Microsoft

SQL Server Table Partitioning

CS340 | SBASSE | LUMS

Table Partitioning

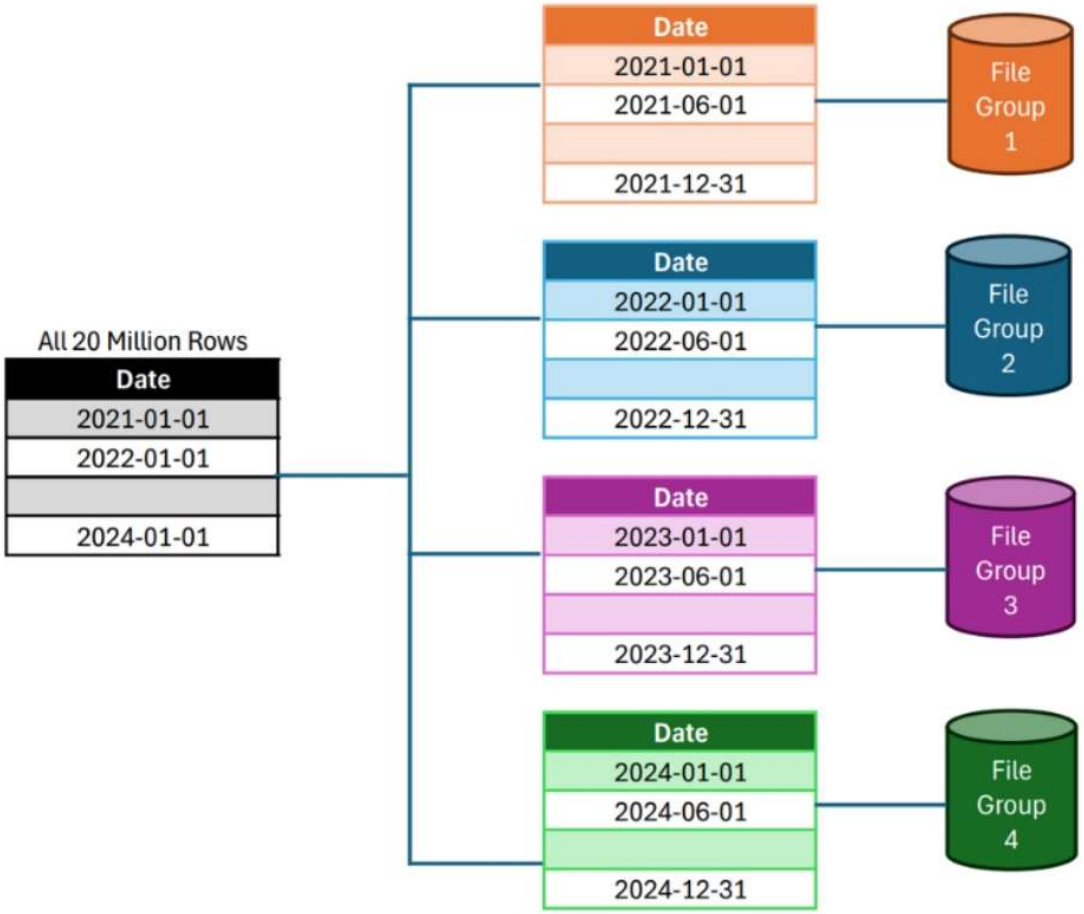
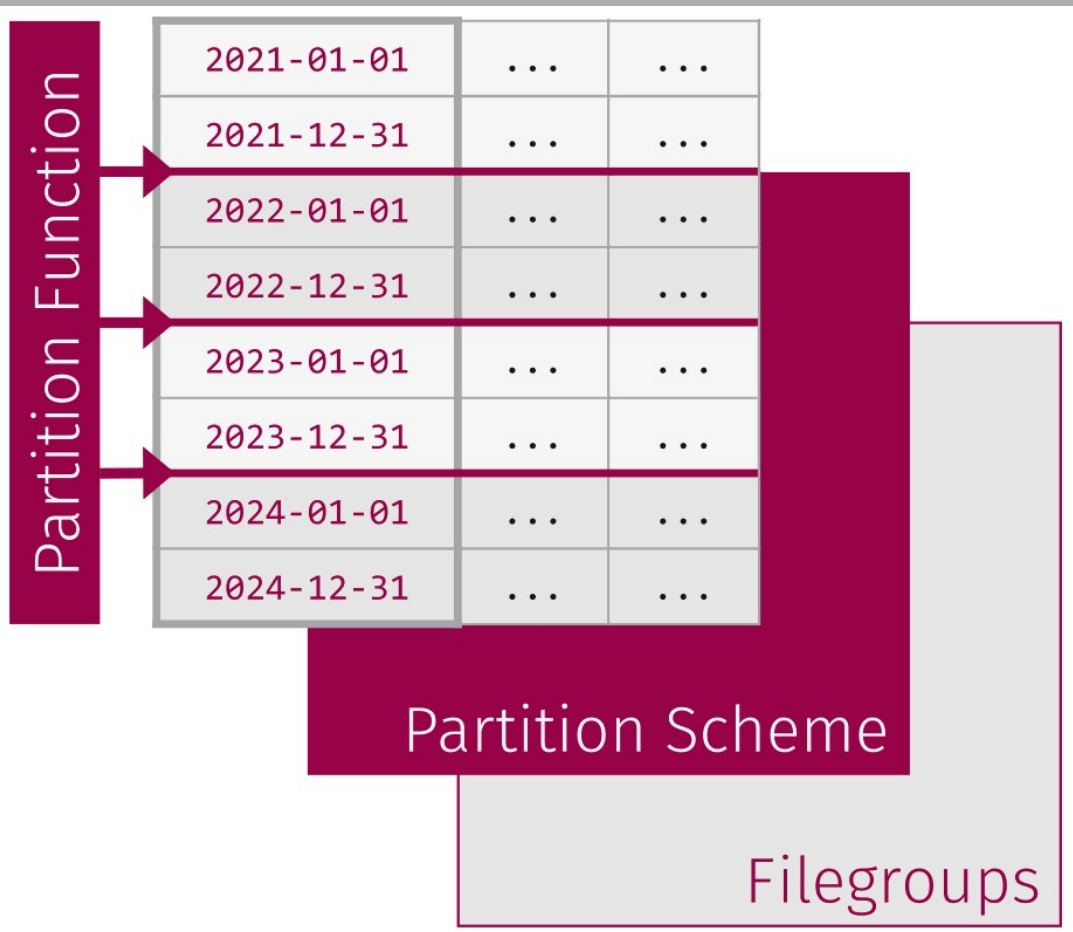
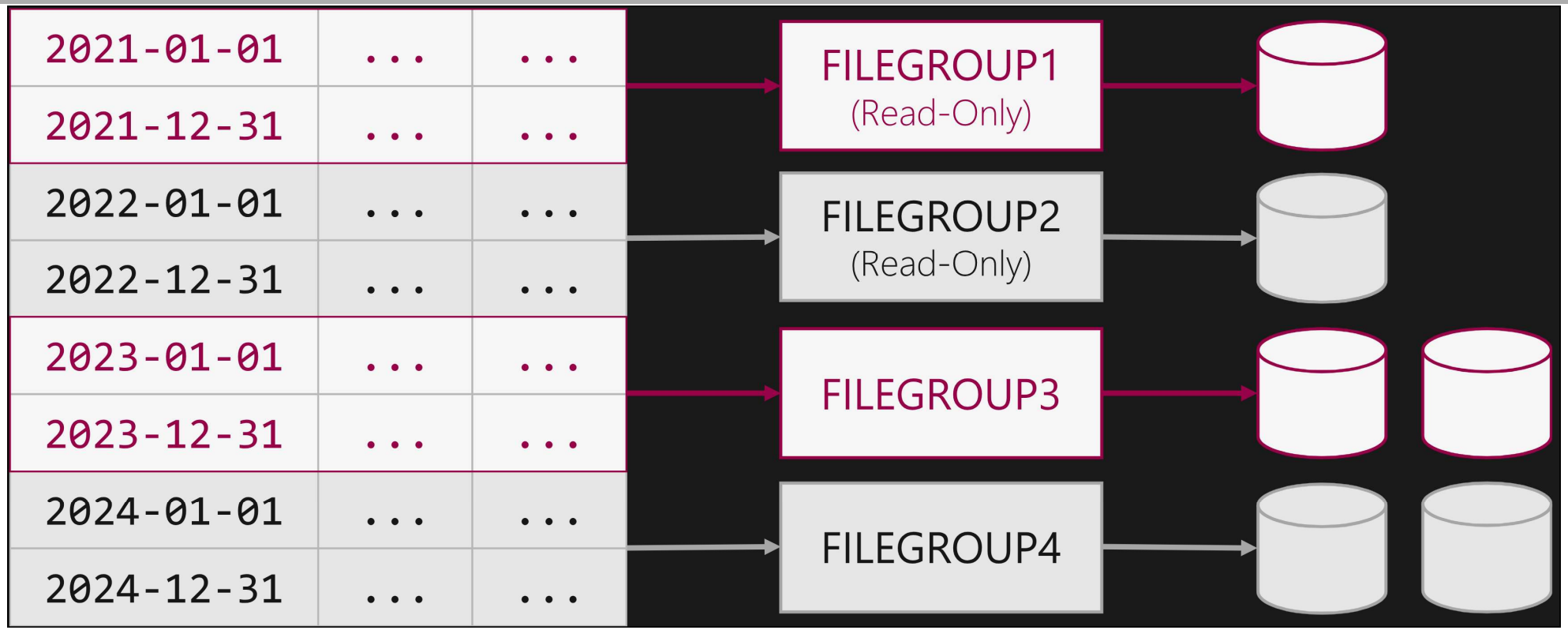
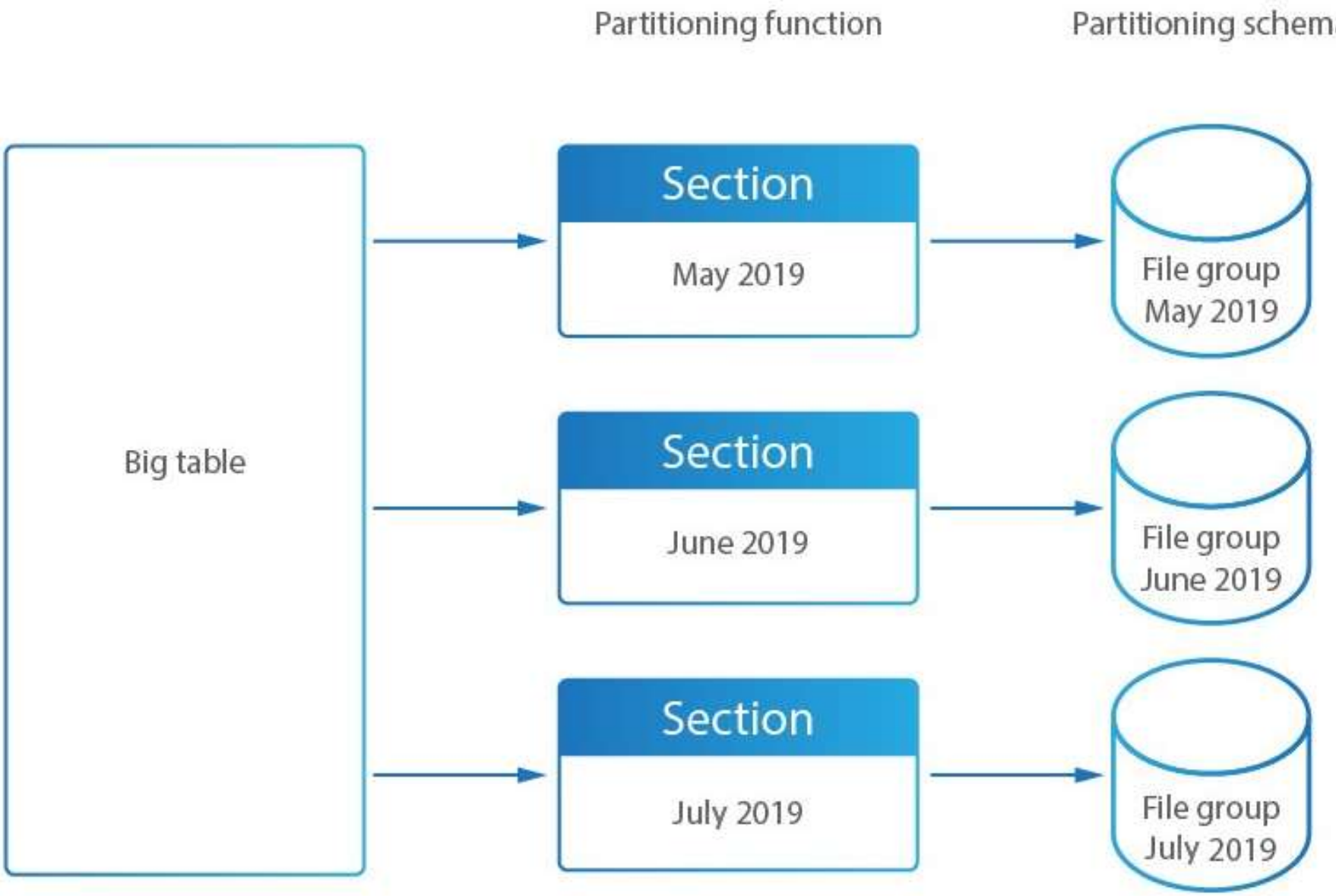


Table Partitioning



Partitioning Scheme

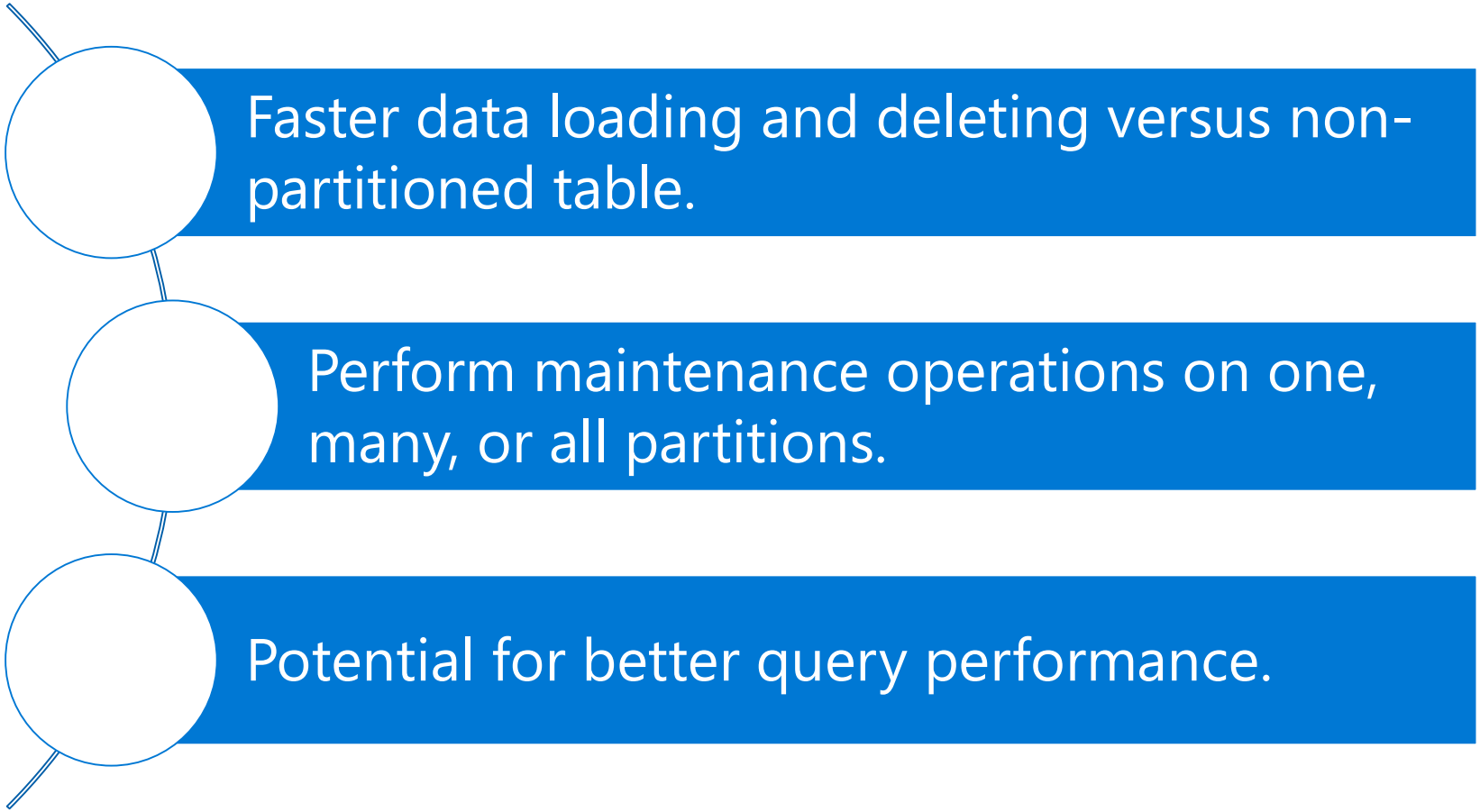



Partitioning function
Partitioning schema
Big table
Section
May 2019
Section
June 2019
Section
July 2019
File group
May 2019
File group
June 2019
File group
July 2019

What is Table Partitioning

- Technique in SQL Server that allows you to split a large table into smaller, more manageable pieces (partitions) based on the values in a specific column
- Each partition behaves like a separate table internally
- Partitions are invisible to the user: Treated as one entity when queried.

Why use partitioning?



When do you choose partitioning?

- Large table causes maintenance or performance concerns
- When to consideration:
  - Removing a large volume of data is slow or causes blocking
  - Loading a large volume of data is slow or causes blocking
  - Backup / maintenance exceeds maintenance window
- Partitioning for performance
  - Parallel operations on large tables
  - Prunes data easier, reduces I/O

Is it in all versions/editions?

- Prior to SQL Server 2016 SP1
  - No, Enterprise Edition only.
- As of SQL Server 2016 SP1+, yes.
- Up to 15,000 partitions
  - Prior to SQL Server 2012, limited to 1,000.

Microsoft

Table partitioning basics

Partitioning Building Blocks

1. Partitioning column
2. Partitioning function
3. Partitioning scheme

1. Partitioning Column

- Must be a [computed] column on the table
- A valid data type
  - (not ntext, text, image, xml, varchar(max), nvarchar(max), or varbinary(max))
- If clustered, column must be PK or the clustered index.

- If partition column is not PK:
  - Avoid NULLable columns in partition colum
  - NULL partition column will reside in leftmost partition
- Partition column to divide table
  - Should be relatively balanced
- A common filtering column
  - Enables partition elimination in query processing

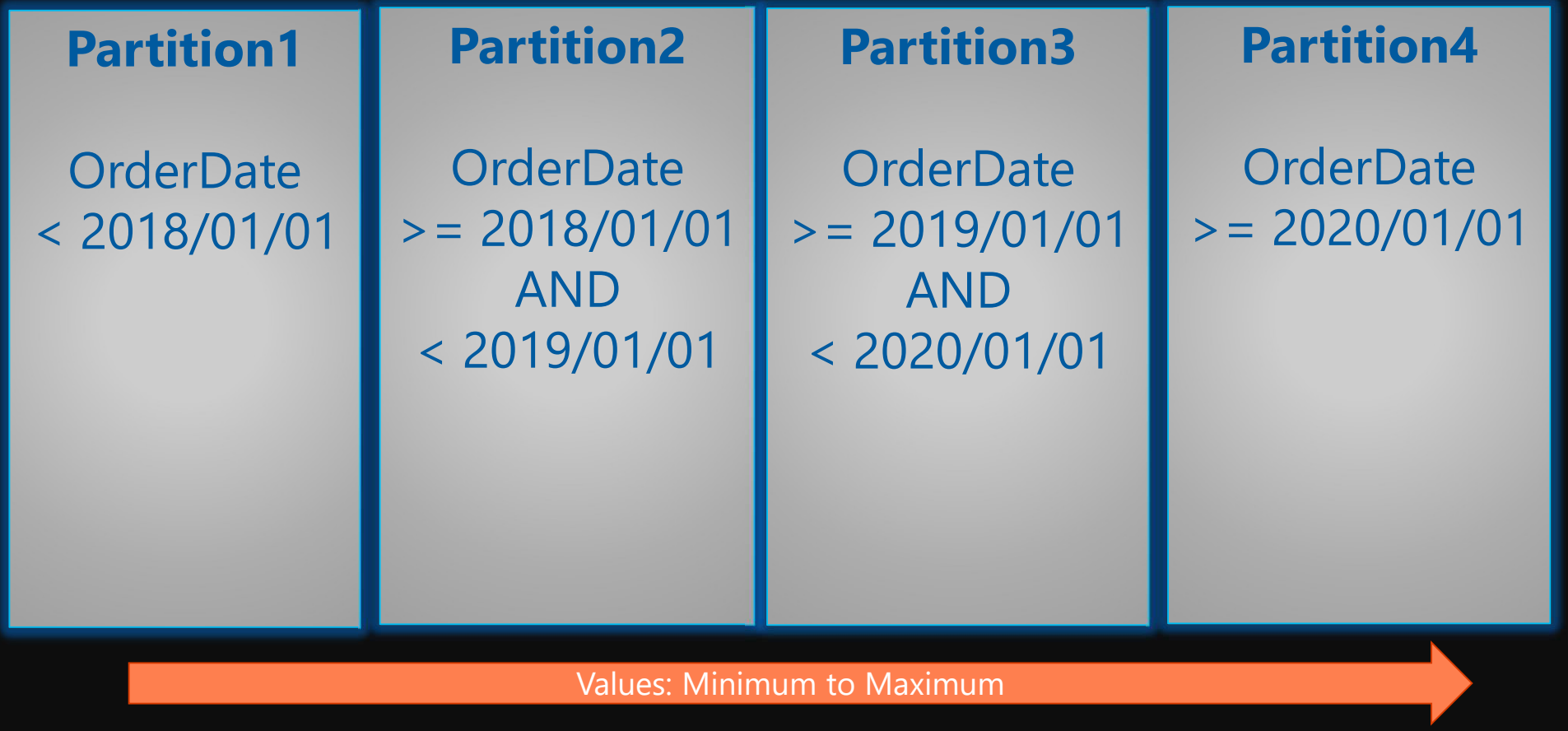
2. Partition Function

- Defines how the rows will be mapped to the partitions
- Based on the partitioning column

```
CREATE PARTITION FUNCTION OrderDatePF (datetime)
AS RANGE RIGHT FOR VALUES
('2018/01/01', '2019/01/01', '2020/01/01');
```

RANGE RIGHT means that each boundary value belongs to the partition to its right.

Function – OrderDatePF



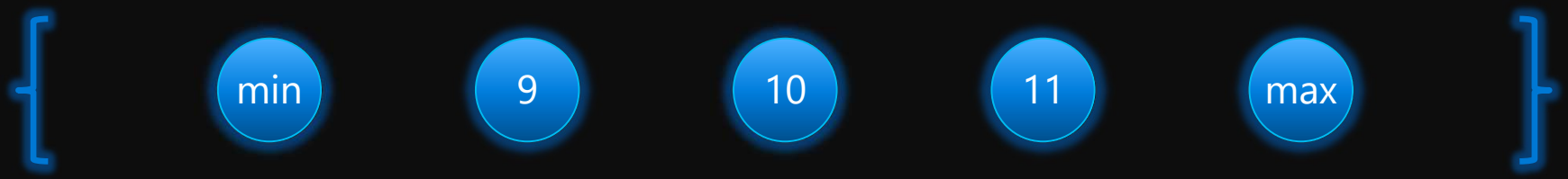
Range Types – RIGHT or LEFT

- As part of the function, you need to determine if your partition will range right or range left.
- Range right - "begins at" - lower boundary
- Range left - "ends at" - upper boundary

```
CREATE PARTITION FUNCTION OrderDatePF (datetime)
AS RANGE RIGHT FOR VALUES
('2018/01/01', '2019/01/01', '2020/01/01');
```

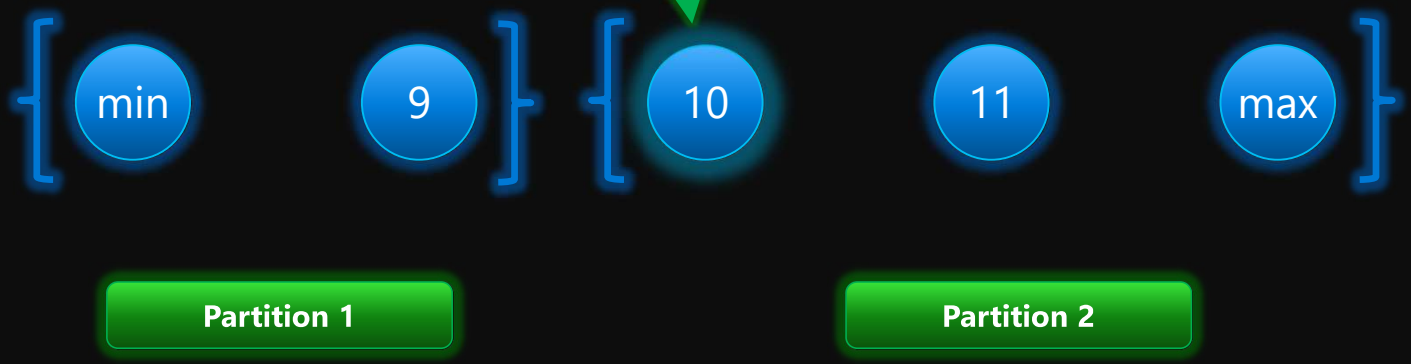
Note: RANGE RIGHT would require the full timestamp to keep date values of the same year in the same partition. Example: '2019/01/01 23:59:59.997'

Range Type Example – Unpartitioned Table



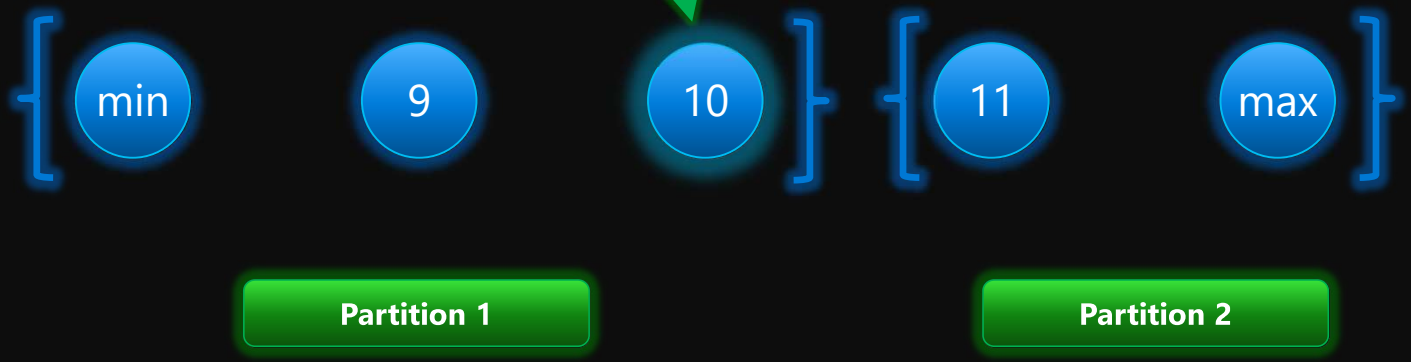
RANGE RIGHT

```
CREATE PARTITION FUNCTION PF1_Right (int)
AS RANGE RIGHT FOR VALUES (10);
```



RANGE LEFT

```
CREATE PARTITION FUNCTION PF1_Left (int)
AS RANGE LEFT FOR VALUES (10);
```



Microsoft

Paritioning Scheme

Partitioning Scheme

A **partitioning scheme** defines **how data is distributed across partitions** based on a **partition function**.

- The partition function, as defined earlier, maps rows to partitions based on the values in a specified column (often a date or ID).
- Then partitioning scheme maps those partitions to filegroups.

Benefits of Parititioning:

- **Improved Query Performance**: Queries can target specific partitions.
- **Efficient Maintenance**: You can switch, truncate, or rebuild partitions individually.
- **Better Storage Management**: Distribute data across multiple filegroups/disks.

Partitioning Scheme to Map Each Partition to different filegroup

```
CREATE PARTITION SCHEME YearPS
AS PARTITION OrderDatePF
TO (FG1, FG2, FG3, FG4);


CREATE TABLE Orders
   (OrderId INT, OrderDate DATETIME, Col3,etc...)
ON YearPS(OrderDate);
```

Partitioning Scheme: Example

Imagine a table storing sales data for multiple years. You can partition it by year:

```
CREATE PARTITION FUNCTION pfYear(INT)
AS
RANGE LEFT FOR VALUES (2019,2020,2021,2022);
```

The above maps each year to a different filegroup and creates a table using this scheme. Queries for a specific year will only scan the relevant partition.

Partitioning Scheme