WINDAYS

TECHNOLOGY

# LAYERED PARTITIONING

Josip Šaban, Hypo Alpe Adria AG, Klagenfurt

Microsoft

# About

- Finished FER in year 2004, Cotrugli MBA in year 2011
- Holds all certificates for SQL Server since version 2000
- Also holds .NET, Project Server and Biztalk certifications
- Certified PM on CompTIA and PRINCE2 methodologies
- Works as a senior BI analyst in Hypo-Alpe-Adria, Klagenfurt
- Also tries to help grow his own company in Zagreb
- Finishing doctorial studies on FOI and preparing PMP exam

# Summary

Partitioning is the one of the key methods that can enhance query performance, but there is no guarantee - we show the why and how of partitioning in very large data sets.

Other key methods we need are indexing and data compression.

# A real – world case

- Help! My table is getting too big!
- One installation has a reporting system containing both fact and dimension tables
  - Fact tables are 100+ GB in size (including all indexes) quickly approaching 1 billion rows
  - Dimension tables are less than 1 GB in size (including all indexes) with less than 100 thousand rows
- A middle tier application has been designed to dynamically create and execute queries to run reports custom-designed by clients

# What are very large data sets?

- There is no official definition
- Typically occupying TB range, OLAP or OLTP with large amount of users
- Billions of rows

# What is too big where tables are concerned?

- SQL Server can store Multi-TB tables, so the answer is always contextual
- Big enough to cause performance issues (on the *current* hardware)
- Big enough to cause maintenance operations to take too long or simply not be practical at all (on the *current* hardware)
- Big due to containing large amounts of historic / “processed” / “completed" data

# What options do I have?

- Option 1: Do nothing (the told you so/job gamble option)
- Option 2: Upgrade the hardware (the sticky plaster option)
- Option 3: If no extra disk space is available (the bail out option):
  - Delete the older portion of the data
  - Move the older portion of the data to Windows SQL Database (Azure)
- Option 4: If disk space is available (the scale out option):
  - Partition the data
    - Distributed partitioned views
    - Partitioned views
    - Partitioning (Enterprise only)

# Agenda

- Setting the stage
  - Introduction to indexing
  - Introduction to compression
- Main event…partitioning in very large data sets
  - Change the way of thinking and designing large tables
  - Introduction to partitioning
  - Advanced partitioning

# Setting the stage

XV
WINDAYS
TECHNOLOGY

Microsoft

The story of indexing

# INTRODUCTION TO INDEXING

# Indexing basics

- Index Considerations
  - Can dramatically increase query performance
  - Adds overhead for index maintenance
- Best Practices
  - Base design on real-world workloads
  - Scenarios:
    - Retrieving ranges of data
    - Retrieving specific values

# Basic indexes

- Clustered index
  - – Controls the physical order of rows
  - – Does not require disk space
  - – One per table (may include multiple columns)
  - – Created by default on tables’ Primary Key column

# Basic indexes

- Non-Clustered Index
  - Physical data structures that facilitate data retrieval
  - Can have many indexes
  - Indexes may include many columns
  - Covering index
  - Filtered index

# Advanced indexes

- FULLTEXT index
  - Stores the info about significant words and their location within the columns of a database table
- SPATIAL Index
  - Provides the ability to perform certain operations more efficiently on spatial objects (spatial data or Geometry data type) in a column of the geometry data type

# Advanced indexes

- XML Index
  - XML indexes can be created on xml data type columns
- Columnstore Index
  - New feature of SQL 2012
  - This index does not use the B-Tree structure but column-oriented storages
  - https://msdn.microsoft.com/en-us/library/gg492088.aspx

The story of compression

# INTRODUCTION TO COMPRESSION

# Basic compression

- Why compress
  - – Disk throughput much slower than memory and CPU
  - – Need less disk space
  - – Two kinds of compression
    - Backup compression ( not discussed here )
    - Data compression

# Enabling compression

- Alter Table [TableName] Rebuild Partition = All with (Data_compression = Compression Type on Partitions (x to n))
    - Compression Types
        - Row
        - Page
        - None
- Alter Index [IndexName] on [TableName] Rebuild with (Data_compression = Compression Type)
    - Compression Types
        - Row
        - Page
        - None

# Page compression – used in DWH

Pre-Fix

Dictionary

CI structure

Page Header

| | | |
|---|---|---|
| aaabb<br>0 4b | aaaab<br>0 4b | abcd |
| aaabcc | bbbb<br>0bbbb | abcd |
| aaaccc<br>3ccc | aaaacc | bbbb<br>0bbbb |

WINDAYS TECHNOLOGY

Microsoft

# Database Compression Cost/Benefit Analysis

| Benefits | Cost |
|---|---|
| **Performance improvements** | **Increased CPU utilisation** |
| • More data in memory | |
| • Reduced I/O | |
| **Reduced disk space usage** | |
| • Database data files | |
| • Backup files | |
| **Reduced time to backup** | |
| **Cost Savings** | |

# How to choose

- sp_estimate_data_compression_savings
- Quick Rule of thumb :
  - ROW is low-cost, generally 10% CPU overhead. Use it on everything on OLTP
  - PAGE is more expensive, but compresses more. Use it on everything on DWH
- Compression strategy documentation
  - https://technet.microsoft.com/en-us/library/dd894051(v=sql.100).aspx

# How to choose - effectiveness

- Good Compression

- Numeric or fixed length character fields that don’t use all the allocated bytes
- Repeating data or prefix values

- Poor or no Compression

- Fields using up all the allocated bytes
- Not much repeated data
- Repeated with non-repeating prefixes
- Out of row data
- FILESTREAM data

# The main event

Why and how of partitioning

# INTRODUCTION TO PARTITIONING

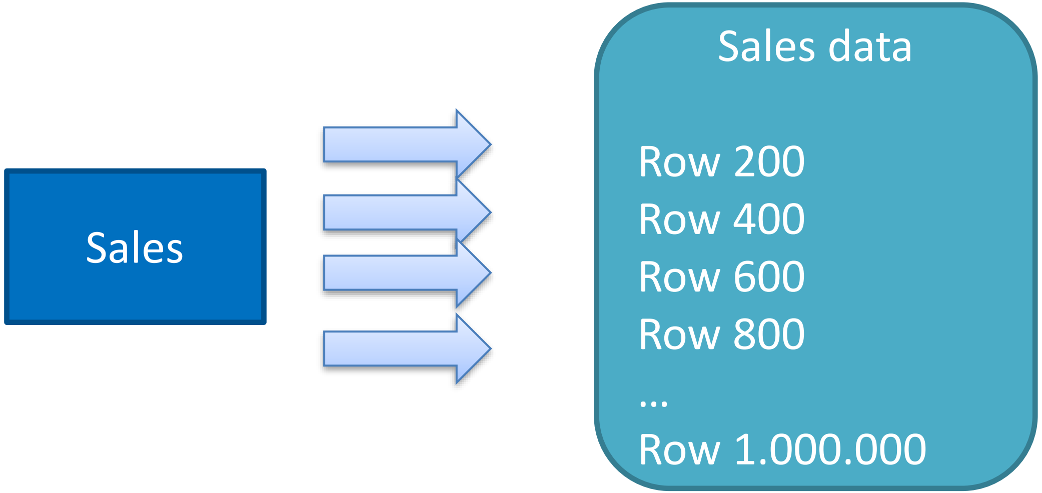
# What is partitioning?

- In SQL Server all tables have at least one partition
  - "In SQL Server, all tables and indexes in a database are considered partitioned, even if they are made up of only one partition. Essentially, partitions form the basic unit of organization in the physical architecture of tables and indexes. This means that the logical and physical architecture of tables and indexes comprised of multiple partitions mirrors that of single-partition tables and indexes."
  - Partitioned Table and Index Concepts
    - https://msdn.microsoft.com/en-us/library/ms190787.aspx
  - Partitioned Table and Index Strategies Using SQL Server 2008
    - http://msdn.microsoft.com/en-us/library/dd578580.aspx

# Advantages of partitioning

- Improves Query Performance
- Better Data Manageability and Cost Effective
- Deleting and Moving data is faster from partitions
- Narrow downs the index maintenance window
- OLTP/DSS – Operational/Non-Operational data

# A standard table...

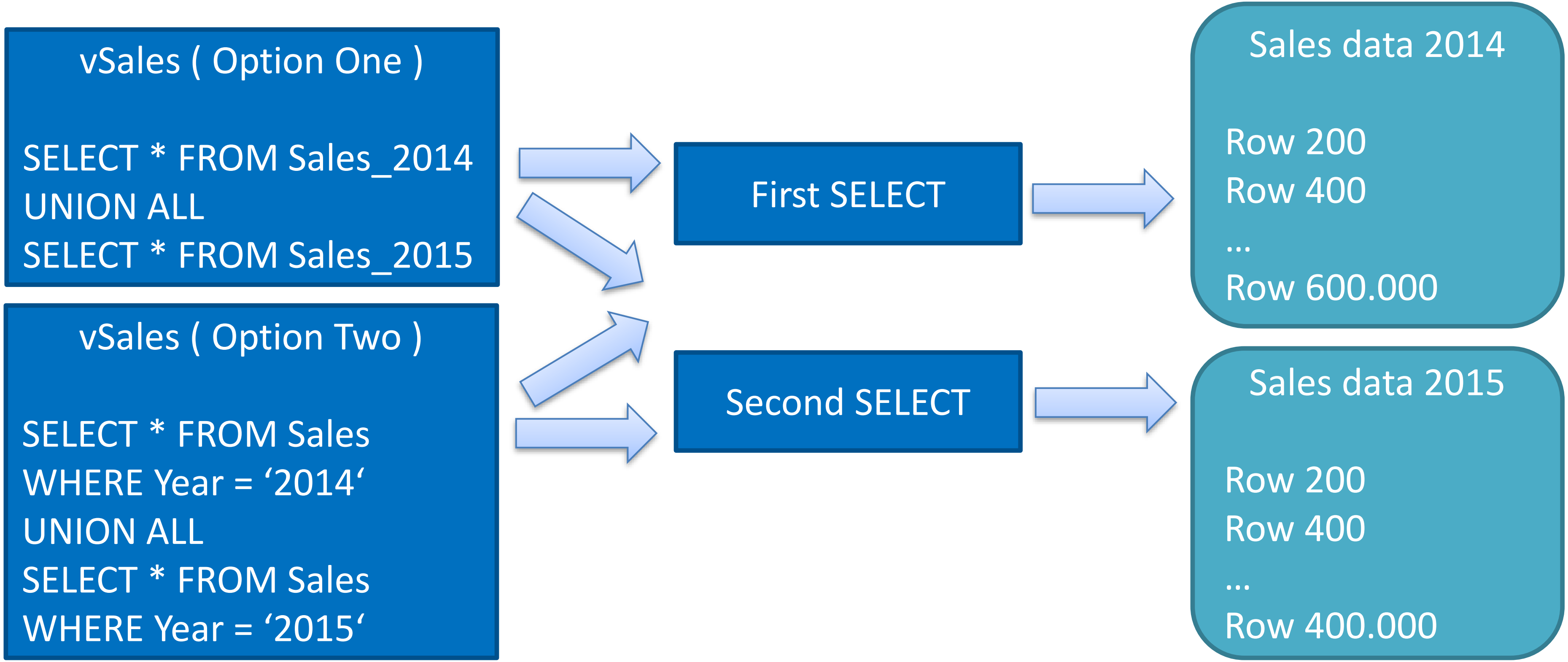


Sales

Sales data

Row 200
Row 400
Row 600
Row 800
…
Row 1.000.000

# Partitioned views…



vSales ( Option One )

```
SELECT * FROM Sales_2014
UNION ALL
SELECT * FROM Sales_2015
```

vSales ( Option Two )

```
SELECT * FROM Sales
WHERE Year = ‘2014‘
UNION ALL
SELECT * FROM Sales
WHERE Year = ‘2015‘
```

First SELECT

Second SELECT

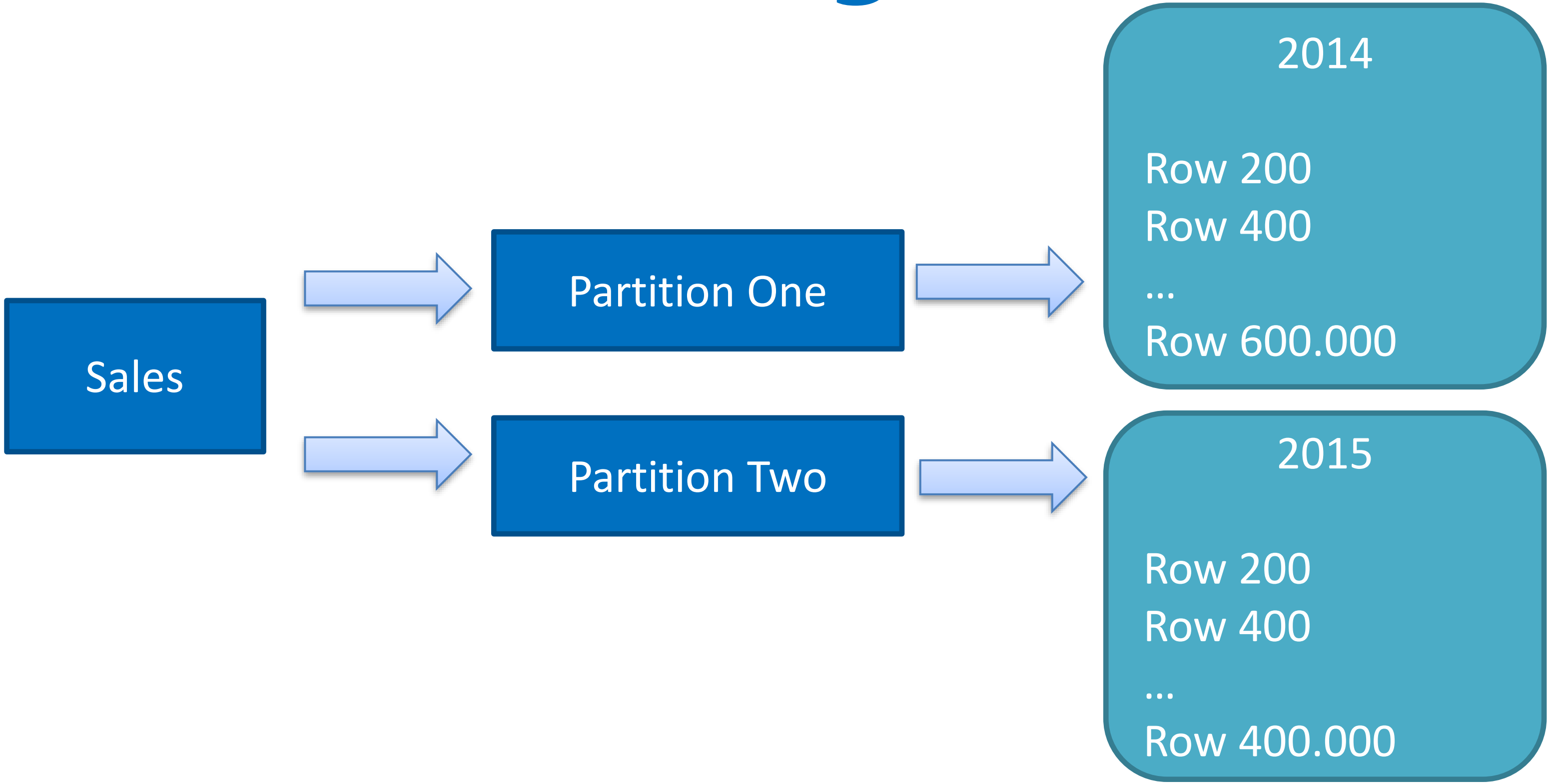
Sales data 2014

Row 200
Row 400
…
Row 600.000

Sales data 2015

Row 200
Row 400
…
Row 400.000

# Partitioning…



Sales

Partition One

Partition Two

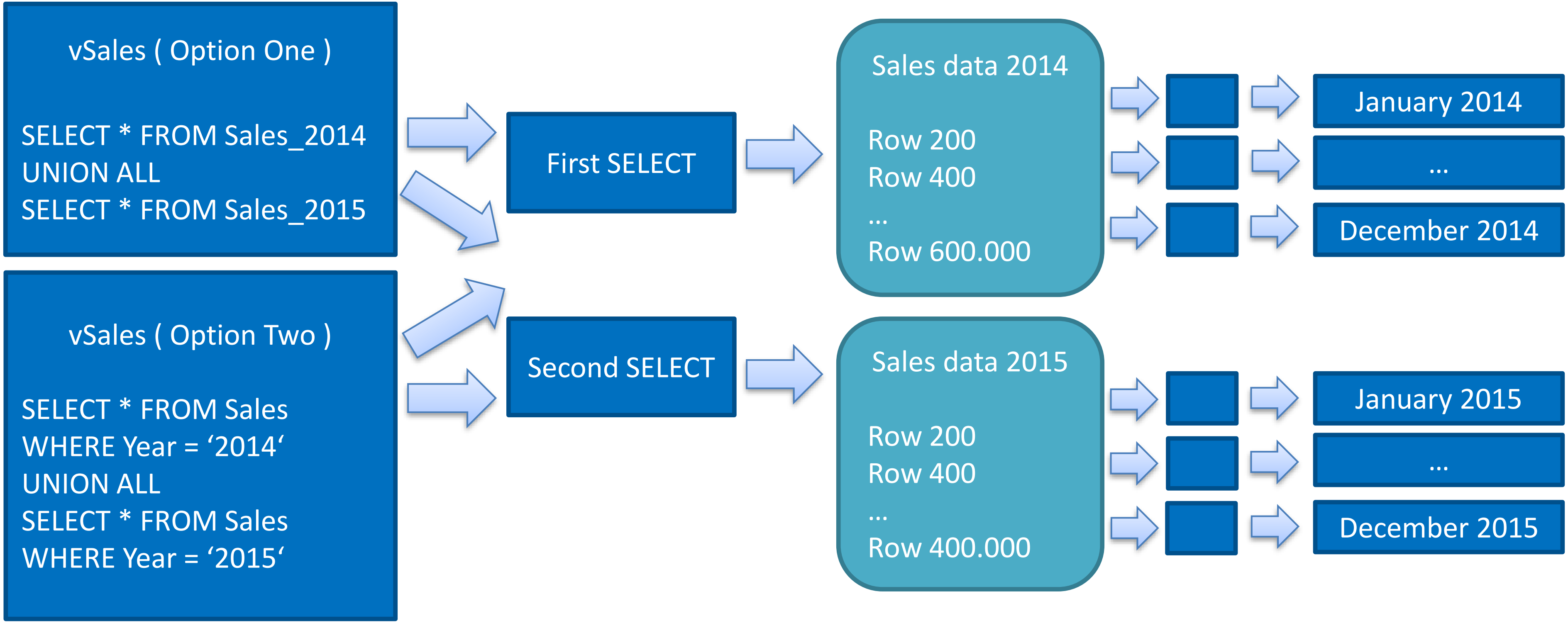
2014

Row 200
Row 400
…
Row 600.000

2015

Row 200
Row 400
…
Row 400.000

# Partitioned views and Partitioning…



vSales ( Option One )

```
SELECT * FROM Sales_2014
UNION ALL
SELECT * FROM Sales_2015
```

vSales ( Option Two )

```
SELECT * FROM Sales
WHERE Year = ‘2014‘
UNION ALL
SELECT * FROM Sales
WHERE Year = ‘2015‘
```

First SELECT

Second SELECT

Sales data 2014

Row 200
Row 400
…
Row 600.000

Sales data 2015

Row 200
Row 400
…
Row 400.000

January 2014

…

December 2014

January 2015

…

December 2015

# Partitioned Views Vs. Partitioning

- Partitioned Views
  - Any Version
  - Partition elimination
  - Different indexes per "partition"
  - Replication friendly (all types)
  - "Easy" feature
  - High IO during "partitioning"
  - Can support multiple constraints on multiple columns

- Partitioning
  - Enterprise only
  - Partition elimination
  - Same indexes for all partitions
  - Replication friendly (transactional)
  - "Complex" feature
  - Designed specifically to ease the pain of moving in and out large volumes of data
  - Partitioned on a single column

# Data (Horizontal) partitioning

- You can partition ONLY by one column
- Large table/index can be split into multiple manageable portions
  - Horizontal partitioning takes groups of rows in a single table and allocates them in semi-independent physical sections
- SQL Server's horizontal partitioning is RANGE based

# Horizontal ranges are based on a partition key

- A single column in the table
  - Use a computed column if you must, but make sure it performs well as a criterion and works for joins
- Typically a date or integer value
- Consider:
  - A column you will join on
  - A column you can always use as a criterion

# Choosing a Partitioning Column

- Should reflect the best way to subdivide the target table and get a balanced distribution of data
- Used as a filter in most of the queries run against the table, otherwise you will not get the benefit of any partition elimination (accessing only the partitions needed as opposed to the whole table)
- Good candidates:
  - Date time columns: order date, inserted date, etc.
  - Countries: customer country

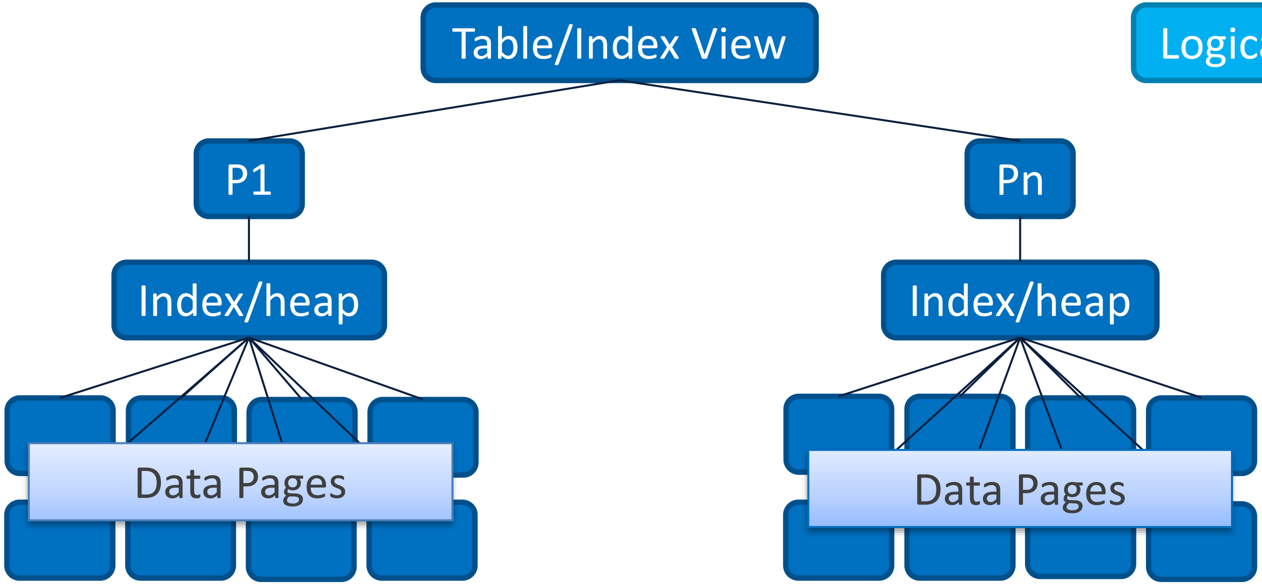
# Partition concepts (1)

- PARTITION FUNCTION
  - Used to specify partition boundary values
  - Two types (represents boundary data directions)
    - LEFT – myself and my left range values
    - RIGHT – myself and my right range values
- PARTITION SCHEME
  - Logical & Partitions physically aligned
  - Span over single or multiple file groups
  - Can specify each partition can go to a individual file group or all partitions can go into a single file group

# Partition concepts (2)

- SPLIT/MERGE
  - SPLIT
    - Introduces new boundary
    - Partition will be added to respective side (L/R)
  - MERGE
    - Deletes boundary
    - Partition will be merged to the respective side (L/R)
- SWITCH IN/OUT
  - Moving partition from partitioned table to other partitioned table called “in”
  - Moving partition from partitioned table to non partitioned table called “out”

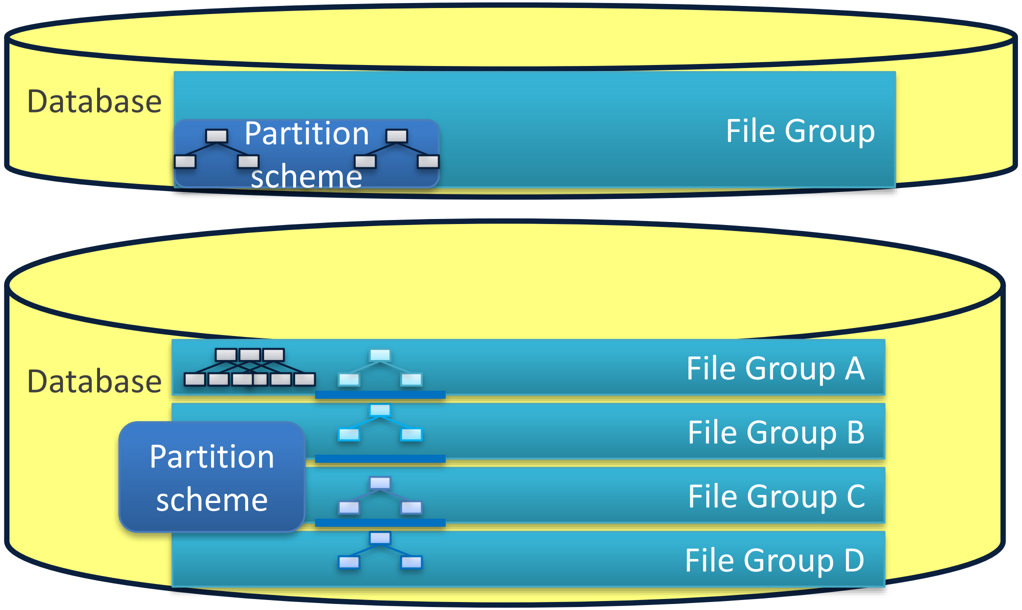
# Data Partitioning Architecture



Logical representation

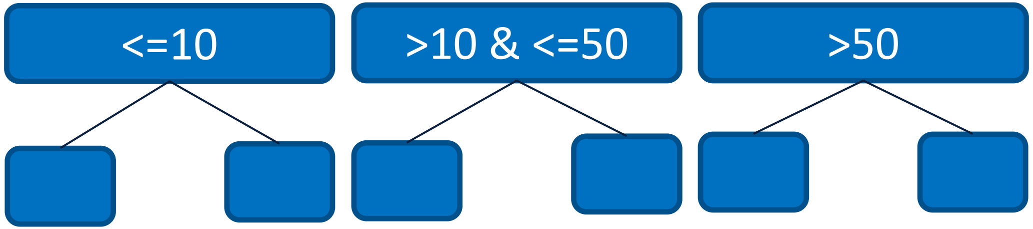
# Partition Physical Architecture

```
CREATE PARTITION SCHEME
MyPartitionScheme_ps
AS PARTITION MyPartitionFunctionName_pfn
ALL TO ([FG]) -- specifying single file group
-- TO ([FGA], [FGB]) -- specifying multiple file groups
```



# Partitioning Functions

```
CREATE PARTITION FUNCTION
MyPartitionFunctionName_pfn(int) AS
RANGE LEFT FOR VALUES(10,50)
```

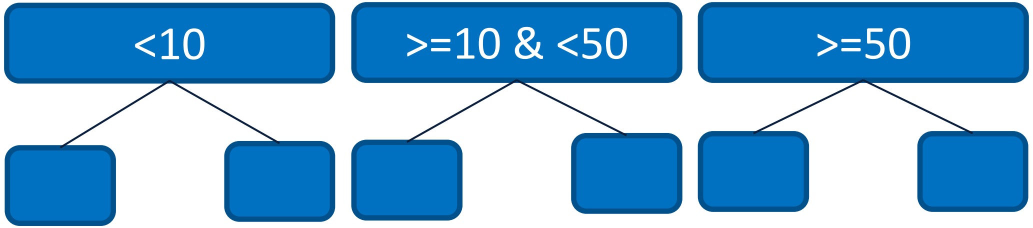


<=10

>10 & <=50

>50

```
CREATE PARTITION FUNCTION
MyPartitionFunctionName_pfn(int) AS
RANGE RIGHT FOR VALUES(10,50)
```

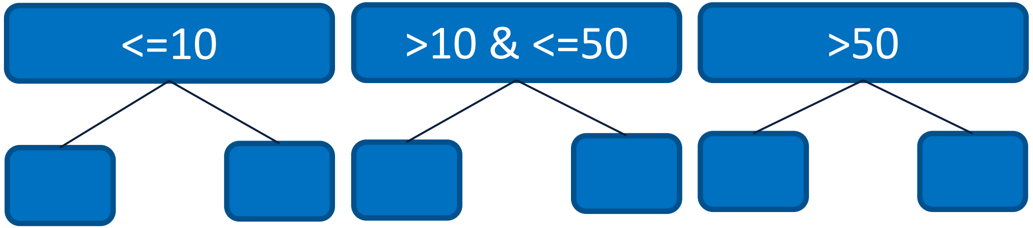


<10

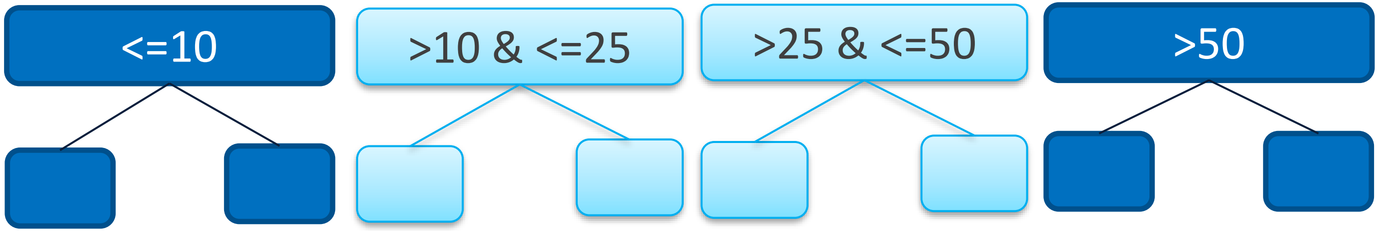
>=10 & <50

>=50

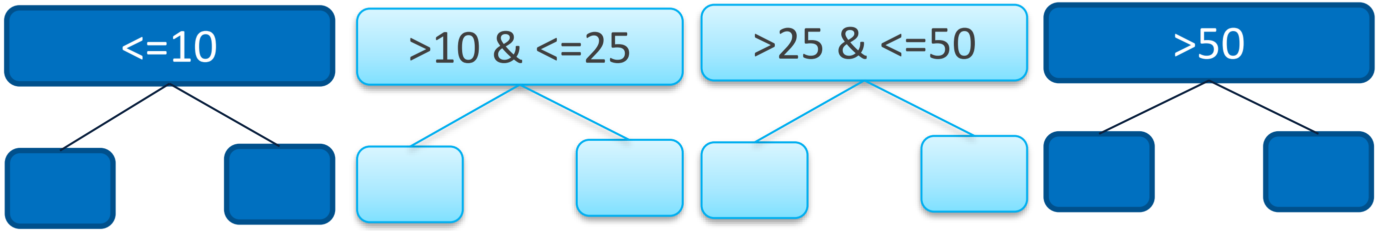
# SPLIT – LEFT boundary



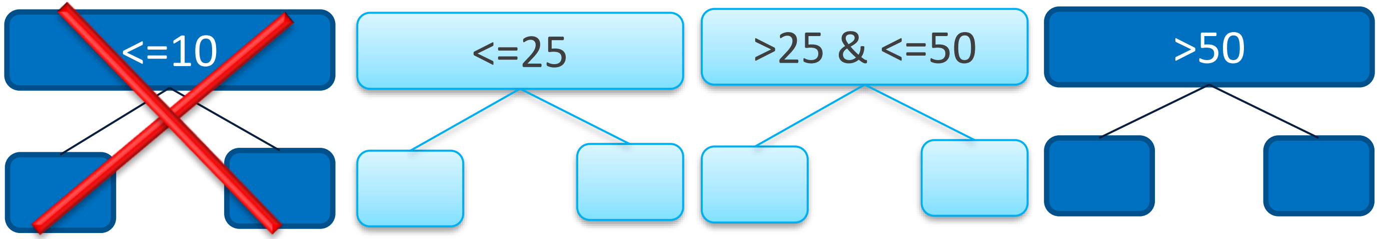
```
ALTER PARTITION SCHEME MyPartitionScheme_ps
NEXT USED [FG];  -- Specifying the group for new boundary
ALTER PARTITION FUNCTION MyPartitionFunction_pfn()
SPLIT RANGE(25)
```



# MERGE – LEFT boundary



```
ALTER PARTITION FUNCTION
MyPartitionFunctionName_pfn()
MERGE RANGE (10)
```



# Indexes can be created on the partition scheme…or not

## Aligned Indexes

- Located on your partitioning scheme (or an identical partitioning scheme)
- Must contain the partitioning key
- If the partitioning key is not specified, it will be added for you. Note: this affects  your primary key for the table!
- Indexes are aligned by default unless it is otherwise specified at creation time
- Perform better for aggregations and when partition elimination can be used

## Non-aligned indexes

- Physically located elsewhere- either non partitioned or on a non-identical partitioning scheme
- May perform better with single-record lookup
- Allow unique indexes (because they do not have to contain the partitioning key)
- However, the presence of these preclude partition-switching!

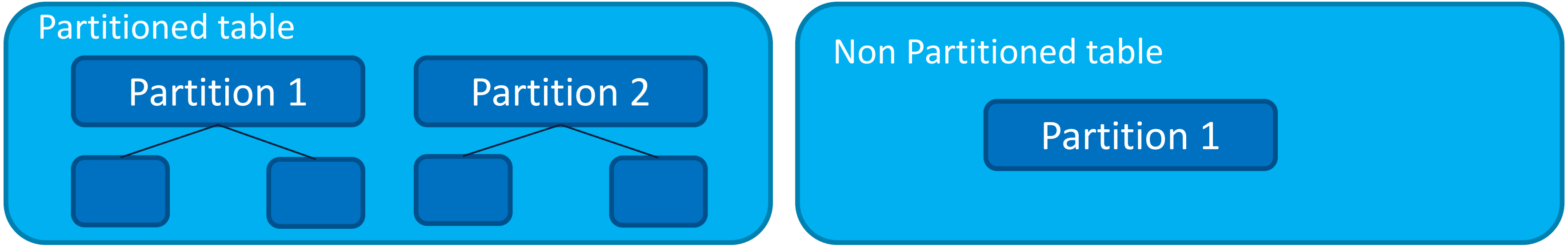
# Index rebuilds and compression

- Individual partitions *cannot* be rebuilt online
  - The entirety of a partitioned index *can* be rebuilt online
  - Individual partitions can be compressed
- For fact tables with archive data, older partitions can be rebuilt once with compression
  - Their file groups can then be made read-only

# Switching

- Requires all indexes to be aligned
- Compatible with filtered indexes
- Data may be switched in or out only within the same file group
- Is a metadata-only operation requiring a schema modification lock
  - This can be blocked by DML operations, which require a schema stability lock
- Is an exceptionally fast way to load or remove a large amount of data from a table!
- https://msdn.microsoft.com/en-us/library/ms191160.aspx

# SWITCH OUT



Partitioned table

Partition 1

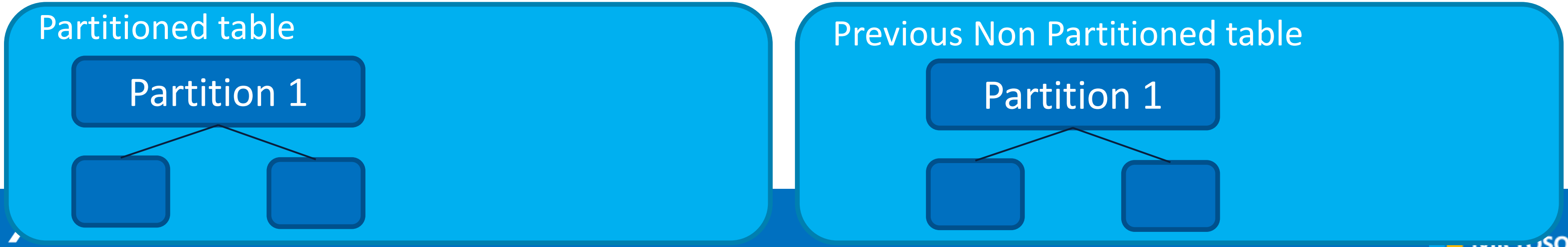
Partition 2

Non Partitioned table

Partition 1

```
ALTER TABLE PartitionedTableName
SWITH PARTITION
2 TO NonPartitionedTableName
-- <Partition number> TO DestionationPartitionedTableName<destination Partition number>
```

**SWITCH IN/OUT says “Hey, its your partition by updating metadata.”**



Partitioned table

Partition 1

Previous Non Partitioned table

Partition 1

# SQL Server Partitioning Myths (1)

- Myth 1: Partitioning is a "Scale-Out" solution
- Myth 2: Partitions must be created on different file groups
- Myth 3: To partition a non-partitioned table you will need to drop and recreate it
- Myth 4: Partitioning an existing table is a strictly offline operation

# SQL Server Partitioning Myths (2)

- Myth 5: SWITCH-ing partitions OUT or IN in only a few seconds
- Myth 6: Altering a partition function is a metadata only operation
- Myth 7: Partitioned tables improve query performance
- Myth 8: Partitioned tables ease maintenance of VLDBs and Very Large Tables

# Demo

- Let us start writing some code…
  - Simulating partitioning on Standard Edition
  - Partitioning demo
    - Concepts
    - Advanced demo

# Partitioning without Enterprise Edition

- SQL Server 7.0 introduced partitioning through partitioned views
- In the world of partitioning, the number of rows queried is reduced according the granularity of the partitioning scheme
- There are, of course, some downsides
  - Queries that cross the partition rules will take longer
  - True partitioning requires SQL Server Enterprise

# Contacts

- E-mail: josipsaban@gmail.com
- LinkedIn: www.linkedin.com/in/jsaban

Microsoft