Horizontal Table Partitioning

Dealing with a manageable slice of the pie.

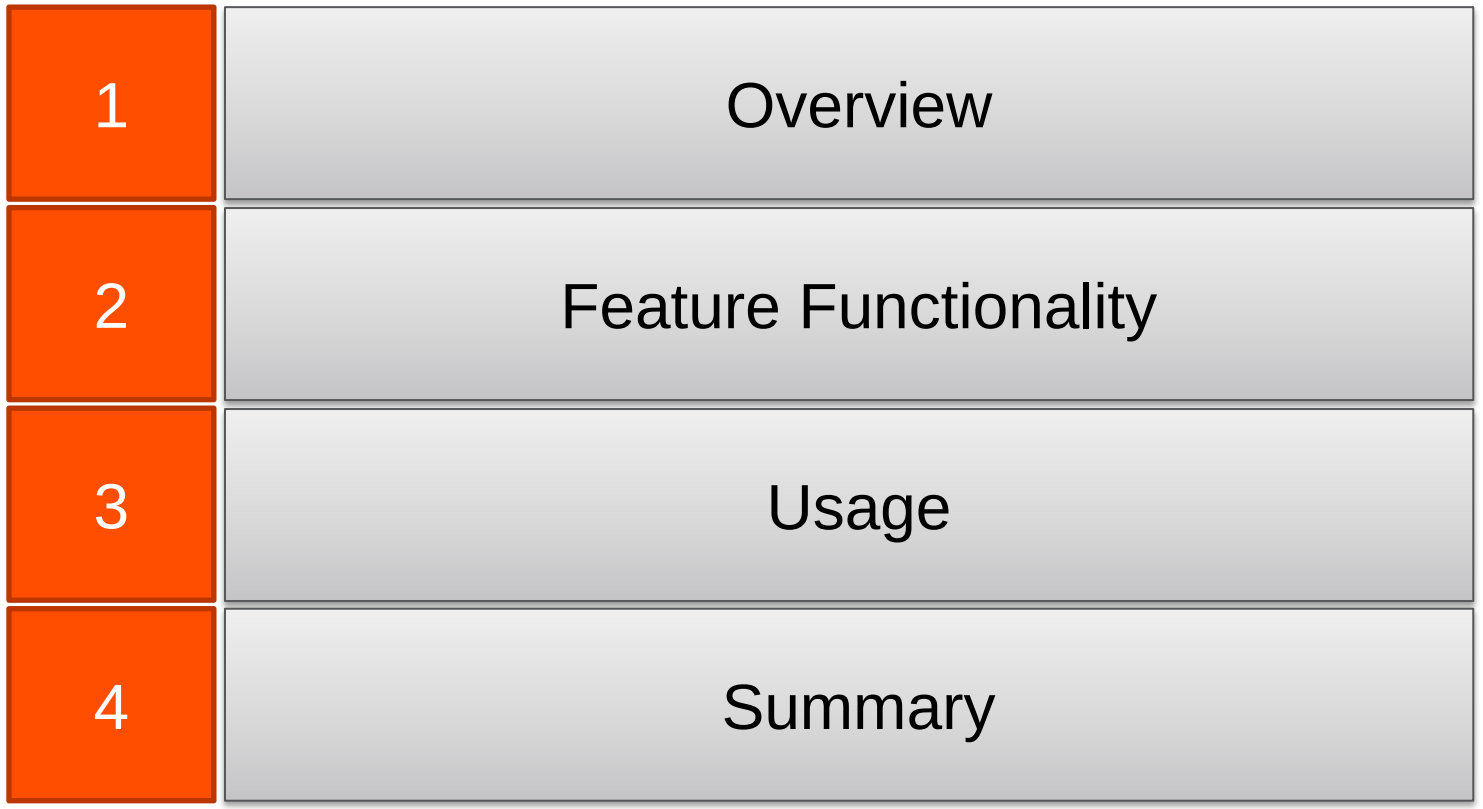
Richard Banville
Fellow, OpenEdge Development
August 7, 2013

PROGRESS EXCHANGE 2013
DISCOVER. DEVELOP. DELIVER.

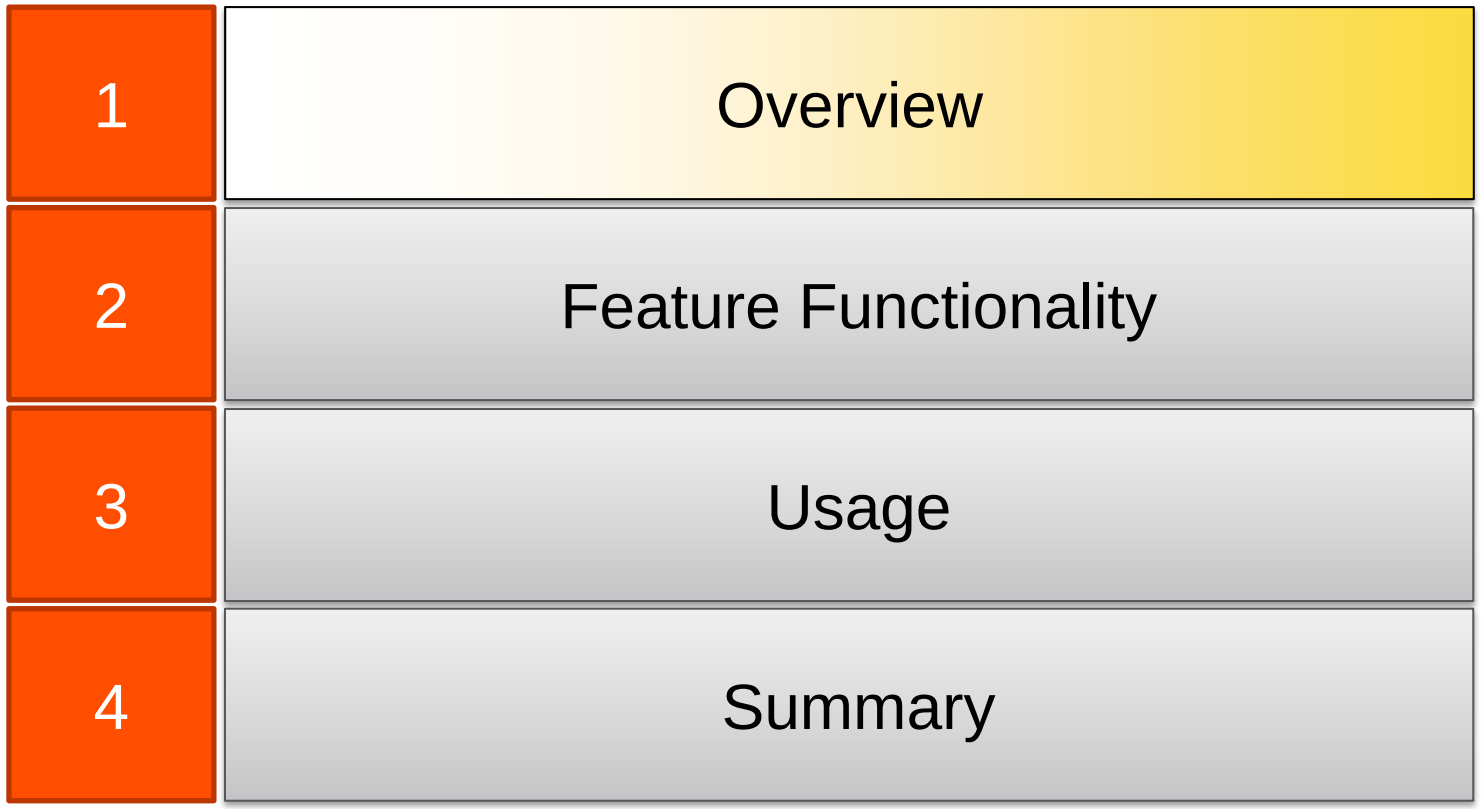
Agenda

Overview
Functionality
Usage
Summary

Agenda



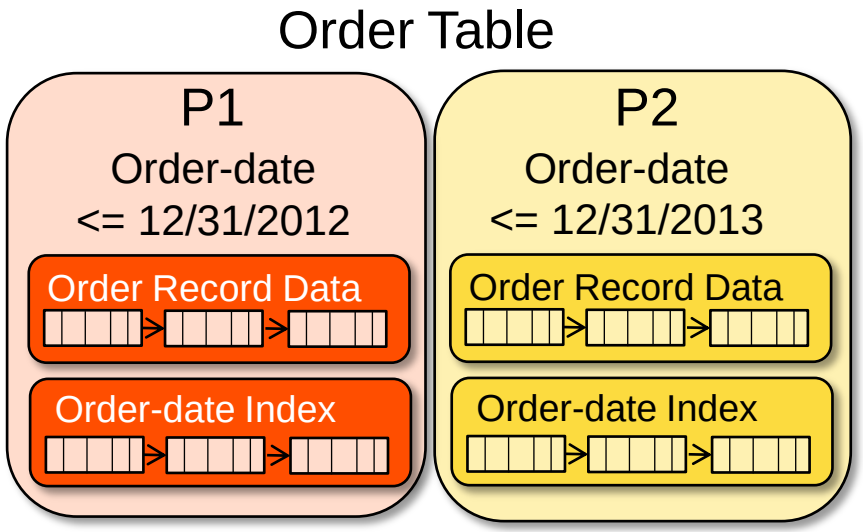
Agenda

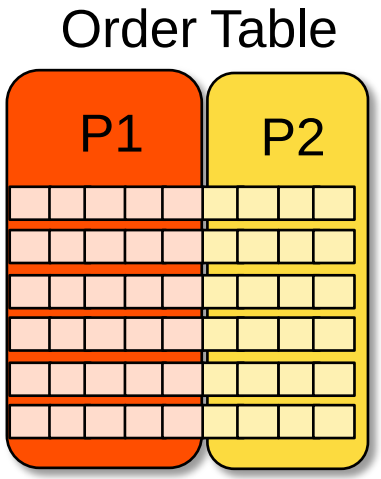


Overview: *Horizontal Table Partitioning*

What is it…

- Table instance stored in multiple self-contained locations
  - Typically used in OLTP deployments
  - Based on user specified column value & storage area
    - Different from partitioning by user identity (multi-tenancy)
  - Row does not span partitions
  - Referred to as just Table Partitioning
- As opposed to Vertical Table Partitioning
  - Typically OLAP deployments
  - Table is partitioned vertically by particular columns
  - To reconstruct row, several partitions must be visited
  - Usually partition by related columns within table





Overview: *Table Partitioning*

What is it good for…

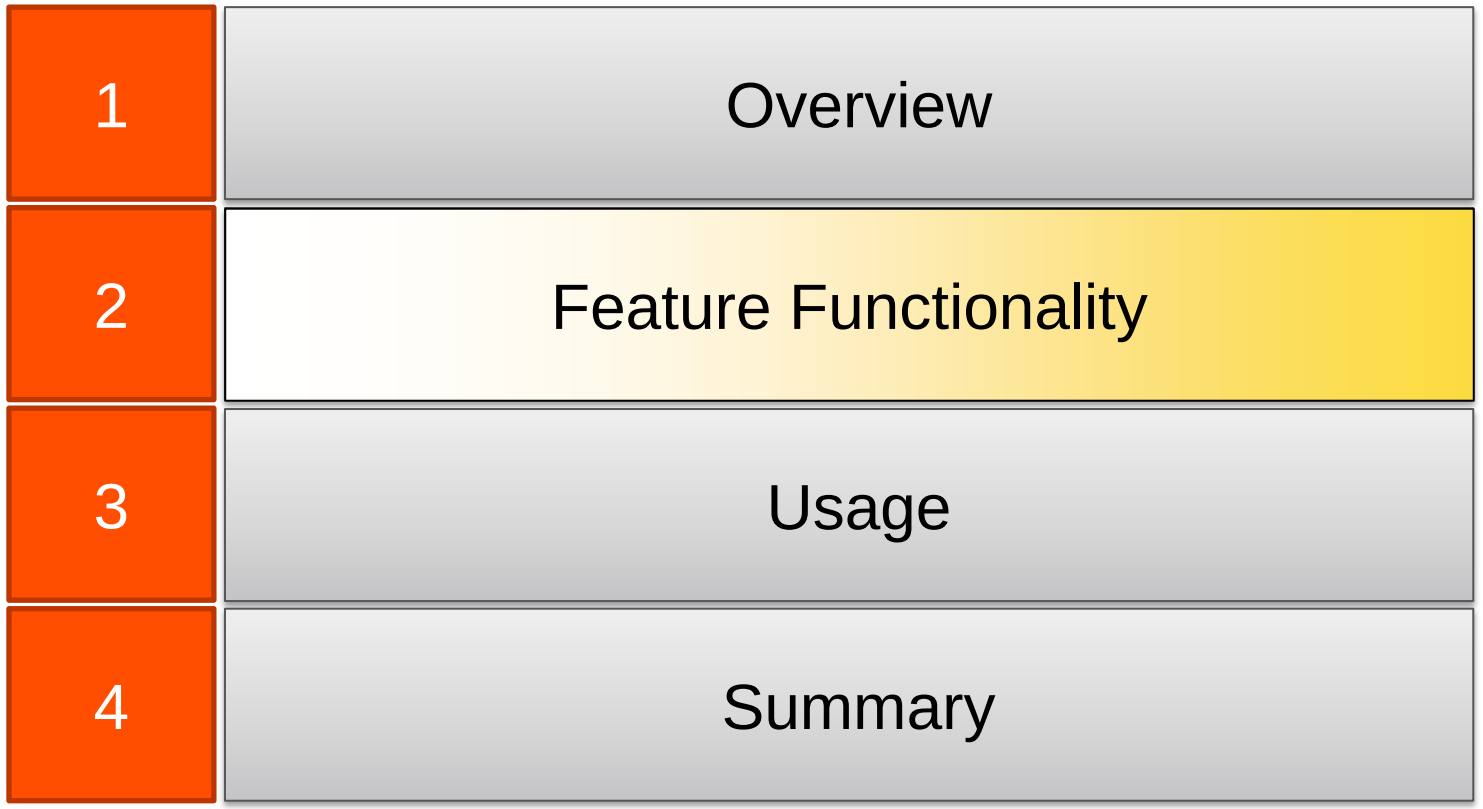
Advantages

- Performance impact
  - Partition elimination for queries (“pruning”)
  - Improved concurrency
    - For random activity
- Availability
- Maintenance advantages
  - Purge, Archive
  - Repair, rebuild
  - Partition level tuning

Disadvantages

- Partition alignment & lookup (insert/delete)
  - Update of partition aligned key values
  - Missing aligned columns in where clause
- DBA getting it right
  - Knowledge of application table definition & physical layout
  - Repartitioning costs
- More complex deployment

Agenda



Disclaimer

- This presentation is for informational purposes only, and the reader is hereby cautioned that actual product development may vary significantly from it

- This presentation may not be interpreted as any commitment on behalf of Progress, and future development, timing and release of any features or functionality described in this presentation remains at our sole discretion

Features

- Create “named” data partitioned storage based on column value
  - Table partition based on a single value (List Partitioning)
  - Table partition based on a value range (Range Partitioning)
    - Ability to provide open ended ranges*
  - Sub-partition partitioning
    - Combination of multiple list and/or range partitioning on the same table
  - Composite partitioning
    - Multiple partition definitions with storage in the same physical partition
    - Table partition based on multiple values in a single column
    - Initially supported for migration only
- Partition level pruning
- Partition level locking (administrative operations)

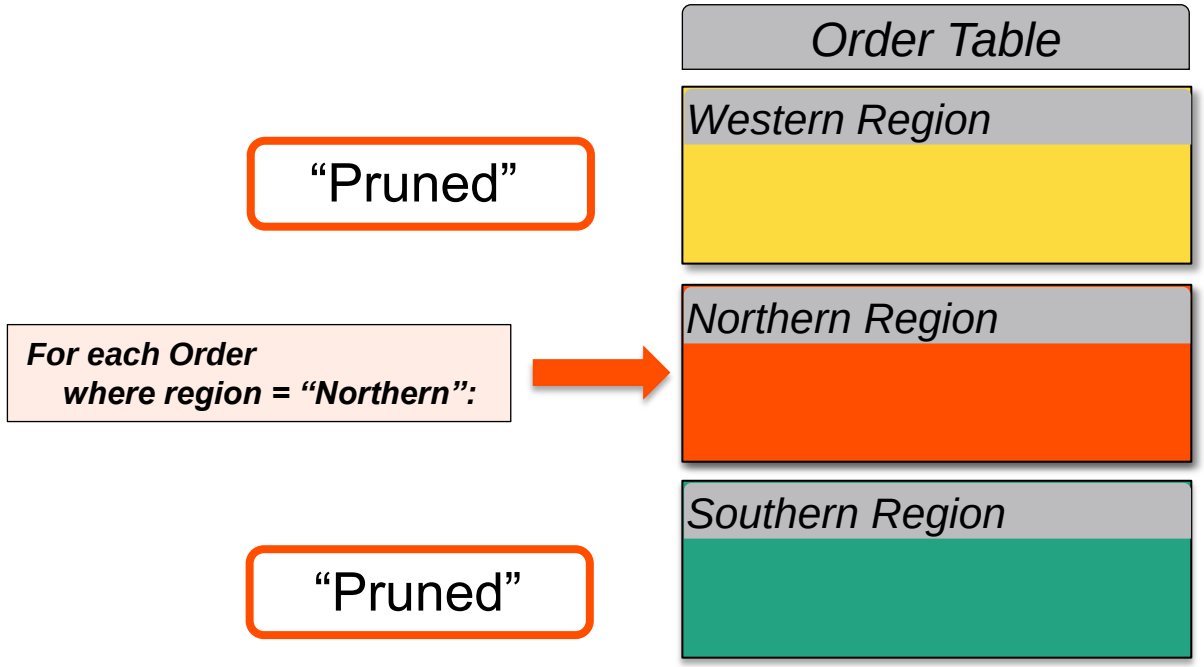
List Partitioning: Data Access

Partition based on a single value (List Partitioning)

List Partitioning

By region

Value stored == partition definition



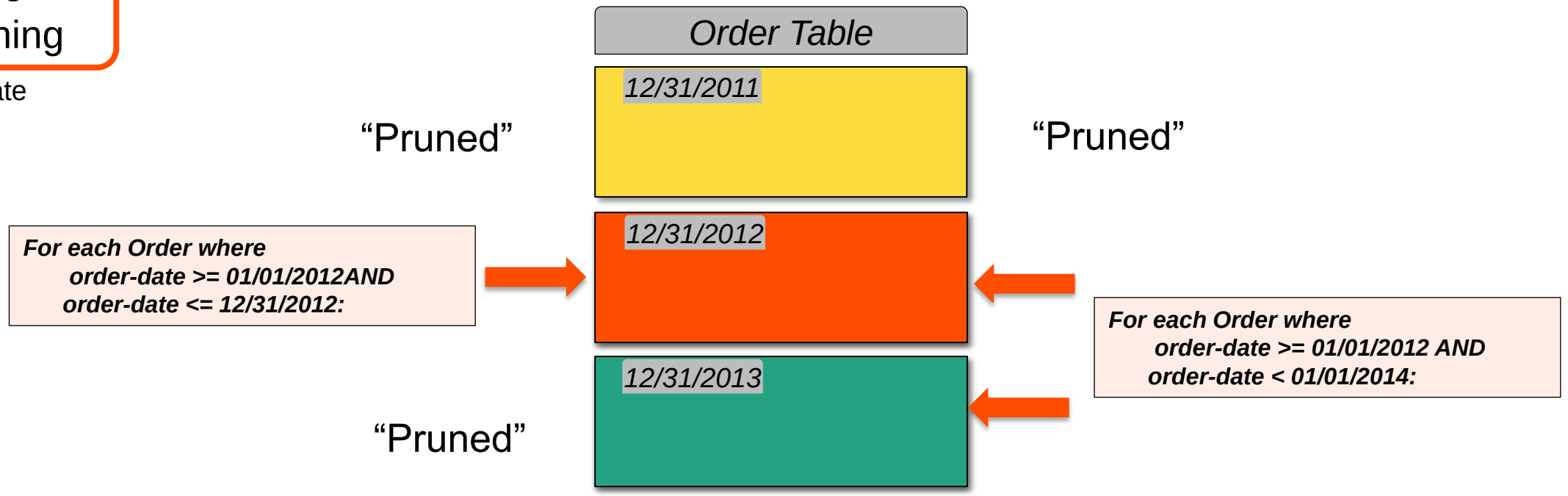
Range Partitioning: Data Access

Partition based on a value range (Range Partitioning)

Range Partitioning

By order-date

Value stored <= partition definition



NOTE: Pruning optimization: Use >= or <= rather than > or <.
The pruning is a "bit" more efficient!

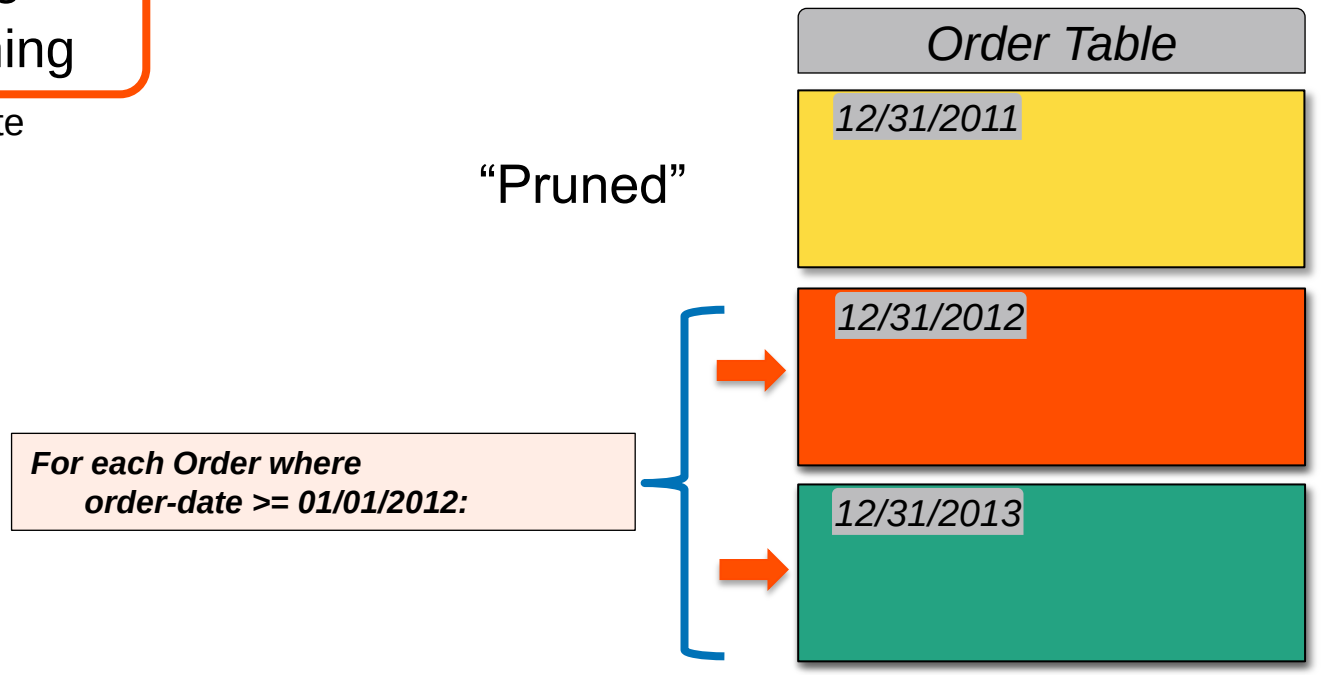
Range Partitioning: Data Access

Partition based on a value range (Range Partitioning)

Range Partitioning

By order-date

Value stored <= partition definition

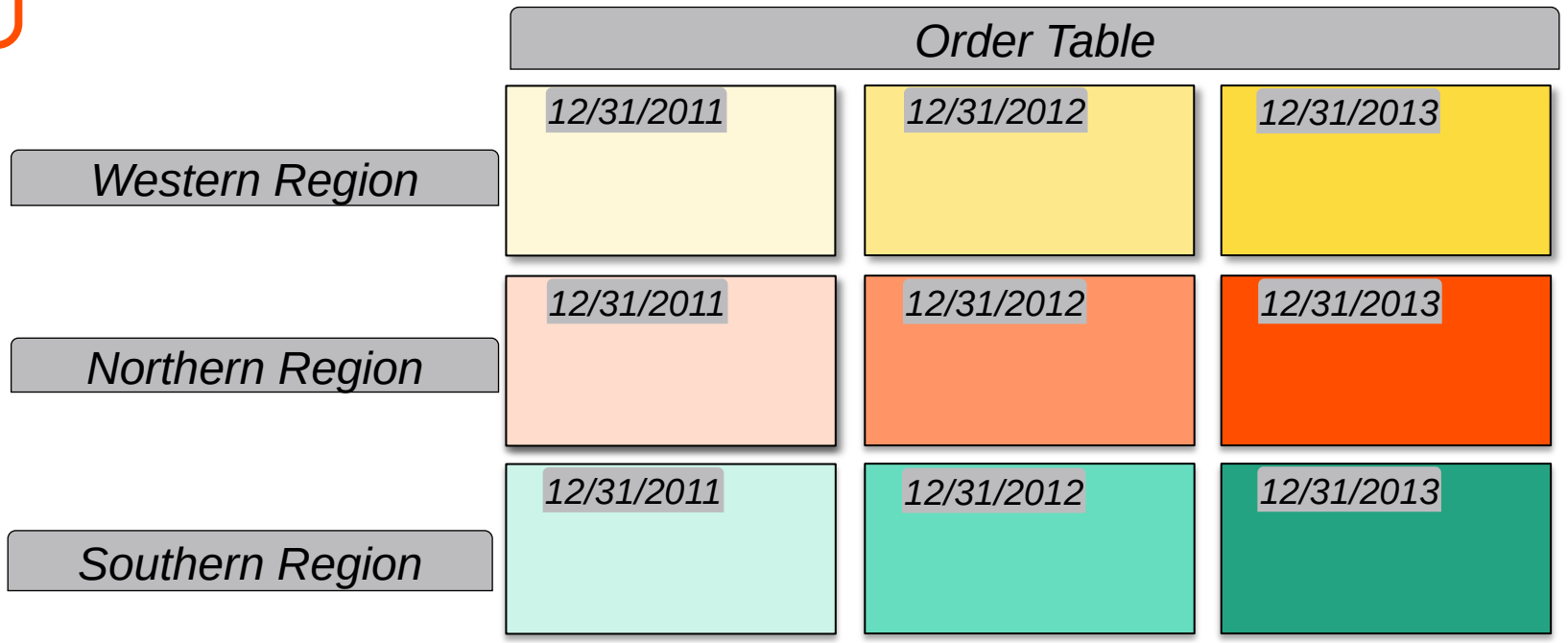


Sub-partitioning by Region & Date

Sub-partitioning

9 partition example

Sub-partition by region & order-date w/in region



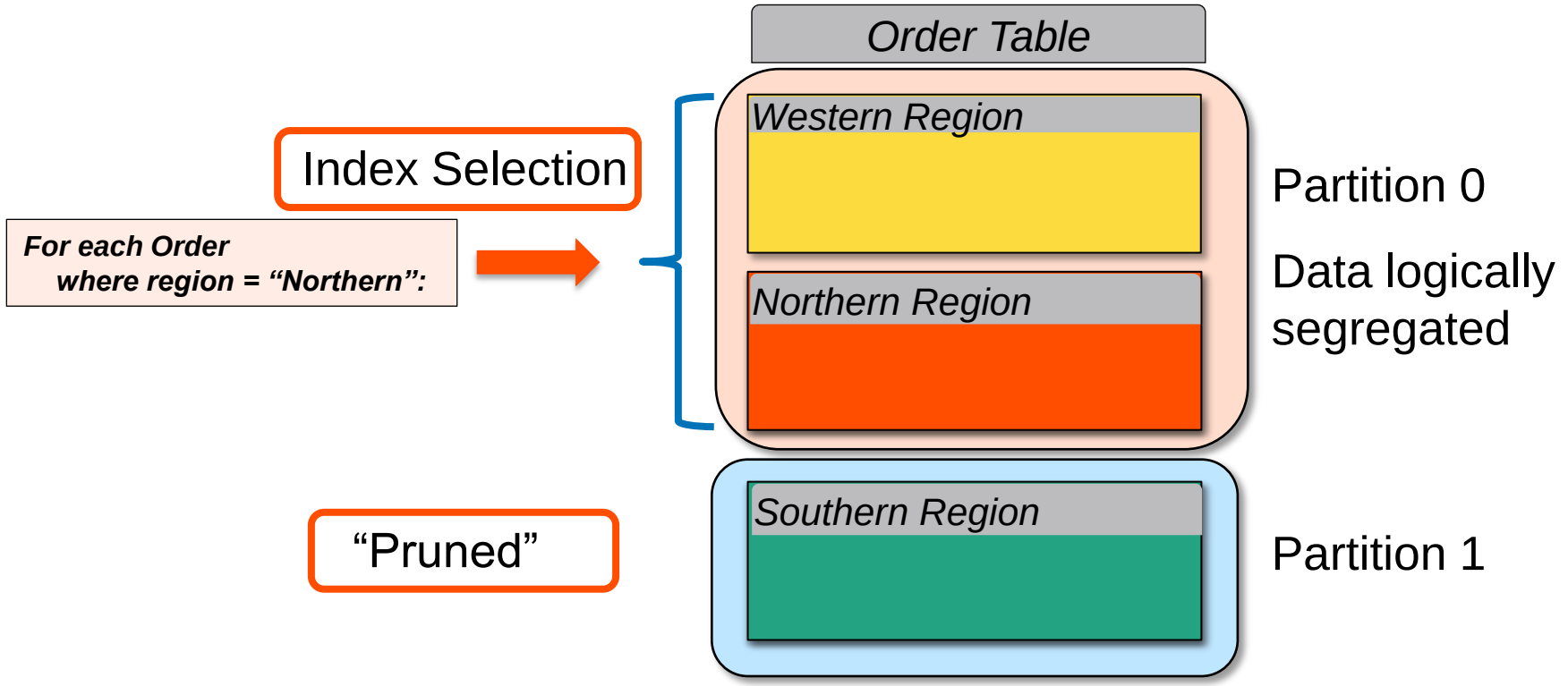
- *Only last partitioned column may be a range partition.*
- *Range partition can be any "indexable" field type*

Composite Partitions by Region

Composite Partitioning

by region

Value stored == partition definition

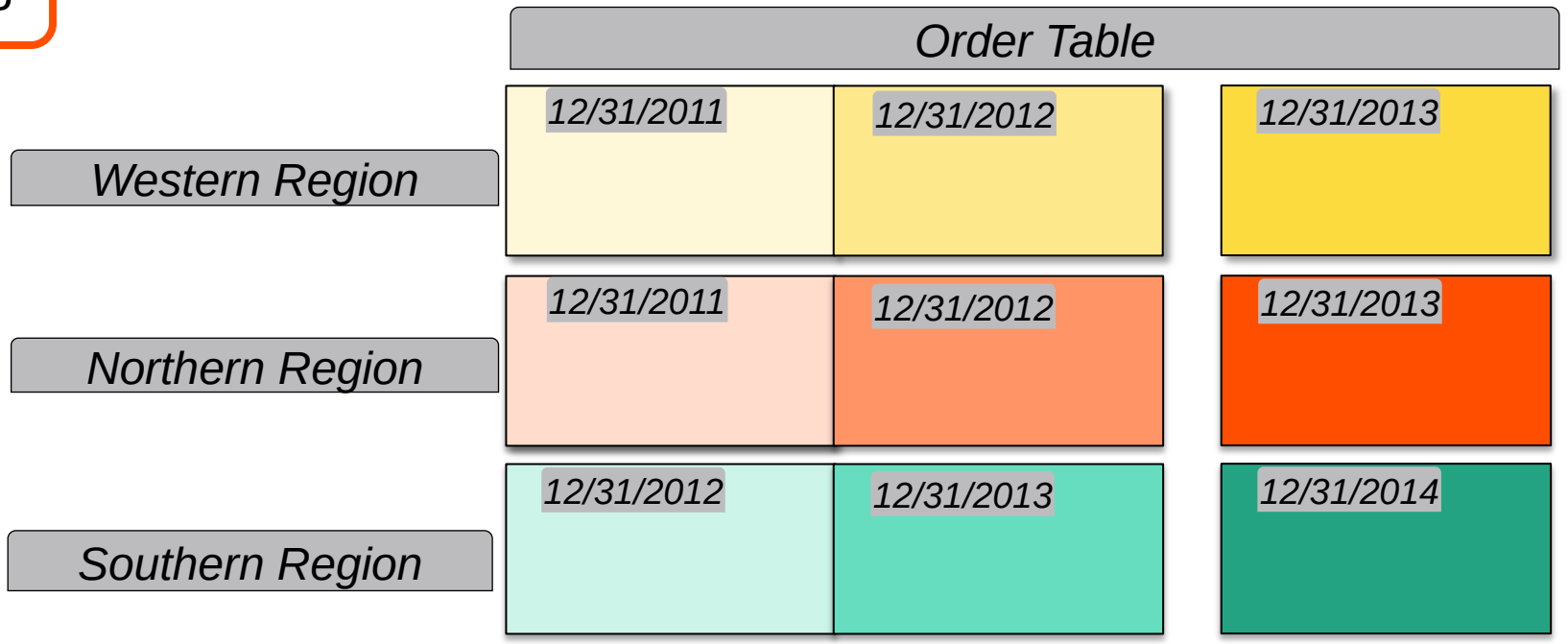


- *Implemented for migration only.*

Composite Partitioning by Region & Date

Composite Partitioning

Sub-partition by region & order-date w/in region

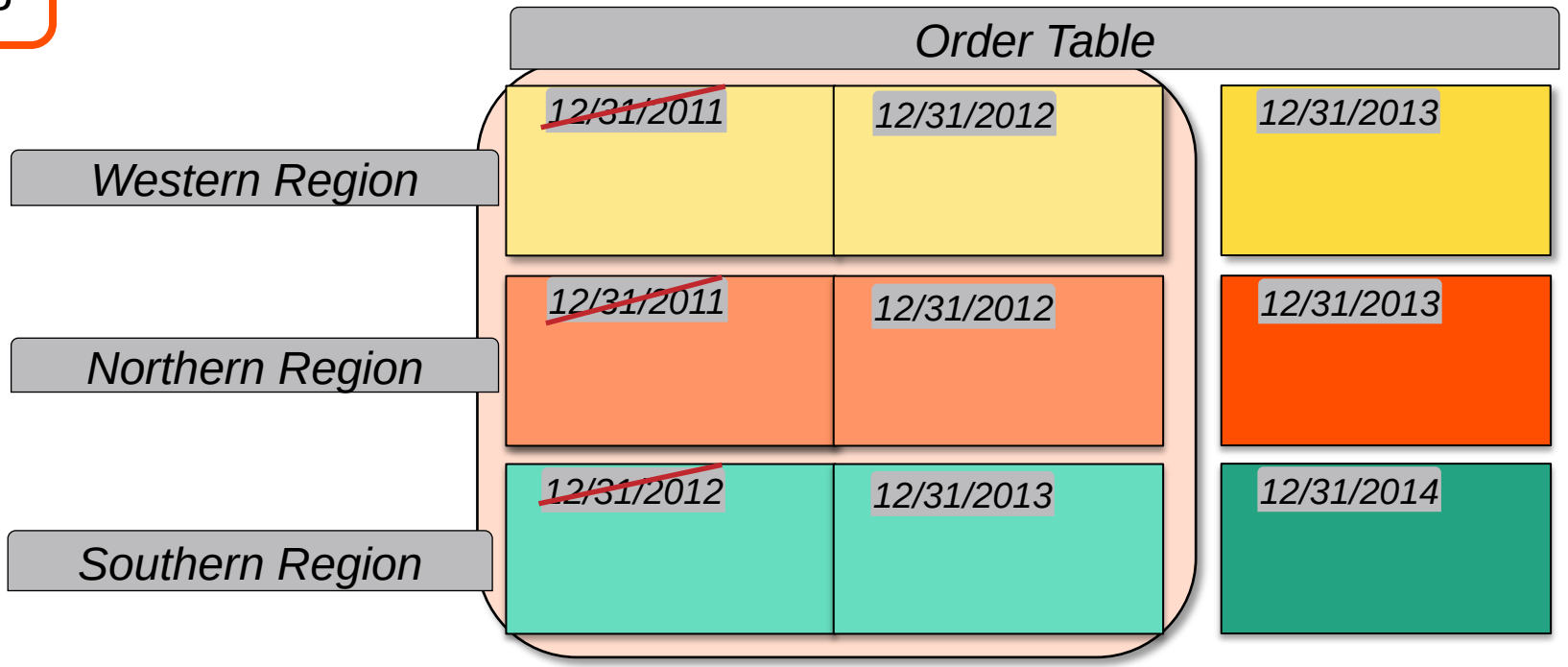


- *Implemented for migration only.*
- *Only one range value per unique list value set*

Composite Partitioning by Region & Date

Composite Partitioning

Sub-partition by region & order-date w/in region



Partition 0

Data logically segregated

Partitions 1-3

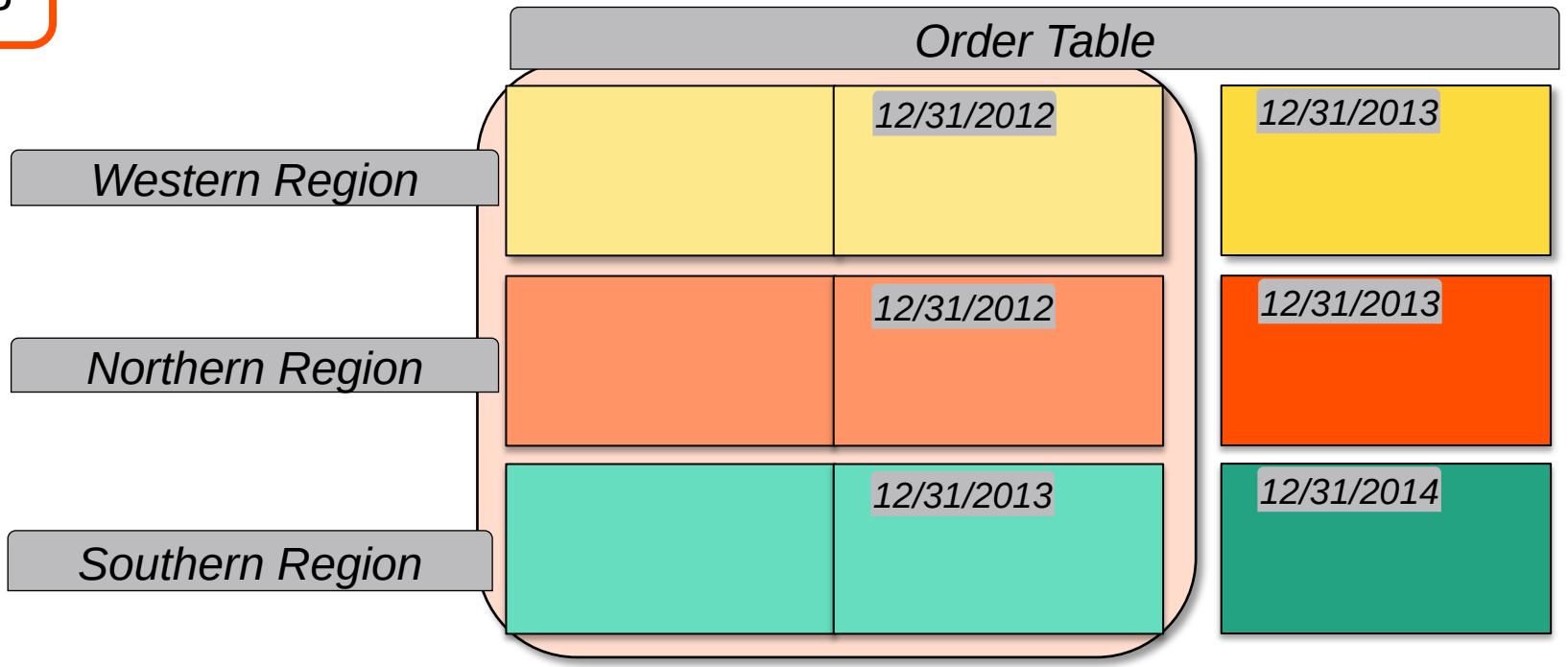
Data physically segregated

- *Implemented for migration only.*
- *Only one range value per unique list value set*

Composite Partitioning by Region & Date

Composite Partitioning

Sub-partition by region & order-date w/in region



Partition 0

Data logically segregated

Partitions 1-3

Data physically segregated

- *Implemented for migration only.*
- *Only one range value per unique list value set*

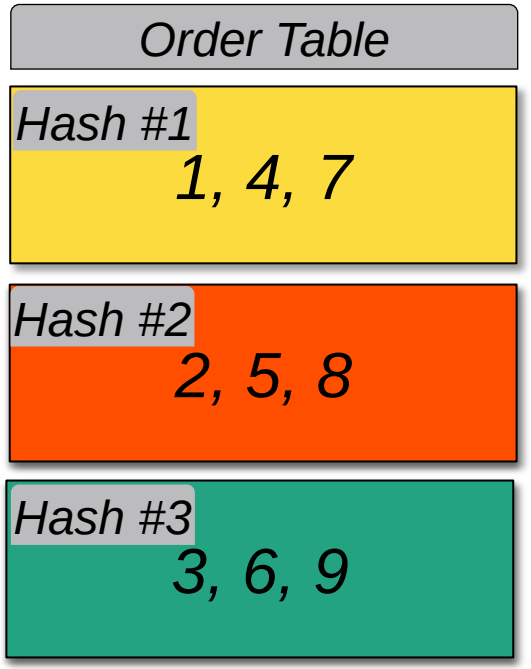
Hash Partitioning: Data Access

Hash Partitioning

No plans to implement

Create Order.
Assign order-num = next-value(order-seq).

Apply hash function to order-num.
Store data in resulting partition #.



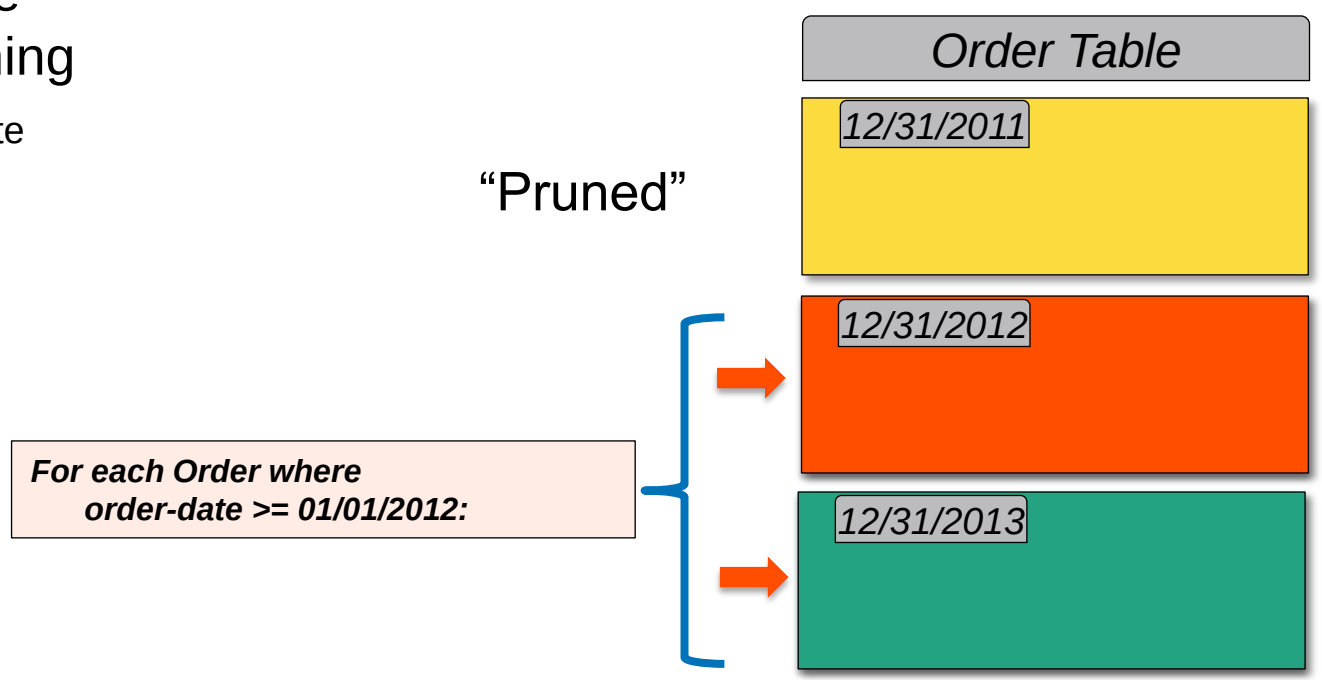
Index Support

- Local index support
  - One index b-tree per partition
    - Index MUST be partition aligned
- Global index support
  - One index b-tree containing all the table's data
  - Ability to index across partitions
    - Typically for non-partitioned aligned sort order
  - Provides uniqueness support for non-partitioned columns
    - Unique "cust-num" or sort by "name" for example
- Composite index support
  - One index b-tree containing entries for multiple partition definitions
  - Table data for composite partition stored in same partition

Index Types: Data Access

Range Partitioning

By order-date

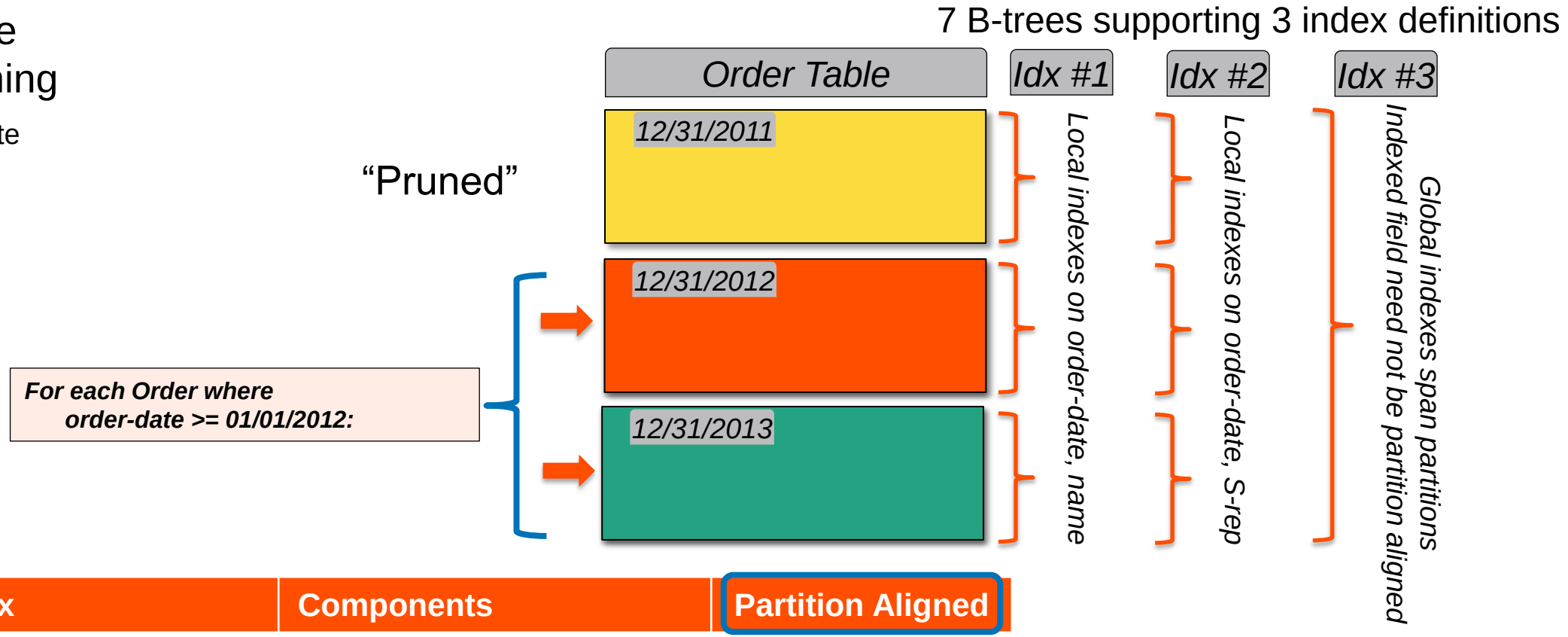


Index	Components	Partition Aligned
Index #1	{Order-Date, Name}	YES
Index #2	{Order-Date, S-rep}	YES
Index #3	{Cust-num}	NO

Index Types: Data Access

Range Partitioning

By order-date

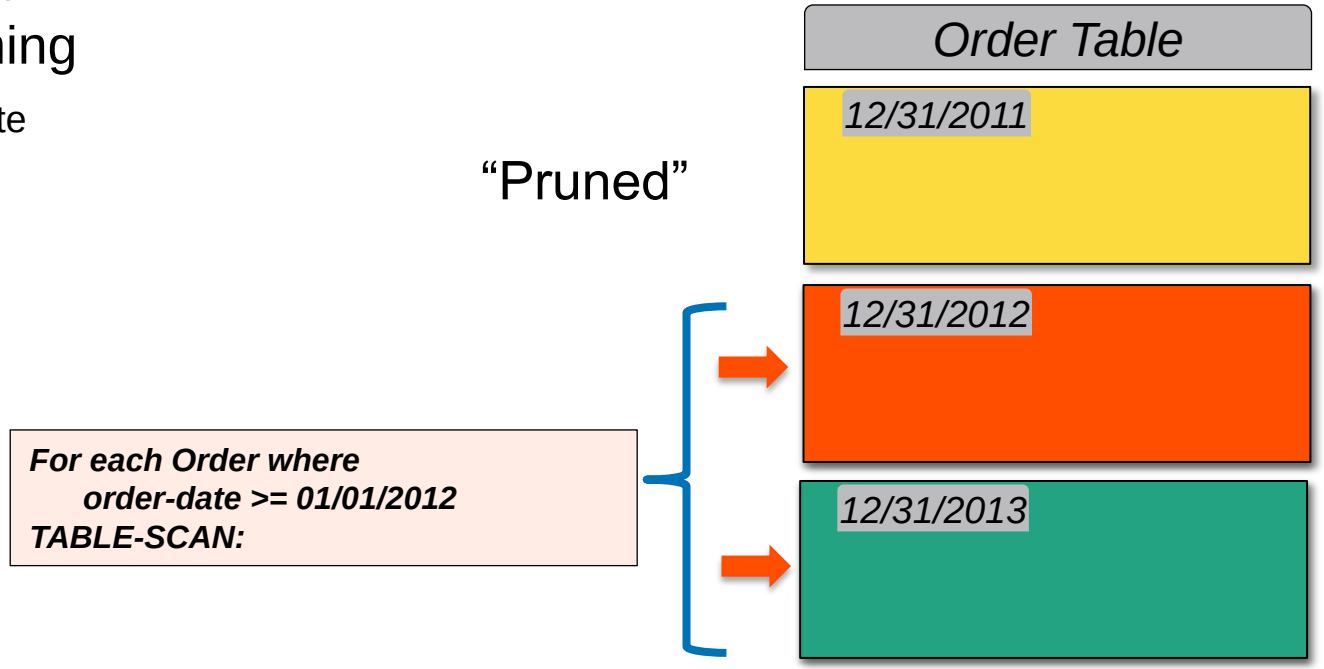


Index	Components	Partition Aligned
Index #1 (local)	{Order-Date, Name}	YES
Index #2 (local)	{Order-Date, S-rep}	YES
Index #3 (global)	{Cust-num}	NO

Table Scanning: Data Access

Range Partitioning

By order-date



- Restrict data retrieved without using an index.
- Ordered by partition. "Random" order within partition.
  - (Cluster order, not rowid order)

Space Allocation and Access

- Limits
  - 32,765 partitions per table
  - 15 column sub-partition definition
- Separate partition definition and space allocation capabilities
  - Can store (table, index, LOB) partitions for same table in different Type II Storage Areas
  - Can define partitions without allocating space
  - Can specify “open ended” range partitions
    - Everything <= High-range
- Mark a partition as read only
  - Avoid backing up read only partitions (via incremental backup)
- Take partitions offline and back online

Some General Restrictions

- Partitioning Restrictions
  - Partitioning exists for objects in Type II Storage areas only
  - Partition cannot span storage areas
  - No temp table support
  - Re-partitioning only supported through dump & load
    - Different partition column or type (change of values via split or merge)
  - No overlapping range values or “gaps” for range partitioning
  - Only 1 range component per partitioned table
    - Also, must be last component
  - No partitioning of multi-tenant tables*

Some More General Restrictions

- Indexing Restrictions
  - Local indexes ALWAYS follow table partition definition
    - They are always partitioned aligned
  - Word indexes cannot be local indexes
    - They are not partition aligned
  - Local index partition Id same as matching table partition Id
- Utilities
  - Database wide operations will continue to be database wide
    - Auditing, replication, transparent data encryption, backup, restore, roll forward, authorization

Migration

- Application transparency
  - Add/delete/split/merge without ABL re-compile
  - No new language syntax to access partitioned table (find by rowid is special!)
- GUI for partition management
  - via OpenEdge Management (OEM) and OpenEdge Explorer (OEE)
- Migration
  - Networked access backward compatible (?)
  - Ability to enable partitioning on current table without data movement
    - Leverage composite partitioning (next section)
  - Must have partition aligned “composite” index
  - Ability to split out existing data into new partition (online)
  - OEM/OEE “Tool” will help with list partition migration
    - Identify current unique values

Migration

- Application transparency
  - Add/delete/split/merge without re-compile
  - No new language syntax to access partitioned table
- GUI for partition management
  - via OE Management and OE Explorer
- Migration
  - Networked access
  - Ability to enable partitioning on current table without data movement
    - Leverage composite partitioning
  - Must have partition aligned “global” index
  - Ability to split out existing data into new partition (online)
  - Tool will help with list partition migration
    - Identify current unique values

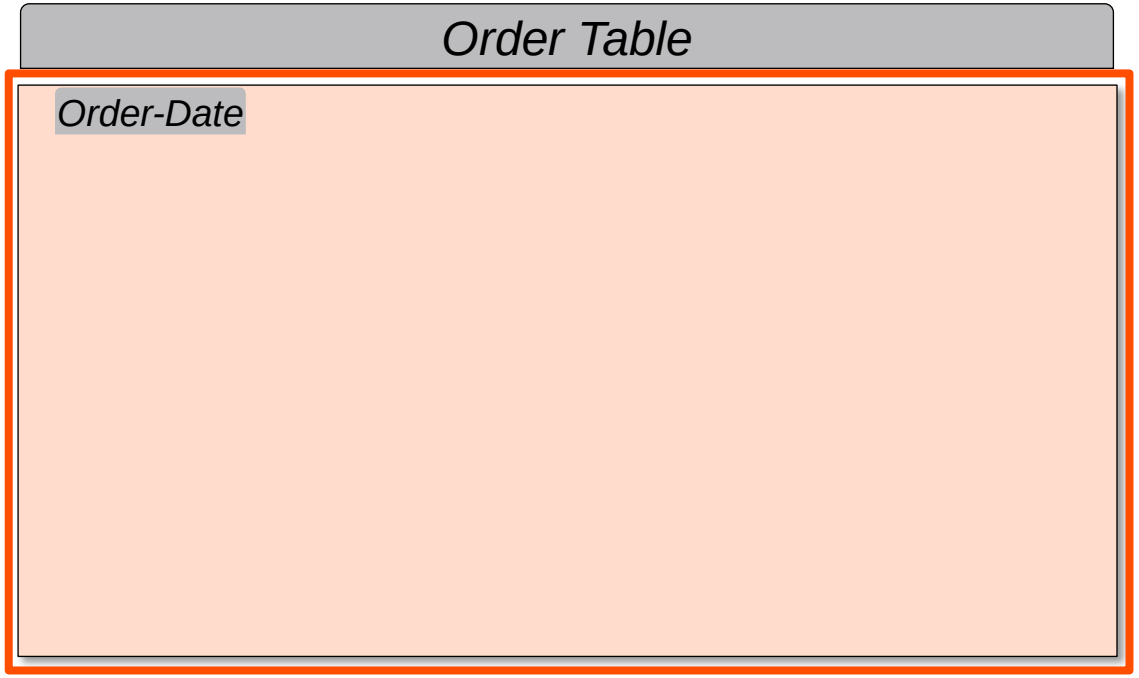
More pictures please…

Migration: Range partitioning by Order-Date

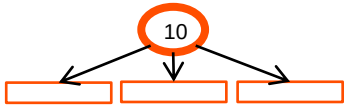
Migration

Partition aligned index by order-date

All existing data in current partition 0



Index #10
Partition #0

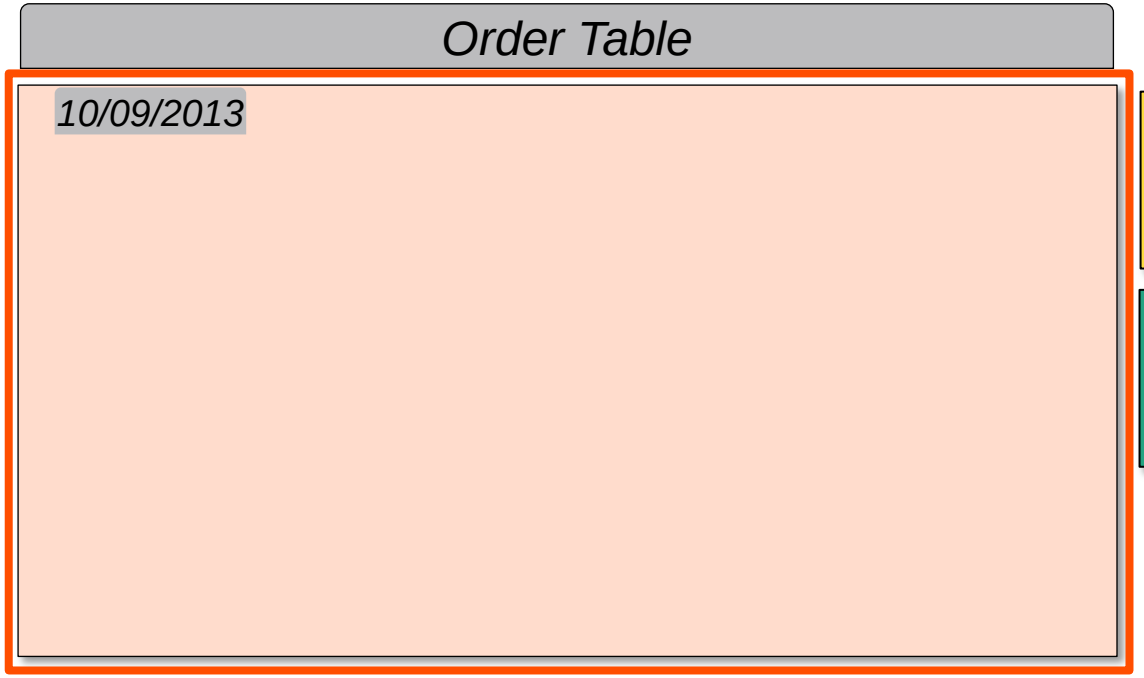


Migration: Range partitioning by Order-Date

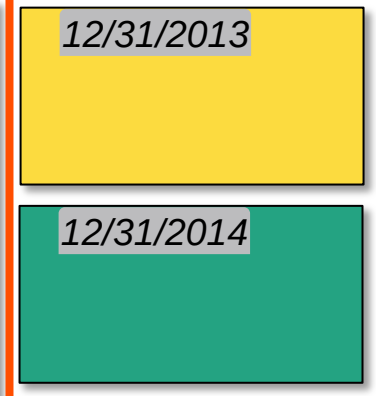
Migration

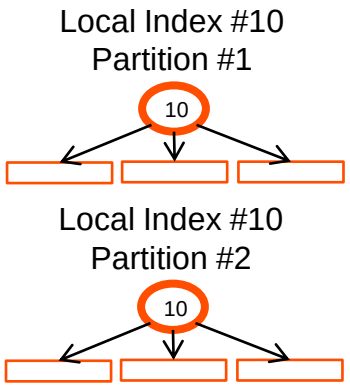
Partition aligned index by order-date

All existing data in current partition 0

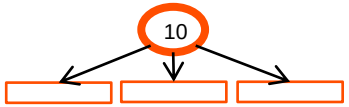


All new data stored in proper partition





Index #10
Partition #0

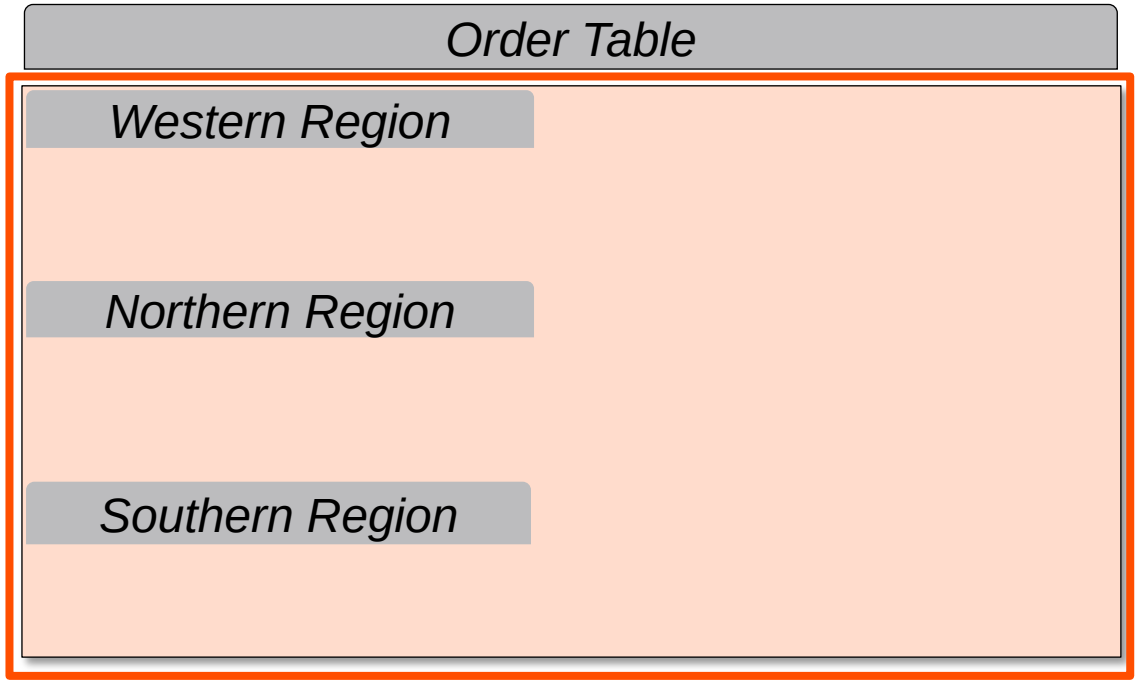


Migration: Composite list partitioning by Region

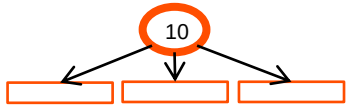
Migration

Partition aligned index by region

All existing data in current partition 0



Composite Index #10
Partition #0

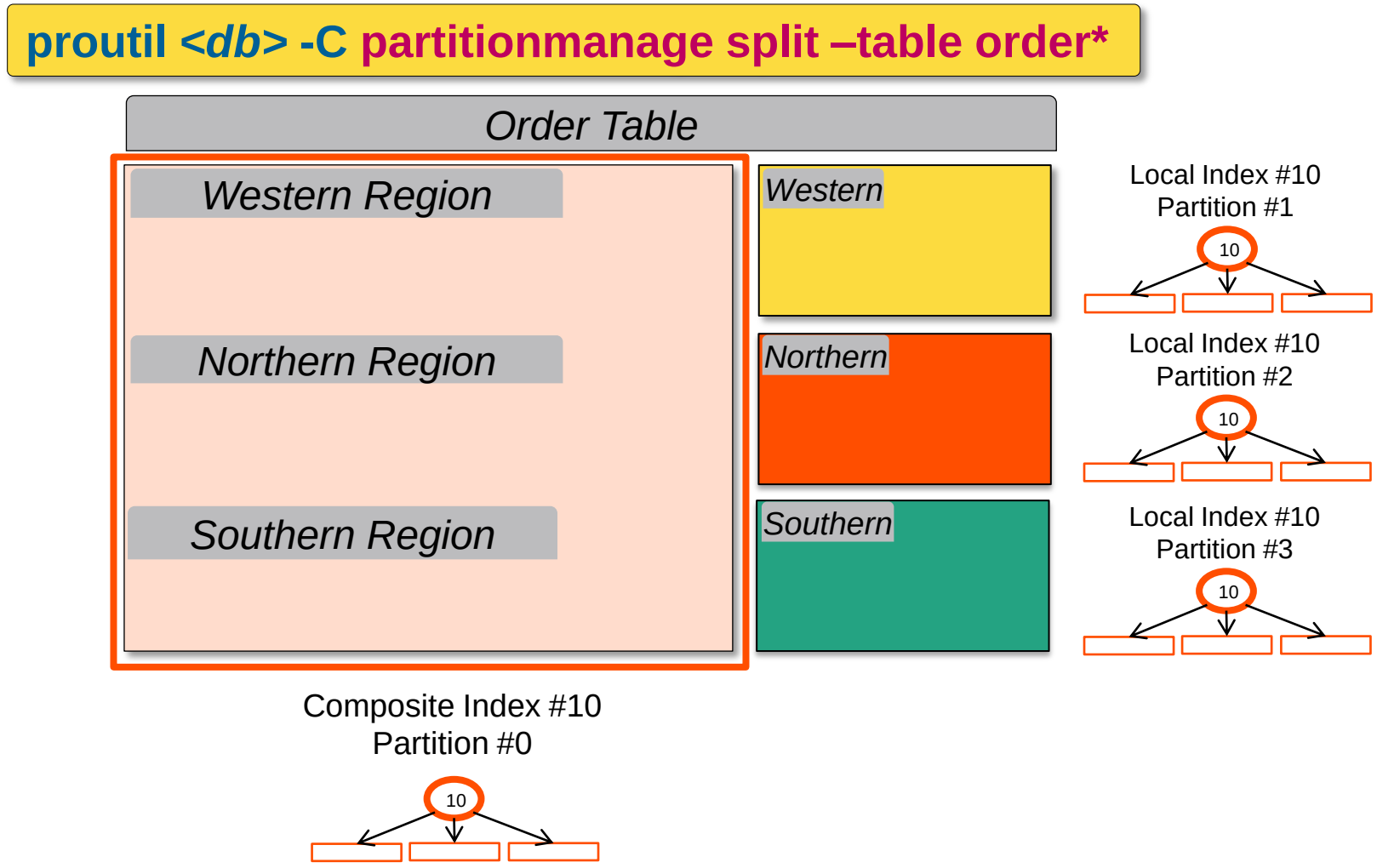


Migration: Composite list partitioning by Region

Migration

Partition aligned index by region

- New partitions created
  - Tool creates unique list entries
- Data migrated via split utility (as time allows)



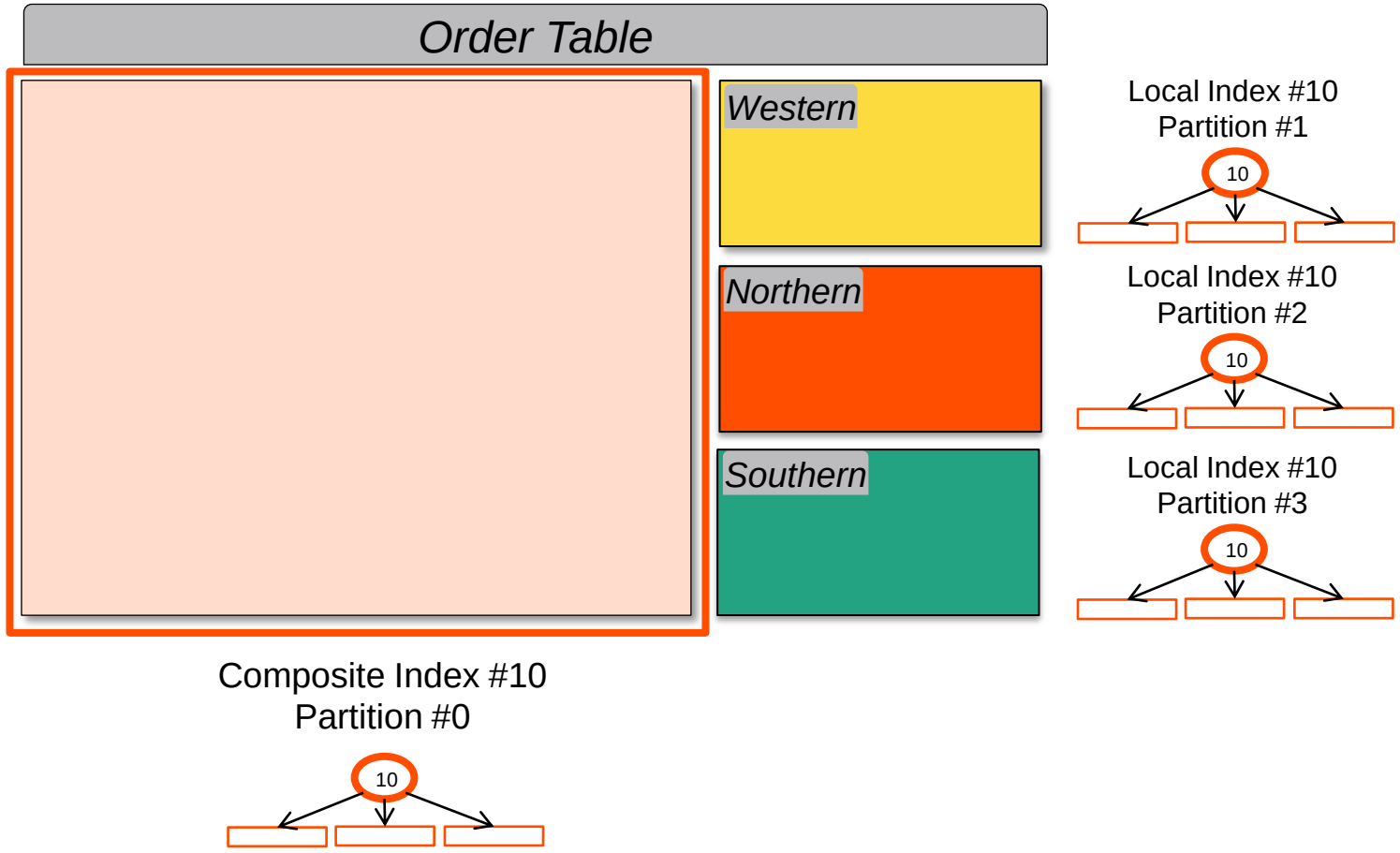
Migration: List partitioning by Region

Migration

Partition aligned index by region

- New partitions created
  - Tool creates unique list entries
- Data migrated via split utility (as time allows)
- Original space recaptured via truncate

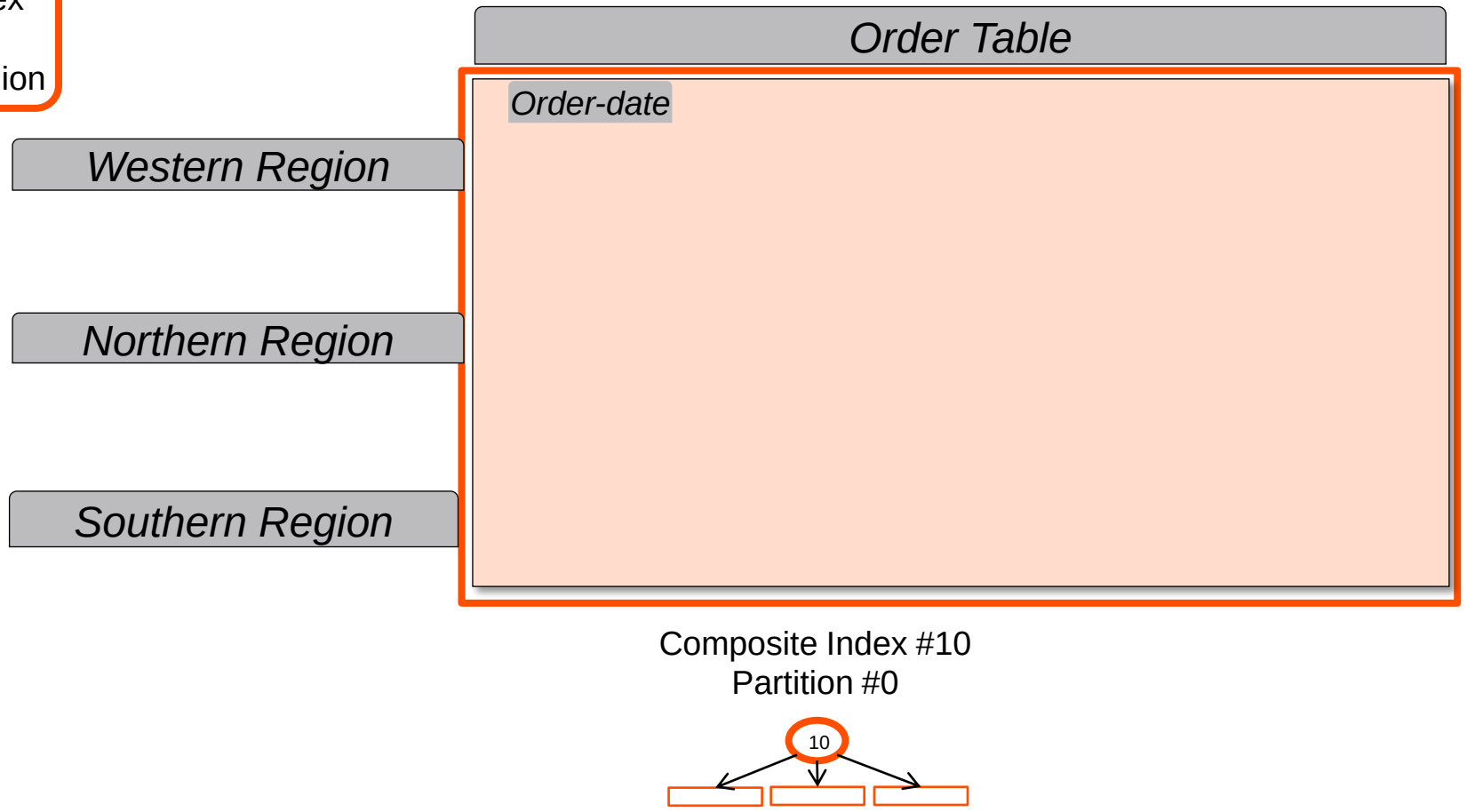
proutil <*db*> -C partitionmanage truncate –table order*



Composite & Sub-partitioning by Region & Order-date

Migration

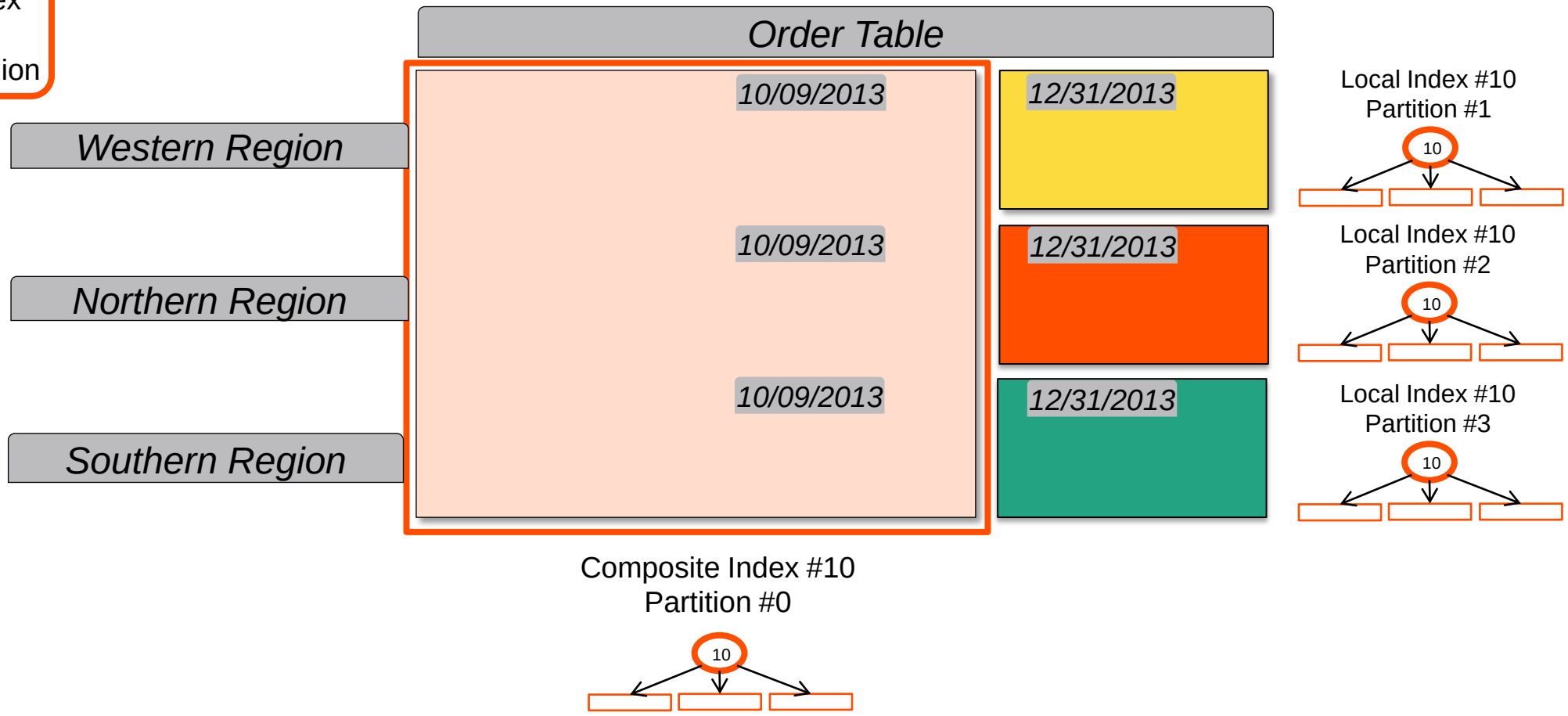
Partition aligned index
by region &
order-date within region



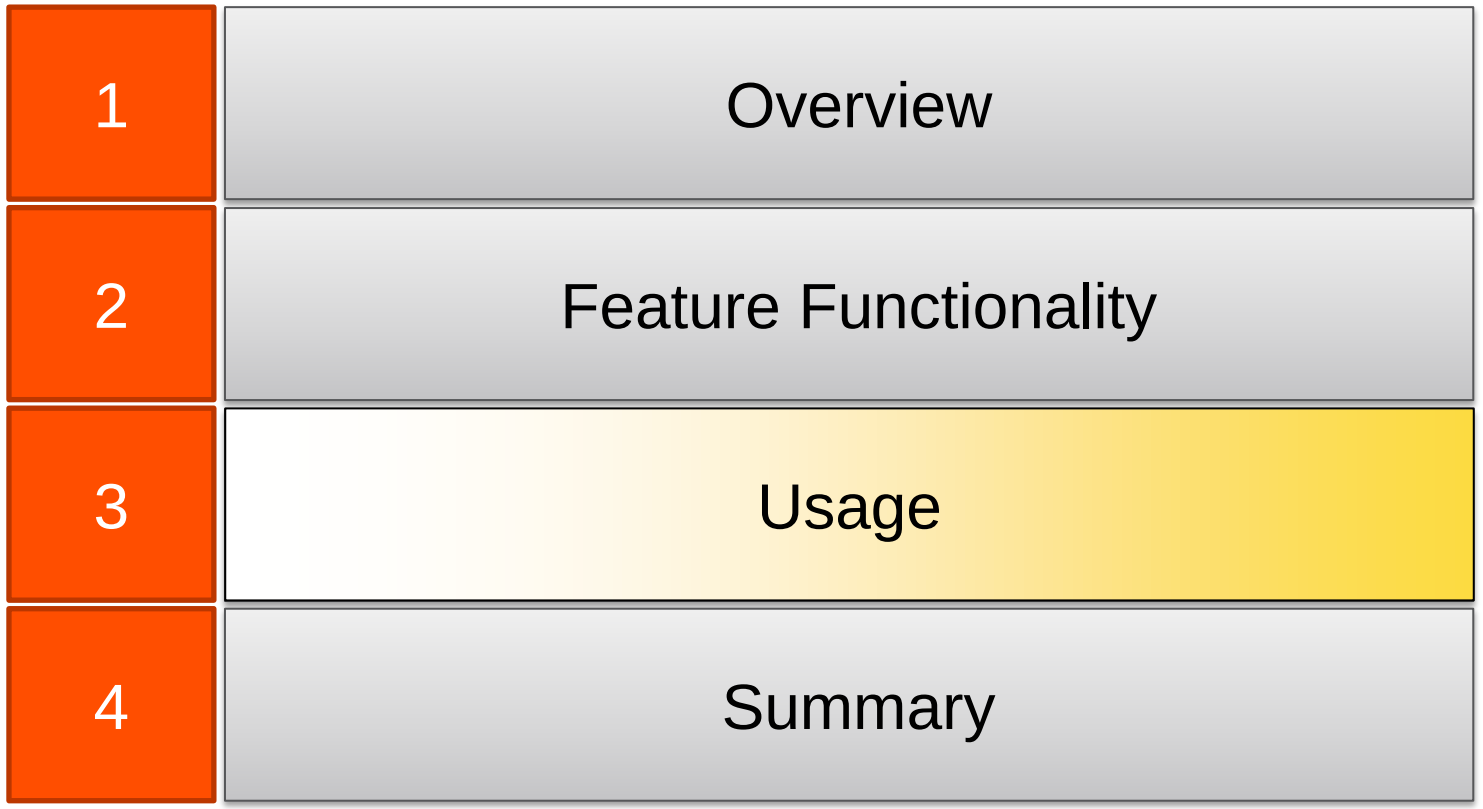
Composite & Sub-partitioning by Region & Order-date

Migration

Partition aligned index
by region &
order-date within region



Agenda

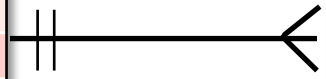


Enablement

proutil <*db*> -C enabletablepartitioning

- _Partition-Policy (-352)
  - Describes partition at the “table” level
- _Partition-Policy-Detail (-353)
  - Defines each individual partition

Column	Name	Type
2	_Partition-Policy-Name	character
3	_Object-Number	integer
4	_DataArea-default	integer
5	_IndexArea-default	integer
6	_LobArea-default	integer
7	_Allocation-default (None, immediate, delayed)	character
8	_Num-Columns	integer
9	_Column-Name	character[16]
10	_Has-Range	logical
11	_Description	character
12	_Misc	character[16]



Column	Name	Type
2	_Object-Number	integer
3	_Partition-Id	integer
4	_Partition-Name	character
5	_Partition-Column-Value	character[16]
6	_Partition-Internal-Value	raw
7	_Attributes	Logical[64] [1] = 1 space allocated [2] = 1 this is a sub-partition [3] = 1 lowest level sub-partition [4-63] unused
8	_Description	character
9	_ianum-Data	Integer
10	_ianum-Index	Integer
11	_ianum-Lob	integer
12	_Misc	character[16]

DDL Support

- ABL API support will be provided

Associate table w/partitioning

Define partition for table

```
define variable policy as IPartitionPolicy no-undo.
define variable detail as IPartitionPolicyDetail no-undo.
policy = Service:NewPartitionPolicy("Order date")
    policy:DefaultAllocation = "Immediate"
    policy:Table = tbl…
 detail = Service:NewPartitionPolicyDetail("Olddata")
    detail:SetValue(12/31/2012)…
    policy:Detail:Add(detail).
```

- Used by OpenEdge Explorer and OpenEdge Management
- OpenEdge SQL will provide it own DDL support

New Maintenance Functionality

proutil <*db*> -C partitionmanage truncate partition <*pname*> table <*tname*> numRecs <*#recs per txn*>

- Proutil partition operations (online all without major bi impact)
  - View
  - Rename
  - Truncate / delete / archive
    - Truncate recaptures space
    - Delete recaptures space & removes definition
    - Archive combines dump & truncate
  - Multiple concurrent operations
  - Supports OpenEdge Replication

New Partition Maintenance Utilities

proutil <*db*> -C partitionmanage split | merge table <*table*> partition <*src*> <*targ*>

- Split & Merge
  - Data moved in/out of partitions
  - Transactional scoping by groups of record/index operations
    - Data for same partition definition spans physical partitions
    - Only ever one copy of the data
  - Online operation with access to non-split / non-merge data
    - New split/merge transitional state for partitions
  - Multiple concurrent operations
  - Supports OpenEdge Replication
  - Recovery of operation restarts where left off

Administration

ABL APIs & Dictionary

- New APIs for DDL support
- Dump and Load of Data & Definitions
  - Partition policies managed as record data, not .df
- Data Dictionary TTY and GUI support
  - Restricted to minimal changes (_file and _index)

Data Admin Console (OpenEdge Explorer/Management)

- Subset of multi-tenancy object level support
  - Partition Create/Move
    - Move partitions prior to allocation vs after allocation
  - Partition Allocation State
  - Buffer pool assignment
  - Dump/Load support

Utility Support

- Monitoring
  - Promon and VSTs will support partition level monitoring
- Utilities with partition level only option:
  - Table/index move support
- Utilities with partition and table level options:
  - Partition level recovery (via improved TDR functionality)
  - Analysis tools dbanalys/idxanaly/tabanalys/chanalys
  - Analysis output
    - Optionally provide .csv type formatted file (rather than a hierarchical report)
  - Set/display create/toss limits
  - View/manage alternate buffer pool (vewB2)
  - dbtool

Utility Support (continued)

- Index Check
- Index Rebuild, Index Fix, Index Compact
- Index Activate/Deactivate
- Binary dump / load
- Bulkload & file definition
- SQL dump/load

  * NOTE: Load based on data value, not partition specification
    - Can load to one or many partitions

- Other
  - Move schema: mvsch
  - dbrpr limited partition support

Agenda

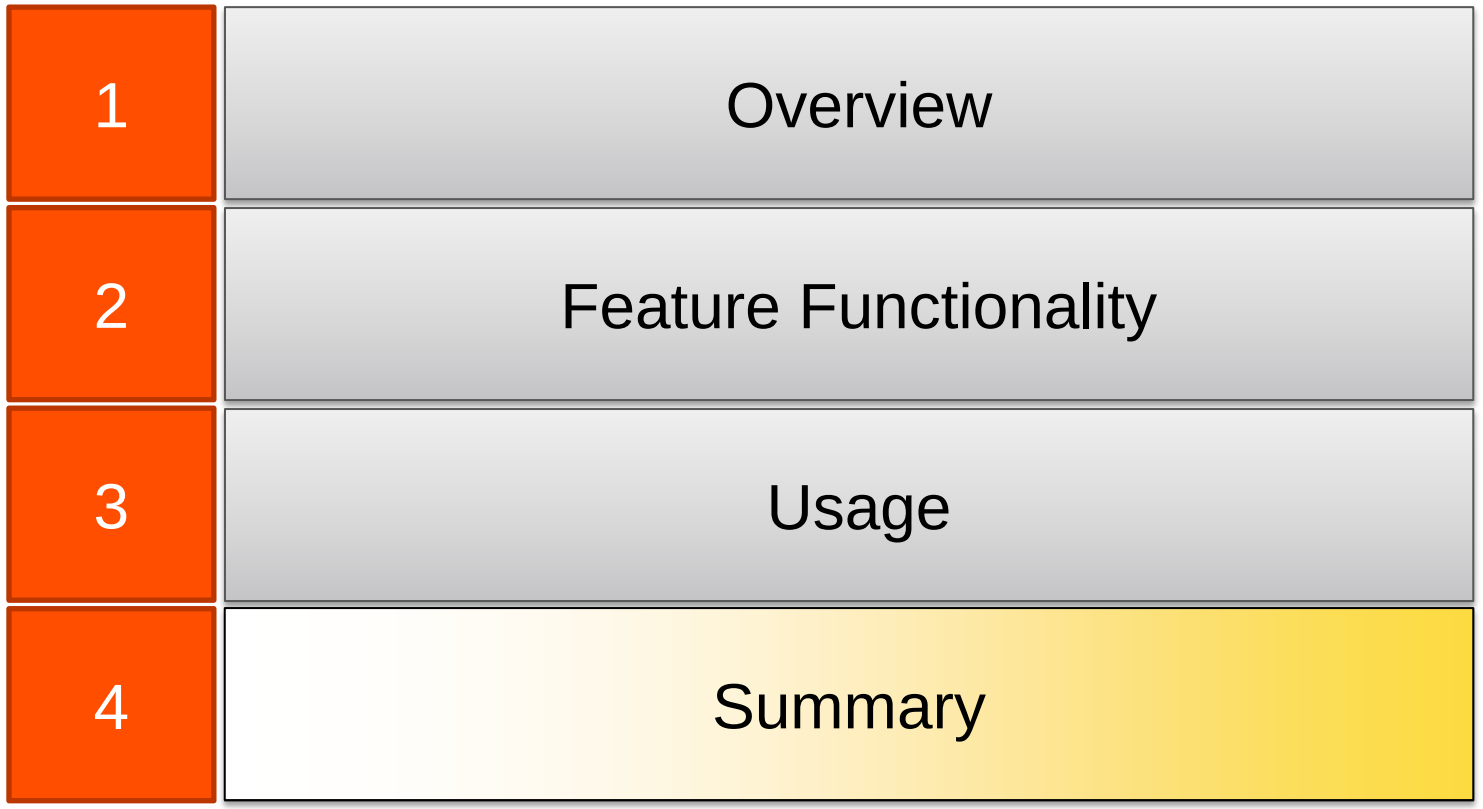


Table Partitioning Summary

Partition Types

- List Partitioning
- Range Partitioning
- List-range Partitioning
- Sub-partitioning
- Composite Partitioning

Indexing Support

- Global Index
- Local Index (aligned)
- Composite
- Automatic pruning

Migration

- In place migration
- No application changes
- List and range migration

Maintenance

- Existing Object level utilities made partition aware

New Maintenance

- Truncate / Delete
- View / Rename
- Split / Merge

Table Partitioning Early Software Access Program

Tech Preview coming: Q1 2014

Objective: Demonstrate basic table partition functionality

- Usual “Phased” rollout approach
- Capabilities*
  - Partition creation & maintenance
  - Migration examples

  * Subject to feature availability at the time
- White papers and examples
- Special presentations and webinars

- Participation
  - Access by enrollment only
  - For enrollment & more information
    - Contact: Rob Holzel: holzel@progress.com
      - OpenEdge Product Management

PROGRESS