Getting Started With Table Partitioning

Just a few steps away

Richard Banville
OpenEdge Development
Progress Software

The Steps to Table Partitioning

1. Why are you partitioning?
2. Identify tables
   a. Partitioning Strategy
   b. See white paper
3. Preparation
   a. Physical: Type II Storage Areas
   b. Code Cleansing
   c. Data Cleansing
   d. Index Considerations
4. Migration / adoption strategy
   a. Dump / Load
   b. “In place” migration
5. Current backup
6. Enable partitioning
   a. Database
   b. Tables / indexes / lobs
7. Split out existing data
8. Truncate / de-allocate “initial” partition

1. Why Are You Partitioning?

Advantages

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

Cautions

- Partition alignment & lookup (insert / delete)
  - Update of partition aligned key values
  - Missing aligned columns in where clause or record creation
- Getting it right the first time
  - Knowledge of application table definition & physical layout
  - Repartitioning costs
- More complex deployment (potentially)

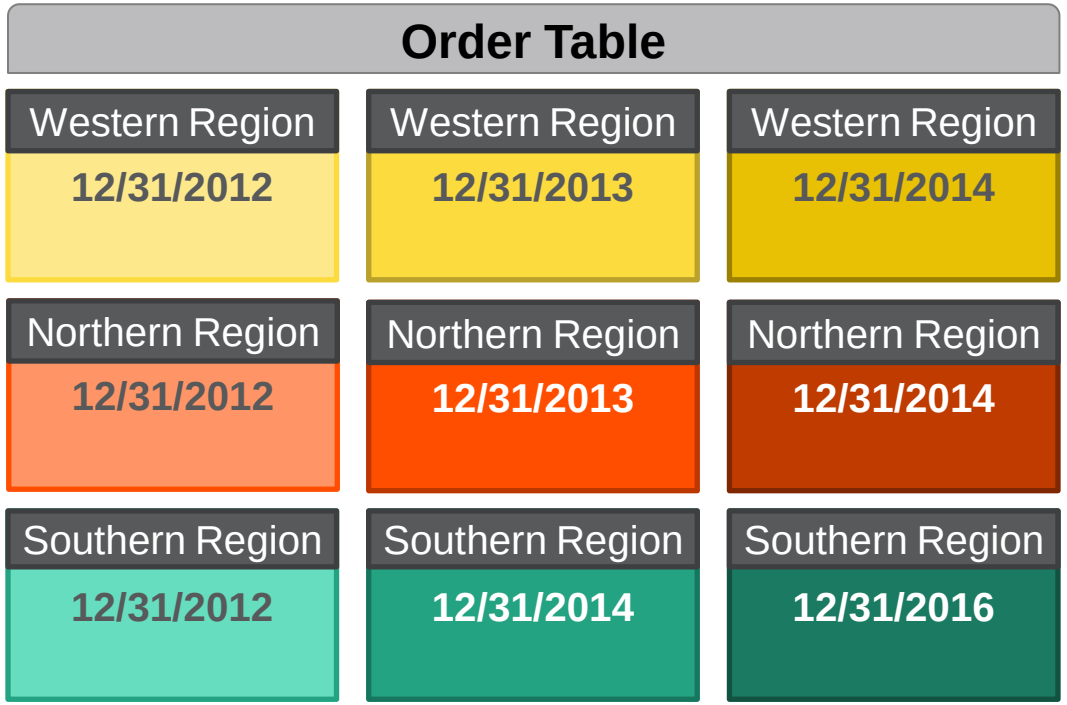
2. Identify Tables & Partitioning Strategy

- "Stable" data logically grouped by
  - Chronological events (range partitions)
  - Discrete list of values (list partitions)
  - Sub-partition partitioning
    - Partitioning the same table according to the values of more than one column
- Table / index maintenance too high
  - Operational time
  - Data availability
- For more information:
  - Presentation: Table Partitioning Application and Design
  - White paper: //community.progress.com/technicalusers/f/18/t/9294.aspx

2. Identify Tables & Partitioning Strategy

Sub-Partitioning

- Strategy first!
- Options: List, Range or Sub-partitioning
- Sub-partition by region & order-date w/in region
- 9 partition example



- Only last partitioned column may be a range partition
  - Range partition can be any “indexable” field type
- List / ranges must be inclusive, not necessarily symmetrical
- Storage considerations for new partitions

3. Preparation: Physical Layout and Creation

- Type II Storage Areas
  - All existing data (Table, indexes, lobs)
  - New storage considerations
    - # Storage areas
    - Extent location
    - Cluster size
    - Records per block
    - Toss limits
    - Buffer pool assignment

3. Preparation: Code Cleansing

- Recid / Rowid usage
  - Recids will NOT work
  - Rowids can change at runtime
- Record Creation
  - Partition *field values* must have associated partition definition
    - Partition columns cannot be UNKNOWN
  - Avoid multiple assign statements
  - Use partition defined initial values

NOTE: Same r-code with or without table partitioning

PROGRESS SPARK

3. Preparation: Data Cleansing

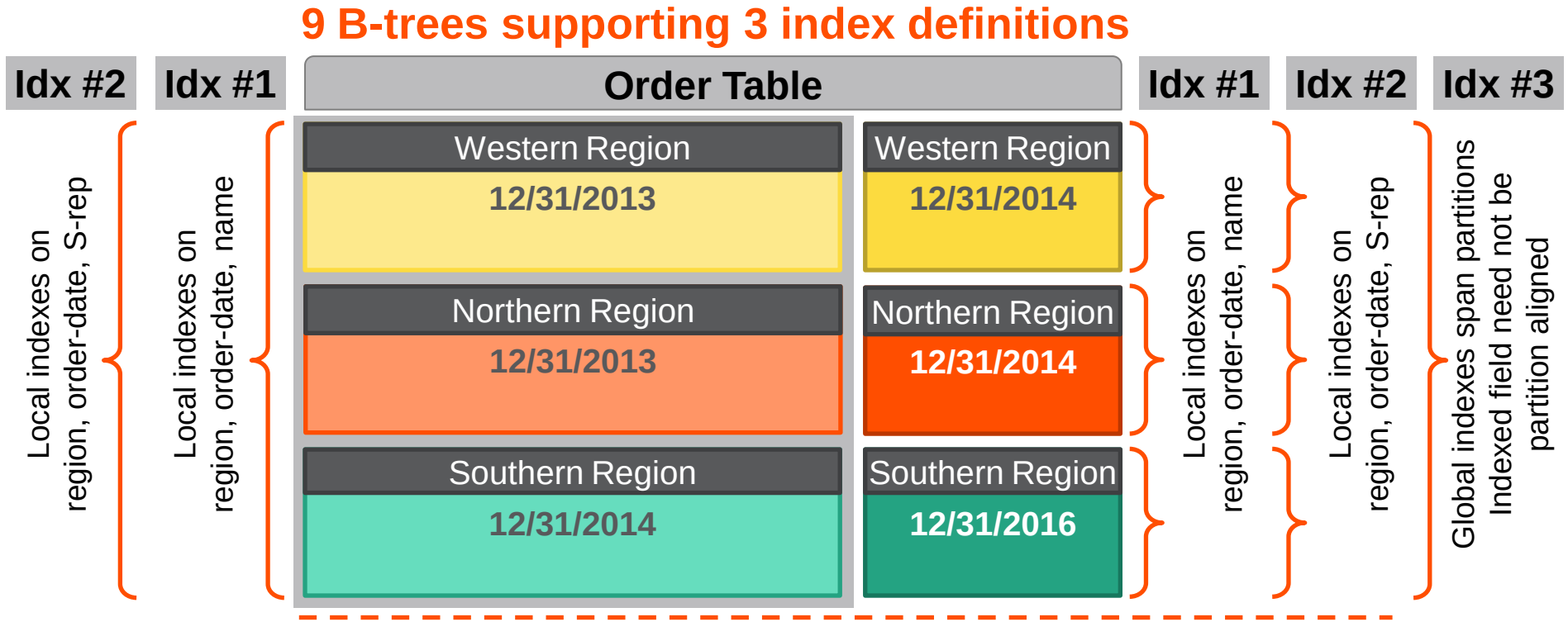
- Partition aligned field values
  - Existing partition *field values* must be associated with defined partition
  - Columns in partition definition cannot be UNKNOWN
- Purge first
- Update existing data based on partition scheme

Further details in *TP Application and Design* presentation

3. Preparation – Index Support

Index Support

- Sub-partition by region & order-date w/in region
- 4 partition example
- Minimum of 1 partition aligned index



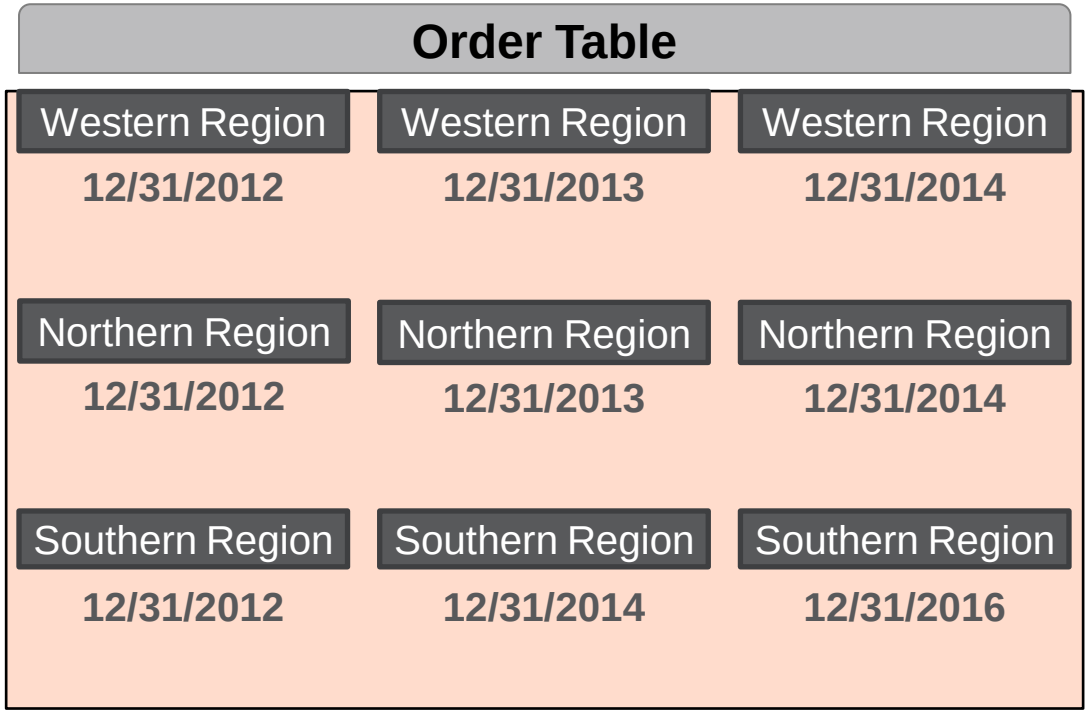
Index	Components	Partition Aligned
Index #1 (local)	{Region, Order-Date, Name}	YES
Index #2 (local)	{Region, Order-Date, S-rep}	YES
Index #3 (global)	{Cust-num}	NO

PROGRESS SPARK

4. Determine Migration/Adoption Strategy – New Table (Dump / Load)

Sub-Partitioning

- 9 partition example
- Sub-partition by region & order-date w/in region
- All data currently in ONE partition (partition 0)



1. Dump data*
2. Drop table*
3. Add partitioned table
4. Define / Add partitions
5. Load data
6. You're good to go!

PROGRESS SPARK

4. Determine Migration/Adoption Strategy – New Tables (Dump / Load)

Sub-Partitioning

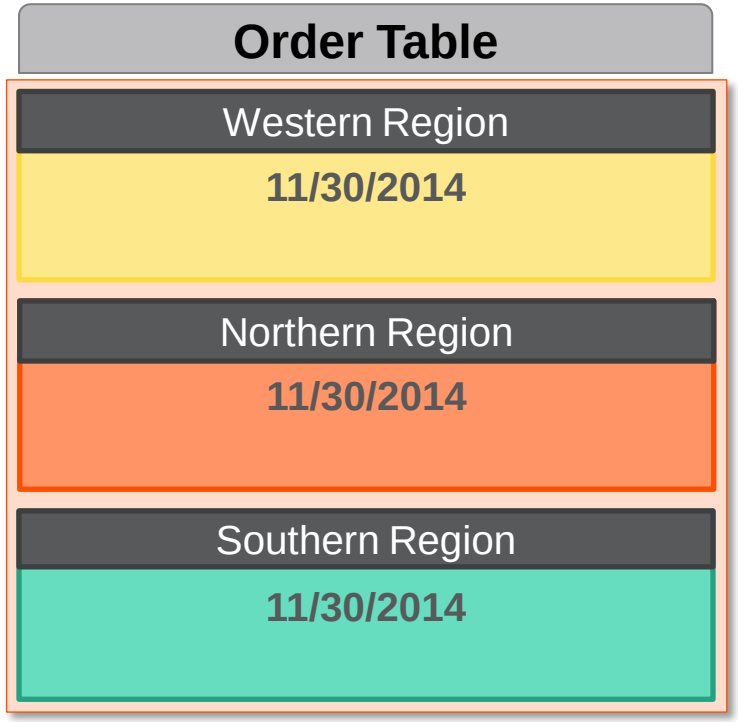
- 9 partition example
- Sub-partition by region & order-date w/in region

1. Dump data*
2. Drop table*
3. Add partitioned table
4. Define / Add partitions
5. Load data
6. You're good to go!

4. Determine Migration/Adoption Strategy – In Place Migration

Composite Partitioning

- Single composite partition example
- Sub-partition by region & order-date w/in region
- All data currently in ONE partition (partition 0)



Partition 0

Existing Data
logically segregated

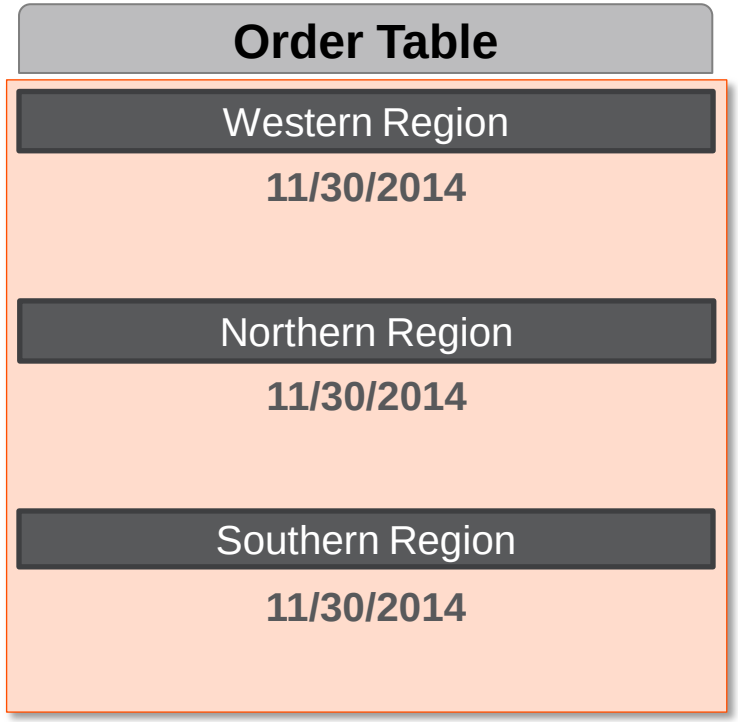
- Implemented for migration only
- Tool creates list of unique entries
- Can modify range of given list

PROGRESS SPARK

4. Determine Migration/Adoption Strategy – In Place Migration

Composite Partitioning

- Single composite partition example
- Sub-partition by region & order-date w/in region
- All data currently in ONE partition (partition 0)



Partition 0

Existing Data
logically segregated

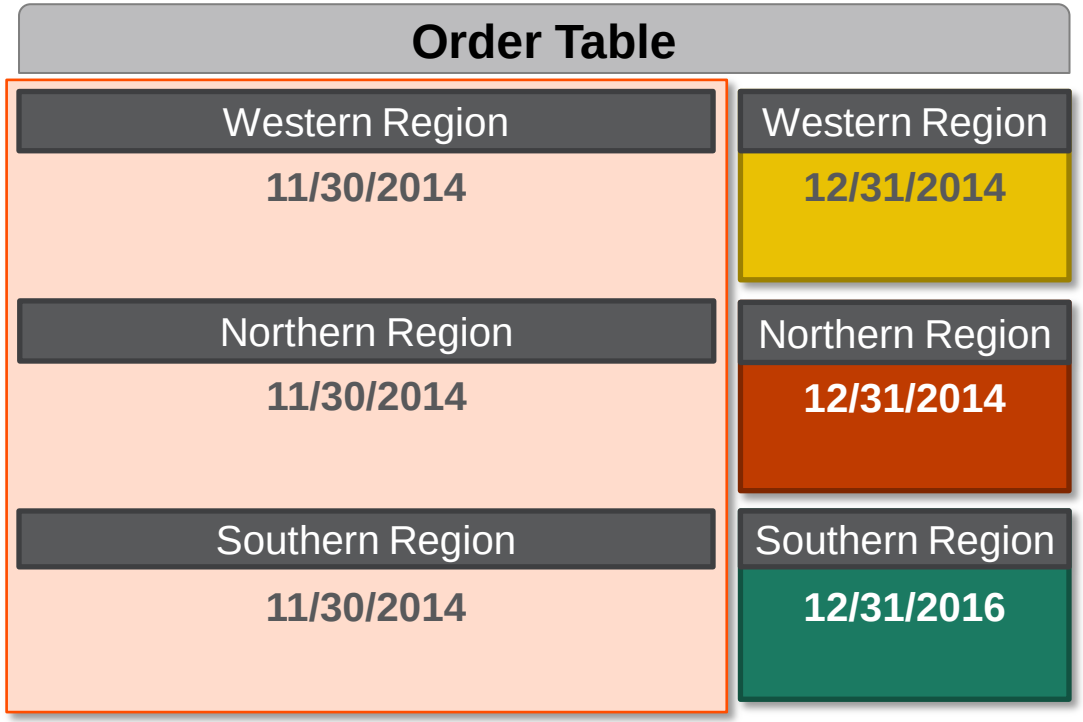
- Implemented for migration only
- Tool creates list of unique entries
- Can modify range of given list

PROGRESS SPARK

4. Determine Migration/Adoption Strategy – In Place Migration

Composite Partitioning

- Single composite partition example
- Sub-partition by region & order-date w/in region
- All current data in ONE partition (partition 0)
- New data goes to new partition



Partition 0

Existing data
logically segregated

Partition 1–3

New data
physically segregated

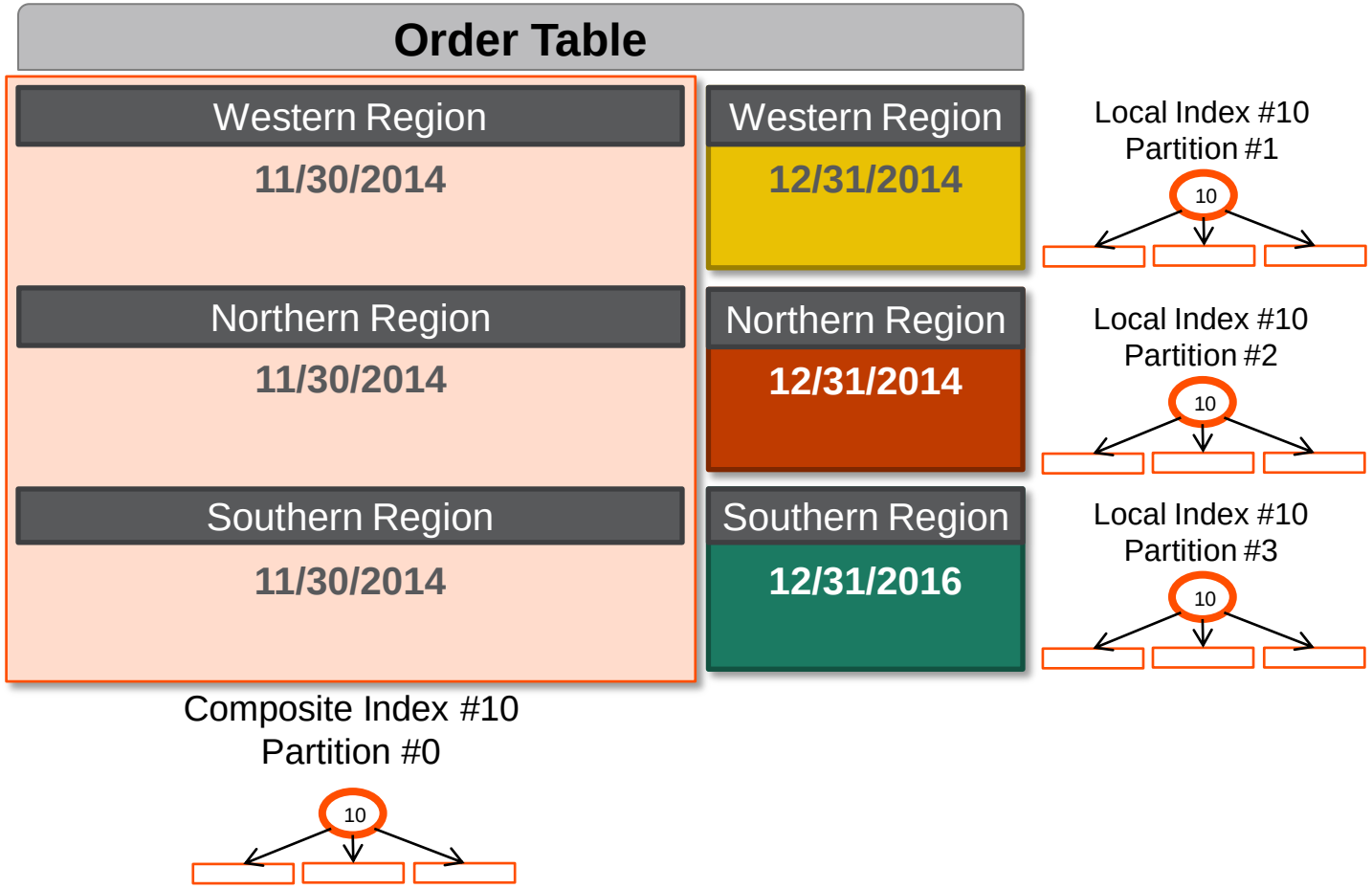
- Implemented for migration only
- Tool creates list of unique entries
- Can modify range of given list

PROGRESS SPARK

4. Determine Migration/Adoption Strategy – In Place Migration

Composite Partitioning

- Single composite partition example
- Sub-partition by region & order-date w/in region
- Existing aligned indexes marked as “local”
- New local b-trees created



- Implemented for migration only
- Tool creates list of unique entries
- Can modify range of given list

5) Have a good backup! 6) Enable partitioning – Database level

proutil <*db*> -C enabletablepartitioning

- _Partition-Policy (-352)
  - Describes partition at the “table” level
  - Lookup requires Table #

Column	Name	Type
2	_Partition-Policy-Name	char
3	_Object-Number	Integer
4	_DataArea-default	Integer
5	_IndexArea-default	Integer
6	_LobArea-default	Integer
7	_Allocation-default (None, immediate, delayed)	Char
8	_Num-Columns	Integer
9	_Column-Name	char16]
10	_Has-Range	Logical
11	_Description	char
12	_Misc	char[16]

- _Partition-Policy-Detail (-353)
  - Defines each individual partition
  - Lookup requires Table # AND PartitionId

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]

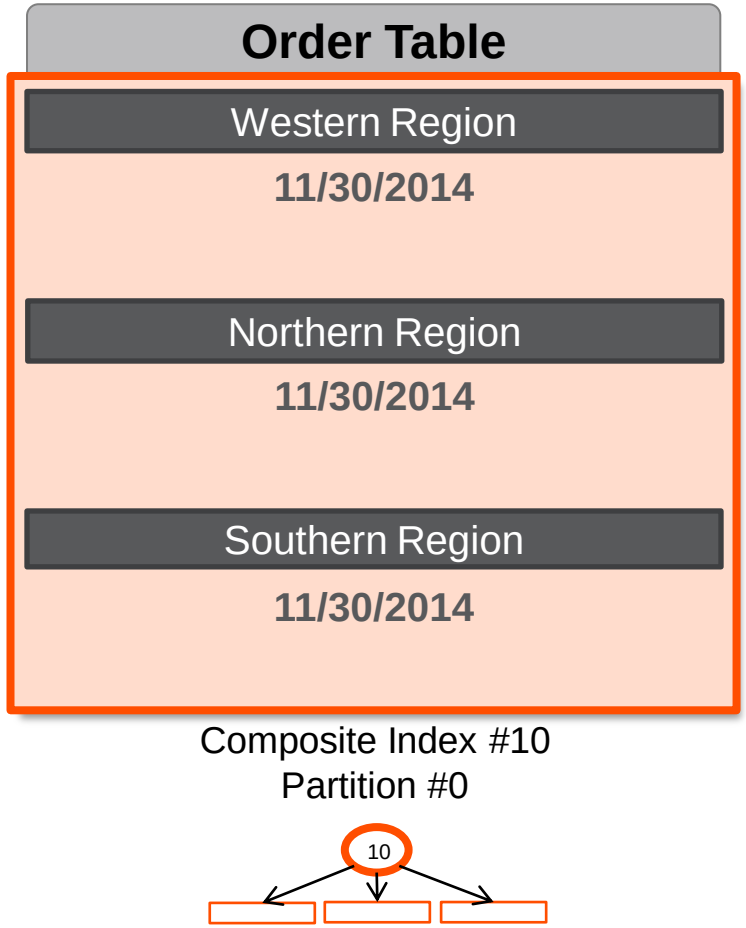
PROGRESS SPARK

6. Enable Partitioning – Table and Indexes

In Place Migration

- Sub-partition by region & order-date w/in region
- Single composite partition
- Composite partition aligned “local” index

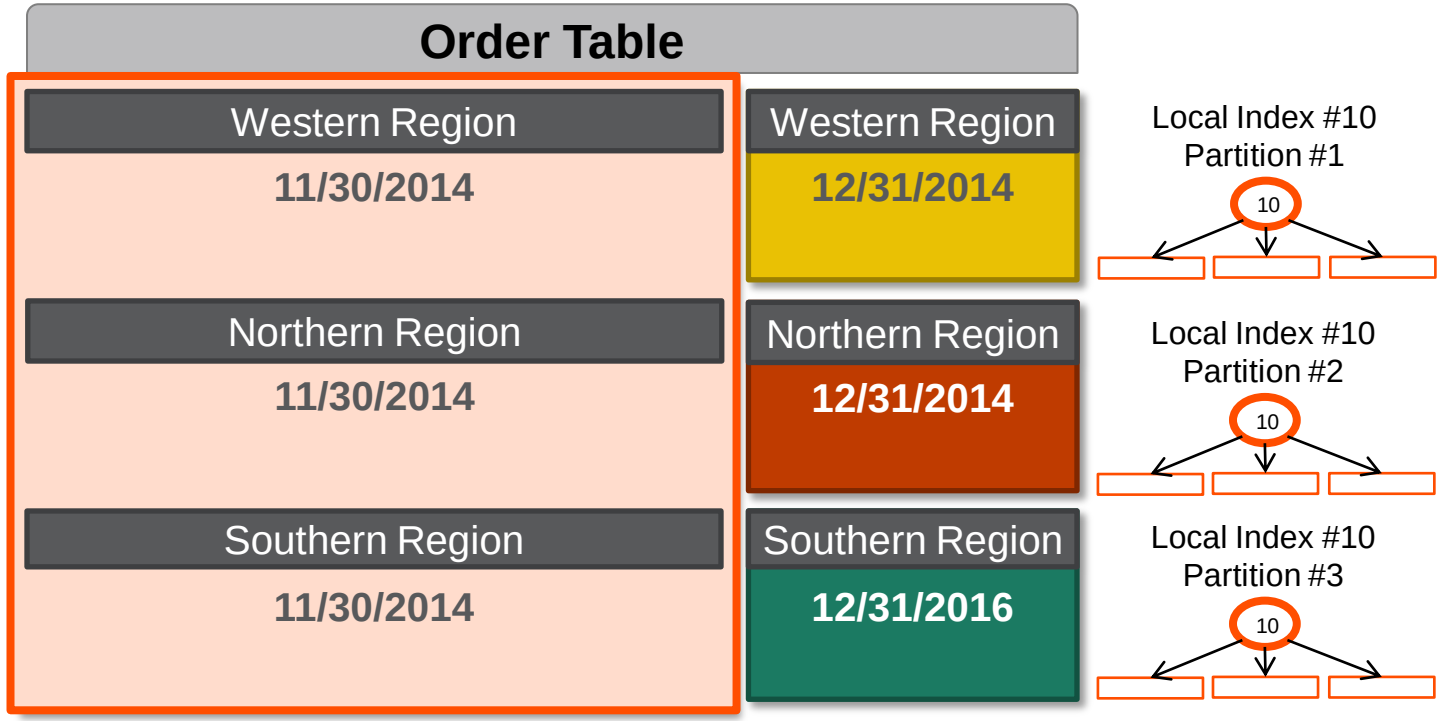
- Storage considerations for new partitions



6. Enable Partitioning – Table and Indexes

In Place Migration

- Sub-partition by region & order-date w/in region
- Composite partition
- 3 newly added partitions
- Partition aligned index b-trees created



- Storage considerations for new partitions
- New data to new partitions

Partition 1–3

New data
physically segregated

7. Split Out Existing Data (Optional)

- Identify created partition as a “split target”
- Move data into target partitions

```
proutil <db> -C partitionmanage split table <name>
                partition  <name>  | composite "initial“
                useindex <name>    recs <number>
```

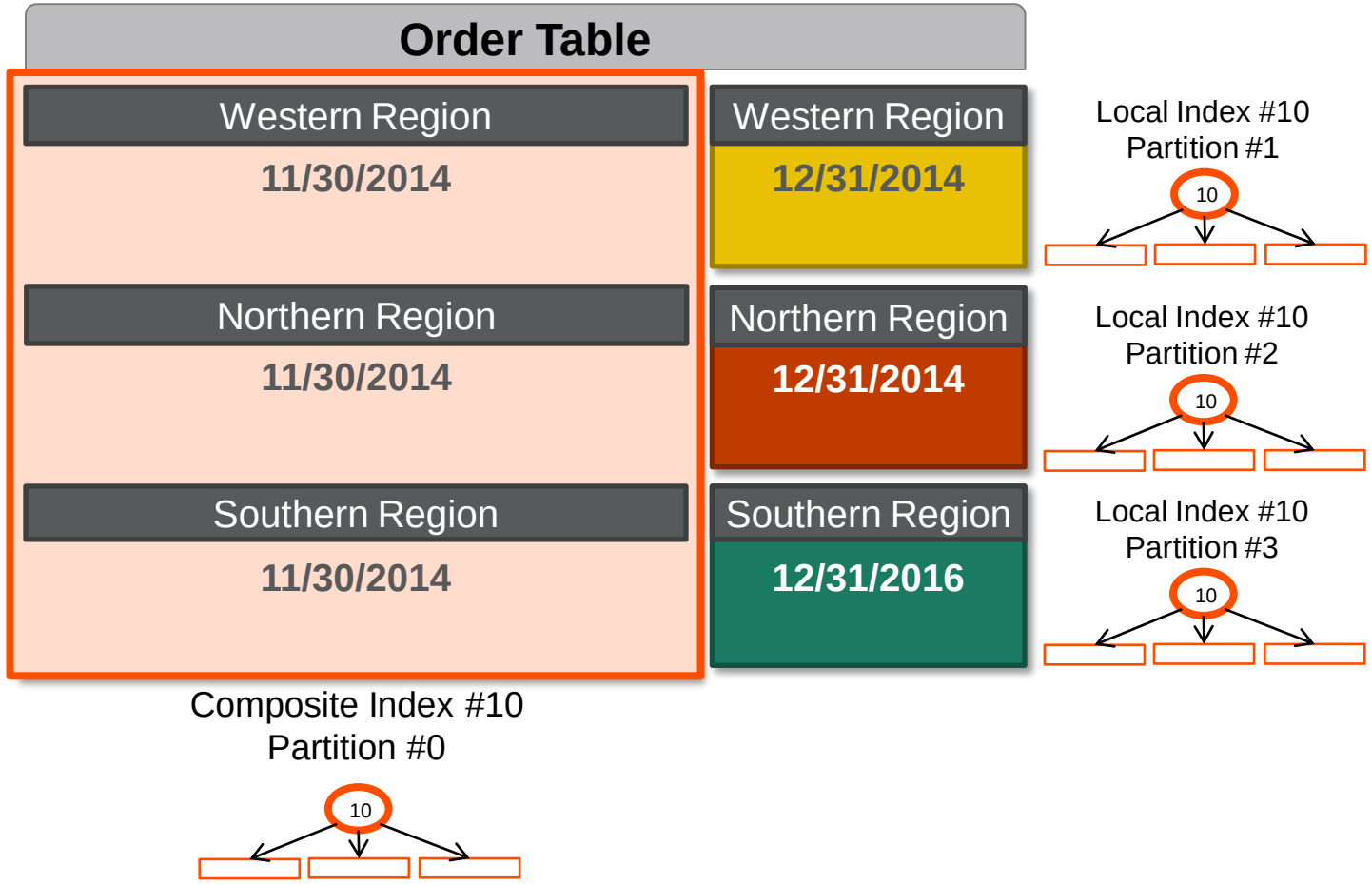
- Transactional scoping by groups of record/index operations
  - Data for same partition definition spans physical partitions
  - Only ever one copy of the data
- Recovery of operation restarts where it left off
- Online operation with full access to non-split data
  - New split transitional state for partitions
- Multiple concurrent operations allowed
- OpenEdge Replication fully supported

7. Split Out Existing Data

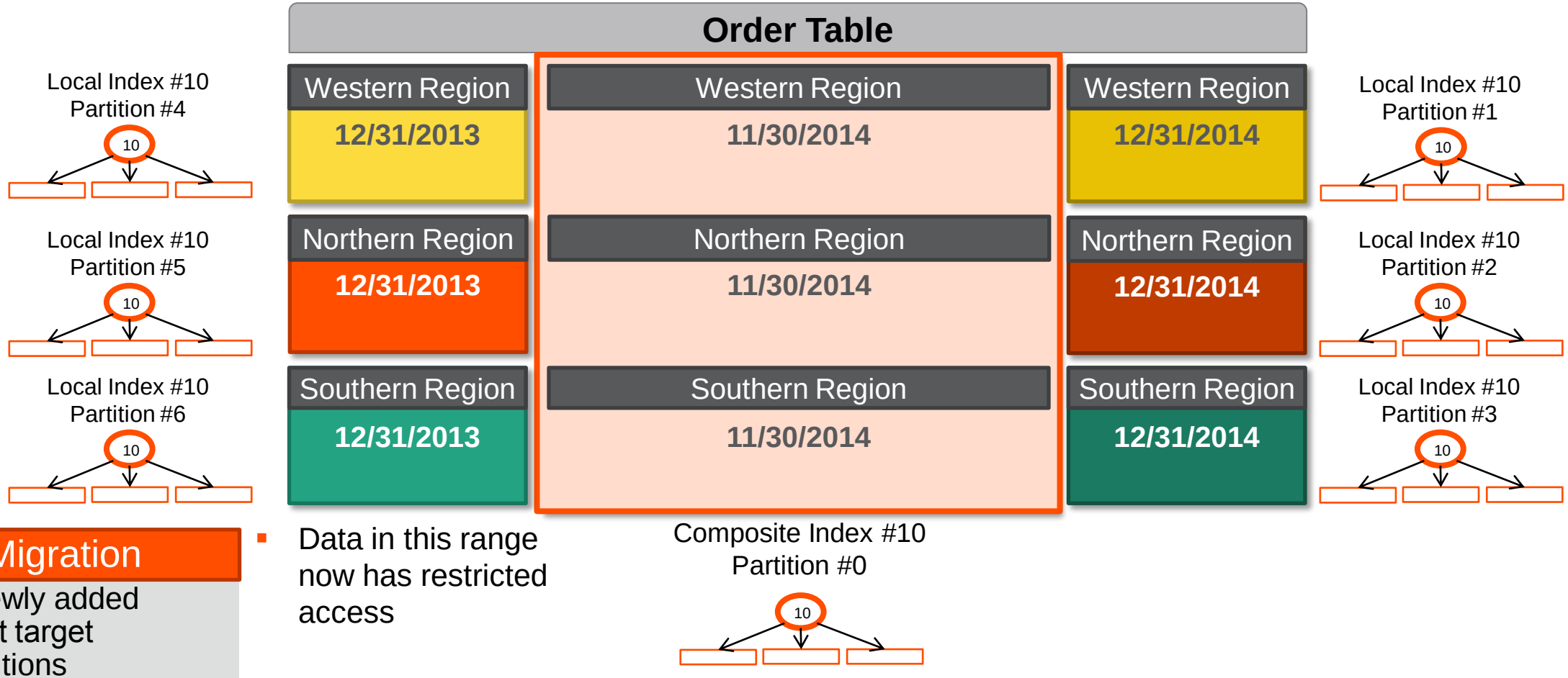
Migration

- Composite partition
- 3 newly added partitions
- Partition aligned indexed
- Sub-partition by region & order-date w/in region

- Create new partitions
- Mark as split target" via OpenEdge Management



7. Split Out Existing Data



Migration

- 3 newly added "split target partitions
- New partition aligned indexes

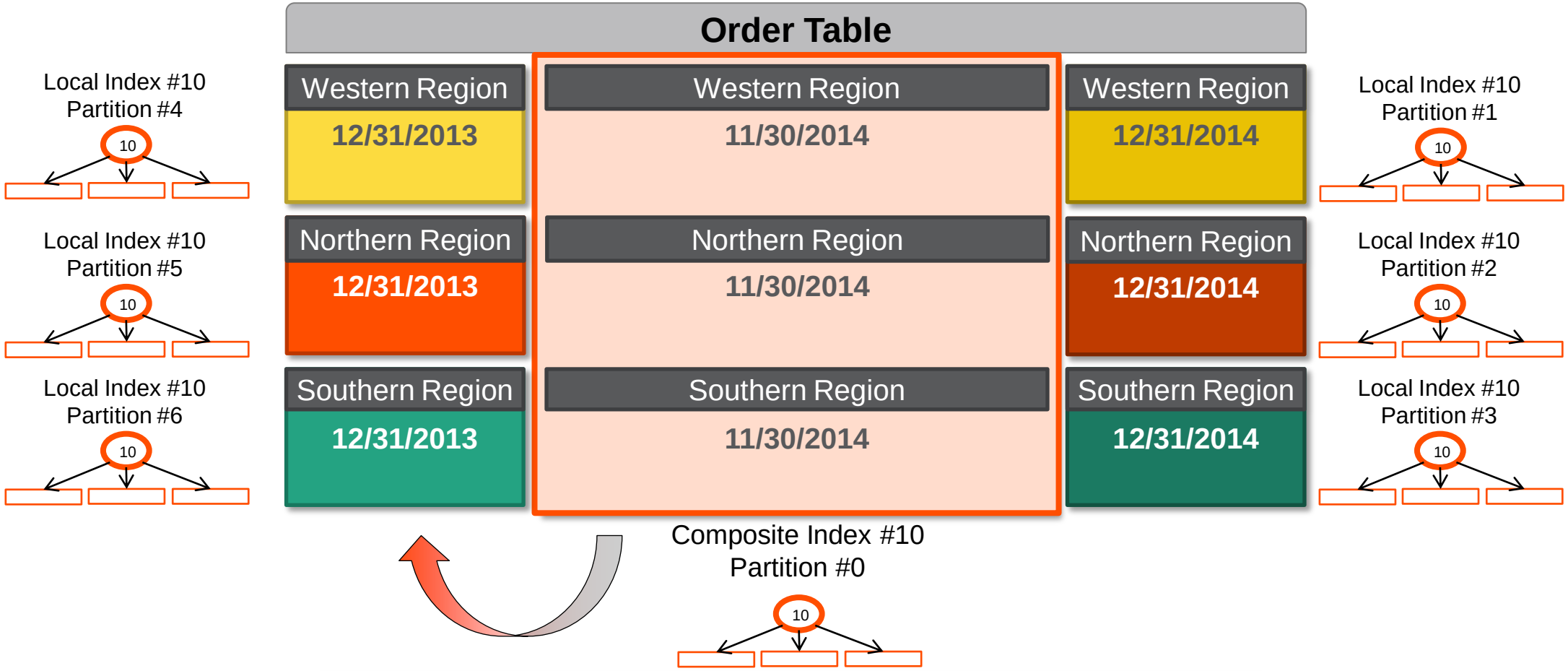
- Data in this range now has restricted access

Composite Index #10 Partition #0

- Create new partitions
- Mark as split target" via OpenEdge Management

PROGRESS SPARK

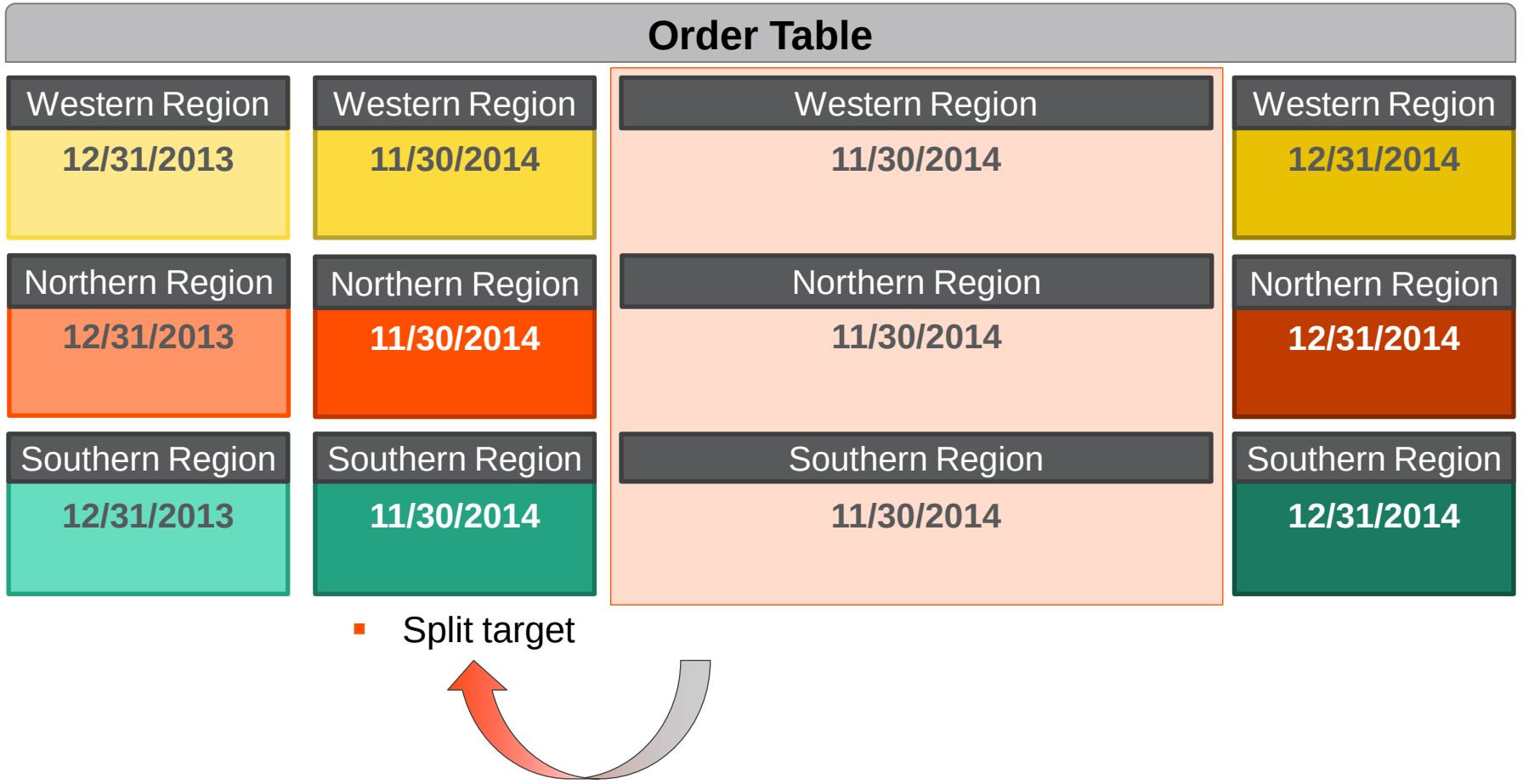
7. Split Out Existing Data



Migration

proutil <*db*> -C partitionmanage split table *order* composite *"initial"* recs *1000*

7. Split Out Existing Data

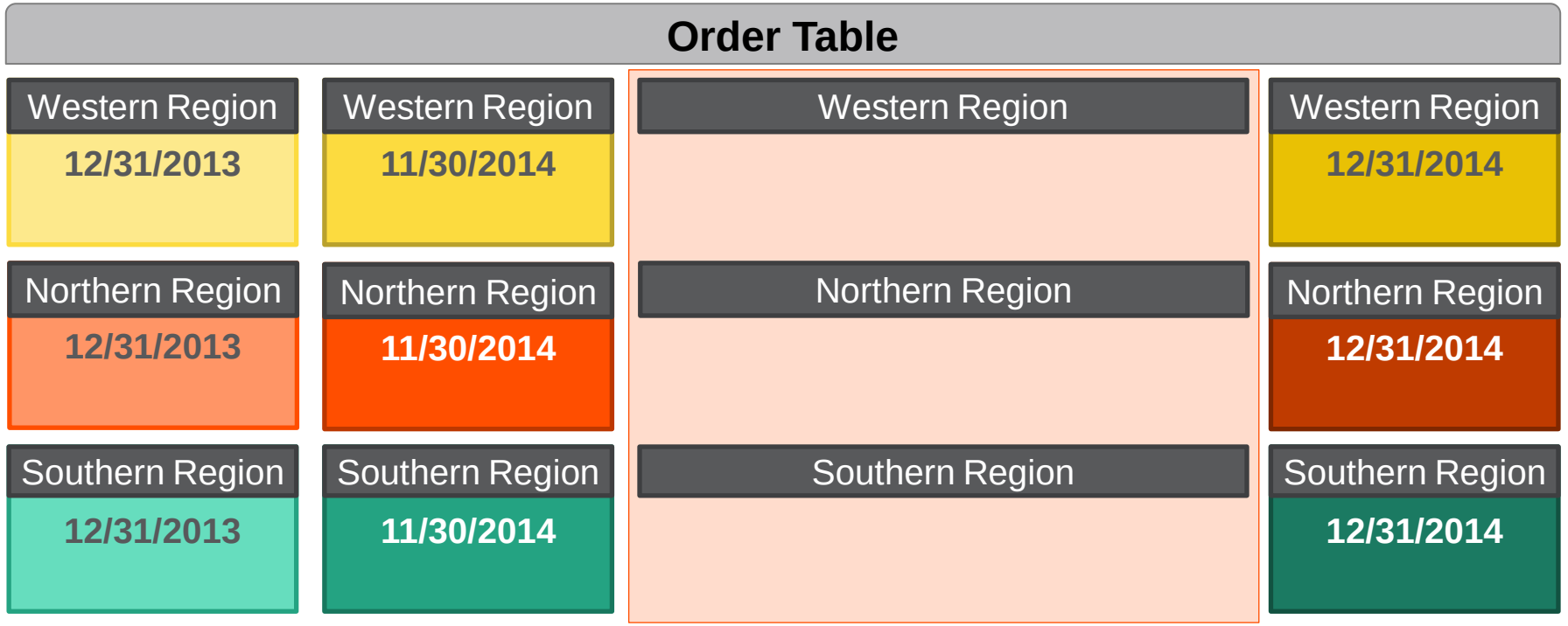


- Split target

Migration

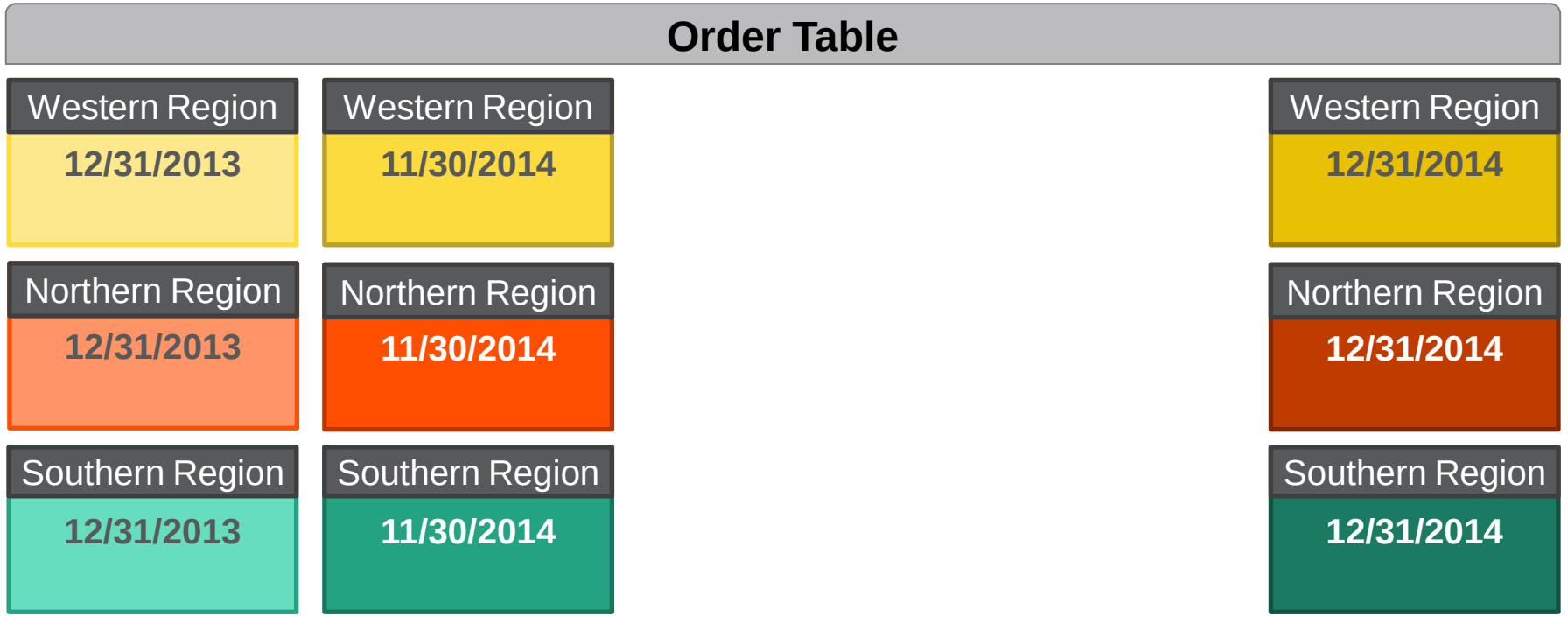
proutil <*db*> -C partitionmanage split table *order* composite *"initial"* recs *1000*

8. Truncate / De-Allocate “Initial” Partition



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

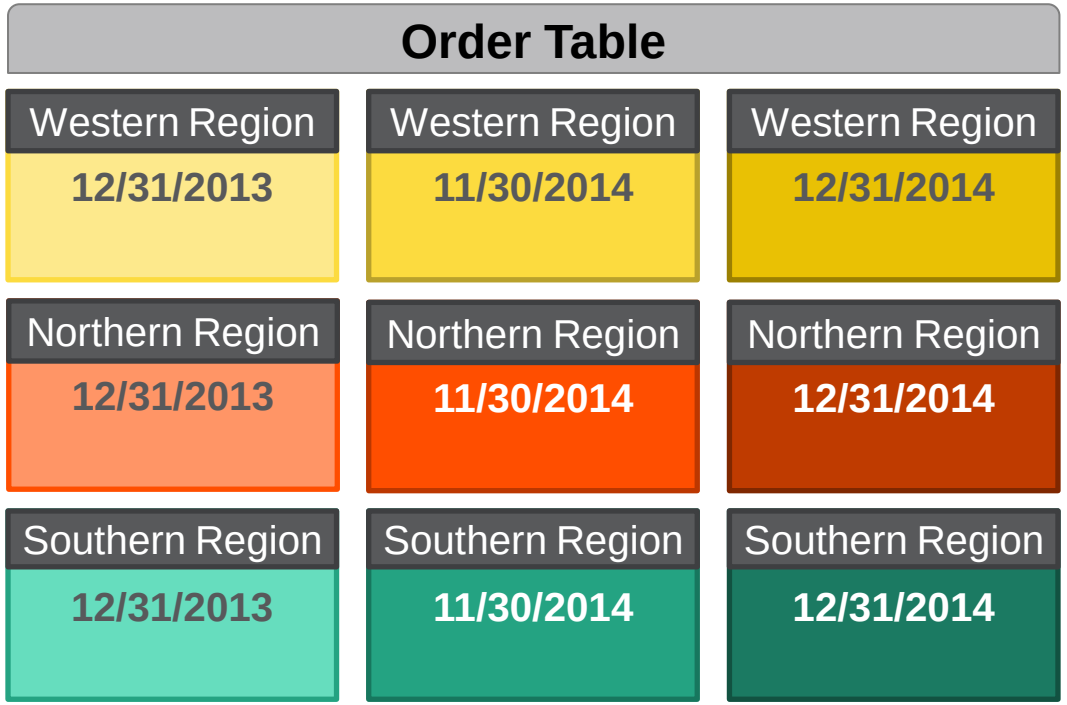
8. Truncate / De-Allocate “Initial” Partition



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

8. Truncate / De-Allocate “Initial” Partition

Space Reclaimed



Demo Time!

New Course: Implementing Progress OpenEdge Table Partitioning

- Description: This course teaches the key tasks to partition tables in an OpenEdge RDBMS database. First, you will be introduced to the concepts, types, and tasks of OpenEdge table partitioning. Then, you will learn how to prepare for table partitioning and enable partitioning for a database. Next, you will learn how to create new partitioned tables and partition existing non-partitioned tables. Finally, you will learn how to manage partitions, maintain indexes, and gather statistics for partitioned tables and indexes.
- Course duration: Equivalent to 2 days of instructor-led training
- Audience: Database Administrators who want to partition Progress OpenEdge RDBMS tables
- Version compatibility: This course is compatible with OpenEdge 11.4.
- After taking this course, you should be able to:
  - Describe Progress OpenEdge table partitioning.
  - Create new partitioned tables
  - Partition existing tables
  - Manage partitions
  - Maintain indexes
  - Gathering statistics for partitioned tables and indexes

PROGRESS SPARK


APJ
PROGRESS SPARK

Want to Learn More About OpenEdge 11?

- Role-based learning paths are available for OpenEdge 11
- Each course is available as Instructor-led training or eLearning
- Instructor-led training:
  - **$500 per student per day**
  - https://www.progress.com/support-and-services/education/instructor-led-training
- eLearning:
  - Via the Progress Education Community (https://wbt.progress.com):
  - OpenEdge Developer Catalog**: $1500 per user per year**
  - OpenEdge Administrator Catalog: **$900 per user per year**
- User Assistance videos: https://www.progress.com/products/pacific/help/openedge

APJ PROGRESS SPARK