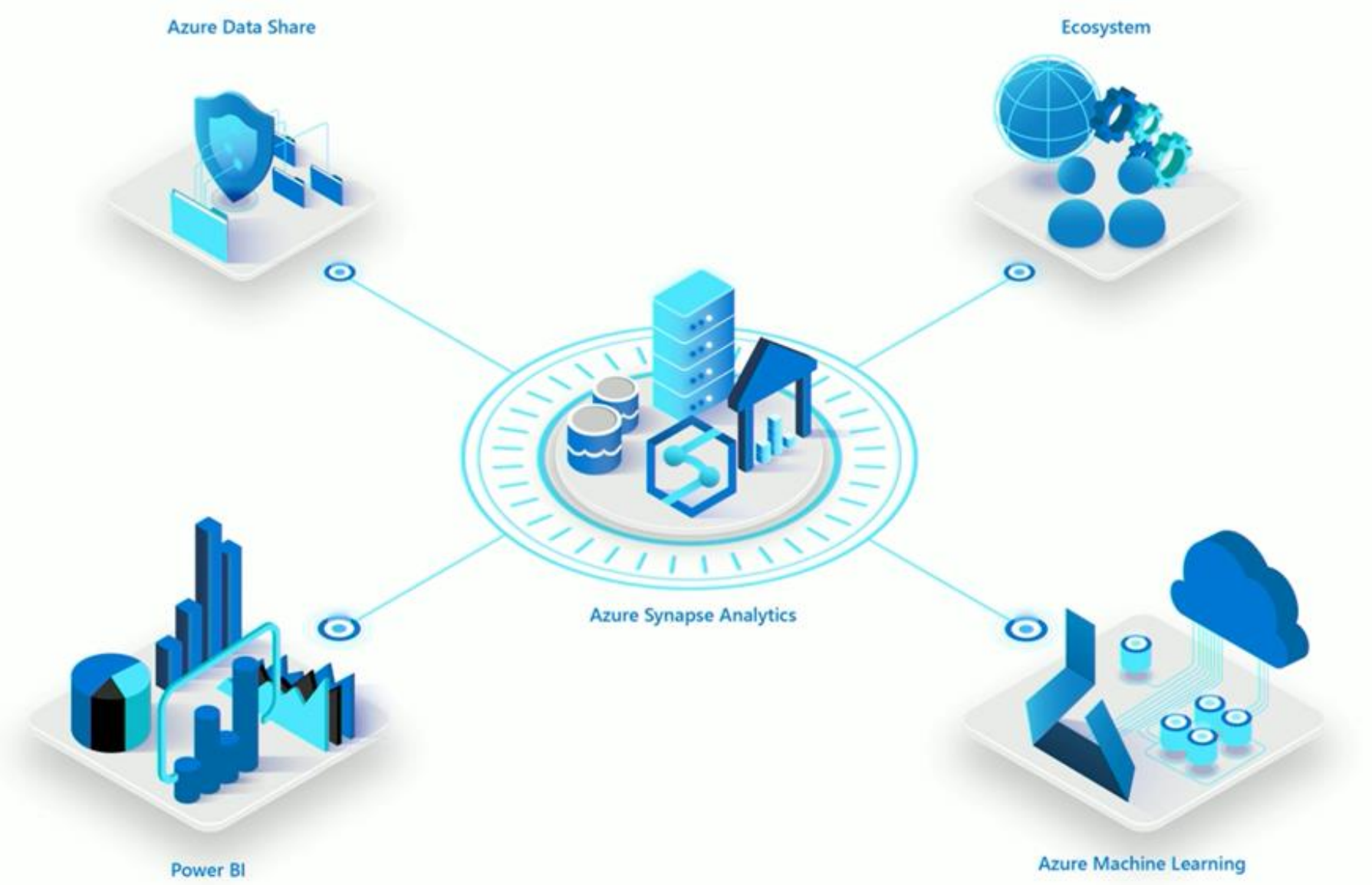
Azure Synapse Analytics Service

Eshant Garg
Data Engineer, Architect, Advisor
eshant.garg@gmail.com

Azure Synapse Analytics


Azure Data Share
Ecosystem
Azure Synapse Analytics
Power BI
Azure Machine Learning

Introduction

Why Warehousing in Cloud

Azure Synapse Service – a Game Changer

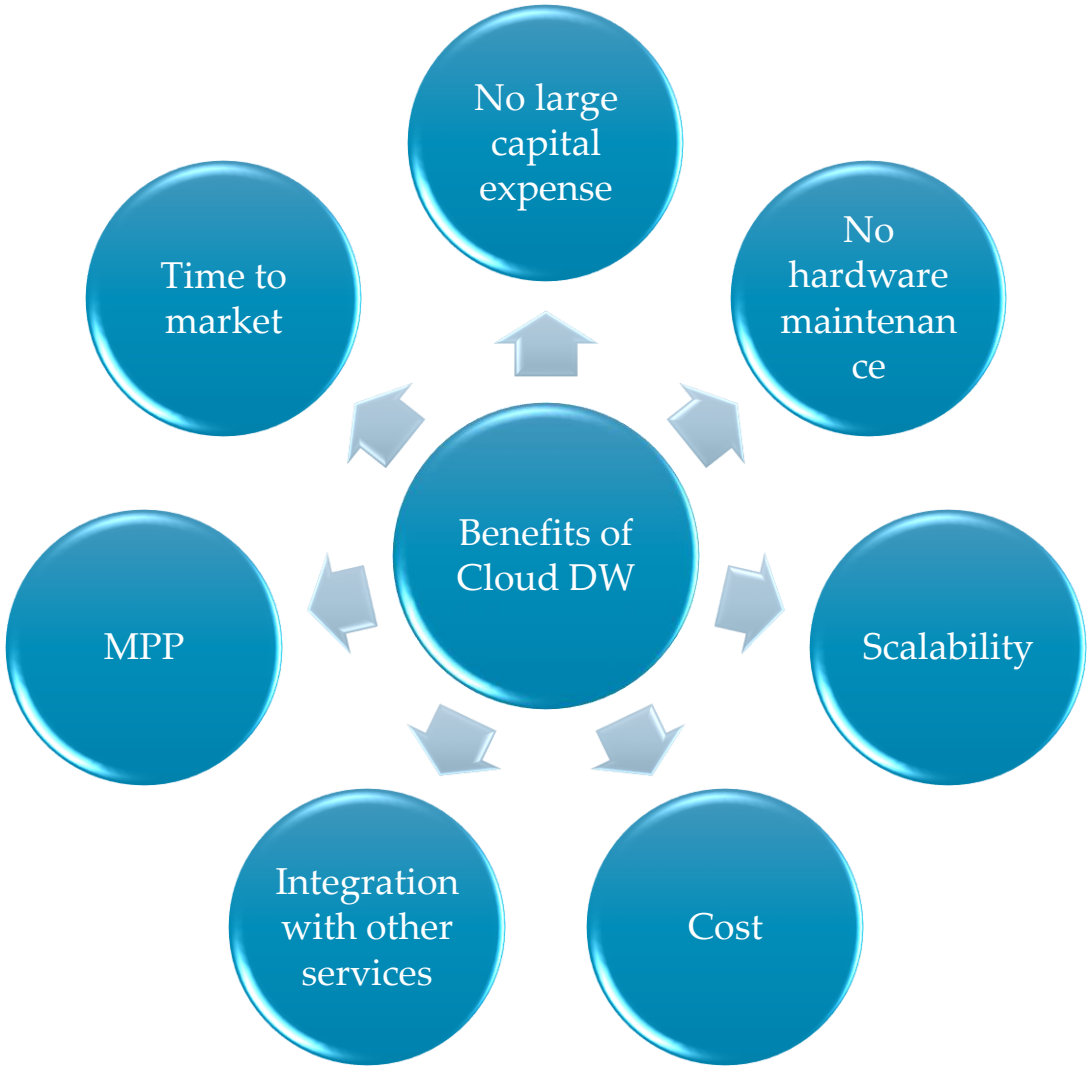
Traditional vs Modern DW Architecture

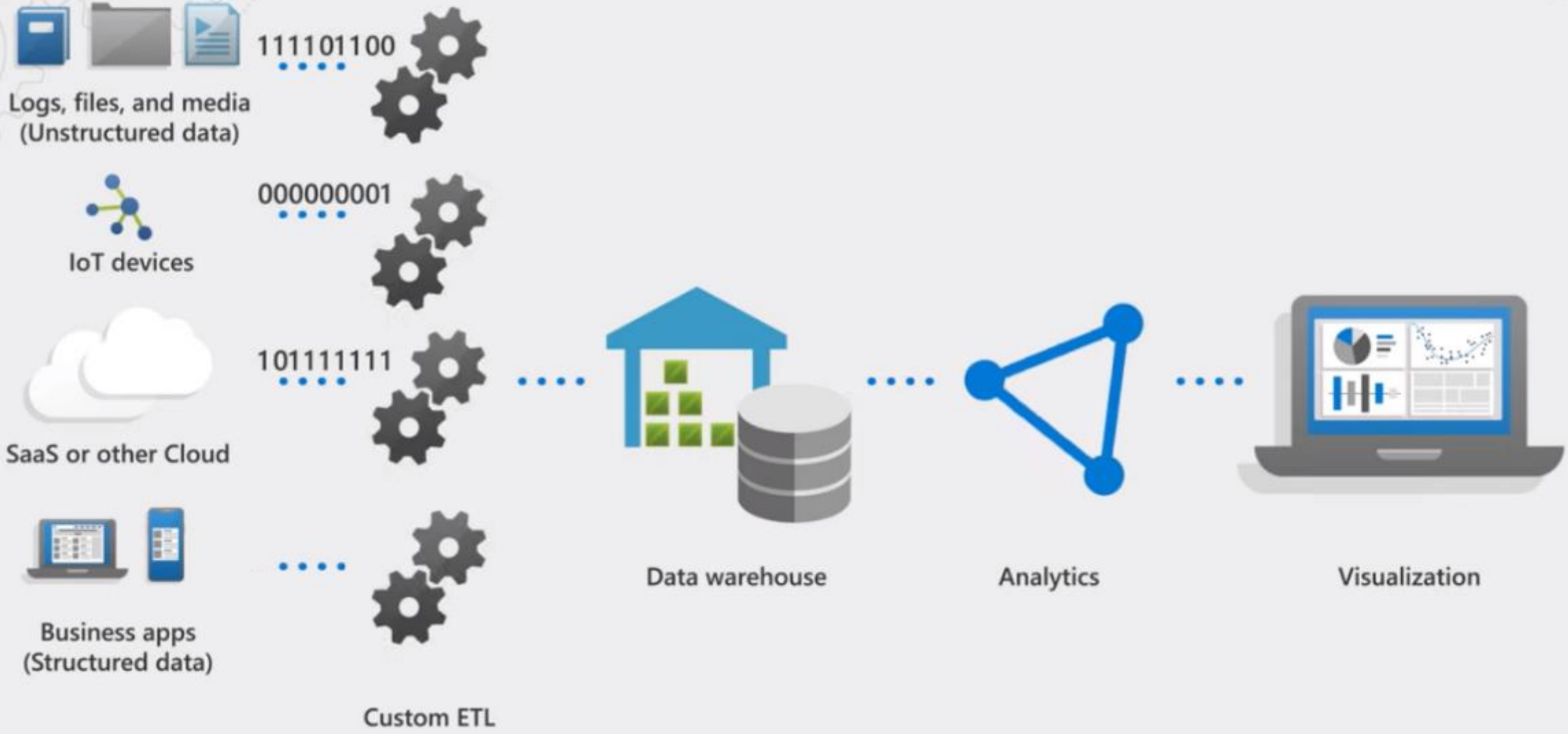
Modern vs Synapse DW Architecture

Azure Synapse Studio – Unified Experience

Demo – Provision, Scale, Pause, Firewall settings

Why Warehousing in Cloud?

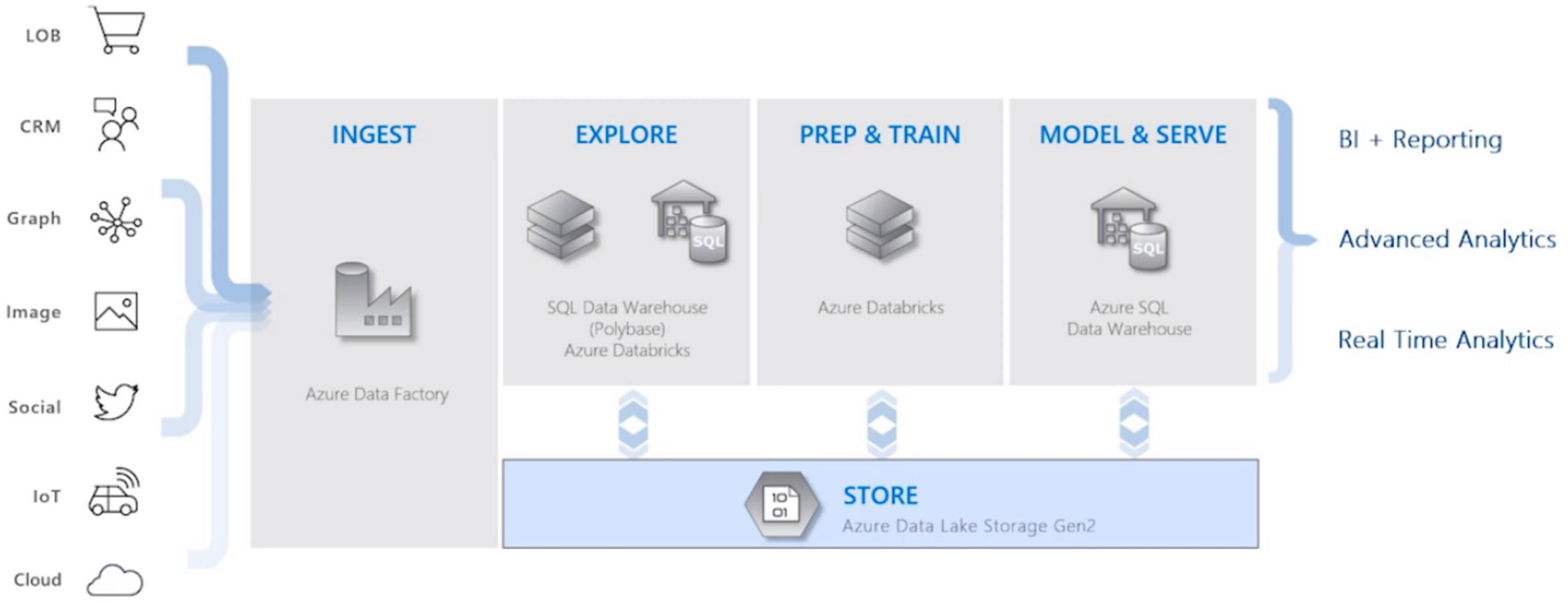




Old data warehouse

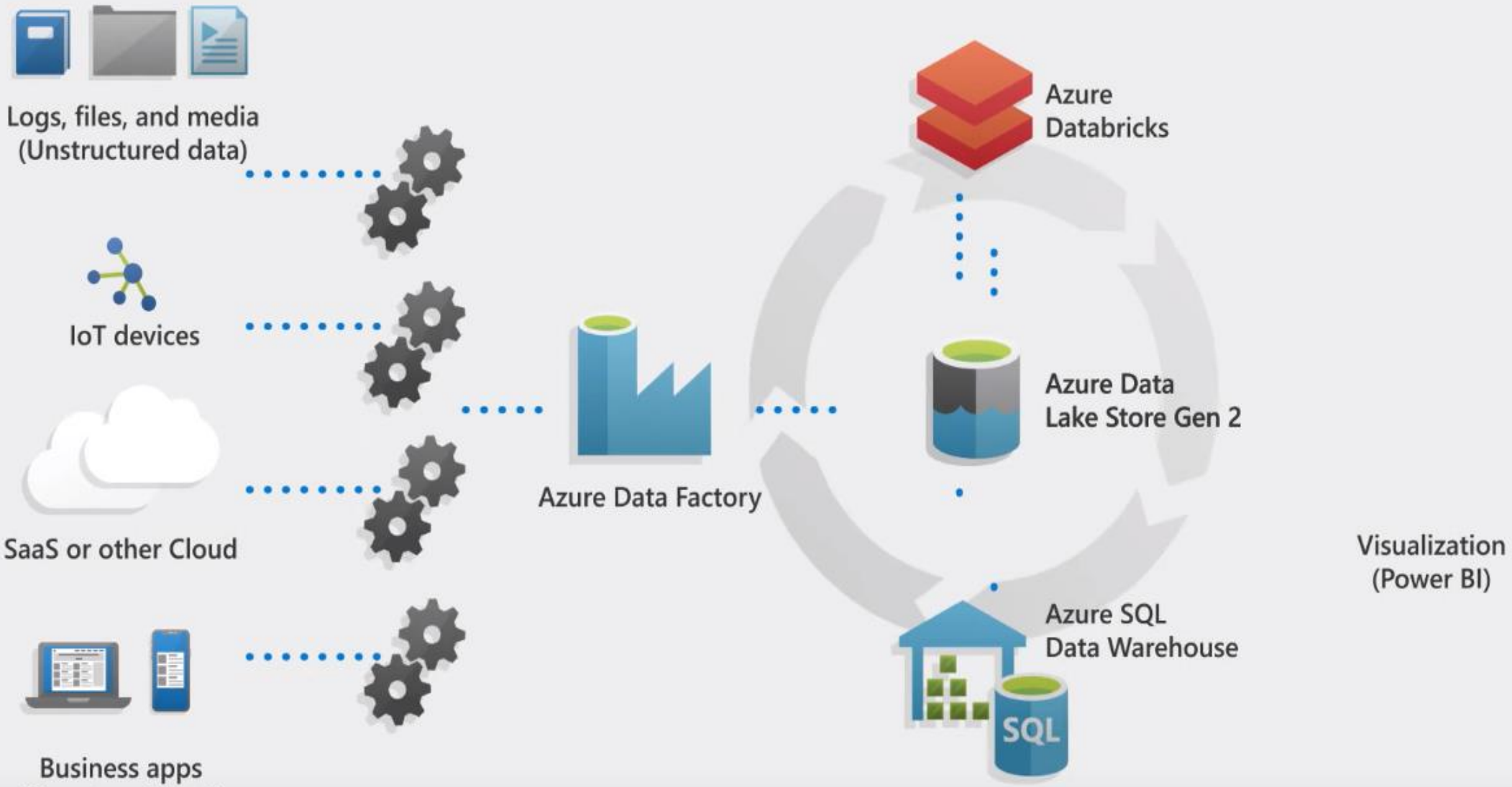
Source: Microsoft

Modern Data Warehousing



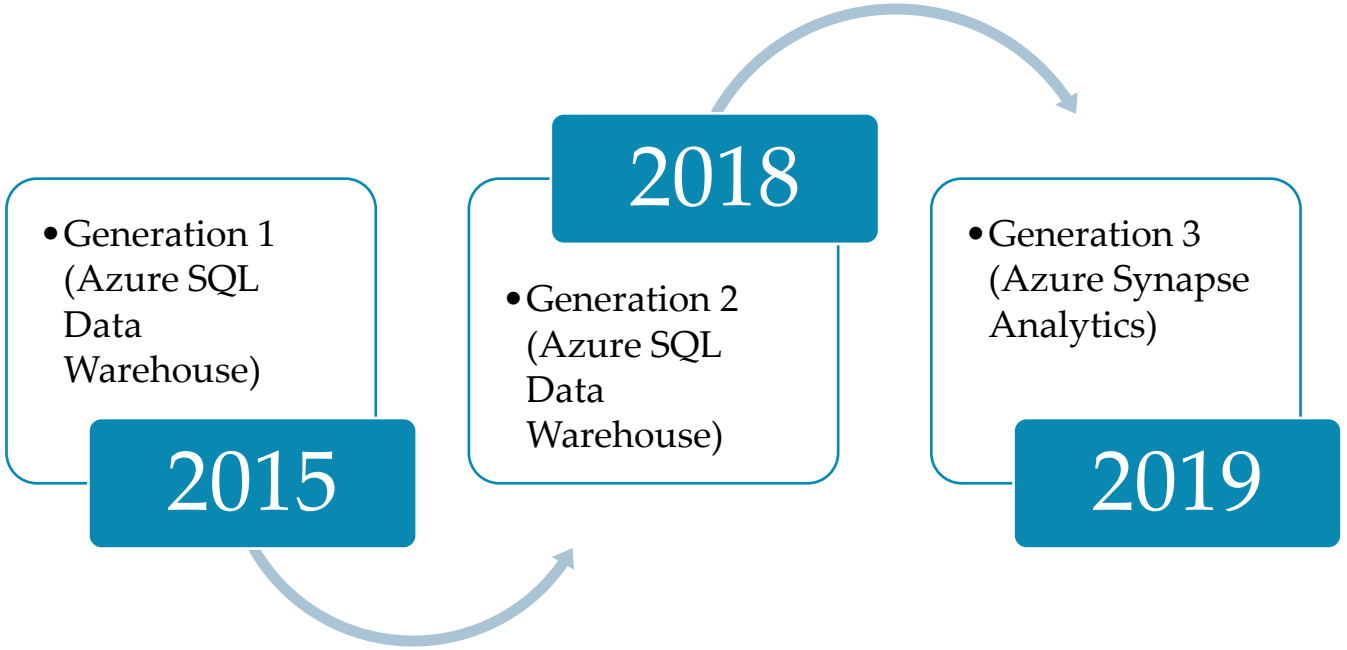
Source: Microsoft

Modern Data Warehouse



Source: Microsoft

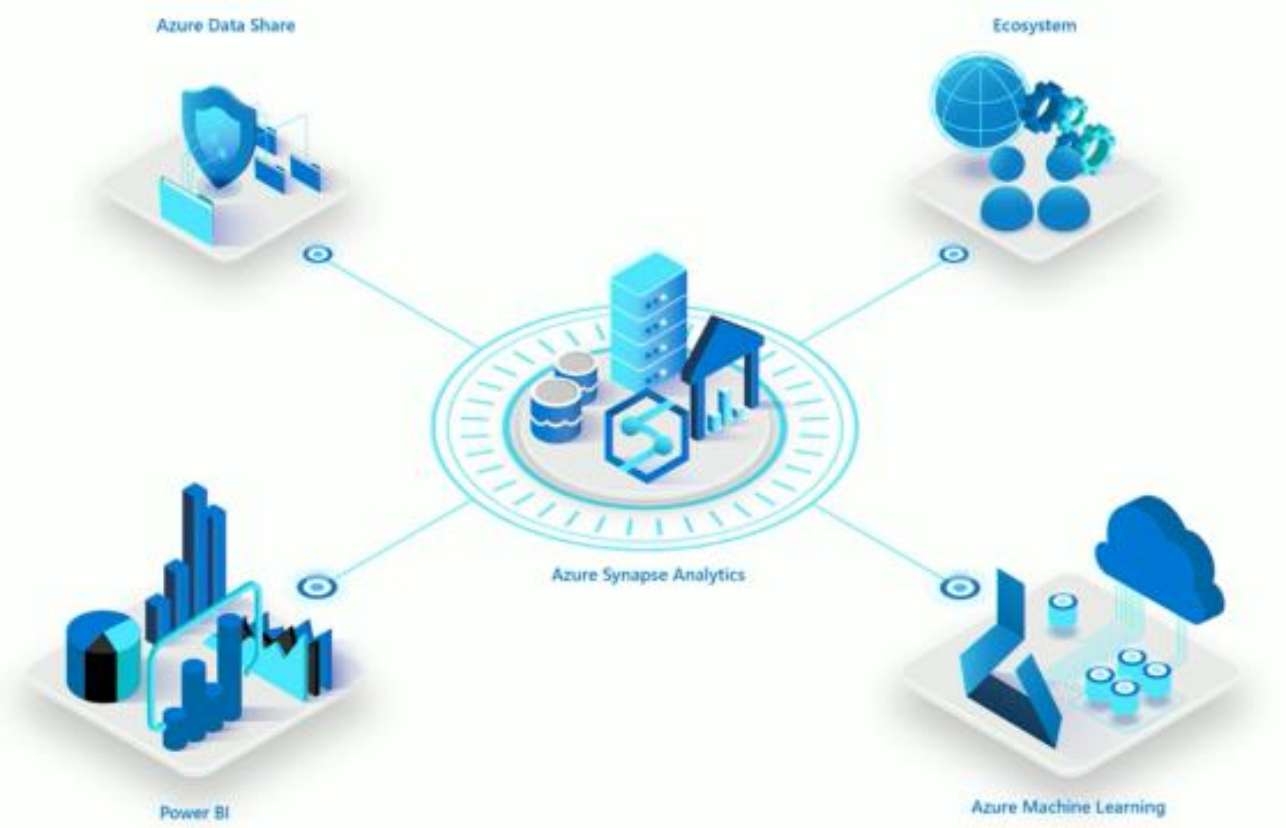
Azure SQL Data warehouse



Azure Synapse Analytics

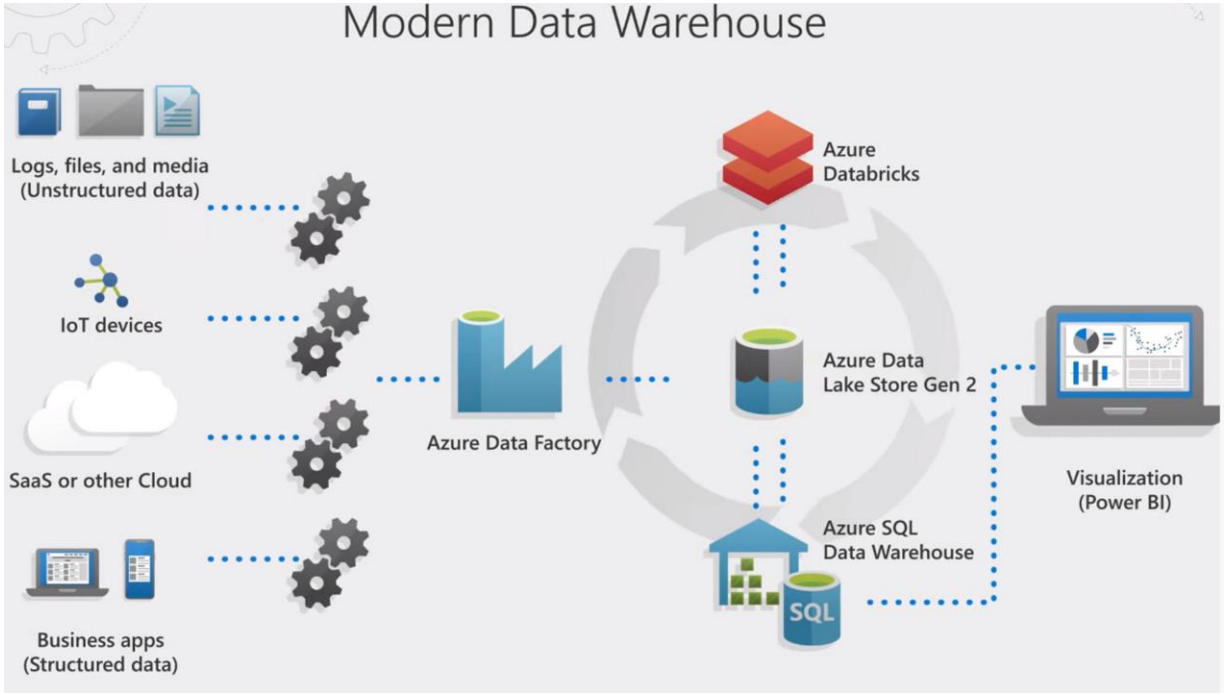
"Synapse is the next generation of Azure SQL Data Warehouse, blending big data analytics, data warehousing, and data integration into a single unified service that provides end-to-end analytics with limitless scale."

Azure Synapse Analytics

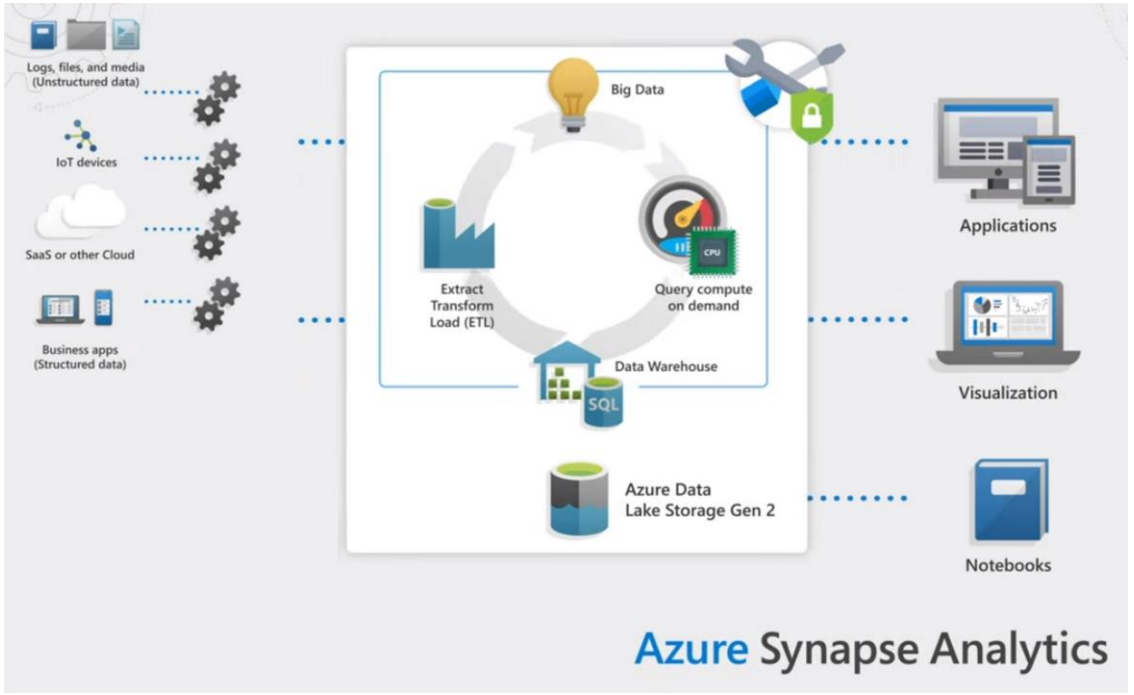


Source: Microsoft

Modern vs Synapse Architecture

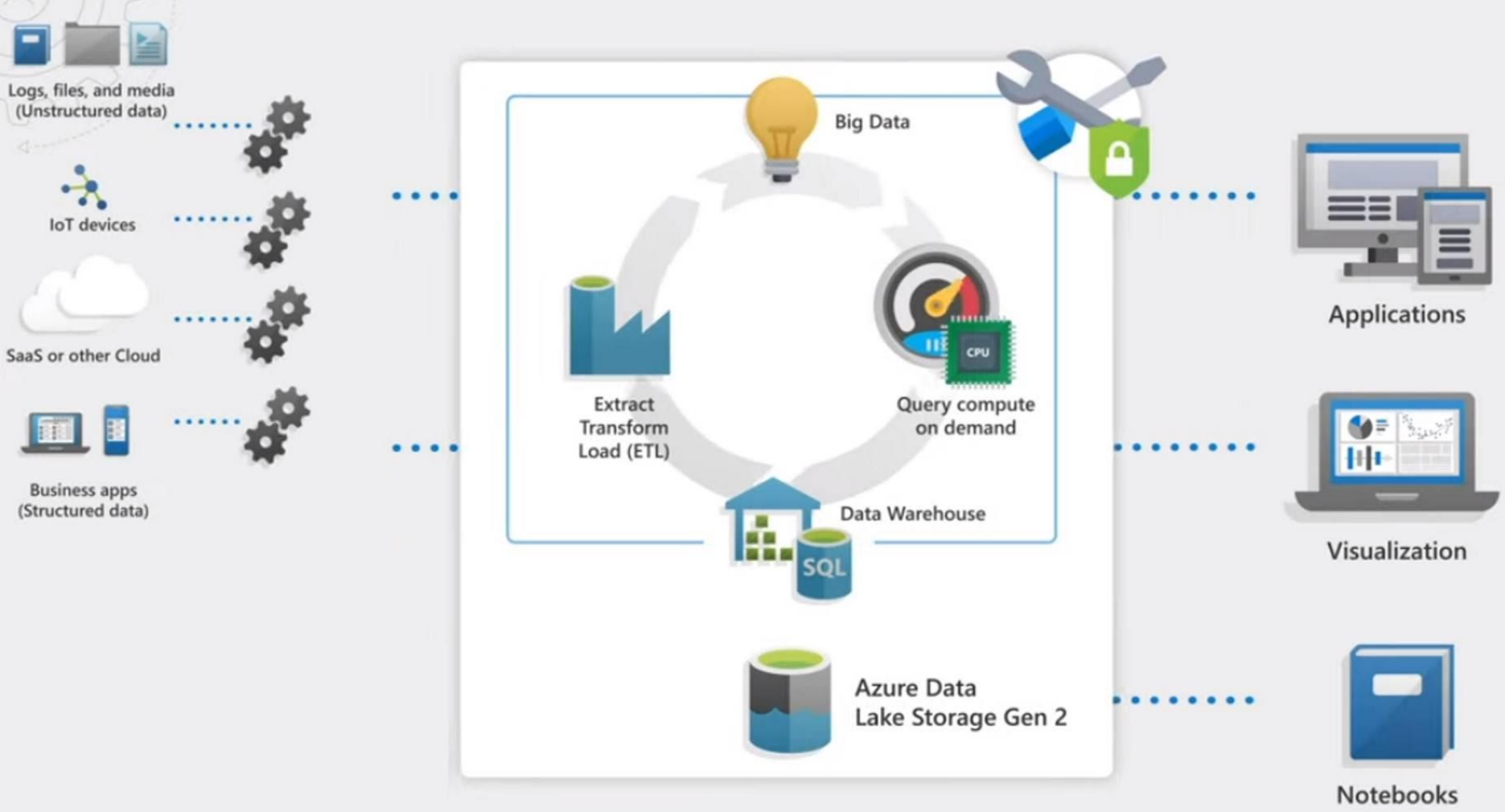


Modern Data Warehouse Architecture

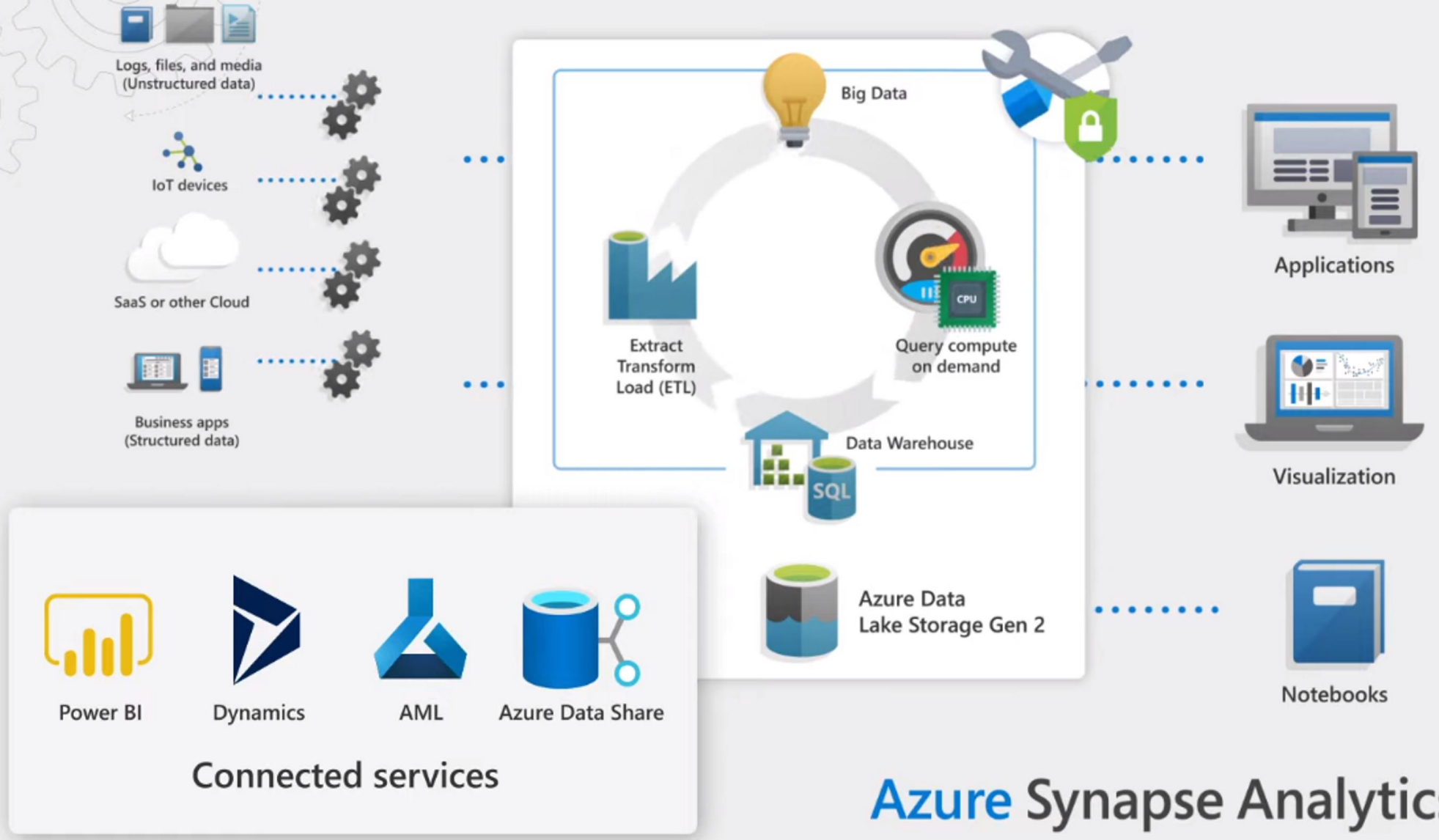


Synapse Analytics Architecture

Source: Microsoft



Azure Synapse Analytics



Source: Microsoft

Azure Synapse Analytics

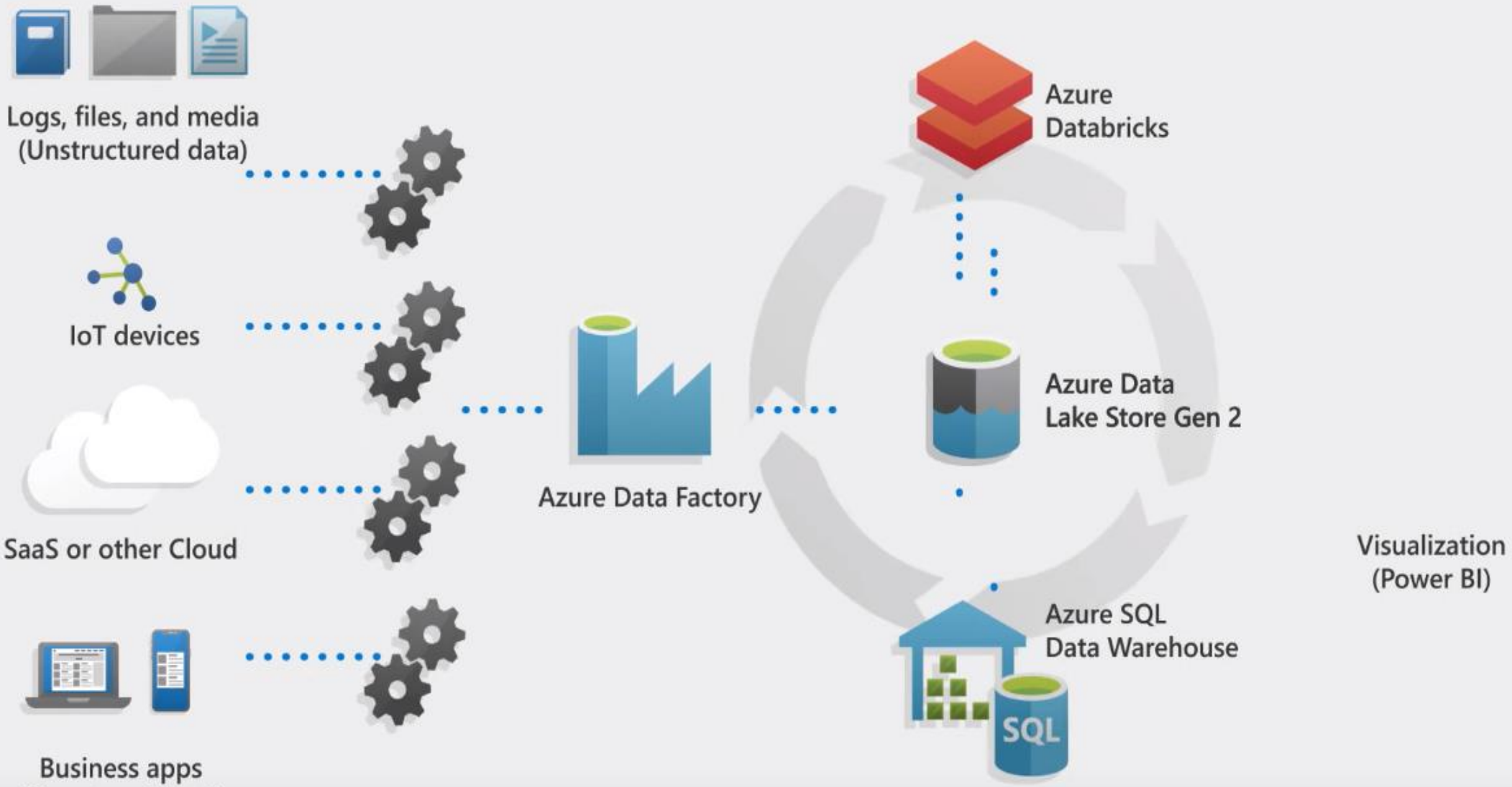
Source: Microsoft

Demo – Provision Azure Synapse Service

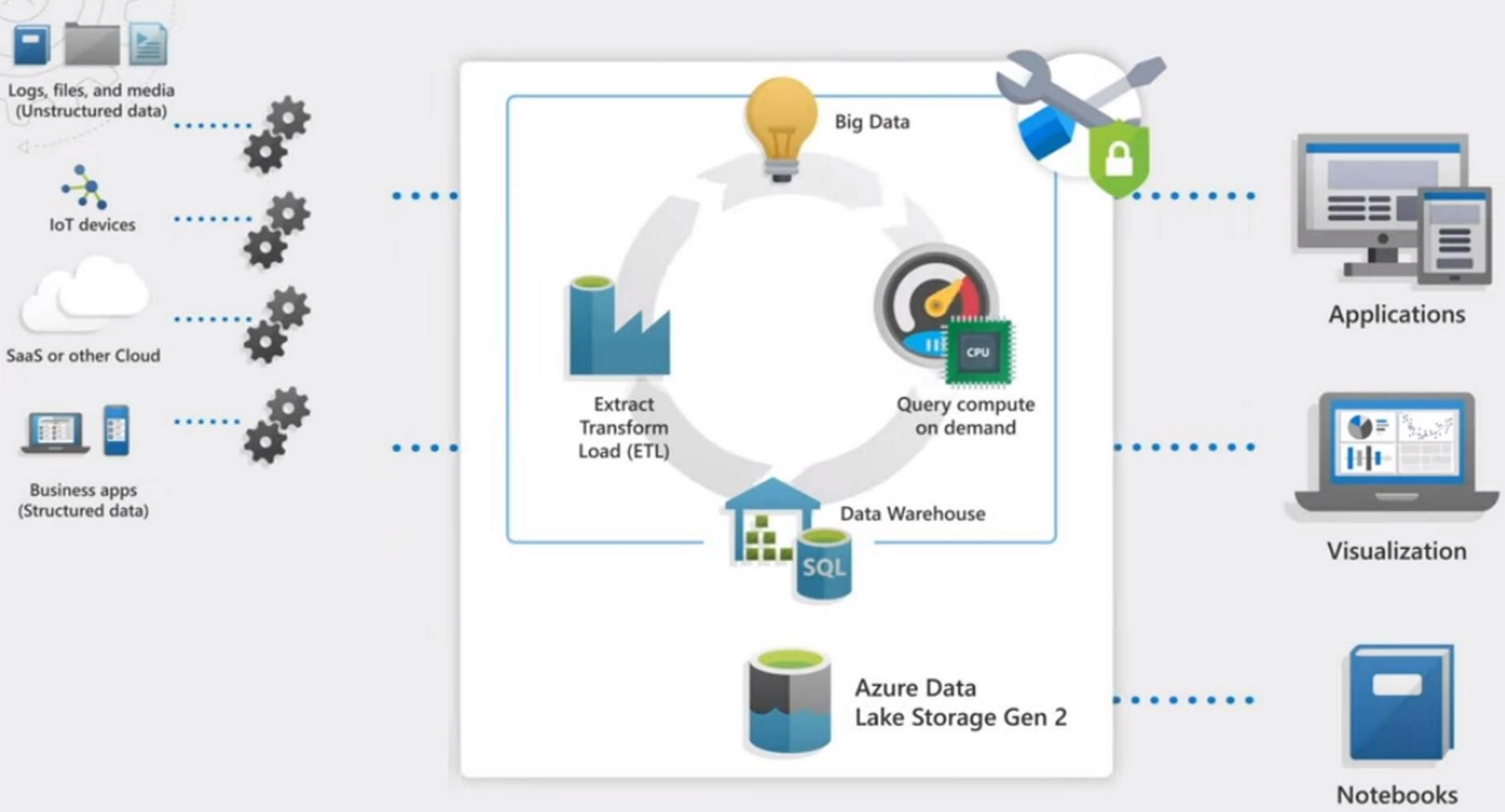
1. Create SQL Server
2. Create Synapse SQL Pool (Azure SQL Data Warehouse)
3. Pause/Resume Compute Node
4. Create Firewall Rule
5. Connect with Microsoft SQL Server Management Studio

Azure Synapse Analytics

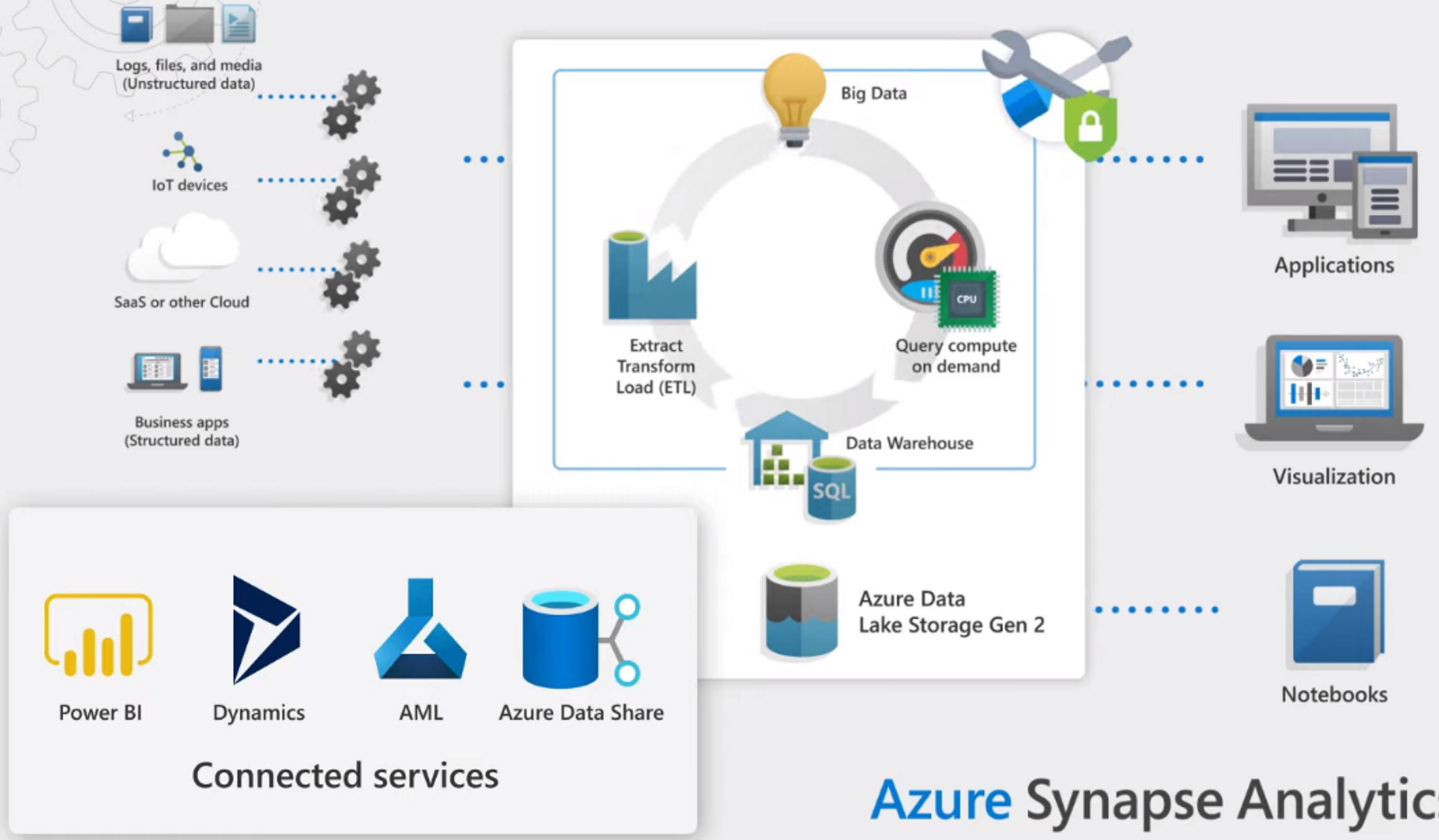
Modern Data Warehouse



Source: Microsoft



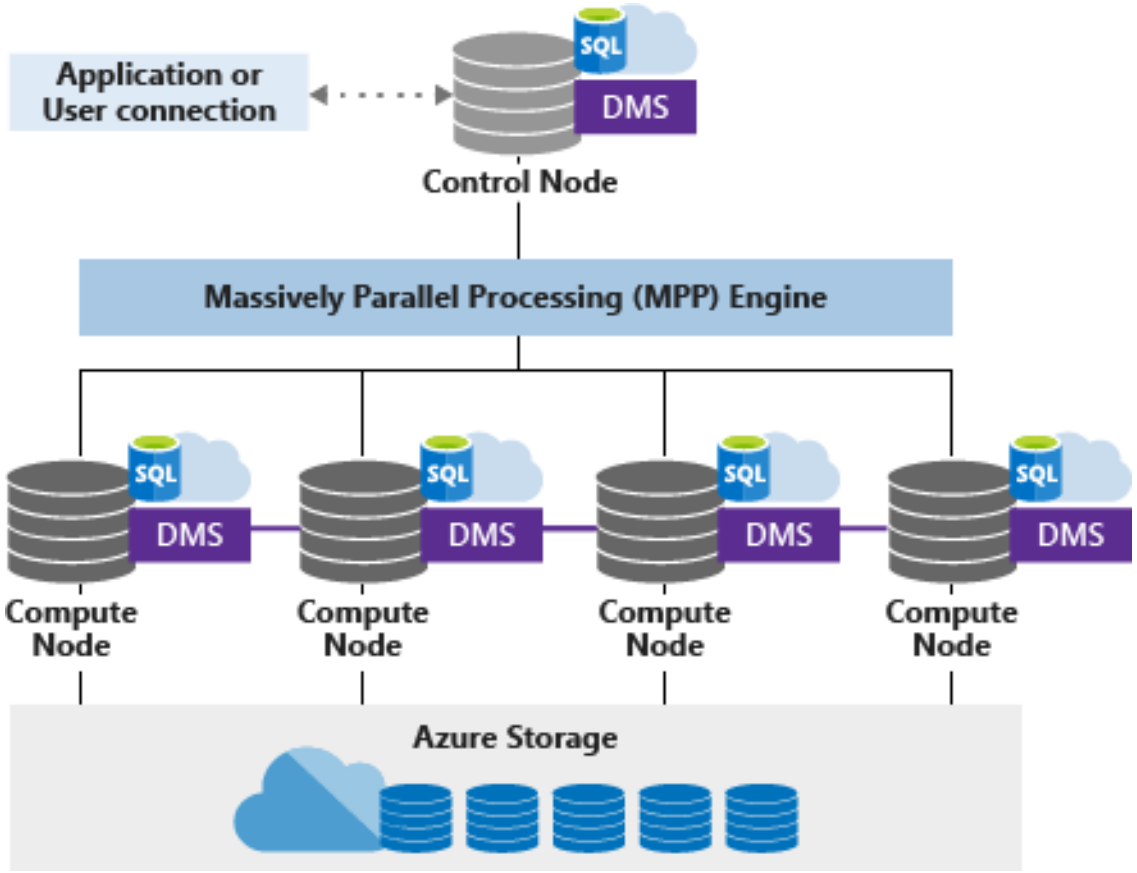
Azure Synapse Analytics



Source: Microsoft

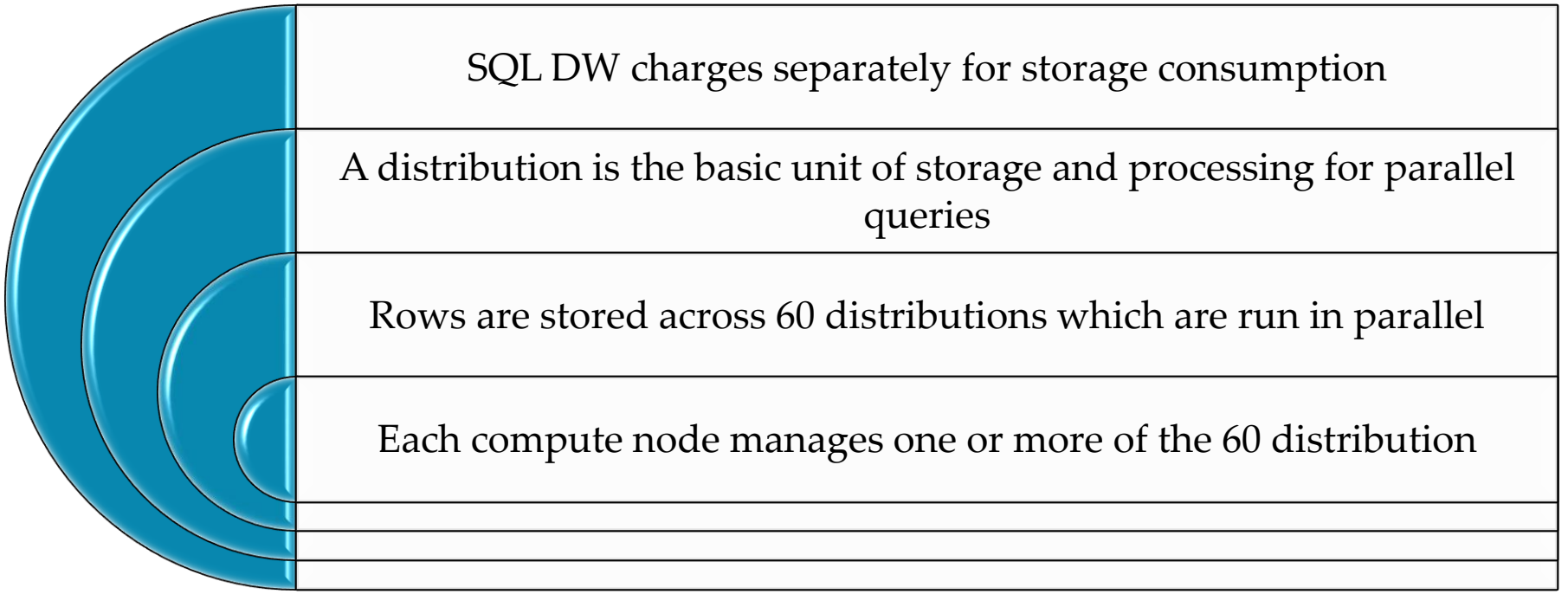
Azure Synapse MPP Architecture

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

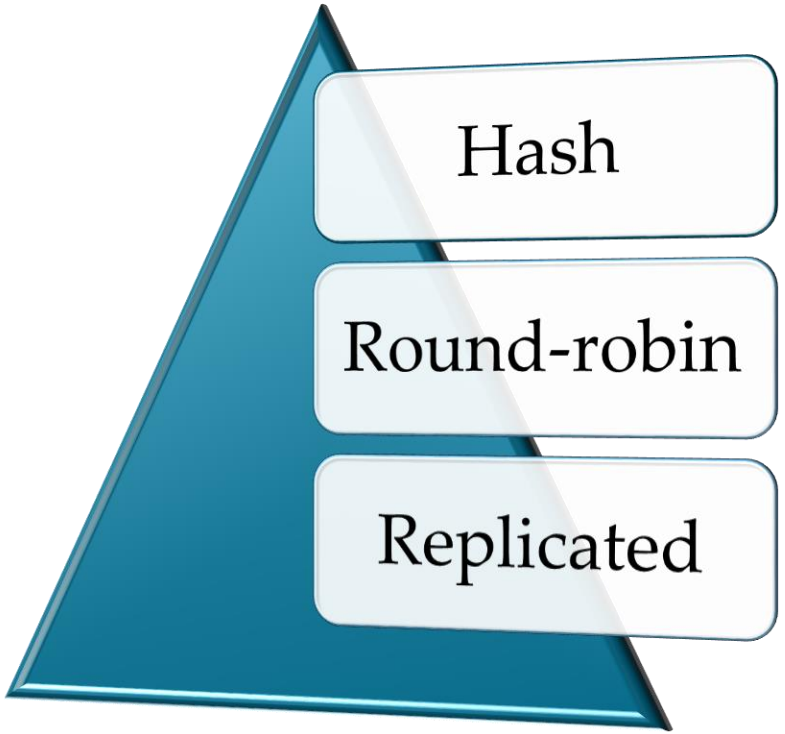


Source: Microsoft

Azure Storage and Distribution



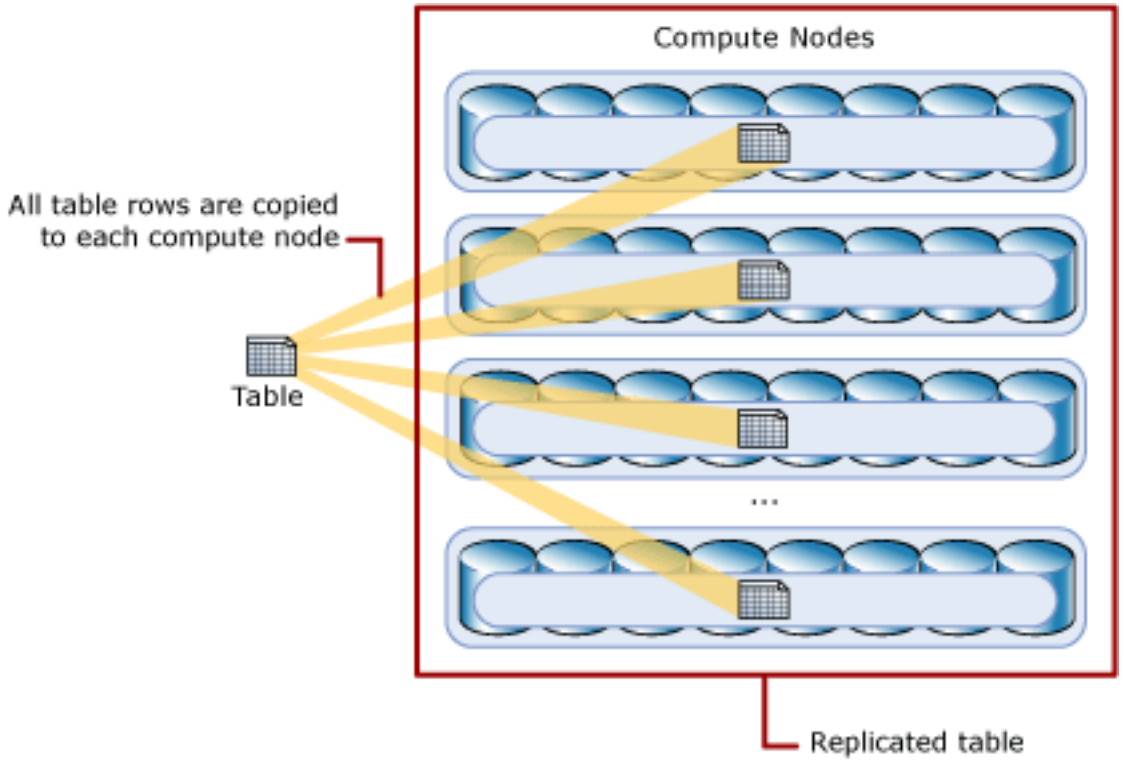
Sharding Patterns



Replicated Tables

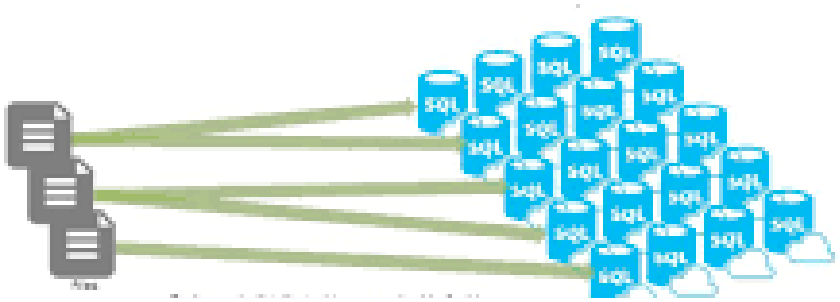
- Caches a full copy on each compute node.
- Used for small tables

```
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



- 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

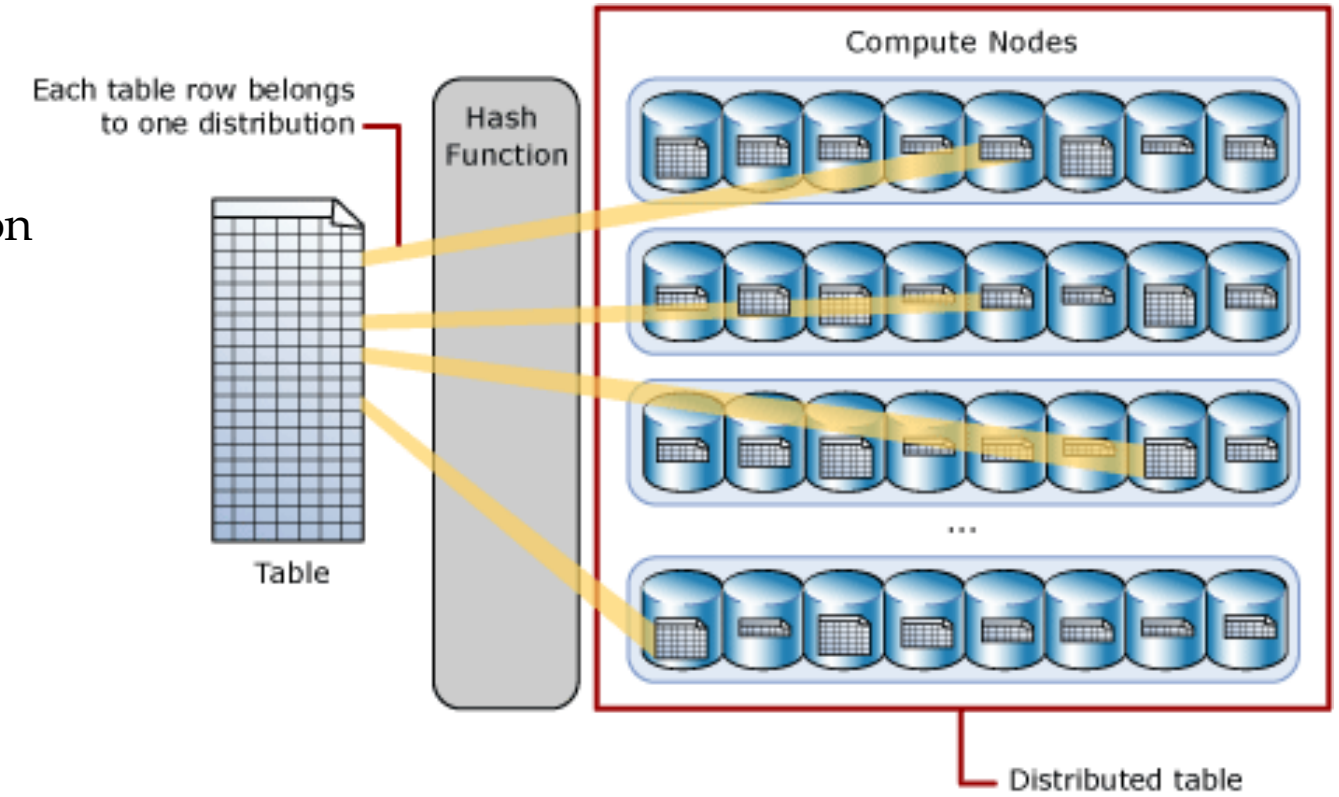
```
CREATE TABLE [dbo].[Dates](
    [Date] [datetime2](3) ,
    [DateKey] [decimal](38, 0) ,
    ..
    ..
    [WeekDay] [nvarchar](100) ,
    [Day Of Month] [decimal](38, 0)
)

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

Source: Microsoft

Hash Distribution Tables

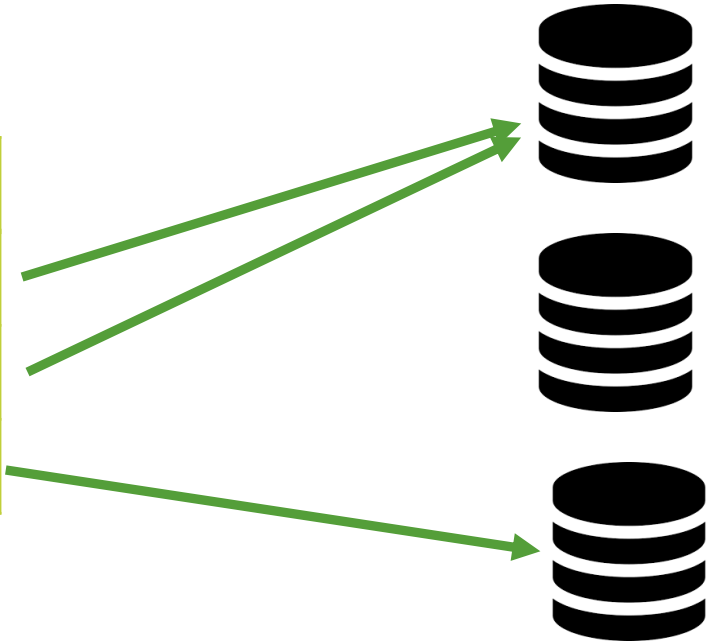
- Highest performance for large tables
- Each row belong to one particular distribution
- It is used mostly for larger tables



Source: Microsoft

Hash Distribution Tables

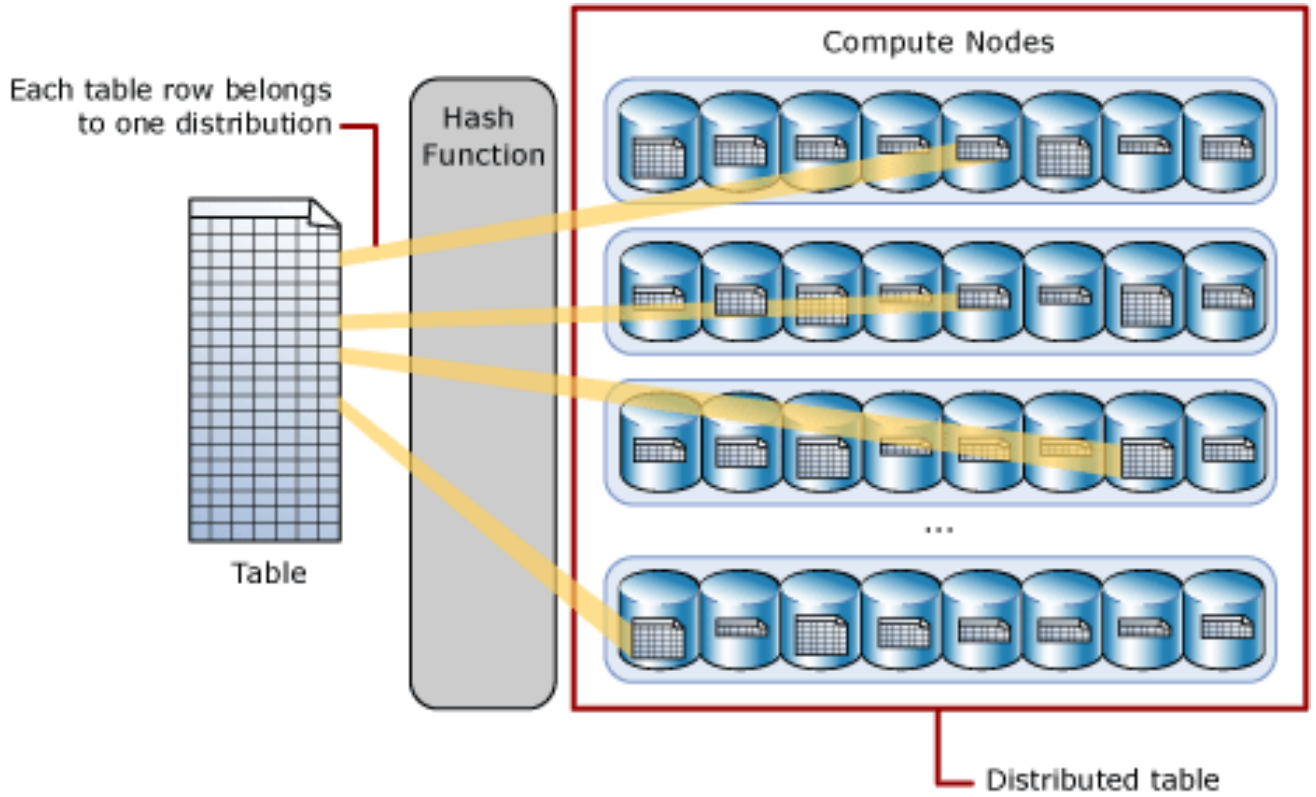
Record	Product	Store
1	Soccer	New York
2	Soccer	Los Angeles
3	Football	Phoenix



Hash Distribution Tables

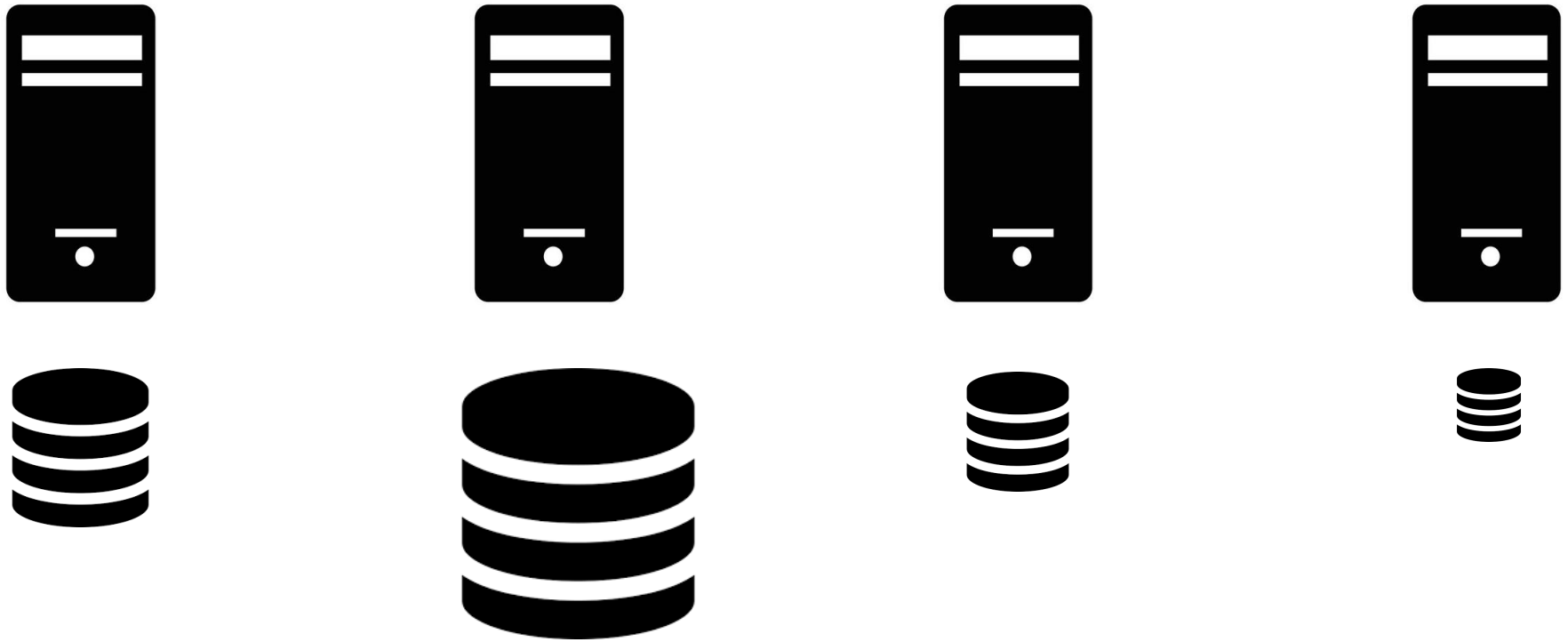
- Highest performance for large tables
- Each row belong to one particular distribution
- It is used mostly for larger tables

```
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])
)
;
```

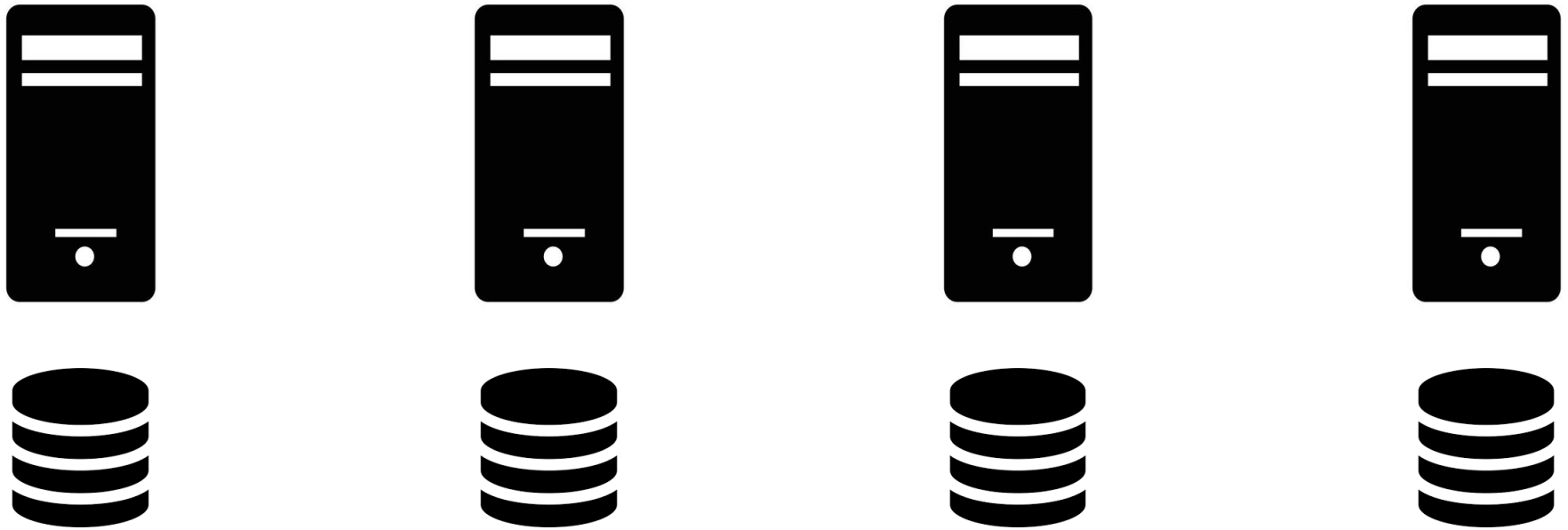


Source: Microsoft

Avoid Data Skew



Even Distribution

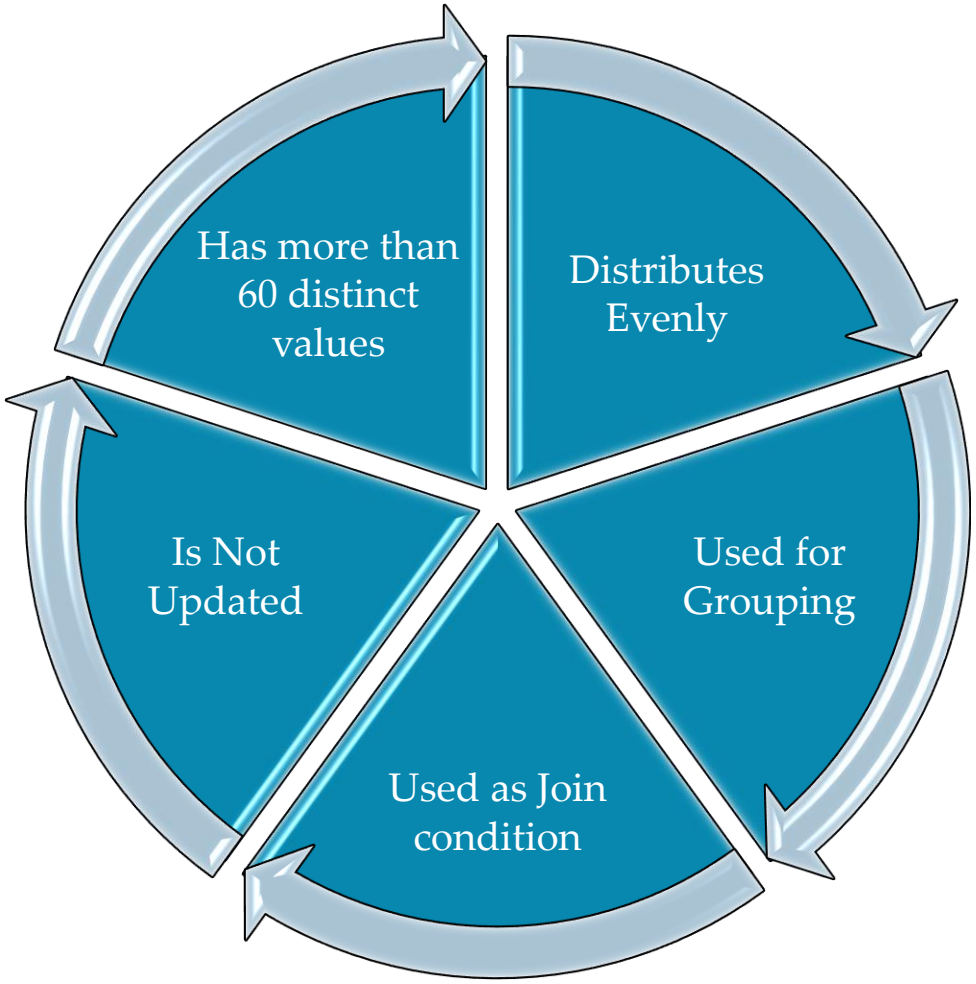


Distribution key

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.

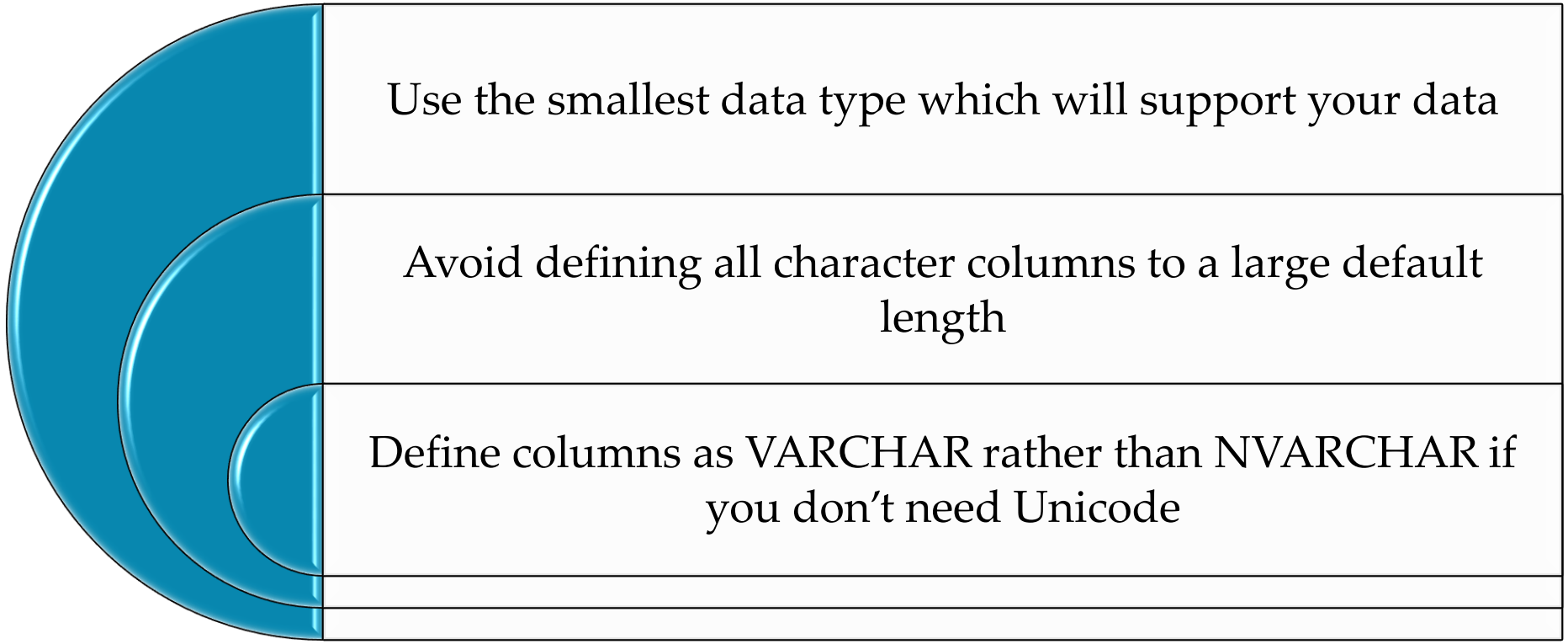
Good Hash Key



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

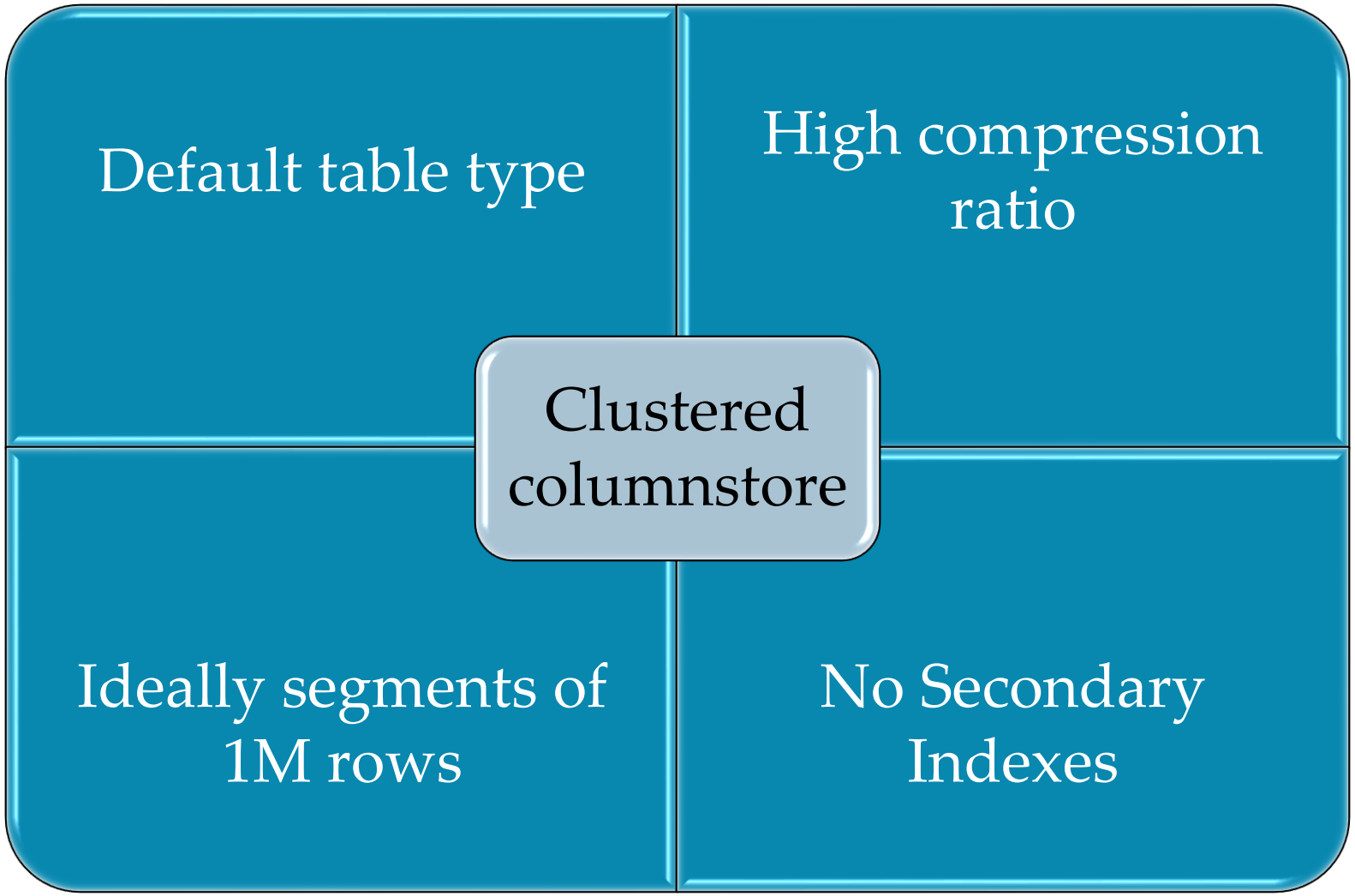
The goal is to not only save space but also move data as efficiently as possible.

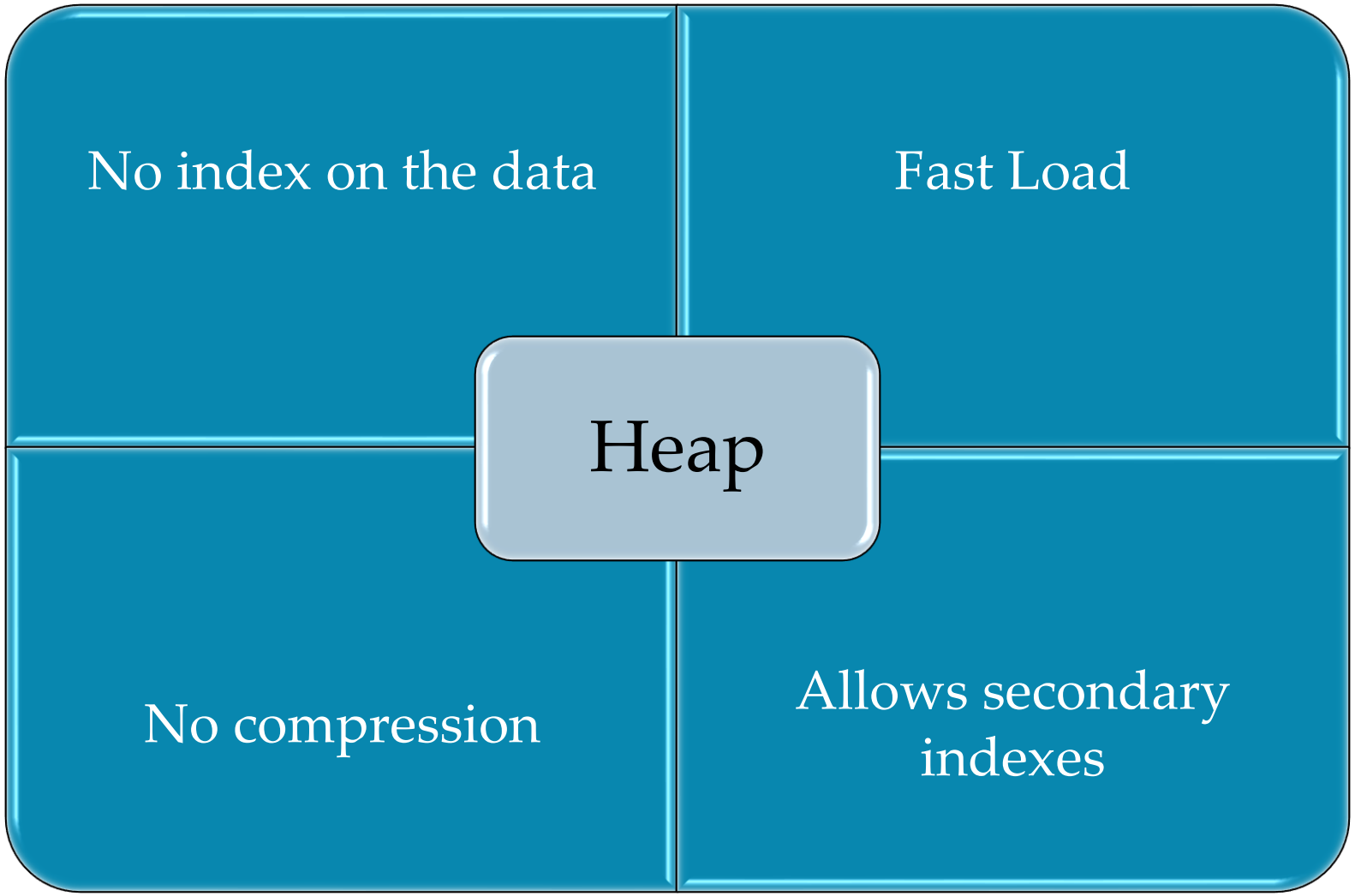
Data types

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

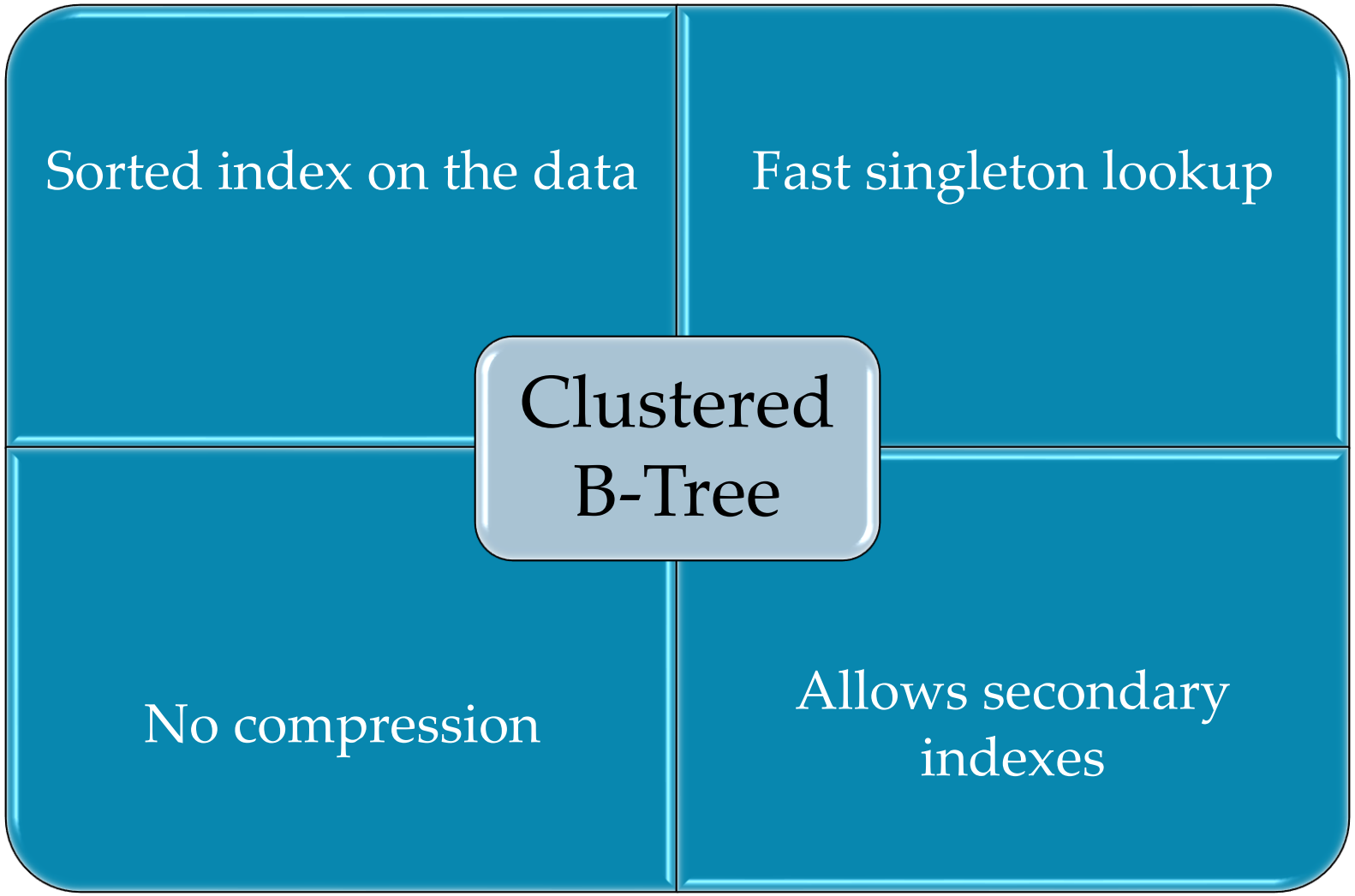
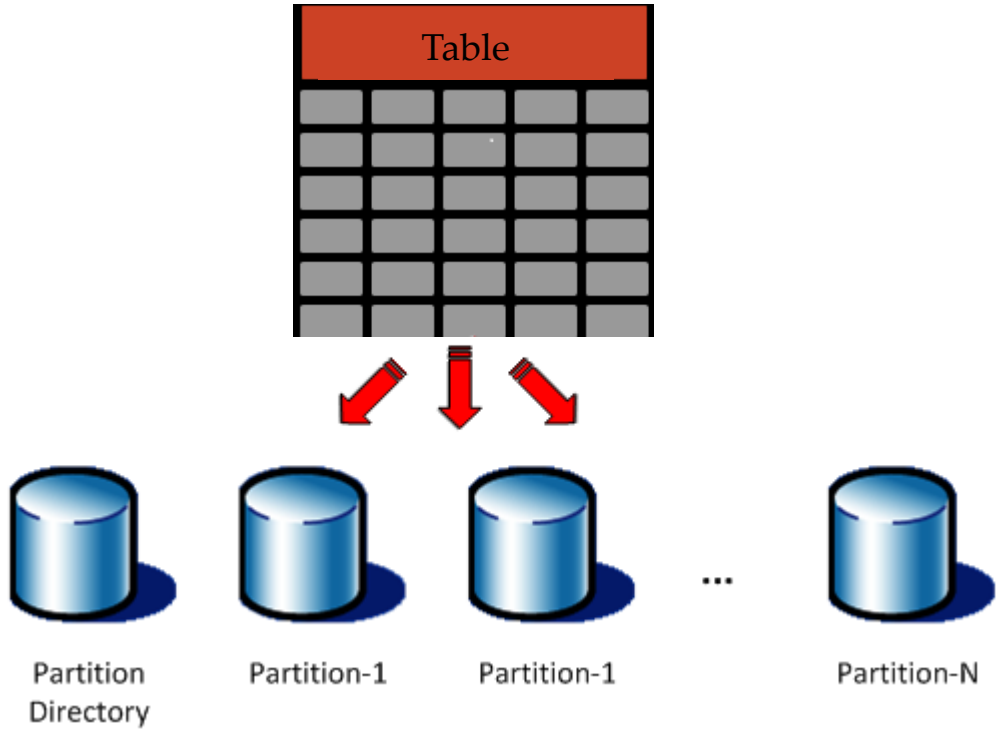
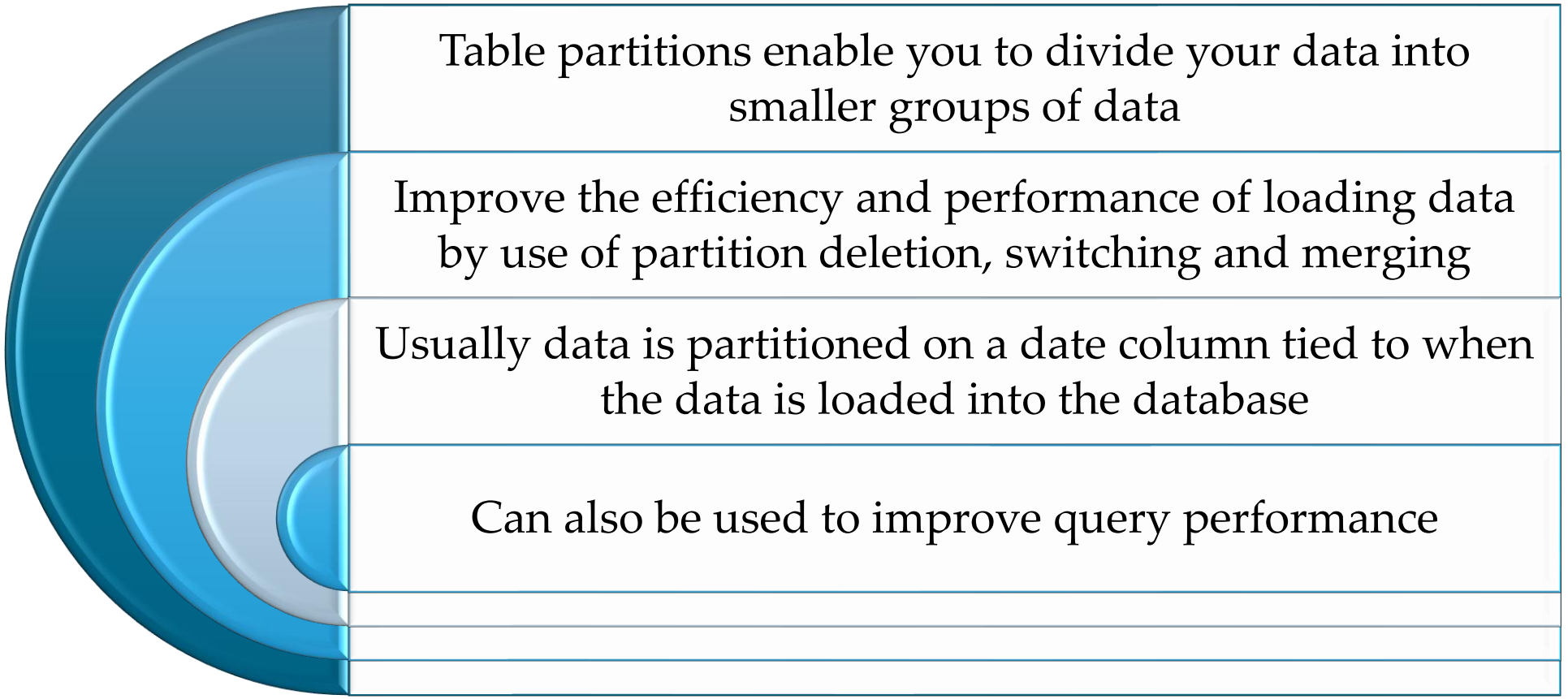

Sorted index on the data
Fast singleton lookup
Clustered B-Tree
No compression
Allows secondary indexes

Table Partitioning

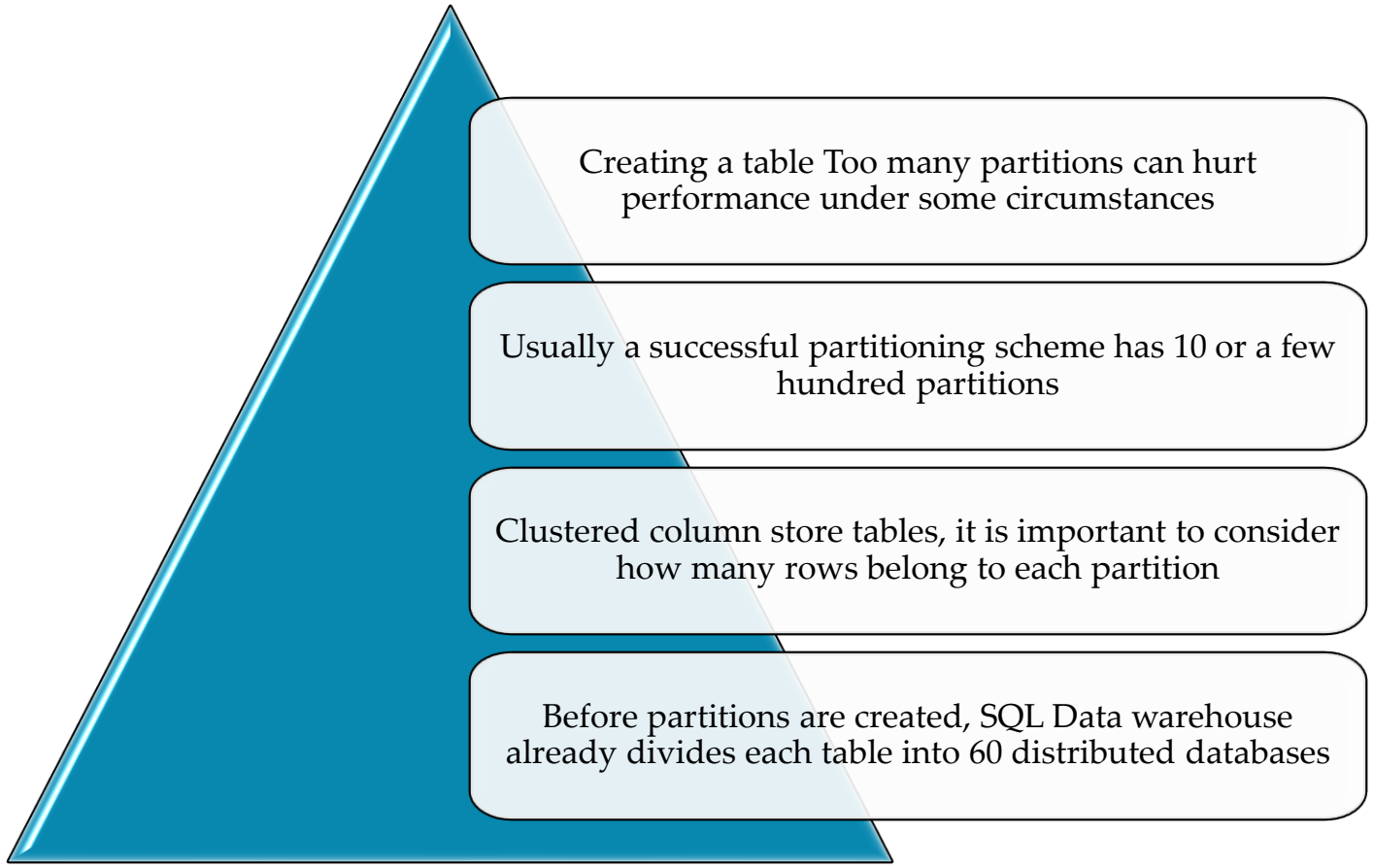


Partitioning



Why Partitioning?

Partitions best practices



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

Example

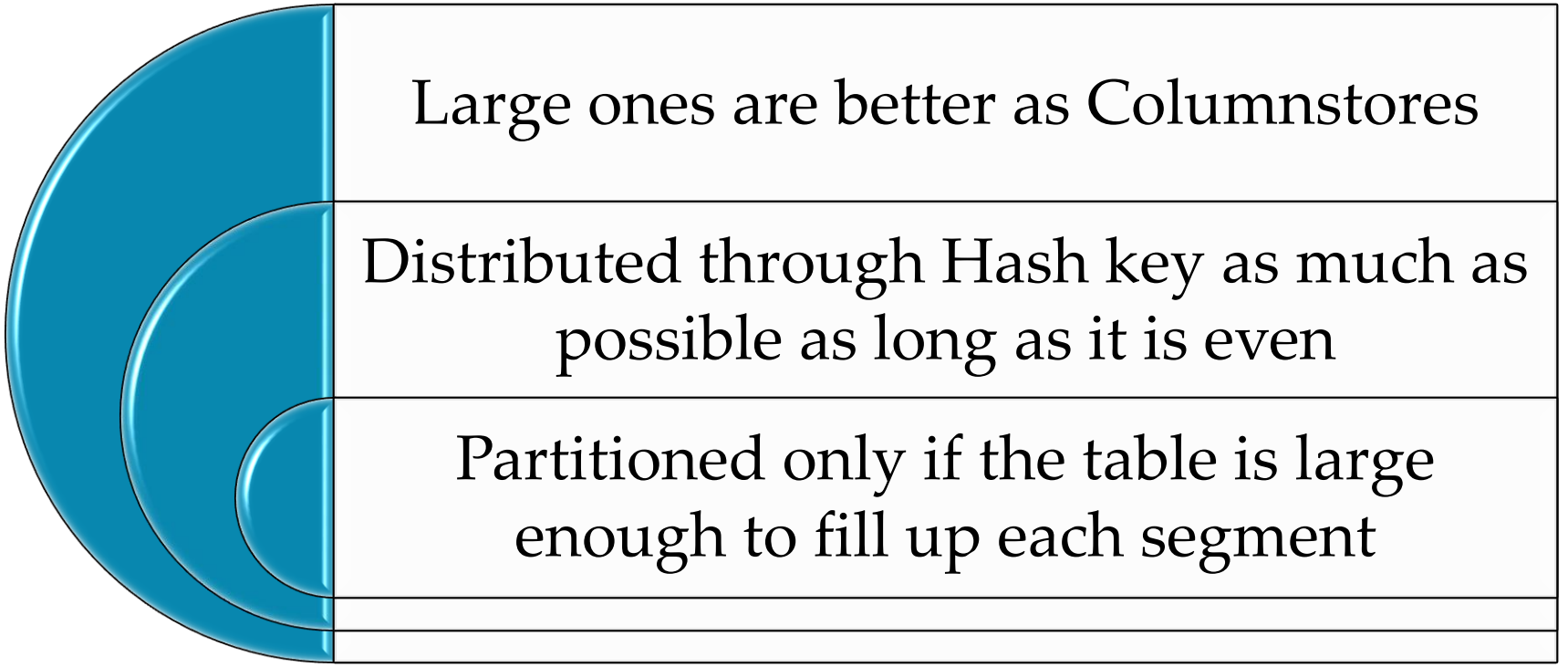
60 Distributions  365 Partitions  21900 Data Buckets

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

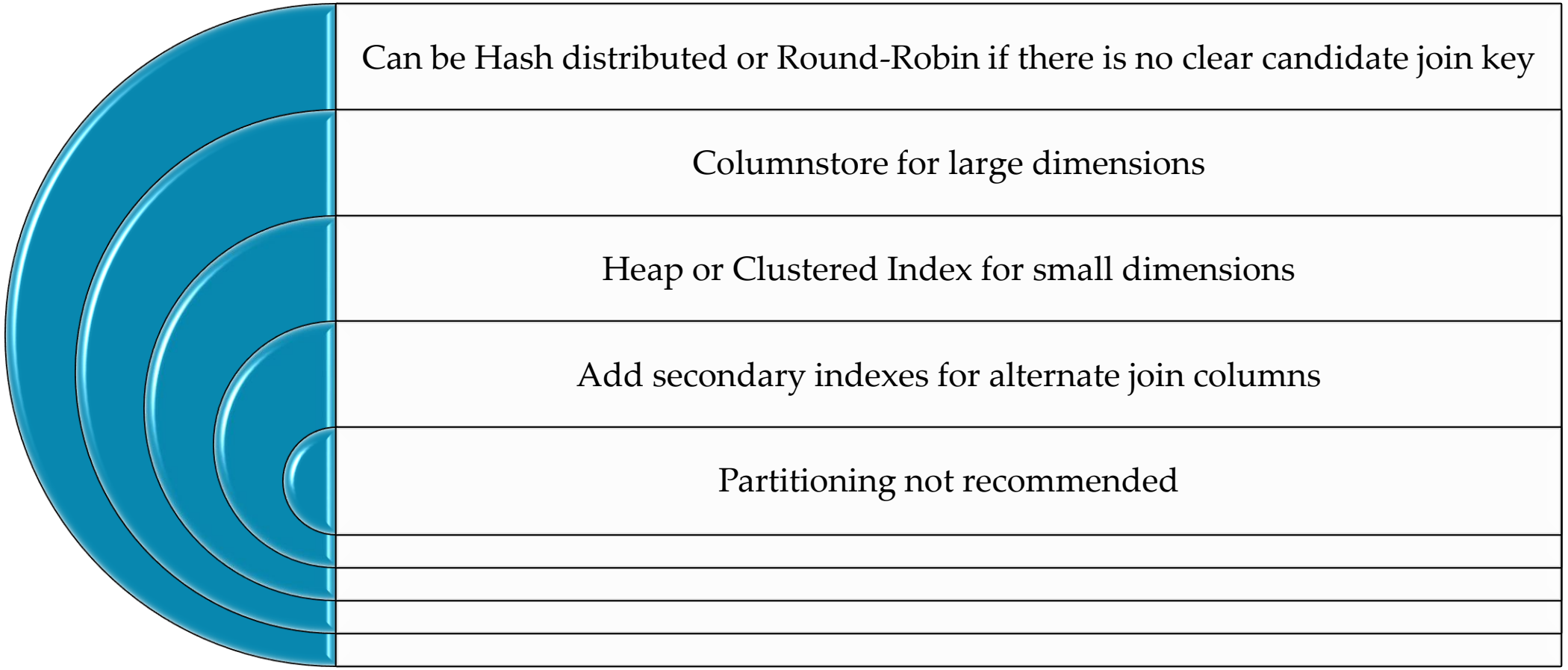
Lower Granularity (week, month) can perform better depending on how much data you have.

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



Best Practices for Data Laod

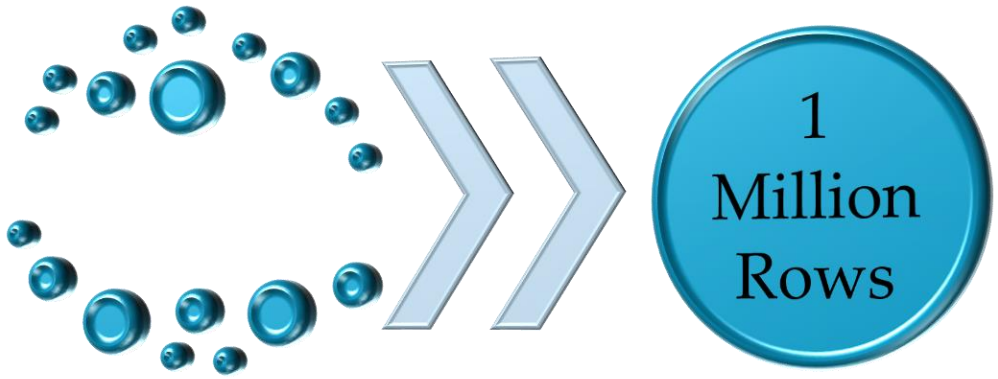
Data Warehouse Readers

Your DWUs have a direct impact on how fast you can load data in parallel

No of DWU	**100**	**200**	**300**	**400**	**500**	**600**	**1000**	**1500**	**2000**
Readers	8	16	24	32	40	48	60	60	60
Writers	60	60	60	60	60	60	60	60	60

Optimize Insert Batch Size

Avoid trickle insert pattern. Ideal batch size is 1 million or more direct or in a file

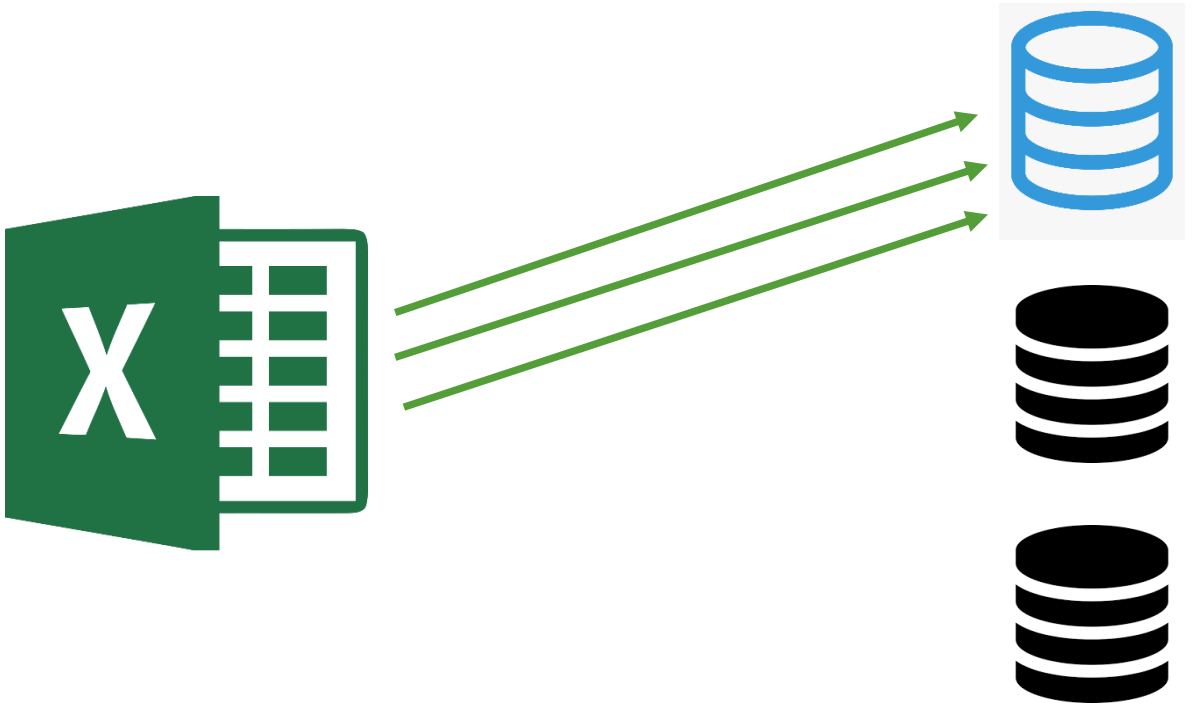


Combine rows to make it a batch of 1 million

Ideal batch size

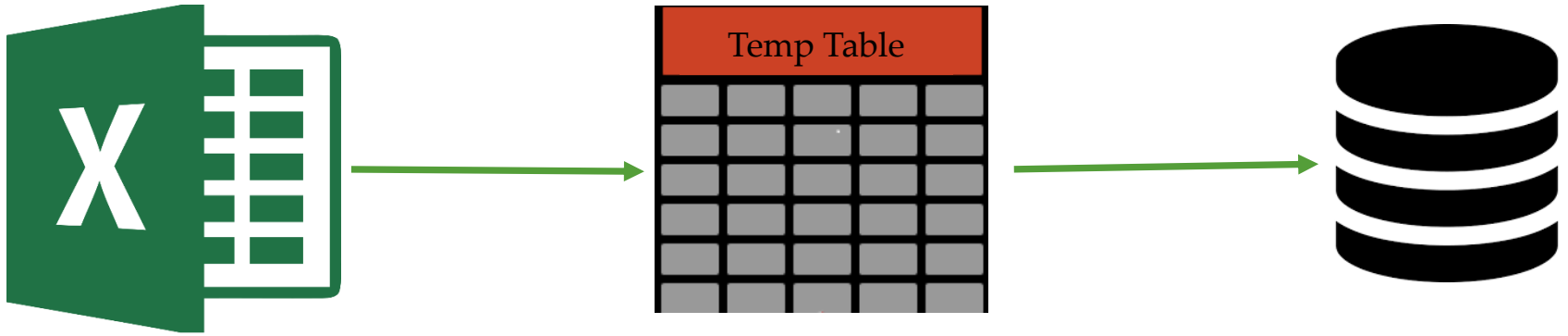
Avoid ordered data

Data ordered by distribution key can introduce hot spots that slow down the load operation



Using temporary tables

Stage and transform on a Temp Heap table before moving to permanent storage.



CREATE TABLE AS

```
CREATE TABLE #tmp_fct
WITH
(
DISTRIBUTION = ROUND_ROBIN
)
AS
SELECT *
FROM
[dbo].[FactInternetSales];
```

- Fully Parallel operation
- It is minimally logged
- It can change: distribution, table type, partitioning

Loading Methods

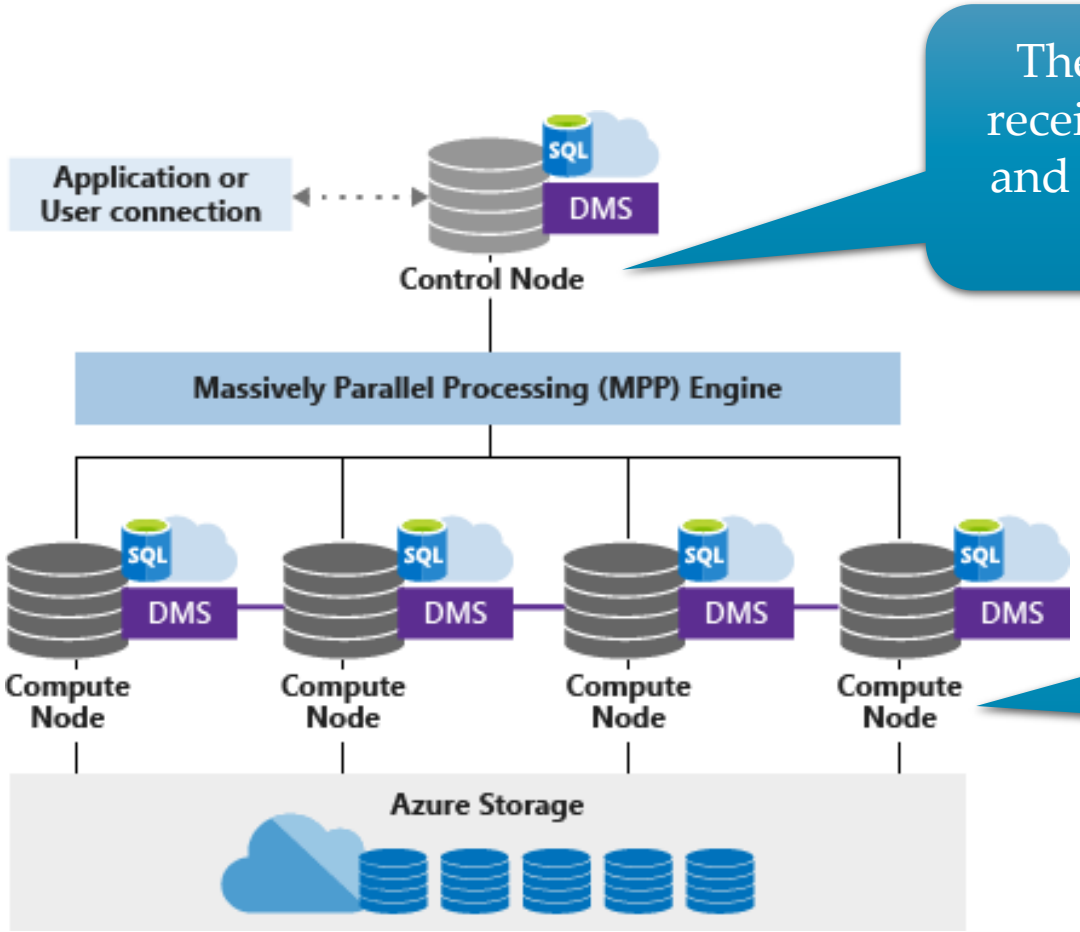
Single client loading methods

- SSIS
- Azure Data Factory
- BCP
- Can add some parallel capabilities but are bottlenecked at the control node

Parallel readers loading methods

- PolyBase
- Reads from Azure blob Storage and loads the contents into Azure SQL DW
- Bypasses the Control Node and loads directly into the Compute Nodes

Control Node

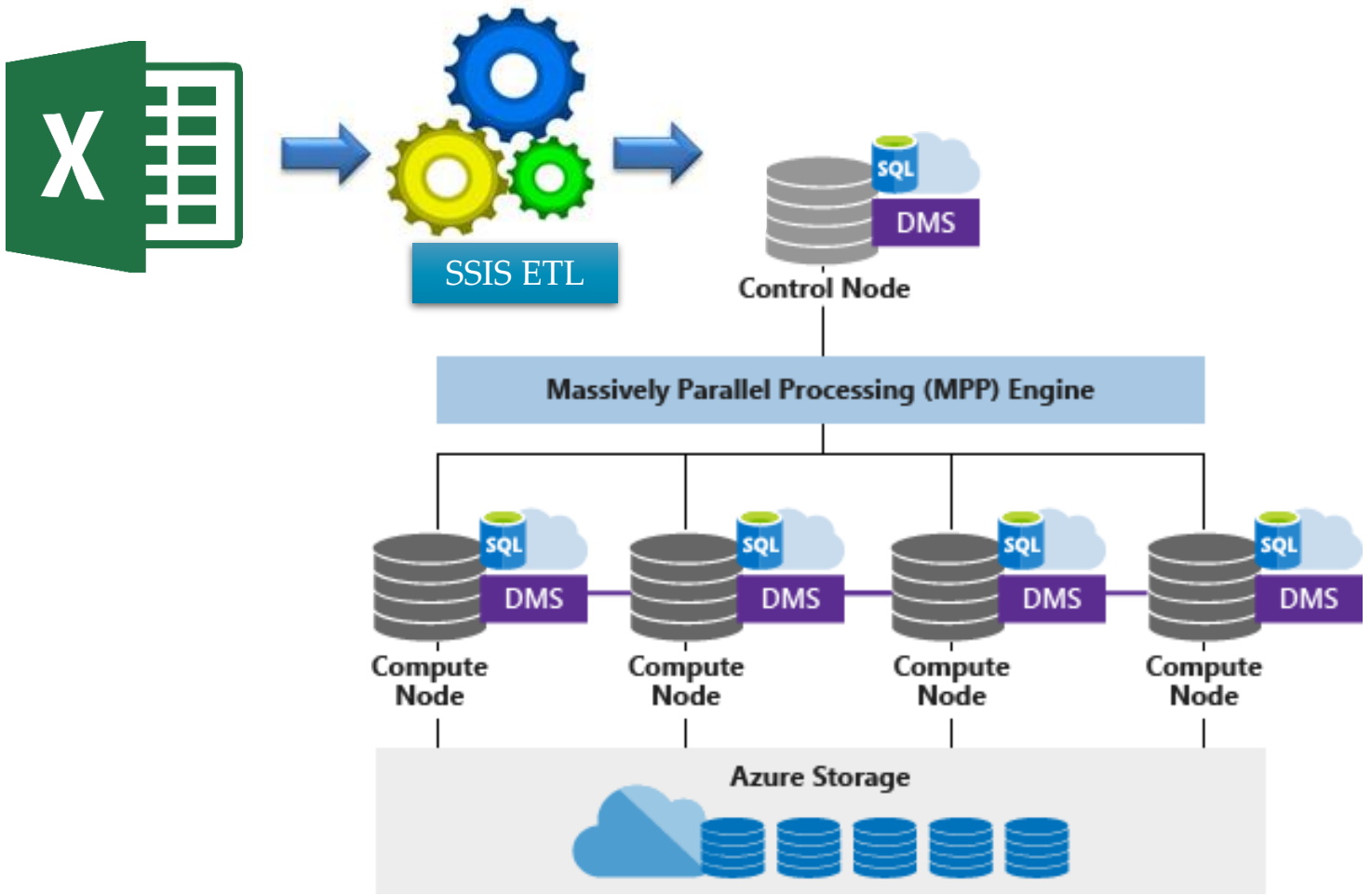


The Control node receives connections and orchestrates the queries

The Compute nodes do processing on the data and scale with the DWUs.

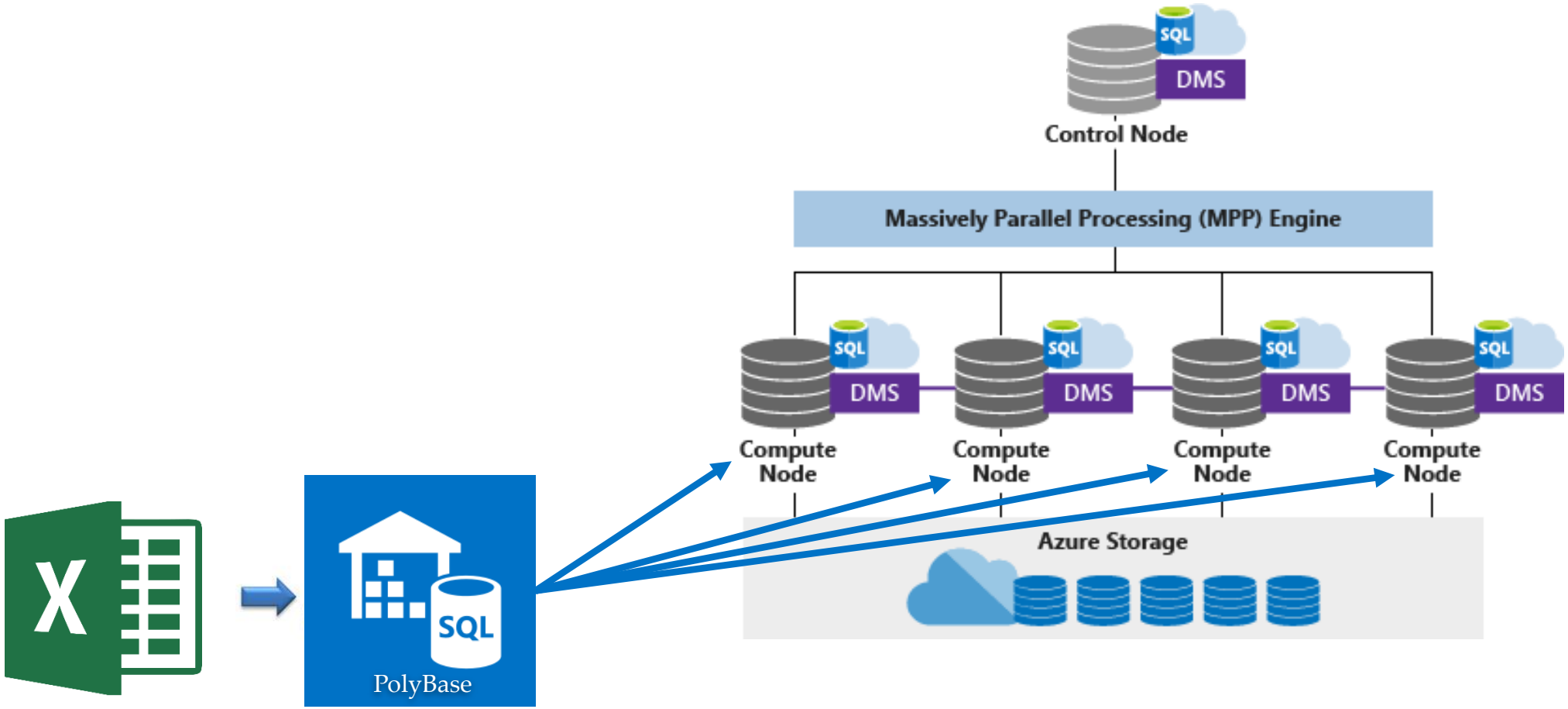
Source: Microsoft

Loading with SSIS



Source: Microsoft

Loading with PolyBase

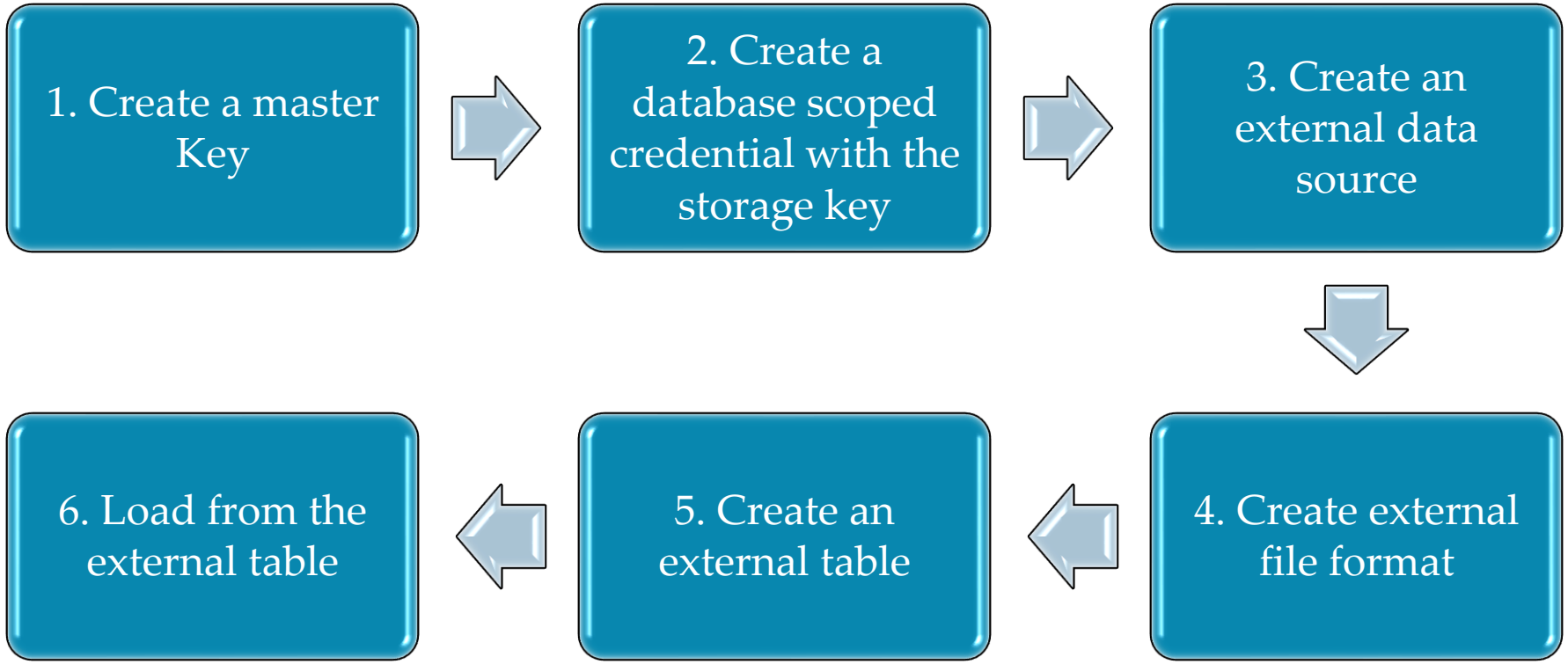


Source: Microsoft

PolyBase can load data from UTF-8 delimited text files and popular Hadoop file formats like RC File, ORC, and Parquet

Multiple readers will not work against a compressed file

PolyBase Setup



DEMO

Load Dimension table from on-premises to Azure Synapse Data warehouse using SSIS

Demo – PolyBase

1. Export table to flat file
2. Create blob storage account
3. Upload flat file to blob storage
4. Run PolyBase 6 steps process
5. Monitor and confirm successful migration
6. Confirm 60 distributions in destination table

Azure Synapse Analytics