



DB2 Data Management Software

DB2 Managing Large Partitioned Tables in V8

SHARE in Long Beach, CA
February 23 - 27, 2004
Session 1346

Craig Friske
DB2 Development, Silicon Valley Lab
friske@us.ibm.com

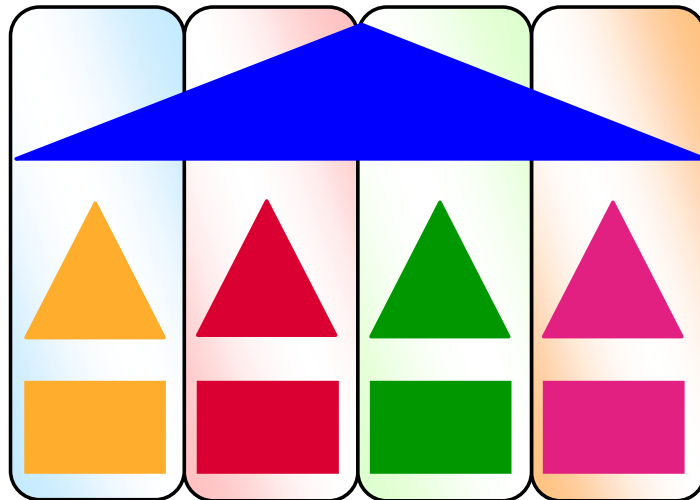
© 2004 IBM Corporation

Agenda

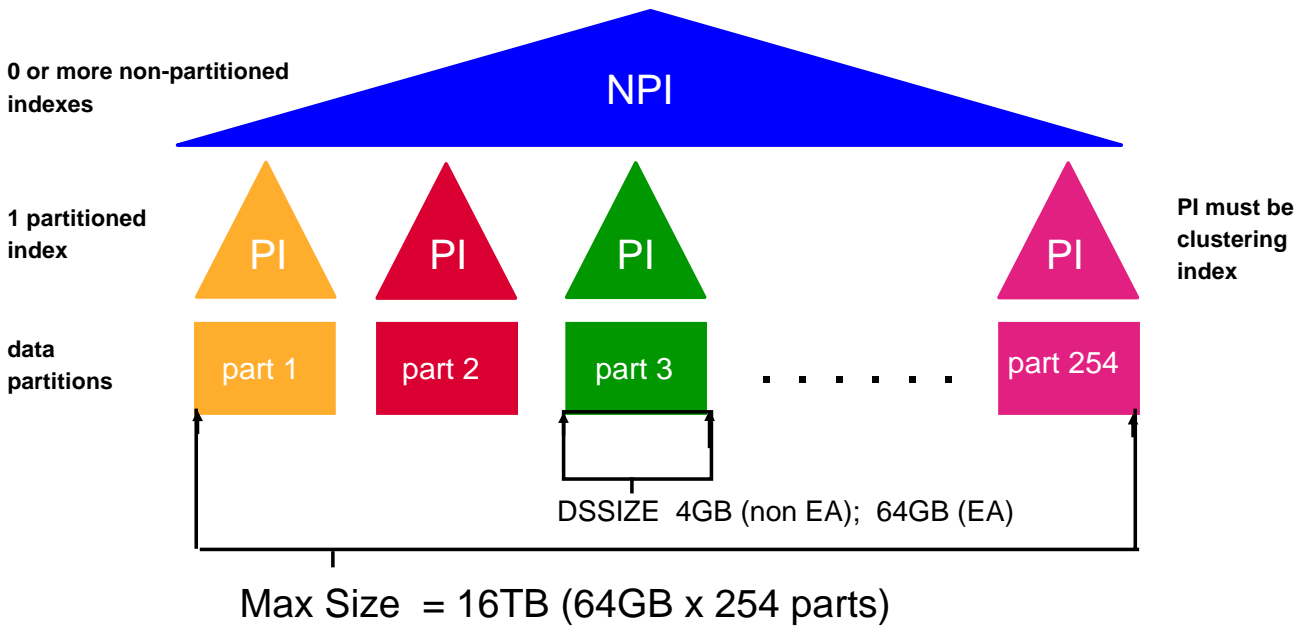
- V8 Partitioned Table Overview
- Changing partition(ing) attributes
 - f* Adding partitions
 - f* Rotating partitions
- Partitioning without a partitioning index
- Utilities and commands
 - f* SCOPE PENDING
 - f* Display
 - f* REORG ... REBALANCE
- System planning



V8 Partitioned Tables

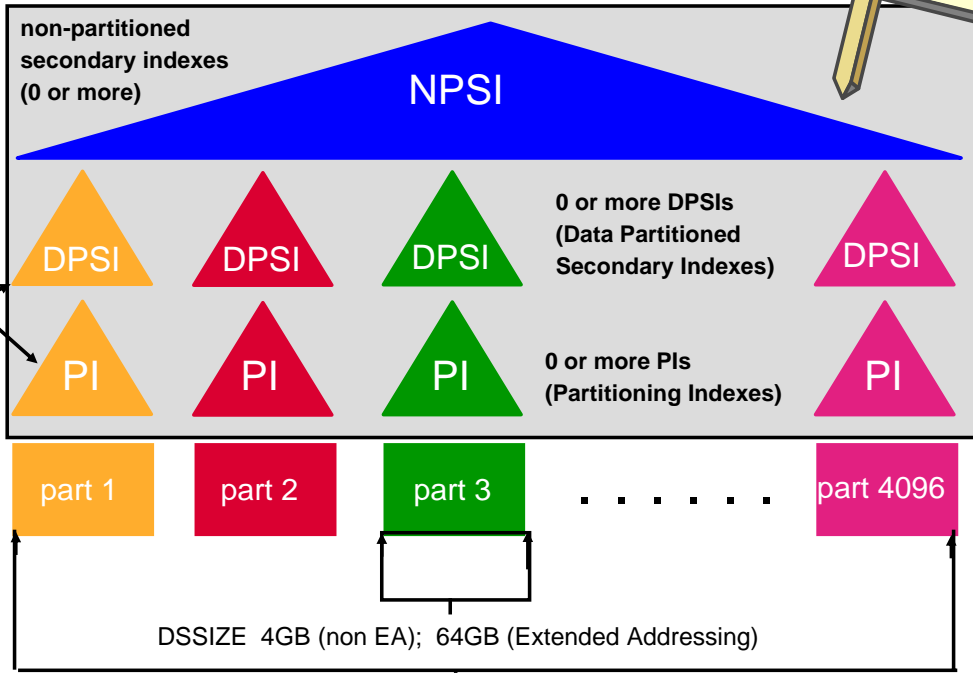


V7 Partitioned Table



V8 Partitioned Table

1 partition per day for 10 years!



indexes can be padded or not padded

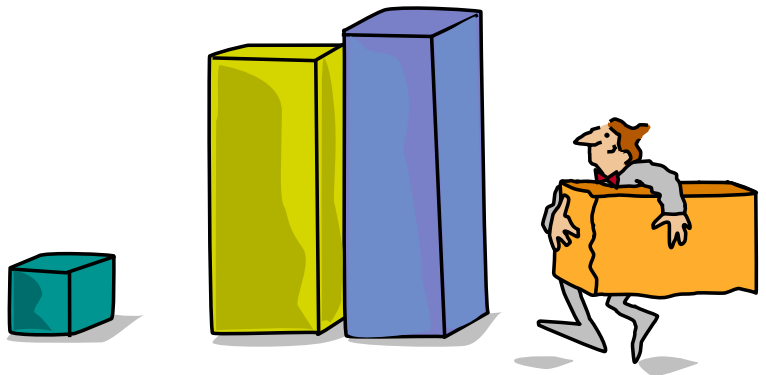
any index can be clustering index

partitioned indexes (0 or more)

data partitions

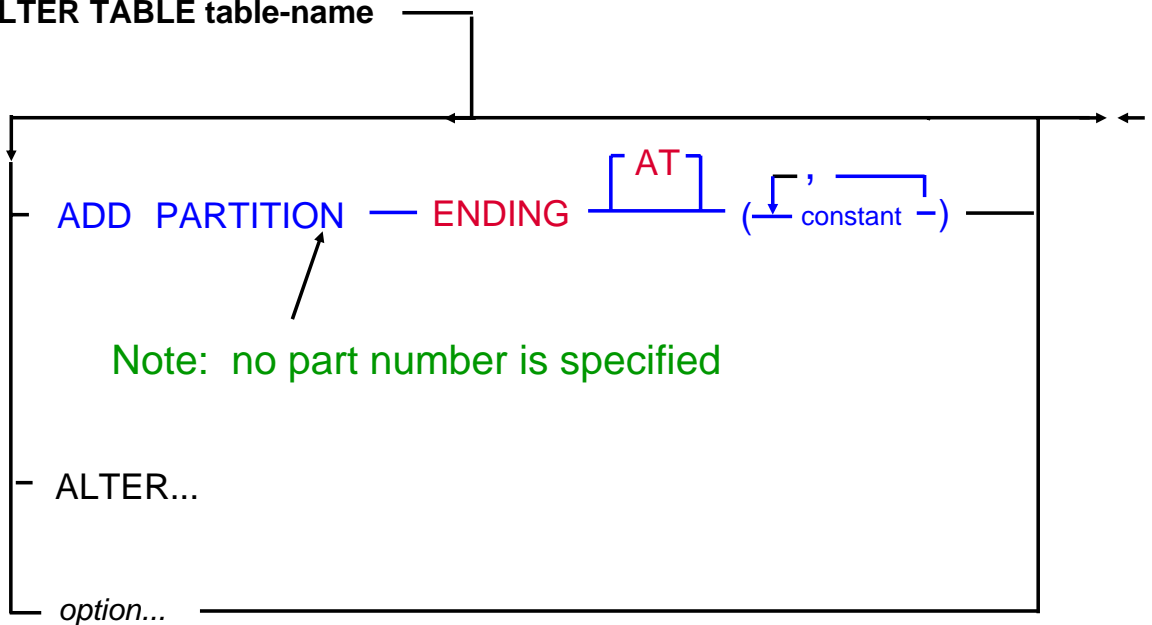
Max Size = 128TB (32GB x 4096 parts; 64GB x 2048 parts)

Changing Partition Attributes



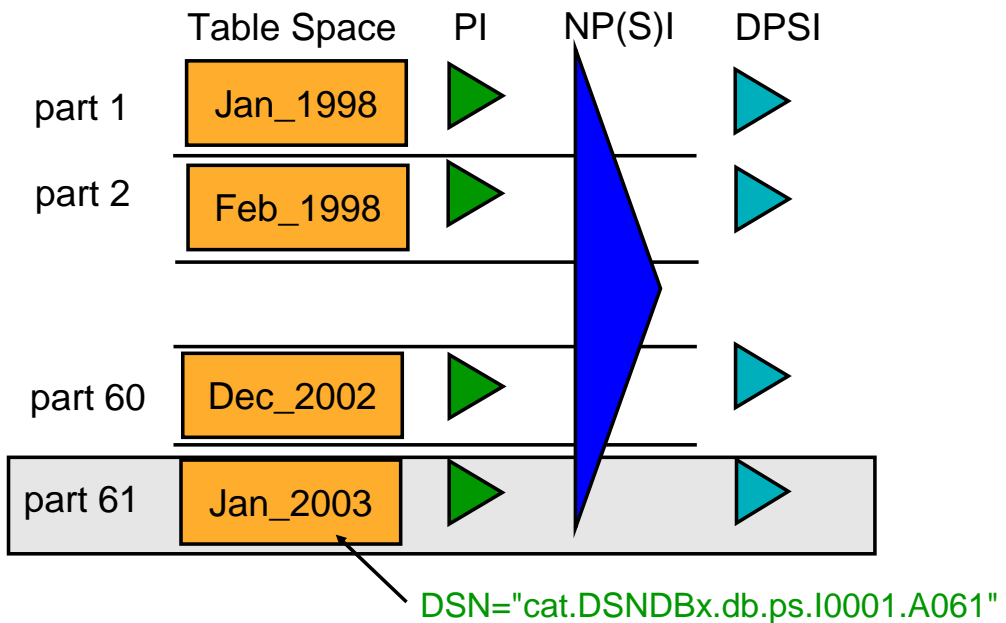
Add partition syntax

ALTER TABLE table-name



Add Partition Example

`ALTER TABLE . . . ADD PART ENDING AT ("01/31/2003");`



Operational Considerations

- Immediate availability
 - f* Table is quiesced and plans, packages and cached statement are invalidated
 - Necessary as access path may be optimized to read only certain partitions

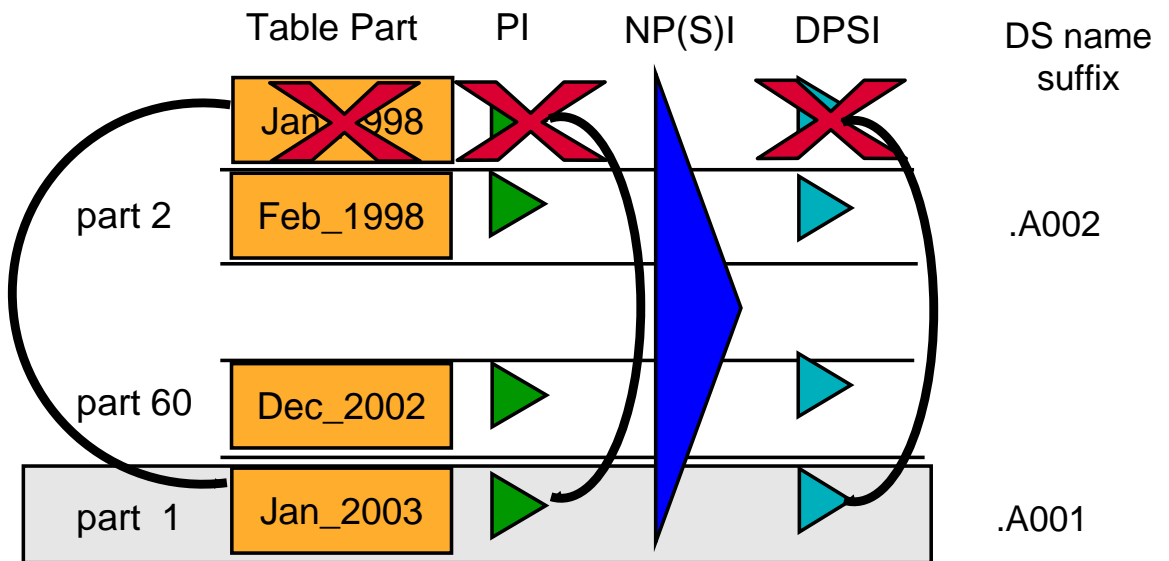
- Next physical partition number is assigned
 - f* **Maximum number of partitions based on table definition**

- Attributes of previous logical partition are used, e.g., PRIQTY
 - f* Run ALTER TABLESPACE statements afterwards

- If previous partition is in REORP, new one inherits as well

- Recover to point before a partition was added will reset it to empty.

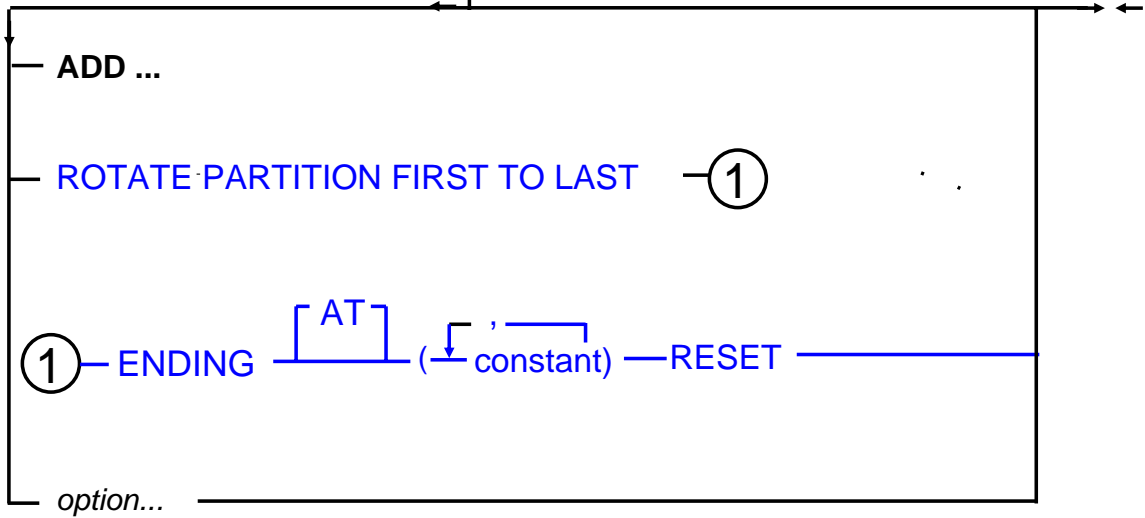
Rotate Partition Overview



- Old data is rolled away and partition reused

Rotate Partition Syntax

ALTER TABLE table-name



Rotate Partition Example

```
ALTER TABLE . . . ALTER PART ROTATE VALUES("01/31/2003") RESET;
```

physical logical

part 2

1

Feb_1998

part 60

59

Dec_2002

REORP

part 1

60

Jan_2003

REORP

REORP inherited from
previous partition;
otherwise immediately
available

Adding and rotating parts can mix up the partition order

See `SYSIBM.SYSTABLEPART LOGICAL_PART` and `PARTITION`

Rotate Partition...

- Immediate availability (no REORG necessary)
- Lowest logical partition becomes last logical partition
- Specified boundary must be new "high value"
- Last partition limit key is enforced
- Recover to previous PIT blocked as SYSCOPY and SYSLGRNX entries are deleted

Operational Considerations

- Data is deleted so you may wish to unload it first
- Rotate will issue individual deletes
 - f* Consider impact on logging and performance
 - f* DBD lock held for the duration of the ROTATE
- Suggestion: run `LOAD . . . PART n REPLACE` with a dummy `SYSREC` dataset to empty out the partition first
 - f* If NPIs exist, consider the performance

Drop Partitioning Index

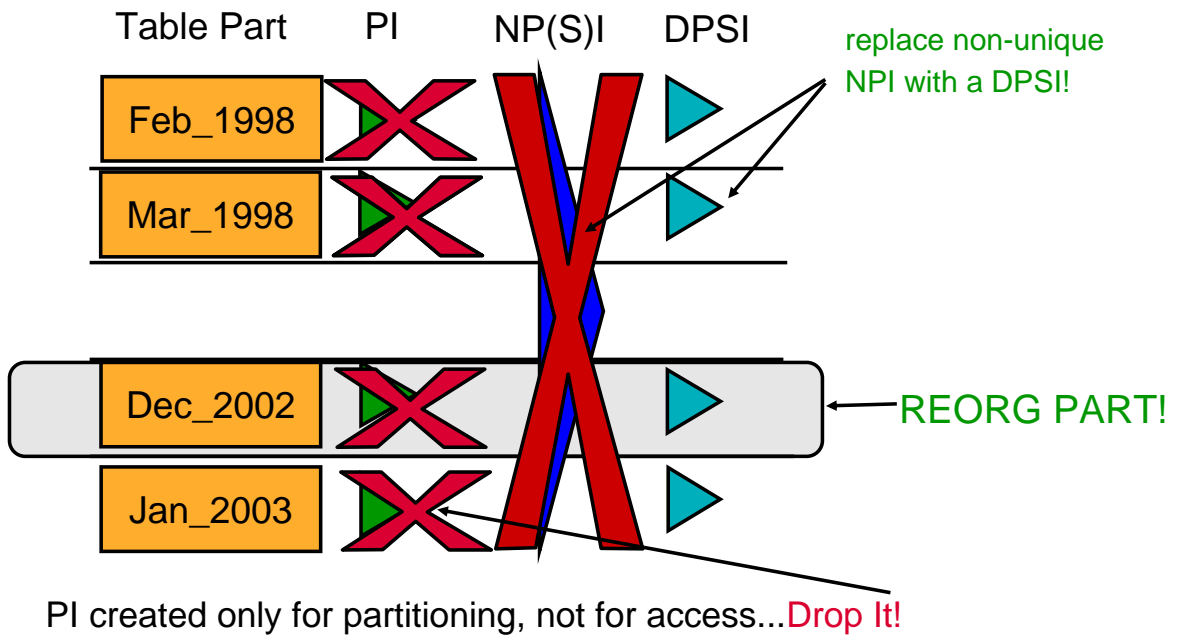







Table-Controlled Partitioning

- No dependence on indexes (index-controlled partitioning)
- Managed with new catalog columns: 

Catalog Table	Catalog Column	Index-Controlled	Table-Controlled
SYSTABLESPACE	PARTITIONS	x	x
SYSTABLEPART	LIMITKEY LIMITKEY_INTERNAL	x (external format)	x x 
SYSINDEXPART	LIMITKEY	x (internal format)	
SYSTABLES	PARTKEYCOLUMN	implicit in index	x 
SYSCOLUMNS	PARTKEY_SYSCOLSEQ PARTKEY_ORDERING	implicit in index implicit in index	x  x 

- Limit Key enforcement on last partition
- Converted to table-controlled when new partition function exploited

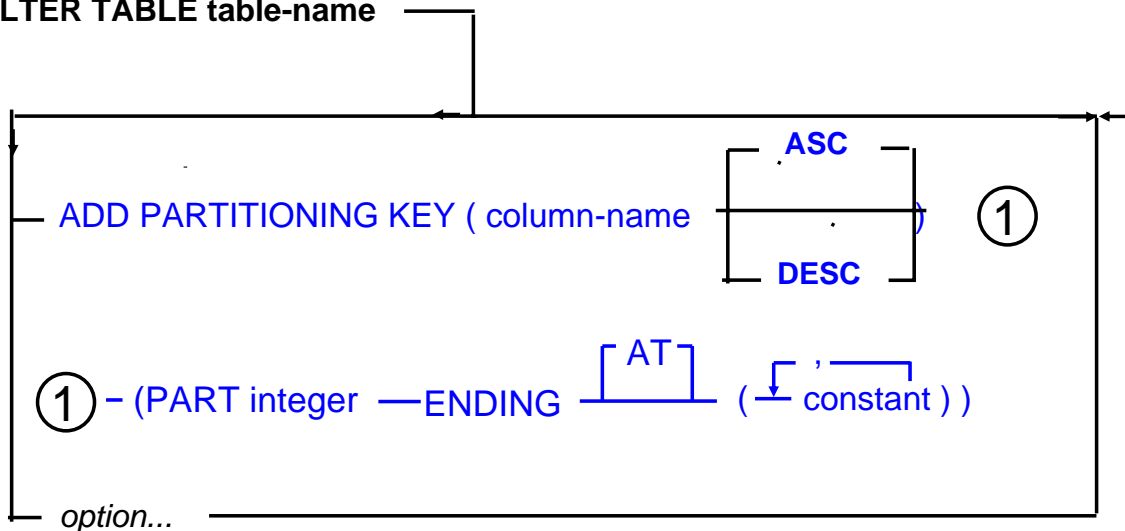
Converting to table controlled partitioning

- To convert existing tables to table-controlled partitioning, it is not necessary to drop and recreate the table
- Start with an existing table with a single PI
- Use any new function and a conversion is triggered from index-controlled to table-controlled partitioning:
 - Drop the partitioning index (partitioning switched to table-based)
 - Create index PARTITIONED (DPSI or PI)
 - Add a partition
 - Rotate partitions
 - Create INDEX VALUES but no CLUSTER keyword
 - **ALTER existing partitioning index NOT CLUSTER**

Add Partitioning Key Syntax

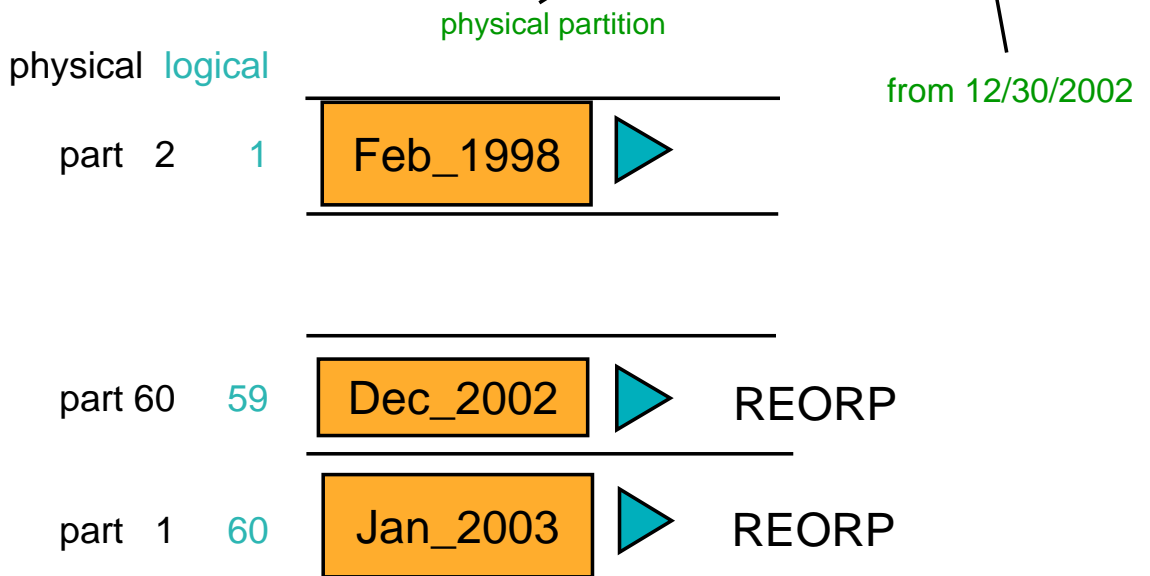
- Follows a CREATE TABLE ... LIKE statement
 - f* Flexibility to change table type or partition boundaries

ALTER TABLE table-name

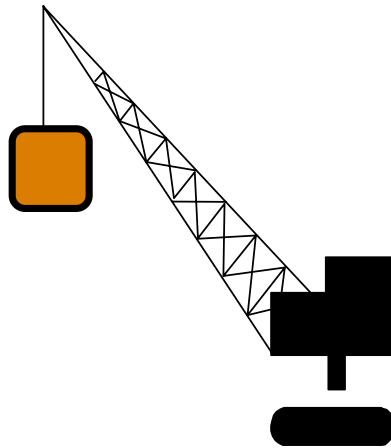


Alter Partition Boundary Example

ALTER TABLE . . . ALTER PART 60 VALUES ("12/31/2002");



Utilities and Commands



DISPLAY DATABASE

```
DSNT360I = *****
DSNT361I = * DISPLAY DATABASE SUMMARY 896
          * GLOBAL
DSNT360I = *****
DSNT362I = DATABASE = DB STATUS = RW 898
          DBD LENGTH = 4028
NAME  TYPE  PART  STATUS
-----  ---  -----  -----
TS     TS    0002   RW
  -THRU  4095
TS     TS    4096   RW,REORP
TS     TS    0001   RW,REORP
XDPSI  IX     D0001   RW
  -THRU  4096
XNPSI  IX     L*     RW
XPI    IX     0001   RW
  -THRU  4096
***** DISPLAY OF DATABASE DB ENDED
```

Display on 4096 part table
after

- 1) Alter Part 4096 (<LK)
- 2) ROTATE

Other REORG TABLESPACE enhancements

- SCOPE PENDING
- Discard option with SHRLEVEL CHANGE
- SORTKEYS and SORTDATA are now default for REORG TABLESPACE
 - f* If no clustering index, first index defined is used
 - f* If table space has no indexes, SORTDATA will operate as pre-V8
- REORG INDEX supports "INDEXSPACE" syntax

REORG ... SCOPE PENDING option

- Reorganizes the table space or part(s) that are in REORG-pending (REORP) or advisory REORG-pending state (AREO*)
- For part range, the adjacent high and low parts not included in the range must not be in REORP
- SYSCOPY records are written for only those actually reorganized

SCOPE PENDING

	ts	DBET State
part 1	Jan_1998	
part 2	Feb_1998	AREO*
part 3	Mar_1998	AREO*
part 13	Oct_2002	REORP
part 14	Nov_2002	REORP
part 15	Dec_2002	

**Parts 2,3,13,14
will be REORG'd**

... SCOPE PENDING PART 2:14 ...

	ts	DBET State
part 1	Jan_1998	
part 2	Feb_1998	AREO*
part 3	Mar_1998	AREO*
part 13	Oct_2002	REORP
part 14	Nov_2002	REORP
part 15	Dec_2002	REORP

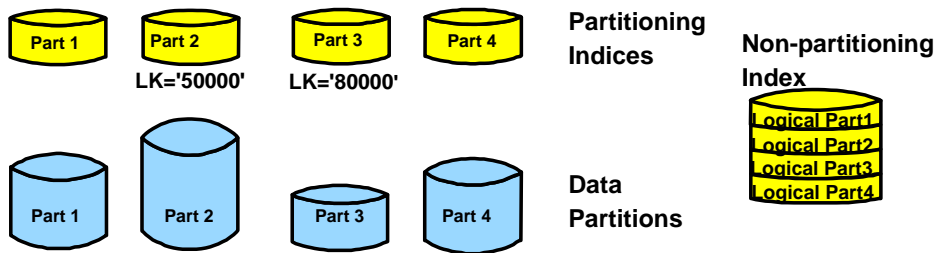
Adjacent part 15 in REORP blocks REORG



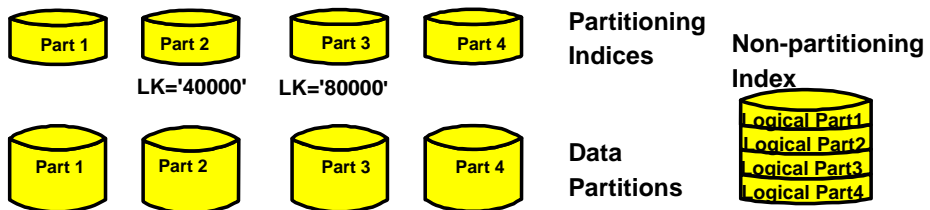
Rebalancing overview

Before REORG

- Sets new partition boundaries for even distribution of rows across the partitions being reorganized



After REORG TABLESPACE REBALANCE



What does rebalance do?

- Updates limit values and keys in the catalog
- Unloads rows in tablespace or range
- Sorts by clustering index and divides by number of parts
 - f* Won't be perfect if lots of duplicate keys exist
- Invalidates plans, packages and dynamic cache
- When clustering does not match partitioning, REORG must be run twice:
 - f* Once to move rows to right partition
 - f* Second to sort in clustering sequence

Rebalance vs. alter partitioning keys + REORG

- REORG . . . REBALANCE generates new partitioning values
 - f* One step automated process
 - f* Alter partition gives you more control to allow for future skewed growth
 - f* Consider when limit keys semantically meaningful

- REORG . . . REBALANCE leaves table space available
 - f* Alter part limit leaves affected partitions in REORP
 - f* But REBALANCE is not currently supported with SHRLEVEL CHANGE or REFERENCE

- Rebalance is not supported for table spaces with LOBs



Additional utility operations -- summary

- REBUILD INDEX: Scope Pending support and "INDEXSPACE" syntax.
- CHECK INDEX: can be run on partition of DPSI, or logical partition of NPSI
- CHECK DATA: When running on entire table space, sort must be done for DPSI keys. In basically all other cases, sort is avoided
- RUNSTATS: may be run against single partitions for DPSIs. Partition level statistics are used to update aggregate statistics for the entire table.
- Partition parallelism: DPSIs allow for totally concurrent operations with PART keyword, as do PIs
 - f* LOAD, REORG, REBUILD INDEX, CHECK INDEX
- Work data sets may require more space if there is a mixture of DPSIs and NPSIs

System planning and administration

- DSMAX -- number of data sets backing a secondary index could increase
- Catalog / directory growth
 - f* DPSIs:
 - SYSINDEXPART
 - SYSCOLDISTSTATS
 - SYSINDEXSTATS
 - DSNDB01.DBD01
 - f* For indexes which are COPY YES:
 - SYSLGRNX
 - SYSCOPY
 - f* For indexes whose HISTORY statistics are collected:
 - SYSINDEXPART_HIST
 - SYSINDEXSTATS_HIST
 - SYSCOLDISTSTATS_HIST
- Larger EDM pool requirements for DBDs
- Programs / queries that access SYSINDEXPART may need to change!!
- TYPE2 is deprecated keyword in syntax for CREATE / ALTER INDEX

Session Title: DB2 Partition Management and Utilities
Session 1346

Craig Frske
IBM
friske@us.ibm.com