IDUG

IDUG
2024 NA Db2 Tech Conference

Implicitly or Explicitly Defined Db2 Objects – the Good, the Bad and the Ugly

Steen Rasmussen

Broadcom

@IDUGDb2
#IDUG_NA24

Session Code: SQL1 | Platform: Db2 z/OS

Abstract

Across the many releases of Db2 for z/OS, the database engine has introduced several significant changes to which and how objects can be created implicitly. This session will dive into the type of objects Db2 can create automatically, describing advantages and pitfalls for their respective usage. In the end, how this is accomplished will depend heavily on the SQL syntax and keywords used.

Agenda

- What can be defined implicitly & how to “semi control” behavior
- CURRENT RULES special register impact
- Challenges having implicitly/explicitly defined objects
- Advantages using implicitly/explicitly defined objects
- Real life scenarios to illustrate differences
  - Starting with the most simple use case gradually increasing complexity

Why Bother Using One or The Other(1)

- Do you have an object naming convention in place ?
  - It will be violated using Implicit objects
- Are you using LISTDEF processing and wildcarding ?
  - You might find this a challenge
- Are you using catalog queries based on your naming convention
  - This might get a lot harder – wildcarding based on naming standard probably not possible (unless you are very lucky)
- If implicitly defined database names used
  - There's a LIMIT
  - You can't explicitly create DBs prefixed DSN***nnnnn***
- Think about PRIQTY/SECQTY – you might have to ALTER unless *(MGEXTSZ)*

Let's have a closer look at why you might want to consider whether you want to utilize IMPLICITLY defined objects or not.

1) If you have the need to use your site's naming convention, you might not be able to take advantage of implicitly defined objects since you can't control the names of these objects.
2) If you are using LISTDEF and wildcarding for utility processing, you might have some challenges based on how you use implicitly defined objects.
3) If you are using your own queries – or even tooling and depend on wildcarding for the object names, this could provide a challenge so the queries might have to be adjusted.
4) Depending on how you utilize IMPLICITLY defined objects – and how much, you might run out of database names.
5) Lastly, if you don't use the sliding scheme but want to explicitly specify the QTY's, and depending on your ZPARM setting (TSQTY & IXQTY), you might have to ALTER the PRIQTY/SECQTY after the object is created.

Why Bother Using One or The Other(2)

- You can create certain DSN* databases

```
.CONNECT M10A
 BPA0198I: CURRENT FUNCTION LEVEL IS V12R1M508
RETCODE =      0

.OPTION NOERRORS NOSQLERRORS NOLOG SQLFORMAT(SQL)
RETCODE =      0


CREATE DATABASE DSNSTEEN;
 DSNT400I SQLCODE = 000,  SUCCESSFUL EXECUTION
CREATE DATABASE DSNT0000;
 DSNT400I SQLCODE = 000,  SUCCESSFUL EXECUTION
CREATE DATABASE DSN88888;
 DSNT408I SQLCODE = -20074, ERROR:  THE OBJECT DSN88888 CANNOT BE
          CREATED BECAUSE THE FIRST THREE CHARACTERS ARE RESERVED FOR
          SYSTEM OBJECTS
 DSNT418I SQLSTATE    = 42939 SQLSTATE RETURN CODE
 DSNT415I SQLERRP     = DSNXICDB SQL PROCEDURE DETECTING ERROR
 DSNT416I SQLERRD     = 2  0  0  -1  0  0 SQL DIAGNOSTIC INFORMATION
 DSNT416I SQLERRD     = X'00000002'  X'00000000'  X'00000000'
          X'FFFFFFFF'  X'00000000'  X'00000000' SQL DIAGNOSTIC
          INFORMATION
 BPA0012E: DB2 SQL/DDL ERROR HAS OCCURRED - ROLLBACK ISSUED.
RETCODE =      8
```

You can't explicitly create DBs prefixed DSN***nnnnn.***

However, you can create DB's prefixed with DSN as long as it's not all numeric – despite what the Db2 message indicates. In this example I was able to create DSNSTEEN but getting message DSNT408I when the first four bytes are DSN***n***

What can be created IMPLICITLY

- Database for Table if not specified.
- Tablespace for Table if not specified.
- Index for Uniqueness / Constraint if the Table's Tablespace Implicitly created.
- ROWID GENERATED BY DEFAULT when RULES=STD or Tablespace Implicitly created.
- xLOB objects depends on RULES and whether Tablespace Implicitly created (use cases to follow).
- For XML the DOCID

What can be created IMPLICITLY

- *Are you happy with the DEFAULT Tablespace/Indexspace attributes ?*
- *If not - talk to your favorite Db2 SYSPROG to modify ZPARM attributes/values (those you can modify)*

```
       ----------- Thread Terminator DSNZPARM Display ---------- 23/07/10 17:25
Command ====>                                                   Scroll ==> CSR
                                                                LINE 319 OF 363

DB2 SSID ==> D12A        Lmod Name ==> D12APARM       Assem Date ==> 04/19/23

--------------------------------------------------------------------------------
  DESCRIPTION                                      VALUE      KEYWORD
  ------------------------------------------------ ---------- -----------------
  Default 16K buffer pool for implicit tablespaces BP16K1     TBSBP16K
  Default 32K buffer pool for implicit tablesspace BP32K1     TBSBP32K
  Default 8K buffer pool for implicit tablespaces. BP8K1      TBSBP8K
```

```
CREATE TABLESPACE IDUGEU23
       USING STOGROUP SYSDEFLT
           PRIQTY -1
           SECQTY -1
           ERASE NO
       BUFFERPOOL BP1
       DSSIZE 4G
       CLOSE YES
       LOCKMAX SYSTEM
       SEGSIZE 32
       INSERT ALGORITHM 0
       FREEPAGE 0
       PCTFREE 5 FOR UPDATE 0
       GBPCACHE CHANGED
       DEFINE NO
       LOGGED
       TRACKMOD YES
       MAXPARTITIONS 256
       COMPRESS NO
       LOCKSIZE ROW
       MAXROWS 255
       CCSID EBCDIC
       NUMPARTS 1;
```

Before exploiting IMPLICITLY defined objects – and even EXPLICILTLY defined objects where you don't specify all the attributes, it might be worth to understand your ZPARM DEFAULT settings.

SET CURRENT RULES = ‘xxx’

- DB2 is the default
  - Will eliminate many of the Implicit Object definitions
- STD
  - Will enable more Implicit Object definitions

(will be covered in detail later using a PBR use case with a CLOB)

DB2 is the default value and has been around for quite a while.
Using STD will open the doors for more objects to be created IMPLICITLY and we will cover these later.

IDUG

Leading the Db2 User Community for 35 Years

Use Case 1 : The Simple Scenario

#IDUGDb2

Most simple use case where implicit objects can be introduced (1)

- Table created in 4 different ways:
  - DB and TS Explicit (1) and DB and/or TS Implicit (2+3+4)

```
   CREATE TABLE RASST02.IDUGEU23TB1
       (DEPTNO CHARACTER(3) FOR SBCS DATA NOT NULL
      ,CONSTRAINT DEPTNO PRIMARY KEY
      (DEPTNO)                  ) IN IDUGEU24.IDUGEUTS;
==================================================================
   CREATE TABLE RASST02.IDUGEU23TB2
       (DEPTNO CHARACTER(3) FOR SBCS DATA NOT NULL
      ,CONSTRAINT DEPTNO PRIMARY KEY
      (DEPTNO)                  ) IN DATABASE IDUGEU24;
==================================================================
   CREATE TABLE RASST02.IDUGEU23TB3
       (DEPTNO CHARACTER(3) FOR SBCS DATA NOT NULL
      ,CONSTRAINT DEPTNO PRIMARY KEY
      (DEPTNO)                  ) ;
==================================================================
SET CURRENT RULES = 'STD' ;
   CREATE TABLE RASST02.IDUGEU23TB4
       (DEPTNO CHARACTER(3) FOR SBCS DATA NOT NULL
      ,CONSTRAINT DEPTNO PRIMARY KEY
      (DEPTNO)                  ) ;
```

The final results on next page

There are basically four different ways to create a table – meaning the syntax.

1) The database and tablespace are explicitly referenced.
2) The tablespace name isn't specified – only the database.
3) Neither the tablespace or database names are specified.
4) Same as option 3) but using CURRENT RULES='STD'

Most simple use case where implicit objects can be introduced (2)

- Explicit DB.TS requires Explicit Unique Index.
  - DB or TS Implicitly created - Unique Index Implicitly created
  - CURRENT RULES no impact

Necessary to explicitly create UNIQUE INDEX

TABLE NAME	SCHEMA	DEPENDENT TS/IX		SCHEMA/DB
IDUGEU23TB1	RASST02	IDUGEU23TB1		RASST02
		* TABLE SPACE	1	
		IDUGEUTS		IDUGEU24
IDUGEU23TB2	RASST02	IDUGEU23TB2		RASST02
		* TABLE SPACE	1	
		* INDEX	1	
		IDUGEU23		IDUGEU24
		IDUGEU23_#_5KN		RASST02
IDUGEU23TB3	RASST02	IDUGEU23TB3		RASST02
		* TABLE SPACE	1	
		* INDEX	1	
		IDUGEU23		DSN01572
		IDUGEU23_#_AXN		RASST02
IDUGEU23TB4	RASST02	IDUGEU23TB4		RASST02
		* TABLE SPACE	1	
		* INDEX	1	
		IDUGEU23		DSN01573
		IDUGEU23_#_37N		RASST02

1) When the database and tablespace is specified explicitly, it is necessary to explicitly create the UNIQUE index for the constraint.
2) When the tablespace is omitted it is implicitly created in the specified database – at least the database name can be controlled. In this case the tablespace name is a substring of the tablename but it is a coincident. The UNIQUE index is implicitly created – maybe you don't like the name but that's what it is.
3) The third scenario, an IMPLICIT database is provided = DSN01572. This can be a challenge if the tablespace/table has to be dropped/recreated since you probably won't get the same names.
4) Using CURRENT RULES in this use case makes no difference.

Now include CLOB (1)

- Let's look at another object type and implicit create options.

```
CREATE TABLE RASST02.IDUGNA24TB1
    (DEPTNO CHARACTER(3) FOR SBCS DATA NOT NULL
    ,LOBDATA CLOB(1M) WITH DEFAULT NULL
    ,LOB_ROWID ROWID NOT NULL GENERATED ALWAYS  ) IN IDUGNA24.IDUGNATS;
======================================================================
CREATE TABLE RASST02.IDU24T23TB2
    (DEPTNO CHARACTER(3) FOR SBCS DATA NOT NULL
    ,LOBDATA CLOB(1M) WITH DEFAULT NULL
    ,LOB_ROWID ROWID NOT NULL GENERATED ALWAYS  ) IN DATABASE IDUGNA24;
======================================================================
CREATE TABLE RASST02.IDUGNA24TB3
    (DEPTNO CHARACTER(3) FOR SBCS DATA NOT NULL
    ,LOBDATA CLOB(1M) WITH DEFAULT NULL
    ,LOB_ROWID ROWID NOT NULL GENERATED ALWAYS  ) ;
======================================================================
SET CURRENT RULES = 'STD' ;
CREATE TABLE RASST02.IDUGNA24TB4
    (DEPTNO CHARACTER(3) FOR SBCS DATA NOT NULL
    ,LOBDATA CLOB(1M) WITH DEFAULT NULL
    ,LOB_ROWID ROWID NOT NULL GENERATED ALWAYS  ) ;
************************** Bottom of Data **************************
```

Next use case is very similar but instead of a PRIMARY KEY we now have a LOB column.
Otherwise the four methods are identical to the previous scenario.

Now include CLOB (2)

- Only using Implicit tablespace (but Explicit DB) – at least all objects are in the same DB *(think LISTDEF, SQL etc.)*

```
TABLE NAME        TABLE/INDEX       SCHEMA/DB TYPE
IDUGNA24TB1       IDUGNA24TB1       RASST02   TABLE
                  * TABLE SPACE   1
                  IDUGNATS          IDUGNA24  TABLE SPACE
IDUGNA24TB2       IDUGNA24TB2       RASST02   TABLE
                  * TABLE SPACE   2
                  * TABLE         1
                  * INDEX         1
                  IDUGNA24          IDUGNA24  TABLE SPACE
                  L0NRFSDJ          IDUGNA24  TABLE SPACE
                  IDUGNLOBDA0NRFVHWL RASST02  TABLE
                  IIDUGNLOBDA0NRFWS1 RASST02  INDEX
IDUGNA24TB3       IDUGNA24TB3       RASST02   TABLE
                  * TABLE SPACE   2
                  * TABLE         1
                  * INDEX         1
                  IDUGNA24          DSN01574  TABLE SPACE
                  L0MLQQY5          DSN01574  TABLE SPACE
                  IDUGNLOBDA0MLQUIC2 RASST02  TABLE
                  IIDUGNLOBDA0MLQUQK RASST02  INDEX
IDUGNA24TB4       IDUGNA24TB4       RASST02   TABLE
                  * TABLE SPACE   2
                  * TABLE         1
                  * INDEX         1
                  IDUGNA24          DSN01575  TABLE SPACE
                  L0MLQ5EI          DSN01575  TABLE SPACE
                  IDUGNLOBDA0MLQ9P3X RASST02  TABLE
                  IIDUGNLOBDA0MLQXZI RASST02  INDEX
```

- Explicitly referenced DB.TS incomplete until all AUX objects created explicitly.

- All other use cases are complete – downside is the provided object names.

The outcome is the same as the previous scenario – when tablespace and database is specified explicitly, you will have to manually create the AUX objects. Not a big deal for a PBG table.

IDUG

Leading the Db2 User Community for 35 Years

IDUG

Use Case 2 : Things Can Get Messy

#IDUGDb2

Things can get messy (1)

- Object details
  - PBG MAXPARTITIONS 2 NUMPARTS 2
  - Two CLOB columns
  - Explicitly defined :
    - 1 DB
    - 1 PBG TS
    - 4 LOB TS (2 LOBS x 2 PARTITIONS)
    - 1 BASE TB (and potentially base indexes)
    - 4 AUX tables
    - 4 AUX indexes
    - (Lots of typing – but naming convention maintained – so far …….)

In this use case we have a four partitioned PBG.
Since we define two partitions and the table has 2 LOB columns, we need 4 AUX tablespaces/tables and indexes.

This is a lot of typing to make sure all objects are created so Db2 considers the object (table) in a COMPLETE status and operational.

The good news is the desired naming convention can be maintained since everything created explicitly.

Things can get messy (2)

- Next step is to ADD a PBG partition (ALTER TABLE ADD PART)
  - ALTER MAXPARTITIONS from 2 -> 3
  - This will cause 2 LOB tablespaces to be added IMPLICITLY
  - Naming convention goes South
- Next ADD CLOB column to the base table
  - Explicitly definition:
  - 3 LOB tablespaces (3 partitions)
  - 3 AUX tables
  - 3 AUX indexes
- IMPLICITLY or EXPLICITLY defined depends on SET CURRENT RULES

Next task is to add a PBG partition due to growth, so MAXPARTITIONS altered to 3. This means we need two additional AUX tablespaces – these are created implicitly by Db2 – and our beautiful naming convention is lost.

Then we need to add a new LOB column, so we have to choose between EXPLICIT / IMPLICIT objects.
If EXPLICIT is the choice, we need 9 additional objects to be created.

If CURRENT RULES='STD' is used, we will save some typing to do.

Things can get messy (3)

- Status after initial create (naming convention maintained) :

```
 CMD      NAME      CREATOR   DNAME                DCREATOR  TYPE
 ________ IDUGEU23  RASST02   IDUGEU23             RASST02   DATA BASE
                              * TABLE SPACE    5
                              * TABLE          5
                              * INDEX          5
 ________                     BASELOBA             IDUGEU23  TABLE SPACE
 ________                     BASELOBB             IDUGEU23  TABLE SPACE
 ________                     BASELOBC             IDUGEU23  TABLE SPACE
 ________                     BASELOBD             IDUGEU23  TABLE SPACE
 ________                     BASEPBG              IDUGEU23  TABLE SPACE
 ________                     BASELOBA_PART1_CLO> RASST02   TABLE
 ________                     BASELOBB_PART2_CLO> RASST02   TABLE
 ________                     BASELOBC_PART1_CLO> RASST02   TABLE
 ________                     BASELOBD_PART2_CLO> RASST02   TABLE
 ________                     BASEPBG_TAB          RASST02   TABLE
 ________                     BASELOBA_PART1_IX1  RASST02   INDEX
 ________                     BASELOBA_PART2_IX1  RASST02   INDEX
 ________                     BASELOBC_PART1_IX2  RASST02   INDEX
 ________                     BASELOBC_PART2_IX2  RASST02   INDEX
 ________                     BASEPBG_TAB_IX       RASST02   INDEX
****************************** BOTTOM OF DATA ********************
```

Things can get messy (4)

- Status after ADD CLOB column.
  - You can control the EXPLICIT/IMPLICIT (via CURRENT RULES)
  - The cost is sacrificing the naming convention – but a lot easier (think about 100 partitions)

DNAME	DCREATOR	TYPE
DAVETCOL05SEOEGCFT	BLADA08	TABLE
DAVETCOL07SEOES8Y1	BLADA08	TABLE
DAVETS1_TAB	BLADA08	TABLE
DAVETS1A_PART1_CLO>	BLADA08	TABLE
DAVETS1A_PART2_CLO>	BLADA08	TABLE
DAVETS1B_PART1_CLO>	BLADA08	TABLE
DAVETS1B_PART2_CLO>	BLADA08	TABLE
DAVETS1C_PART1_CLO>	BLADA08	TABLE
DAVETS2C_PART2_CLO>	BLADA08	TABLE
DAVETS3C_PART3_CLO>	BLADA08	TABLE
DAVETS1_TAB_IX	BLADA08	INDEX
DAVETS1A_PART1_IX1	BLADA08	INDEX
DAVETS1A_PART2_IX1	BLADA08	INDEX
DAVETS1B_PART1_IX2	BLADA08	INDEX
DAVETS1B_PART2_IX2	BLADA08	INDEX
DAVETS1C_PART1_IX3	BLADA08	INDEX
DAVETS1C_PART2_IX3	BLADA08	INDEX
DAVETS1C_PART3_IX3	BLADA08	INDEX
IDAVETCOL05SEOEGB8	BLADA08	INDEX
IDAVETCOL07SEOESWT	BLADA08	INDEX

DB2 max part 2->3

DNAME	DCREATOR	TYPE
BASELOBA_PART1_CLO>	RASST02	TABLE
BASELOBB_PART2_CLO>	RASST02	TABLE
BASELOBC_PART1_CLO>	RASST02	TABLE
BASELOBD_PART2_CLO>	RASST02	TABLE
BASEPBG_TAB	RASST02	TABLE
BASEPCOL050HIMIBKY	RASST02	TABLE
BASEPCOL070HIMQ8AD	RASST02	TABLE
BASEPCOL080HINJRGD	RASST02	TABLE
BASEPCOL080HINRZBY	RASST02	TABLE
BASEPCOL080HINYG6N	RASST02	TABLE
BASELOBA_PART1_IX1	RASST02	INDEX
BASELOBA_PART2_IX1	RASST02	INDEX
BASELOBC_PART1_IX2	RASST02	INDEX
BASELOBC_PART2_IX2	RASST02	INDEX
BASEPBG_TAB_IX	RASST02	INDEX
IBASEPCOL050HIMI8T	RASST02	INDEX
IBASEPCOL070HIMQ6C	RASST02	INDEX
IBASEPCOL080HINJGU	RASST02	INDEX
IBASEPCOL080HINR9N	RASST02	INDEX
IBASEPCOL080HINZXS	RASST02	INDEX

STD add lob col

The LEFT side is without using CURRENT RULES so everything is explicitly defined. The two IMPLICITLY defined AUX tables/indexes are from ALTER MAXPARTITIONS from 2 to 3, so since we had two LOB columns, Db2 creates these objects implicitly so the object is operational.

The RIGHT hand side is when RULES='STD" was used to ADD the new LOB column as well as altering the MAXPARTITIONS. Everything handled by Db2 automatically – but we lost the naming convention.

IDUG

Leading the Db2 User Community for 35 Years

IDUG

Use Case 3 : Things Can Also Be Easy – Looking at PBR's

#IDUGDb2

Things can also be easy (1)

- PBR table with one LOB and 6 partitions.
  - Implicit DB and TS created (violating our naming convention *(DSN05572.HSBCIMP1) chosen by Db2)*.
  - All AUX objects created – object creation complete and ready to use.

```
CREATE TABLE RASST02.HSBCIMP1
        (COUNTRY VARCHAR(50)
        ,CITY VARCHAR(30)
        ,ZIPCODE CHARACTER(10)
        ,RESIDENTS INTEGER
        ,LOBDATA CLOB(2M) INLINE LENGTH 0
        ,LOB_ROWID ROWID GENERATED ALWAYS )
PARTITION BY RANGE
        (COUNTRY NULLS LAST ASC)
        ( PARTITION 1 ENDING ('G')   INCLUSIVE
        , PARTITION 2 ENDING ('J')   INCLUSIVE
        , PARTITION 3 ENDING ('M')   INCLUSIVE
        , PARTITION 4 ENDING ('ST')  INCLUSIVE
        , PARTITION 5 ENDING ('T')   INCLUSIVE
        , PARTITION 6 ENDING ('ZZZ') INCLUSIVE );
```

Let's add a little complexity to illustrate how helpful RULES = 'STD' can be – meaning letting Db2 implicitly define/create the objects.

Things can also be easy (2)

- If we really like to obtain DB.TS naming convention
  - Explicitly create HSBCDB.HSBCIMP2 (no SET CURRENT RULES, so DB2 used)
  - We now have some work to do

```
CREATE TABLE RASST02.HSBCIMP2
        (COUNTRY VARCHAR(50)
        ,CITY VARCHAR(30)
        ,ZIPCODE CHARACTER(10)
        ,RESIDENTS INTEGER
        ,LOBDATA CLOB(2M) INLINE LENG
        ,LOB_ROWID ROWID GENERATED AI
PARTITION BY RANGE
        (COUNTRY NULLS LAST ASC)
        ( PARTITION 1 ENDING ('G')    INCLUSIVE
        , PARTITION 2 ENDING ('J')    INCLUSIVE
        , PARTITION 3 ENDING ('M')    INCLUSIVE
        , PARTITION 4 ENDING ('ST')   INCLUSIVE
        , PARTITION 5 ENDING ('T')    INCLUSIVE
        , PARTITION 6 ENDING ('ZZZ')  INCLUSIVE )
IN HSBCDB.HSBCIMP2;
```

```
 DB2 Object ===> T                    Option  ===> DI   Where => N
 Table Name ===> HSBCIMP2           > Creator ===> RASST02              >
  Qualifier ===> *                  > N/A     ===> *                    >
Loc: LOCAL ---------- SSID: D12A ----------RASST02 -        LINE 1 OF 3   >
CMD      NAME     CREATOR   DNAME              DCREATOR  TYPE
-------- HSBCIMP2 RASST02   HSBCIMP2           RASST02   TABLE
                            * TABLE SPACE    1
--------                    HSBCIMP2           HSBCDB    TABLE SPACE
****************************** BOTTOM OF DATA ******************************
```

All AUX objects are missing – 18 CREATE statements to do …..

As mentioned earlier, we can maintain our desired naming convention by explicitly creating everything, so if the table is created in an explicit database and tablespace we need to manually/explicitly create 18 objects (6 partitions each having an AUX tablespace, table and index).

Things can also be easy (3)

- Same scenario (explicit DB.TS) but using CURRENT RULES='STD'

```
SET CURRENT RULES = 'STD';
CREATE TABLE RASST02.HSBCIMP3
        (COUNTRY VARCHAR(50)
        ,CITY VARCHAR(30)
        ,ZIPCODE CHARACTER(10)
        ,RESIDENTS INTEGER
        ,LOBDATA CLOB(2M) INLINE LENGTH 0
        ,LOB_ROWID ROWID GENERATED ALWAYS )
PARTITION BY RANGE
        (COUNTRY NULLS LAST ASC)
        ( PARTITION 1 ENDING ('G')   INCLUSIVE
        , PARTITION 2 ENDING ('J')   INCLUSIVE
        , PARTITION 3 ENDING ('M')   INCLUSIVE
        , PARTITION 4 ENDING ('ST')  INCLUSIVE
        , PARTITION 5 ENDING ('T')   INCLUSIVE
        , PARTITION 6 ENDING ('ZZZ') INCLUSIVE )
IN HSBCDB.HSBCIMP3;
```

Same scenario with a 6-partitioned PBR and one LOB column – but tablespace and database created explicitly.

The only difference is we're using CURRENT RULES = 'STD'

Things can also be easy (4)

- Naming convention maintained for DB and base TS (*no DSNnnnnn*)
  - Object status considered COMPLETE by Db2

```
RQTDI          ----------- RC/Q Table Drop Impact ------------ 2023/06/28 11:37
COMMAND ===>                                                   SCROLL ===> CSR

Loc: LOCAL ---------- SSID: D12A ----------RASST02 -        LINE 01 OF 23    >
CMD       NAME      CREATOR   DNAME                DCREATOR  TYPE
-------- HSBCIMP3 RASST02    HSBCIMP3             RASST02   TABLE
                              * TABLE SPACE    7
                              * TABLE          6
                              * INDEX          6
--------                      HSBCIMP3             HSBCDB    TABLE SPACE
--------                      LXYTYC4K             HSBCDB    TABLE SPACE
--------                      LXYTYJL3             HSBCDB    TABLE SPACE
--------                      LXYTYLDJ             HSBCDB    TABLE SPACE
--------                      LXYTYSNZ             HSBCDB    TABLE SPACE
--------                      LXYTY05L             HSBCDB    TABLE SPACE
--------                      LXYTY8LX             HSBCDB    TABLE SPACE
--------                      HSBCILOBDAXYTYGKGI   RASST02   TABLE
--------                      HSBCILOBDAXYTYNEOF   RASST02   TABLE
--------                      HSBCILOBDAXYTYOEEC   RASST02   TABLE
--------                      HSBCILOBDAXYTYVWJD   RASST02   TABLE
--------                      HSBCILOBDAXYTYYP3C   RASST02   TABLE
--------                      HSBCILOBDAXYTY439C   RASST02   TABLE
--------                      IHSBCILOBDAXYTYH2O   RASST02   INDEX
--------                      IHSBCILOBDAXYTYOTX   RASST02   INDEX
--------                      IHSBCILOBDAXYTYPXO   RASST02   INDEX
--------                      IHSBCILOBDAXYTYVPP   RASST02   INDEX
--------                      IHSBCILOBDAXYTYZO9   RASST02   INDEX
--------                      IHSBCILOBDAXYTY4IU   RASST02   INDEX
****************************** BOTTOM OF DATA ******************************
```

A lot less typing and naming convention partial maintained : Since the PBR table was created in an explicit database/tablespace, all the AUX objects remain in the same database.

IDUG

Leading the Db2 User Community for 35 Years

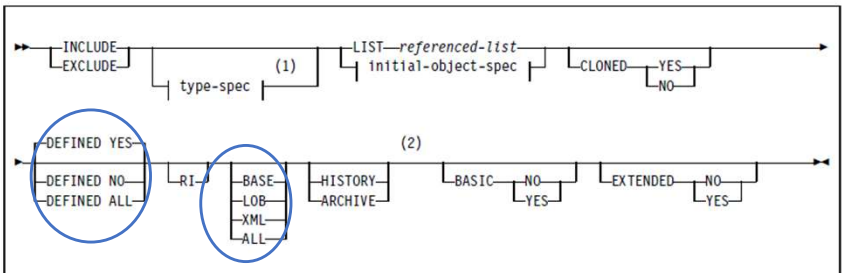
LISTDEF Processing Considerations

#IDUGDb2

LISTDEF Processing(1)

- Syntax from IBM Db2 Utility Guide – focus on two keywords.

list-options:



Notes:

1 You must specify *type-spec* if you specify DATABASE.

2 HISTORY or ARCHIVE can be specified either before or after the BASE, LOB, XML, and ALL keywords. However, HISTORY and ARCHIVE are always processed last, after all other keywords are handled.

LISTDEF Processing(2)

- Partial wildcarding on Tablespace name can be challenging – no control of Implicit Tablespace names

TEN TABLESPACES :

PBG start = 2 parts.
2 LOBs = 4 pagesets.
ADD part = 1 pageset.
ADD LOB for 3 PBG parts = 3 pagesets.

Initially PBG 2-PART and 2x2 AUX explicitly created.

```
DNAME                  DCREATOR  TYPE
IDUGEU23               RASST02   DATA BASE
* TABLE SPACE    10
* TABLE          10
* INDEX          10
BASELOBA               IDUGEU23  TABLE SPACE
BASELOBB               IDUGEU23  TABLE SPACE
BASELOBC               IDUGEU23  TABLE SPACE
BASELOBD               IDUGEU23  TABLE SPACE
BASEPBG                IDUGEU23  TABLE SPACE
L0HIMDZT               IDUGEU23  TABLE SPACE
L0HIMML4               IDUGEU23  TABLE SPACE
L0HINFZA               IDUGEU23  TABLE SPACE
L0HINNXU               IDUGEU23  TABLE SPACE
L0HINV69               IDUGEU23  TABLE SPACE
BASELOBA_PART1_CLO>    RASST02   TABLE
BASELOBB_PART2_CLO>    RASST02   TABLE
```

EXPLICIT AUX DEFINE YES

IMPLICIT AUX DEFINE NO and new part

Depending on how you want to utilize LISTDEF – wildcarding can be difficult if done on the tablespace level due to the naming convention.
In this case the DB.TS was specified EXPLICITLY so not a huge challenge compared to these objects being created implicitly.
Also pay attention to the mix of DEFINE YES/NO tablespaces.

```
TABLE NAME        TBLSPACE
BASELOBA_PART1_CL> BASELOBA
BASELOBB_PART2_CL> BASELOBB
BASELOBC_PART1_CL> BASELOBC
BASELOBD_PART2_CL> BASELOBD
BASEPBG_TAB       BASEPBG
BASEPCOL050HIMIBKY L0HIMDZT
BASEPCOL070HIMQ8AD L0HIMML4
BASEPCOL080HINJRGD L0HINFZA
BASEPCOL080HINRZBY L0HINNXU
BASEPCOL080HINYG6N L0HINV69
```

LISTDEF Processing(3)

- Why do we have TEN tablespaces ?
  - PBG NUMPARTS 2 = 1 base tablespace
  - 2 LOB cols and 2 partitions = 4 AUX tablespaces
  - ADD PART – since 2 LOBs = 2 AUX tablespaces
  - ADD LOB col – since 3 partitions = 3 AUX tablespaces

LISTDEF Processing(4)

- Only DEFINED objects picked up
  - In this use case only explicitly and defined included in LISTDEF
  - Two LOB columns in this two part PBG are the only explicitly defined and DEFINE YES

```
OPTIONS PREVIEW

LISTDEF LIST1 INCLUDE TABLESPACE IDUGEU23.*
```

```
LISTDEF LIST1 -- 00000005 OBJECTS
   INCLUDE TABLESPACE IDUGEU23.BASELOBA
   INCLUDE TABLESPACE IDUGEU23.BASELOBB
   INCLUDE TABLESPACE IDUGEU23.BASELOBC
   INCLUDE TABLESPACE IDUGEU23.BASELOBD
   INCLUDE TABLESPACE IDUGEU23.BASEPBG
```

Even though we wildcard on the tablespace level – including all tablespaces in the DATABASE, only the EXPLICITLY defined tablespaces already instantiated are picked up.
So basically we're missing 5 tablespaces – let's see how to handle these……

LISTDEF Processing(5)

- LISTDEF with keyword DEFINED NO illustrates this
  - Here instantiated pagesets NOT picked up

```
LISTDEF LIST1 INCLUDE TABLESPACE IDUGEU23.* ALL DEFINED NO
```

```
LISTDEF LIST1 -- 00000005 OBJECTS
  INCLUDE TABLESPACE IDUGEU23.L0HIMDZT
  INCLUDE TABLESPACE IDUGEU23.L0HIMML4
  INCLUDE TABLESPACE IDUGEU23.L0HINFZA
  INCLUDE TABLESPACE IDUGEU23.L0HINNXU
  INCLUDE TABLESPACE IDUGEU23.L0HINV69
```

If we INCLUDE tablespaces NOT DEFINED – then we don't get the instantiated and implicitly defined tablespaces.

LISTDEF Processing(6)

- Combining LOB & DEFINED ALL
  - Will process all AUX objects – but not BASE PBG

```
OPTIONS PREVIEW

LISTDEF LIST1 INCLUDE TABLESPACE IDUGEU23.* LOB DEFINED ALL
```

```
LISTDEF LIST1 -- 00000009 OBJECTS
  INCLUDE TABLESPACE IDUGEU23.BASELOBA
  INCLUDE TABLESPACE IDUGEU23.BASELOBB
  INCLUDE TABLESPACE IDUGEU23.BASELOBC
  INCLUDE TABLESPACE IDUGEU23.BASELOBD
  INCLUDE TABLESPACE IDUGEU23.L0HIMDZT
  INCLUDE TABLESPACE IDUGEU23.L0HIMML4
  INCLUDE TABLESPACE IDUGEU23.L0HINFZA
  INCLUDE TABLESPACE IDUGEU23.L0HINNXU
  INCLUDE TABLESPACE IDUGEU23.L0HINV69
```

This is cool – if we specify both LOB and DEFINED ALL, we get everything but the BASE TABLESPACE – next page to get the syntax picking up everything – if that's what you need.

LISTDEF Processing(7)

- Two options to include everything:
  - You have to use ALL instead of LOB – then only one LISTDEF needed
  - OR use two INCLUDEs as below

```
OPTIONS PREVIEW

LISTDEF LIST1 INCLUDE TABLESPACE IDUGEU23.*     DEFINED ALL
              INCLUDE TABLESPACE IDUGEU23.* LOB DEFINED ALL
```

```
LISTDEF LIST1 -- 00000010 OBJECTS
   INCLUDE TABLESPACE IDUGEU23.BASELOBA
   INCLUDE TABLESPACE IDUGEU23.BASELOBB
   INCLUDE TABLESPACE IDUGEU23.BASELOBC
   INCLUDE TABLESPACE IDUGEU23.BASELOBD
   INCLUDE TABLESPACE IDUGEU23.BASEPBG
   INCLUDE TABLESPACE IDUGEU23.L0HIMDZT
   INCLUDE TABLESPACE IDUGEU23.L0HIMML4
   INCLUDE TABLESPACE IDUGEU23.L0HINFZA
   INCLUDE TABLESPACE IDUGEU23.L0HINNXU
   INCLUDE TABLESPACE IDUGEU23.L0HINV69
```

You have basically two options:

Either have two INCLUDES – one for the BASE objects and one for the LOB's whether these are instantiated or not using DEFINED ALL.

If you want everything, use ALL instead of LOB and DEFINED ALL

IDUG

Leading the Db2 User Community for 35 Years

Challenges to Consider When Extracting DDL from the Catalog – Understand Your Tooling of choice

#IDUGDb2

Extract / Generate DDL from the Catalog(1)

- Not an issue when everything explicitly defined.
- Challenge when implicitly or mixed implicitly/explicitly.
  - IDUGEU23TB3 table was created without IN DB / IN DB.TS
  - Not possible to "use the same object names"
  - Tooling (incl. your own) have to "THINK" – comment out implicit objects.
  - Depending on your environment – might be necessary to modify prior to execution.

```
--     CREATE DATABASE DSN01572
--            BUFFERPOOL BP1 INDEXBP    BP2
--            STOGROUP SYSDEFLT;
--
--     CREATE TABLESPACE IDUGEU23
--            USING STOGROUP SYSDEFLT
--                PRIQTY -1 SECQTY -1
--            MAXPARTITIONS 256 NUMPARTS 1;

   CREATE TABLE RASST02.IDUGEU23TB3
       (DEPTNO CHARACTER(3) FOR SBCS DATA    NOT NULL
       ,DEPTNAME VARCHAR(36) FOR SBCS DATA   NOT NULL
       ,MGRNO CHARACTER(6) FOR SBCS DATA
       ,ADMRDEPT CHARACTER(3) FOR SBCS DATA  NOT NULL
       ,LOCATION CHARACTER(16) FOR SBCS DATA
       ,SDEPTNO CHARACTER(4) FOR SBCS DATA
       ,CONSTRAINT DEPTNO PRIMARY KEY   (DEPTNO)  );
--
--     CREATE UNIQUE INDEX RASST02.IDUGEU23_#_AXN
--            ON RASST02.IDUGEU23TB3
--           (DEPTNO ASC)
--            PIECESIZE 4194304K;
```

One topic to consider when IMPLICIT objects exist in your environment is how to generate DDL – especially when we're talking about migrating the DDL to other environments (like from test to systems test and production etc.)

If the database and tablespace were created implicitly, you can't really apply meaningful object names matching your naming convention. Many Db2 sites are using the same DB names and TS names in the various environments, but this will be pure luck for implicit names.

You have to think about this from the tooling perspective as well – what are your options and how do you want to handle these.
The tooling in this case COMMENTS OUT the IMPLICIT objects in order to "MIRROR" the source environments, so you do have the option to remove the comments and then manually modify the implicit names to match your naming convention – at least you have ALL THE DDL so using CHANGE ALL might be a valid path.

Extract / Generate DDL from the Catalog(2)

- Think of the PBG tablespace used earlier
  - Object details
    - PBG MAXPARTITIONS 2 NUMPARTS 2
    - Two CLOB columns
    - Explicitly defined :
      - 1 DB
      - 1 PBG TS
      - 4 LOB TS (2 LOBS x 2 PARTITIONS)
      - 1 BASE TB (and potentially base indexes)
      - 4 AUX tables
      - 4 AUX indexes
  - Then a PBG partition was added -> Two LOB tablespaces added IMPLICITLY.
  - Nice mix of Implicit/Explicit objects ……. *(recommended solution next page)*

One major "pain point" to consider is when you have a mix of IMPLICIT and EXPLICIT objects – you might end up with INVALID DDL.
Let's look at one of the previous use cases covered:

We started with a PBG with TWO partitions and two LOB columns. Everything was explicitly defined.
We then added a third PBG partition resulting in two AUX tablespaces created implicitly.

You can't really create the TARGET DDL in the exact same way since the tablespaces can't be created from scratch using a mix – if you want the exact same look and feel – you will have to follow the exact same steps taken earlier – not really a great idea.
Instead there's a better way to handle this ……. Next page !

Extract / Generate DDL from the Catalog

- DDL is invalid if mixing IMPLICIT/EXPLICIT created objects.
- Don't specify NUMPARTS -> one is defined at creation time.
- When LOAD/INSERT needs another partition – Db2 will grow dynamically using implicit objects.

- Schema synchronization might be a challenge.
  - Naming convention mapping can't be done.
  - Table's tablespace mapping – you probably will have to live with different names *(often tablespace names are identical)*.
- Why not use "profile" to specify object names ?
  - Maybe a promise from IBM at IDUG EMEA 2023 ……

My recommended approach is to OVERRIDE the NUMPARTS to be ONE.
You might not have sufficient storage/space to hold the data in case you are migrating both DDL and DATA – BUT – once data is inserted or loaded, Db2 will dynamically increase the PBG partitions – just make sure MAXPARTITIONS don't mess up this case.

Thank You

Any Additional Questions or Comments ?

IDUG

2024 NA Db2 Tech Conference

@IDUGDb2
#IDUG_NA24

IDUG

Implicitly or Explicitly Defined Db2 Objects – the Good, the Bad and the Ugly

Steen Rasmussen

steen.rasmussen@Broadcom.com

Session Code SQL1

Please fill out your session evaluation!