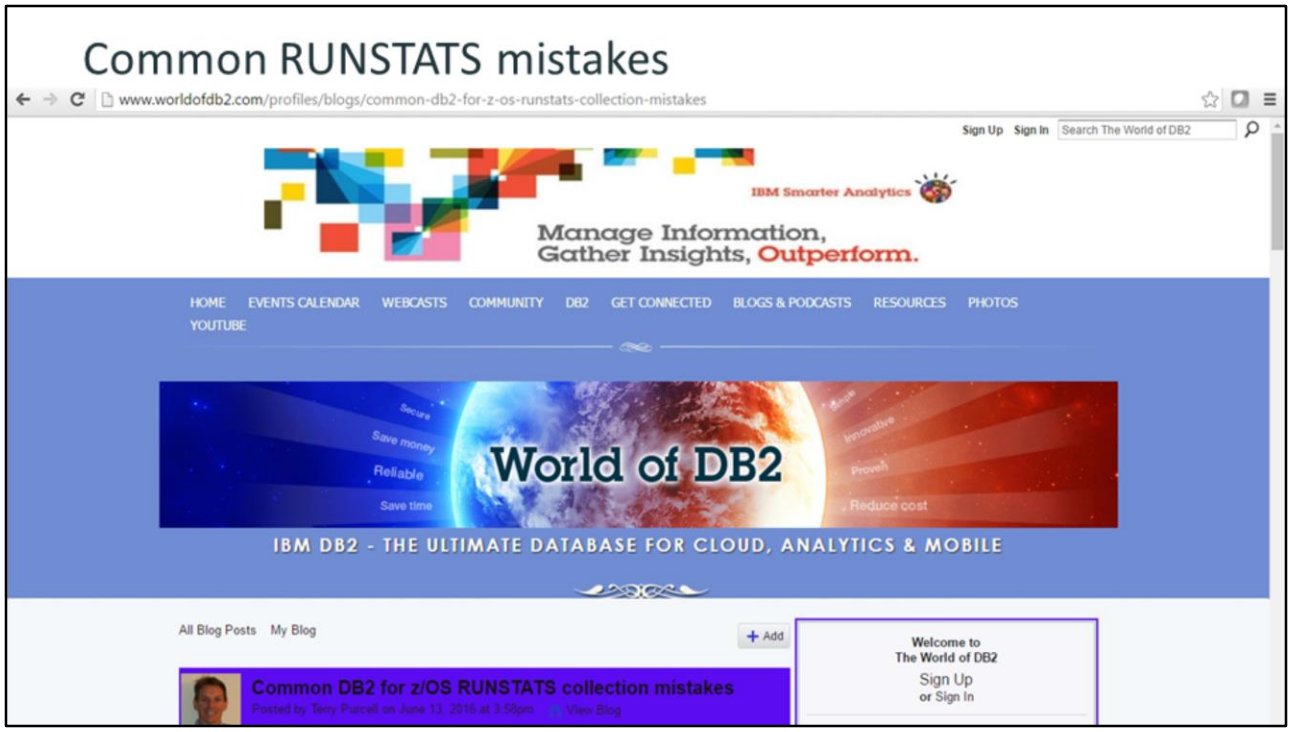
DB2 10 and DB2 11 has added more interesting options to the REBIND process allowing you to better control when access paths are changed : APREUSE and APCOMPARE are those options you might want to look into.
Both these options as well as the ability to also preserve access paths for dynamic statements are not discussed in this presentation – but several presentations are available out there on the crazy web.



Earlier we discussed the importance of having access path information available, and even though you always rebind or bind your packages with explain(YES), if that package later was invalidated and DB2 automatically did the rebind (or even someone else) – how can you tell the information stored in the explain-tables is the information actually used when the package executes ?

In order to make sure you know exactly what the access path is, DB2 11 comes to the rescue : it is now possible to retrieve the access path information from the directory. This will really eliminate a lot of the guesswork and uncertainty when debugging what DB2 is doing in terms of access path analysis. The best part is that the package doesn't have to have the EXPLAIN(YES) option enabled when it went through the BIND/REBINDS originally.

Common RUNSTATS mistakes



Part of content from terry's BLOG

- FREQVAL NUMCOLS 4 COUNT 15 doesn't provide NUMCOLS 3, 2 and 1 for multi-column frequency stats
  - How often are these really used : Only EQUAL-predicates can take advantage
  - Excessive usage will cause Optimizer to work overtime
- NUMCOLS X COUNT 0 does clean up old/stale stats
- RUNSTATS INDEX separated from TABLESPACE (due to reorg)

How can we satisfy Terry's Optimizer ?

How can we pick the "correct" options for RUNSTATS ?

Some new FEEDBACK tables come to the rescue !

SYSIBM.SYSSTATFEEDBACK

```
CREATE TABLE "SYSIBM".SYSSTATFEEDBACK
     (TBCREATOR      VARCHAR(128) FOR MIXED DATA    NOT NULL
     ,TBNAME         VARCHAR(128) FOR MIXED DATA    NOT NULL
     ,IXCREATOR      VARCHAR(128) FOR MIXED DATA    NOT NULL
     ,IXNAME         VARCHAR(128) FOR MIXED DATA    NOT NULL
     ,COLNAME        VARCHAR(128) FOR MIXED DATA    NOT NULL
     ,NUMCOLUMNS     SMALLINT                       NOT NULL
     ,COLGROUPCOLNO  VARCHAR(254) FOR BIT DATA      NOT NULL
     ,"TYPE"         CHARACTER(1) FOR MIXED DATA    NOT NULL
     ,DBNAME         VARCHAR(24)  FOR MIXED DATA    NOT NULL
     ,TSNAME         VARCHAR(24)  FOR MIXED DATA    NOT NULL
     ,REASON         CHARACTER(8) FOR MIXED DATA    NOT NULL
     ,BLOCK_RUNSTATS CHARACTER(1) FOR MIXED DATA    NOT NULL
     ,REMARKS        VARCHAR(762) FOR MIXED DATA    NOT NULL
     ,LASTDATE       DATE                           NOT NULL
      ) IN DSNDB06.SYSTSSFB
 APPEND NO
 NOT VOLATILE CARDINALITY
 DATA CAPTURE NONE
 AUDIT NONE
 CCSID UNICODE
 PARTITION BY SIZE EVERY 64G;
```

This new table has similar functionality and behavior as we know from the RTS tables.

27 27

One of the feedback tables is part of the DB2 catalog :

SYSIBM.SYSSTATFEEDBACK

You can kind of compare this table with the RTS feature introduced initially in DB2 V7 – information is gathered in memory and externalized in conjunction with the RTS statistics/metrics.

schema.DSN_STAT_FEEDBACK

```
CREATE TABLE STEEN.DSN_STAT_FEEDBACK
    ("QUERYNO" INTEGER                                  NOT NULL
    ,APPLNAME VARCHAR(24) FOR MIXED DATA                NOT NULL
    ,PROGNAME VARCHAR(128) FOR MIXED DATA               NOT NULL
    ,"COLLID" VARCHAR(128) FOR MIXED DATA               NOT NULL
    ,GROUP_MEMBER VARCHAR(24) FOR MIXED DATA            NOT NULL
    ,EXPLAIN_TIME TIMESTAMP (6) WITHOUT TIME ZONE       NOT NULL
    ,SECTNOI INTEGER                                    NOT NULL
                                                    WITH DEFAULT
    ,VERSION VARCHAR(122) FOR MIXED DATA                NOT NULL
                                                    WITH DEFAULT
    ,TBCREATOR VARCHAR(128) FOR MIXED DATA              NOT NULL
    ,TBNAME VARCHAR(128) FOR MIXED DATA                 NOT NULL
    ,IXCREATOR VARCHAR(128) FOR MIXED DATA              NOT NULL
    ,IXNAME VARCHAR(128) FOR MIXED DATA                 NOT NULL
    ,COLNAME VARCHAR(128) FOR MIXED DATA                NOT NULL
    ,NUMCOLUMNS SMALLINT                                NOT NULL
    ,COLGROUPCOLNO VARCHAR(254) FOR BIT DATA            NOT NULL
    ,"TYPE" CHARACTER(1) FOR MIXED DATA                 NOT NULL
    ,DBNAME VARCHAR(24) FOR MIXED DATA                  NOT NULL
    ,TSNAME VARCHAR(24) FOR MIXED DATA                  NOT NULL
    ,REASON CHARACTER(8) FOR MIXED DATA                 NOT NULL
    ,REMARKS VARCHAR(254) FOR MIXED DATA                NOT NULL
    )
```

Another addition to the myriad of EXPLAIN tables.

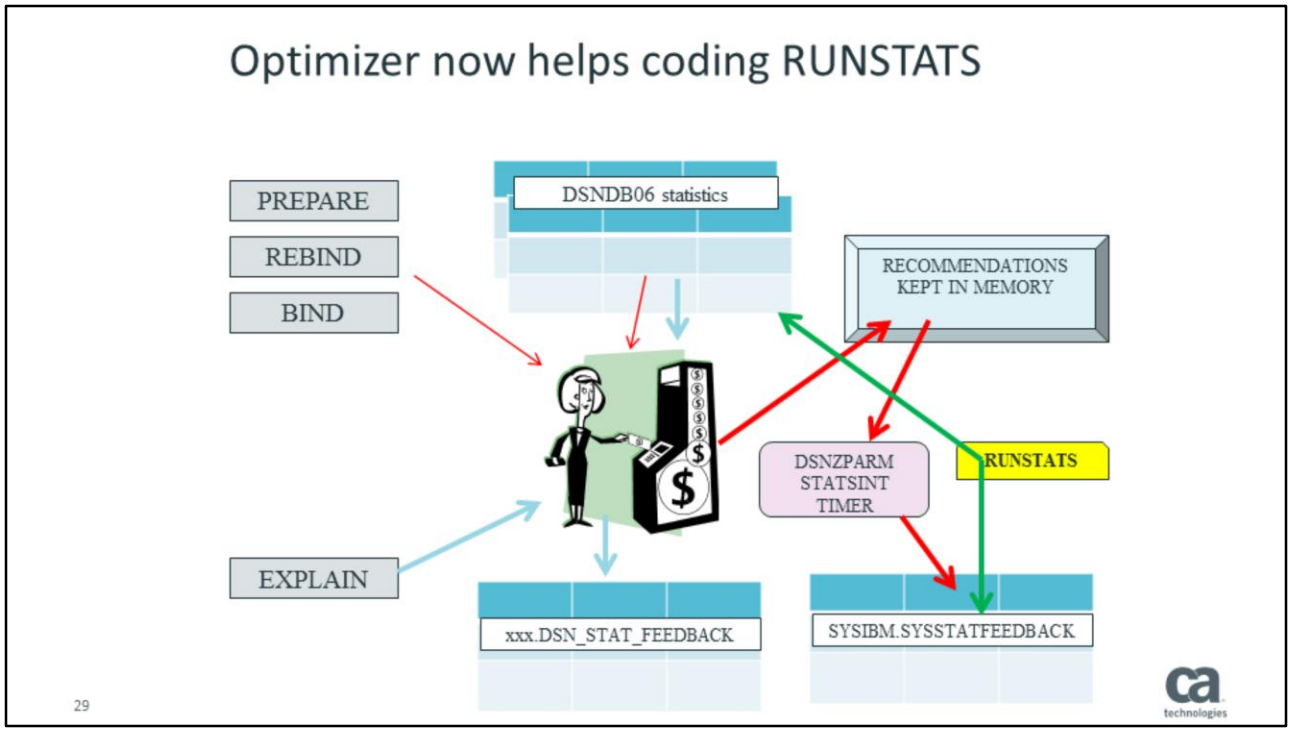
RUNSTATS does NOT cleanup this table as we will see happens with the other FEEDBACK table.

28 28

Besides SYSIBM.SYSSTATSTFEEDBACK – there's a similar table used in the same fashion : DSN_STAT_FEEDBACK

The two FEEDBACK tables are used in two different scenarios. SYSSTATFEEDBACK is populated via BIND, REBIND and PREPARE while the other one via EXPLAIN (requires the table is created with the same schema as the other explain table(s)).

Another difference is that SYSSTATFEEDBACK is cleaned up by RUNSTATS when the "requested" statistics is collected via RUNSTATS – DSN_STAT_FEEDBACK is NOT cleaned up.



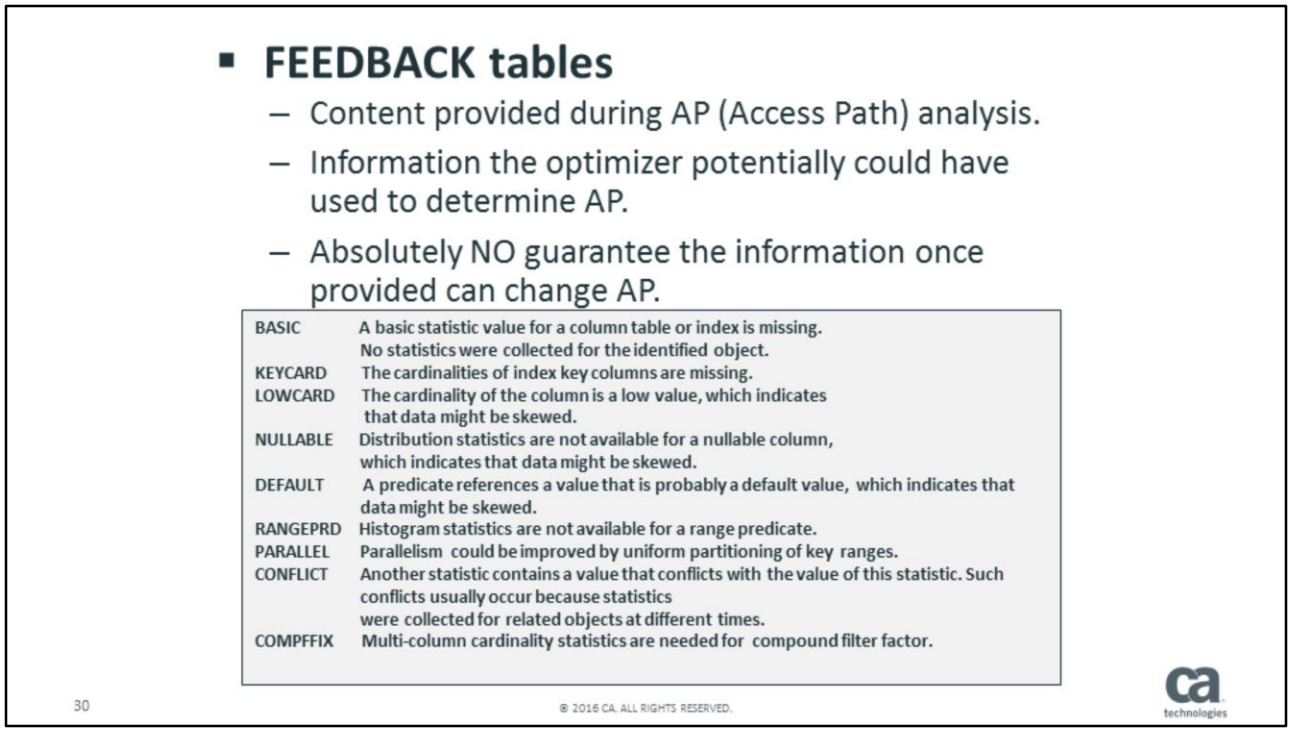
Let us have a close look at how these tables function in real life:

For Bind, Rebind and prepare – in memory – the statistics missing which potentially could have been used by the optimizer is recorded. Based on the same timer used by RTS – this information is externalized into SYSSTATFEEDBACK. Just like RTS, the information can also be externalized by using the ACCESS DB command.

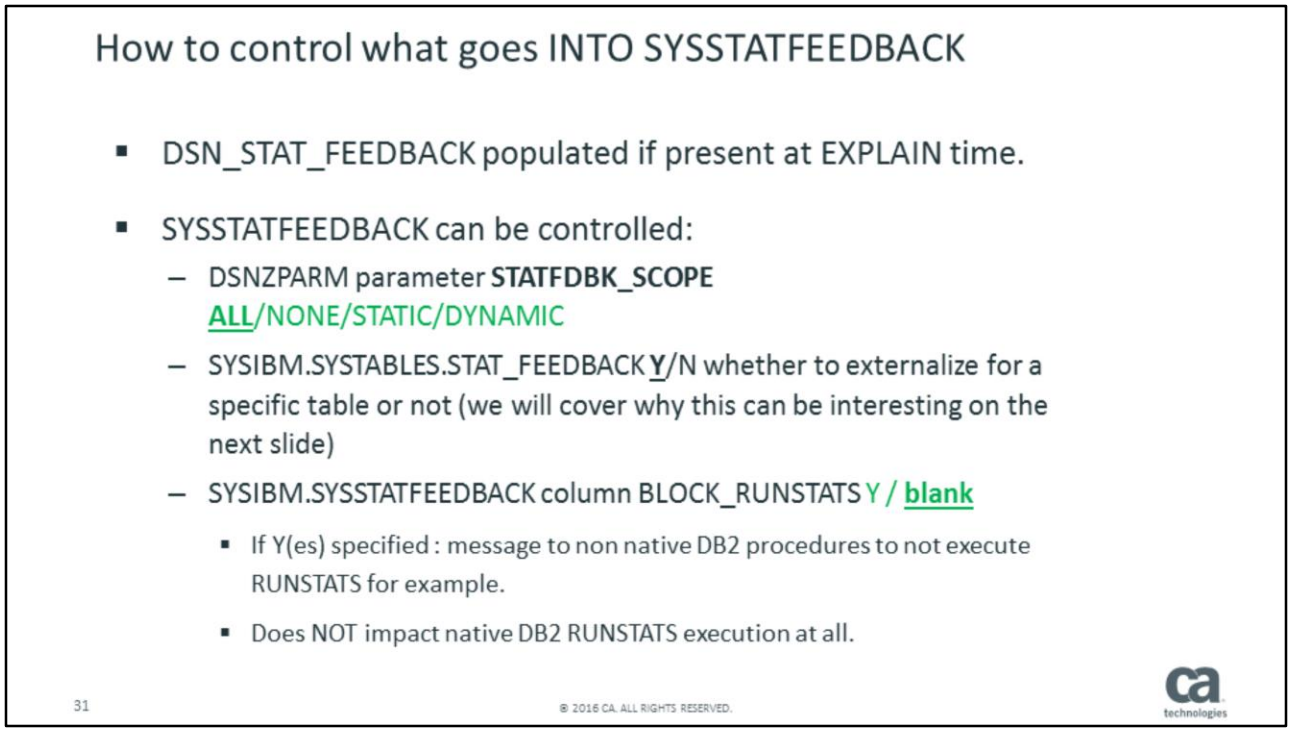
DSN_STAT_FEEDBACK is populated directly into the table during EXPLAIN.

Please remind yourself that the DB2 optimizer is COST based.

This being said – the tables are populated with information which POTENTIALLY could have been used by the optimizer if present, but there is no guarantee that the access path can change – nor that that the “missing” information is needed to collect.



SYSSTATFEEDBACK is a new table used by the Optimizer to report “missing stats”, insufficient stats” and other discrepancies.
The content isn’t ready and readable for the human eye, so it will take some learning unless you want to depend on tools interpreting the content.
At least the column=REASON point you in the right direction.



We already mentioned how the two FEEDBACK tables are populated.

Unlike the FEEDBACK table associated with EXPLAIN (where you really have no control), there are a few parameters you need to be aware of in terms of dealing with SYSIBM.SYSSTATFEEDBACK:

1) STATFDBK_SCOPE is a new DSNZPARM parameter which controls whether you want feedback for STATIC only, DYNAMIC only, BOTH static and dynamic or BOTH static and dynamic.
2) SYSTABLES has a new column where you can describe whether to have DB2 collect the “missing information” or not.
3) Finally – the last point of control is a column in SYSSTATFEEDBACK named BLOCK_RUNSTATS which is NOT controlled by DB2 at all. It is meant to be used by “tools” – BLOCK_RUNSTATS.

It is time to have a closer look at how to execute RUNSTATS based on how the optimizer provides feedback !

SYSIBM.SYSTABLES.STAT_FEEDBACK Y/N whether to externalize for this table or not – here is one reason

(SYSLGRNX can not be RUNSTATS'ed)

IXNAME	COLNAME	NUMCOLUMNS	COLGROUPCOLNO	TYPE	REASON	COLGROUPCOLNO (HEX)
DSNLLX01		3	.....	C	KEYCARD	00 01 00 02 00 07
DSNLLX01		0		I	BASIC	
DSNLLX02		0		I	BASIC	
		0		T	BASIC	
DSNLLX01		2	....	C	KEYCARD	00 01 00 02
	LGRPART	1		C	BASIC	
	LGRPSID	1		C	BASIC	
	LGRDBID	1		C	BASIC	
DSNLLX01		4	.....	C	KEYCARD	00 01 00 02 07 00 0A
	LGRSRBA	1		C	BASIC	
	LGRPART	1		F	DEFAULT	

```
DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = RASST02.RASST02B
DSNUGUTC - RUNSTATS TABLESPACE(DSNDB01.SYSLGRNX) INDEX(ALL) SHRLEVEL CHANGE
DSNUGMAP - UTILITY NOT ALLOWED AGAINST SYSTEM DATABASE
DSNUSVAL - ERROR OCCURRED IN ACCESSING TABLESPACE DSNDB01.SYSLGRNX
```

- Updating column BLOCK_RUNSTATS to "Y" doesn't prohibit RUNSTATS from executing and collecting/updating the statistics specified.
- SYSSTATFEEDBACK will only have externalized information for tables not marked with "N" in SYSIBM.SYSTABLES.STAT_FEEDBACK

32 32

Once you have enabled DB2 to collect "missing statistics" you can start to make sure you collect the "appropriate" RUNSTATS information based on SYSSTATFEEDBACK. One issue to have in mind is – not every table being reported on is eligible tor RUNSTATS – this example is the directory table SYSLGRNX, but as you can see, it is NOT possible to execute RUNSTATS against directory objects, so maybe you should consider excluding DB01 objects by updating SYSTABLES and mark these as NON-collectible ???

Another issue to consider is, even though you mark SYSSTATFEEDBACK.block_runstats as "Y", this doesn't prohibit anyone from collecting RUNSTATS statistics.

Who is cleaning up the FEEDBACK tables ?

- For the extension to EXPLAIN – you have a new task !!
- For the SYSIBM.SYSSTATFEEDBACK table, the cleanup process can happen in a few ways:
  - When RUNSTATS is executed updating the catalog with the information registered in SYSSTATFEEDBACK – auto cleanup.
  - You can SQL DELETE if you want to
  - INSERT / UPDATE isn't possible

```
******************** TOP OF DATA ******************
DSNT408I SQLCODE = -607, ERROR:  OPERATION OR OPTION
INSERT IS NOT DEFINED FOR THIS OBJECT
*************** BOTTOM OF DATA *********************
```

  - ***WISH*** : I would like to see LAST_USED be changed from DATE to TIMESTMP (don't have to use ACCESS DB to externalize).
  - When an object is dropped – cleanup is done too.

33

If you decide to "listen" to the optimizer and start to modify your favorite runstats, you do have a quite a task ahead of you – depending on far you want to go.

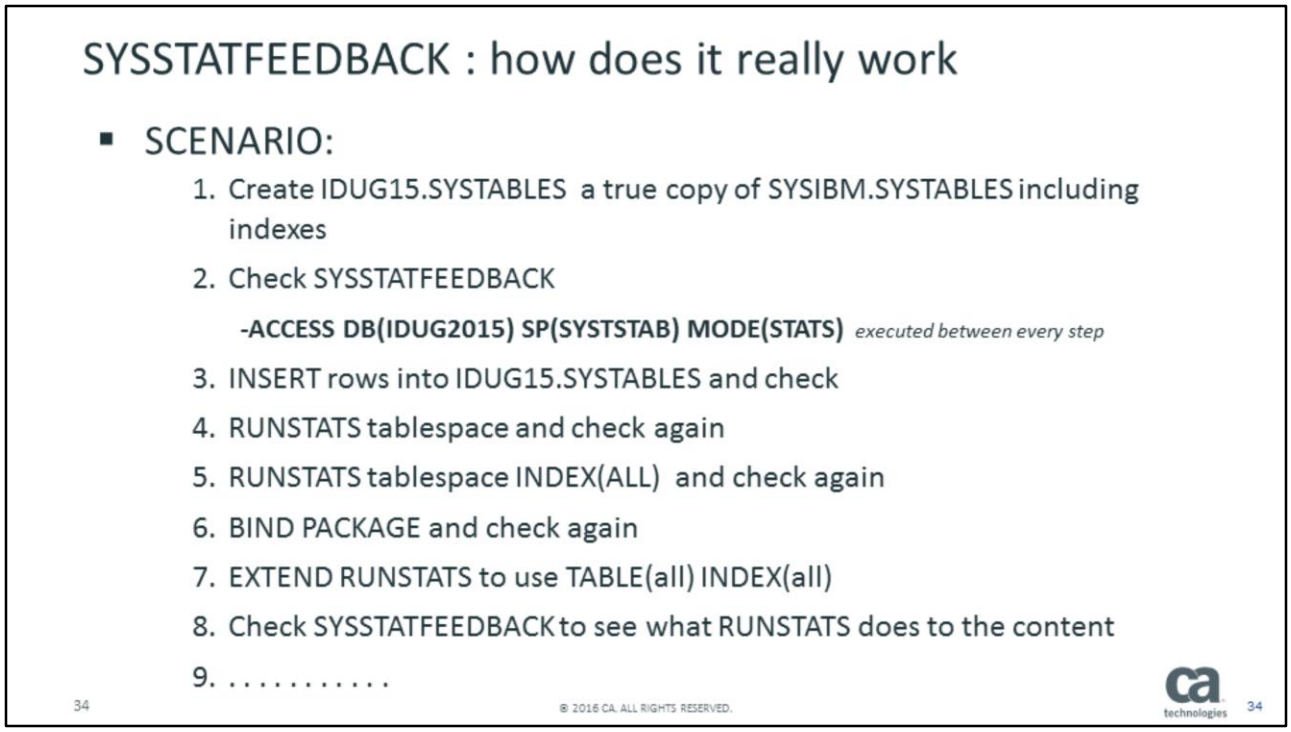
One issue you need to consider is how DB2 (runstats) does the cleanup of SYSSTATFEEDBACK. When you follow the feedback provided and execute the corresponding runstats, the information you used from SYSSTATFEEDBACK leading to the corresponding runstats will be removed by runstats.

You do have the opportunity to SQL DELETE from SYSTSATFEEDBACK yourself if so desired, but the information will probably be re-populated ☺

From my perspective, I would like the column LAST_USED in SYSSTATFEEDBACK to be extended from DATE to TIMESTAMP – and I would like the date to reflect when a RUNSTATS USE last time used the information.

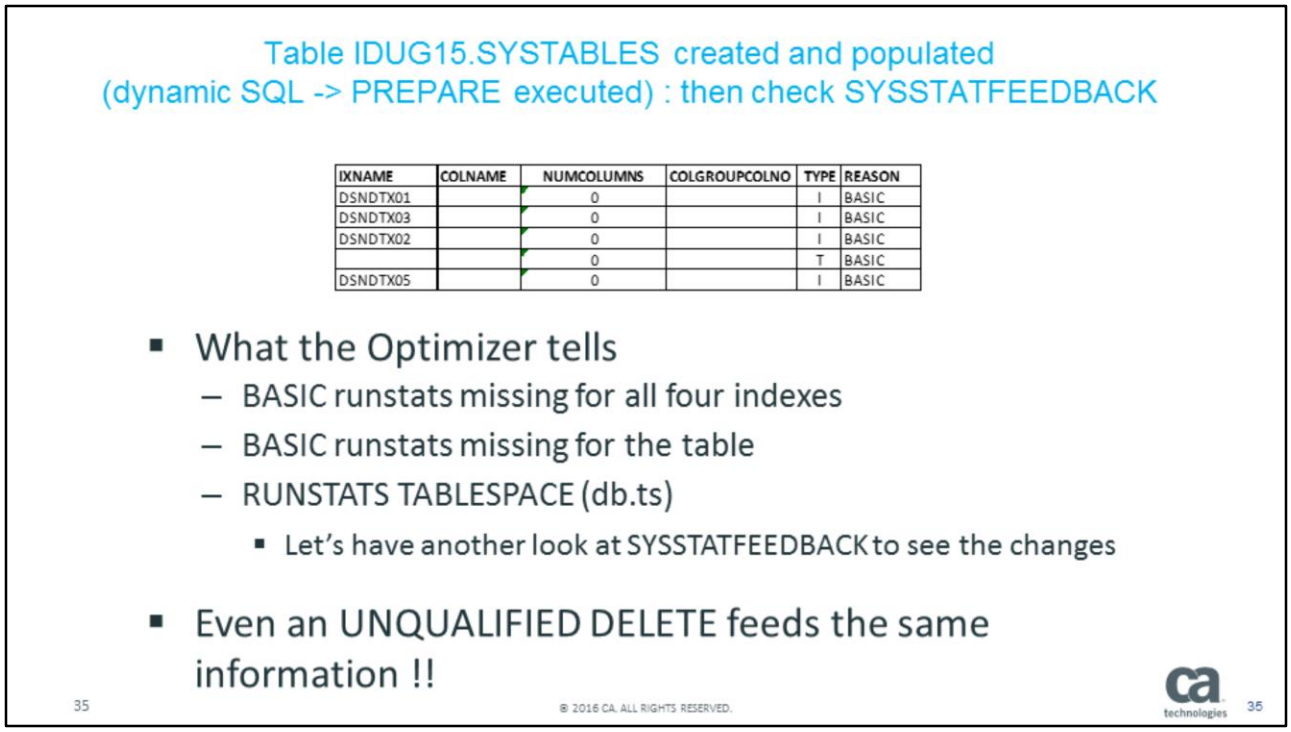
What is neat is – when an object is dropped, SYSTATSFEEDBACk is cleaned up too – just like RTS.

Despote all these nice features – there's always a Christmas wishlist: I would like to see when RUNSTATS executed really used the profile. We have the LAST_USED column so maybe a wish for DB2 Cypress ???



Let's walk through a live scenario to see how the information flows and the changes based on the action you take.

1) We create a copy of SYSTABLES including the associated indexes.
2) Check SYSSTATFEEDBACK to see if anything got inserted. In order NOT to wait for the timer to externalize, and to make sure we look at the latest information, ACCESS DB is executed prior to every view of SYSSTATFEEDBACK.
3) Insert rows into the table and check what happened.
4) RUNSTATS tablespace (basic runstats) and check content
5) RUNSTATS tablespace with INDEX(ALL) and check
6) BIND PACKAGE and check
7) RUNSTATS using TABLE syntax and check
8) One final check into SYSSTATFEEDBACK



SYSSTATFEEDBACK did not get populated by creating a table.

Once the INSERT statement was executed, information was gathered: due to the INSERT statement being PREPARE'd (one of the three tasks which will populate the table).

We got one row TYPE=T telling us RUNSTATS is missing for the tablespace.
We got one row for each of the indexes telling us RUNSTATS INDEX(….) is missing.

Let's RUNSTATS the TABLESPACE as the first task and see the content change ?.

RUNSTATS TABLESPACE and check SYSSTATFEEDBACK

IXNAME	COLNAME	NUMCOLUMNS	COLGROUPCOLNO	TYPE	REASON
DSNDTX01		0		I	BASIC
DSNDTX03		0		I	BASIC
DSNDTX02		0		I	BASIC
DSNDTX05		0		I	BASIC

- What happened ?
  - The TYPE 'T' basic runstats has been removed.
  - Runstats really does clean up entries when the request has been completed.
- What the Optimizer tells
  - BASIC runstats missing for all four indexes.

36

As expected, doing RUNSTATS on the TABLESPACE caused the TYPE=T row to be deleted by runstats.
Let's do the RUNSTATS of the indexes.

RUNSTATS TABLESPACE INDEX(all) and check SYSSTATFEEDBACK

IXNAME	COLNAME	NUMCOLUMNS	COLGROUPCOLNO	TYPE	REASON

- What happened ?
  - The TYPE 'I' basic runstats has been removed.
  - Runstats really does clean up entries when the request has been honored and completed.

RUNSTATS TABLESPACE db.td INDEX(all) executed, and all the remaining rows for this object was cleaned up by RUNSTATS – very nice !!

BIND PACKAGE and check SYSSTATFEEDBACK

IXNAME	COLNAME	NUMCOLUMNS	COLGROUPCOLNO	TYPE	REASON
	NAME	1		C	BASIC
	TYPE	1		C	BASIC

- What happened ?
  - Column=NAME is not the first column in any index but predicates are referencing this column
  - Column=TYPE is not included in any index at all but used in WHERE predicates
- Once RUNSTATS tablespace TABLE(all) INDEX(all) executed the SYSSTATFEEDBACK is empty for this object.

38

Now it is time to BIND a package with statements referencing this table IDUG15.SYSTABLES.

Two rows were inserted into SYSSTATFEEDBACK – both being COLUMNS from the table.
The interesting thing is – column=NAME is NOT the first column in any index, so there must be predicates in the package referencing this column.
Column=TYPE is not in ANY index at all – again this column is used in various WHERE predicates.
TYPE=C means that the Optimizer would like to see CARDINALITY for these columns – maybe a different access path can be chosen

Different and more complex scenario – let's start with the content of SYSSTATFEEDBACK

IXNAME	COLNAME	NUMCOLUMNS	COLGROUPCOLNO	TYPE	REASON	COLGROUPCOLNO (HEX)	LASTDATE
INSASI		6	.....	C	KEYCARD	00 01 00 03 00 04 00 07 00 08 00 0C	2014-12-12
INSASI		5	.....	C	KEYCARD	00 01 00 03 00 04 00 07 00 08	2014-12-12
INSASI		2	....	C	KEYCARD	00 01 00 03	2014-12-12
		0		T	BASIC		2014-12-12
INSASI		0		I	BASIC		2014-12-12
	CONNECTION_ID	1		C	BASIC		2014-10-23
INSASI		3	.....	C	KEYCARD	00 01 00 03 00 04	2014-12-12
INSASI		4	.....	C	KEYCARD	00 01 00 03 00 04 00 07	2014-12-12
INSASI		7	.....	C	KEYCARD	00 01 00 03 00 04 00 07 00 08 00 0C 00 0D	2014-12-12
	SYSTEM_ID	7	.....	C	COMPFFIX	00 03 00 04 00 07 00 08 00 0C 00 0D 00 0E	2014-09-10
	YEAR	1		C	BASIC		2014-10-23
	SYSTEM_ID	1		C	BASIC		2014-10-23
	END_DATE_TIME	1		C	BASIC		2014-12-12
	BEGIN_DATE_TIM	1		C	BASIC		2014-12-12
	DAY	1		C	BASIC		2014-10-23
	MONTH	1		C	BASIC		2014-10-23
	PLAN_NAME	1		C	BASIC		2014-10-23
	REQUESTOR_LOCA	1		C	BASIC		2014-09-10
	SUBSYSTEM	1		C	BASIC		2014-10-23
	PLAN_NAME	1		F	DEFAULT		2014-09-26

- COLGROUPCOLNO (HEX) is not a column – but the content of COLGROUPCOLNO unpacked in a readable format

39

39

The previous scenario was pretty basic, so let us have a closer look at what kind of challenges you might face once you start to exploit this feature.

This table has a LOT of content in SYSSTATFEEDBACK – there really is a lot to digest and think about.

Only one BASIC TABLESPACE and BASIC INDEX RUNSTATS needed – the rest of the information is related to CARDINALITY, so time put the thinking hat on !!
We also have a FREQUENCY information : TYPE=F

Please note the different LASTDATE dates – might give you a clue when / why the information was provided and by who??

Another great piece of information is the NUMCOLUMNS (number of columns which need COLGROUP COLNO) – for readability I have expanded this column into HEX so it can be correlated to the TABLE-COLUMN-NUMBERS.

Different and more complex scenario – let's start with the content of SYSSTATFEEDBACK

- So what can we do with these hexadecimal numbers ?
  - Once you get the grip you are good to go.
  - Let's pick the first one from the grid !!

REASON	COLGROUPCOLNO (HEX)
KEYCARD	00 01 00 03 00 04 00 07 00 08 00 0C

  - We have to look at the TABLE-COLUMN order to digest the numbers from COLGROUPCOLNO.
  - Remember to convert the SYSSTATFEEDBACK column number to decimal when looking at SYSCOLUMNS.

COLNO	COLUMN NAME
1	BEGIN_DATE_TIME
2	END_DATE_TIME
3	SYSTEM_ID
4	SUBSYSTEM
5	RELEASE
6	RECORD_COUNT
7	PLAN_NAME
8	CONNECTION_ID
9	LOCAL_LOCATION
10	REQUESTOR_LOCATION
11	NETWORK_ID
12	YEAR
13	MONTH
14	DAY
15	DAY_OF_WEEK
16	WEEK_NUM
17	TRANS_COUNT
18	CONNTYPE

40

40

If we look at the first row where TYPE=C and NUMCOLUMNS=6, we need to unpack the COLGROUPCOLNO column in order to identify the corresponding column names from SYSCOLUMNS.

Then this information can be transformed into a COLGROUP parameter.

Different and more complex scenario – let's start with the content of SYSSTATFEEDBACK

- Considering the number of COLGROUPCOLNO entities in SYSSTATFEEDBACK, let's see how the optimizer likes the most basic RUNSTATS

```
RUNSTATS TABLESPACE(INSAS.INSIGHT) TABLE(ALL) INDEX(ALL)
```

IXNAME	COLNAME	NUMCOLUMNS	COLGROUPCOLNO	TYPE	REASON	COLGROUPNO (HEX)
	SYSTEM_ID	7	.....	C	COMPFFIX	00 03 00 04 00 07 00 08 00 0C 00 0D 00 0E
	PLAN_NAME	1		F	DEFAULT	

```
RUNSTATS TABLESPACE(INSIGHT.INSAS) TABLE(INSIGHT.APPLICATION_DAILY)
COLGROUP(SYSTEM_ID,SUBSYSTEM,PLAN_NAME,CONNECTION_ID,YEAR,MONTH,DAY)
```

IXNAME	COLNAME	NUMCOLUMNS	COLGROUPCOLNO	TYPE	REASON	COLGROUPNO (HEX)
	PLAN_NAME	1		F	DEFAULT	

```
RUNSTATS TABLESPACE (INSIGHT.INSAS) TABLE(INSIGHT.APPLICATION_DAILY)
COLGROUP(PLAN_NAME) FREQVAL COUNT 20 SHRLEVEL CHANGE
```

41 41

Just by executing the most basic RUNSTATS using TABLE(ALL) and INDEX(ALL) eliminates most of the content from SYSSTATFEEDBACK – and there's only two entries left.

In order to eliminate TYPE=C, we need the COLGROUP parameter but have to "unpack" the 7 columns into HEX and then find the corresponding column numbers from SYSCOLUMNS for that specific table.

Only one left now – TYPE=F so we need FREQVAL. What number to pick ???? I picked 20 and the default is 10, so please use your best judgment.

All of these parameters can now be added to a RUNSTATS PROFILE so you don't need to memorize.

How to avoid populating SYSSTATFEEDBACK

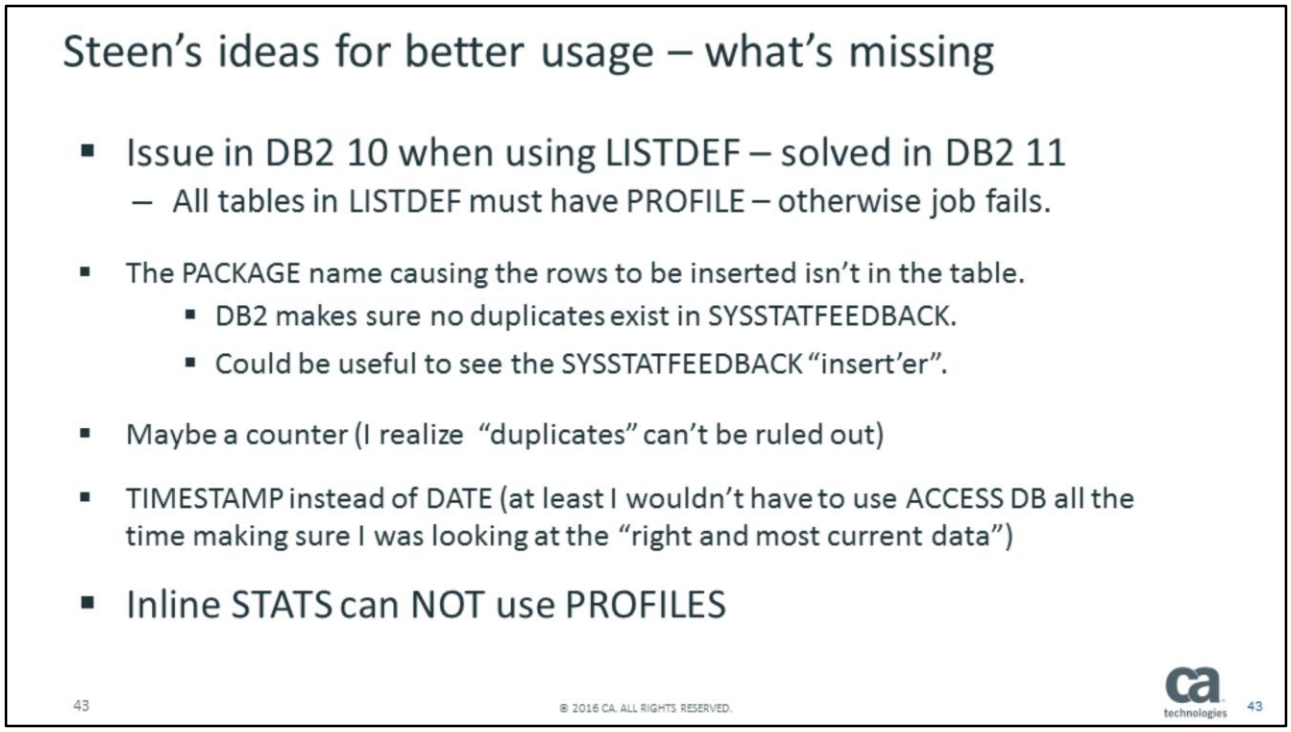
```
For Table => SYSIBM.SYSTABLES                >   Row number=> 3 OF 3
Edit Mode => F                                   Max Char  => 256
Option    =>                                     Status    =>
SSID: D11A ------------------------------------------------------------ RASST02
##.COLUMN NAME        NULL  DATA FOR ROW # 3
A1.NAME                     SYSLGRNX
A2.CREATOR                  SYSIBM
A3.DBNAME                   DSNDB01
A4.STATS_FEEDBACK           N
******************************* BOTTOM OF DATA ********************************
```

- There might be some tables where you don't want SYSSTATFEEDBACK to be populated for various reasons
  - Some tables can not have RUNSTATS executed
  - You might want to ensure RUNSTATS is only executed based on PROFILES
- Update SYSTABLES.STATS_FEEDBACK to be 'N'
  - The Optimizer will not populate SYSSTADFEEDBACK

42

42

We haven't really covered the an associated column (new in DB2 11) in SYSIBM.SYSTABLES.
The column STATS_FEEDBACK will block the optimizer from feeding information into SYSSTATFEEDBACK.



As always Steen has a Christmas wishlist for Santa :

1) One issue which did exist in DB2 10 has been resolved in DB2 11 – if using LISTDEF for your RUNSTATS and just one table doesn't have a profile, the entire process will fail.
2) The PACKAGE name feeding information isn't recorded in SYSSTATFEEDBACK. I understand there probably will be dozens or hundreds of packages, but somehow this information could be useful – perhaps an internal identifier along with one additional table so you can correlate the information.
3) Since the optimizer doesn't feed DUPLICATE information (could be useful if package name was introduced), a counter for the information in SYSSTATFEEDBACK.
4) It would have been nice for this research to have the DATE column expanded to TIMESTAMP. Then I would not have been forced to use ACCESS DB between every case to get an instate view.
5) Finally – INLINE STATS for reorg and load do NOT support profiles . . . . . Maybe NEVER do inline stats anymore ?

Wrap Up and conclusion

- SYSSTATFEEDBACK is a GREAT "tool"
- SYSTABLES_PROFILES is another great autonomic feature
- . . . . . But everything isn't green(er) – despite better !
  - Consider SCHEMA CHANGES where columns are dropped / renamed
  - Let's look at a couple of challenges

```
ALTER TABLE IDUG15.SYSTABLES RENAME COLUMN TYPE TO TYPE4 ;
ALTER TABLE IDUG15.SYSTABLES DROP COLUMN LABEL RESTRICT ;
```

  - **How does these features integrate with SYSSTATFEEDBACK and RUNSTATS_PROFILES ??**

44

DB2 indeed does a great job recording "missing" runstats in SYSSTATFEEDBACK, and once you have identified the needed runstats, instead of memorizing these, you can describe the syntax in the SYSTABLES_PROFILES by using the SET / UPDATE PROFILE syntax in RUNSTATS.

Let's see if the grass really is greener . . . How does DB2 handle schema changes like DROP COLUMN and RENAME COLUMN ?

The DB2 world is getting better – new challenges ?

- DROP column – clean up not instantaneous
  - Reorg execution cleans up SYSSTATFEEDBACK
- RENAME column also renames column in SYSSTATFEEDBACK instantaneous

```
DSNUGUTC -  RUNSTATS TABLESPACE(IDUG2015.SYSTSTAB) SHRLEVEL REFERENCE TABLE . . . .
         USE PROFILE PROFILE
DSNUGPRB - PARSING STATS PROFILE FOR TABLE SYSTABLES
DSNUGPRB - PARSING STATS PROFILE FOR TABLE SYSTABLES COMPLETED
DSNUSVAL - COLUMN LABEL NOT FOUND FOR TABLE IDUG15.SYSTABLES
DSNUSVAL - COLUMN TYPE NOT FOUND FOR TABLE IDUG15.SYSTABLES
DSNUGBAC - UTILITY EXECUTION TERMINATED, HIGHEST RETURN CODE=8
 ********************************* End of Data ********************************
```


DB2 cleans up SYSSTATFEEDBACK very nicely – the RUNSTATS profiles is a different matter – you need to make sure to keep the COLUMN and COLGROUP parameters updated if used in the RUNSTATS profile, and this is a manual process.
Another place where you need to pay attention is STALE statistics. DB2 doesn't provide a mechanism to inform you about this situation.
maintaining schema changes in SYSSTATFEEDBACK.

DB2 12 web cast late March looks very promising for “closing the gaps”.

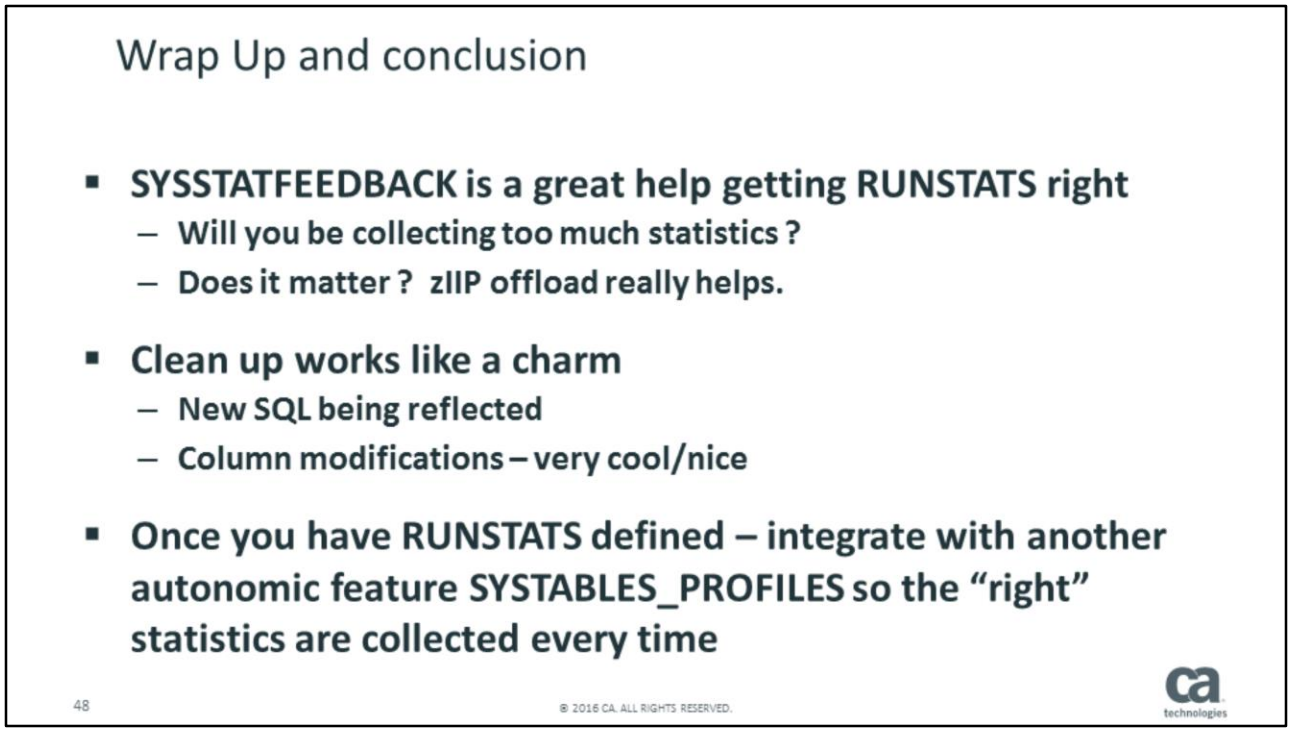
- Inline Stats using profiles.
- Automatic creation of profiles.
- Schema management maintenance of profiles
- . . . . . .

46

The DB2 world is getting better – new challenges ? A response from Terry on DB2 LISTSERV

- In DB2 11 - the optimizer will externalize missing statistics to the catalog (SYSIBM.SYSSTATFEEDBACK after NFM) or during explain (provided table DSN_STAT_FEEDBACK exists - available in CM). This identifies when statistics were missing that could have been used. There is no other method - other than with tooling, to know if existing statistics are actually used by the optimizer.
- If you have an index on C1, C2, C3- and that is the index you want to drop. Then the recommendation is to collect COLGROUP(C1,C2,C3). But collect them AFTER dropping the index, and not BEFORE. If you collect them before, then dropping the index will remove those statistics from SYSCOLDIST. SYSCOLDIST does not know whether the statistics were collected from an index or via COLGROUP - and will remove upon DROP that could have been collected via that index.

47



Integrating the granular defined RUNSTATS derived from SYSSTATFEEDBACK with SYSTABLES_PROFILES can for sure help a lot getting the appropriate statistics generated when needed.

DB2 does a great job maintaining currency for SYSSTATFEEDBACK

Wrap Up

- Hopefully you will benefit from the content and get less nightmares or headaches about RUNSTATS in the future !!

Thank You - QUESTIONS ?

Agenda

1. TITLE OF SECTION ONE
2. TITLE OF SECTION TWO
3. TITLE OF SECTION THREE
4. TITLE OF SECTION FOUR
5. TITLE OF SECTION FIVE
6. TITLE OF SECTION SIX

Title – Title Case, Calibri 28 pt
2 Lines Max

- Bullet 1, Calibri regular 24 pt
  - Sub-bullet, Calibri regular 20 pt
    - Sub-sub-Bullet, Calibri regular 18 pt
      - Sub-sub-sub Bullet, Calibri regular 16 pt
        - Sub-sub-sub-sub Bullet, Calibri regular 14 pt


technologies