EATING THE ELEPHANT

UNDERSTANDING AND IMPLEMENTING SQL SERVER PARTITIONING

Mike Fal - www.mikefal.net

Mike Fal

Working with SQL Server since MSSQL 7.

Experience with different industries.

No cool letters after my name. ;_;

Blog – www.mikefal.net

Twitter - @Mike_Fal

Session Goals

- Understand what partitioning is within SQL Server.
- Be able to select appropriate partitioning keys.
- Know how to create partitioning on a table or index.
- Plan and implement maintenance on partitions.

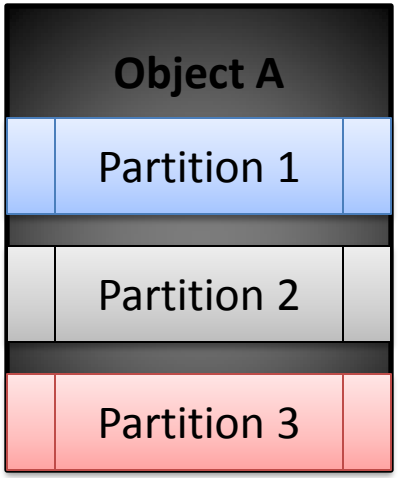
What is partitioning?

Start at the beginning

Definition

- Partitioning – physically segment a table or index into multiple parts

  SQL Server partitioning is horizontal partitioning



Pros and Cons

Benefits

- *Distribute an object across multiple filegroups.*
- *Manage and maintain an object by partition.*
- *Additional layer of engine management for querying and locking.*
- *Can compress individual partitions for space savings*

Drawbacks

- *Only available in Enterprise Edition.*
- *1000 partitions max (15,000 in SQL 2012)*
- *Additional maintenance overhead (more DBA work to do!)*
- *Difficult to implement on existing data, requires data movement.*

When do I partition?

“It depends!”

- Really big tables (several hundred GB)
- Index maintenance takes too long
- Concurrency/performance issues
- **NOT FOR EVERYONE!**

How do I partition?

Better portions

What are we doing?

- Creating boundaries in the data
- SQL Server partitioning is RANGE partitioning
- You can only partition on one field (partition key)
- For tables with clustered index, partition key must be part of the clustered index

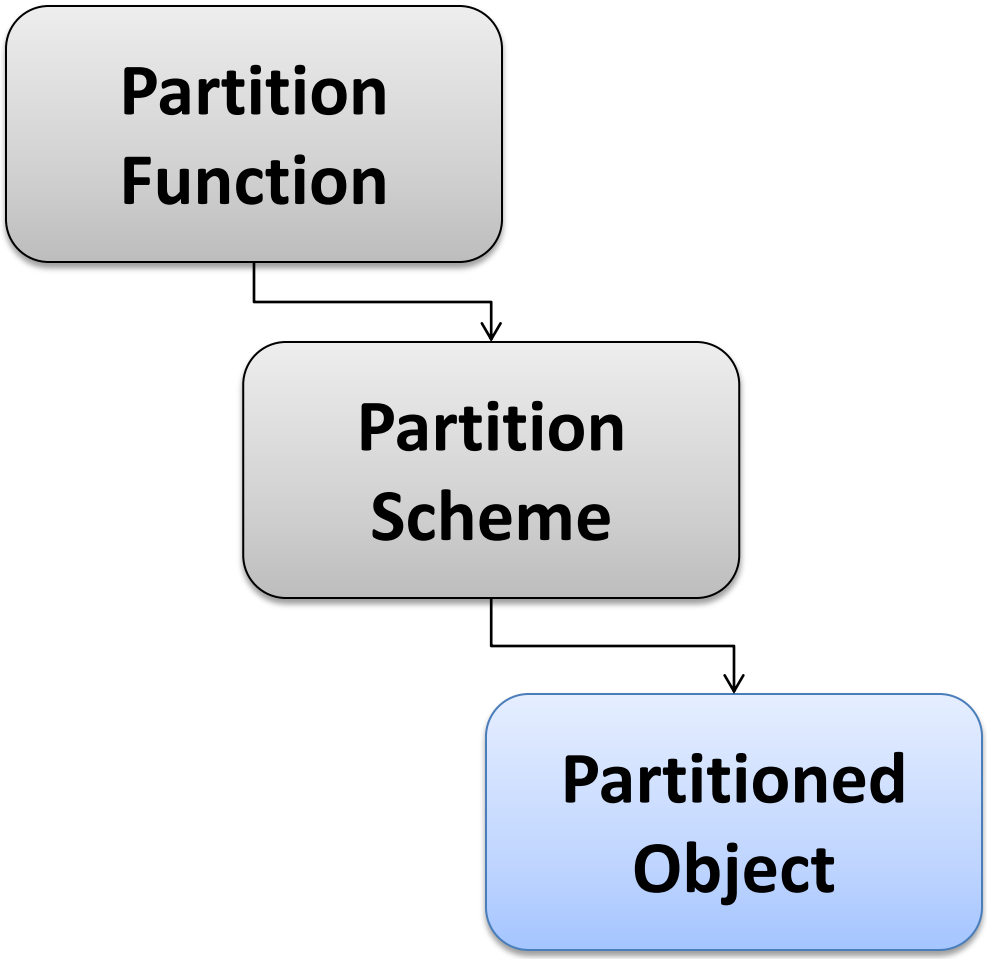
Boundaries

Example: Boundaries at 0, 10, 20, 30, and 40

min…**0**	1…**10**	11…**20**	21…**30**	31…**40**	41…max

- Both “ends” of the partitioning are open ended.
- Boundary inclusion depends on whether we declare LEFT or RIGHT for our range.

Components



Partition Function

CREATE PARTITION FUNCTION <<name>>(type)
AS RANGE FOR VALUES (…)

- Defines the rules of the partition
  - Boundaries (LEFT and RIGHT)
  - Data type
- Used by multiple schemes
- Not directly tied to any storage, tables or indexes (meta data).
- `sys.partition_functions`

LEFT versus RIGHT

- Defines where the boundary exists for the partition
  - LEFT: Where the partition "ends" (default)

min…**0**	1…**10**	11…**20**	21…**30**	31…**40**	41…max

  - RIGHT: Where the partition "starts"

min…-1	**0**…9	**10**…19	**20**…29	**30**…39	**40**…max

Partition Scheme

```
CREATE PARTITION SCHEME <<name>>
AS PARTITION <<partition function>>
TO (…)
```

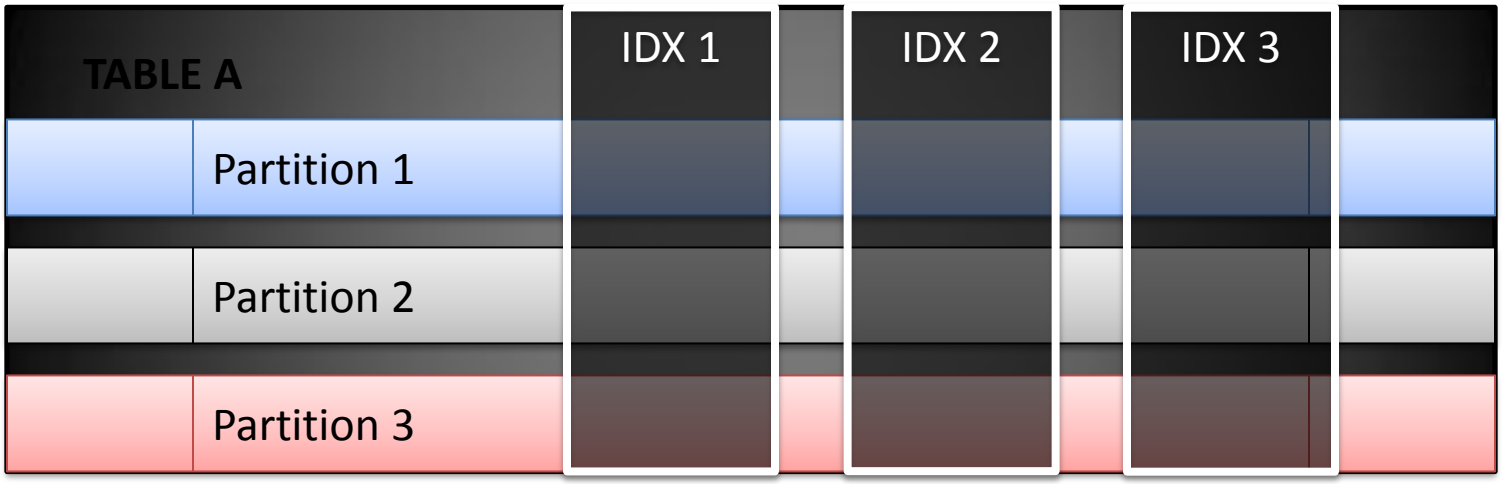
- Maps filegroups (physical storage) to partitions
- Must supply as many filegroups as partitions in function (exception: “ALL to”)
- Can supply an additional filegroup for NEXT USED
- Also meta data, no direct object associations
- `sys.partition_schemes`

Partitioned Object

- Table or Index
- Uses partition scheme for CREATED ON (instead of filegroup)
- Actual objects in database
- `sys.partitions`

Things to keep in mind

- Partition based on access patterns.
- Align partitions (multiple partitioned objects)



Demo

Orders

Field Name	Data Type	Description
Order ID	INT	Order ID, Primary Key
Order Date	Datetime	Order date
Order Amount	Money	Total Order amount
Order Details	CHAR(7000)	Additional order information

How to I maintain my Partitions?

Keeping it clean

Fragmentation

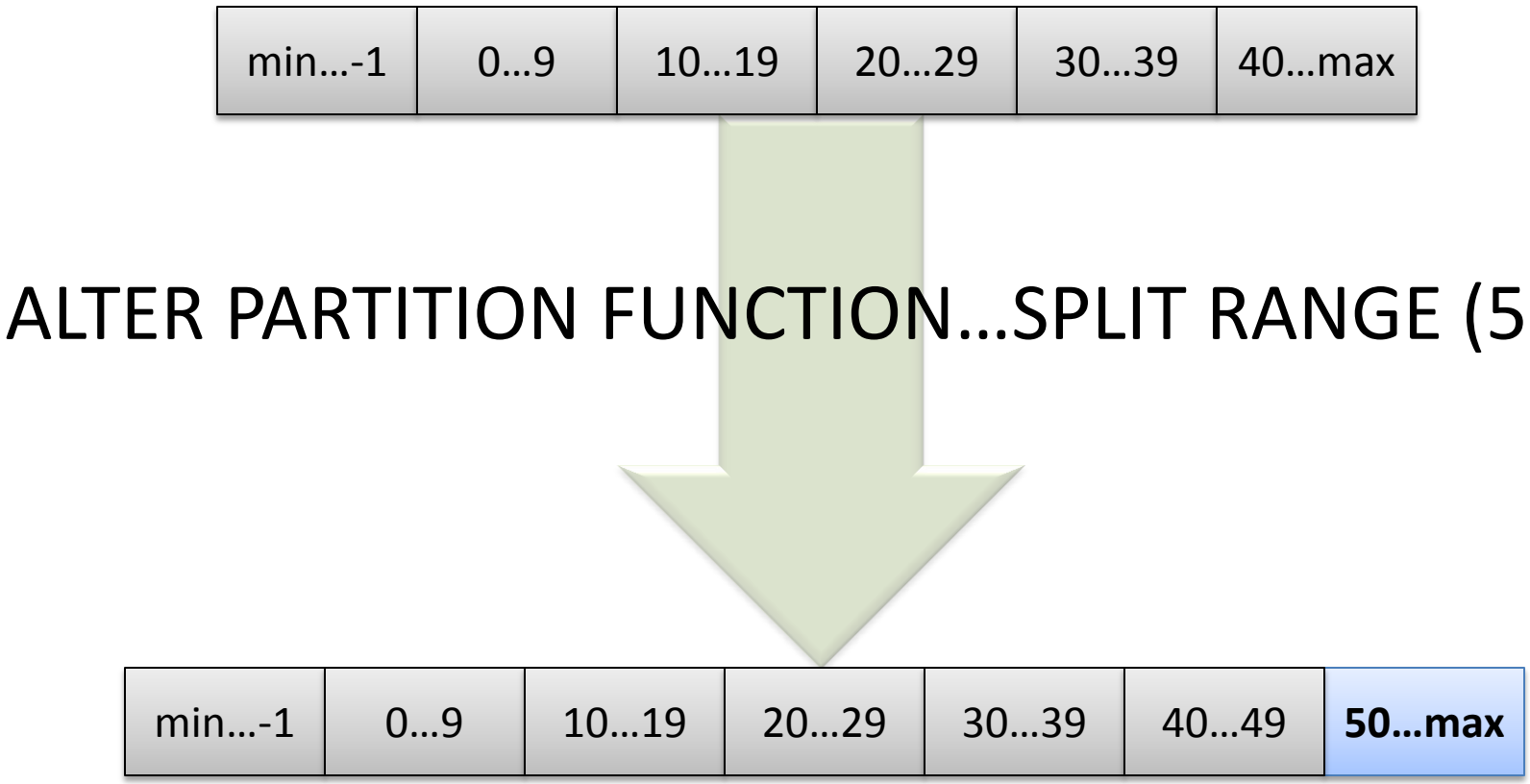
- `sys.dm_db_index_physical_stats`
  - Shows fragmentation by partition
- ALTER INDEX…REBUILD/REORGANIZE
  - PARTITION=n/ALL
- Rebuild/Reorganize only the partitions you need!

Adding New Partitions

`ALTER PARTITION FUNCTION….SPLIT RANGE`

- Adds new boundary
- Uses previously existing range direction (LEFT/RIGHT)
- New partition will use NEXT USED filegroup (even within existing partitions!)

SPLIT

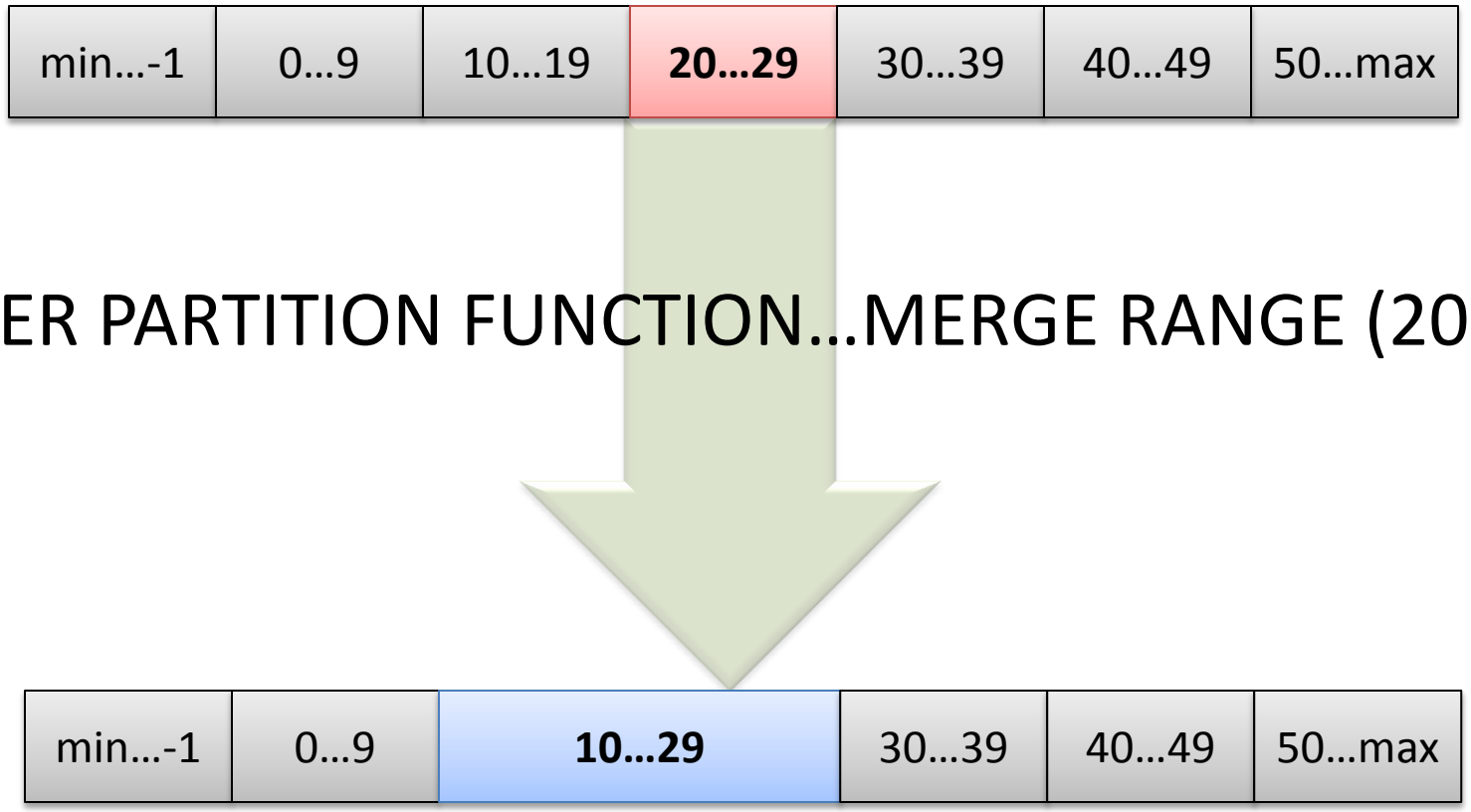

min…-1
0…9
10…19
20…29
30…39
40…max
ALTER PARTITION FUNCTION…SPLIT RANGE (50)
min…-1
0…9
10…19
20…29
30…39
40…49
50…max

Removing Partitions

`ALTER PARTITION FUNCTION…MERGE RANGE`

- Removes existing boundary
- Merges the two neighboring partitions into one
- Data is moved from removed partition's filegroup to new partition's filegroup

MERGE


min…-1
0…9
10…19
20…29
30…39
40…49
50…max
ALTER PARTITION FUNCTION…MERGE RANGE (20)
min…-1
0…9
10…29
30…39
40…49
50…max

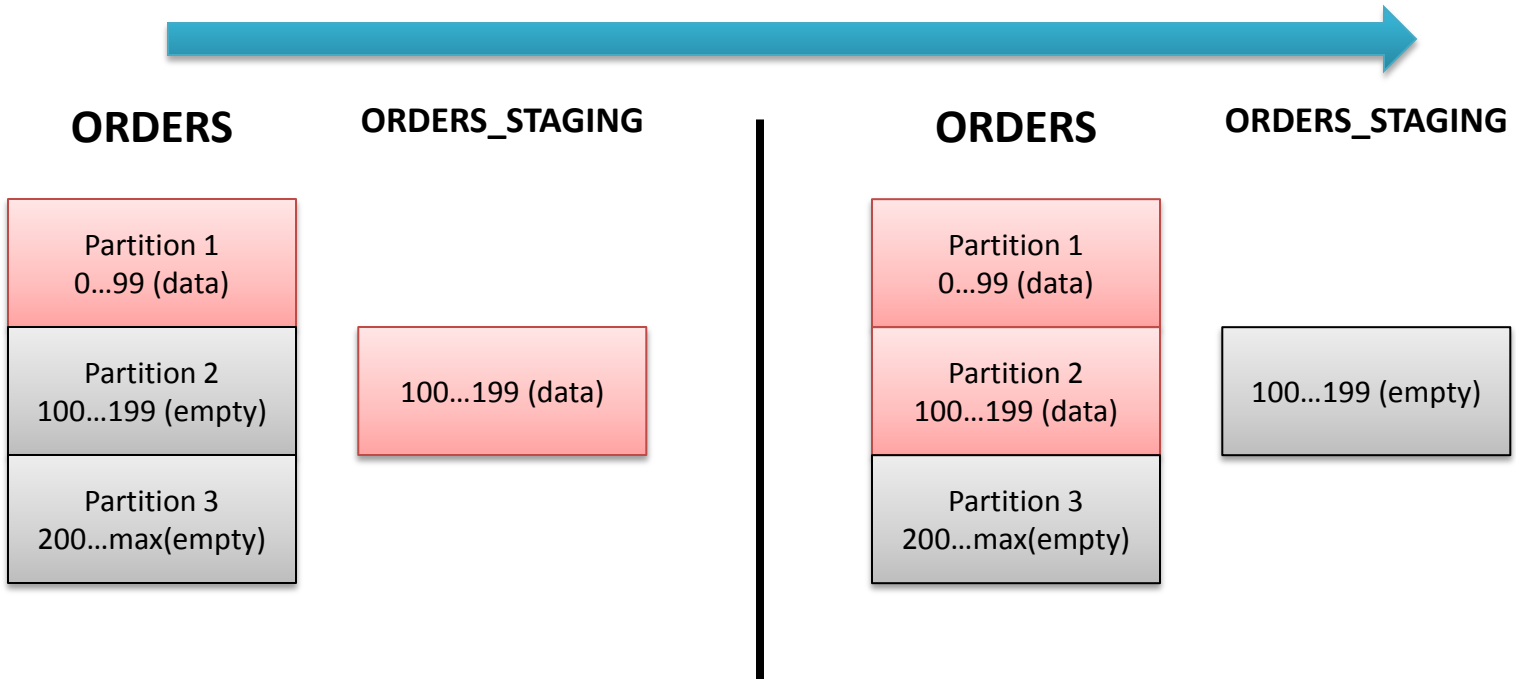
Moving Partitions

```
ALTER TABLE <source table name>
   SWITCH [PARTITION n]
   TO <destination table name> [PARTITION n]
```

- Several requirements
  - Source and destination must be structurally identical
  - Data must be ensured to belong in the destination(CHECK constraint)
  - All indexes must be aligned
  - Source and destination must be in the same filegroup

SWITCH

```
ALTER TABLE [ORDERS_STAGING]
SWITCH [ORDERS] PARTITION 2
```



Demo 2

Finish Line

- Understand what partitioning is within SQL Server.
- Be able to select appropriate partitioning keys.
- Know how to create partitioning on a table or index.
- Plan and implement maintenance on partitions.

Questions

HUH?

www.mikefal.net

@Mike_Fal