The Db2 12 catalog – what happened since Db2 11

Steen Rasmussen
Principal Engineering Services Architect,

CA technologies
It is Db2 and not DB2
Agenda

- Dropped catalog objects.
- New catalog objects.
- Old Db2 features added to the catalog.
- Modified catalog tablespaces, catalog tables, columns, indexes, relationships.
- New Db2 12 features reflected in the catalog.

This presentation has 5 main topics:

- Objects removed from the Db2 catalog.
- New tablespaces and tables as well as indexes.
- Part of the catalog can adopt Db2 features which were made available in Db2 10.
- A few catalog objects have been modified too.
- Finally, a glance into existing objects with modified information reflecting the new Db2 12 features.
DISCLAIMER

• Expressions are purely my own – not CA technologies.

• This presentation is based on using a real life Db2 12 system FUNCTION LEVEL(V12R1M500) with maintenance as of November 2016.

• Discrepancies do exist comparing with Db2 SQL Reference Guide APPENDIX A Catalog Tables – and this presentation (as of January 2017).
ABSTRACT

- There are many ways to learn about a new Db2 version, but using the catalog is probably the fastest and most enlightening. Once you look at the revision track in the Db2 Catalog and Directory section of the SQL Reference Guide, study the column changes, additions as well as new catalog tables, you have a very good overview of everything worth to know about that specific Db2 version.

The catalog always reflect the features available in a Db2 release, so browsing through the IBM Db2 SQL Reference Guide to look at the new tables and also existing columns with new content as well as existing tables with new columns can provide a quick view of all the changes in that specific Db2 release.
Why do you have to understand the Db2 Catalog changes?

- You can read the What’s New Guide
- The IDUG website has an excellent White Paper
- You have some kind of tooling to query the catalog!
- If you don’t know what’s IN there – how do you know how to benefit?
- If using your own SQL or homegrown REXX’s or the like:
  - You might have to consider adding additional join predicates
  - You probably need to change FILTERING (WHERE clauses)
  - You might need to code new queries

- And probably many more reasons . . . . . .

You might question why it can be an advantage to understand the catalog changes between releases.

1) After all you can study the release guide
2) IDUG usually provides some kind of white paper highlighting all the new goodies
3) Most Db2 sites have a catalog query tool from IBM or one of the vendors – and some might be using SPUFI or have some homegrown solution in place.

Bottom line, if you don’t understand all the details, you might be missing out some great opportunities.

If you are using SPUFI or a homegrown solution, you probably need to change some of the queries or add new ones, in which case its even more important that you have a good understanding of the changes.
Dropped Catalog Objects
Nothing removed from the Catalog

- No additional PBG conversions like Db2 10 and 11
- Still some old multi-table tablespaces and non-PBG

<table>
<thead>
<tr>
<th>Catalog Objects</th>
<th>Number of Tables</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSALTER with SYSOBDS</td>
<td>3</td>
</tr>
<tr>
<td>SYSCONTX with 3 tables</td>
<td></td>
</tr>
<tr>
<td>SYSDDF with 8 tables</td>
<td></td>
</tr>
<tr>
<td>SYSEBCDC with 2 tables</td>
<td></td>
</tr>
<tr>
<td>SYSGPAUT with SYSRESAUTH</td>
<td></td>
</tr>
<tr>
<td>SYSGRRTNS with 2 tables</td>
<td></td>
</tr>
<tr>
<td>SYSHIST with 11 tables</td>
<td></td>
</tr>
<tr>
<td>SYSJAUXA with SYSJARDATA</td>
<td></td>
</tr>
<tr>
<td>SYSJAUXB with</td>
<td></td>
</tr>
<tr>
<td>SYSJARCLASS_SOURCE</td>
<td></td>
</tr>
<tr>
<td>SYSJAVA with 4 tables</td>
<td></td>
</tr>
<tr>
<td>SYSPLUXA with SYSROUTINESTEXT</td>
<td></td>
</tr>
<tr>
<td>SYSPLUXB with SYSROUTINES_TREE</td>
<td></td>
</tr>
<tr>
<td>SYSROLES with 2 tables</td>
<td></td>
</tr>
<tr>
<td>SYSSEQ with SYSEQUENCES</td>
<td></td>
</tr>
<tr>
<td>SYSSEQ2 with 2 tables</td>
<td></td>
</tr>
<tr>
<td>SYSSTATS with 9 tables</td>
<td></td>
</tr>
<tr>
<td>SYSTARG with SYSKEYTARGETS</td>
<td></td>
</tr>
<tr>
<td>SYSXML with 2 tables</td>
<td></td>
</tr>
</tbody>
</table>

Nothing has been removed from DSNDB06 unlike what was the case in Db2 10 and Db2 11 where some multi-table tablespaces were moved into PBG tablespaces.

Db2 12 has NOT converted any of the remaining multi-table tablespaces, so this is the list of some “good old objects”.
You might question why these haven’t been converted to UTS since many Db2 features require UTS – maybe IBM has no plans to exploit UTS-only features for these objects ??
New catalog objects.
Catalog objects

- A historical view of the evolution in terms of #objects

<table>
<thead>
<tr>
<th></th>
<th>#TS</th>
<th>#LOB Cols</th>
<th>#TB</th>
<th>#TB Cols</th>
<th>#IX</th>
<th>#Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Db2 V1R1M0</td>
<td>11</td>
<td>0</td>
<td>25</td>
<td>291</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>.......</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Db2 V8</td>
<td>23</td>
<td>2</td>
<td>85</td>
<td>1333</td>
<td>133</td>
<td></td>
</tr>
<tr>
<td>Db2 V9</td>
<td>26</td>
<td>3</td>
<td>104</td>
<td>1714</td>
<td>166</td>
<td></td>
</tr>
<tr>
<td>Db2 10</td>
<td>77</td>
<td>18</td>
<td>116</td>
<td>2036</td>
<td>234</td>
<td>192</td>
</tr>
<tr>
<td>Db2 11</td>
<td>115</td>
<td>21</td>
<td>151</td>
<td>2231</td>
<td>250</td>
<td>197</td>
</tr>
<tr>
<td>Db2 12</td>
<td>149</td>
<td>33</td>
<td>185</td>
<td>2934</td>
<td>275</td>
<td>233</td>
</tr>
</tbody>
</table>

- Not all objects included (SYSTSTAB, DSNPROGAUTH, Accelerator tables, ...)
- DSNDB01 included since Db2 10
- Constraints include RI and table check constraints
- New tablespaces follow the “rule” from Db2 10/11: PBG and understandable naming convention

This is a short overview illustrating the Db2 catalog objects over the past +30 years. For comparison reasons, SYSTSTAB and a few other tablespace are excluded since they are not “mandatory”. DSNDB01 (the Directory) is included beginning with Db2 10 when these “tables” became select’able.

The good news in Db2 12 is, the naming convention and tablespace attributes follow the standard introduced in Db2 10 and continued in Db2 11.
A few ways to identify where you are on the Db2 Continuous Delivery Model: one being DISPLAY GROUP

One method to find out where you are and where you have been as well as where you can go is to execute the DIS GROUP command.

This is one method where you can verify:

1) Where you are in terms of maintenance – what is the highest possible FUNCTION LEVEL which can be activated due to the current maintenance.
2) Where you were previously
3) What function level you can use
Another method is looking at this new catalog table.

- Where are you: FUNCTION_LVL
- Where did you come from: PREV_FUNCTION_LVL
- Have you been in the future in the past: HIGH_FUNCTION_LVL
- Where is the Db2 catalog: CATALOG_LVL
- No indexes and might not be needed for years

One new table is SYSLEVELUPDATES which is very useful due to the agile approach Db2 has taken – becoming a continuous delivery model.

This table basically provides the same information as the DIS GROUP command, but I expect this table to also illustrate the historical path – still to be determined once new function levels become available.

The historical view of OPERATION TYPE and FUNCTION LEVEL could provide a lot of valuable information once Db2 12 gets “older”.
System Time Temporal available for RTS

- Optional to enable history of RTS tables.
- Two new tablespaces and tables

<table>
<thead>
<tr>
<th>TABLE_NAME</th>
<th>CREATOR</th>
<th>DATABASE</th>
<th>TBLSPACE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSIXSPACESTATS_H</td>
<td>SYSIBM</td>
<td>DSNDB06</td>
<td>SYSTSISH</td>
</tr>
<tr>
<td>SYSTABSPACESTATS_H</td>
<td>SYSIBM</td>
<td>DSNDB06</td>
<td>SYSTSTSH</td>
</tr>
</tbody>
</table>

- History table name length kept less than 18 byte (unfortunately?)

```
ALTER TABLE SYSIBM.SYSINDEXSPACESTATS
ADD VERSIONING USE HISTORY TABLE SYSIBM.SYSIXSPACESTATS_H;

ALTER TABLE SYSIBM.SYSTABLESPACESTATS
ADD VERSIONING USE HISTORY TABLE SYSIBM.SYSTABSPACESTATS_H;
```

Db2 10 introduced three types of Temporal Tables – and now Db2 12 offers the same functionality for two catalog tables: The RTS (Real Time Statistics) tables.

This is an optional feature not being mandated.

In order to activate SYSTEM TIME temporal tables for the RTS tables, two simple ALTER commands are needed (described in the SQL REFERENCE GUIDE).

From my opinion, I would have preferred to simply add “_H” to the RTS tables for the history tables instead of shortening them.
System Time Temporal – future planning?

• What is IBM planning? Maybe this provides a clue! ?

• Catalog Table changes in the past has not very often indicated what we can expect in the future – Db2 12 may be different

• We have had history tables for several releases covering statistics history: SYSIBM.SYSxxxxx_HIST

We just mentioned RTS can be enabled with system time temporal tables, but what’s quite interesting is to have a closer look at which tablespaces and tables exist in the Db2 12 catalog but are not referenced in the SQL Reference Guide’s Appendix A section describing the catalog tables.

This is probably the first time that the Db2 catalog tables are prepared for the future and kind of eludes what to expect.

The _HIST tables have existed for many releases providing the ability to store historical data for RUNSTATS. There are a complete new set of DIFFERENT history tables – let’s have a closer look.
System Time Temporal – future planning?

- One common component for these OLD and NEW objects – all have the same columns appended (see next slide).

<table>
<thead>
<tr>
<th>Tablespace</th>
<th>Table</th>
<th>AUX</th>
<th>Suggested Base Table</th>
<th>Suggested Base Tablespace</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSTSADH</td>
<td>SYSAUDITPOLICIES_H</td>
<td></td>
<td>SYSAUDITPOLICIES</td>
<td>SYSTSADT</td>
</tr>
<tr>
<td>SYSTSFAH</td>
<td>SYSCOLAUTH_H</td>
<td></td>
<td>SYSCOLAUTH</td>
<td>SYSTSFAU</td>
</tr>
<tr>
<td>SYSTSCNH</td>
<td>SYSCONTEXTAUTHID_H</td>
<td></td>
<td>SYSCONTEXTAUTHIDS</td>
<td>SYSCONTX</td>
</tr>
<tr>
<td>SYSTSCNH</td>
<td>SYSCONTEXT_H</td>
<td></td>
<td>SYSCONTEXT</td>
<td>SYSTSCON</td>
</tr>
<tr>
<td>SYSTSCFH</td>
<td>SYSCONTROLS_H</td>
<td></td>
<td>SYSCONTROLS</td>
<td>SYSTSCFT</td>
</tr>
<tr>
<td>SYSTSCDH</td>
<td>SYSCONTROLS_DESC_H</td>
<td></td>
<td>SYSCONTROLS_DESC</td>
<td>SYSTSCD</td>
</tr>
<tr>
<td>SYSTSCHR</td>
<td>SYSCONTROLS_RXTXT_H</td>
<td></td>
<td>SYSCONTROLS_RXTXT</td>
<td>SYSTSCRT</td>
</tr>
<tr>
<td>SYSTSTAH</td>
<td>SYSCXTTRUSTATTR_H</td>
<td></td>
<td>SYSCXTTRUSTATTRS</td>
<td>SYSCONTXT</td>
</tr>
<tr>
<td>SYSTDBH</td>
<td>SYSDBAUTH_H</td>
<td></td>
<td>SYSDBAUTH</td>
<td>SYSTDBU</td>
</tr>
<tr>
<td>SYSTSPKH</td>
<td>SYSPACKAUTH_H</td>
<td></td>
<td>SYSPACKAUTH</td>
<td>SYSTSPKA</td>
</tr>
<tr>
<td>SYSTPLAH</td>
<td>SYSPLANAUTH_H</td>
<td></td>
<td>SYSPLANAUTH</td>
<td>SYSTPLA</td>
</tr>
<tr>
<td>SYSTSAUH</td>
<td>SYSPRESAUTH_H</td>
<td></td>
<td>SYSPRESAUTH</td>
<td>SYSPPAUT</td>
</tr>
<tr>
<td>SYSTSAUH</td>
<td>SYSPRESAUTH_H</td>
<td></td>
<td>SYSPRESAUTH</td>
<td>SYSPPAUT</td>
</tr>
<tr>
<td>SYSTSAH</td>
<td>SYSROUTEAUTH_H</td>
<td></td>
<td>SYSROUTEAUTH</td>
<td>SYSTSRAU</td>
</tr>
<tr>
<td>SYSTSCH</td>
<td>SYSSCHEMAAUTH_H</td>
<td></td>
<td>SYSSCHEMAAUTH</td>
<td>SYSTSCM</td>
</tr>
<tr>
<td>SYSTSAH</td>
<td>SYSSEQUENCEAUTH_H</td>
<td></td>
<td>SYSSEQUENCEAUTH</td>
<td>SYSSSEQZ</td>
</tr>
<tr>
<td>SYSTSSH</td>
<td>SYSTABAUTH_H</td>
<td></td>
<td>SYSTABAUTH</td>
<td>SYSTSTAU</td>
</tr>
<tr>
<td>SYSTSUAH</td>
<td>SYSSHUSERAUTH_H</td>
<td></td>
<td>SYSSHUSERAUTH</td>
<td>SYSSHUSER</td>
</tr>
<tr>
<td>SYSTSVH</td>
<td>SYSVIEWDEP_H</td>
<td></td>
<td>SYSVIEWDEP</td>
<td>SYSTSVDW</td>
</tr>
</tbody>
</table>

To me these tables seem to indicate that we can expect all the tables holding authorizations (GRANT’s) to be temporal as well – they all have a couple of attributes in common (please see next page).
**System Time Temporal – future planning?**

- These appended columns could indicate a potential enablement of System Time Temporal Table (just like the RTS tables can be enabled in Db2 12)

```sql
CREATE TABLE "SYSIBM"."SYSSEQUENCEAUTH"
(GRANTOR VARCHAR(128) FOR MIXED DATA NOT NULL,
GRANTEE VARCHAR(128) FOR MIXED DATA NOT NULL,
"SCHEMA" VARCHAR(128) FOR MIXED DATA NOT NULL,
"NAME" VARCHAR(128) FOR MIXED DATA NOT NULL,
GRANTEETYPE CHARACTER(1) FOR MIXED DATA NOT NULL,
AUTHHOWGOT CHARACTER(1) FOR MIXED DATA NOT NULL,
ALTERAUTH CHARACTER(1) FOR MIXED DATA NOT NULL,
USEAUTH CHARACTER(1) FOR MIXED DATA NOT NULL,
"COLLID" VARCHAR(128) FOR MIXED DATA NOT NULL,
CONFOKEN CHARACTER(8) FOR BIT DATA NOT NULL,
GRANTEETS TIMESTAMP (6) WITHOUT TIME ZONE NOT NULL,
IBMREQD CHARACTER(1) FOR MIXED DATA NOT NULL,
GRANTORTYPE CHARACTER(1) FOR MIXED DATA NOT NULL WITH DEFAULT,
"SYS_START" TIMESTAMP (12) WITHOUT TIME ZONE NOT NULL,
"SYS_END" TIMESTAMP (12) WITHOUT TIME ZONE NOT NULL,
"PERIOD SYSTEM_TIME" (SYS_START, SYS_END)
) IN DSNDB06.SYSEQ2
```

All these tables have had the special columns/attributes appended which are needed to turn them into temporal tables. However – if you try to enable history, Db2 will block you from doing so.
System Time Temporal – future planning?

- Another view of “preparation for the future”
- System Time changes – but not enabled to history tables.

```sql
-- SYSXXX.SYSAUDITPOLICIES WILL BE ALTERED VIA THESE NATIVE Db2 COMMANDS

ALTER TABLE SYSXXX.SYSAUDITPOLICIES
    ADD SYS_START TIMESTAMP (12) NOT NULL Generated ALWAYS AS ROW BEGIN;
ALTER TABLE SYSXXX.SYSAUDITPOLICIES
    ADD SYS_END TIMESTAMP (12) NOT NULL Generated ALWAYS AS ROW END;
ALTER TABLE SYSXXX.SYSAUDITPOLICIES
    ADD TRANS_START TIMESTAMP (12) NOT NULL Generated ALWAYS AS TRANSACTION START ID;
ALTER TABLE SYSXXX.SYSAUDITPOLICIES
    ADD PERIOD SYSTEM_TIME (SYS_START, SYS_END);
```

Doing a schema compare between these tables in Db2 11 and Db2 12 illustrated that the Db2 upgrade process has appended the needed columns to the base catalog tables.
System Time Temporal – future planning?

- Maybe we are getting ready to get an “autonomic audit trail” of GRANT/REVOKE
- If so – changes are required for some catalog objects.
  - SYSSEQUENCIAUTH sits in a tablespace with TWO tables
  - VCAT defined tablespace (like prior to other tablespaces converted to PBG)

```
RQTST 19.0 ------ RC/Q Table Space Table Inquiry ------- 2016/12/31 09:17
COMMAND ===> SCROLL ===> CBR
Db2 Object ===> TS  Option ===> T  Where => N  
Table Space ===> SYSSEQ2  > Creator ===> *  > 
Data Base ===> DSNDB06  > N/A  ===> *  > 
Loc: LOCAL  SSID: D12A  RASST02  =  LINE 1 OF 3  >
CMD        NAME            CREATOR DATABASE N. ROWS
------------        ------------      -------      -------      -------
SYSSEQ2          SYSIBM        DSNDB06     
SYSSEQUENCEAUTH  SYSIBM        DSNDB06  252
SYSSEQUENCESDEP  SYSIBM        DSNDB06  284

******************************************************************************
```

My personal opinion is we’re getting prepared to provide an automatic method to audit who is granting/revoking which auth’s from who.

If this is going to happen, there are a couple of schema changes needed since temporal tables have to reside in UTS tablespaces and violations do exist:

1) VCAT defined tablespaces not yet converted to PBG
2) Some tablespaces have more than one table.
Another interesting topic is SYSVIEWDEP_H
  - This is not a table holding authorizations
  - Maybe due to TRANSFER OWNERSHIP?

Your guess as good as mine

When looking at the previous pages, there clearly seems to be a theme.
SYSVIEWDEP doesn’t really fall into the category of holding AUTH’s, so why does this table follow the same pattern?
New Catalog Objects

- Dynamic Query Stabilization
  - An extension to what Db2 11 delivered for static queries.
  - Two tables and five LOBs.

Let's have a look at the NEW catalog tables which really can be used.

I have divided them into the Db2 12 themes to hopefully get a better overview of the new Db2 12 features.

Db2 11 offered the opportunity to lock down the access path for static packages/queries. Db2 12 has extended this to DYNAMIC SQL.

In order to support this – 5 new tables and 2 AUX objects are available.
New Catalog Objects

- SYSDYNQRY and SYSDYNQRYDEP indexes
  - Indexes seem to follow “normal standard”
  - Might consider reversing first two columns of DSNDQX04

<table>
<thead>
<tr>
<th>TABLE NAME</th>
<th>INDEX NAME</th>
<th>INDEXED COLUMN</th>
<th>UNQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSDYNQRY</td>
<td>DSNDQX01</td>
<td>SDQ_STMT_ID</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COPTID</td>
<td>D</td>
</tr>
<tr>
<td>DSNDQX02</td>
<td></td>
<td>CURSCHEMA</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>QUERY_HASH</td>
<td>D</td>
</tr>
<tr>
<td>DSNDQX11</td>
<td></td>
<td>STBLGRP</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDQ_STMT_ID</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COPTID</td>
<td>D</td>
</tr>
<tr>
<td>DSNDQX03</td>
<td></td>
<td>SDQ_STMT_ID</td>
<td>D</td>
</tr>
<tr>
<td>DSNDQX04</td>
<td></td>
<td>BQUALIFIER</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BNAME</td>
<td>D</td>
</tr>
<tr>
<td>DSNDQX05</td>
<td></td>
<td>CLASS</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AUTHID</td>
<td>D</td>
</tr>
<tr>
<td>DSNDQX12</td>
<td></td>
<td>CLASS</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BTYPE</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BAUTH</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AUTHID_TYPE</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BQUALIFIER</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BNAME</td>
<td>D</td>
</tr>
</tbody>
</table>

The indexes created for these tables seem to match what’s needed if this feature will be heavily exploited.

The only modification which should be considered is to create one additional index reversing BQUALIFIER and BNAME depending on how much this will be used and how many distinct object qualifiers do exist (pretty much like CREATOR, NAME for SYSINDEXES, SYSTABLES etc.)
New Catalog Objects

- **IN-MEMORY** indexes (aka FTB) – another autonomic feature with control.
  - Eligible for UNIQUE indexes where key size < 64K
  - DSNZPARM : INDEX_MEMORY_CONTROL
  - Control which indexes not to be considered / when
  - TS=SYSTSICO
  - TYPE could indicate future options (no check what’s inserted – NO RI)

```sql
CREATE TABLE "SYSIBM".SYSINDEXCONTROL
  (SSID CHARACTER(4) FOR MIXED DATA WITH DEFAULT NULL
   ,"PARTITION" SMALLINT WITH DEFAULT NULL
   ,IXNAME VARCHAR(128) FOR MIXED DATA NOT NULL
   ,IXCREATOR VARCHAR(128) FOR MIXED DATA NOT NULL
   ,"TYPE" CHARACTER(1) FOR MIXED DATA NOT NULL WITH DEFAULT 'F'
   ,ACTION CHARACTER(1) FOR MIXED DATA NOT NULL WITH DEFAULT 'A'
   ,"MONTH_WEEK" CHARACTER(1) FOR MIXED DATA WITH DEFAULT NULL
   ,"MONTH" SMALLINT WITH DEFAULT NULL
   ,"DAY" SMALLINT WITH DEFAULT NULL
   ,FROM_TIME TIME WITH DEFAULT NULL
   ,TO_TIME TIME WITH DEFAULT NULL
  ) IN DSNDB06.SYSTSICO
```

Another new table is SYSINDEXCONTROL which is used to administer a new feature for in-memory indexes.

The look and feel is pretty similar to SYSINDEXCLEANUP in the nature that you can specify which indexes should take advantage of this FTB feature as well as when.

Looking at the TYPE column, this could indicate preparation for the future since only ‘F’ is current a valid option, so why need a type?
New Catalog Objects

• SYSINDEXCONTROL has no indexes defined

• You might consider creating one depending on usage:
  • If you don’t let Db2 decide when to use
  • If you want to highly customize which indexes to use FTB
  • If you want to be very specific about which weeks/days and timeframe

```sql
CREATE TABLE "SYSIBM".SYSINDEXCONTROL
  (SSID CHARACTER(4) FOR MIXED DATA WITH DEFAULT NULL,
   "PARTITION" SMALLINT WITH DEFAULT NULL,
   IXNAME VARCHAR(128) FOR MIXED DATA NOT NULL,
   ICREATOR VARCHAR(128) FOR MIXED DATA NOT NULL,
   "TYPE" CHARACTER(1) FOR MIXED DATA NOT NULL WITH DEFAULT 'F',
   ACTION CHARACTER(1) FOR MIXED DATA NOT NULL WITH DEFAULT 'A',
   MONTH_WEEK CHARACTER(1) FOR MIXED DATA WITH DEFAULT NULL,
   "MONTH" SMALLINT WITH DEFAULT NULL,
   "DAY" SMALLINT WITH DEFAULT NULL,
   FROM TIME TIME WITH DEFAULT NULL,
   TO TIME TIME WITH DEFAULT NULL
  ) IN DSNDB06.SYSTSICO
```

There are no indexes defined for SYSINDEXCONTROL, so if you plan on really controlling this feature instead of letting Db2 master the decision, you should consider a user-defined index based on usage.
New Catalog Objects

• Maintain session data on target server to reduce network traffic.
  • Three base tables and two AUX tables.
  • Supports transaction rerouting and pooling.
  • Distributed clients need session data (special registers and global variables) to be returned.

<table>
<thead>
<tr>
<th>Db2 Object</th>
<th>Option</th>
<th>Where</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table Name</td>
<td>Creator</td>
<td>&gt; N/A</td>
</tr>
<tr>
<td>Qualifier</td>
<td>*</td>
<td>&gt; *</td>
</tr>
<tr>
<td>Loc: LOCAL</td>
<td>SSID: D12A</td>
<td>&gt; LINE 1 OF 5</td>
</tr>
</tbody>
</table>

If you have applications where you need to reduce network traffic by maintaining session data, three base tables and two AUX objects are delivered to support this feature.
New Catalog Objects

- SYSSESSION indexes seem to be appropriate and no immediate need to create your own for performance:

<table>
<thead>
<tr>
<th>TABLE NAME</th>
<th>INDEX NAME</th>
<th>INDEXED COLUMN</th>
<th>COLSEQ</th>
<th>ORD</th>
<th>CLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSSESSION</td>
<td>DSNSNX02</td>
<td>TOKEN</td>
<td>1</td>
<td>A</td>
<td>N</td>
</tr>
<tr>
<td>SYSSESSION_DATA</td>
<td>DSNSNX03</td>
<td>AUXID</td>
<td>1</td>
<td>A</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AUXVER</td>
<td>2</td>
<td>A</td>
<td>N</td>
</tr>
<tr>
<td>SYSSESSION_EX</td>
<td>DSNSNX04</td>
<td>TOKEN</td>
<td>1</td>
<td>A</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>DSNSNX05</td>
<td>TOKEN</td>
<td>1</td>
<td>A</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GVID</td>
<td>2</td>
<td>A</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LOCATOR</td>
<td>3</td>
<td>A</td>
<td>N</td>
</tr>
<tr>
<td>SYSSESSION_GV</td>
<td>DSNSNX01</td>
<td>AUXID</td>
<td>1</td>
<td>A</td>
<td>N</td>
</tr>
<tr>
<td>SYSSESSION_STATUS</td>
<td>DSNSNX06</td>
<td>TOKEN</td>
<td>1</td>
<td>A</td>
<td>N</td>
</tr>
</tbody>
</table>

The indexes delivered with SESSION DATA tables seem to be sufficient and I see no need to create your own indexes for performance reasons.
New Catalog Objects

- New indexes on existing tables:
  - DSNDCX06 on SYSCOLUMNS
    (TBCREATOR ASC, TBNAME ASC, COLNO ASC)
  - DSNOVX04 on SYSVARIABLES
    (TYPESCHEMA ASC, TYPENAME ASC)
  - DSNQYX04 on SYSQUERY
    (QUERY_SEC_HASH ASC, SCHEMA ASC, SOURCE ASC)

- Changed indexes:
  - ALTER INDEX DSNATX02 ADD COLUMN (UNLOADAUTH ASC)
    • SYSTABAUTH non-unique index
  - ALTER INDEX DSNOTX01 ADD COLUMN (VERSION ASC)
    • Unique Clustering for SYSTRIGGERS

Three new indexes are created on existing catalog tables and two existing indexes have an additional column appended. SYSTABAUTH has a new column appended – UNLOADAUTH – which is another new Db2 12 feature. SYSTRIGGERS has VERSION appended which also is a new column for a new feature.
Modified catalog objects
(\textit{Db2 12 <-> Db2 11 compare})
Modified Objects

• Encoding scheme a new attribute on the column level

```
ALTER TABLE SYSCOLUMNS
ADD ENCODING_SCHEME CHAR ( 1 )
NOT NULL WITH DEFAULT FOR MIXED DATA ;

ALTER TABLE SYSCONTROLS
ADD REGENERATETS TIMESTAMP ( 12 )
NOT NULL WITH DEFAULT ;
```

• Row permissions and column masks can be
   REGENERATED via ALTER command and timestamp
   recorded *(alter command did exist in Db2 11 as well)*

Another new feature is that it is now possible to have the CCSID on the column level and not being dictated on the table level.

To support this, SYSCOLUMNS has a new column=ENCODING_SCHEME appended.

Even though Db2 11 had an ALTER command to REGENERATE row permissions and column masks, SYSCONTROLS has a new column added illustrating when this command was executed.
Modified Objects

- Two new columns for SYSENVIRONMENT – not described in the SQL Reference Guide “Catalog Tables”.

```sql
ALTER TABLE SYSIBM.SYSENVIRONMENT
ADD CREATEDTS TIMESTAMP ( 12 )
NOT NULL WITH DEFAULT ;

ALTER TABLE SYSIBM.SYSENVIRONMENT
ADD APPLCOMPAT VARCHAR ( 10 )
NOT NULL WITH DEFAULT 'V11R1' FOR MIXED DATA;
```

SYSENVIRONMENT has two new columns appended, but these are not described in appendix A in the SQL Reference Guide, so I am not sure whether they are used or not.

One new column is when the row was inserted and another column is APPLCOMPAT.

To me it seems like these two columns are useful however.
Modified Objects

• SYSINDEXES ready for partition wise attributes and new tablespace type *(Italic marked reserved for the future).*

<table>
<thead>
<tr>
<th>Column</th>
<th>Data Type</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSIZE</td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>PAGENUM</td>
<td>CHAR (1)</td>
<td>NOT NULL WITH DEFAULT 'A';</td>
</tr>
<tr>
<td>PARTKEYCOLNUM</td>
<td>SMALLINT</td>
<td>NOT NULL WITH DEFAULT;</td>
</tr>
<tr>
<td>STATUS</td>
<td>CHAR (1)</td>
<td>NOT NULL WITH DEFAULT;</td>
</tr>
<tr>
<td>INDEXSTATUS</td>
<td>VARCHAR (30)</td>
<td>NOT NULL WITH DEFAULT;</td>
</tr>
<tr>
<td>PARTITIONS</td>
<td>SMALLINT</td>
<td></td>
</tr>
<tr>
<td>PQTY</td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>STORTYPE</td>
<td>CHAR (1)</td>
<td></td>
</tr>
<tr>
<td>STORNAME</td>
<td>VARCHAR (128)</td>
<td></td>
</tr>
<tr>
<td>VCATNAME</td>
<td>VARCHAR (24)</td>
<td></td>
</tr>
<tr>
<td>FREEPAGE</td>
<td>SMALLINT</td>
<td></td>
</tr>
<tr>
<td>PCTFREE</td>
<td>SMALLINT</td>
<td></td>
</tr>
<tr>
<td>GBPCACHE</td>
<td>CHAR (1)</td>
<td></td>
</tr>
<tr>
<td>SECQTYI</td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>ENFORCED_CONS</td>
<td>CHAR (1)</td>
<td>NOT NULL WITH DEFAULT;</td>
</tr>
<tr>
<td>IMPLICIT</td>
<td>CHAR (1)</td>
<td></td>
</tr>
<tr>
<td>REGENERATETS</td>
<td>TIMESTAMP (12)</td>
<td>NOT NULL WITH DEFAULT;</td>
</tr>
</tbody>
</table>

A large number of new columns have been added to SYSINDEXES. All the columns in the box marked in ITALIC are considered for future use. These new columns are related to the new tablespace type (PBR2 or tablespaces with relative addressing) where partition level attributes will be available etc.
Modified Objects

• SYSINDEXPART also ready for relative page numbering.

```
DSSIZE    INTEGER;
PAGENUM   CHAR ( 1 ) NOT NULL WITH DEFAULT 'A';
LIMITKEY_EXTERNAL VARCHAR ( 765 ) NOT NULL WITH DEFAULT ;
```

The same goes for SYSINDEXPART which is ready for the new tablespace type PBR2.
Modified Objects

• Real Time Statistics tables (besides System Time Temporal columns)

```
ALTER TABLE SYSIBM.SYSINDEXSPACESTATS
  ADD GETPAGES BIGINT WITH DEFAULT 0 ;

ALTER TABLE SYSIBM.SYSTABLESPACESTATS
  ADD GETPAGES BIGINT WITH DEFAULT 0 ;
```

We already covered the Real Time Statistics changes where it’s now possible to enable System Time Temporal. Besides this change both RTS tables now also counts the GETPAGE activity.
Modified Objects

• SYSPACKAGE has some very nice pieces of information
  • How the BIND was performed (eg. Auto-Rebind)
  • If CONCENTRATE statement is enabled
  • When APREUSE was disabled
  • Db2 FUNCTION_LEVEL when the row was inserted

| ORIGIN CHAR ( 1 ) NOT NULL WITH DEFAULT; |
| APREUSE_NO_FL VARCHAR ( 10 ) NOT NULL WITH DEFAULT; |
| APREUSE_NO_TS TIMESTAMP NOT NULL WITH DEFAULT; |
| CONC_STMT CHAR ( 1 ) NOT NULL WITH DEFAULT 'N'; |
| FUNCTION_LVL VARCHAR ( 10 ) NOT NULL WITH DEFAULT; |

• SYSPACKCOPY has the same columns appended.
• SYSQUERY has FUNCTION_LVL appended.

SYSPACKAGE has some new columns which seem extremely useful.

1) How the package got bound: REBIND, BIND, Automatic Rebind etc.
2) FUNCTION-LEVEL when APREUSE(NO) was executed as well as timestamp.
3) If CONCENTRATE_STATEMENT was specified for literals/constants.
4) Finally the Db2 FUNCTION-LEVEL when the row was inserted.
### Modified Objects

- **SYSPACKSTMT** has new columns matching **SYSQUERY**

```sql
QUERYID BIGINT NOT NULL WITH DEFAULT - 1;
QUERY_HASH CHAR (16) NOT NULL WITH DEFAULT X'00000000000000000000000000000000' FOR BIT DATA;
QUERY_HASH_VERSION INTEGER NOT NULL WITH DEFAULT -1;
```

- **SYSPENDINGDDL** partition-range for logical parts to be reorganized instantiating pending changes

```sql
REORG_SCOPE_LOWPART SMALLINT;
REORG_SCOPE_HIGHPART SMALLINT;
```

SYSPACKSTMT has been appended with the same columns as SYSQUERY so it’s possible to join these.

Since attributes can be altered for PBR2 partitions independently, SYSPENDINGDDL now keeps track of the highest and lowest partition with PENDING changes since adjacent logical partitions must be reorganized together to materialize pending definition changes.
Modified Objects

• SYSROUTINES describes if obfuscation is active

```
ALTER TABLE SYSIBM.SYSROUTINES
ADD WRAPPED CHAR ( 1 ) NOT NULL WITH DEFAULT ;

ALTER TABLE SYSIBM.SYSROUTINES
ADD REGENERATETS TIMESTAMP ( 12 )
NOT NULL WITH DEFAULT ;
```

• And yet another catalog table has REGENERATE timestamp appended.

• SYSTABLES also has REGENERATE timestamp.

Routines can be obfuscated and REGENERATED using ALTER so these are reflected as new columns.
Modified Objects

- SYSTABLEPART reflects the new tablespace type (absolute or relative page numbering)
- Partition level physical attributes reflected.

```
TYPE          CHAR ( 1 ) ;
PAGENUM       CHAR ( 1 ) NOT NULL WITH DEFAULT 'A' ;
BPOOL         CHAR ( 8 ) ;
PGSIZE        SMALLINT ;
DSSIZE        INTEGER ;
MEMBER_CLUSTER CHAR ( 1 );
COMPRESSRATIO SMALLINT NOT NULL WITH DEFAULT - 1 ;
```

Due to the new tablespace type (PBR2 or relative page addressing), SYSTABLEPART has 7 new columns reflecting this feature along with the attributes which now can be different for the individual partitions.
Modified Objects

- SYSTABLESPACE.
  - Relative/absolute page numbering.
  - Interesting that this table now has inherited columns from SYSTABLEPART.

```
PAGENUM       CHAR ( 1 ) NOT NULL WITH DEFAULT 'A';
INSERTALG     SMALLINT NOT NULL WITH DEFAULT;
PQTY          INTEGER;
STORTYPE      CHAR ( 1 );
STORNAME      VARCHAR ( 128 );
VCATNAME      VARCHAR ( 24 );
FREEPAGE      SMALLINT;
PCTFREE       SMALLINT;
COMPRESS      CHAR ( 1 );
GBPCACHE      CHAR ( 1 );
TRACKMOD      CHAR ( 1 );
SECTRYI       INTEGER;
PCTFREE_UPD   SMALLINT;
PCTFREE_UPD_CALC SMALLINT;
COMPRESSRATIO SMALLINT NOT NULL WITH DEFAULT - 1;
```

SYSTABLESPACE also reflects the new tablespace type.

What is interesting is that some attributes are inherited from SYSTABLEPART like PCTFREE, SECTRYI etc.
Modified Objects

• SYSTRIGGERS
  • Versioning available (like NSP’s)
  • Trigger text can be obfuscated

```
VERSION VARCHAR (122) NOT NULL WITH DEFAULT;
ORIGINAL_CONTOK CHAR (8) FOR BIT DATA;
REGENERATETS TIMESTAMP NOT NULL WITH DEFAULT;
ACTIVE CHAR (1) FOR MIXED DATA;
WRAPPED CHAR (1) NOT NULL WITH DEFAULT;
```

Triggers can now be versioned like Native Stored Procedures and they can also be obfuscated.
New Db2 12 features reflected in the catalog
(existing columns with additional value/meaning)
Db2 12 seems to have less changes to existing columns compared to the past couple of releases. This section will only cover some of these.

**PERIOD** in `syscolumns`, `syskeycoluse` and `syskeys` : For temporal tables, ENDING can now be either inclusive or exclusive.

**SYSDEPENDENCES** has an indicator whether the trigger is the old style or it's an advanced trigger and the same goes for `syspackage` and `syspackdep`.
User Defined Functions available

- Some of these existed since several Db2 releases
  - Execute highlvl.SDSNSAMP(DSNTEJ2U)
  - Requires C / C++ compiler license
  - Use as samples to create you own – or modify
  - Here are just a few of the available UDF’s

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALTDATE</td>
<td>Returns the current date or a user-specified date in a user-specified format</td>
</tr>
<tr>
<td>ALTTIME</td>
<td>Returns the current time or a user-specified time in a user-specified format</td>
</tr>
<tr>
<td>CURRENCY</td>
<td>Returns a floating-point number as a currency value</td>
</tr>
<tr>
<td>DAYNAME</td>
<td>Returns the name of the day of the week on which a date in ISO format falls</td>
</tr>
<tr>
<td>MONTHNAME</td>
<td>Returns the name of the month in which a date in ISO format falls</td>
</tr>
<tr>
<td>TABLE_LOCATION</td>
<td>Returns the location name of a table or view after resolving any aliases</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>Returns the unqualified name of a table or view after resolving any aliases</td>
</tr>
<tr>
<td>TABLE_SCHEMA</td>
<td>Returns the schema name of a table or view after resolving any aliases</td>
</tr>
<tr>
<td>WEATHER</td>
<td>Shows how to use a user-defined table function to make non-relational data available for SQL manipulation</td>
</tr>
</tbody>
</table>

UDF’s are not widely used as far as I have experienced, but I stumbled across this section while preparing this presentation. Most of these routines are not new, but I am positive some of the attendees will find these useful. Source code do exist for all of these (only a fraction listed here) so you can easily pick up what’s needed and then change to do whatever you need.
User Defined Functions available

- In member DSNTEJ2U – find the UDF needed and look for EXTERNAL NAME
- This is the example for ALTDATE using member DSN8DUAD

```c
module name = DSN8DUAD (DB2 sample program)
DESCRIBITIVE NAME = Current date reformatter (CDF)

LICENSED MATERIALS - PROPERTY OF IBM
$FILE-DB2
(C) COPYRIGHT 1988, 2004 IBM CORP. ALL RIGHTS RESERVED.
STATUS = VERSION 0

Function: Returns the current date in one of these 34 formats:
D MONTH YY D MONTH YYYY DE MONTH YY DE MONTH YYYY
D-M-YY D-M-YYYY D-M-YYYY D-M-YYYY
S/M/YY S/M/YYYY S/M/YYYY S/M/YYYY
M/D/YY M/D/YYYY M/D/YYYY M/D/YYYY
Y/T/H/D YYYY/H/DD YYYY/H/DD YYYY/H/DD
Y/T/H/DD YYYY/H/DD YYYY/H/DD YYYY/H/DD
where:
D: Suppress leading zeros if the day is less than 10
M: Suppress leading zeros if the month is less than 10
Y: Use a capital Roman numeral for year
T: Use a two-digit year format
```

Have a look at highlvl.SDSNSAMP (DSNTEJ2U) and find the UDF you can take advantage from and look for EXTERNAL NAME.

This is a snippet from the ALTDATE routine which can automate the conversion of a date into 30+ formats.
This presentation was prepared using primarily a Db2 12 system FUNCTION LEVEL V12R1M500.

I did take a peek into the IBM Db2 SQL Reference Guide to validate new columns and column content.

Also, please take a look at the white paper created late 2016 by a lot of famous Db2’ers around the world.
Wrap Up

Thank You - QUESTIONS ?