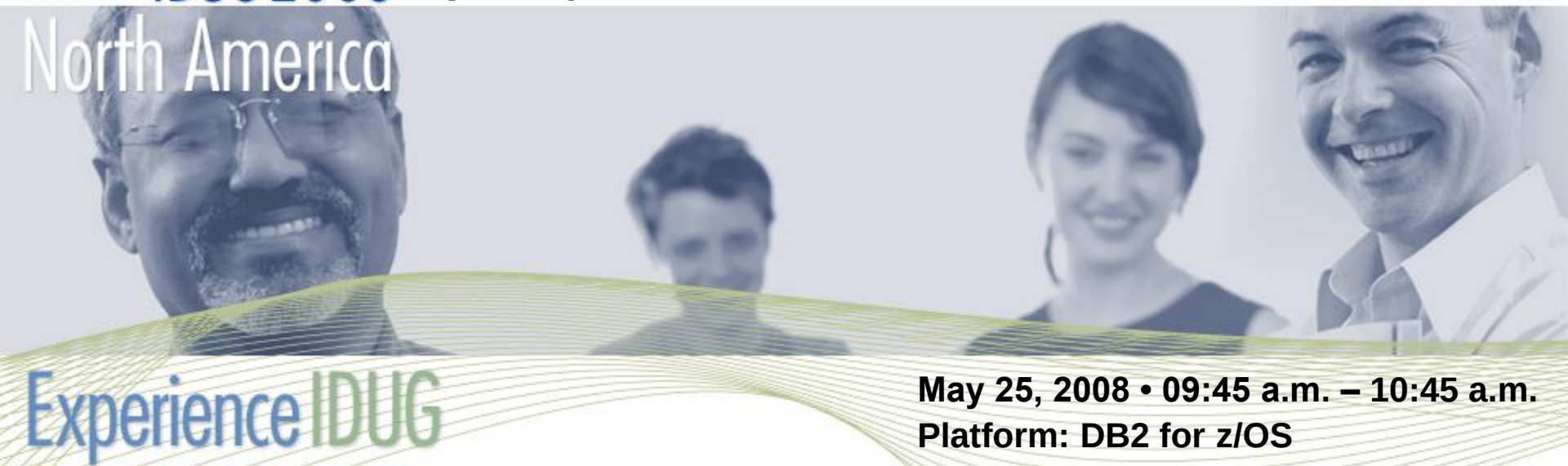
Session: A09

The Clones Have Landed – Watch Out !

Steen Rasmussen
CA inc.



May 25, 2008 • 09:45 a.m. – 10:45 a.m.
Platform: DB2 for z/OS

Abstract

This presentation will describe the possibilities and limitations exploiting Cloned tables in DB2 9 for z/OS.

The new SQL syntax and statements associated with Cloned Tables will be covered by using a live DB2 9 system and real data to illustrate the changes.

Since some Utilities operate on the table/index level and some operate on the tablespace level – the impact on each Utility will be covered as well as the new keywords and commands – again by using a live DB2 environment to illustrate the implications.

Finally - the catalog changes and new parameters will be covered when utilizing Cloned tables.

Agenda

- What is a Cloned Table, where can it be useful and what are the limitations
- DDL Syntax Changes and SQL impact exploiting Cloned tables
- Utility changes and considerations when executing utilities against Cloned tables
- Catalog Changes including attributes when utilizing Cloned tables
- Recommendations and experiences based on a case study using Cloned tables

- Inconsistencies between this presentation and the documentation might exist since a “REAL DB2” used

What is a CLONE ?

- **From GOOGLE search:**

- Cloning, in horticulture and biology, any organism whose genetic information is identical to that of a "mother organism" from which it was created.
- clone (function), various functions in computer programming including:
  - clone (Linux system call), a Linux system call that duplicates a thread.
  - clone (Java method), a Java function that duplicates an object.

Why do we want to Clone ?

- **Another GOOGLE search (modified):**
  - **Why Clone?**
  - Research advances over the past decade have told us that, with a **little work**, we humans can clone just about anything we want, from frogs to monkeys and probably even ourselves – **AND NOW EVEN DB2 TABLES !** So, we can clone things, but why would we want to?
  - **Cloning animal models of disease**
  - Much of what researchers learn about human disease comes from studying animal models such as mice. Often, animal models are genetically engineered to carry disease-causing mutations in their genes. Creating these transgenic animals is a time-intensive process that requires trial-and-error and several generations of breeding. **Cloning technologies** might **reduce the time needed to make a transgenic** animal **model**, and the **result would be a population of genetically identical** animals for study.

DB2 Clone Definition

- A table with the exact same attributes as a table that already exists at the current server.
    - The clone is created in the same table space as the base table.
    - The clone is structurally identical to the base table in every way.
    - Same indexes, before triggers, and LOB objects.
    - Clone tables can only be created in range-partitioned or partition-by-growth table space that is managed by DB2 (aka Universal Tablespace).

DB2 for z/OS Clones

- Which challenge can cloned tables address
  - Basically an “Online Load Replace” without the outage /data being inaccessible
  - Online “Staging Tables” can be handled better
  - Huge Update jobs causing concurrency issues
    - So far DB2 users have implemented “alternate” methods in order to minimize the outage
    - CREATE TABLE yyy LIKE xxx
    - DML / LOAD to populate table
    - (Log Tool to apply DML activity)
    - RENAME “OLD” table
    - RENAME “NEW” table
    - Views in programs to avoid program changes
    - REBIND which could be a challenge

Limitations / Restrictions

- DSNDB06, DSNDB01 cannot be cloned
- RTS tables neither
- No VCAT defined tablespace
- Can not Clone a Clone
- FASTSWITCH not allowed on either object
- Index cannot be created on Clone – adopted from base table like other objects
- No RENAME table
- No RI allowed
- Has to be a UTS (Universal Table Space) SQL-148
- No involvement of MQT's
- No AFTER Triggers
- No Online Schema Changes – only one active version (or other ALTERS)
- No temporary table (declared temporary or global created or DSNDB07)
- Only one table in the tablespace
- No DEFINE NO – every VSAM dataset including indexes must exist prior to adding the clone

What is Cloned/Not Cloned

- Table definition
- Column attributes
- Check Constraints
- Indexes
- Authorizations
- Before trigger
- LOB objects
- Views
- Identity column

Create UTS

- Universal Tablespace is mandatory in order to Clone

```
CREATE  TABLESPACE IDUGUTS1
 IN IDUGDB01
 USING STOGROUP SYSDEFLT
 ERASE NO FREEPAGE 0 PCTFREE 5
 BUFFERPOOL BP1
 LOCKSIZE ANY
 SEGSIZE 64 MAXPARTITIONS 4
 LOCKMAX SYSTEM
 CCSID EBCDIC ;

DSNT400I SQLCODE = 000
```

```
CREATE  TABLESPACE IDUGUTS2
 IN IDUGDB01
 USING STOGROUP SYSDEFLT
 ERASE NO FREEPAGE 0 PCTFREE 5
 BUFFERPOOL BP1
 LOCKSIZE ANY
 SEGSIZE 64 NUMPARTS 4
 LOCKMAX SYSTEM
 CCSID EBCDIC ;

DSNT400I SQLCODE = 000
```

Partition by GROWTH

```
D91A.DSNDBD.IDUGDB01.IDUGUTS1.I0001.A001

D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0001.A001
D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0001.A002
D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0001.A003
D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0001.A004
```

Partition by RANGE

Create Base Tables

- The base tables are created and a user-id granted

```
CREATE  TABLE RASST02.IDUGTB1
(COL01 CHAR ( 4 ) NOT NULL
                  WITH DEFAULT
,COL02 INTEGER    NOT NULL
                  WITH DEFAULT
)
IN IDUGDB01.IDUGUTS1
DATA CAPTURE CHANGES   ;
```

Partition by GROWTH (use **PARTITION BY SIZE** on create table to describe when new part added

Partition by RANGE

```
CREATE  TABLE RASST02.IDUGTB2
(COL01 CHAR ( 4 )     NOT NULL
                      WITH DEFAULT
                      FOR SBCS DATA
,COL02 INTEGER        NOT NULL
                      WITH DEFAULT
,COL03 CLOB ( 20 K )  NOT NULL
                      WITH DEFAULT
,COL04 ROWID
       GENERATED ALWAYS NOT NULL  )
partition by range  (col01 nulls last asc)
(partition    1  ending at  ('CCCC'),
 partition    2  ending at  ('KKKK'),
 partition    3  ending at  ('SSSS'),
 partition    4  ending at  ('ZZZZ') )
IN IDUGDB01.IDUGUTS2
CCSID  EBCDIC  ;
```

```
GRANT SELECT     ON TABLE RASST02.IDUGTB1    TO STEEN ;
GRANT ALL        ON TABLE RASST02.IDUGTB2    TO STEEN ;
```

Create AUX objects

- Auxiliary tables and indexes also created

```
CREATE AUXILIARY TABLE
RASST02.IDUGAUXTB_1
       IN IDUGDB01.IDUGAUX1
       STORES RASST02.IDUGTB2
       COLUMN COL03 PART  1 ;
CREATE AUXILIARY TABLE
RASST02.IDUGAUXTB_2
       IN IDUGDB01.IDUGAUX2
       STORES RASST02.IDUGTB2
       COLUMN COL03 PART  2 ;
```

```
CREATE AUXILIARY TABLE
RASST02.IDUGAUXTB_3
       IN IDUGDB01.IDUGAUX3
       STORES RASST02.IDUGTB2
       COLUMN COL03 PART  3 ;
CREATE AUXILIARY TABLE
RASST02.IDUGAUXTB_4
       IN IDUGDB01.IDUGAUX4
       STORES RASST02.IDUGTB2
       COLUMN COL03 PART  4 ;
```

INDEX NAME	**TABLE NAME**	**COLUMN NAME**
INDEX NAME	TABLE NAME	COLUMN NAME
IDUGTB1_IX0	IDUGTB1	COL01
IDUGTB2_AUX1	IDUGAUXTB_1	AUXID
		AUXVER
IDUGTB2_AUX2	IDUGAUXTB_2	AUXID
		AUXVER
IDUGTB2_AUX3	IDUGAUXTB_3	AUXID
		AUXVER
IDUGTB2_AUX4	IDUGAUXTB_4	AUXID
		AUXVER
IDUGTB2_IX1	IDUGTB2	COL04
IDUGTB2_IX2	IDUGTB2	COL02

Create Views for base objects

- One view created for each of the two base tables

```
CREATE  VIEW RASST02.IDUGTB1_VIEW
            ( COL01 , COL02 ) AS
SELECT   COL01 , COL02
FROM     RASST02.IDUGTB1 ;

CREATE  VIEW RASST02.IDUGTB2_VIEW
            ( COL03 ) AS
SELECT   COL03
FROM     RASST02.IDUGTB2  ;
```

- It is possible to have different views pointing to Base and Clone tables

VSAM datasets prior to clone

- All VSAM datasets exist as expected

```
Command - Enter "/" to select action                    Tracks %Used XT  Device
-------------------------------------------------------------------------------
         D91A.DSNDBD.IDUGDB01.IDUGAUX1.I0001.A001          150    ?   1  3390
         D91A.DSNDBD.IDUGDB01.IDUGAUX2.I0001.A001          150    ?   1  3390
         D91A.DSNDBD.IDUGDB01.IDUGAUX3.I0001.A001          150    ?   1  3390
         D91A.DSNDBD.IDUGDB01.IDUGAUX4.I0001.A001          150    ?   1  3390
         D91A.DSNDBD.IDUGDB01.IDUGTB1R.I0001.A001           15    ?   1  3390
         D91A.DSNDBD.IDUGDB01.IDUGTB2R.I0001.A001           15    ?   1  3390
         D91A.DSNDBD.IDUGDB01.IDUGUTS1.I0001.A001           15    ?   1  3390
         D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0001.A001           15    ?   1  3390
         D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0001.A002           15    ?   1  3390
         D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0001.A003           15    ?   1  3390
         D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0001.A004           15    ?   1  3390
         D91A.DSNDBD.IDUGDB01.IDUG1MY4.I0001.A001           15    ?   1  3390
         D91A.DSNDBD.IDUGDB01.IDUG1OZ3.I0001.A001           15    ?   1  3390
         D91A.DSNDBD.IDUGDB01.IDUG1XG5.I0001.A001           15    ?   1  3390
         D91A.DSNDBD.IDUGDB01.IDUG11R6.I0001.A001           15    ?   1  3390
         D91A.DSNDBD.IDUGDB01.IDUG17F5.I0001.A001           15    ?   1  3390
****************************** End of Data Set list ******************************
```

Create Clone Table

- Prior to clone – data inserted into base tables

CREATOR	TABLE	COUNT
RASST02	IDUGTB1	15
RASST02	IDUGTB2	6
RASST02	IDUGTB1_VIEW	15
RASST02	IDUGTB2_VIEW	6

- Time to CLONE

```
ALTER TABLE RASST02.IDUGTB1 ADD CLONE
STEEN.IDUGTB1_CLONE ;

ALTER TABLE RASST02.IDUGTB2 ADD CLONE
STEEN.IDUGTB2_CLONE ;
```

Create Clone Table

- VSAM datasets after cloning

```
D91A.DSNDBD.IDUGDB01.IDUGAUX1.I0001.A001
D91A.DSNDBD.IDUGDB01.IDUGAUX1.I0002.A001
D91A.DSNDBD.IDUGDB01.IDUGAUX2.I0001.A001
D91A.DSNDBD.IDUGDB01.IDUGAUX2.I0002.A001
D91A.DSNDBD.IDUGDB01.IDUGAUX3.I0001.A001
D91A.DSNDBD.IDUGDB01.IDUGAUX3.I0002.A001
D91A.DSNDBD.IDUGDB01.IDUGAUX4.I0001.A001
D91A.DSNDBD.IDUGDB01.IDUGAUX4.I0002.A001
D91A.DSNDBD.IDUGDB01.IDUGTB1R.I0001.A001
D91A.DSNDBD.IDUGDB01.IDUGTB1R.I0002.A001
D91A.DSNDBD.IDUGDB01.IDUGTB2R.I0001.A001
D91A.DSNDBD.IDUGDB01.IDUGTB2R.I0002.A001
D91A.DSNDBD.IDUGDB01.IDUGUTS1.I0001.A001
D91A.DSNDBD.IDUGDB01.IDUGUTS1.I0002.A001
D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0001.A001
D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0001.A002
D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0001.A003
D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0001.A004
D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0002.A001
D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0002.A002
D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0002.A003
D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0002.A004
```

```
D91A.DSNDBD.IDUGDB01.IDUG1MY4.I0001.A001
D91A.DSNDBD.IDUGDB01.IDUG1MY4.I0002.A001
D91A.DSNDBD.IDUGDB01.IDUG1OZ3.I0001.A001
D91A.DSNDBD.IDUGDB01.IDUG1OZ3.I0002.A001
D91A.DSNDBD.IDUGDB01.IDUG1XG5.I0001.A001
D91A.DSNDBD.IDUGDB01.IDUG1XG5.I0002.A001
D91A.DSNDBD.IDUGDB01.IDUG11R6.I0001.A001
D91A.DSNDBD.IDUGDB01.IDUG11R6.I0002.A001
D91A.DSNDBD.IDUGDB01.IDUG17F5.I0001.A001
D91A.DSNDBD.IDUGDB01.IDUG17F5.I0002.A001
```

- Fifth qualifier / node name / INSTANCE reflects the cloned VSAM datasets
- Every I0001 spacename replicated to I0002

Create Clone Table

- What happened to the catalog
  - Five catalog tables touched
    - SYSTABLESPACESTATS
      - Nine inserts
      - INSTANCE=2 identifies cloned objects since DBID and PSID are identical
    - SYSINDEXSPACESTATS
      - Seven inserts
      - INSTANCE=2 identifies cloned objects
    - SYSTABLES
      - Twelve updates and 2 inserts
      - TYPE='C' for cloned tables, RUNSTATS info set to -1
    - SYSCOLUMNS
      - Six inserts and RUNSTATS info set to -1
    - SYSTABLESPACE
      - Six updates
      - CLONE='Y' but INSTANCE=1 since base is still the "active table"

Catalog Activity	
Updates	18
Deletes	0
Inserts	24
Elapsed time	37 secs

Exchange

- EXCHANGE command will point the table to the “other” instance / different VSAM dataset

```
EXCHANGE DATA BETWEEN TABLE RASST02.IDUGTB1 AND STEEN.IDUGTB1_CLONE;

EXCHANGE DATA BETWEEN TABLE STEEN.IDUGTB1_CLONE AND RASST02.IDUGTB1;

EXCHANGE DATA BETWEEN TABLE RASST02.IDUGTB2 AND STEEN.IDUGTB2_CLONE;
```

	BEFORE EXCHANGE	AFTER EXCHANGE
SELECT COUNT(*) FROM RASST02.IDUGTB1	15	0
SELECT COUNT(*) FROM RASST02.IDUGTB2	6	0
SELECT COUNT(*) FROM RASST02.IDUGTB1_VIEW	15	0
SELECT COUNT(*) FROM RASST02.IDUGTB2_VIEW	6	0
SELECT COUNT(*) FROM STEEN.IDUGTB1_CLONE	0	15
SELECT COUNT(*) FROM STEEN.IDUGTB2_CLONE	0	6

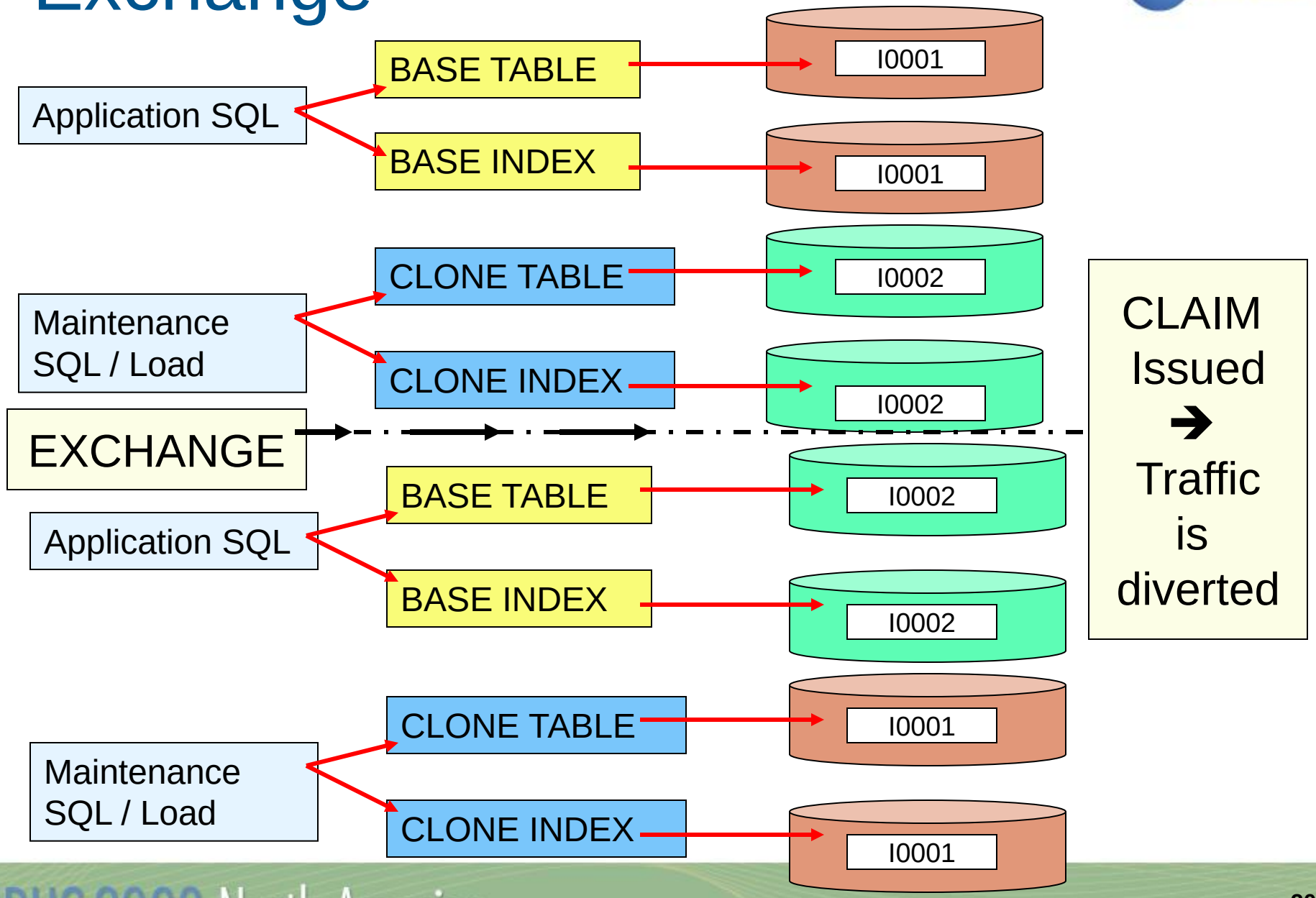
  - Transparent for Applications and SQL – no changes needed to get “new data”
  - The views follow the table and the pageset

Exchange

- Unlike OSE – no plans or packages are invalidated
- Statement cache not flushed
  (RUNSTATS executed for the base is valid for the clone too – covered later)
- No QUIESCE is taken
- CLAIM used to stop traffic and lead it in the other direction (the other pageset)

  - Very simple process
  - Utility implications covered later

Exchange


Application SQL
BASE TABLE
BASE INDEX
I0001
I0001
Maintenance SQL / Load
CLONE TABLE
CLONE INDEX
I0002
I0002
EXCHANGE
CLAIM Issued
➔
Traffic is diverted
Application SQL
BASE TABLE
BASE INDEX
I0002
I0002
Maintenance SQL / Load
CLONE TABLE
CLONE INDEX
I0001
I0001

Exchange

- What happened to the catalog
  - Five catalog tables touched
    - SYSTABLESPACESTATS
      - Nine inserts
    - SYSINDEXSPACESTATS
      - Seven inserts
    - SYSTABLES
      - Twelve updates and 2 inserts
    - SYSCOLUMNS
      - Six inserts
    - SYSTABLESPACE
      - Six updates
      - INSTANCE updated to be 2 in order to point the table to the I0002 datasets, since the clone is the “new active table”

Drop Clone

- Getting rid of the CLONE is very easy

```
ALTER TABLE IDUGTB1 DROP CLONE ;
```

- But it can be dangerous
  - In this case all my data is gone
  - Which might be OK …….

```
SELECT COUNT(*) FROM RASST02.IDUGTB1                0
SELECT COUNT(*) FROM RASST02.IDUGTB2                0
SELECT COUNT(*) FROM RASST02.IDUGTB1_VIEW           0
SELECT COUNT(*) FROM RASST02.IDUGTB2_VIEW           0
SELECT COUNT(*) FROM STEEN.IDUGTB1_CLONE          SQL-204
SELECT COUNT(*) FROM STEEN.IDUGTB2_CLONE            6
```

Drop Clone

- Dropping the CLONE will leave the “ACTIVE” VSAM dataset name and remove the “inactive”
  - In this case I have to live with the I0002 dataset

```
D91A.DSNDBD.IDUGDB01.IDUGUTS1.I0002.A001
D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0001.A001
D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0001.A002
D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0001.A003
D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0001.A004
D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0002.A001
D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0002.A002
D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0002.A003

D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0002.A004
```

  - Consider any SMS routines, DSN1COPY etc.
- Dropping the base table will also drop the clone and remove all datasets

Data Manipulation

- Let's look at data manipulation
- The clone for IDUGTB1 is created again
  - Now the "inactive" dataset has INSTANCE=1 since the "original" dataset prior to the CLONE had INSTANCE=2
  - Always look at SYSTABLESPACE.INSTANCE to see which pageset is the active one
    (DB2 command output is another method – will be covered later)

```
D91A.DSNDBD.IDUGDB01.IDUGUTS1.I0001.A001
D91A.DSNDBD.IDUGDB01.IDUGUTS1.I0002.A001
D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0001.A001
D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0001.A002
D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0001.A003
D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0001.A004
D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0002.A001
D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0002.A002
D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0002.A003
D91A.DSNDBD.IDUGDB01.IDUGUTS2.I0002.A004
```

Data Manipulation

- Data can be inserted, updated, deleted and loaded for both Table types
  - Load operates on the table level (one of the only utilities)
  - Rows can be inserted in both base tables and clones
  - Table content B4 and after EXCHANGE of IDUGTB2

```
SELECT COUNT(*) FROM RASST02.IDUGTB1          : 105
SELECT COUNT(*) FROM RASST02.IDUGTB2          :  93
SELECT COUNT(*) FROM RASST02.IDUGTB1_VIEW     : 105
SELECT COUNT(*) FROM RASST02.IDUGTB2_VIEW     :  93
SELECT COUNT(*) FROM STEEN.IDUGTB1_CLONE      :  10
SELECT COUNT(*) FROM STEEN.IDUGTB2_CLONE      :   6
```

```
SELECT COUNT(*) FROM RASST02.IDUGTB1          : 105
SELECT COUNT(*) FROM RASST02.IDUGTB2          :   6
SELECT COUNT(*) FROM RASST02.IDUGTB1_VIEW     : 105
SELECT COUNT(*) FROM RASST02.IDUGTB2_VIEW     :   6
SELECT COUNT(*) FROM STEEN.IDUGTB1_CLONE      :  10
SELECT COUNT(*) FROM STEEN.IDUGTB2_CLONE      :  93
```

RUNSTATS

- RUNSTATS can only be executed against the BASE objects

```
RUNSTATS TABLESPACE IDUGDB01.IDUGUTS1 INDEX(ALL)
```

- Not against the CLONE

```
RUNSTATS TABLESPACE IDUGDB01.IDUGUTS1
TABLE(STEEN.IDUGTB1_CLONE) COLUMN(ALL)

RUNSTATS UTILITY MAY NOT BE RUN AGAINST CLONE OBJECT
STEEN.IDUGTB1_CLONE
UTILITY EXECUTION TERMINATED, HIGHEST RETURN CODE=8
```

- If “refresh” of data results in very “different data” (influence Access Path selection) – a subsequent RUNSTATS after EXCHANGE might be needed

RUNSTATS

SYSCOLUMNS - BASE TABLE

NAME	COL01
TBNAME	IDUGTB1
TBCREATOR	RASST02
COLCARD	- 1
HIGH2KEY	8JGG
LOW2KEY	BBOG
DEFAULT	Y
STATSTIME	2008-01-20-18.39.20.873393
COLCARDF	+0.16E+02
CREATEDTS	2007-12-12-19.02.20.602604

SYSCOLUMNS - CLONE TABLE

NAME	COL01
TBNAME	IDUGTB1_CLONE
TBCREATOR	STEEN
COLCARD	- 1
HIGH2KEY	
LOW2KEY	
DEFAULT	Y
STATSTIME	0001-01-01-00.00.00.000000
COLCARDF	-0.1E+01
CREATEDTS	2008-01-02-12.52.46.979395

SYSTABLES - BASE TABLE

NAME	IDUGTB1
CREATOR	RASST02
TYPE	T
DBNAME	IDUGDB01
STATUS	
TBCREATOR	STEEN
TBNAME	IDUGTB1_CLONE
STATSTIME	2008-01-20-18.39.20.873393
CARDF	+0.105E+03
TABLESTATUS	
NPAGESF	+0.1E+01
SPACEF	+0.72E+03

SYSTABLES - CLONE TABLE

NAME	IDUGTB1_CLONE
CREATOR	STEEN
TYPE	C
DBNAME	IDUGDB01
STATUS	
TBCREATOR	RASST02
TBNAME	IDUGTB1
STATSTIME	0001-01-01-00.00.00.000000
CARDF	-0.1E+01
TABLESTATUS	
NPAGESF	-0.1E+01
SPACEF	-0.1E+01

Explain

- Explain can be executed against base and clone

```
EXPLAIN PLAN SET QUERYNO=001 FOR
SELECT * FROM STEEN.IDUGTB1_CLONE ORDER BY 1;
DSNT400I SQLCODE = 000,  SUCCESSFUL EXECUTION

SELECT * FROM PLAN_TABLE
QUERYNO                    1
QBLOCKNO                   1
APPLNAME
PROGNAME                   BPASQL8
PLANNO                     1
METHOD                     0
CREATOR                    STEEN
TNAME                      IDUGTB1_CLONE
TABNO                      1
ACCESSTYPE                 I
MATCHCOLS                  0
ACCESSCREATOR              RASST02
ACCESSNAME                 IDUGTB1_IX0
INDEXONLY                  N
```

- Notice the table accessed is the clone
- But index is the one created for the base (Index information shared too)
- Remember only **ONE** set of catalog statistics exists

Online Schema Evolution

- Online Schema Evolution ALTER statements are not supported for objects involved in Clone
- If needed – use this process
  - DROP Clone
  - Execute ALTER
  - ADD Clone
- SQL error messages are getting a lot better

```
ALTER INDEX IDUGTB1_IX0 ADD COLUMN (COL02) ;

DSNT408I SQLCODE = -650, ERROR:  THE ALTER STATEMENT CANNOT BE
                                 EXECUTED, REASON 18
DSNT418I SQLSTATE   = 56090 SQLSTATE RETURN CODE
DSNT415I SQLERRP    = DSNXIAIX SQL PROCEDURE DETECTING ERROR
DSNT416I SQLERRD    = 24  0  0  -1  0  0 SQL DIAGNOSTIC INFORMATION
```

Online Schema Evolution - Messages

-650 THE ALTER STATEMENT CANNOT BE EXECUTED, REASON *reason*
Explanation: The ALTER statement cannot be executed for one of the following reasons:

1 Alter to type 1 index is not allowed for the index whose associated table space has a LOCKSIZE specification of ROW.
2 Alter to type 1 index is not allowed for the index defined with UNIQUE WHERE NOT NULL.
3 Alter to type 1 index is not allowed for the index whose associated table space has been
defined as a LARGE table space.
4 Alter to type 1 index is not allowed for an index on an ASCII table.
5 An ALTER statement with a PIECESIZE clause is not allowed for a partitioning index.
6 An ALTER statement with a PIECESIZE 4G clause is not allowed for non-partitioned indexes on a non-large table.
11 An ALTER statement with an ENDING AT clause is not allowed for an index on a partitioned base table with LOB columns or the table itself.
12 ALTER INDEX is not allowed when there is a pending SQL statement.
13 ALTER TABLE is not allowed when there is a pending SQL statement.
14 An ALTER INDEX statement with an ENDING AT clause is not allowed when using table-controlled partitioning.
15 Using ALTER to change the attributes of a partition.
16 ALTER TABLE cannot be used to drop clone when the table does not have a defined clone.
17 ALTER TABLE cannot be used to drop clone when the table itself is a clone.
18 ALTER INDEX is not allowed when the table has defined a clone.
19 ALTER TABLESPACE is not allowed when it's containing table has a defined clone.

The new and improved messages removes a lot of the guess-work when “invalid” DDL is executed

DDL for Clones/Base

- Create new index has to be for the BASE table

```
CREATE  INDEX RASST02.IDUGTB1_IX_CLONE ON STEEN.IDUGTB1_CLONE
        ( COL02 ASC , COL01 ASC)
          USING STOGROUP SYSDEFLT ERASE NO
          FREEPAGE 0 PCTFREE 10 BUFFERPOOL BP0 CLOSE YES ;

 DSNT408I SQLCODE = -159, ERROR:  THE STATEMENT REFERENCES
          STEEN.IDUGTB1_CLONE WHICH IDENTIFIES A CLONE RATHER THAN A
          TABLE
```

- Doesn't matter which instance is active

```
CREATE  INDEX RASST02.IDUGTB1_IX_CLONE ON RASST02.IDUGTB1
        ( COL02 ASC , COL01 ASC)
          USING STOGROUP SYSDEFLT ERASE NO
          FREEPAGE 0 PCTFREE 10 BUFFERPOOL BP0 CLOSE YES ;
 DSNT404I SQLCODE = 610, WARNING:  A CREATE/ALTER ON OBJECT
          RASST02.IDUGTB1_IX_CLONE CLONE HAS PLACED OBJECT IN REBUILD
          PENDING

IDUG1TYQ IXB2  L*   RW
IDUG1TYQ IXC1  L*   RW,PSRBD
```

DDL for Clones/Base

- The “good old” ALTER statements prior to Online Schema Evolution not supported either
  - Remember the CLONE adopts every attribute
  - ALTER cannot be executed on CLONE

```
ALTER   TABLE RASST02.IDUGTB1
        ADD COL03 INTEGER WITH DEFAULT ;

DSNT408I SQLCODE = -148, ERROR:  THE SOURCE TABLE RASST02.IDUGTB1
                                 CANNOT BE ALTERED. REASON 11


ALTER   TABLE RASST02.IDUGTB1 AUDIT CHANGES;

DSNT408I SQLCODE = -148, ERROR:  THE SOURCE TABLE RASST02.IDUGTB1
                                 CANNOT BE ALTERED. REASON 11
```

  - Again – the SQL-error descriptions are very helpful
- Another ALTER not being supported is enabling COPY for indexes

UTILITIES and CLONE Tables

- LOAD works on the table level – can be executed on both BASE and CLONE
- RUNSTATS only works on the BASE table/index level
- REBUILD INDEX ?
- COPY / COPYTOCOPY / MERGECOPY ?
- RECOVER ?
- MODIFY RECOVERY ?
- REORG tablespace / index ?
- QUIESCE ?
- REPAIR ?
- CHECK DATA / INDEX / LOB ?
- REPORT ?

REBUILD INDEX

- Remember the “new CLONE” index ended up in PSRBD after creating an index on the base
- REBUILD INDEX has a new keyword

```
DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = RASST02.RASST02Y
DSNUGTIS - PROCESSING SYSIN AS EBCDIC
DSNUGUTC - REBUILD INDEX(RASST02.IDUGTB1_IX_CLONE) SORTDEVT SYSDA SORTNUM 3 CLONE
DSNUCRIB - INDEXES WILL BE BUILT IN PARALLEL, NUMBER OF TASKS = 3
DSNUCRUL - UNLOAD PHASE STATISTICS - NUMBER OF RECORDS PROCESSED=10
DSNUCRIB - UNLOAD PHASE COMPLETE - ELAPSED TIME=00:00:00
DSNURBXC - SORTBLD PHASE STATISTICS - NUMBER OF KEYS=10 FOR INDEX
                                      RASST02.IDUGTB1_IX_CLONE

DSNUCRIB - SORTBLD PHASE STATISTICS. NUMBER OF INDEXES = 1
DSNUCRIB - SORTBLD PHASE COMPLETE, ELAPSED TIME = 00:00:01
DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```

- If CLONE keyword not specified – only the base tables index is being rebuilt

UTILITIES and CLONE Tables

- All the remaining mentioned utilities work basically the same way
  - Specify CLONE to execute against clone
    - Utility will only process clone table data.
  - If LIST keyword used to specify a list of objects, only those objects in the list that contain clone tables or indexes on clone tables - other objects in the list are ignored
  - DSN1xxxx utilities will process CLONE objects
    - VSAM dataset (instance number) differentiates as well as PSID for Clones which have high-order-bit ‘1’
  - BUT – let’s look closer at Backup and Recovery

IMAGE COPY

- Output messages don't indicate base or clone

```
COPY TABLESPACE IDUGDB01.IDUGUTS1  DSNUM    ALL
     DEVT SYSDA COPYDDN COPYA001
     FULL YES
     SHRLEVEL REFERENCE
COPY TABLESPACE IDUGDB01.IDUGUTS1  DSNUM    ALL
     DEVT SYSDA COPYDDN COPYA002   CLONE
     FULL YES
     SHRLEVEL REFERENCE

DSNUBBID - COPY PROCESSED FOR TABLESPACE IDUGDB01.IDUGUTS1
                   NUMBER OF PAGES=3
                   AVERAGE PERCENT FREE SPACE PER PAGE = 19.33
                   PERCENT OF CHANGED PAGES =  1.66
                   ELAPSED TIME=00:00:00
DSNUBAFI - DB2 IMAGE COPY SUCCESSFUL FOR TABLESPACE IDUGDB01.IDUGUTS1
DSNUGUTC -  COPY TABLESPACE IDUGDB01.IDUGUTS1 DSNUM ALL DEVT SYSDA COPYDDN
DSNUBBID - COPY PROCESSED FOR TABLESPACE IDUGDB01.IDUGUTS1
                   NUMBER OF PAGES=3
                   AVERAGE PERCENT FREE SPACE PER PAGE = 31.66
                   PERCENT OF CHANGED PAGES =  1.66
                   ELAPSED TIME=00:00:00
DSNUBAFI - DB2 IMAGE COPY SUCCESSFUL FOR TABLESPACE IDUGDB01.IDUGUTS1
```

IMAGE COPY

- Like Online Schema Evolution operations register a lot in SYSCOPY – so does Clone activity

DBNAME	TSNAME	ICTYPE	STYPE	NPAGESF	TTYPE	INSTANCE
IDUGDB01	IDUGUTS1	F		+0.18E+03		1
IDUGDB01	IDUGUTS1	F		+0.18E+03		2
IDUGDB01	IDUGUTS1	A	E	-0.1E+01		2
IDUGDB01	IDUGUTS1	A	E	-0.1E+01		1
IDUGDB01	IDUGUTS1	A	E	-0.1E+01		1
IDUGDB01	IDUGUTS1	A	E	-0.1E+01		2
IDUGDB01	IDUGUTS1	A	E	-0.1E+01		2
IDUGDB01	IDUGUTS1	A	E	-0.1E+01		1
IDUGDB01	IDUGUTS1	A	E	-0.1E+01		2
IDUGDB01	IDUGUTS1	A	E	-0.1E+01		1
IDUGDB01	IDUGUTS1	A	E	-0.1E+01		2

- SYSCOPY instance number illustrates which VSAM dataset has been copied
  - STYPE = E :
    - The data set numbers of a base table and its associated clone table are exchanged.

RECOVERY

- Scenario :
  - Quiesce
  - FULL Image Copy
  - EXCHANGE
  - Recover to PIT prior to EXCHANGE

DBNAME	TSNAME	ICTYPE	TIMESTAMP	STYPE	INSTANCE
IDUGDB01	IDUGUTS1	A	2008-01-21-15.23.18.653277	E	2
IDUGDB01	IDUGUTS1	A	2008-01-21-15.23.18.653277	E	1
IDUGDB01	IDUGUTS1	F	2008-01-21-15.21.43.732620		1
IDUGDB01	IDUGUTS1	F	2008-01-21-15.21.43.626071		2
IDUGDB01	IDUGUTS1	Q	2008-01-21-15.19.42.916289	W	1
IDUGDB01	IDUGUTS1	Q	2008-01-21-15.19.42.874840	W	2

RECOVERY

- Recover CLONE from BASE image copy

```
RECOVER TABLESPACE IDUGDB01.IDUGUTS1 CLONE TOCOPY RASST02.ICBASE.AFTERQU

DSNUCASA - RECOVER CANNOT PROCEED FOR TABLESPACE IDUGDB01.IDUGUTS1
           BECAUSE A SYSIBM.SYSCOPY RECORD HAS BEEN ENCOUNTERED WHICH HAS
           DBNAME=IDUGDB01 TSNAME=IDUGUTS1 DSNUM=0 ICTYPE=A STYPE=E
           STARTRBA=X'000CC3619458' LOWDSNUM=0 HIGHDSNUM=0
DSNUCBDR - RECOVERY COMPLETE, ELAPSED TIME=00:00:00
DSNUGBAC - UTILITY EXECUTION TERMINATED, HIGHEST RETURN CODE=8
```

  - Why is this “legal” ?
    - Image copy taken for BASE now “belongs” to the clone object since EXCHANGE executed after the image copy was taken

RECOVERY

- Recover CLONE from CLONE image copy

```
RECOVER TABLESPACE IDUGDB01.IDUGUTS1 CLONE TOCOPY RASST02.ICCLONE.AFTERQU

DSNUCATO - TOCOPY DATASET NOT FOUND
RECOVERY COMPLETE, ELAPSED TIME=00:00:00
UTILITY EXECUTION TERMINATED, HIGHEST RETURN CODE=8


DATABASE  TSNAME    DSN     OPERATION      DATE       TIME     SEQ   DEVICE   SHR
IDUGDB01  IDUGUTS1  0       FULL COPY   2008/01/21 15:21:43  0     3390      R

DSN          => RASST02.ICCLONE.AFTERQU
VOLSER       =>
START_RBA    => X'000CC3612ACA'
ICBACKUP     =>
ICUNIT       => D
GROUP_MEMBER =>
JOB NAME     => RASST02I
AUTHID       => RASST0
LOGICAL_PART => 0
OLDEST_VER   => 0
```

  - The image copy taken for the clone now “belongs” to the base (different instance number)

RECOVERY

- Recover TORBA - then DB2 cannot use the instance number as the excuse ☺

```
RECOVER TABLESPACE IDUGDB01.IDUGUTS1 TORBA X'000CC35D70BE'

DSNUCASA - RECOVER CANNOT PROCEED FOR TABLESPACE IDUGDB01.IDUGUTS1
           BECAUSE A SYSIBM.SYSCOPY RECORD HAS BEEN ENCOUNTERED WHICH HAS
           DBNAME=IDUGDB01 TSNAME=IDUGUTS1 DSNUM=0 ICTYPE=A STYPE=E
           STARTRBA=X'000CC3619458' LOWDSNUM=0 HIGHDSNUM=0
DSNUCBDR - RECOVERY COMPLETE, ELAPSED TIME=00:00:00
DSNUGBAC - UTILITY EXECUTION TERMINATED, HIGHEST RETURN CODE=8
```

  - It is not possible to recover to a point prior to EXCHANGE
  - Taking an image copy after EXCHANGE might be a good investment
  - Or save the data from the other instance before new manipulation
  - It does NOT help to DROP the clone either ☺

Report Tablespaceset/Recovery

- Both Report utilities provide details for base and clone objects

```
DSNU050I  DSNUGUTC -  REPORT TABLESPACESET TABLESPACE IDUGDB01.IDUGUTS1

TABLESPACE SET REPORT:

TABLESPACE          : IDUGDB01.IDUGUTS1
  TABLE             : RASST02.IDUGTB1
  INDEXSPACE        : IDUGDB01.IDUGTB1R
  INDEX             : RASST02.IDUGTB1_IX0
  INDEXSPACE        : IDUGDB01.IDUG1TYQ
  INDEX             : RASST02.IDUGTB1_IX_CLONE



CLONE TABLESPACE SET REPORT:

TABLESPACE          : IDUGDB01.IDUGUTS1
  CLONE TABLE       : RASST02.IDUGTB1C
  INDEXSPACE        : IDUGDB01.IDUGTB1R
  INDEX             : RASST02.IDUGTB1_IX0
  INDEXSPACE        : IDUGDB01.IDUG1TYQ
  INDEX             : RASST02.IDUGTB1_IX_CLONE
```

MODIFY RECOVERY

- Cleanup of SYSCOPY and SYSLGRNX can be done for clones too – new SYSCOPY entries

```
DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = RASST02.RASST02M
DSNUGTIS - PROCESSING SYSIN AS EBCDIC
DSNUGUTC - MODIFY RECOVERY TABLESPACE IDUGDB01.IDUGUTS1
           DELETE AGE(2) CLONE
DSNUMODA - MODIFY COMPLETED SUCCESSFULLY
```

TSNAME	ICTYPE	STYPE	INSTANCE
IDUGUTS1	A	E	1
IDUGUTS1	A	E	2
IDUGUTS1	F		1
IDUGUTS1	F		2
IDUGUTS1	M	R	2

ICTYPE='M' -> Modify Recovery

SYSIBM.SYSCOPY

When ICTYPE=A, the values are:

A	A partition was added to a table.
C	A column was added to a table and an index in different commit scopes.
E	The data set numbers of a base table and its associated clone table are exchanged.
G	An index was regenerated
L	The logging attribute of the table space was altered to LOGGED.
N	An index was altered to not padded
O	The logging attribute of the table space was altered to NOT LOGGED.
P	An index was altered to padded
R	A table was altered to rotate partitions.
V	A column in a table was altered for a numeric data type change and the column is in an index.
Z	A column that is in the key of an index that was versioned prior to DB2 Version 8 was altered.

When ICTYPE=C, the values are:

L	The logging attribute of the table space was altered to LOGGED.
O	The logging attribute of the table space was altered to NOT LOGGED.

When ICTYPE=F, the values are:

A	ADD PARTITION execution
C	DPSMS concurrent copy ("I" instance of the table space)
J	DPSMS concurrent copy ("J" instance of the table space)
R	ROTATE FIRST TO LAST
S	LOAD REPLACE(NO)
V	ALTER INDEX NOT PADDED
W	REORG LOG(NO)
X	REORG LOG(YES)
blank	DB2 image copy

The MERGECOPY utility, when used to merge an embedded copy with subsequent incremental copies, also produces a record that contains ICTYPE=F and the STYPE of the original image copy (R, S, W, or X).

Study the ICTYPE and STYPE combinations very carefully in order not to be "fooled"

The previous slide has STYPE='R' which could mean ROTATE – but not for ICTYPE='M'

When ICTYPE = M and the MODIFY RECOVERY utility was executed to delete SYSCOPY and/or SYSLGRNX records, the value is R.

When ICTYPE=O, the values are:

R	Reordered format
B	Basic row format

When ICTYPE=P, the values are:

C	Recover to a point in time without using logonly with consistency.
L	Recover to a point in time using logonly without consistency.
M	Recover to a point in time using logonly with consistency.
blank	Recover to a point in time without using logonly without consistency.

When ICTYPE=Q and option WRITE(YES) is in effect when the quiesce point is taken, the value is W.

When ICTYPE=R, S, W, or X and the operation is resetting REORG pending status, the value is A.

When ICTYPE=R, S, W, or X and the operation is first materializing the default value for a row change timestamp column, the value is T.

When ICTYPE=T, this field indicates which COPY utility was terminated by the TERM UTILITY command or the START DATABASE command with the ACCESS(FORCE) option. The values are:

F	COPY FULL YES
I	COPY FULL NO

For other values of ICTYPE, the value is blank.

DB2 Commands

- Ever seen DB2 Command output from TCP where partitions have been rotated and added ?
  - This is just another chapter where it takes a little time to “adjust” and interpret

```
NAME     TYPE PART   STATUS
-------- ---- ----- ----------------

IDUGUTS1 TS    0001 RW

IDUG1TYQ IX    L*   RW
IDUGTB1R IX    L*   RW
```

  - At this point one tablespace with two indexes

DB2 Commands

- IDENTITY column added to table
  - Table went into REORP – REORG executed
  - ALTER TABLE ADD CLONE resulted in SQL-148 REASON=09
  - COPY, REORG and MODIFY DELETE AGE(*) executed
  - Now CLONE is altered / added

```
NAME     TYPE PART  STATUS
-------- ---- ----- -----------------
IDUGUTS1 TSB1  0001 RW,COPY
IDUGUTS1 TSC2  0001 RW
IDUG1TYQ IXB1  L*   RW
IDUG1TYQ IXC2  L*   RW
IDUGTB1R IXB1  L*   RW
IDUGTB1R IXC2  L*   RW
```

B1=BASE and instance=1
C2=CLONE and instance=2

DB2 Commands

- After EXCHANGE

```
NAME     TYPE PART  STATUS
-------- ---- ----- ---------

IDUGUTS1 TSB2  0001 RW
IDUGUTS1 TSC1  0001 RW,COPY
IDUG1TYQ IXB2  L*   RW
IDUG1TYQ IXC1  L*   RW
IDUGTB1R IXB2  L*   RW
IDUGTB1R IXC1  L*   RW
```

Before EXCHANGE

```
NAME     TYPE PART  STATUS
-------- ---- ----- ------------

IDUGUTS1 TSB1  0001 RW,COPY
IDUGUTS1 TSC2  0001 RW
IDUG1TYQ IXB1  L*   RW
IDUG1TYQ IXC2  L*   RW
IDUGTB1R IXB1  L*   RW
IDUGTB1R IXC2  L*   RW
```

- B1 was the base – now becomes B2.
  - Switches from INSTANCE=1 to 2 – but still BASE table
- C2 was the clone – now becomes B1
  - Switches from INSTANCE=2 to 1 – but still the CLONE table
- The PAGESET remains in the status it had prior to Exchange

DB2 Commands

- After DROP of CLONE
- What was the clone and then became ACTIVE now remains the ACTIVE pagesets after clone dropped (notice COPYP is gone)

```
NAME     TYPE PART   STATUS
-------- ---- ----- ----------------
IDUGUTS1 TSB2  0001 RW
IDUG1TYQ IXB2  L*   RW
IDUGTB1R IXB2  L*   RW
```

- Like mentioned earlier – you will have to live with the new Instance numbers if EXCHANGE executed an odd number of times
- STOP/START support CLONE keyword

Interesting topic

- Identity column values “shared”
- This scenario first inserts into BASE – then inserts into CLONE

For Table => RASST02.IDUGTB1

BASE

COL01	COL02	COL3IDENT
1	10	0
2	10	120
3	10	122
4	10	124
6	10	126
7	10	128
8	10	130
9	10	132
12	10	134
18	10	136

For Table => RASST02.IDUGTB1C

CLONE

COL01	COL02	COL3IDENT
100	30	138
101	30	140
102	30	142
103	30	144
104	30	146
105	30	148
106	30	150
107	30	152
108	30	154
109	30	156
109	30	158
110	30	160

Oddities discovered

Some SQL messages still “cryptic”

```
CREATE  TABLESPACE IDUGUTS2
   IN IDUGDB01
   USING STOGROUP SYSDEFLT
   ERASE NO FREEPAGE 0 PCTFREE 5
   BUFFERPOOL BP1 LOCKSIZE
ANY
   SEGSIZE 64 NUMPARTS 4
   LOCKMAX SYSTEM;
DSNT400I SQLCODE = 000,  SUCCESSFUL
EXECUTION
CREATE  TABLE RASST02.IDUGTB2
(COL01 CHAR ( 4 )     NOT NULL
                      WITH DEFAULT
                      FOR SBCS DATA
,COL02 INTEGER        NOT NULL
                      WITH DEFAULT
,COL03 CLOB ( 20 K )  NOT NULL
                      WITH DEFAULT
,COL04 ROWID
       GENERATED ALWAYS NOT NULL)
       PARTITION BY SIZE EVERY 1G
IN IDUGDB01.IDUGUTS2 ;
DSNT408I SQLCODE = -763, ERROR:  INVALID
TABLE SPACE NAME IDUGUTS2
```

-763 INVALID TABLE SPACE NAME *table-space-name*
Explanation: The named table space is invalid for one of the following reasons:
It is a LOB table space and therefore cannot reside in a work file database.
It is a LOB table space and therefore cannot contain a non-auxiliary table.
It is a LOB table space and LOGGED was specified, but the associated base table space is defined as NOT LOGGED.
It is not a LOB table space and therefore cannot contain an auxiliary table.
System action: The statement cannot be executed. **Programmer response:** Either create the LOB table space in a non-workfile database.
Create the table in a non-LOB table space. Create the auxiliary table in a LOB table space.
In the CREATE or ALTER of a LOB table space use the same LOG attribute that you used for the base table space.

The Clones have landed – Watch Out !

Steen Rasmussen
CA inc.
steen.rasmussen@ca.com