



IBM Software Group

DB2 Information Management Technical Conference

Session - Z21 Online Schema Evolution... ALTER Extensions in V8

DB2 Information Management Technical Conference

Craig Friske - IBM DB2 for z/OS Development

friske@us.ibm.com

DB2 Information Management Software



 business on demand software

Agenda

- Historical Perspective Overview
- 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)



To change a definition today....

- Unload data
- Drop base tables
- Create base tables, 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

- 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

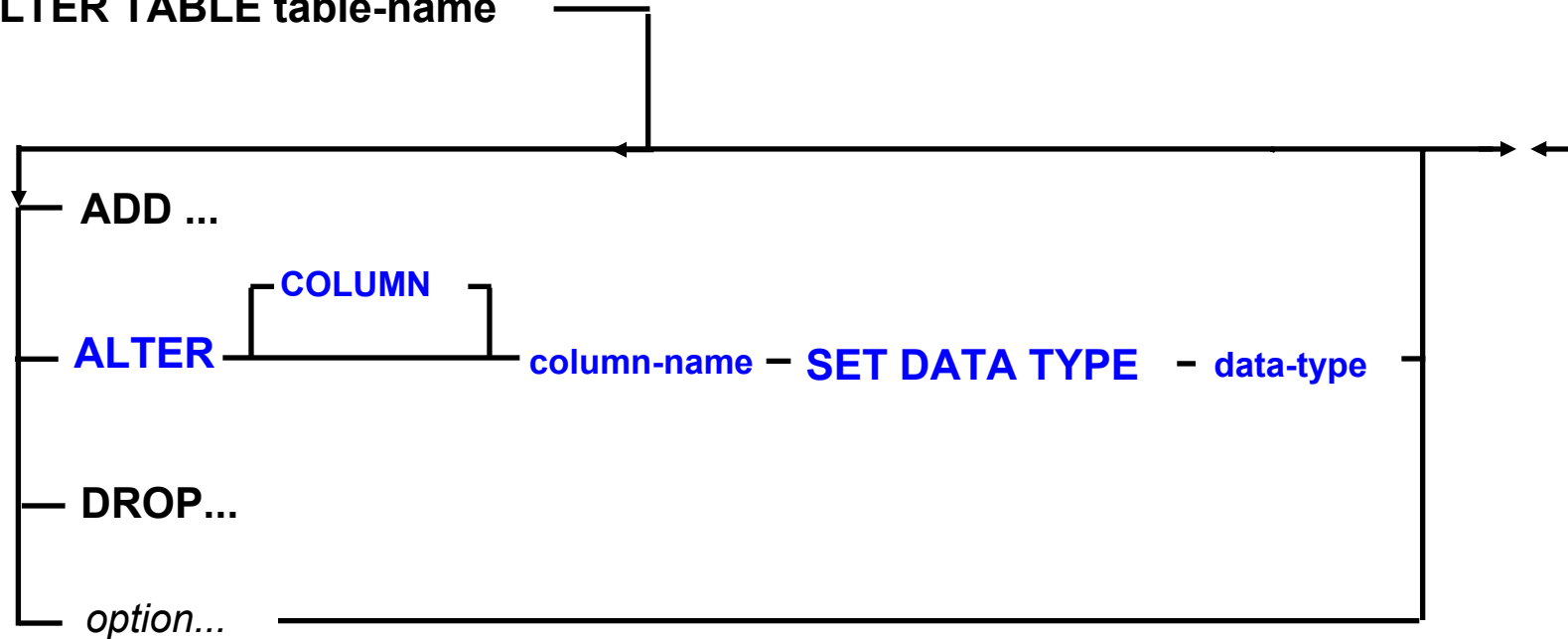
FROM DATA TYPE	TO DATA TYPE
smallint	integer
smallint	real
smallint	double
smallint	>= decimal(5.0)
integer	double
integer	>=decimal(10,0)
float(4) or real	float(8) or double
<=decimal(15,n)	double
decimal(n,m)	decimal(n+x,m+y)
char(n)	char(n+x)
char(n)	varchar(n+x)
varchar(n)	char(n+x)
varchar(n)	varchar(n+x)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

Unique index not allowed



ALTER ... DATA TYPE syntax

ALTER TABLE table-name



Example:

```
ALTER TABLE CUST
```

```
  ALTER COLUMN LASTNAME
```

```
    SET DATA TYPE VARCHAR(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
- Default values are regenerated
- Check constraints are regenerated.
- Views are regenerated
- Runstats values are updated or invalidated
 - e.g. HIGH2KEY, LOW2KEY, frequency stats



**Tablespace
is still
available
(AREO*)**

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)



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 identity column
 - Column has a FIELDPROC
 - Part of a materialized query table
 - An EDITPROC or VALIDPROC exists on the table



Planning Considerations

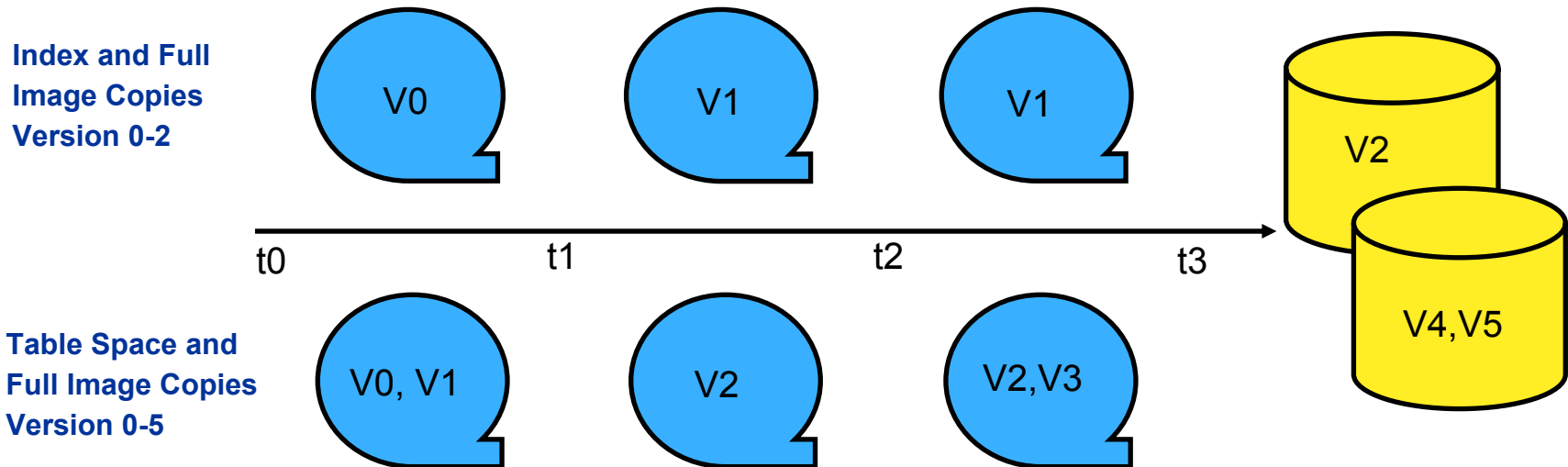
- Assess what programs need to change
 - Host variables may need to be extended to handle the extra length
- Schedule ALTERs before a REORG with inline statistics to minimize performance degradation (or LOAD REPLACE)
 - Invalidates cache to revert to optimal access path
- If REORG not run, collect statistics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

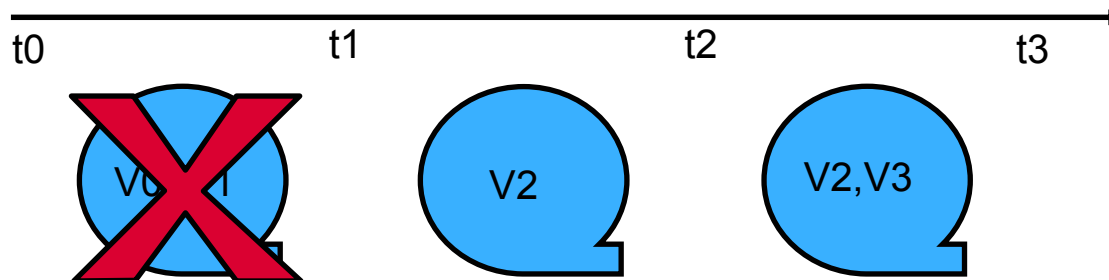
	OLDEST_VERSION	CURRENT_VERSION	VERSION
SYSTABLESPACE	0	5	
SYSTABLEPART	0		
SYSTABLES			5
SYSINDEXES	0	2	2
SYSINDEXPART	0		
SYSCOPY	0, 2, 2 / 0,1,1		

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

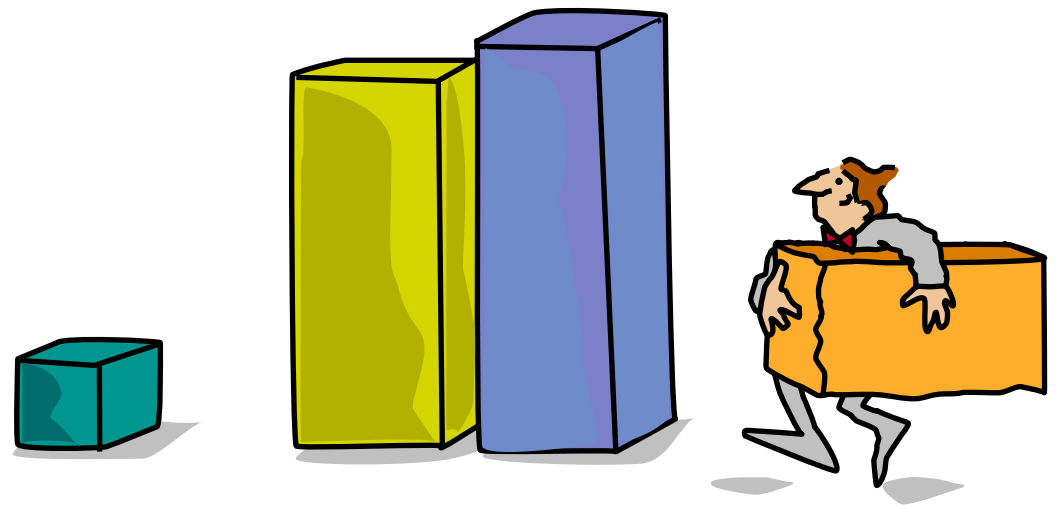
Managing Versions

- 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

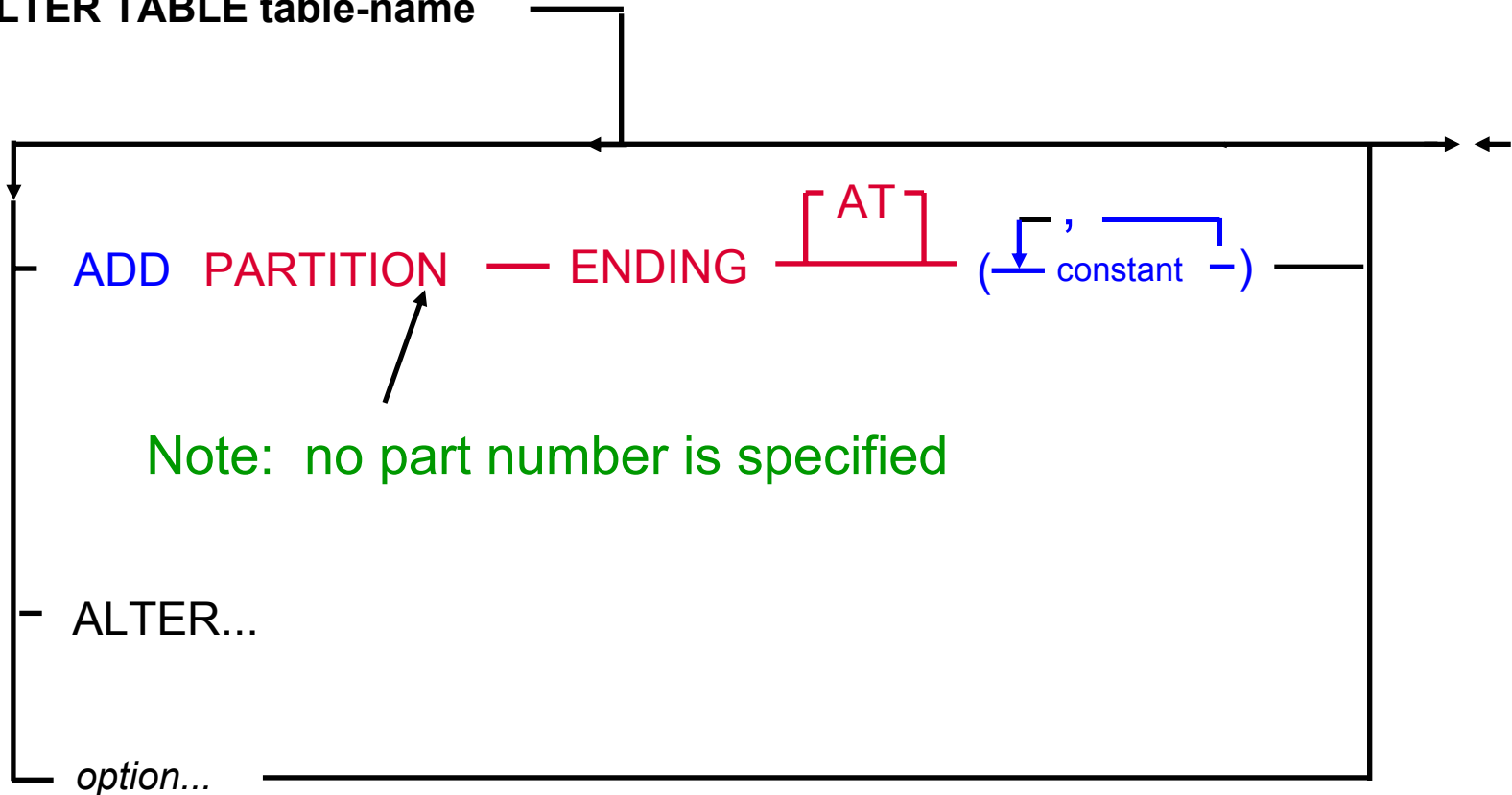


Changing Partition Attributes



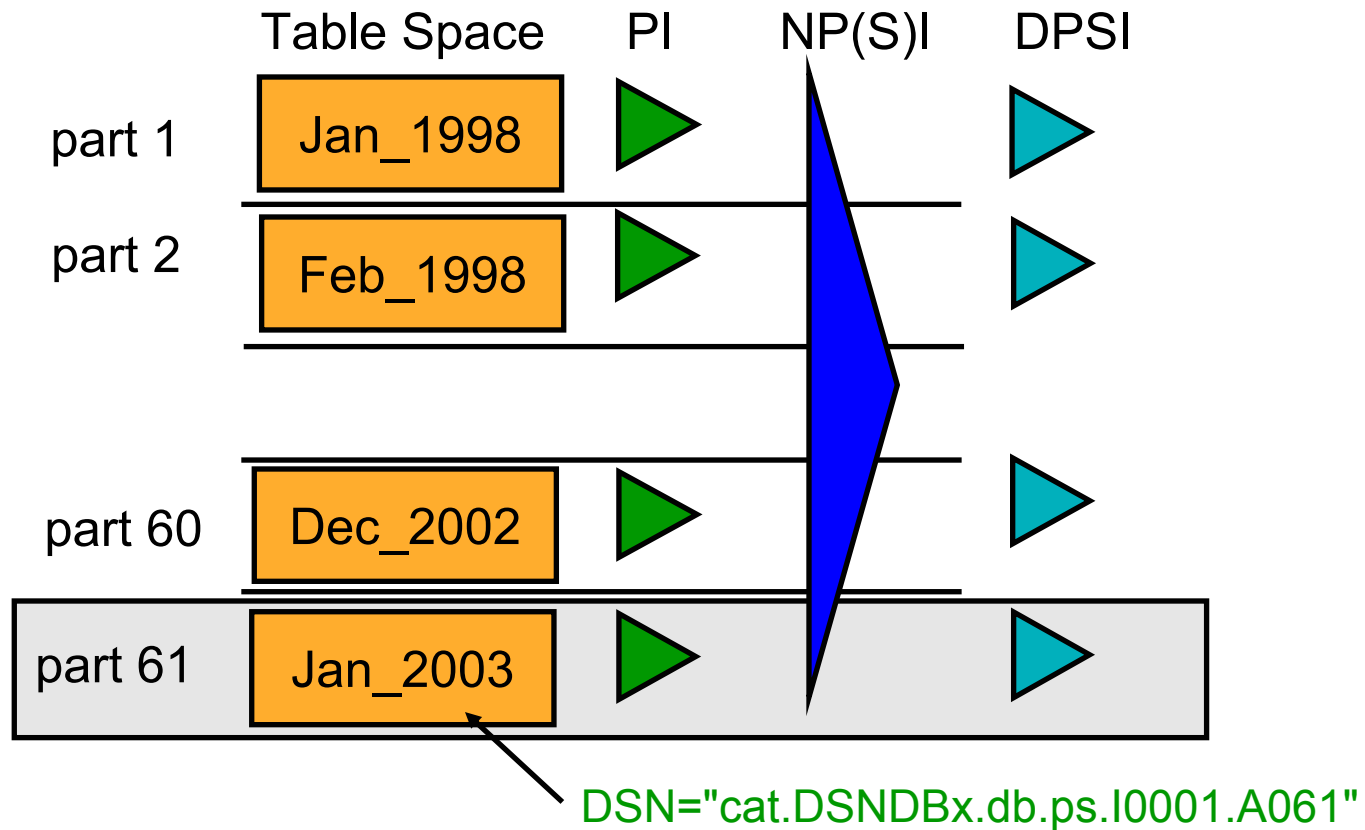
Add partition syntax

ALTER TABLE table-name



Add Partition Example

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

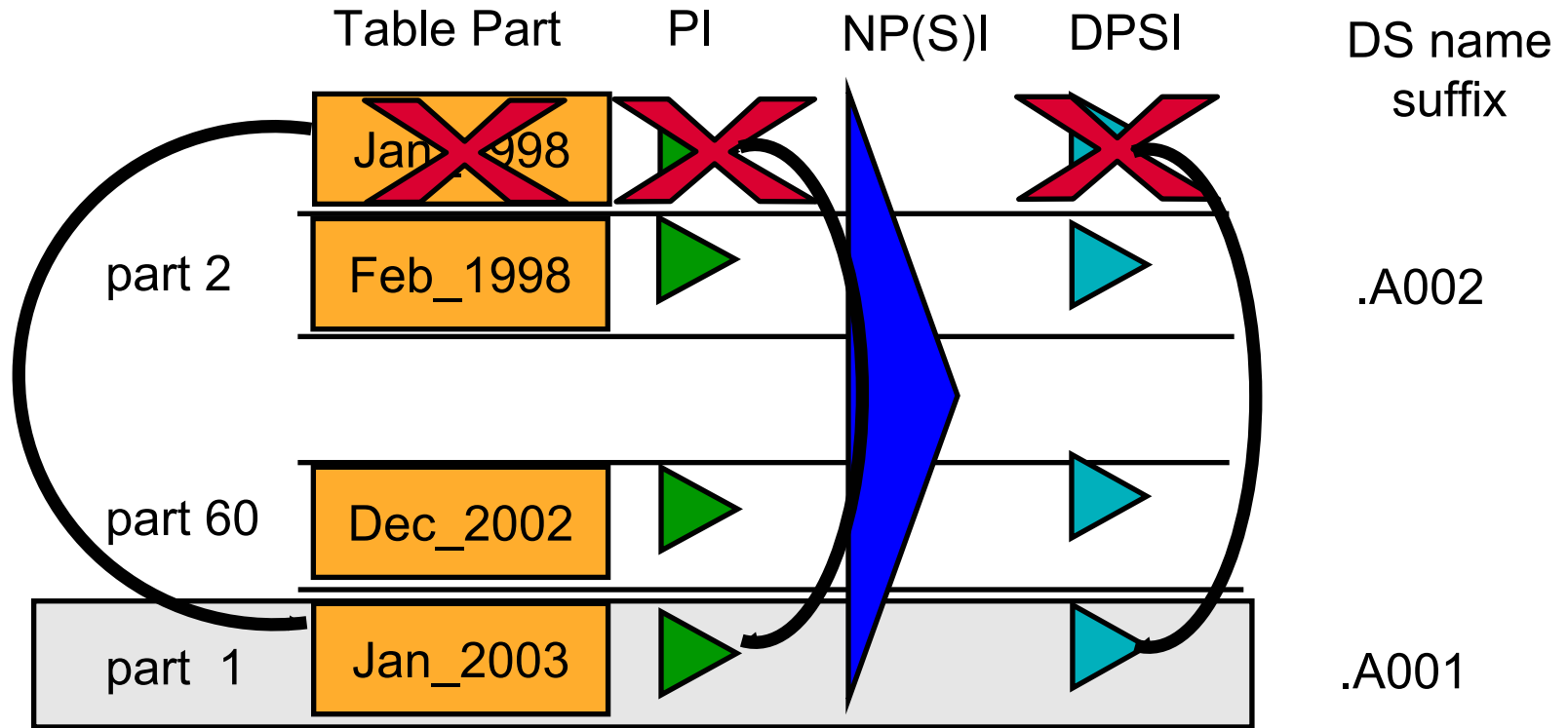


Operational Considerations

- Immediate availability
 - 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

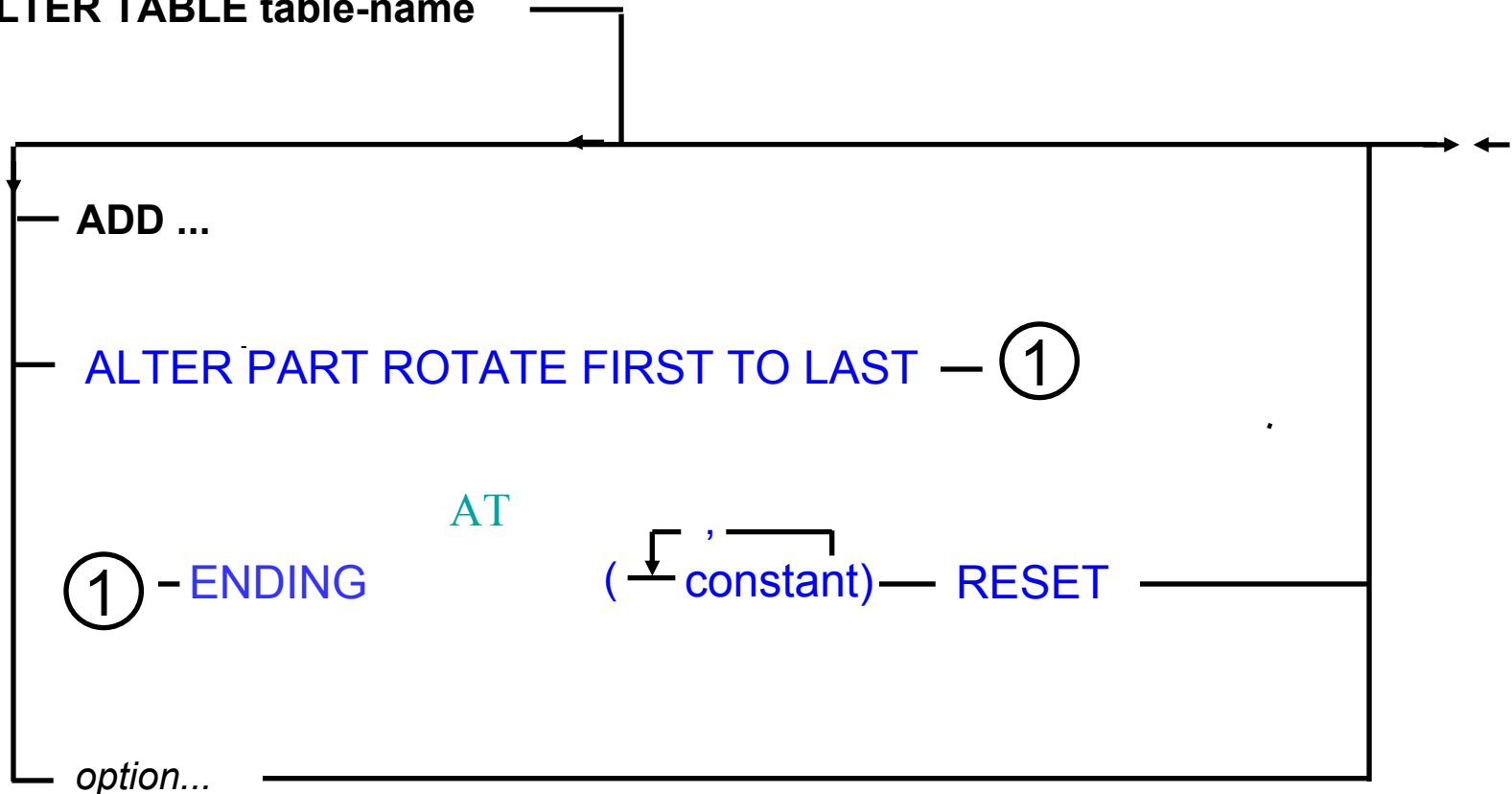


Old data is rolled away and partition reused



Rotate Partition Syntax

ALTER TABLE table-name



Rotate Partition Example

```
ALTER TABLE ... ALTER PART ROTATE ...ENDING ("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;

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

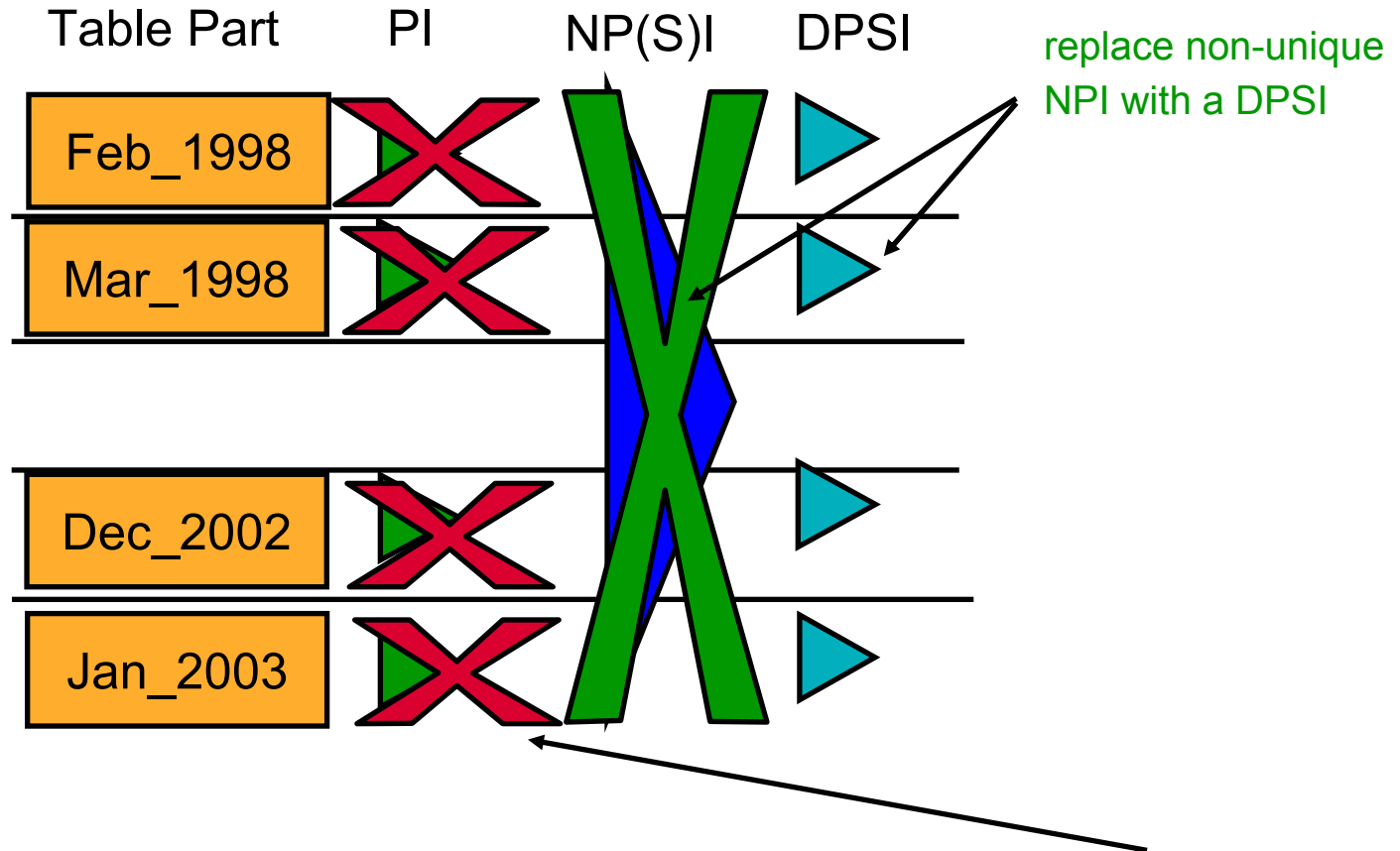


Operational Considerations

- Data is deleted so you may wish to unload it first
- Rotate will issue individual deletes
 - 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








Drop Partitioning Index



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

Table-Controlled Partitioning

- No dependence on indexes like index-controlled partitioning
- New catalog columns to help manage ()

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



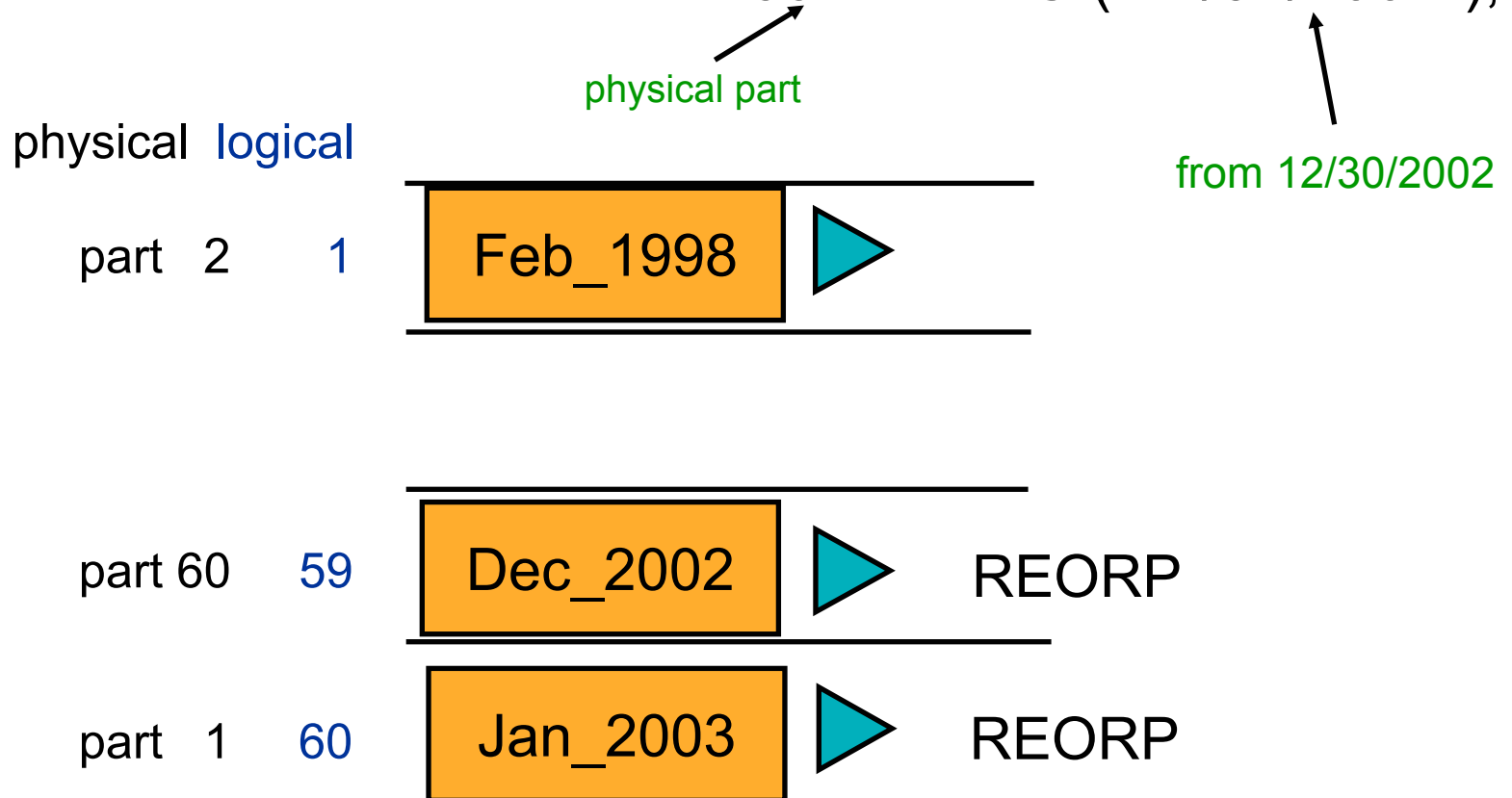
Converting to table controlled partitioning

- 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**



Alter Partition Boundary Example

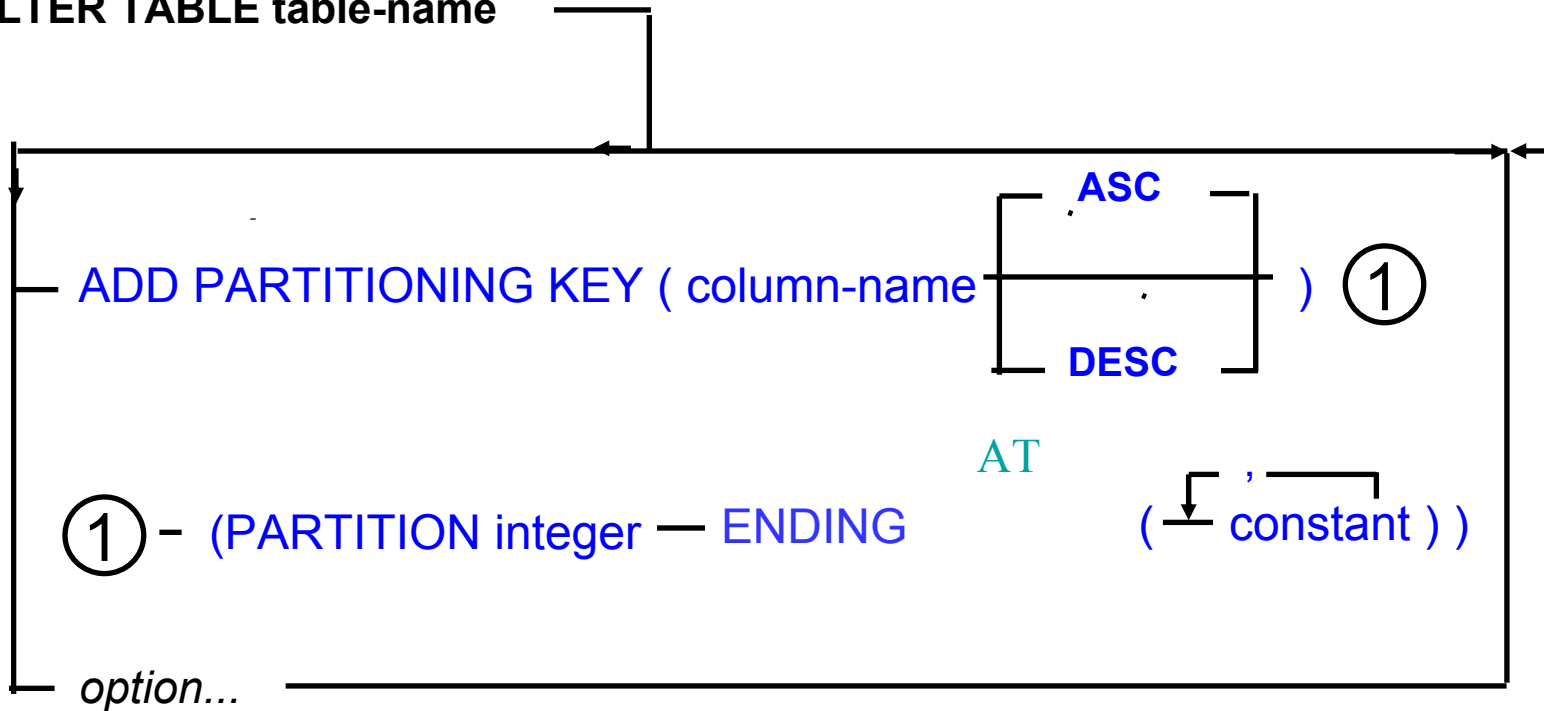
ALTER TABLE . . . ALTER PART 59 ENDING ("12/31/2002");



Add Partitioning Key Syntax

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

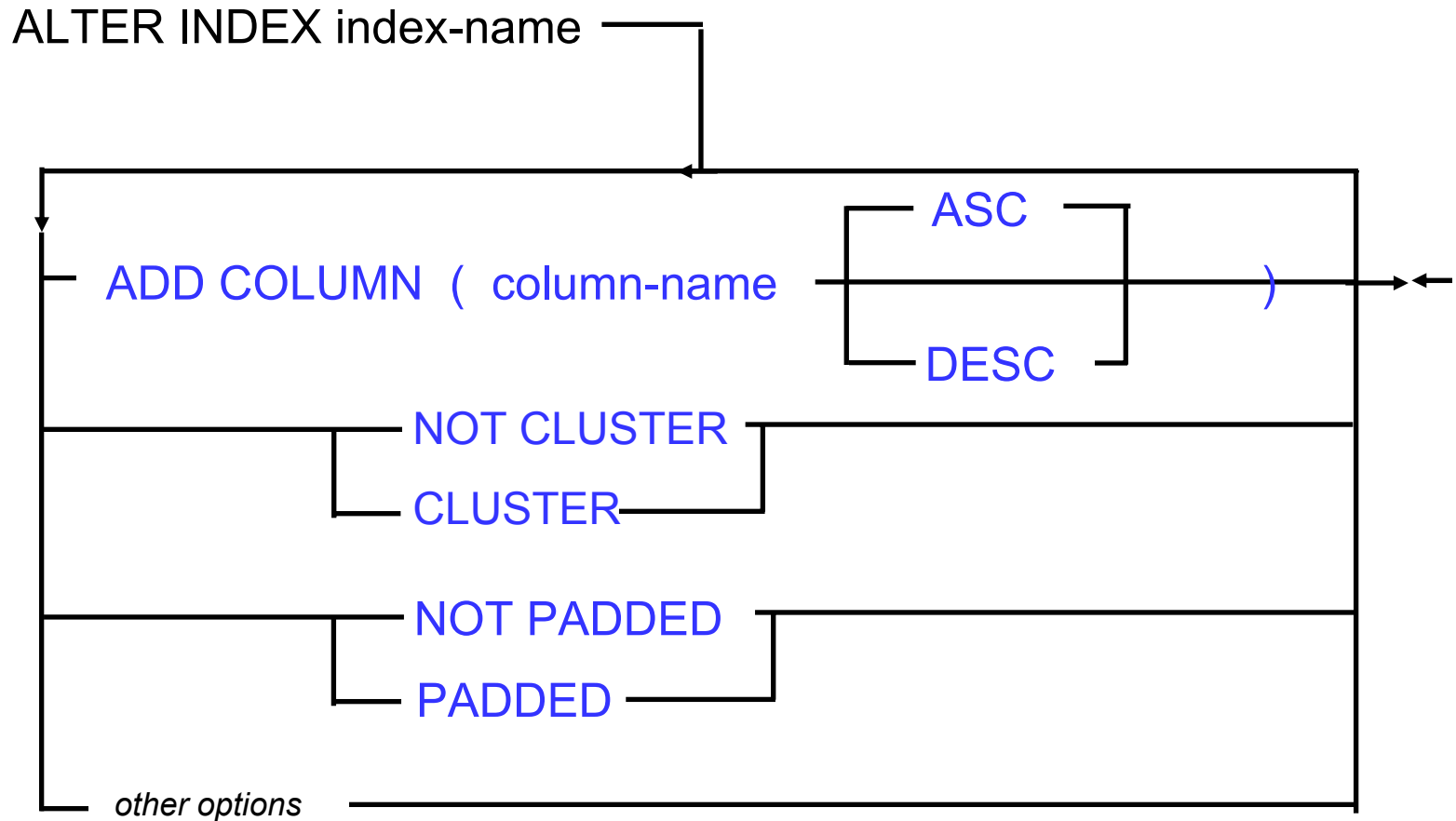
ALTER TABLE table-name



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;



Immediate
availability!
(Index in
AREO*)

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
 - ALTER INDEX index1 NOT CLUSTER
Will continue to be used until new clustering index is defined
 - 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 RBDP
 - Reset by REORG, LOAD REPLACE, or REBUILD
- ALTER INDEX NOT PADDED sets index to RBDP
 - Reset by REORG TABLESPACE, LOAD REPLACE or REBUILD INDEX
 - Index must be rebuilt from data
 - Optimizer may choose index for index only access

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 RBDP may be chosen (possible resource unavailable)
 - Dynamic PREPARE
 - Indexes in RBDP avoided
 - 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



Summary

- Ability to ALTER table columns and indexes rather than DROP and CREATE
- Improved management of partitioned table spaces with ADD and ROTATE of partitions
- Table-Controlled Partitioning enables dropping of the partitioning index and changes altering of partitions and creating a table like a partitioned table.
- Index attributes including to clustering, padding, and adding columns is allowed

