Azure Data Warehouse Architecture

Azure
Synapse
Analytics

Introduction

MPP or Massive Parallel Processing

Storage & Data Distribution (Hash, Round-robin, Replicate)

Data types and Table types (Columstore, Heap, Clustered B-tree index)

Partitioning and Distribution key

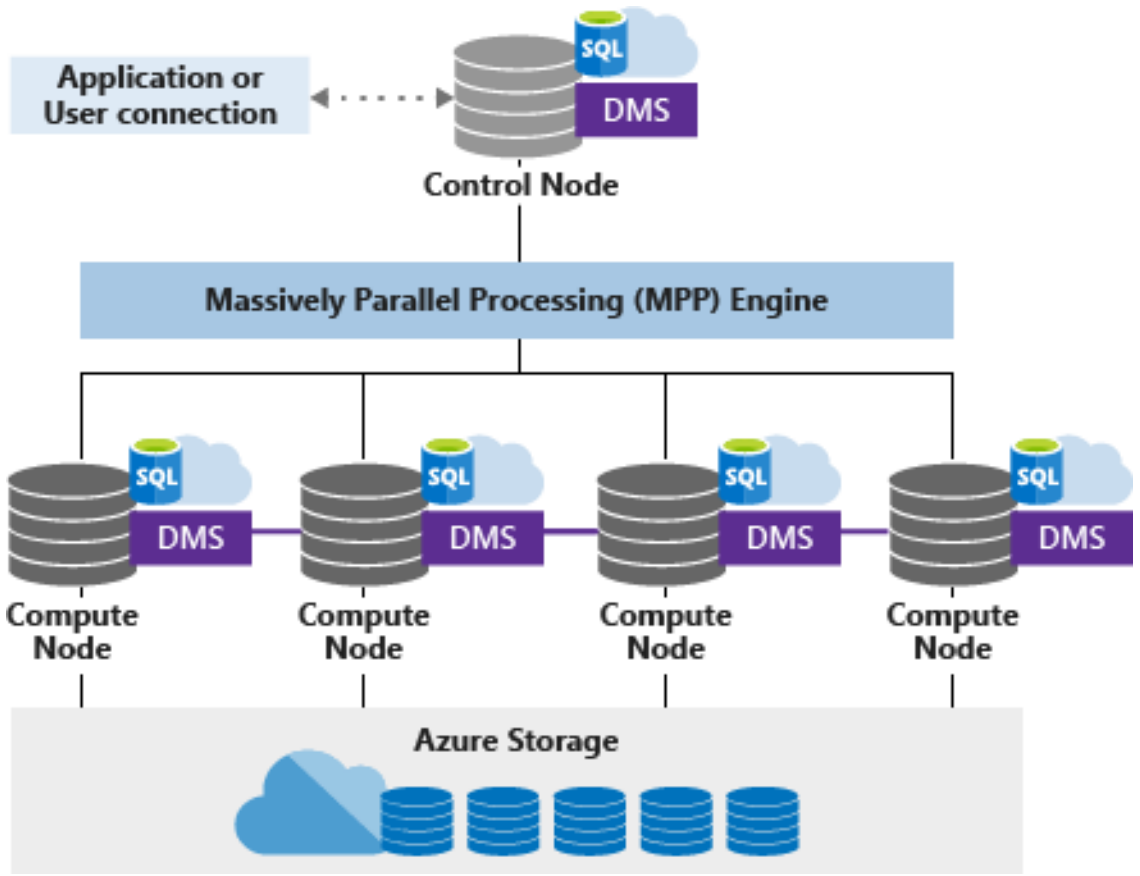
Applications in Dimensional modeling

Demo – Table Analysis before Migration to Cloud

Azure Synapse Analytics

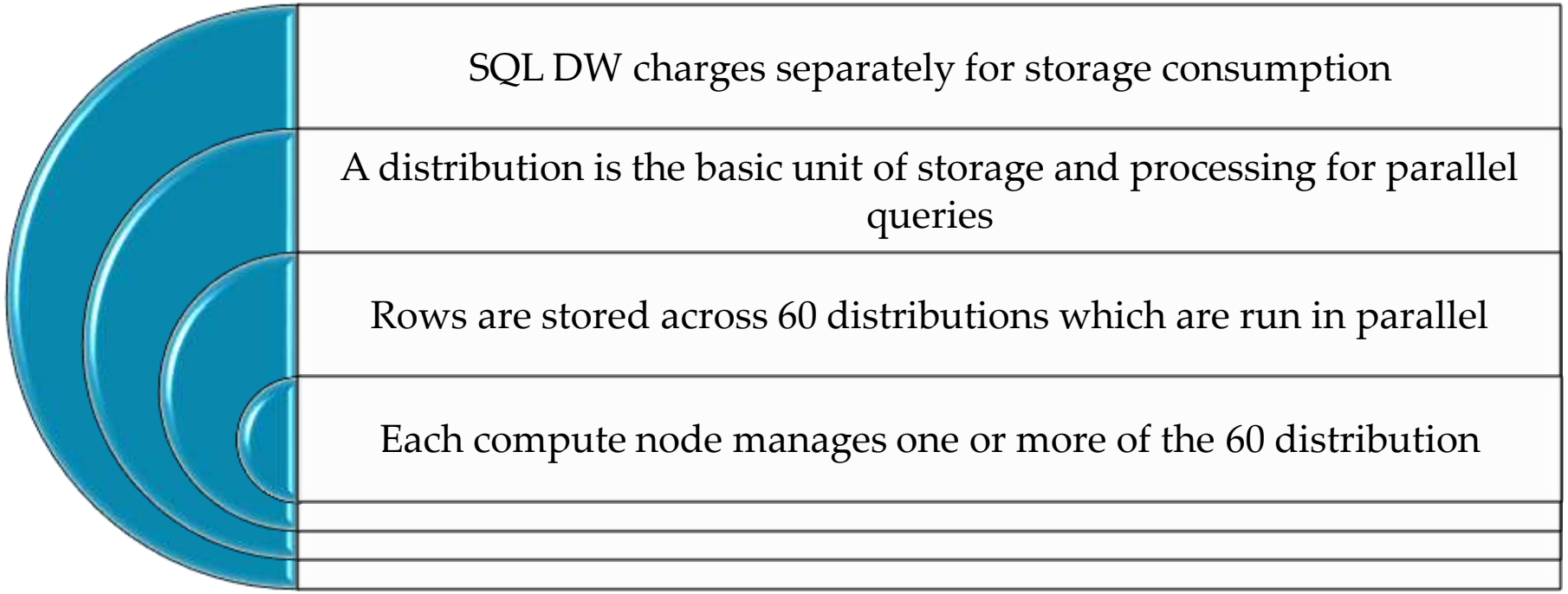
Azure Synapse MPP Architecture

DWU	Loading 3 Tables	Ran Report
100	15	20
500	3	4

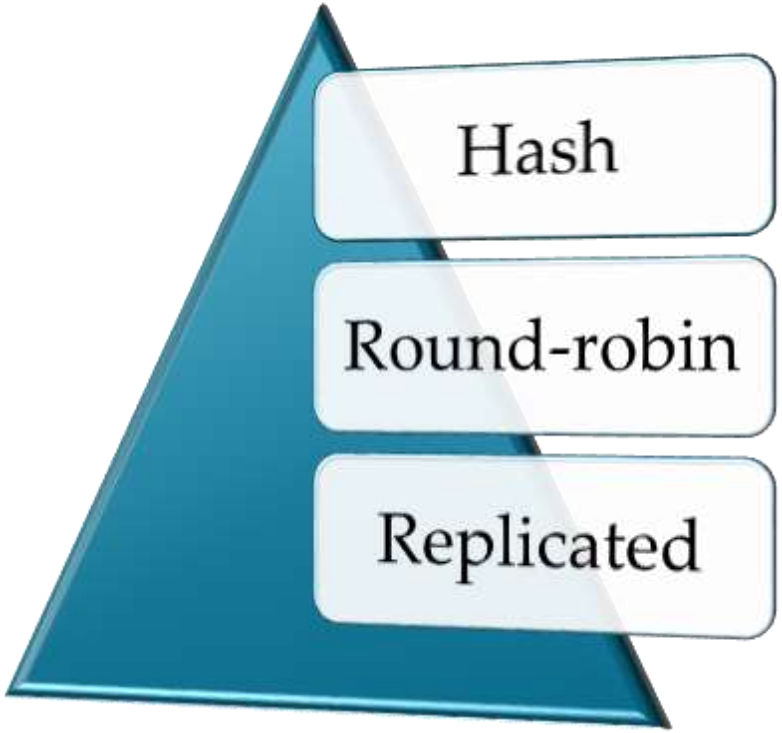


Source: Microsoft

Azure Storage and Distribution



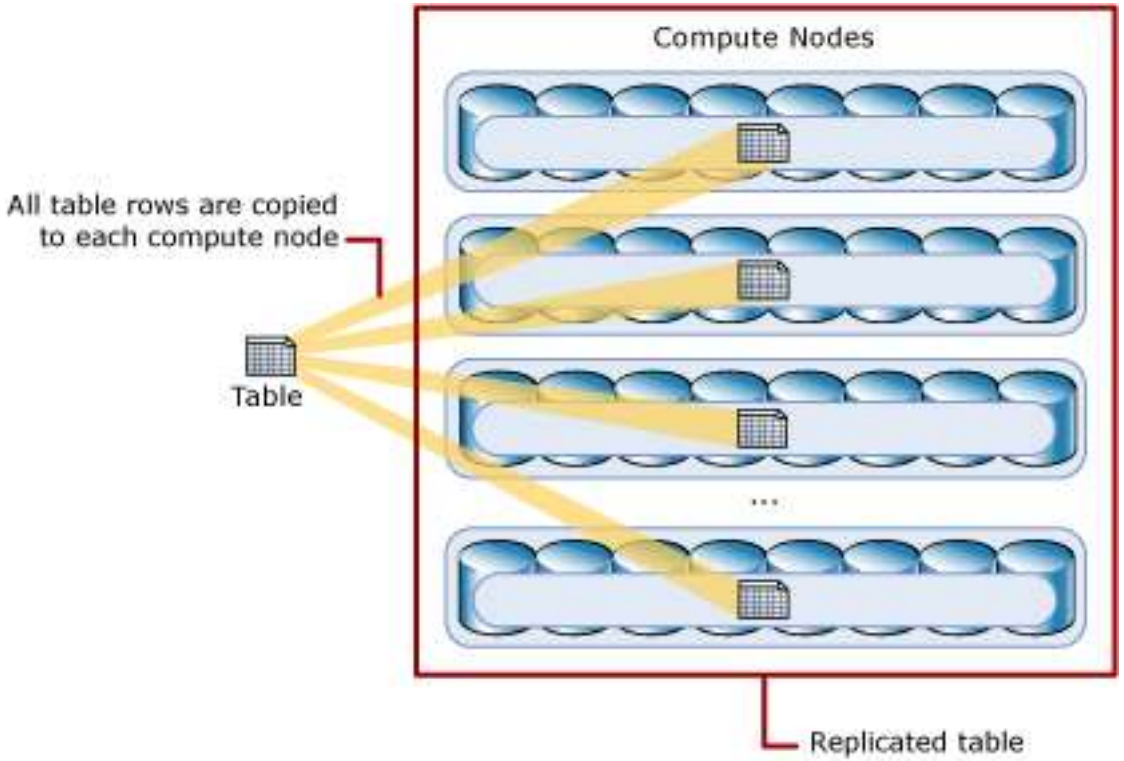
Sharding Patterns



Replicated Tables

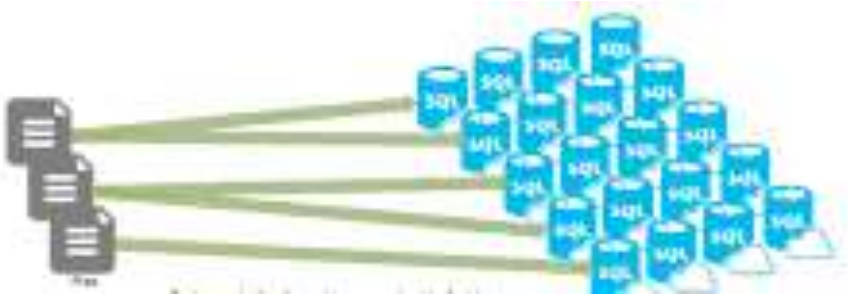
- Caches a full copy on each compute node.
- Used for small tables

```
CREATE TABLE [dbo].[BusinessHierarchies](
    [BookId] [nvarchar](250)  ,
    [Division] [nvarchar](100) ,
    [Cluster] [nvarchar](100) ,
    [Desk] [nvarchar](100) ,
    [Book] [nvarchar](100) ,
    [Volcker] [nvarchar](100) ,
    [Region] [nvarchar](100)
)
WITH
(
    CLUSTERED COLUMNSTORE INDEX
,    DISTRIBUTION = REPLICATE
)
;
```



Source: Microsoft

Round Robin tables



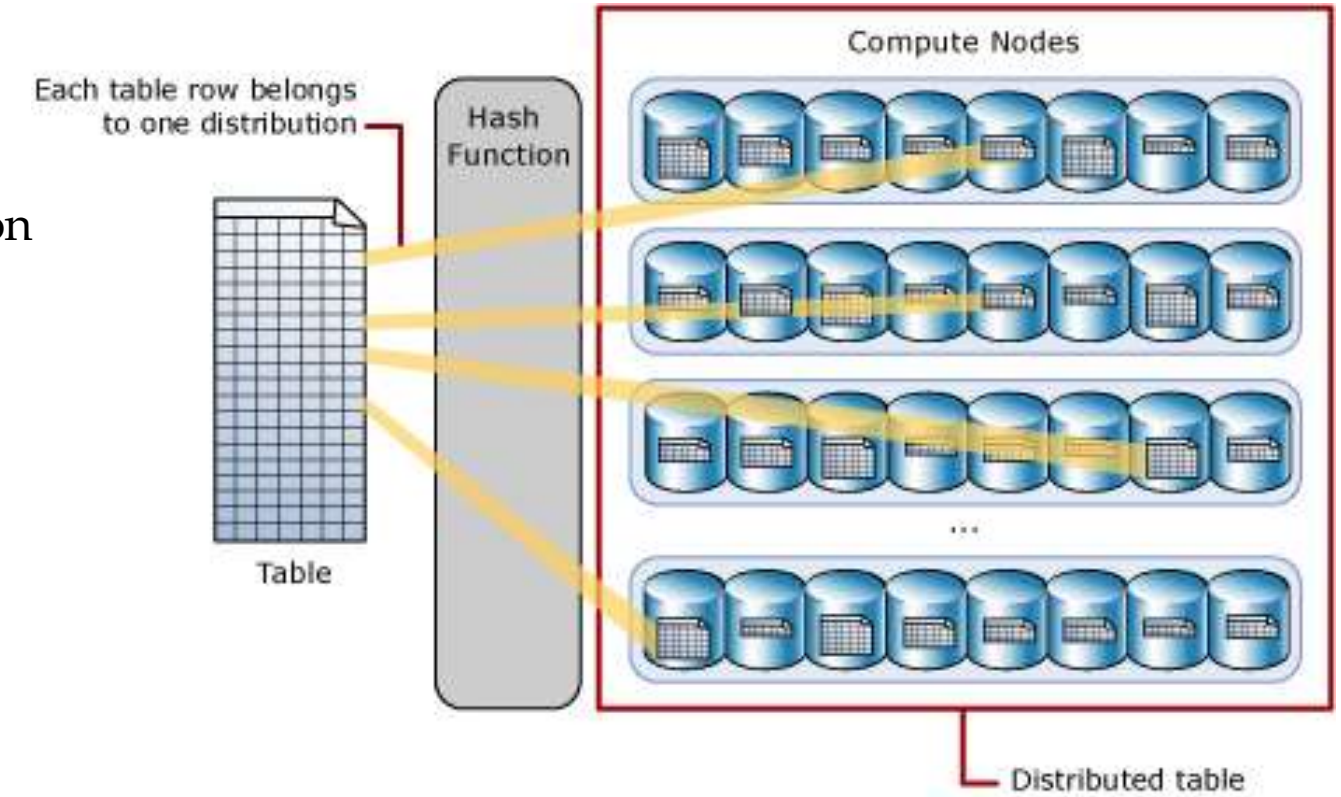
```
CREATE TABLE [dbo].[Dates](
    [Date] [datetime2](3) ,
    [DateKey] [decimal](38, 0) ,
    ..
    ..
    [WeekDay] [nvarchar](100) ,
    [Day Of Month] [decimal](38, 0)
)

WITH
(
    CLUSTERED COLUMNSTORE INDEX
,    DISTRIBUTION = ROUND_ROBIN
)
;
```

- Generally use to load staging tables
- Distribute data evenly across the table without additional optimization
- Joins are slow, because it requires to reshuffle data
- Default distribution type

Source: Microsoft

Hash Distribution Tables

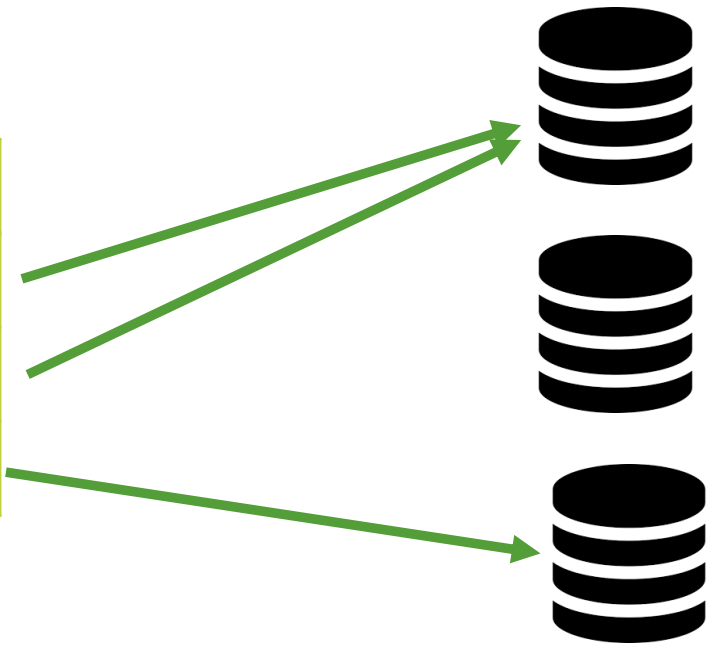
- Highest performance for large tables
- Each row belong to one particular distribution
- It is used mostly for larger tables



Source: Microsoft

Hash Distribution Tables

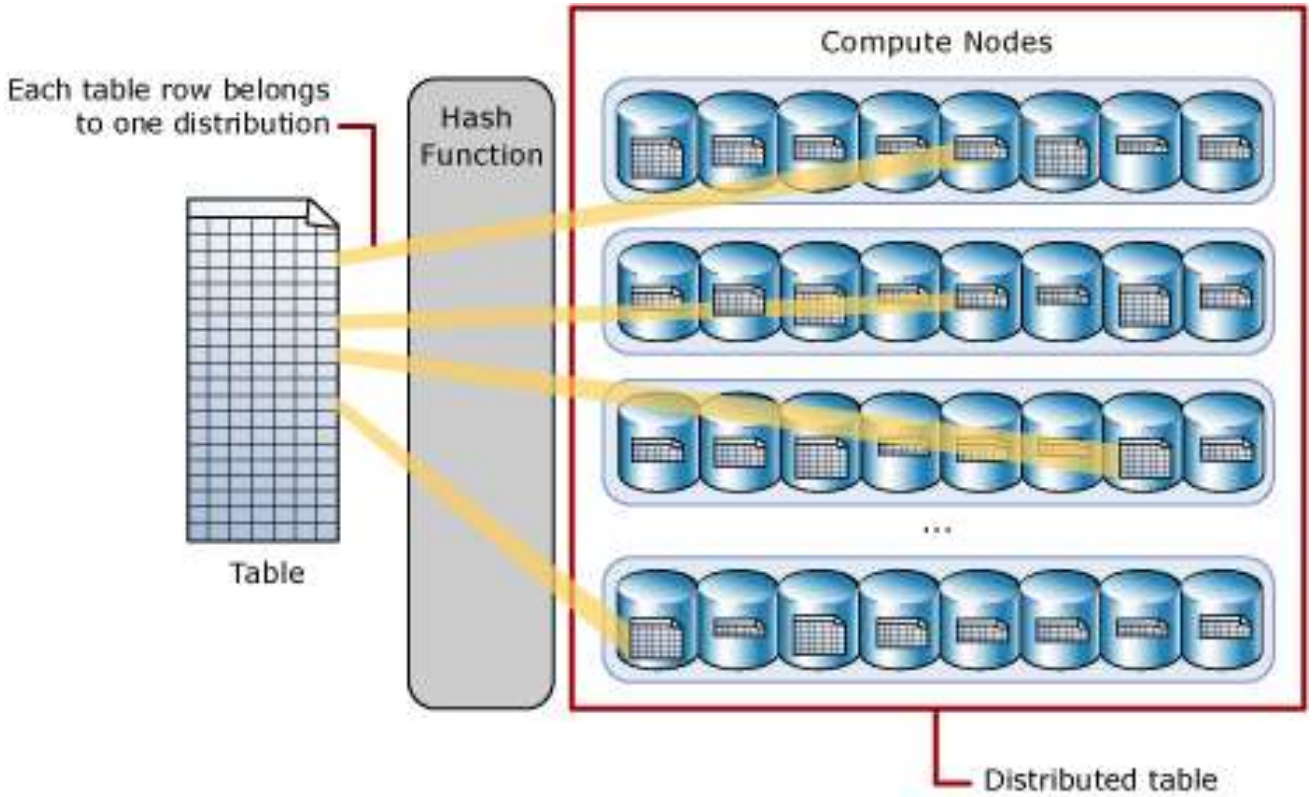
Record	Product	Store
1	Soccer	New York
2	Soccer	Los Angeles
3	Football	Phoenix



Hash Distribution Tables

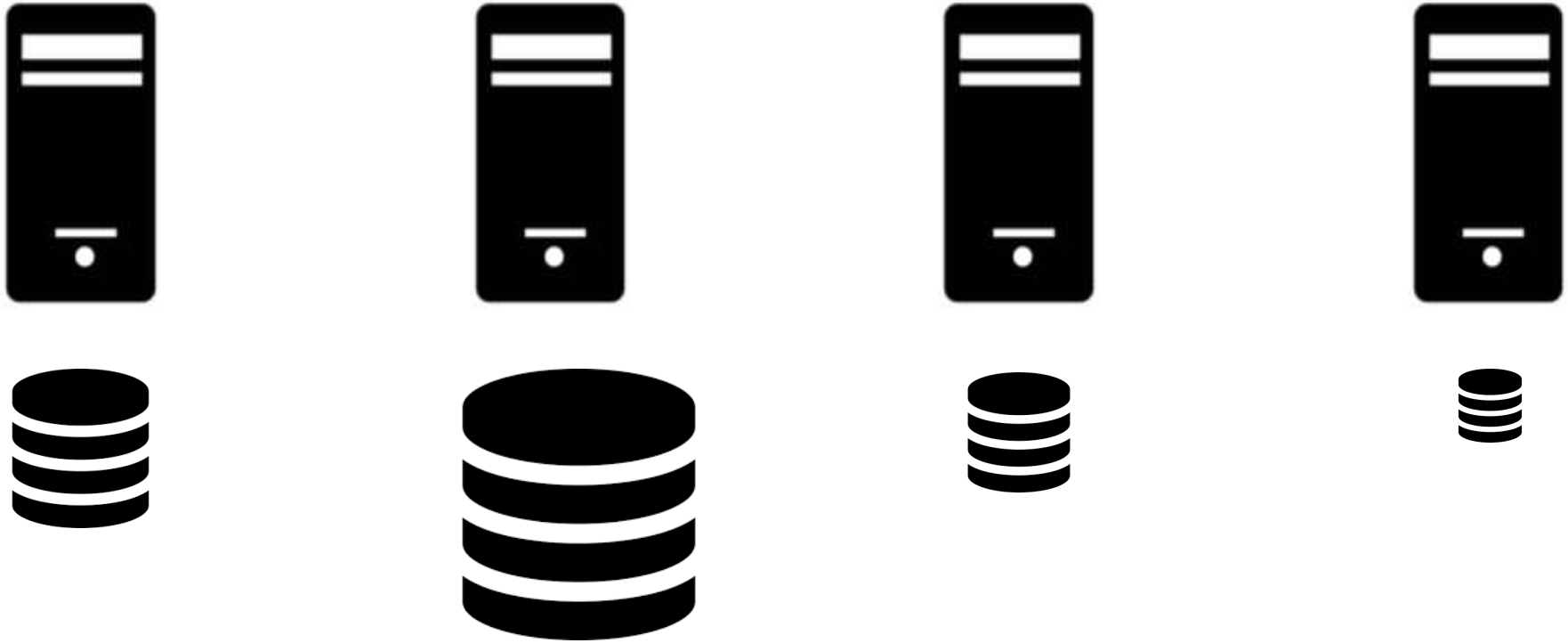
- Highest performance for large tables
- Each row belong to one particular distribution
- It is used mostly for larger tables

```
CREATE TABLE [dbo].[EquityTimeSeriesData](
   [Date] [varchar](30) ,
   [BookId] [decimal](38, 0) ,
   [P&L] [decimal](31, 7) ,
   [VaRLower] [decimal](31, 7)
)
WITH
(
   CLUSTERED COLUMNSTORE INDEX
,   DISTRIBUTION = HASH([P&L])
)
;
```

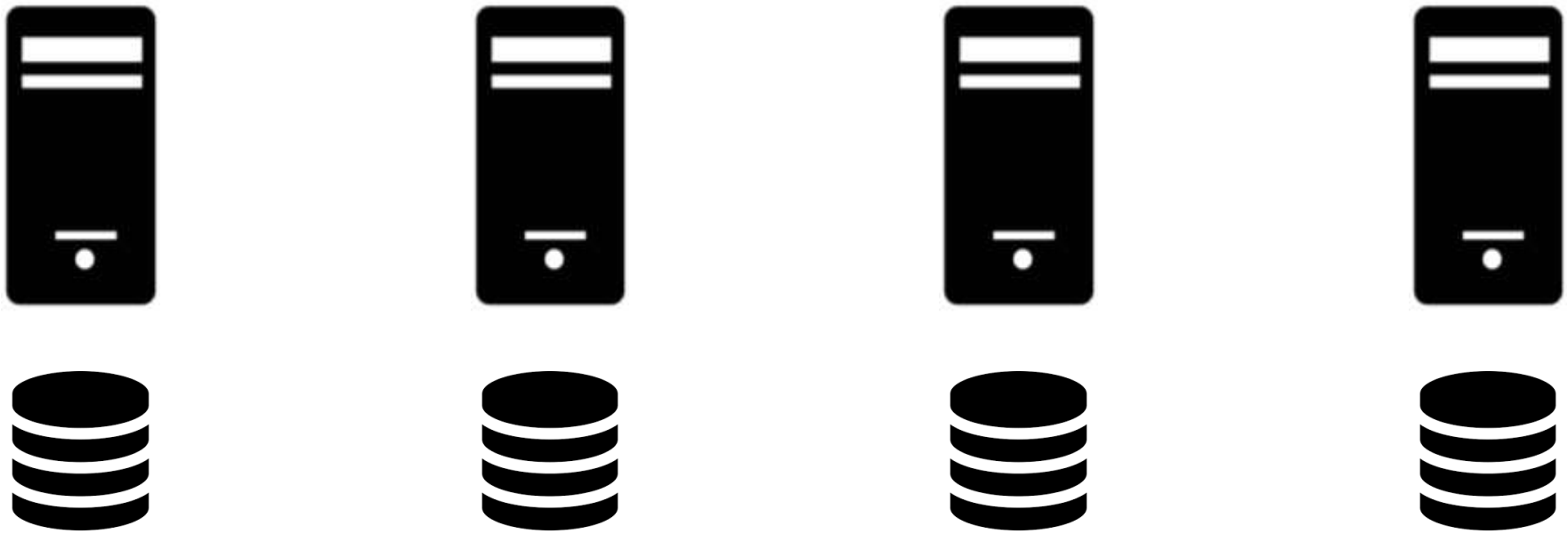


Source: Microsoft

Avoid Data Skew



Even Distribution

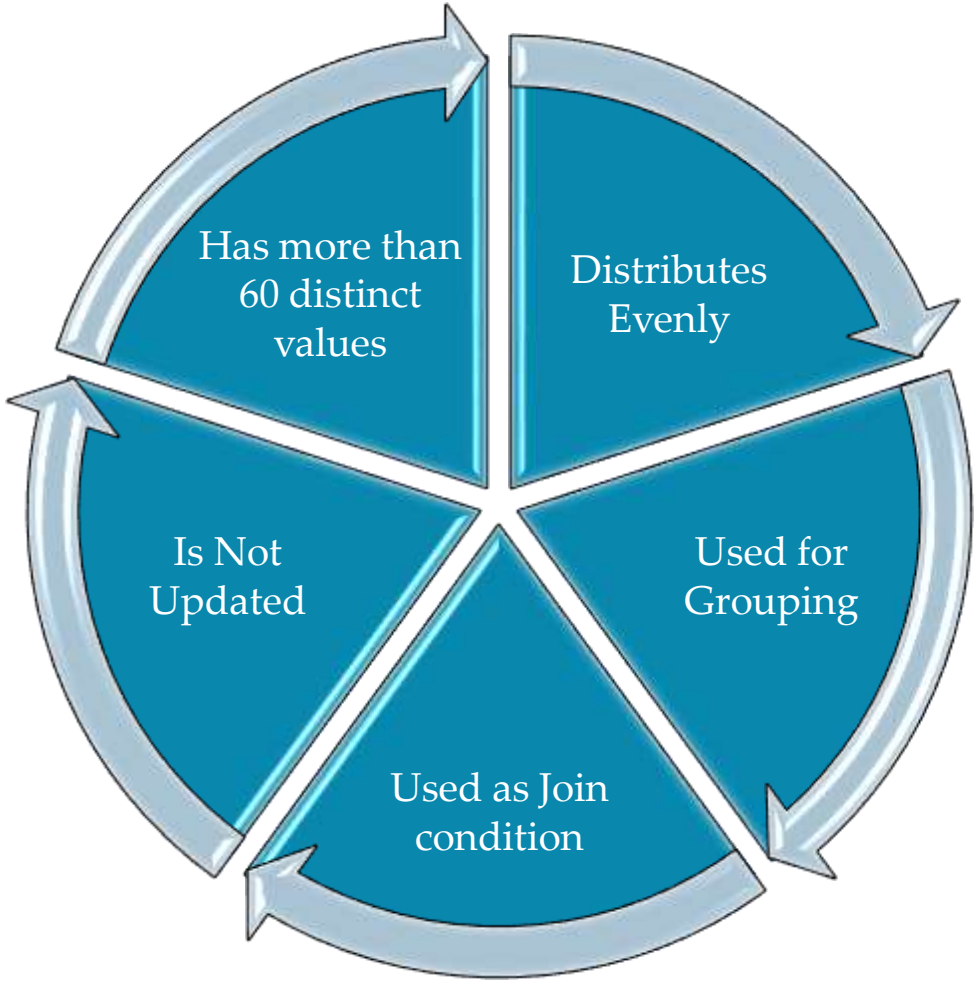


Distribution key

Determines the method in which Azure SQL Data Warehouse spreads the data across multiple nodes.

Azure SQL Data Warehouse uses up to 60 distributions when loading data into the system.

Good Hash Key



What Data Distribution to Use?

Type	Great fit for	Watch out if…
Replicated	Small-dimension tables in a star schema with less than 2GB of storage after compression	• Many write transaction are on the table (insert/update/delete) • You change DWU provisioning frequently • You use only 2-3 columns, but your table has many columns • You index a replicated table
Round-robin (default)	• Temporary/Staging table • No obvious joining key or good candidate column.	Performance is slow due to data movement
hash	• Fact tables • Large dimension tables	The distribution key can't be updated

Data types

- Use the smallest data type which will support your data
- Avoid defining all character columns to a large default length
- Define columns as VARCHAR rather than NVARCHAR if you don't need Unicode

Data types

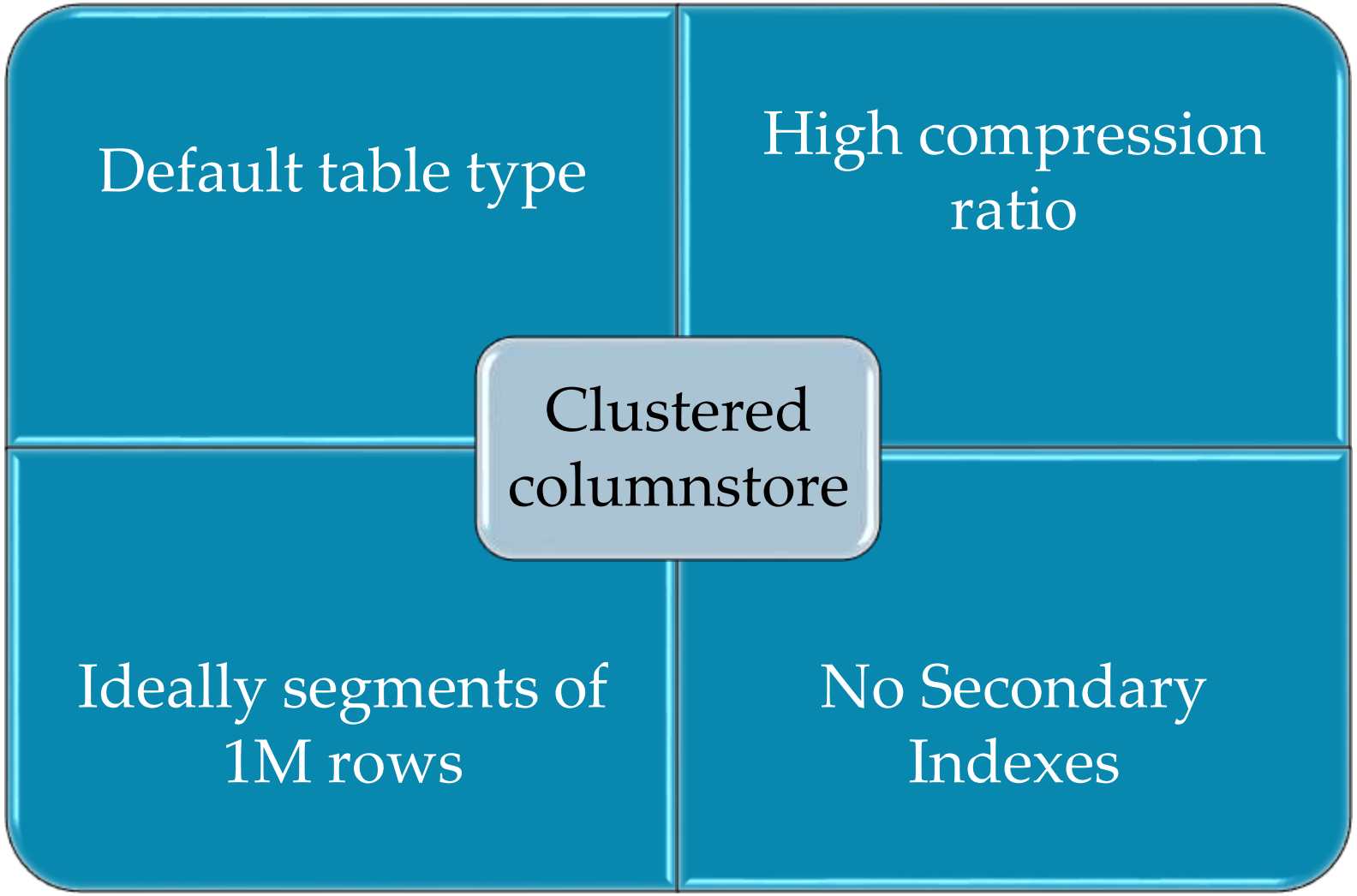
The goal is to not only save space but also move data as efficiently as possible.

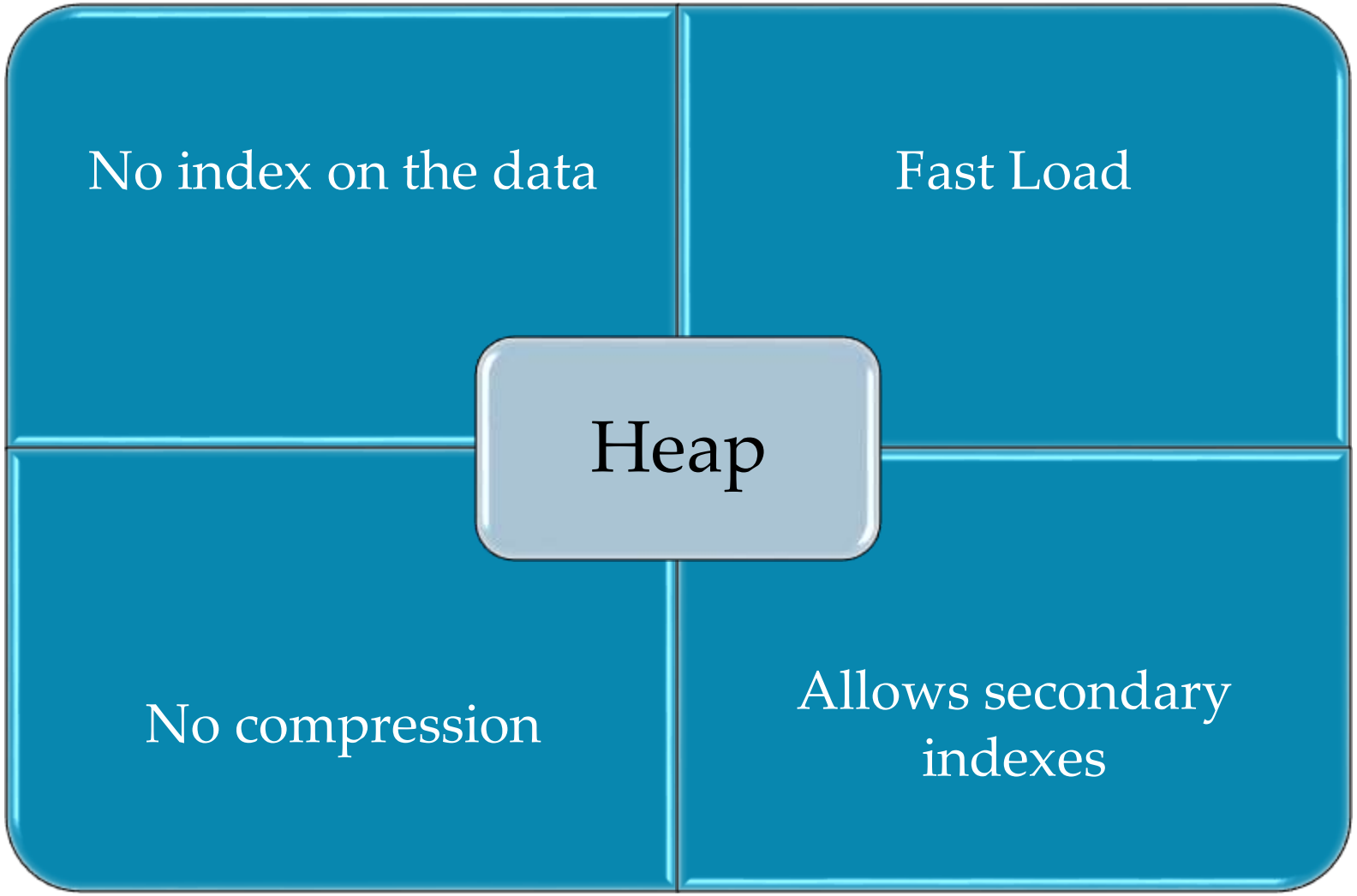
Data types

Some complex data types (XML, geography, etc) are not supported on Azure SQL Data Warehouse yet.

Table types

Clustered columnstore	• Updateable primary storage method • Great for read-only
Heap	• Data is not in any particular order. • Use when data has no natural order.
Clustered Index	• An index that is physically stored in the same order as the data being indexed


Default table type
High compression ratio
Clustered columnstore
Ideally segments of 1M rows
No Secondary Indexes


No index on the data
Fast Load
Heap
No compression
Allows secondary indexes

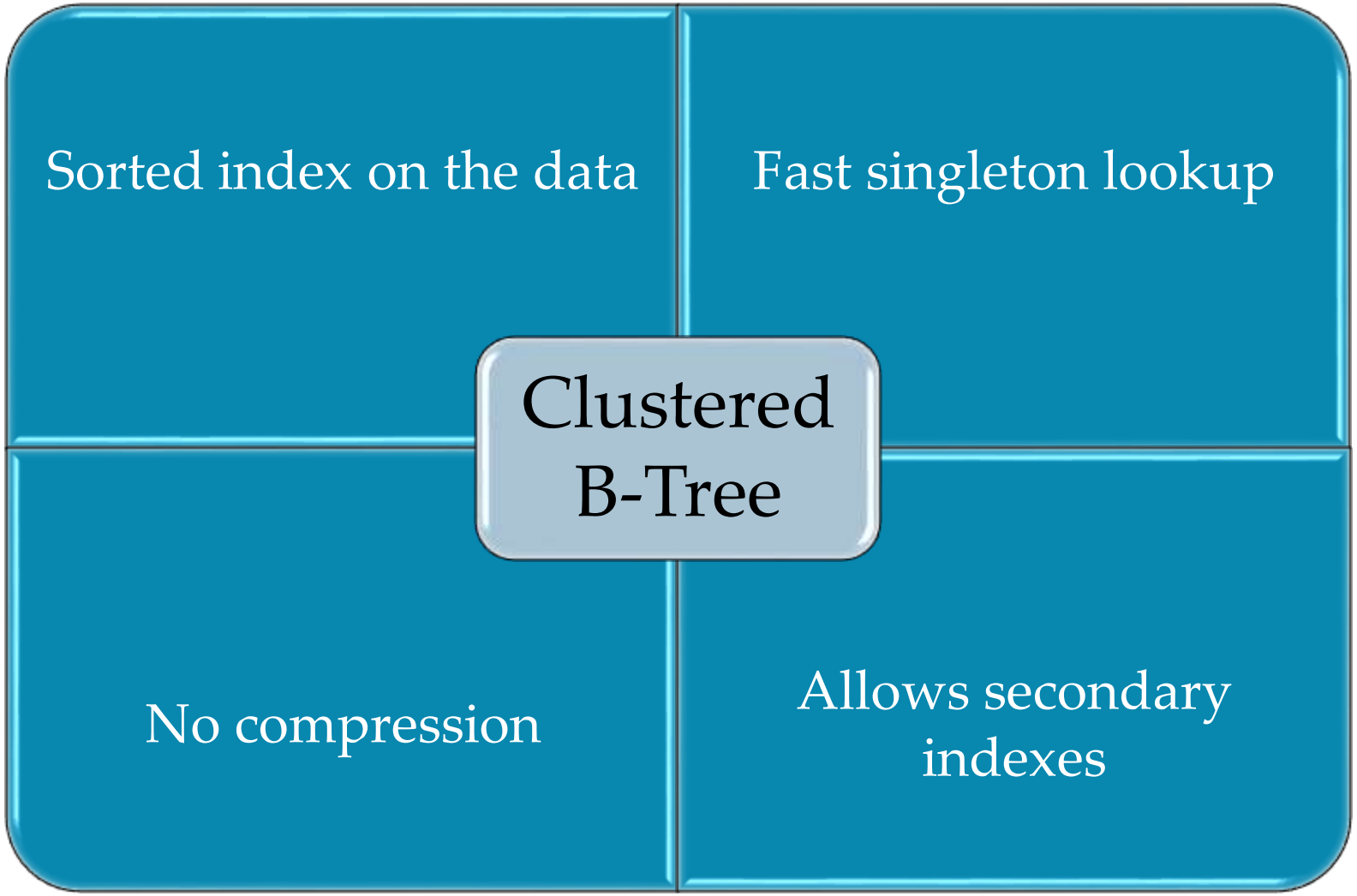
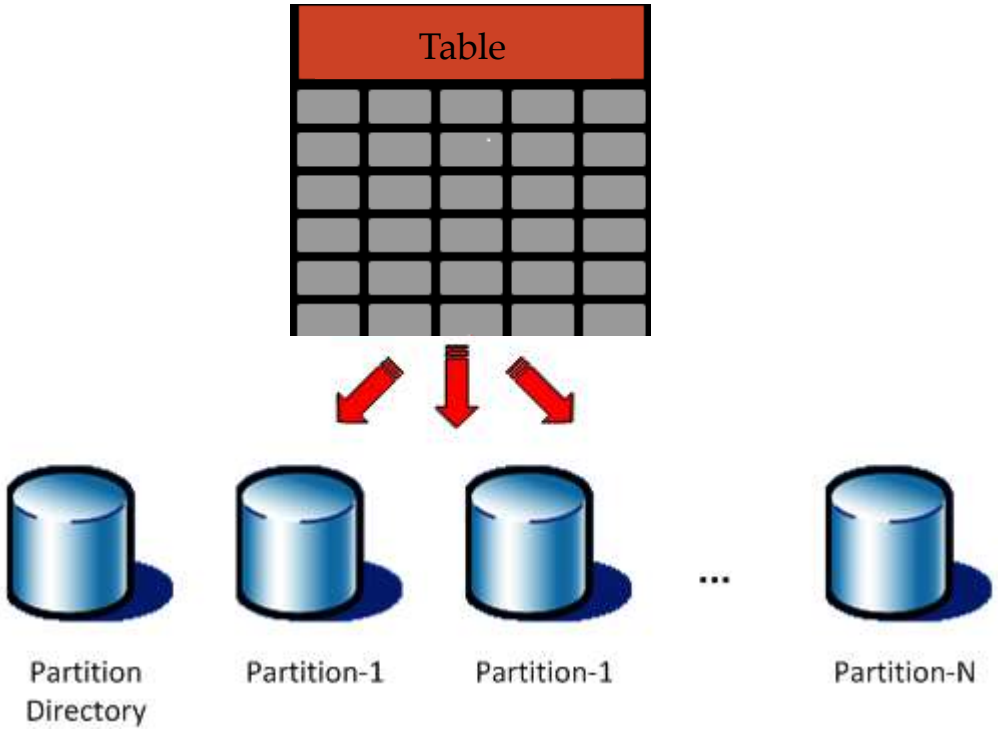

Sorted index on the data
Fast singleton lookup
Clustered B-Tree
No compression
Allows secondary indexes

Table Partitioning

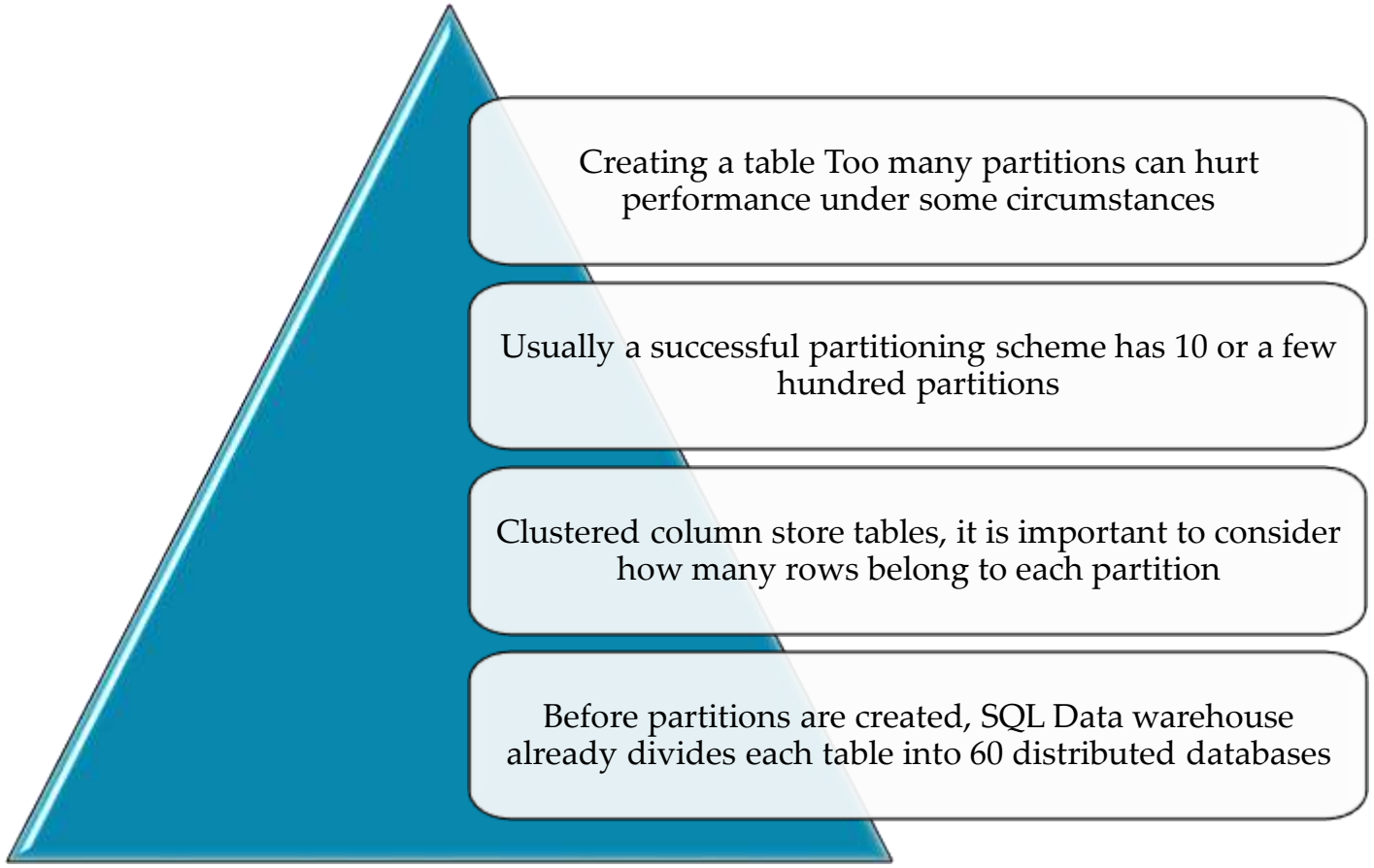


Partitioning

- Table partitions enable you to divide your data into smaller groups of data
- Improve the efficiency and performance of loading data by use of partition deletion, switching and merging
- Usually data is partitioned on a date column tied to when the data is loaded into the database
- Can also be used to improve query performance

Why Partitioning?

Partitions best practices



A highly granular partitioning scheme can work in SQL Server but hurt performance in Azure SQL Data Warehouse.

Example

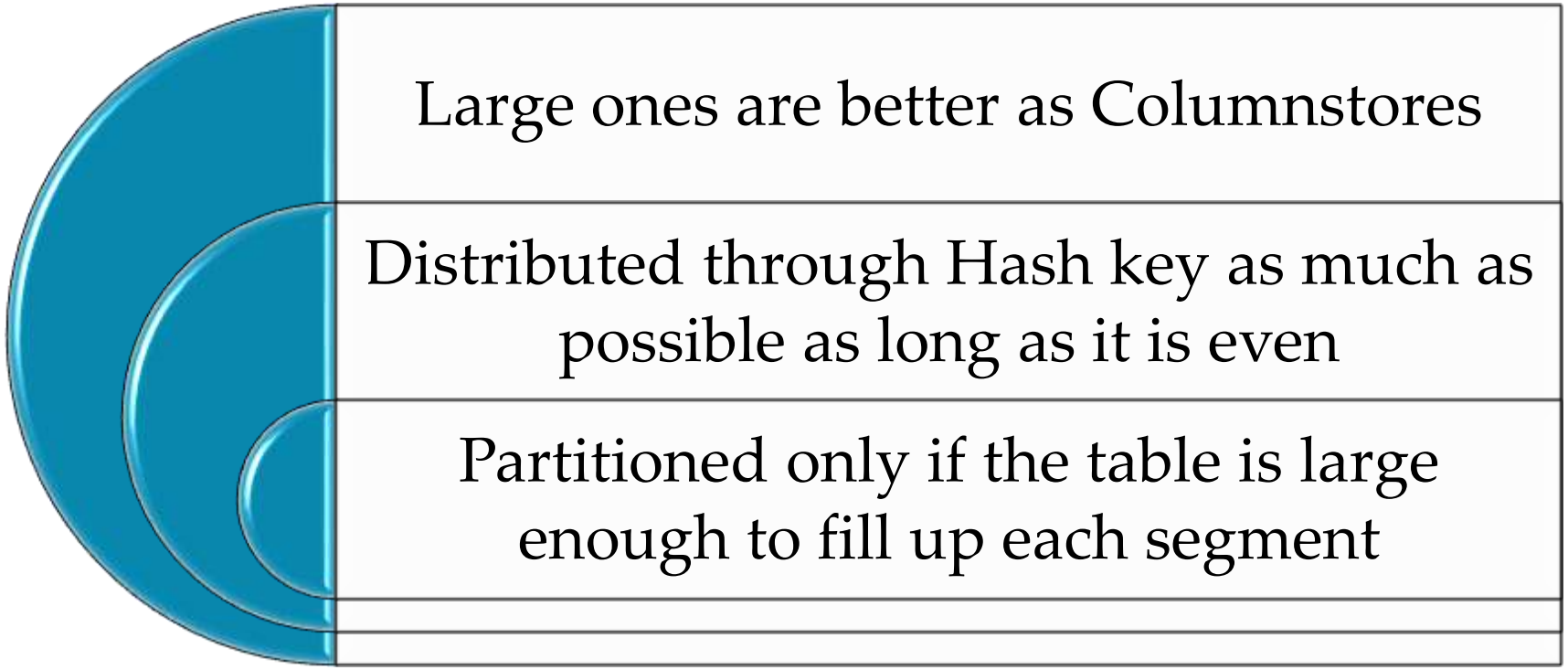
60 Distributions × 365 Partitions = 21900 Data Buckets

21900 Data Buckets × Ideal Segment Size (1M Rows) = 21 900 000 000 Rows

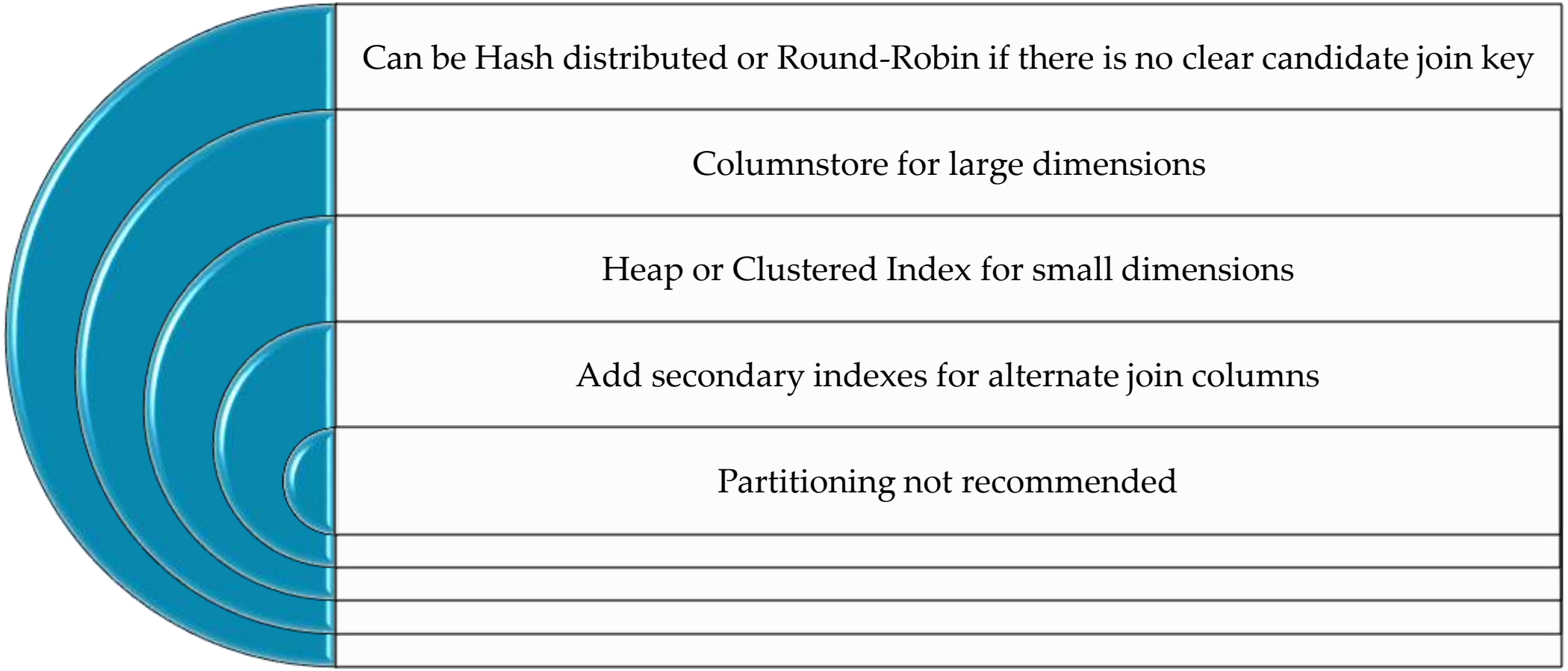
Lower Granularity (week, month) can perform better depending on how much data you have.

How do we apply these principles to a Dimensional model?

Fact Tables


Large ones are better as Columnstores
Distributed through Hash key as much as possible as long as it is even
Partitioned only if the table is large enough to fill up each segment

Dimension Tables



DEMO

Analyse data distribution at On-premises Datawarehouse before migrating to Azure Synapse Data Pool.

- We will use Microsoft's AdventureworksDW database as on-premises data warehouse.
- We will analyse one dimension and one fact table.
- Same process can be repeated to other tables of on-premises database.

Summary

MPP or Massive Parallel Processing

Billing = Compute + Storage

Data Distribution (Hash, Round-robin, Replicate)

Data types and Table types

Partitioning Data

Best practice – Fact and Dimension table design

Demo – Analyse Data Distribution

Azure Synapse Analytics