IBM Software Group

DB2 for z/OS Version 8
Data Partitionning
Online Schema Evolution

Jeudi 17 Mars 2005

e business on demand software

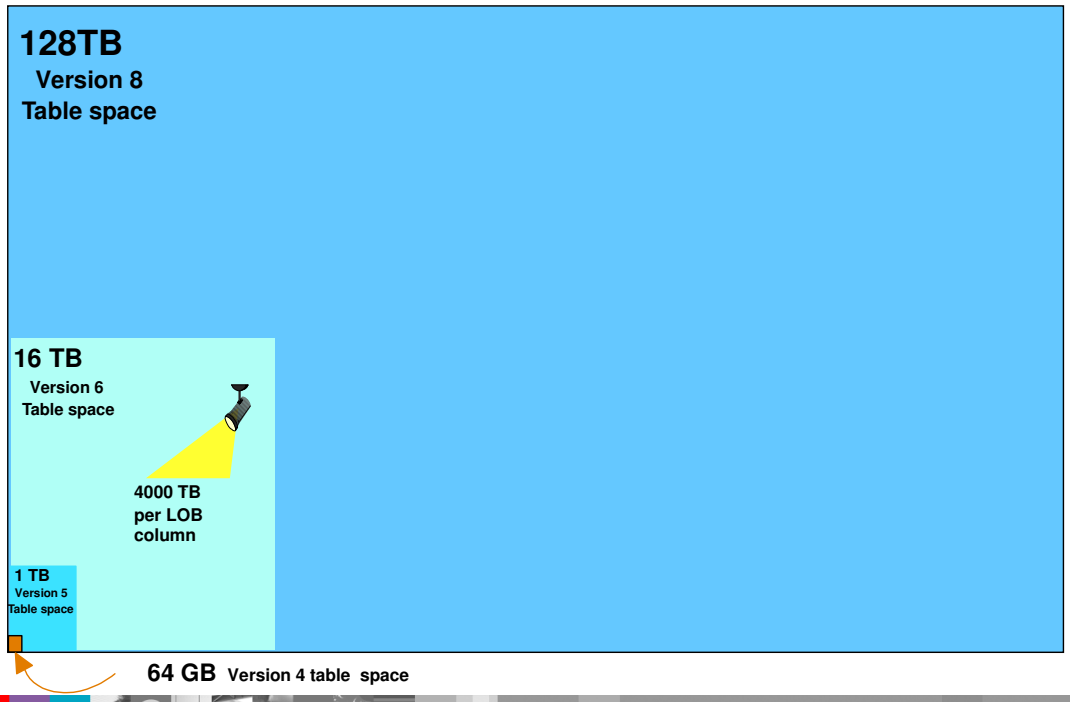
Catherine Chochoy
Technical Sales DB2 z/OS et Tools DB2
catherine_chochoy@fr.ibm.com

Agenda

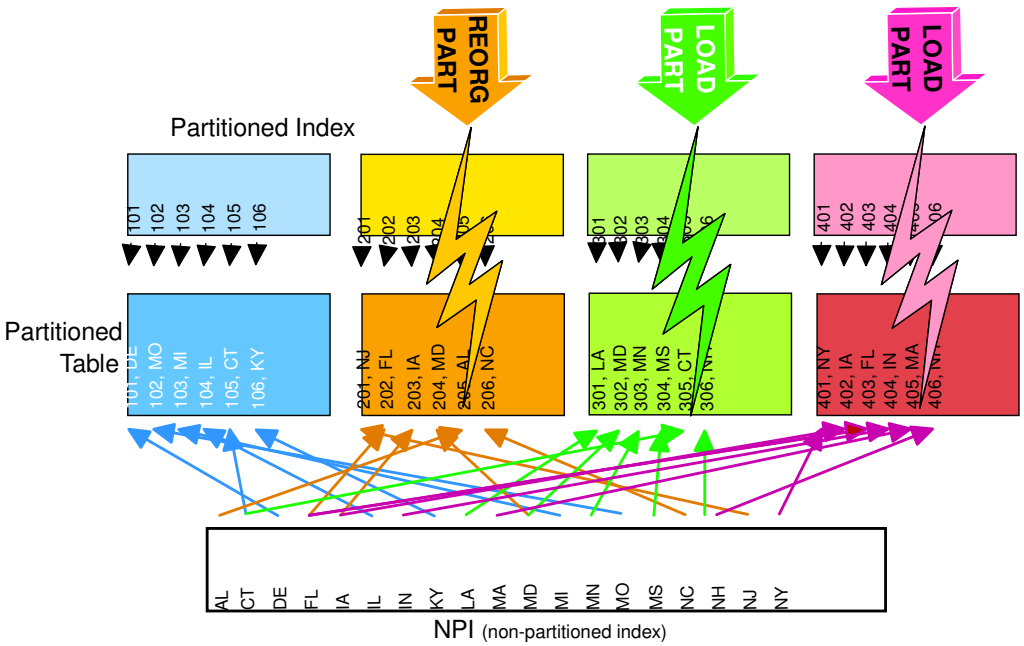
- Partitionning and clustering enhancements
- On line schema evolution
  - Changing table columns
  - Changing partition(ing) attributes
  - Changing index attributes
  - Summary

DB2 Objects Keep Getting Bigger!

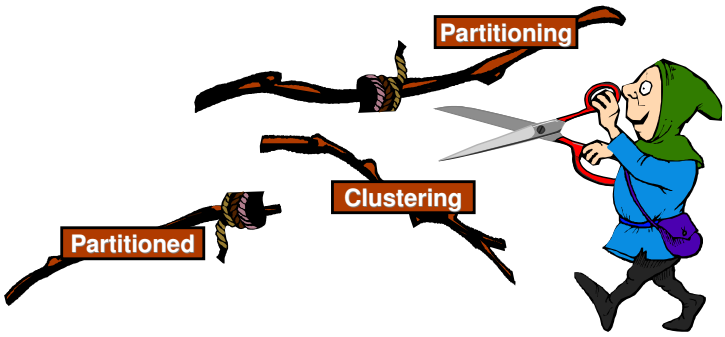
V4 >>>V8 TS size X 2048



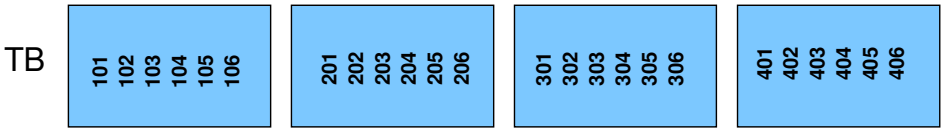
V7 - Contention on NPI



V8 Partitioned Tables - Table-Controlled Partitioning



No indexes are required for partitioning!!



Partitioned table

V8 - Creating Partitioned Tables

- CREATE TABLESPACE ***tsname*** NUMPARTS n
  (PARTITION 1 USING . . .
  . . .
  PARTITION n USING . . .)
  IN *dbname;*

```
CREATE TABLE CUSTOMER (
        ACCOUNT_NUM          INTEGER,
        CUST_LAST_NM         CHAR(30),
        . . .
        LAST_ACTIVITY_DT     DATE,
        STATE_CD             CHAR(2) )
PARTITION BY ( ACCOUNT_NUM       ASC )
        ( PARTITION 1  ENDING AT (199),
          PARTITION 2  ENDING AT (299),
          PARTITION 3  ENDING AT (399),
          PARTITION 4  ENDING AT (499) )
        IN dbname.tsname;
```

Definition is complete at this point!!

Converting to Table-Controlled Partitioning

- **No need to DROP/CREATE all existing partitioned tables**
- **DB2 will automatically convert to table-controlled partitioning for you when any of the following SQL statements are executed:**
  - DROP the partitioning index
  - ALTER INDEX NOT CLUSTER on the partitioning index
  - ALTER TABLE ... ADD PARTITION
  - ALTER TABLE ... ROTATE PARTITION
  - ALTER TABLE ... ALTER PARTITION n
  - CREATE INDEX ... PARTITIONED
  - CREATE INDEX ... ENDING AT ... omitting cluster keyword
- **Least disruptive approach**
  - ALTER INDEX xpi NOT CLUSTER of the (current) partitioning index
  - ALTER INDEX xpi CLUSTER of the same index

Catalog Support

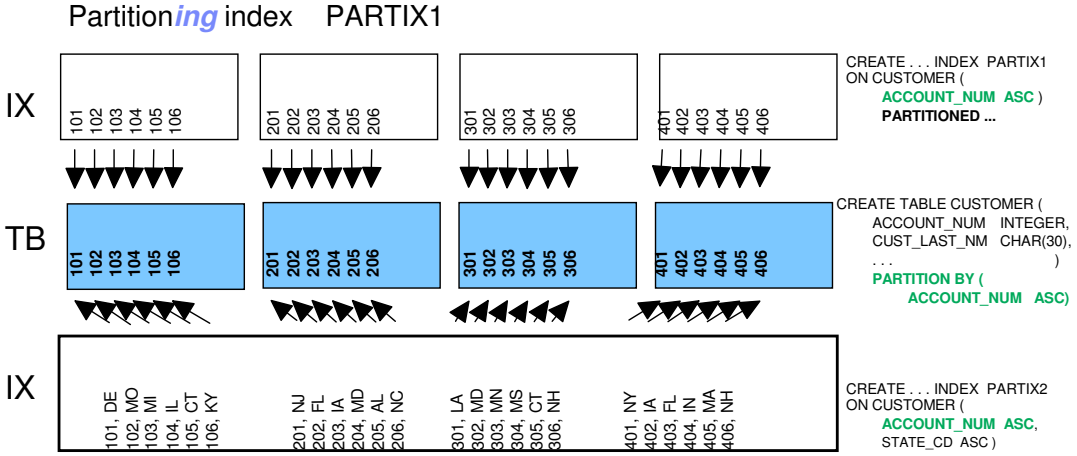
- SYSTABLES
  - PARTKEYCOLNUM
    - Number of columns in the partitioning key
- SYSTABLEPART
  - LIMITKEY_INTERNAL VARCHAR 512
    - Internal format of partition boundary corresponding with LIMITKEY
  - LIMITKEY - used for both index and table controlled partitioned tables
- SYSCOLUMNS
  - PARTKEY_COLSEQ
    - Column's numeric position in table's partitioning key
  - PARTKEY_ORDERING
    - Order of column in partitioning key ('A', 'D')
- SYSINDEXES
  - INDEXTYPE
    - 'P' for partitioning index on table controlled partitioned table
    - 'D' for DPSI
    - '2' type 2 (all....)

DB2 V8 Classification of Indexes

- **An index may / may not be correlated with the partitioning columns of the table**
  - Partitioning index (PI)
  - Secondary index
- **An index may / may not be physically partitioned**
  - Partitioned
  - Non-partitioned
- ***Clustering* index:**
  - Any index may be the clustering index!!
  - The clustering index can be unique / non-unique

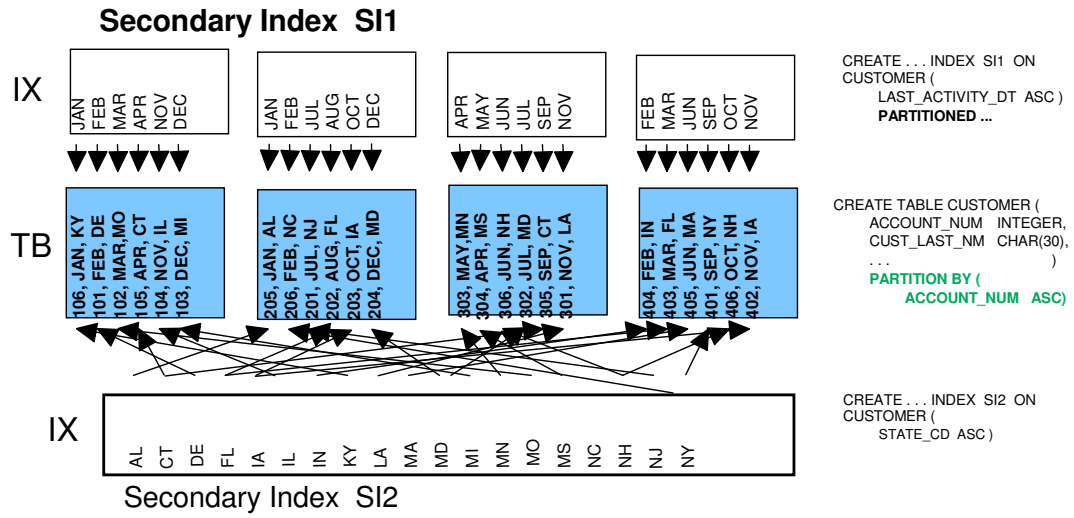
Partition*ing* Indexes

A partitioning index has the same left-most columns, in the same collating sequence, as the columns which partition the table



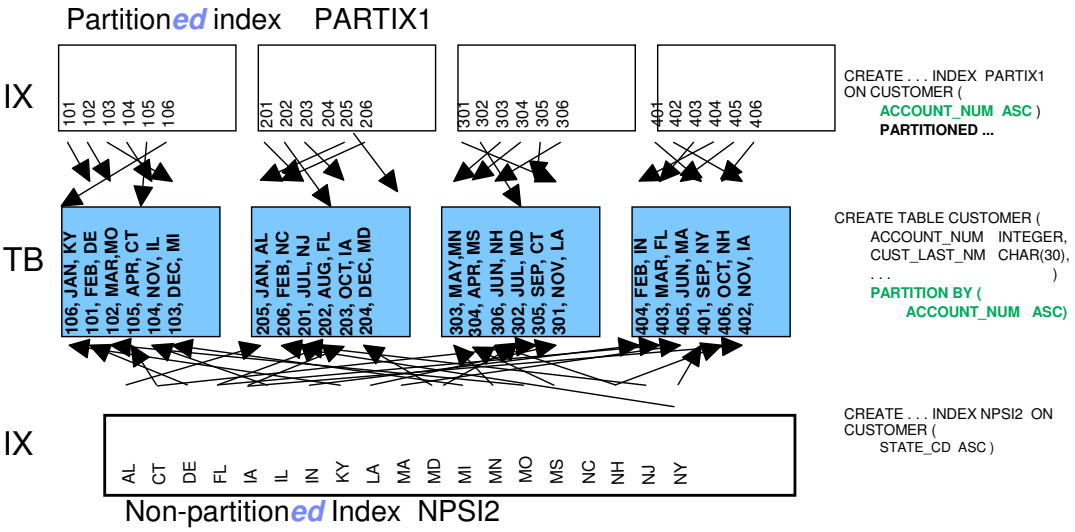
Secondary Indexes

A secondary index is any index which is **not** a partition***ing*** index



Partition***ed*** Index and Non-partition***ed*** Index

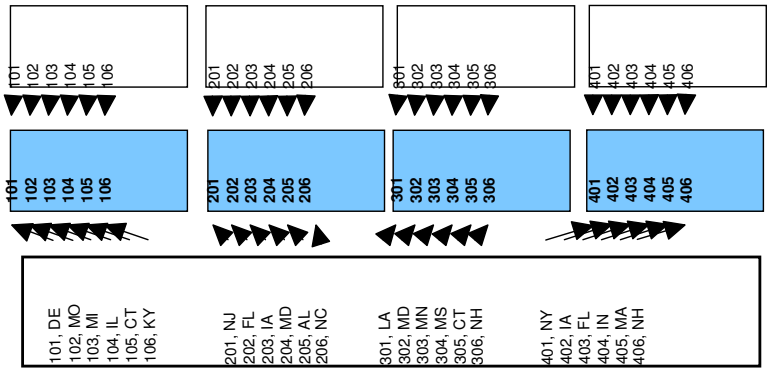
A partitioned index is any index which specifies the PARTITIONED keyword in the CREATE INDEX statement



Partitioning Indexes - Partitioned and Non-Partitioned

Partitioned Partitioning index PARTIX1

```
CREATE . . . INDEX  PARTIX1
   ON CUSTOMER (
      ACCOUNT_NUM  ASC )
   PARTITIONED
```

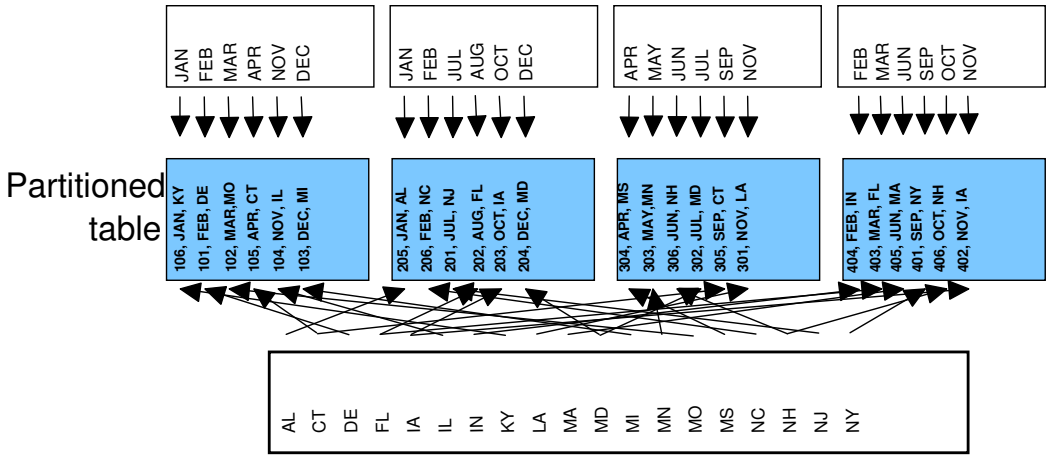


Non-partitioned Partitioning index PARTIX2

```
CREATE . . . INDEX  PARTIX2
   ON CUSTOMER (
      ACCOUNT_NUM ASC,
      STATE_CD ASC )
```

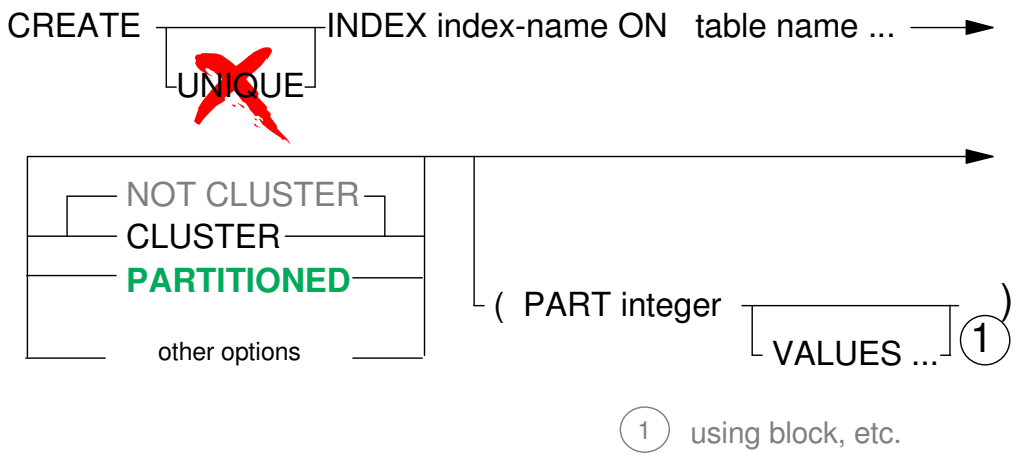

Partitioned and Non-partitioned Secondary Indexes

Data Partitioned Secondary Index (DPSI) -- DPSI1



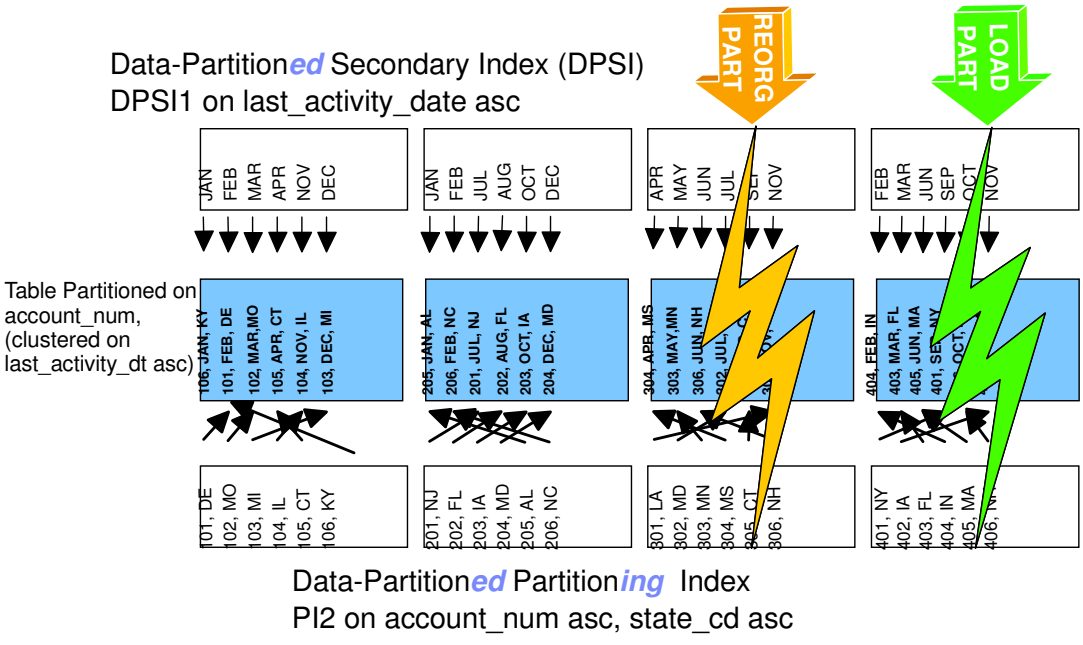
Non-Partitioned Secondary Index (NPSI) -- NPSI2

Creating a Data Partitioned Secondary Index




```
CREATE . . . INDEX dpsi1
    ON CUSTOMER
        (LAST_ACTIVITY_DT        ASC)
        . . .
    PARTITIONED
       USING STOGROUP mysto PQTY . . .
    . . .
```


DPSIs and Utility Operations



DPSI Query Performance

- Query performance characteristics of DPSIs
  - Allows query parallelism
  - Queries with predicates only on secondary index columns will need to scan all partitions
  - DB2 tries to do partition pruning -- The application needs to code explicit partitioning key predicates to allow for partition pruning when a DPSI exists
  - Chosen for queries with predicates on partitioning columns plus secondary index columns
- Secondary indexes
  - NPSIs
    - pro: favor sequential query performance
    - con: partition-level query or utility operations
  - DPSIs
    - pro: favor partition-level query or utility operation
    - con: sequential query performance, although well-suited to partition parallelism

Utility Operations - DPSIs

- CHECK DATA
  - When running on entire table space, sort must be done for DPSI keys. In basically all other cases, sort is avoided.
- CHECK INDEX
  - Can be run on partition of DPSI, or logical partition of NPSI
- RUNSTATS
  - May be run against single partitions, including 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
  - LOAD, REORG, REBUILD INDEX, CHECK INDEX
- Work data sets may require more space if there is a mixture of DPSIs and NPSIs

Design Considerations - Initial Thoughts

- Predicates - are there predicates on partitioning columns?
- Clustering - what are the "large" processes against the data?
- ORDER BY - what is the desired sequence?
- DPSIs are non-unique - query might have to scan all parts, if no predicates on partitioning column(s)
- Frequency of index maintenance (INSERT / UPDATE / DELETE / utilities) against DPSI / NPSI
- Is number of parts on table a consideration for DPSI / NPSI? (for example, 4096 parts)
- Online REORG - frequency any different?
- Considerations for add / rotate partition?

Clustering Indexes

Any index can be the clustering index

The CLUSTER keyword is now optional when creating a partitioning index

Ordering of rows for INSERT and REORG:

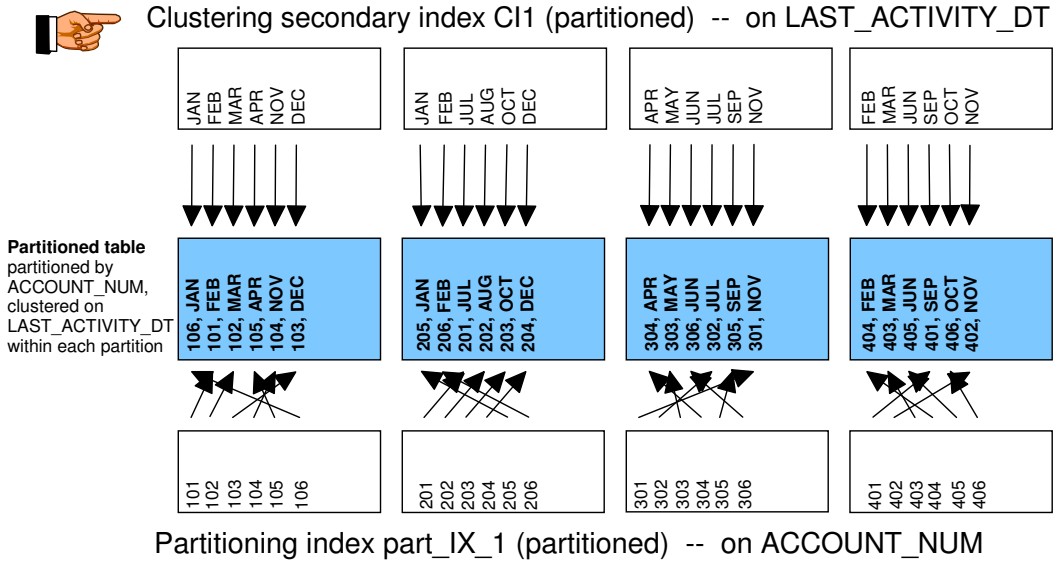
- Partitioning columns determine the proper partition for the row
- The clustering index determines the location within the partition (assuming availability of space at that location)

The clustering index can be changed using ALTER INDEX

Steps to change the clustering index

- ALTER INDEX index1 NOT CLUSTER
- ALTER INDEX index2 CLUSTER
- (Followed by REORG to change the sequence of existing rows)

Clustering Data Partitioned Secondary Index



Agenda

- Partitionning and clustering enhancements
- On line schema evolution
  - Changing table columns
  - Changing partition(ing) attributes
  - Changing index attributes
  - Summary

The availability story so far...

- Code maintenance (installation & PTFs)
  - Data sharing (V4)
- Data maintenance
  - Online RUNSTATS, COPY, REORG, and LOAD (V3-V7)
- Application maintenance
  - Packages, versioning
- Schema maintenance (ALTER, CREATE, DROP)
  - Simple ALTER (primary quantity, etc.), add column, rename table already supported
  - Online schema evolution for tables, partitions and indexes (V8)

ADDRESS THE MAIN AVAILABILITY ISSUE CUSTOMERS FACE

To change a definition today....

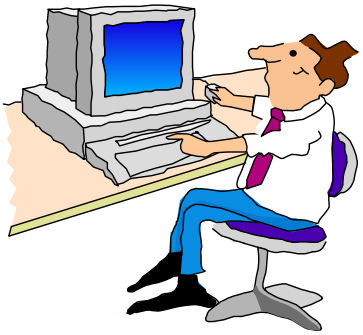
- Unload data
- Drop base tables
- Create base tables
- Create indexes
- Create views
- Define authorizations
- Load data (Copy, Stats, Check)
- Rebind

data availability

DATA UNAVAILABILITY
(minutes, hours, days?)

data availability

Changing Table Columns



Altering column data types h

- CHAR(10) to CHAR(20)
- DEC(5,0) to DEC(10,0)
- CHAR(10) to VARCHAR(40)
- INT to DEC(10,0)
- CHAR(30) to VARCHAR(30)

- CHAR(20) to CHAR(10)
- SMALLINT to DEC(3,0)

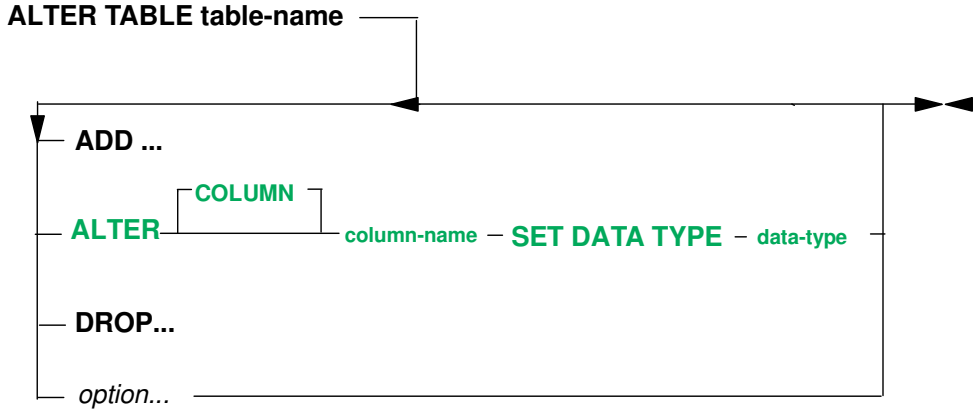
Note: new column definition must be large enough to hold maximum value possible for original column

Supported ALTER ... DATA TYPEs

FROM DATA TYPE	TO DATA TYPE
smallint	integer
smallint	real
smallint	double
smallint	>= decimal(5.0)
integer	double
integer	>=decimal(10,0)
float(4) or real	float(8) or double
<=decimal(15,n)	double
decimal(n,m)	decimal(n+x,m+y)
char(n)	char(n+x)
char(n)	varchar(n+x)
varchar(n)	char(n+x)
varchar(n)	varchar(n+x)
graphic(n)	graphic(n+x)
graphic(n)	vargraphic(n+x)
vargraphic(n)	vargraphic(n+x)
vargraphic(n)	graphic(n+x)

no unique index

ALTER ... DATA TYPE syntax



Example:
ALTER TABLE CUST
ALTER COLUMN LASTNAME
SET DATA TYPE CHAR(40)

Where LASTNAME is currently CHAR(30)

What happens to the table?

- New definition or version is captured in catalog and directory
  - support for 256 concurrent versions per table space
- Existing data remains **unchanged**
- On SELECT, data will be materialized to the new format
- On INSERT / UPDATE, the entire row will be changed to latest format
- REORG changes all rows to the current version

Any other table related changes?

- Table space is placed in AREO* (advisory REORG-pending)
  - Accessible, but some performance degradation until reorg
- Plans, packages and cached dynamic statements referring to the changed column are invalidated
- Runstats values are updated or invalidated
  - e.g. HIGH2KEY, LOW2KEY, frequency stats
- Default values are regenerated.
- Check constraints are regenerated.

A word on system pages...

- V8 system pages contain dictionaries and version information
- Make objects self-defining
  - Stores version information relevant to data in pageset or partition which can be used by tools or utilities like UNLOAD
- Included in incremental copy with SYSTEMPAGES YES
- SYSOBDS contains version 0 history information
  - Used for PIT recovery where information is not in system page

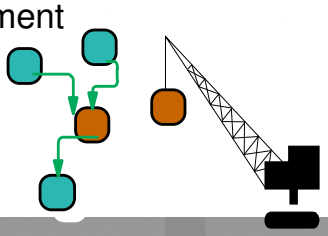
What happens to any dependent indexes?

E.g., if indexes exist on LASTNAME and CUSTNO, LASTNAME ...

- New version created for indexes that reference altered column
  - Up to 16 versions per index
- Immediate access for character data type extensions
  - Index placed in AREO*
- Delayed access for numeric data types
  - Index placed in rebuild pending (RBDP)
    - Deletes are allowed
    - Updates and inserts are allowed for non-unique indexes
    - Dynamic queries are allowed (will avoid RBDP indexes)

What about views?

- Views referencing affected columns are regenerated
  - Ordered regeneration with underlying view done first
  - SYSVTREE and SYSVLTREE searched for dependencies
  - SYSCOLUMNS is updated for the new view definition
  - All plans, packages and dynamic cached statement referencing affected views will be invalidated



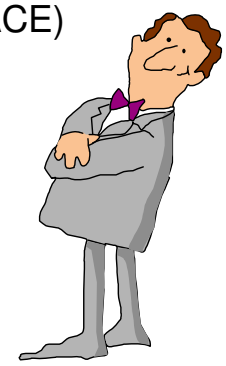
IBM Software Group

Some restrictions

- Data types must be compatible and lengths must be the same or longer
- Disallowed for ROWIDs, DATEs, TIMEs and FOR BIT DATA columns
- Data types and lengths cannot be altered when
  - Column is part of a referential constraint
  - Column is an identify column
  - Column has a FIELDPROC
  - Part of a materialized query table
  - An EDITPROC or VALIDPROC exists on the table

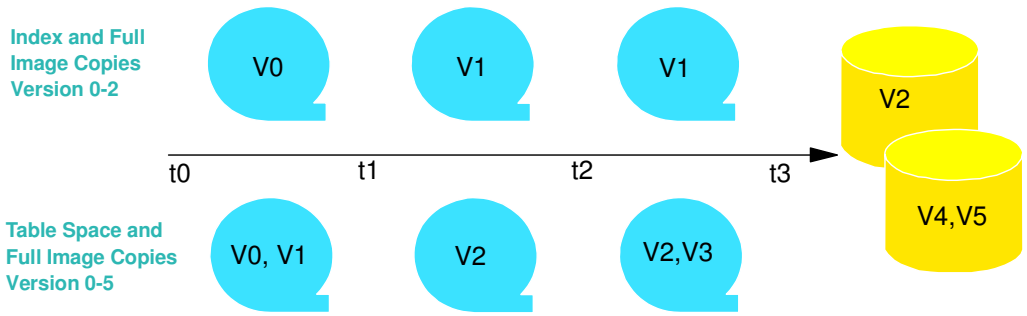
Planning Considerations

- Assess what programs need to change
  - Host variables may need to be extended to handle the extra length
- Schedule ALTERs before a REORG with inline statistics to minimize performance degradation (or LOAD REPLACE)
  - Invalidates cache to revert to optimal access path
- If REORG not run, collect statistics with RUNSTATS
- Bind or automatic rebind plans/packages



Operational Impact

- Recovering data
  - Whether to current or PIT there is no problem as the image copy and SYSOBDS contains version information
  - Log processing will insert and update the proper format
  - Note: it is recovery of data rather than schema



Catalog support for versioning

Oldest for pageset and all available copies - updated by MODIFY

Used to allocate next version number by ALTER

	OLDEST_VERSION	CURRENT_VERSION	VERSION
SYSTABLESPACE	0	5	
SYSTABLEPART	0		
SYSTABLES			5
SYSINDEXES	0	2	2
SYSINDEXPART	0		
SYSCOPY	0, 2, 2 / 0,1,1		

- Versioning is tracked at table space and index level
  - Each table in a segmented table space may be assigned different version numbers

IBM Software Group

Managing Versions

- Up to 256 active versions for table space, up to 16 for indexes
  - Active versions include those in pageset and all image copies
  - Unaltered objects remain at version 0
- Combine ALTERs within a single unit of work (one version)
  - ALTER and DML disallowed in the same unit of work
- REORG and MODIFY before reaching max of 256 versions
  - MODIFY to delete image copies, update lowest_version
  - versions increment 1-255, 1-15, then cycle back to 1 again

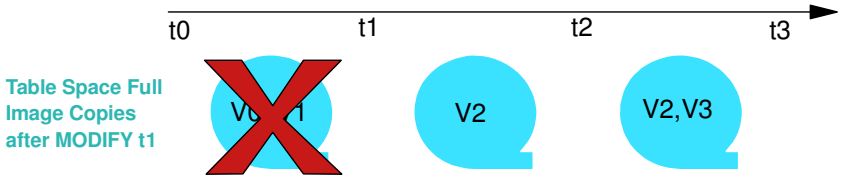
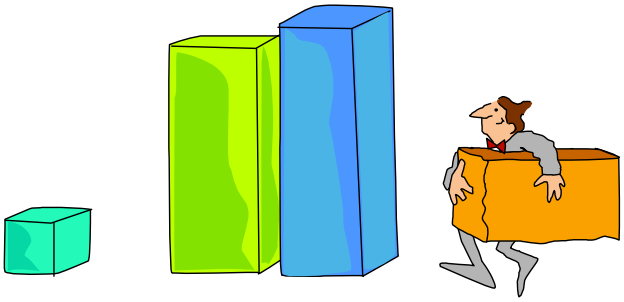
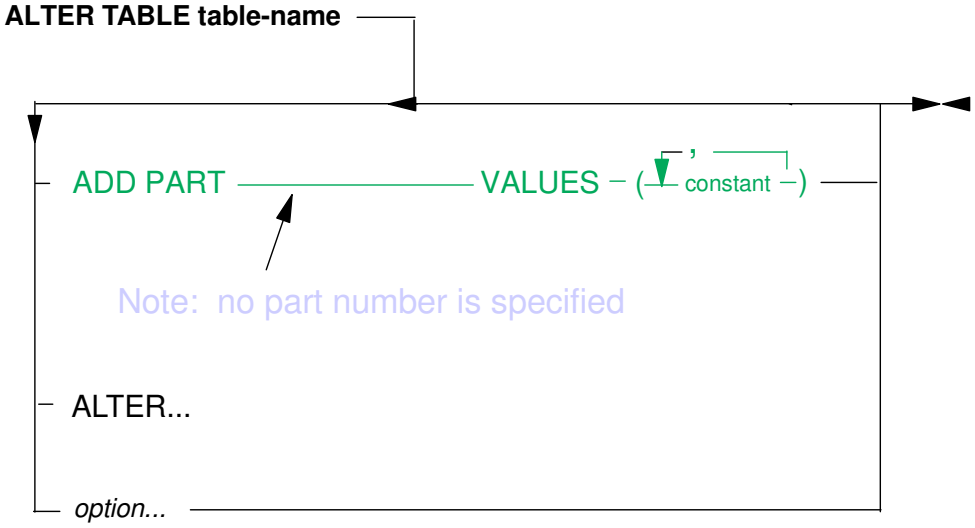


Table Space Full Image Copies after MODIFY t1

Changing Partition Attributes

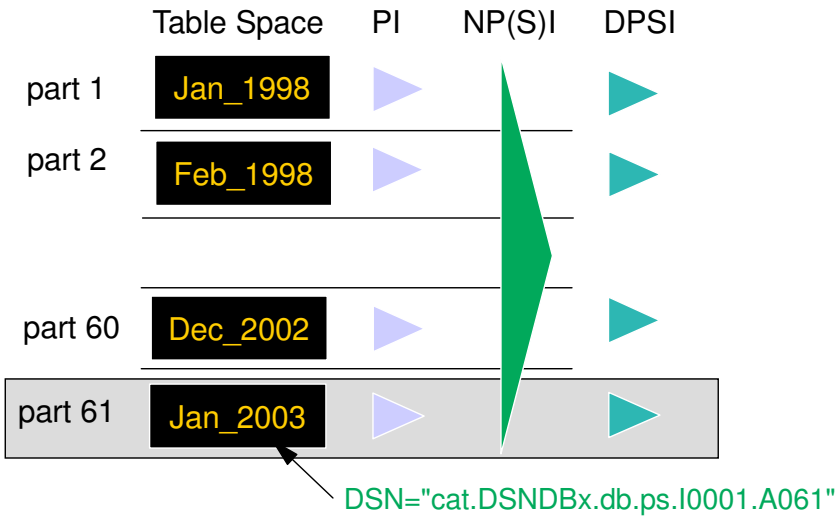


Add partition syntax



Add Partition Example

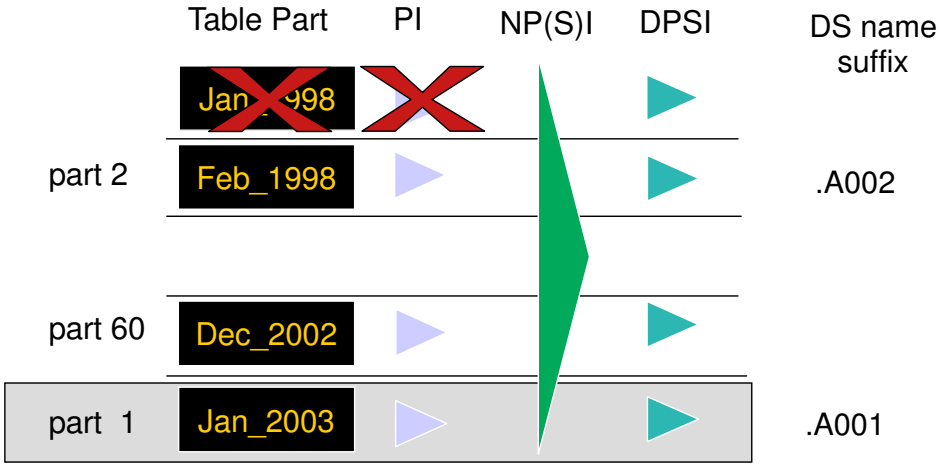
ALTER TABLE . . .ADD PART VALUES("01/31/2003");



Operational Considerations

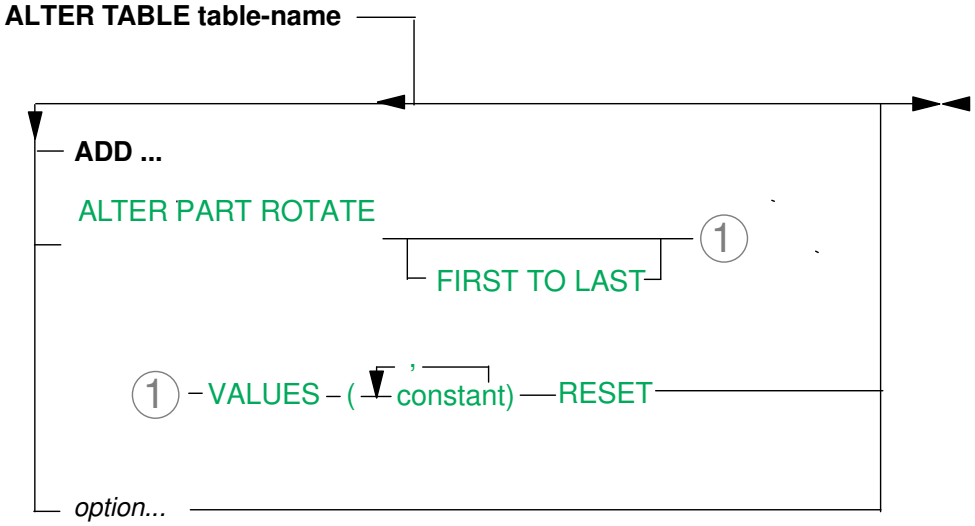
- Immediate availability
  - Stop of table space and partitioned indexes required before adding partition
  - Table is quiesced and plans, packages and cached statement are invalidated
    - Necessary as access path may be optimized to read only certain partitions
- Next physical partition number is assigned
  - Maximum number of partitions based on table definition
- Attributes of previous logical partition are used, e.g., PRIQTY
  - 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



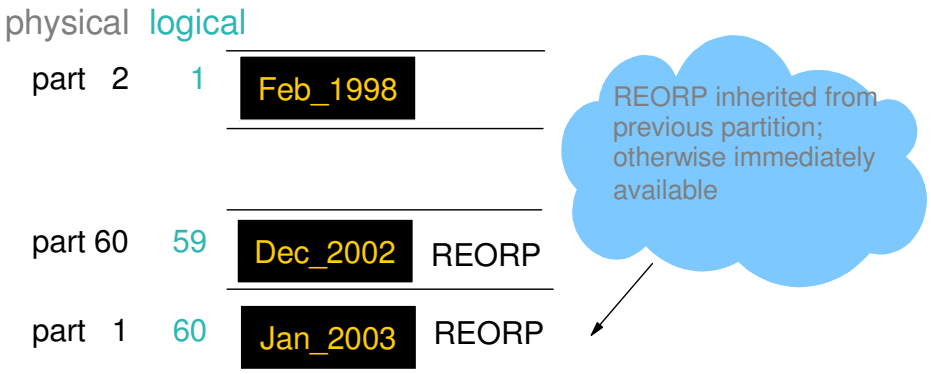
- Old data is rolled away and partition reused

Rotate Partition Syntax



Rotate Partition Example

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



Adding and rotating parts can mix up the partition order

See SYSIBM.SYSTABLEPART LOGICAL_PART and PARTITION

IBM Software Group

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
  - Consider impact on logging and performance
  - 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

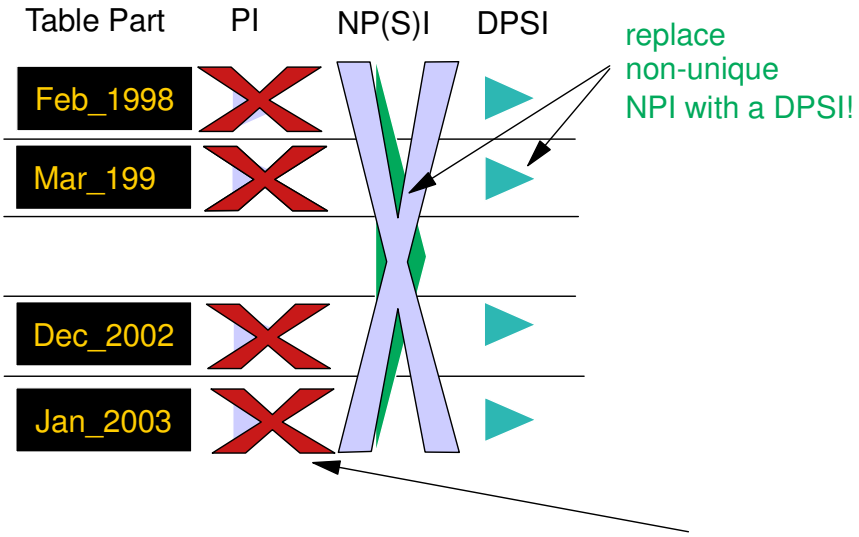
DISPLAY command

- After using ROTATE, the logical order of partitions will be different from the physical order

NAME	TYPE	PART	STATUS
tsn	TS	2	RW
tsn	TS	3	RW
tsn	TS	4	RW
.....			
tsn	TS	59	RW
tsn	TS	60	RW,REORP
tsn	TS	1	RW,REORP

SYSTABLEPART and SYSCOPY have new LOGICAL_PART column

Drop Partitioning Index



- When PI created only for partitioning, not for access... Drop It!

Table-Controlled Partitioning

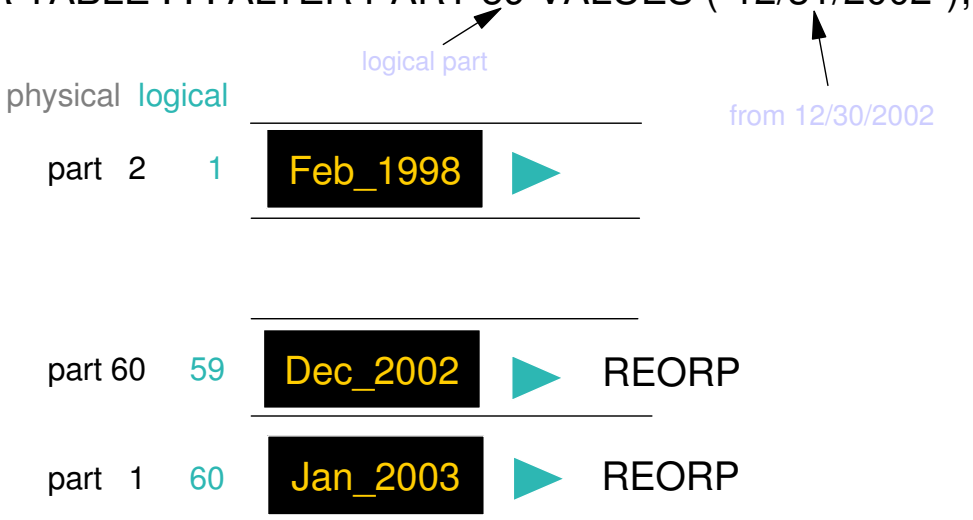
- No dependence on indexes (index-controlled partitioning)
- Managed with new catalog columns ( )

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

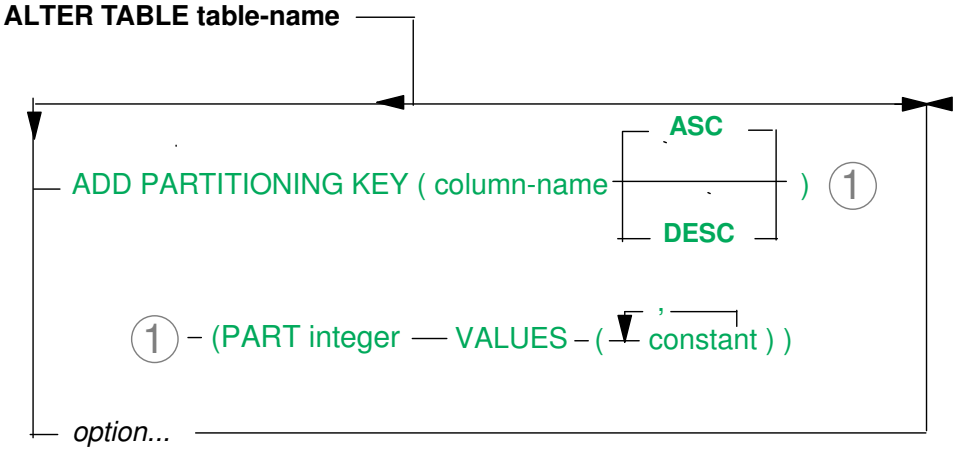
Alter Partition Boundary Example

ALTER TABLE . . . ALTER PART 59 VALUES ("12/31/2002");



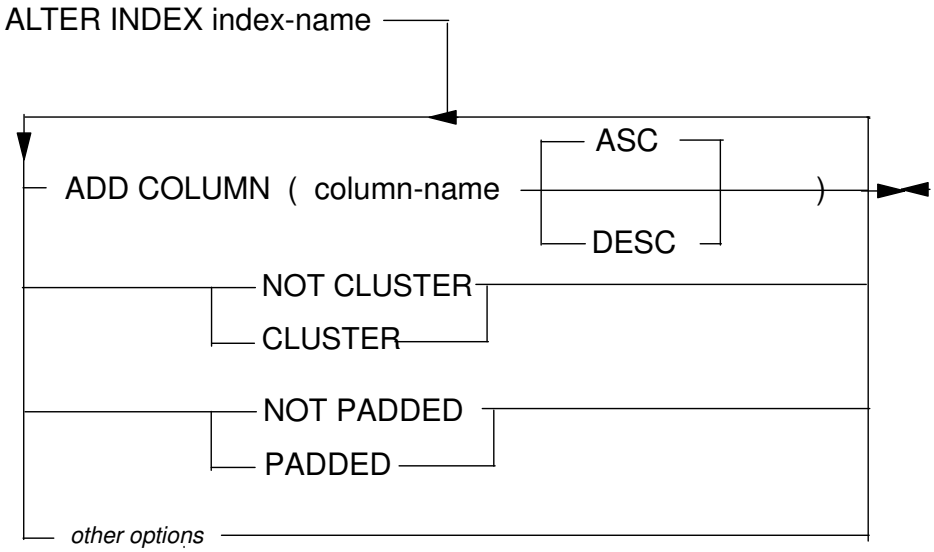
Add Partitioning Key Syntax

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



Changing Index Attributes

Altering index attributes



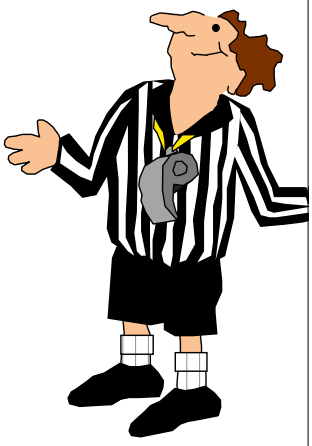
Alter index add column

- Ability to add a column to the end of an index
  - Creates a new version
- When column preexists in the table index is placed in RBDP
- If it's a **new column in the table**, add it to the table and index in the **same UOW**
  - E.g., ALTER TABLE CUST
  - ADD COLUMN NEW_COL;
  - ALTER INDEX CUST_IDX
  - ADD COLUMN NEW_COL ASC;
  - COMMIT;

IBM Software Group

Restrictions

- Cannot exceed 64 columns in an index
- Length maximum
  - 2000-n for padded, where n is #nullable columns
  - 2000-n-2m, where, where m is #varying columns
- Disallowed for
  - System defined indexes
  - Partitioning indexes (index-controlled)
  - Indexes enforcing a primary key unique constraint



Alter clustering attribute of indexes

- Clustering has been unbundled from partitioning
  - A move to table-controlled rather than index-controlled partitioning
  - A partitioning index does not have to be the explicit clustering index

- Change clustering Index with two steps
  1. ALTER INDEX index1 NOT CLUSTER
     Will continue to be used until new clustering index is defined
  2. ALTER INDEX index2 CLUSTER
     Immediate effect -- inserts follow new clustering but needs REORG!

Alter index not padded / padded

- Creates a new index version
- ALTER INDEX PADDED sets index to AREO*
  - Reset by REORG, LOAD REPLACE, or REBUILD
- ALTER INDEX NOT PADDED sets index to ARBDP
  - Reset by REORG TABLESPACE, LOAD REPLACE or REBUILD INDEX
  - Index must be rebuilt from data
  - Optimizer may choose index for index only access
  - Recover to PIT may set ARBDP

Index Availability Review

- Setting RBDP invalidates plans/packages and flushes dynamic cache
- DB2 allows deletes and some inserts if indexes are in RBDP.
- Optimizer will avoid indexes as follows:
  - Static BIND
    - Indexes in ARBDP are avoided for index only access
    - Indexes in RBDP may be chosen (possible resource unavailable)
  - Dynamic PREPARE
    - Indexes in RBDP avoided, ARBDP avoided for index only
  - Cached
    - If cached, PREPARE is bypassed
    - Flushing occurs when index set in RBDP or reset from RBDP
    - RUNSTATS UPDATE NONE REPORT NO flushes cache too
  - Reoptimization
    - Acts the same as initial BIND or PREPARE

To summarize...

- V8 Schema enhancements provide a host of availability and management enhancements with regard to tables and their indexes
- Ability to ALTER table columns and indexes rather than DROP and CREATE
- Improved management of partitioned table spaces and their indexes
  - Unbundling of partitioning and clustering
  - Ability to add, rotate and rebalance partitions
  - CREATE TABLE... LIKE flexibility