

DB2 Regional User Group – NEODEBUG

**z/OS Tablespace Partitioning for DB2 V8:
Maintenance, Performance and more**

Agenda

- > Considerations for Partitioning
Maintenance , Performance , Availability
- > Overview DB2 Physical Storage Options
- > Convert Index to Table Controlled Partitioning
- > Design Case Studies
- > Maintenance and Recovery
- > Key Benefits of V8 partitioning
- > Tips to remember

WikiPedia : Database Partitioning?

- > A **partition** is a division of a logical database or its constituting elements into distinct independent parts
- > Database partitioning is normally done for manageability, performance or availability reasons
- > UDB LUW – Nodes
- > UDB z/OS – SYSPLEX Data Sharing

WikiPedia: Tablespace

- > A **tablespace** is a storage location where the actual data underlying database objects can be kept. It is the logical portion of the database used to allocate storage for all DBMS managed objects, such as table data, indexes, or stored procedures. Once created, a tablespace can be referred to by name when creating database objects.

Design Consideration for Partition

- > Maintenance
- > Performance
- > Availability

Maintenance

- > **Large Volume of Data**

 - Limits on simple, segmented is 64GB

 - Limit of 128 Terabytes for partitioned

- > **Utility Window**

 - What is the allowed DOWN TIME?

- > **Utilities**

 - Reorgs

 - Imagecopy

 - Recovery

Performance

- > **Query, CPU, and SYSPLEX Parallelism**
- > **Reduced I/O**
 - Page Range Scan
- > **Batch**
 - Partition for parallel processing (Region, Department)
 - Work Load Balancing
- > **Active Data**
 - Process only data that is active or current

Availability

- > **Utility and Transaction Access**

 - Concurrent Access

- > **Recovery**

 - Recover one or more partitions in parallel

- > **Manage Growth**

 - Adding New Partitions

Physical Storage: Tablespace

- Simple
- Segmented
- Partitioned
 - Index Control
 - Table Control
- Large Object (LOB)

<i>EMPNO</i>	<i>FIRSTNME</i>	<i>MIDINI T</i>	<i>LASTNAME</i>	<i>WORKDEPT</i>
000010	Christine	I	Hass	A00
000020	Michael	L	Thompson	B01
000030	Sally	A	Kwan	C01
000110	Vincenzo	G	Lucchesi	A00
000120	Sean	J	O'Connell	A00
000130	Dolores	M	Quintana	C01
000140	Heather	A	Nicholls	C01
200010	Dian	J	Hemminger	A00
200120	Greg		Orlando	A00
200140	Kim	N	Natz	C01

<i>WORK DEPT</i>	<i>DEPTNAME</i>
A00	SPIFFY COMPUTER SERVICE DIV.
B01	PLANNING
C01	INFORMATION CENTER
D01	DEVELOPMENT CENTER

Simple

4K Page

000010	Christine	I	Hass	A00
A00	SPIFFY COMPUTER SERVICE DIV.			
000020	Michael	L	Thompson	B01
B01	PLANNING			

4K Page

000030	Sally	A	Kwan	C01
000110	Vincenzo	G	Lucchesi	A00
000120	Sean	J	O'Connell	A00
C01	INFORMATION CENTER			

4K Page

000130	Dolores	M	Quintana	C01
000140	Heather	A	Nicholls	C01
200010	Dian	J	Hemminger	A00
200120	Greg		Orlando	A00

4K Page

200140	Kim	N	Natz	C01
D01	DEVELOPMENT CENTER			

□ EMPL

□ DEPT

Segmented

□ EMPL

□ DEPT

Segment
8K

Segment for table EMPL

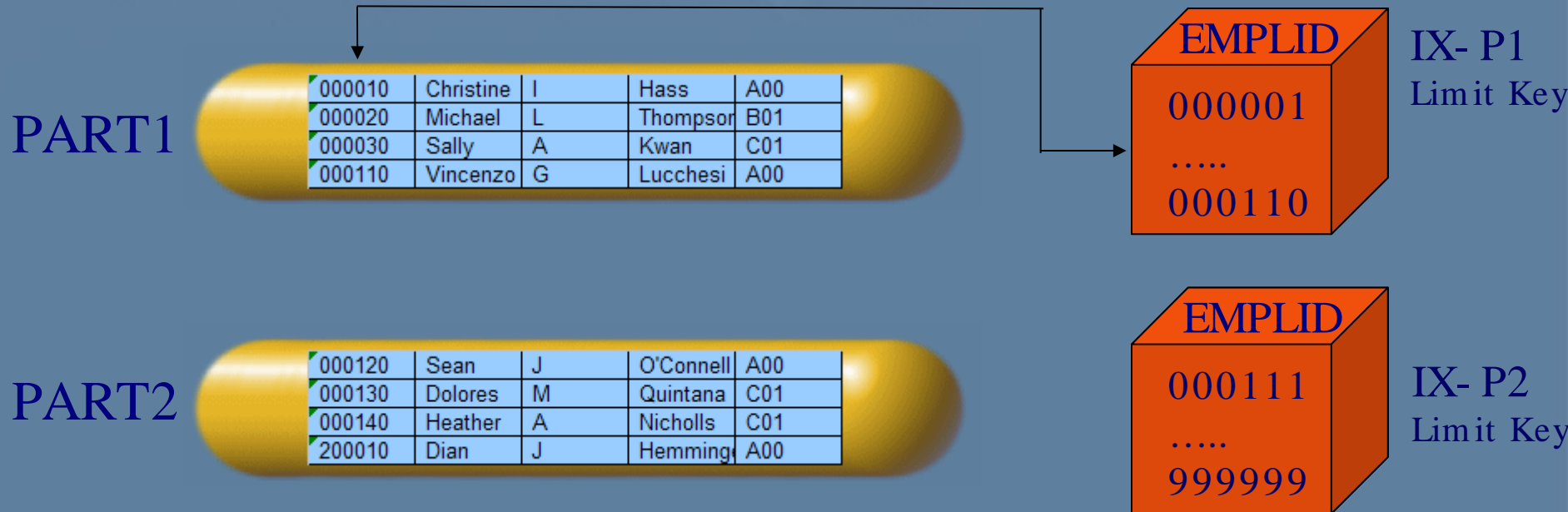
000010	Christine	I	Hass	A00
000020	Michael	L	Thompson	B01
000030	Sally	A	Kwan	C01
000110	Vincenzo	G	Lucchesi	A00
000120	Sean	J	O'Connell	A00
000130	Dolores	M	Quintana	C01
000140	Heather	A	Nicholls	C01
200010	Dian	J	Hemming	A00

Segment
8K

A00	SPIFFY COMPUTER SERVICE DIV.
B01	PLANNING
C01	INFORMATION CENTER
D01	DEVELOPMENT CENTER

200120	Greg		Orlando	A00
200140	Kim	N	Natz	C01

Partitioned - Index Control



V7 Clustering by EMPLID – Partitioning Ind

Create Tablespace - IX Partition

```
CREATE TABLESPACE CSBPTS1B
  IN CSBPDBP
  Numparts 2
  USING STOGROUP CSBPSG
  PriQty 48
  SecQty 48
  ERASE NO
  FREEPAGE      0
  PCTFREE       20
  BUFFERPOOL    BP0
  LOCKSIZE      ROW
  ;
```

```
CREATE TABLE CSBPPIX.EMPL
  (EMPNO CHAR(6) NOT NULL,
  ...
  ) IN CSBPDBPX.CSBPTS1B

CCSID EBCDIC;
CREATE TYPE 2 UNIQUE INDEX
  CSBPPIX.XEMP1
  ON CSBPPIX.EMPL
  ( EMPNO ASC )
  USING STOGROUP YD43TX01
  ...
  CLUSTER
  (PART 1 VALUES ('399999',"),
  PART 2 VALUES ('999999',") )
  BUFFERPOOL BP0
  CLOSE NO
```

Create Tablespace – TB Partition

```
CREATE TABLESPACE CSBPTS1B
IN CSBPDBPX
  NUMPARTS 2
  USING STOGROUP CSBPSG
  PRIQTY 48
  SECQTY 48
  ERASE NO
  FREEPAGE 0
  PCTFREE 20
  BUFFERPOOL BP0
  LOCKSIZE ROW;
```

```
CREATE TABLE CSBPTBL.EMPL
(EMPNO CHAR(6) NOT NULL,
..... 14 Line(s) not Displayed
  SQL_USER CHAR(08) NOT NULL WITH
      DEFAULT CURRENT SQLID,
  PRIMARY KEY(SQL_USER, EMPNO),
  FOREIGN KEY RED (SQL_USER,WORKDEPT)
  REFERENCES CSBPTBL.DEPT
  ON DELETE SET NULL)
IN CSBPPTBL.YDGFTS1B
PARTITION BY (EMPNO ASC)
( PARTITION 1 ENDING AT ('399999')
, PARTITION 2 ENDING AT ('999999'))
CCSID EBCDIC;
```

```
CREATE TYPE 2 UNIQUE INDEX CSBPTBL.XEMP
ON CSBPTBL.EMPL
( SQL_USER ASC,EMPNO ASC)
USING STOGROUP YD43TX01
  PRIQTY 48
  SECQTY 48
  ERASE NO
  CLUSTER
.....;
```

Partition Control Differences

TABLE	INDEX
A partitioning index is not required; clustering index is not required.	A partitioning index is required; clustering index is required.
Multiple partitioned indexes can be created in a table space.	Only one partitioned index can be created in a table space.
A table space partition is identified by both a physical partition number and a logical partition number.	A table space partition is identified by a physical partition number.
The high-limit key is always enforced.	The high-limit key is not enforced if the table space is non-large.

Table Control Benefits

- > No indexes required
- > Cluster different from Partitioning columns
- > Multiple Partitioned Indexes

What type are you using?

- > TABLE – LIMITKEY_INTERNAL
- > INDEX – IXNAME

```

SELECT SUBSTR(DBNAME,1,8) AS DBNAME
      , SUBSTR(TSNAME,1,8) AS TSNAME
      , LOGICAL_PART
      , SUBSTR(LIMITKEY,1,20) AS LIMITKEY
      , PARTITION
      , SUBSTR(IXNAME,1,18) AS IXNAME
      , SUBSTR(LIMITKEY_INTERNAL,1,20) AS LIMITKEY_INTERNAL
FROM SYSIBM.SYSTABLEPART
WHERE (   DBNAME = 'CSBPDBTC'
        AND TSNAME = 'CSBPTS1A')
      OR
      (   DBNAME = 'CSBPDBPX'
        AND TSNAME = 'CSBPTS1A')
ORDER BY DBNAME, TSNAME, LOGICAL_PART
WITH UR;

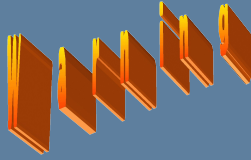
```

	DBNAME	TSNAME	LOGICAL_PART	LIMITKEY	PARTITION	IXNAME	LIMITKEY_INTERNAL
1_	CSBPDBPX	CSBPTS1A	1	'D99'	1	XDEPT1
2_	CSBPDBPX	CSBPTS1A	2	'Z99'	2	XDEPT1
3_	CSBPDBTC	CSBPTS1A	1	'C99'	1		C99.....
4_	CSBPDBTC	CSBPTS1A	2	'Z99'	2		Z99.....

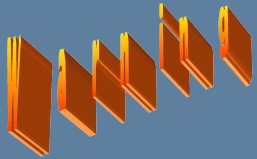
Automatic Conversion to Table Control

- > **CREATE INDEX with**
 - PARTITIONED clause on indexed controlled partitioned table space
 - PART VALUES clause without CLUSTER
- > **ALTER INDEX – “NOT CLUSTER” on partitioned IX**
- > **DROP INDEX on partitioning index**
- > **ALTER TABLE**
 - Add new partition
 - Change partition boundaries
 - Rotate a partition from first to last

Steps to Change Primary Key

- 
- > Step 1: Alter DROP Foreign Key
 - > Step 2: Alter DROP Primary Key
 - > Step 3: DROP Partitioning Index
 - > Step 4: CREATE CLUSTER INDEX
-
- > Step 5: Alter ADD Primary Key
 - > Step 6: Alter Add Foreign Key

Set REORGP on Last Partition when?



- > When adding a new partition, DB2 stored the original high-limit key instead of the default high-limit key. If not previously enforced, DB2 will set the last part into REORGP state.
- > When rotating a new partition, DB2 stored the original high-limit key instead of the default high-limit key. If not previously enforced, DB2 will set the last part into REORGP state.
- > `-DISPLAY DB(*) SPACE(*) RESTRICT`

Physical Design Case Studies

> Case 1 – Number of partitions

A company receives billing data from multiple billing systems daily and has to merge this data and produce customer bills within 24 hours.

> Case 2 – Batch VS Online Partitioning

A company has the requirement to produce month end reports for all account activity, and is also required to give customers access to the last 12 months of account activity.

Case 1: Number of Partitions

- > Business Requirement
- > V7 Design Issue
- > V8 – Redesign Options

Case 1: Business Requirements

- > Receive 5 billing files with millions of records each day.
- > Generate Billing Statements for today's cycle
- > 40 billing cycles
- > Must complete loading and generating billing statement in 24 hours

Case 1: Generate Partition Key

> MOD 40

Mod(days(date),40)

Partition values will be 0 through 39

```
SELECT DAYS(CURRENT DATE) AS CURRENT_DAYS
      , DAYS(CURRENT DATE)
      - (DAYS(CURRENT DATE)/40)*40 AS MOD_40
FROM SYSIBM.SYSDUMMY1
WITH UR;
```

CURRENT_DAYS	MOD_40
732590	30

Case 1: Create Objects – MOD 40

```
CREATE TABLESPACE CSBPTS1B
  IN CSBPDBP
  Numparts 40
  USING STOGROUP SMS
  PriqTY 48
  SecqTY 48
  ERASE NO
  FREEPAGE 0
  PCTFREE 20
  BUFFERPOOL BP0
  LOCKSIZE ROW
;
```

```
CREATE TABLE CSBPPIX.ACCT
( PARTNO SMALLINT NOT NULL,
  ,ACCT_ID CHAR(10) NOT NULL,
  ...
) IN CSBPDBPX.CSBPTS1B
```

```
CCSID EBCDIC:
CREATE TYPE 2 UNIQUE INDEX
CSBPPIX.ACCT_X1
ON CSBPPIX.ACCT
( PARTNO ASC,ACCT_ID)
USING STOGROUP SMS
...
CLUSTER
(PART 1 VALUES (00000) ,
  ....
  PART 40 VALUES (00039) )
BUFFERPOOL BP0
CLOSE NO
```

Case 1: Limitations

- > Required design logic in all applications to process by partition number.
- > Reporting Tools Perform Poorly
 - Partitioning (cluster) key not known to reporting tools
- > Intelligent Purge Routine
 - clean out the correct partition based on the MOD value
- > Outage required to change the number of days and application coding changes.

Case 1: Re-design under V8

- > Use “Date” data type partition key instead of generated MOD PARTNO
- > Improve Ad-Hoc Query Performance
- > Implement Partition Rotation
 - Eased the maintenance of cleaning out the next partition
- > Can extend the number of days if needed without an outage

DB2 V8: Logical vs Physical Partition

SYSIBM.SYSTABLES

LOGICAL_PART	ROW_CNT	LIMITKEY	PARTITION
1	2	2006-10-07	1
2	1	2006-10-08	2
3	1	2006-10-09	3

What happens when I ROTATE FIRST TO LAST?

ALTER TABLE Rotate

```
ALTER TABLE CSBPDAT.TRANSACTION  
ROTATE PARTITION FIRST TO LAST  
ENDING AT ('2006-10-10') INCLUSIVE RESET
```

Catalog After Rotation:

LOGICAL_PART	ROW_CNT	LIMITKEY	PARTITION
1	1	2006-10-08	2
2	1	2006-10-09	3
3	0	2006-10-10	1

What happens when I ADD a PARTITION?

ALTER TABLE ADD Partition

```
ALTER TABLE CSBPDAT.TRANSACTION  
ADD PARTITION  
ENDING AT ('2006-10-11') INCLUSIVE
```

Catalog After ADD Partition:

LOGICAL_PART	ROW_CNT	LIMITKEY	PARTITION
1	1	2006-10-08	2
2	1	2006-10-09	3
3	0	2006-10-10	1
4	-1	2006-10-11	4

ALTER Add Partition Rules

- > PRIQTY/SECQTY of previous LOGICAL partition is used
- > For specific space attributes use alter tablespace / index
- > Cannot be or have a materialized query dependency

Case 2: Batch VS Online Partitioning

- > Business Requirement
- > V7 Design Issue
- > V8 – Redesign Options

Case 2: Business Requirements

- > Store 24 months of transactions (invoices)
- > Million invoices each month
- > Month end processing of invoices by account
- > On-line activity – customer account history

Case 2: Physical Design

- > **To handle the large insert processing**

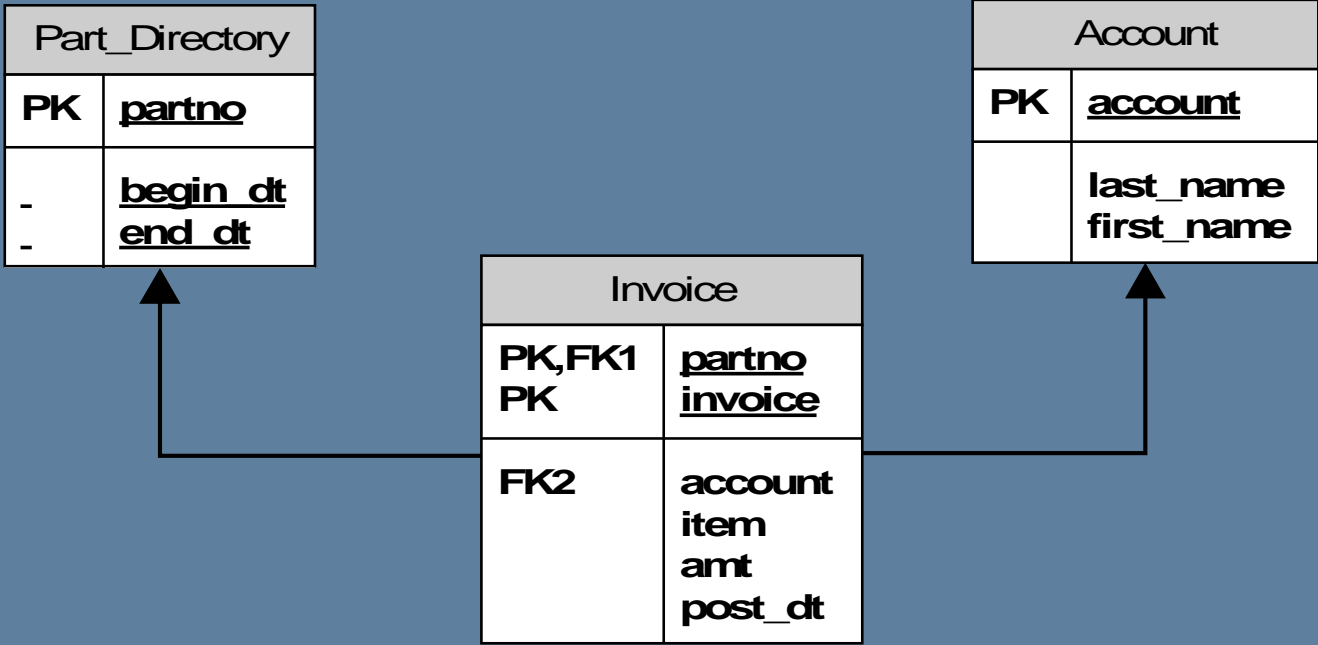
Partition by date range within 24 partitions

- > **To avoid DDL to maintain partitions**

Directory table to contain date range for each partition

Case 2: Create Objects - Batch VS Online

Batch: Process by PARTNO
On-Line: Process by ACCOUNT



Case 2: Advantage of this design

- > No DDL need to add or remove partitions
- > No need to reorganize except for non-partitioned indexes
- > Large volumes (millions) of rows can be inserted with acceptable performance

Case 2: Disadvantage

- > Requires a column without meaning
- > Additional table to maintain and online systems have to include it to access the data.
- > Complex queries are required to access data in more than one partition. More non-partitioned indexes needed.
- > Almost always have to create a second non-partitioned index that has all the columns of the partitioning index except the partno.

Case 2: Re-design under V8

- > Partition by Invoice Posting Date
- > Cluster by Invoice Number
- > Remove directory table;
Use alter table rotate partition first to last
- > Convert to Table Controlled Partitioning

Case 2: Redesign Advantage

- > Removes need for directory table
- > Removes column without meaning
- > Simplifies query access for online processing
- > Reduces the need for maintenance programs to purge out the data.

MAINTENANCE: Query Parallelism

- > When data becomes out of balance in a partitioned table space, performance can decrease.
- > **REORG REBALANCE**
 - DB2 will set the limit keys

MAINTENANCE: Rebalance 356 rows

Before

LOGICAL_PART	ROW_CNT	LIMITKEY	PARTITION	LIMITKEY_I
1	59	2006-02-28	1
2	122	2006-06-30	2
3	175	2006-12-31	3

After

LOGICAL_PART	ROW_CNT	LIMITKEY	PARTITION	LIMITKEY_I
1	80	2006-03-21	1
2	80	2006-06-09	2
3	196	2006-12-31	3

MAINTENANCE: Rebalance Restrictions

- > ***IBM DOCUMENTATION:
Restriction when physical partition numbers
do not match logical partition numbers:***

A REORG REBALANCE might not be possible if the logical and physical partition numbers for the specified table space do not match. This situation can be created by a series of ALTER ROTATEs and ALTER ADD PARTs.

- > My “TEST” shows that this works fine.

MAINTENANCE: Rebalance ROTATE

Before

LOGICAL_PART	ROW_CNT	LIMITKEY	PARTITION	LIMITKEY_I
1	30	2006-11-30	2
2	22	2006-12-31	3
3	90	2007-03-31	1
4	266	2007-12-31	4

After

LOGICAL_PART	ROW_CNT	LIMITKEY	PARTITION	LIMITKEY_I
1	80	2007-01-28	2
2	80	2007-04-18	3
3	80	2007-07-07	1
4	168	2007-12-31	4

Getting Help: DB2 Listserve

Search Results - DB2-L - Windows Internet Explorer

http://www.idugdb2-l.org/adminscripts/wa.exe

LISTSERV(R) 1.8e

Archive Search

[Subscriber's Corner](#)
[Server Archives](#)
[List Archives](#)

[Help](#)
[Log off](#)

[Subscriber's Corner](#)
[Archive Search](#)

[Return to Search Form...]

Search Results:
DB2-L: 22 matches.

Item #	Date	Time	Recs	Subject
088591	06/07/17	08:39	109	Re: v8 z/os-looking for reorg rebalance experiences
088524	06/07/13	09:43	42	v8 z/os-looking for reorg rebalance experiences
088543	06/07/13	08:25	114	Re: v8 z/os-looking for reorg rebalance experiences
088413	06/07/05	08:20	21	Re: v8 z/os-looking for reorg rebalance experiences
088385	06/07/01	10:29	40	Re: v8 z/os-looking for reorg rebalance experiences
088358	06/06/30	07:40	11	v8 z/os-looking for reorg rebalance experiences
087752	06/06/02	08:42	47	Re: improper REBALANCE
087666	06/05/30	17:10	255	Re: improper REBALANCE
087674	06/05/30	11:24	27	Re: improper REBALANCE
087602	06/05/24	15:05	95	Re: improper REBALANCE

RECOVERY- TO POINT IN TIME

What happens when I recover a tablespace to a point in time after:

- > ALTER TABLE ROTATE FIRST TO LAST
- > REORG REBALANCE

RECOVERY: ALTER ISSUE

- > You CANNOT recover to a point in time prior to the ALTER TABLE ROTATE FIRST TO LAST

```
DSNU556I -DB8A DSNUCASA - RECOVER CANNOT PROCEED FOR TABLESPACE  
CSBPPDAT.YDGFTS1A DSNUM 1  
BECAUSE A SYSIBM.SYSCOPY RECORD HAS BEEN ENCOUNTERED WHICH HAS  
DBNAME=CSBPPDAT TSNAME=YDGFTS1A DSNUM=1 ICTYPE=A  
STARTRBA=X'00007EE42CDC' LOWDSNUM=0 HIGHDSNUM=0
```

RECOVERY: SYSCOPY REPORT

TSNAME	DSNUM	LOGICAL_PART	ICTYPE	ICDATE	START_RBA
YDGFTS1A	1	3	A	061027	00007EE42CDC
YDGFTS1A	0	0	Q	061027	00007EE2385B
YDGFTS1A	3	3	F	061027	00007EE1D4B0
YDGFTS1A	2	2	F	061027	00007EE0AF86
YDGFTS1A	1	1	F	061027	00007EDF89D4

RECOVERY: REBALANCE ISSUE

- > DB2 will let you recovery to a point in time before the rebalance

Potential Lost of Data

- > Tablespace left in REORGP

REORG with DISCARD DD

- > Indexes are left in RBDP

RECOVERY: SYSCOPY REPORT

TSNAME	DSNUM	LOGICAL_PART	ICTYPE	ICDATE	START_RBA
YDGFTS1A	3	2	F	061027	00007EF97AF0
YDGFTS1A	2	1	F	061027	00007EF85553
YDGFTS1A	1	3	F	061027	00007EF72F92
YDGFTS1A	0	0	F	061027	00007EECD356
YDGFTS1A	0	0	W	061027	00007EEB6ED6
YDGFTS1A	0	0	Q	061027	00007EEA37E3
YDGFTS1A	3	2	F	061027	00007EE9D4EA
YDGFTS1A	2	1	F	061027	00007EE8B22A
YDGFTS1A	1	3	F	061027	00007EE78BEC
YDGFTS1A	1	3	A	061027	00007EE42CDC
YDGFTS1A	0	0	Q	061027	00007EE2385B
YDGFTS1A	3	3	F	061027	00007EE1D4B0
YDGFTS1A	2	2	F	061027	00007EE0AF86
YDGFTS1A	1	1	F	061027	00007EDF89D4

RECOVERY: Row count

Before Rebalance

LOGICAL_PART	ROW_CNT	LIMITKEY	PARTITION	LIM
1	334	2006-11-30	2
2	22	2006-12-31	3
3	0	2007-03-31	1

After Rebalance

LOGICAL_PART	ROW_CNT	LIMITKEY	PARTITION	LIM
1	80	2006-03-21	2
2	80	2006-06-09	3
3	196	2007-03-31	1

RECOVERY: Row count after recover part 1

After Recover, Reorg, Rebuild

LOGICAL_PART	ROW_CNT	LIMITKEY	PARTITION	LIM
1	80	2006-03-21	2
2	80	2006-06-09	3
3	0	2007-03-31	1

LOST 196 rows.

RECOVERY: All Parts If Rebalanced

BEFORE REBALANCE

- > FULL IMAGECOPY all partitions
- > QUIESCE
- > REORG REBALANCE

RECOVERY STEPS

- > RECOVER ALL PARTS TORBA
- > REBUILD INDEX
- > REORG TABLESPACE

Key Benefits in V8 Partitioning

1. Add new PARTITION without dropping the table.
2. ROTATE the partitions reducing the need to build extra empty partitions.
3. PARTITION without an INDEX
4. CLUSTER different from partitioning key columns

Tips To Remember

- > Use Table Controlled Partitioning
- > Avoid generating a partitioning column
- > Imagecopy all partitions before and after REORG REBALANCE

Summary

- Upgrade to DB2 V8 ASAP!
- DB2 V8 has many features that will reduce the TCO “Total Cost of Ownership”. The partitioning features are just one piece of the TCO, reducing the cost to maintain the data and reduce the cost of outages.

Q & A



Speaker Contact Information

Troy Coleman

SoftBase

1-800-669-7076 X334

Direct: 1-847-776-0618

Email: troy.coleman@softbase.com

THANK YOU

