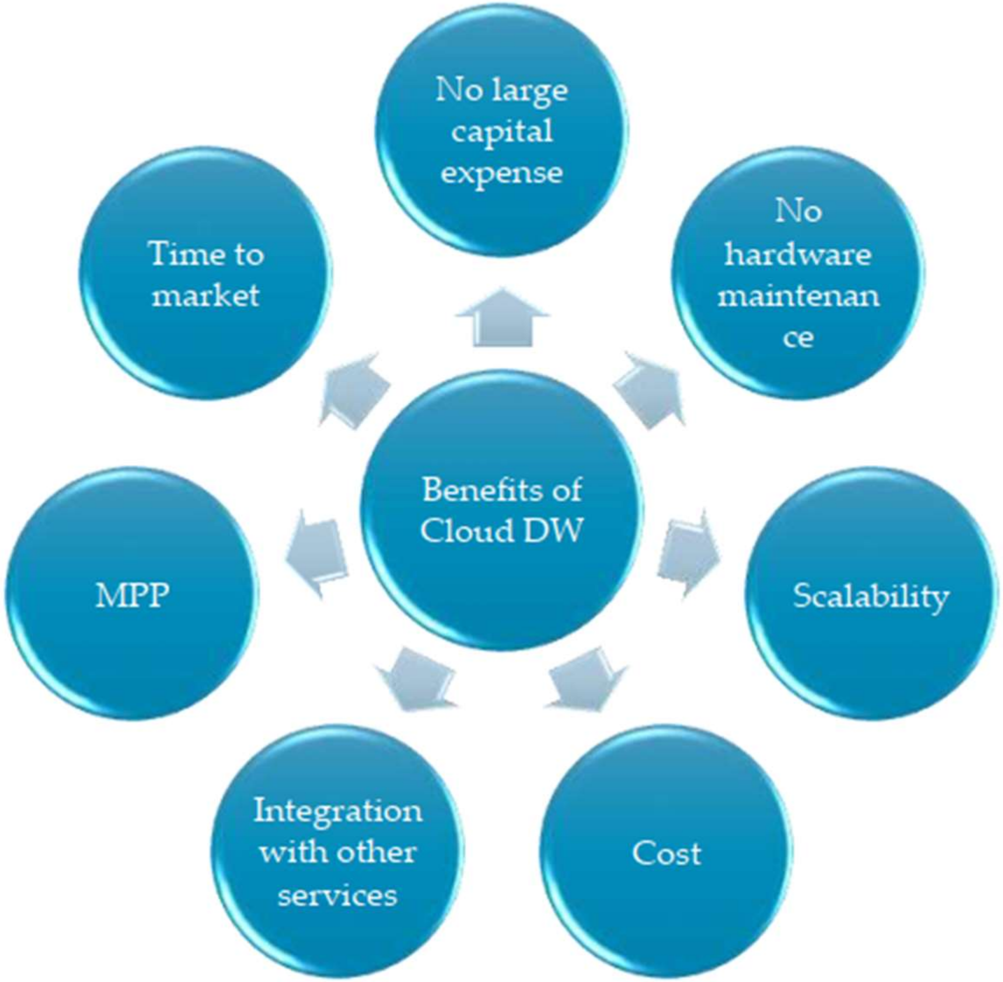
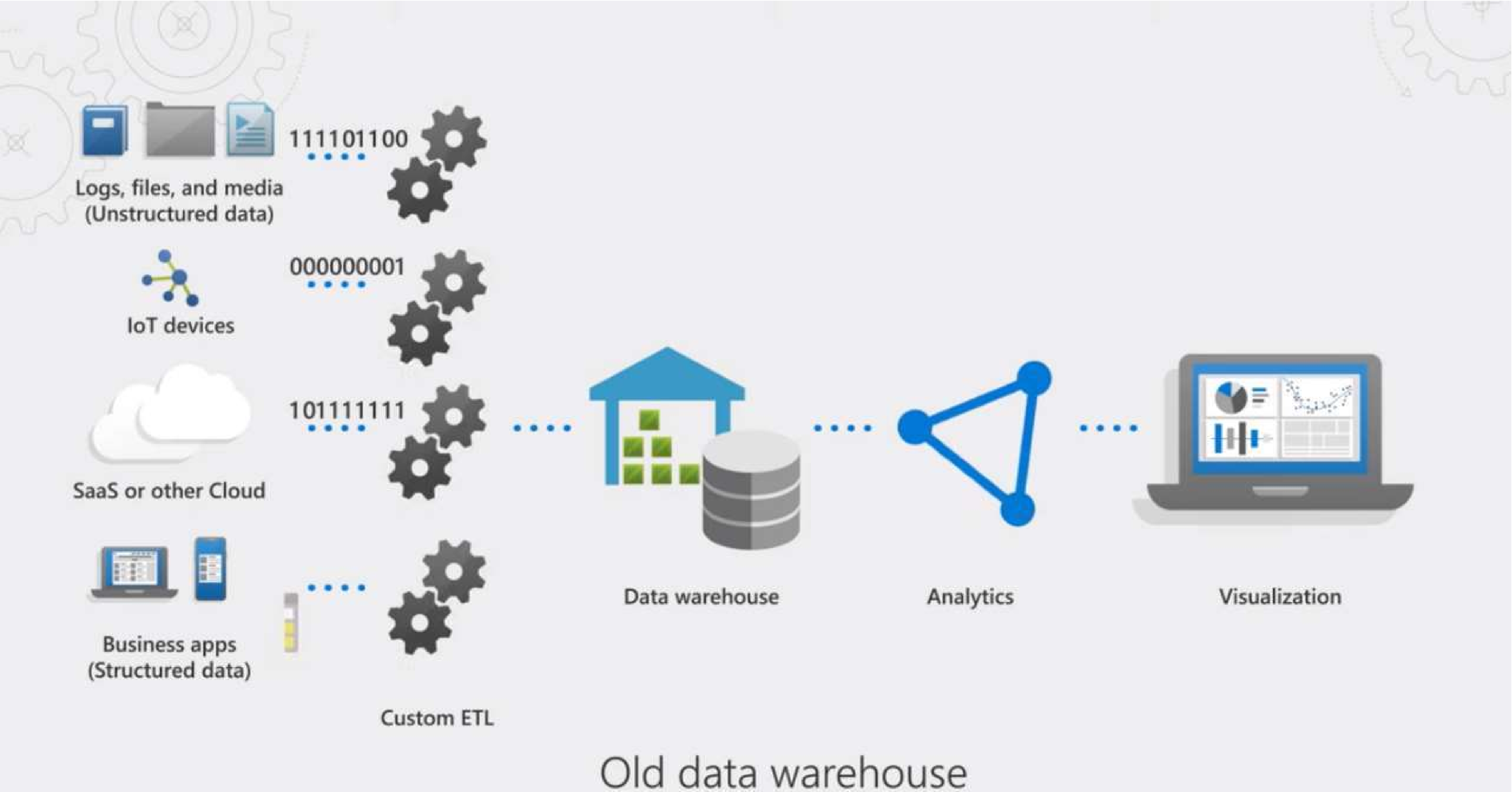
Azure Synapse Analytics

Why Warehousing in Cloud?





Logs, files, and media
(Unstructured data)

IoT devices

SaaS or other Cloud

Business apps
(Structured data)

Custom ETL

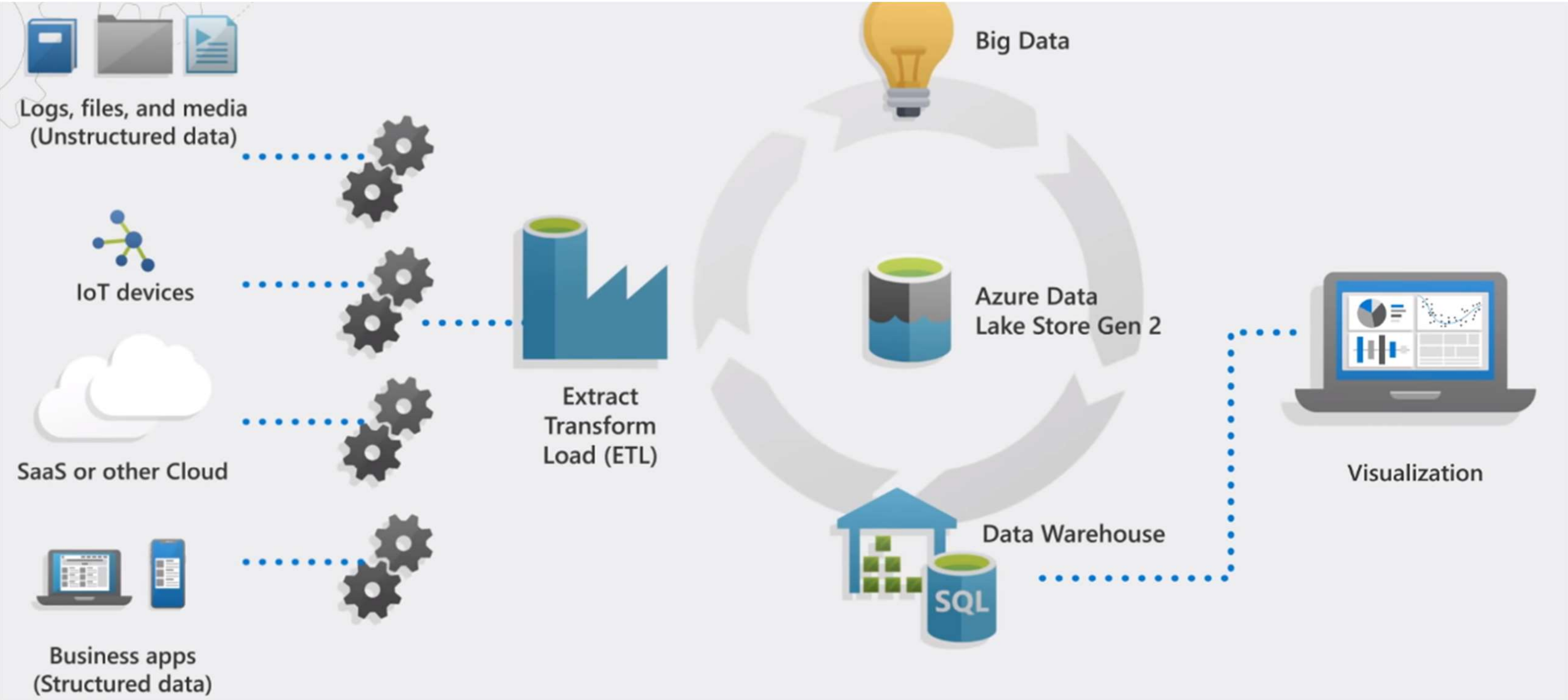
Data warehouse

Analytics

Visualization

Old data warehouse

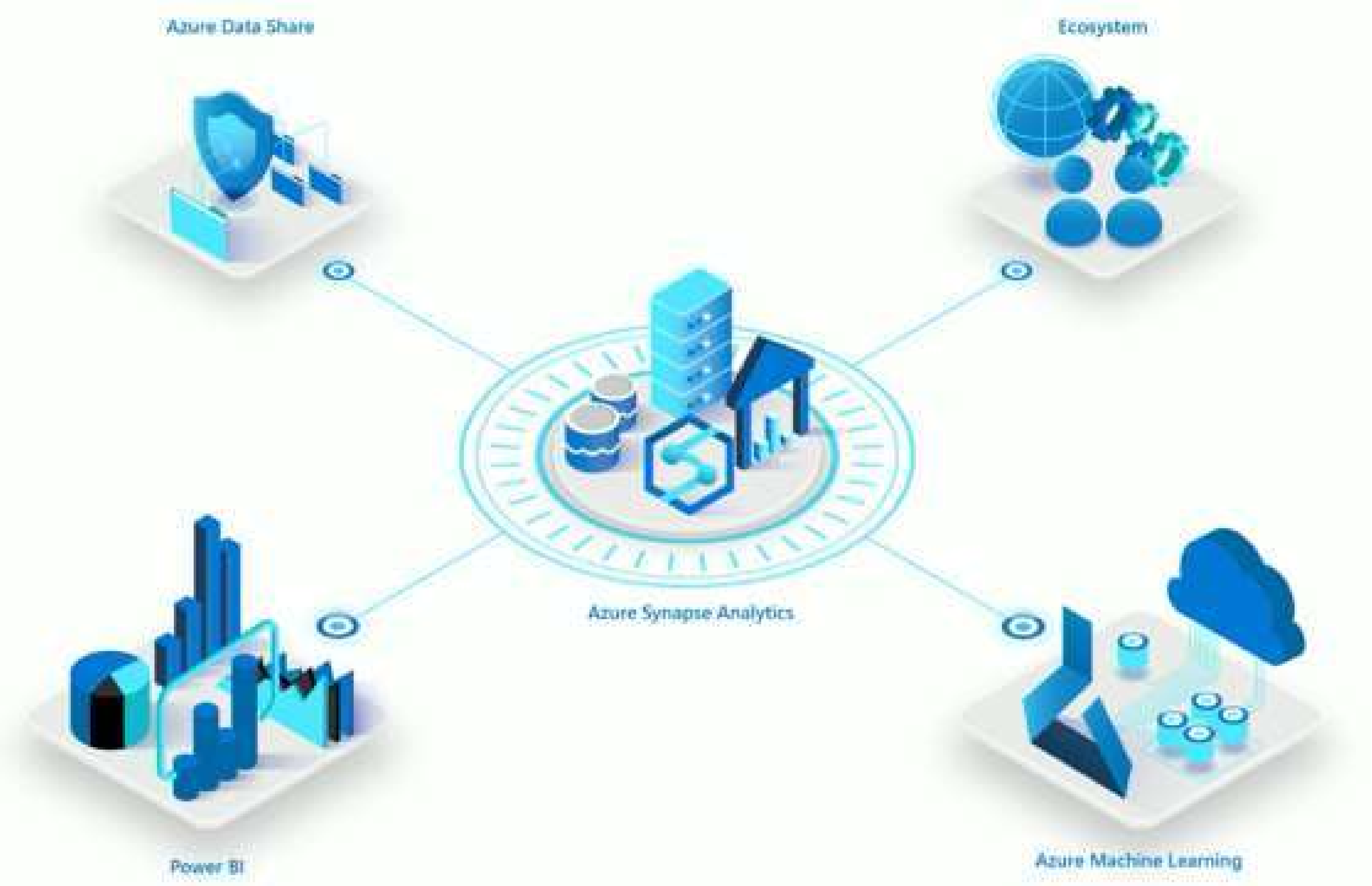
Modern Data Warehouse



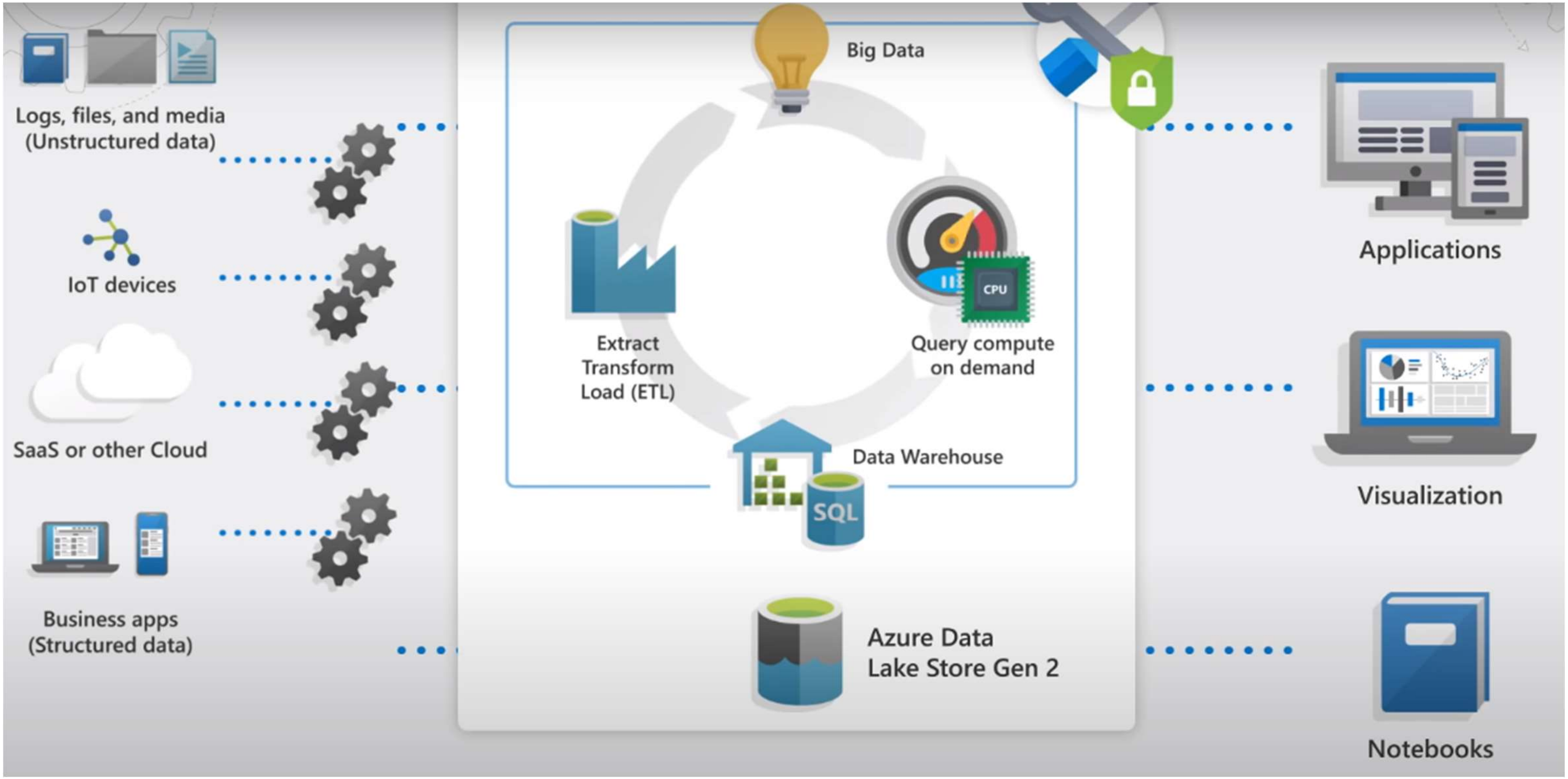
Azure Synapse Analytics

- Next generation of Azure SQL Data Warehouse
- Blending into a single unified service
  - Big data analytics
  - Data warehousing and
  - Data integration
- Provides end-to-end analytics with limitless scale.

Azure Synapse Analytics

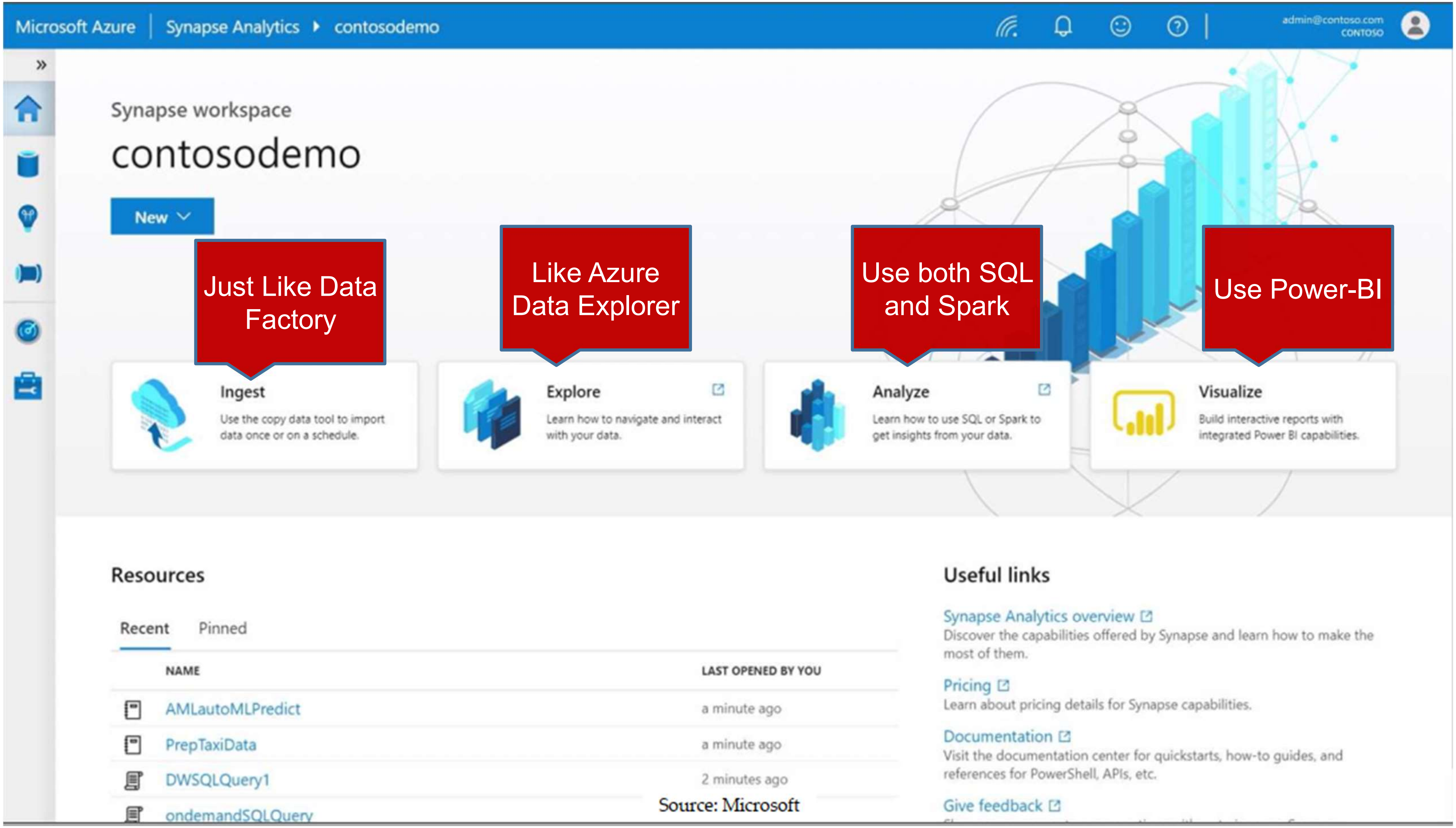


Azure Synapse Analytics



Hands-on 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



Microsoft Azure | Synapse Analytics ▸ contosodemo

admin@contoso.com
CONTOSO

Synapse workspace

contosodemo

New ∨

Just Like Data Factory

Like Azure Data Explorer

Use both SQL and Spark

Use Power-BI

Ingest
Use the copy data tool to import data once or on a schedule.

Explore
Learn how to navigate and interact with your data.

Analyze
Learn how to use SQL or Spark to get insights from your data.

Visualize
Build interactive reports with integrated Power BI capabilities.

Resources

Recent | Pinned

NAME	LAST OPENED BY YOU
AMLautoMLPredict	a minute ago
PrepTaxiData	a minute ago
DWSQLQuery1	2 minutes ago
ondemandSQLQuery	

Useful links

Synapse Analytics overview
Discover the capabilities offered by Synapse and learn how to make the most of them.

Pricing
Learn about pricing details for Synapse capabilities.

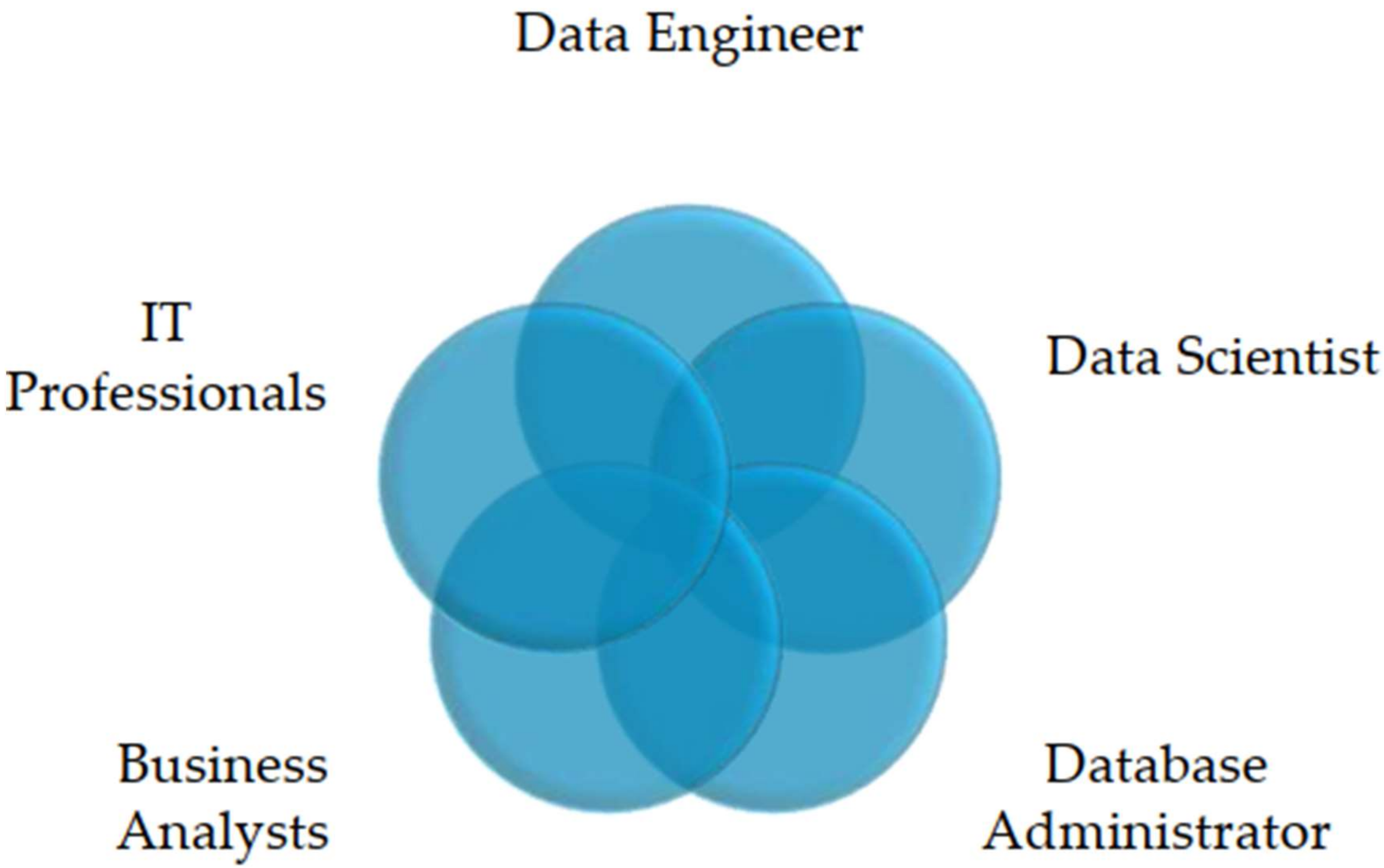
Documentation
Visit the documentation center for quickstarts, how-to guides, and references for PowerShell, APIs, etc.

Give feedback

Source: Microsoft

Azure Synapse Analytics

- Unified experience for all data professionals



Azure Synapse Analytics architecture

1. Applications

- Connect to issue T-SQL commands
- Single point of entry for Synapse SQL

2. Control node

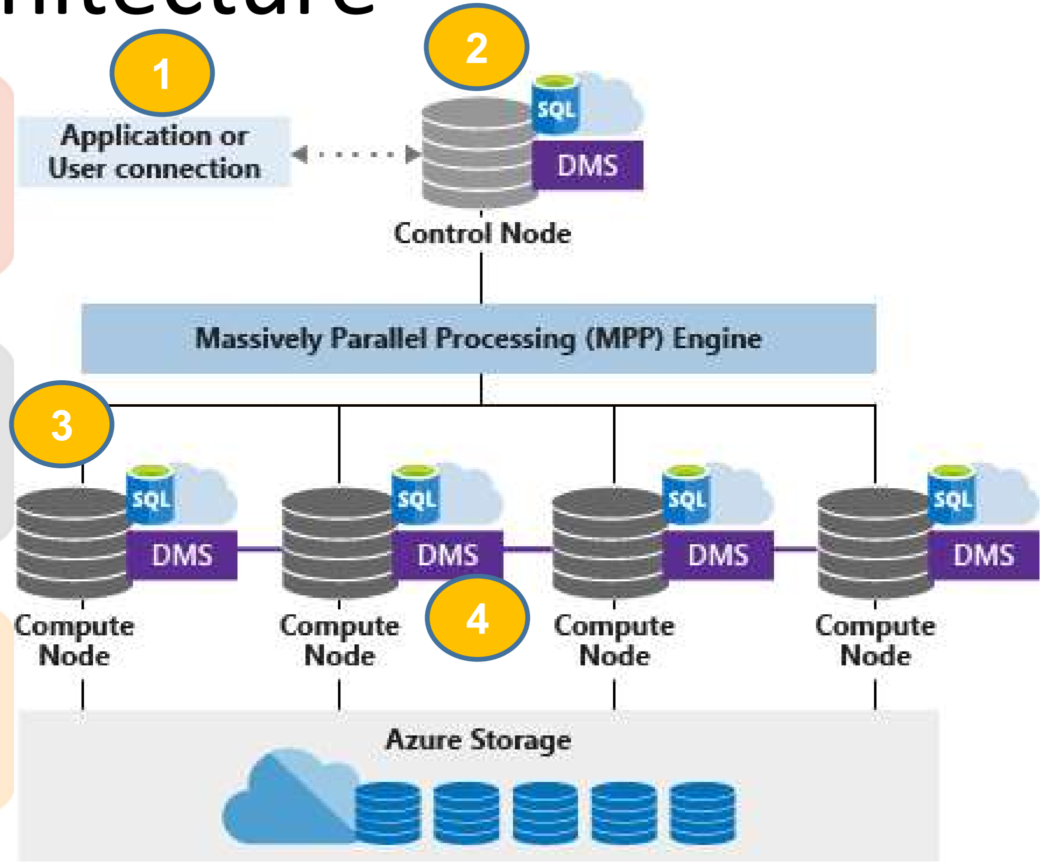
- Runs the MPP engine
- Optimizes queries for parallel processing
- Passes operations to Compute nodes

3. Compute nodes

- Store all user data in Azure Storage
- Run the parallel queries.

4. DMS

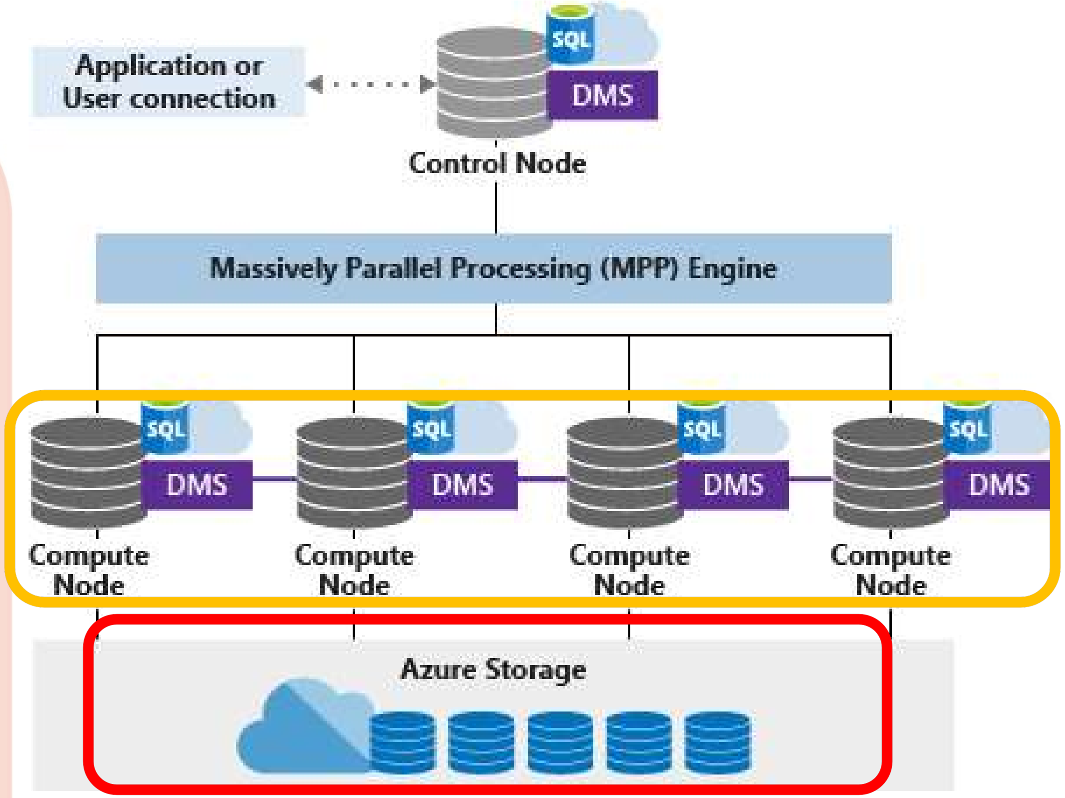
- Moves data across the nodes
- To run queries in parallel



Azure Synapse Analytics architecture

Decoupled storage & compute

- Independently size compute power irrespective of your storage needs.
- Grow or shrink compute power without moving data.
- Pause compute capacity so you only pay for storage
- Resume compute capacity



Azure Storage

To keep your user data safe

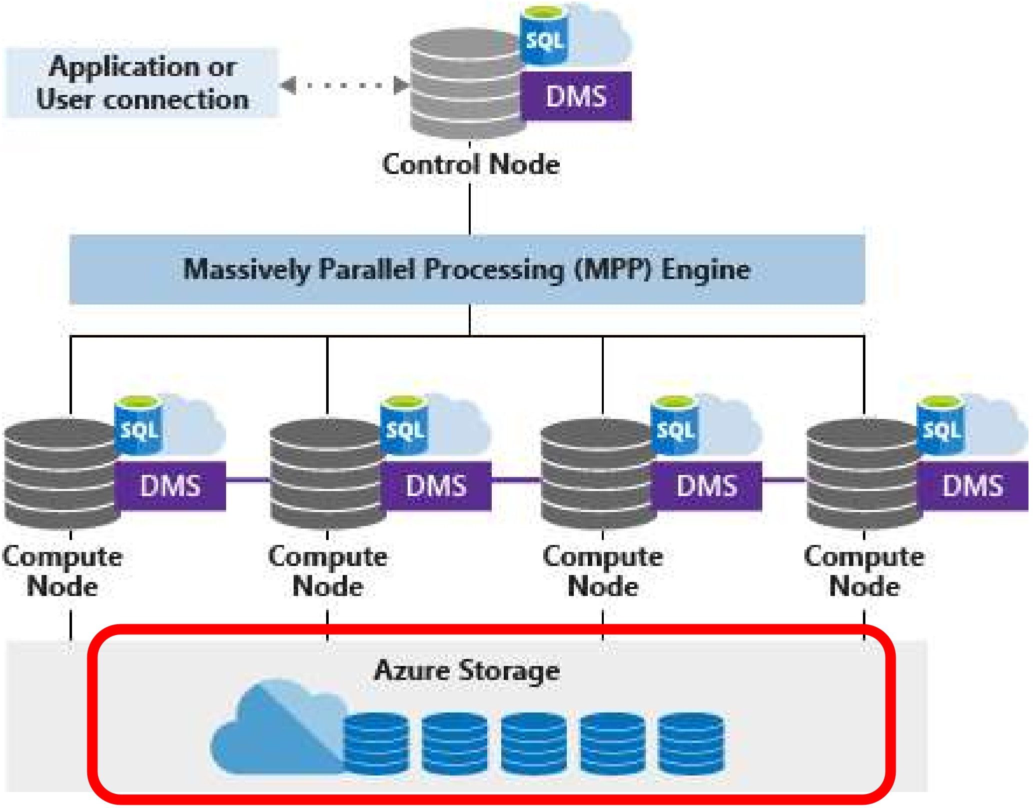
Separate charge for storage

Data is sharded to optimize the performance

Can choose sharding pattern

Supported sharding patterns:

- Hash
- Round Robin
- Replicate



Control node

Brain of the architecture.

It’s the front end

Interacts

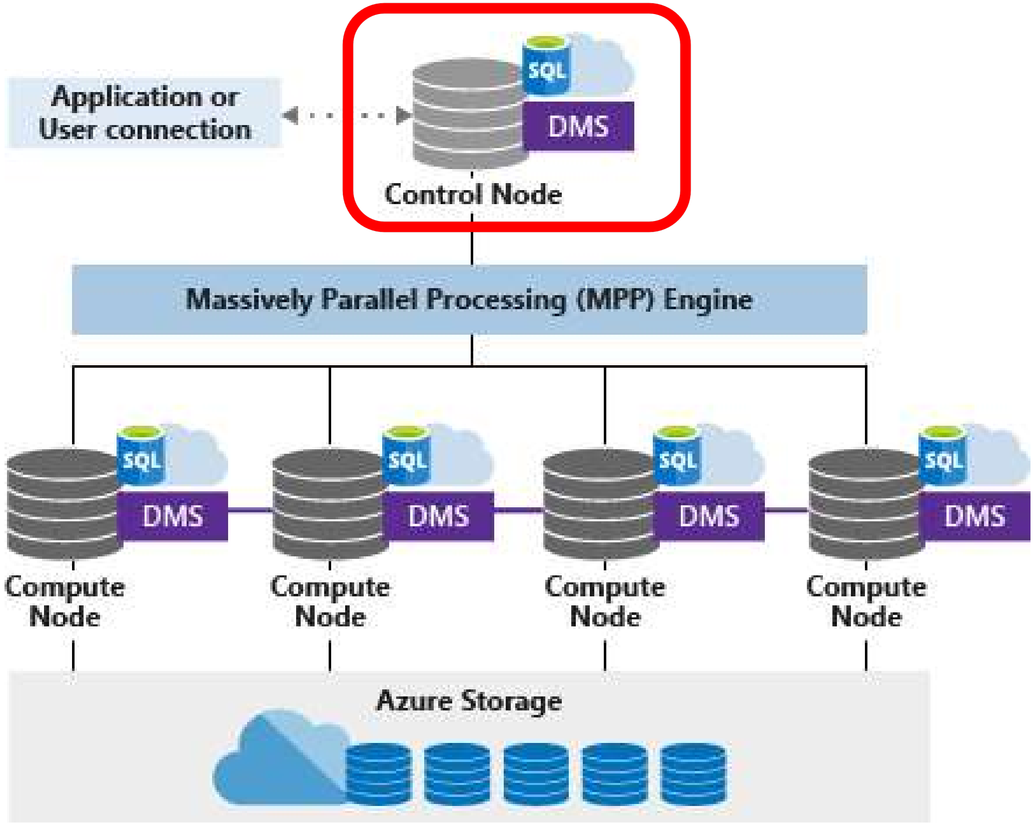
- With all applications and connections.

MPP engine runs on Control node

- To optimize and coordinate parallel queries.

When T-SQL query is submitted

- Control node transforms it into queries that run against each distribution in parallel.



Distributions

Synapse SQL runs query

- The work is divided into 60 smaller queries that run in parallel.

Each of the 60 smaller queries

- Runs on one of the underlying data distribution.

Distribution

- The basic unit of storage and
- Processing for parallel queries that run on distributed data.

Compute nodes

Provide computational power

“Distributions”

- Map to Compute nodes for processing

More compute resources

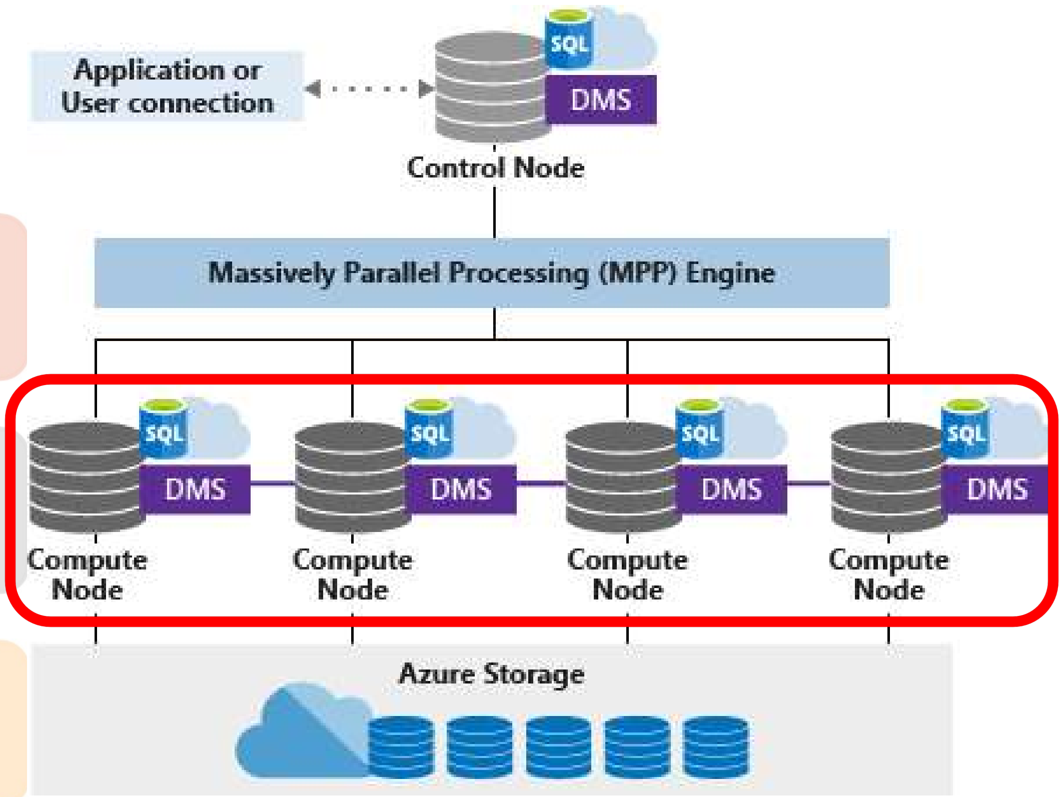
- “Distributions” are remapped to available Compute nodes

The number

- Ranges from 1 to 60

Each Compute node

- Has a node ID



Data Movement Service

Data transport technology

Coordinates

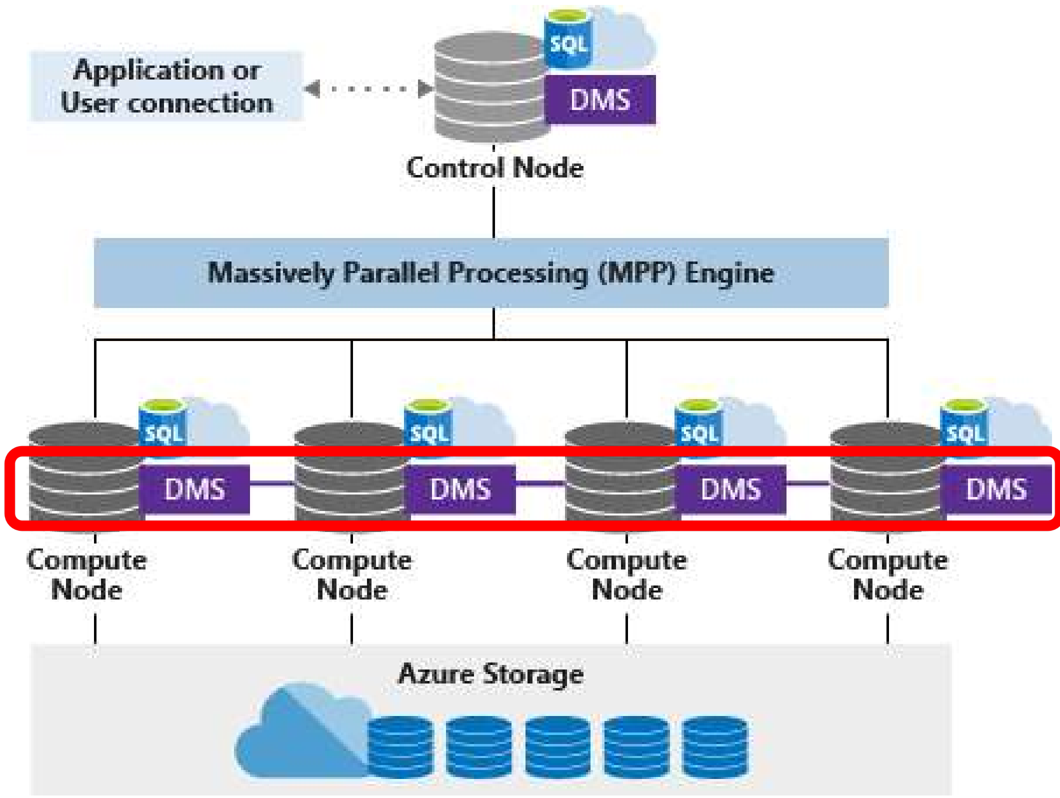
- Data movement between the Compute nodes

Require data movement

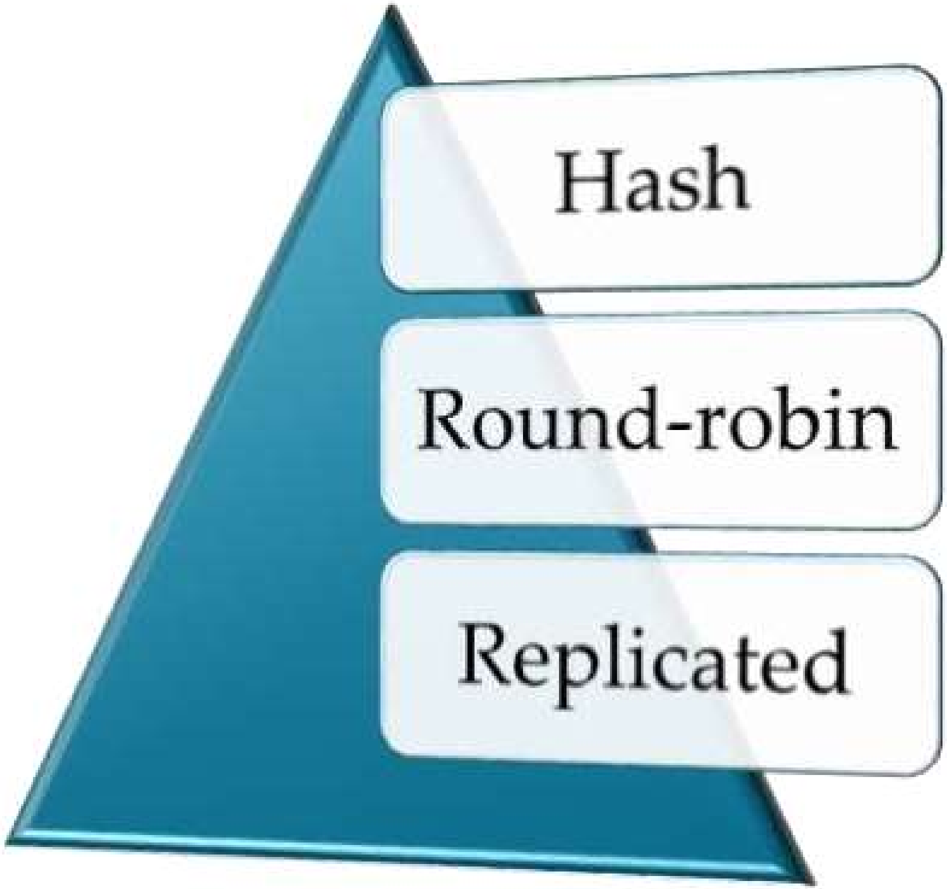
- Some queries require data movement to ensure the parallel queries return accurate results

DMS ensures

- When data movement is required, DMS ensures the right data gets to the right location.



Sharding Patterns



Hash

Round-robin

Replicated

Hash-distributed tables

Can deliver

- Highest query performance for joins and aggregations

How to shard data

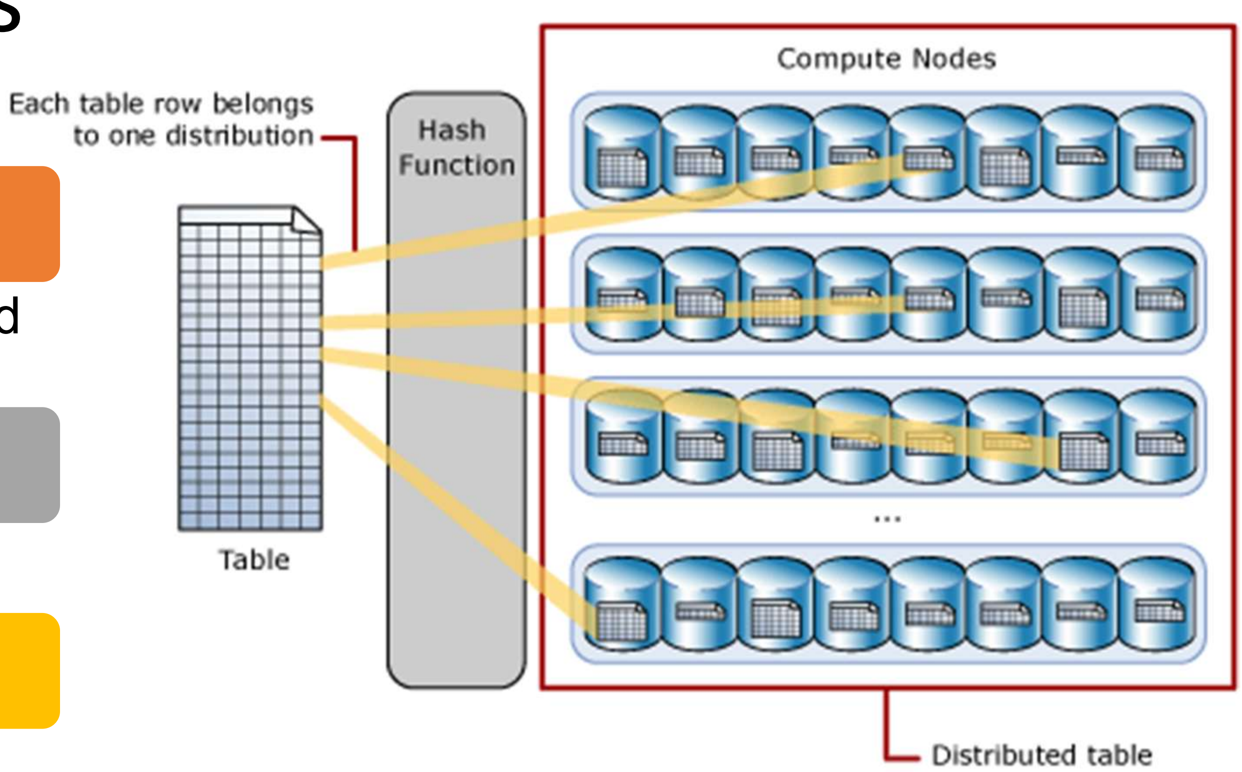
- A hash function is used

Distribution column

- One of the columns is designated

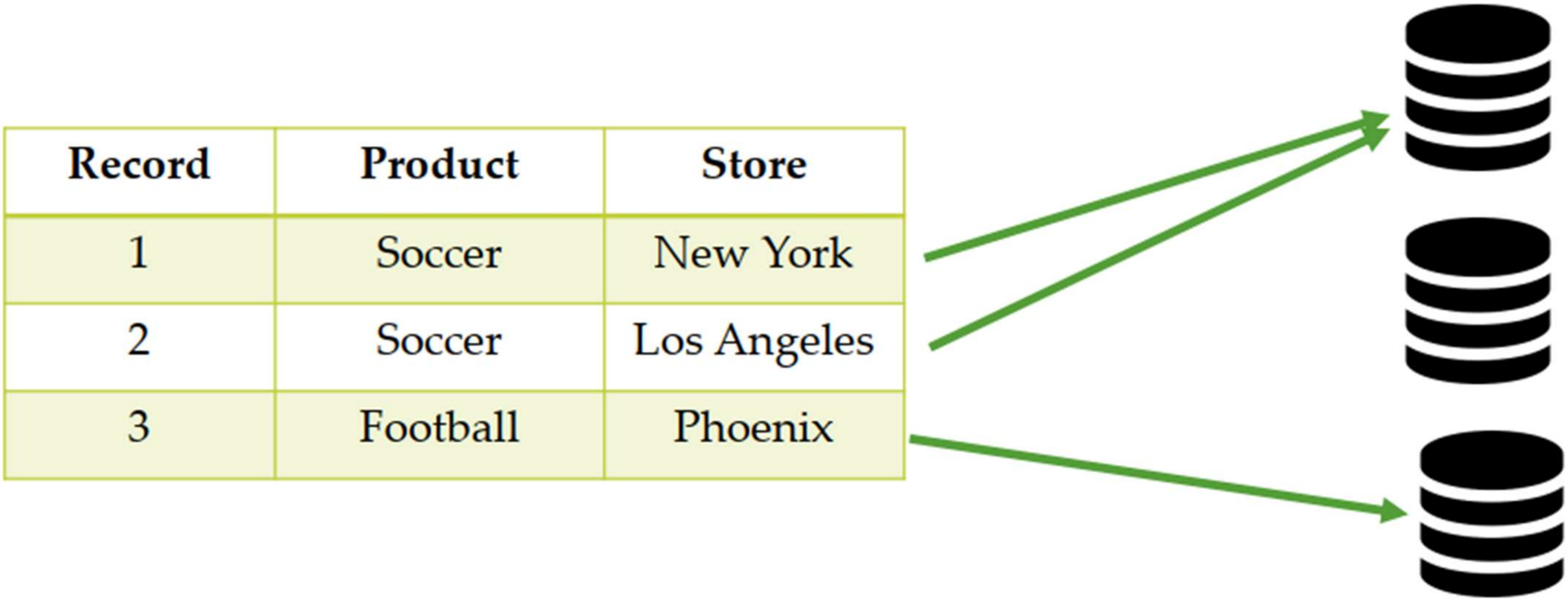
Uses values in distribution colum

- To assign each row to a distribution.



- Each row belongs to one distribution.
- Hash algorithm assigns each row to one distribution.

Hash-distributed tables



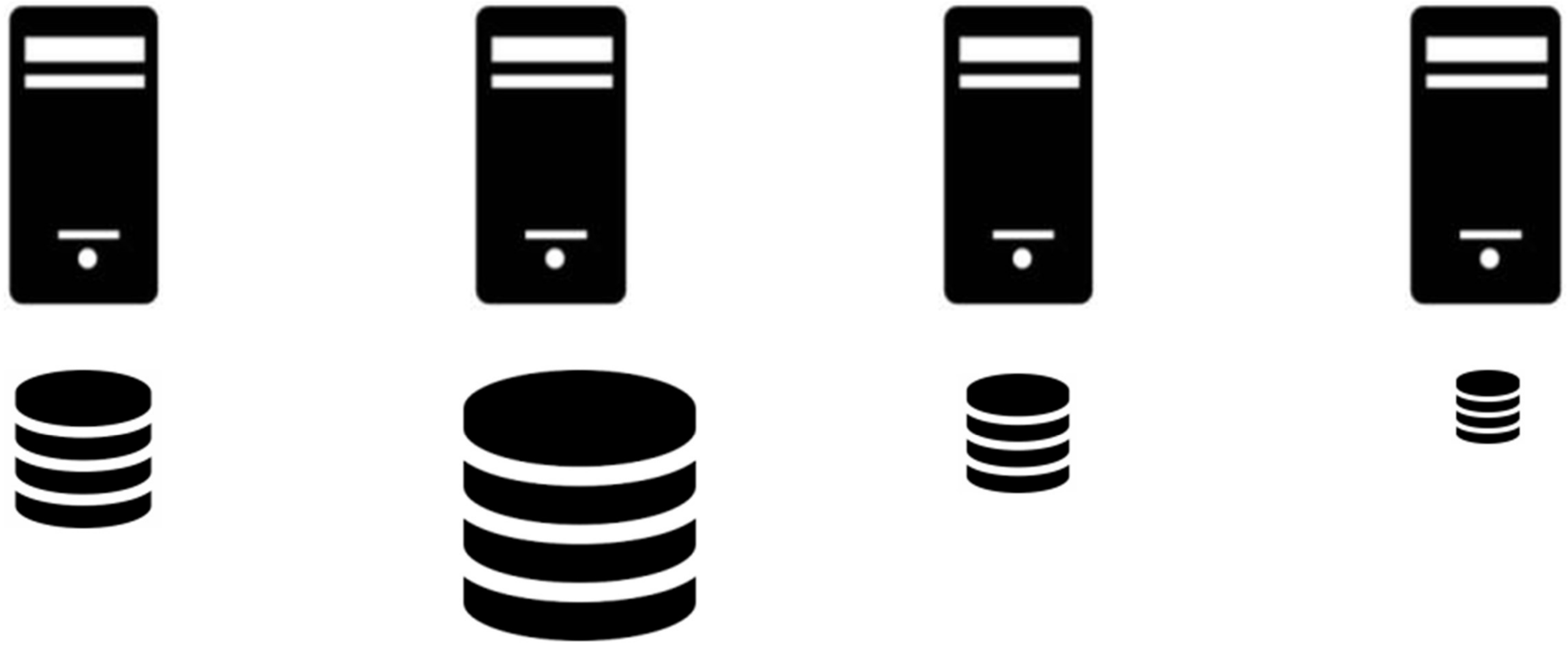
Record	Product	Store
1	Soccer	New York
2	Soccer	Los Angeles
3	Football	Phoenix

Hash-distributed tables

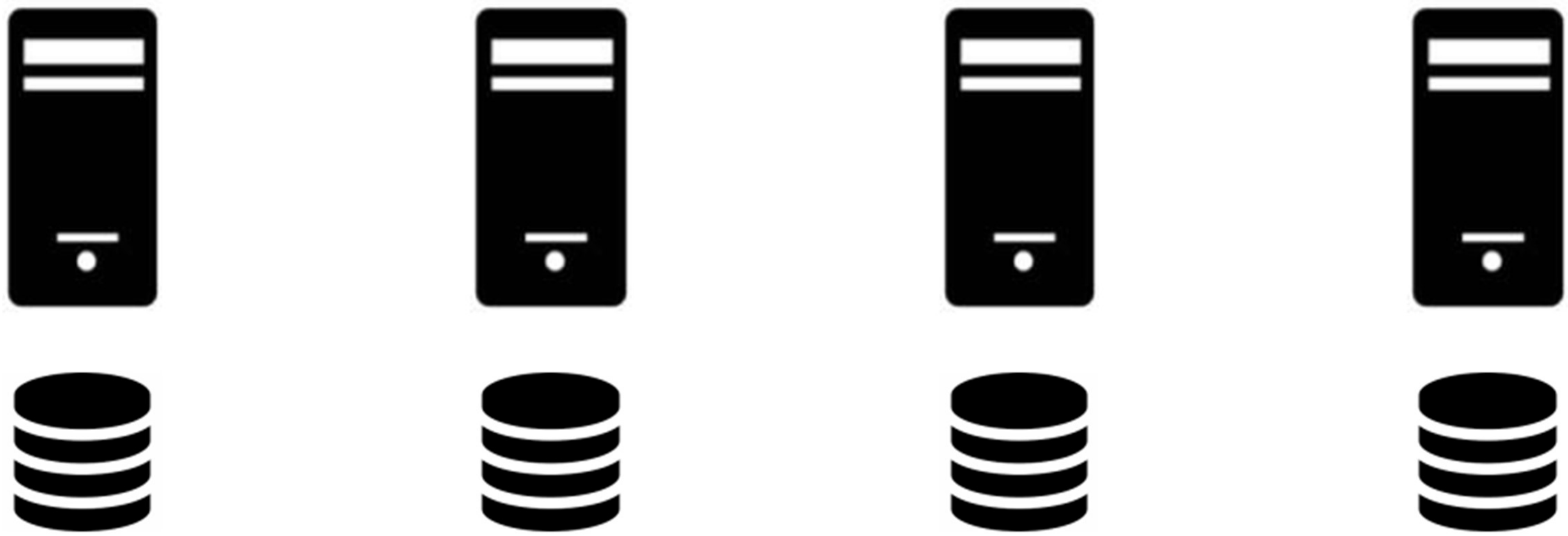
```
CREATE TABLE [dbo].[EquityTimeSeriesData](
[Date] [varchar](30) ,
[BookId] [decimal](38, 0) ,
[P&L] [decimal](31, 7) ,
[VaRLower] [decimal](31, 7)
)
WITH
(
CLUSTERED COLUMNSTORE INDEX
, DISTRIBUTION = HASH([P&L])
) ;
```

Distribution Key

Avoid Data Skew



Even Distribution



Distribution Key

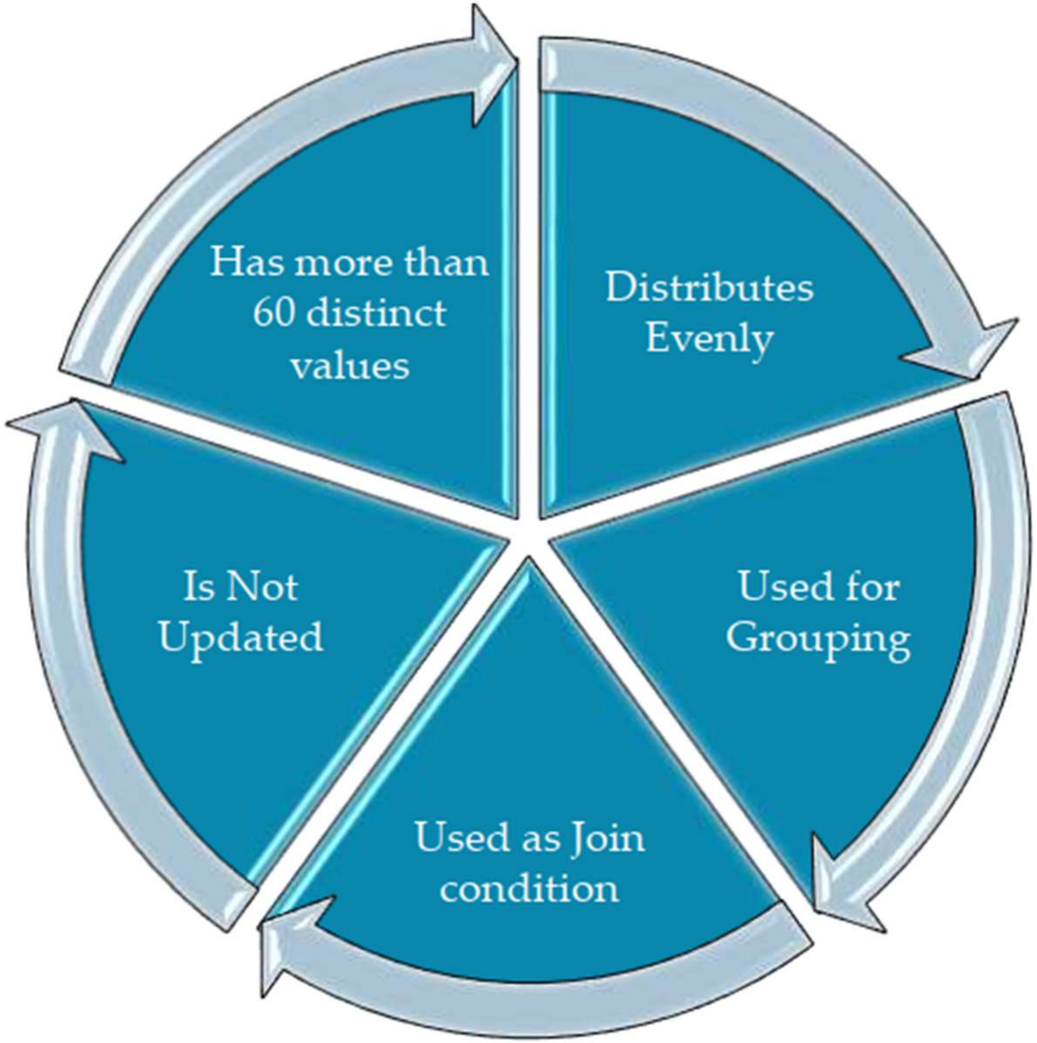
Using which

- Azure SQL Data Warehouse spreads the data across multiple nodes.

Up to 60 distributions

- Are used when loading data into the system

Good Hash Key



Round-robin distributed tables

Default distribution type

Simplest table to create

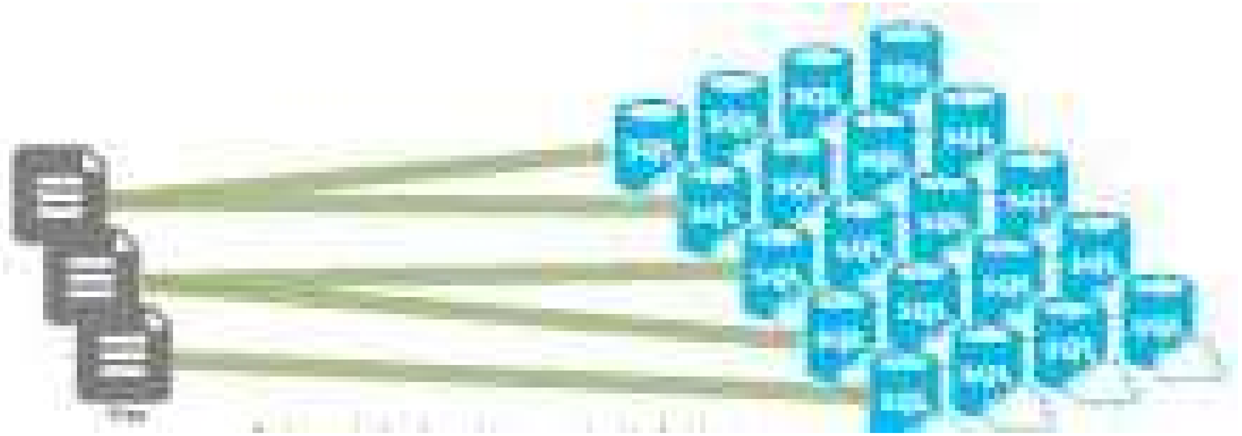
Distributes data evenly

- Across the table without any further optimization.

Takes additional time

- Joins require reshuffling data, which.

```
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) ;
```



Replicated Tables

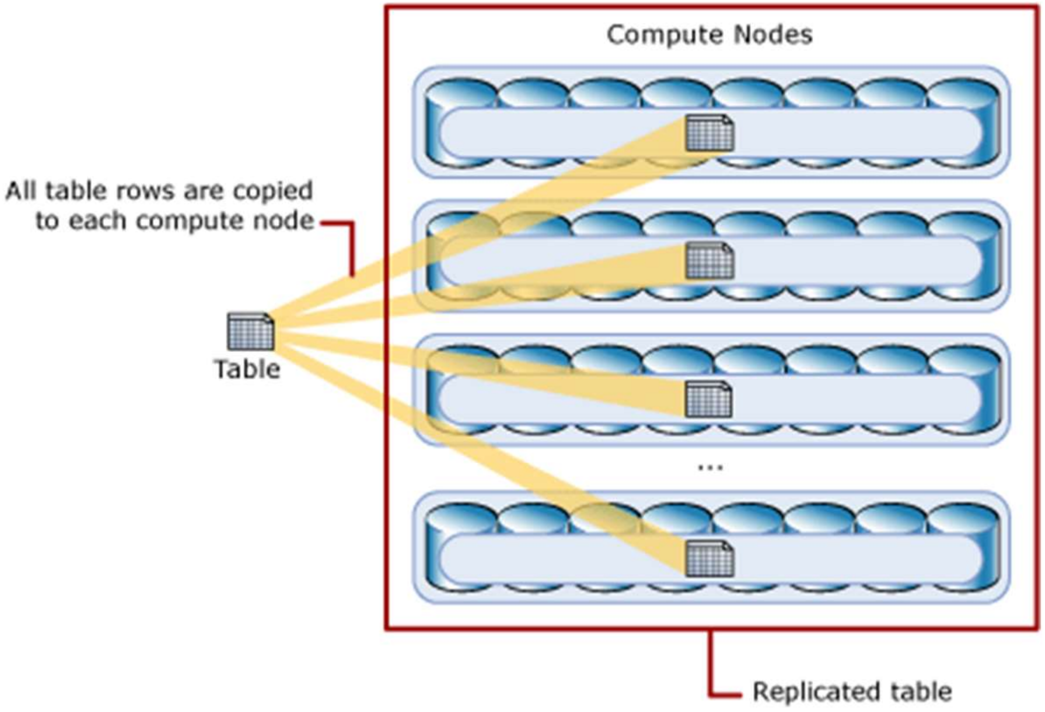
For small tables.

Caches

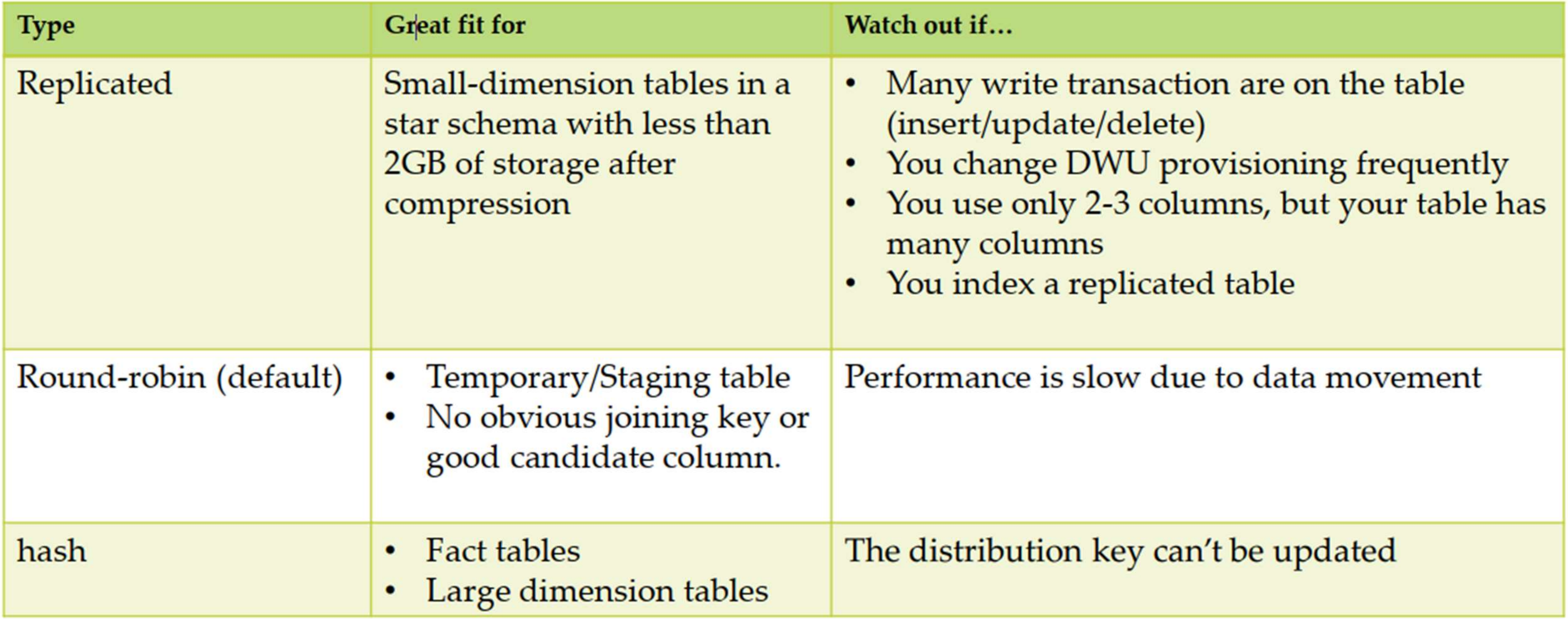
- A full copy of table on each compute node

Best utilizedwith 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);
```



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 Warehouse Units (DWUs)

Combination of

- CPU
- Memory
- I/O

Are bundled

- Into units of compute scale called Data Warehouse Units (DWUs).

Increase DWUs

- For higher performance

How many data warehouse units do I need?

Begin By

- Selecting a smaller DWU.

Monitor

- Application performance as test data loads into the system
- Observing the number of DWUs selected compared to the performance observe.

Peak Activity

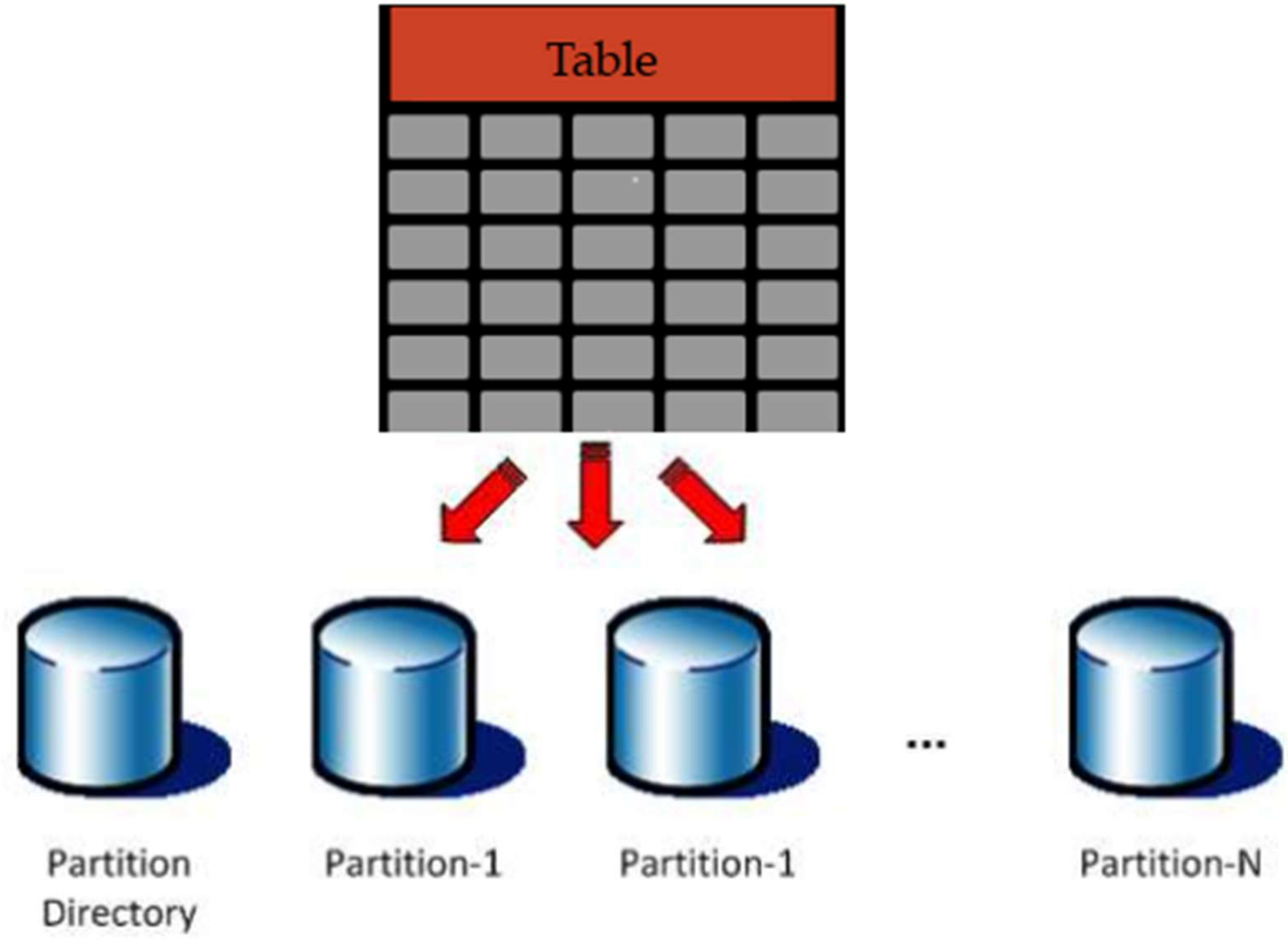
- Identify any additional requirements for periodic periods of peak activity.

Significant Peaks

- Workloads that show significant peaks in activity may need to be scaled frequently.

Table Partitioning

Table Partitioning



Partitioning

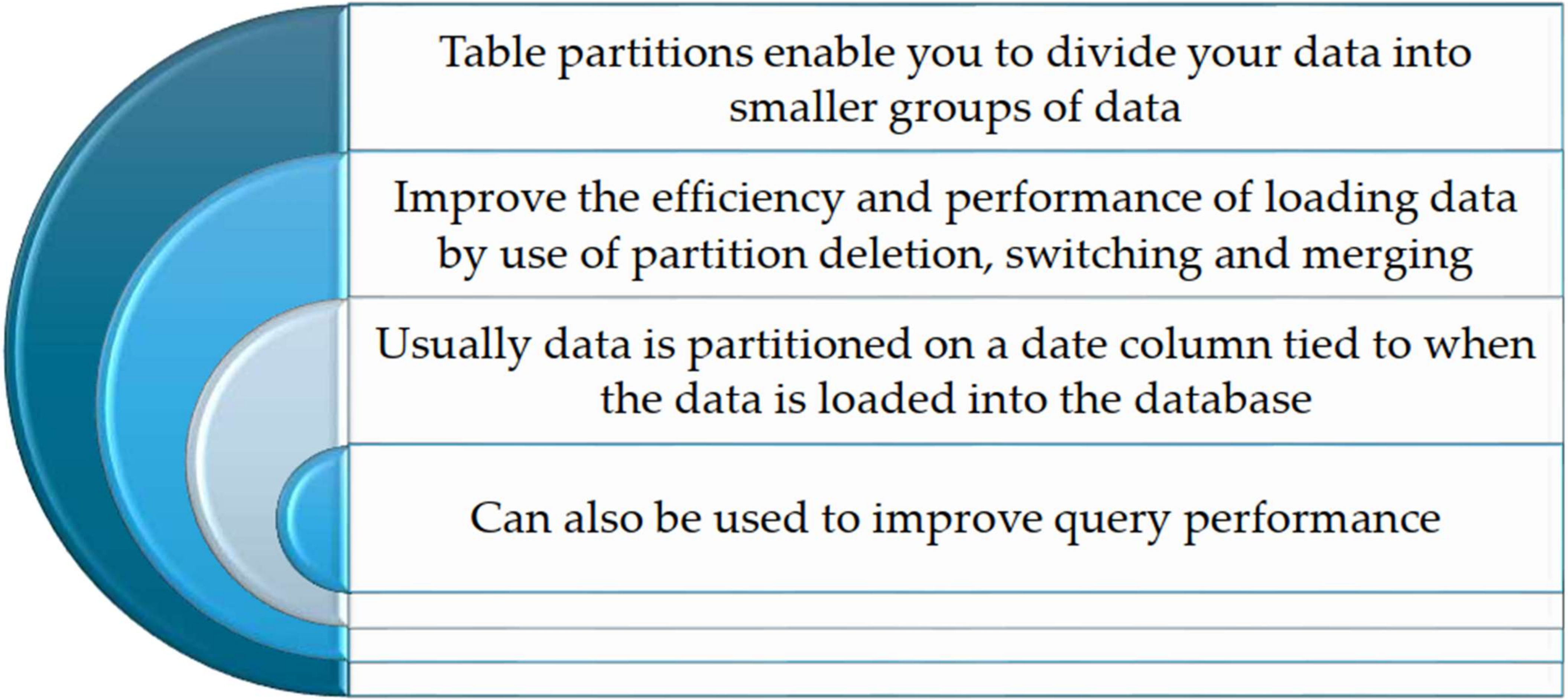


Table partitions enable you to divide your data into smaller groups of data

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

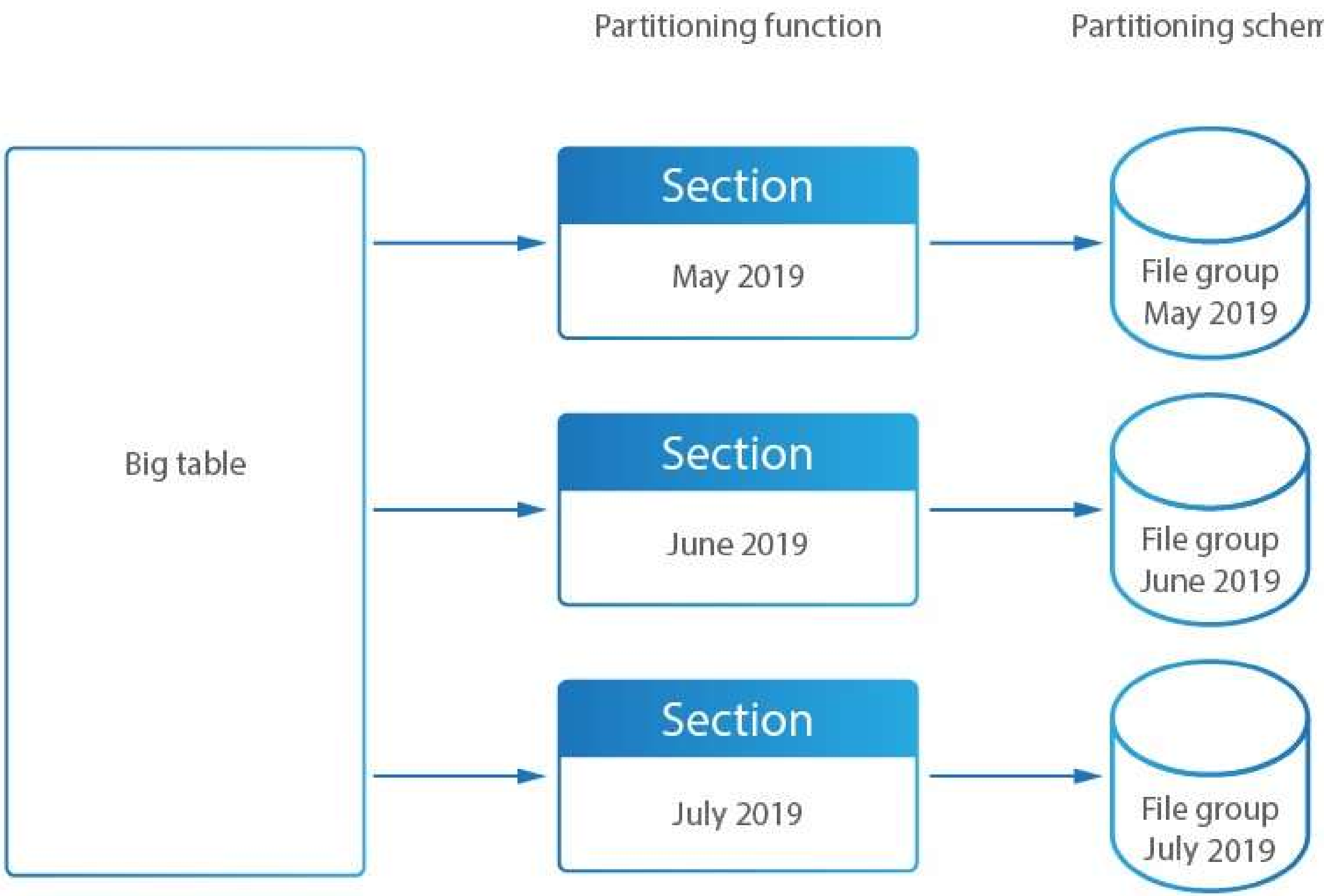
Why Partitioning?

Easy load or unload data

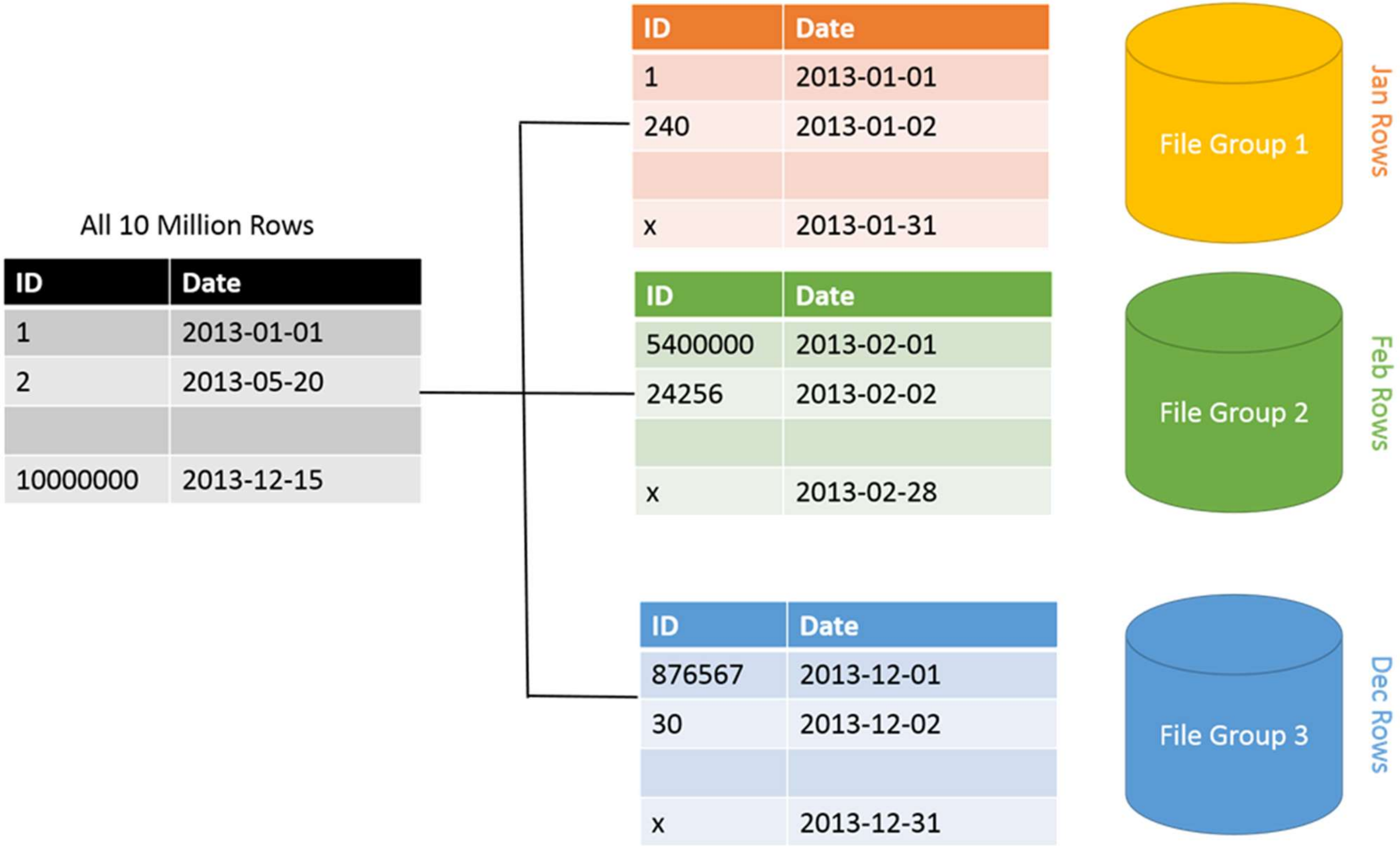
Easy maintenance, rebuilds or reorganizes

Improve Query performance

Partitioning



Partitioning



All 10 Million Rows

ID	Date
1	2013-01-01
2	2013-05-20
10000000	2013-12-15

ID	Date
1	2013-01-01
240	2013-01-02
x	2013-01-31

File Group 1

Jan Rows

ID	Date
5400000	2013-02-01
24256	2013-02-02
x	2013-02-28

File Group 2

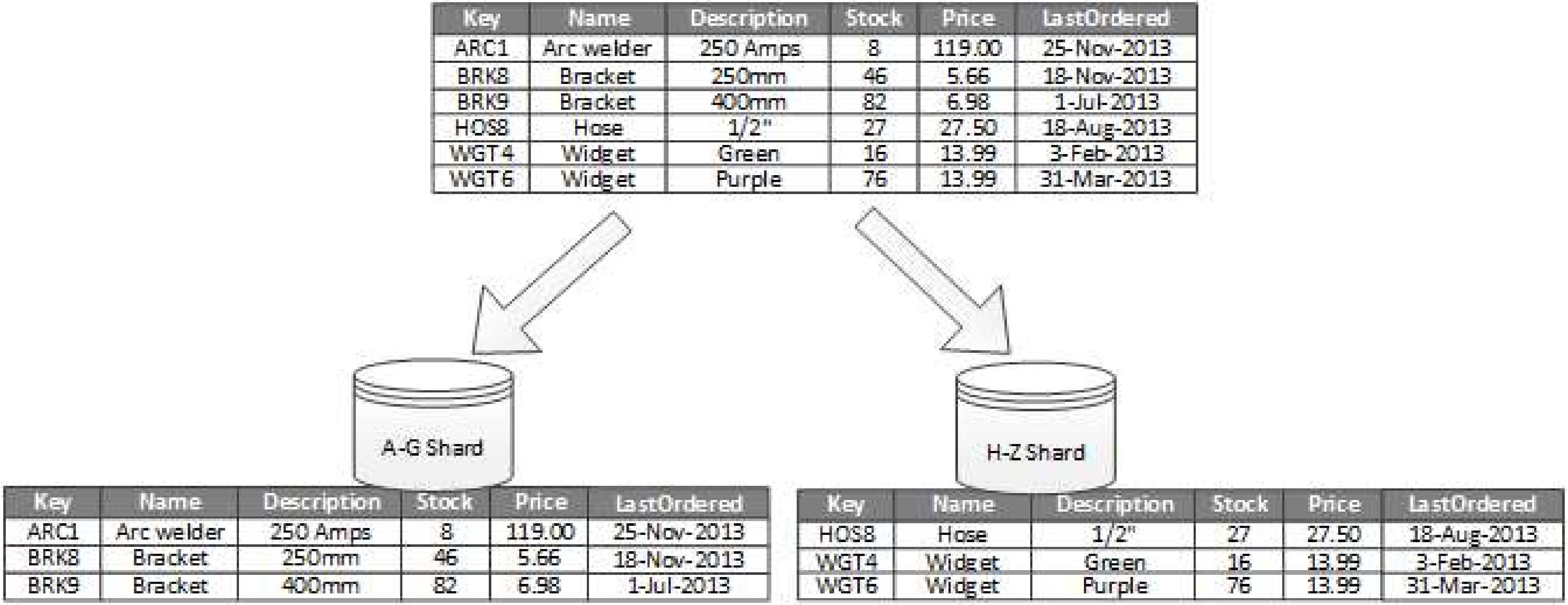
Feb Rows

ID	Date
876567	2013-12-01
30	2013-12-02
x	2013-12-31

File Group 3

Dec Rows

Sharding

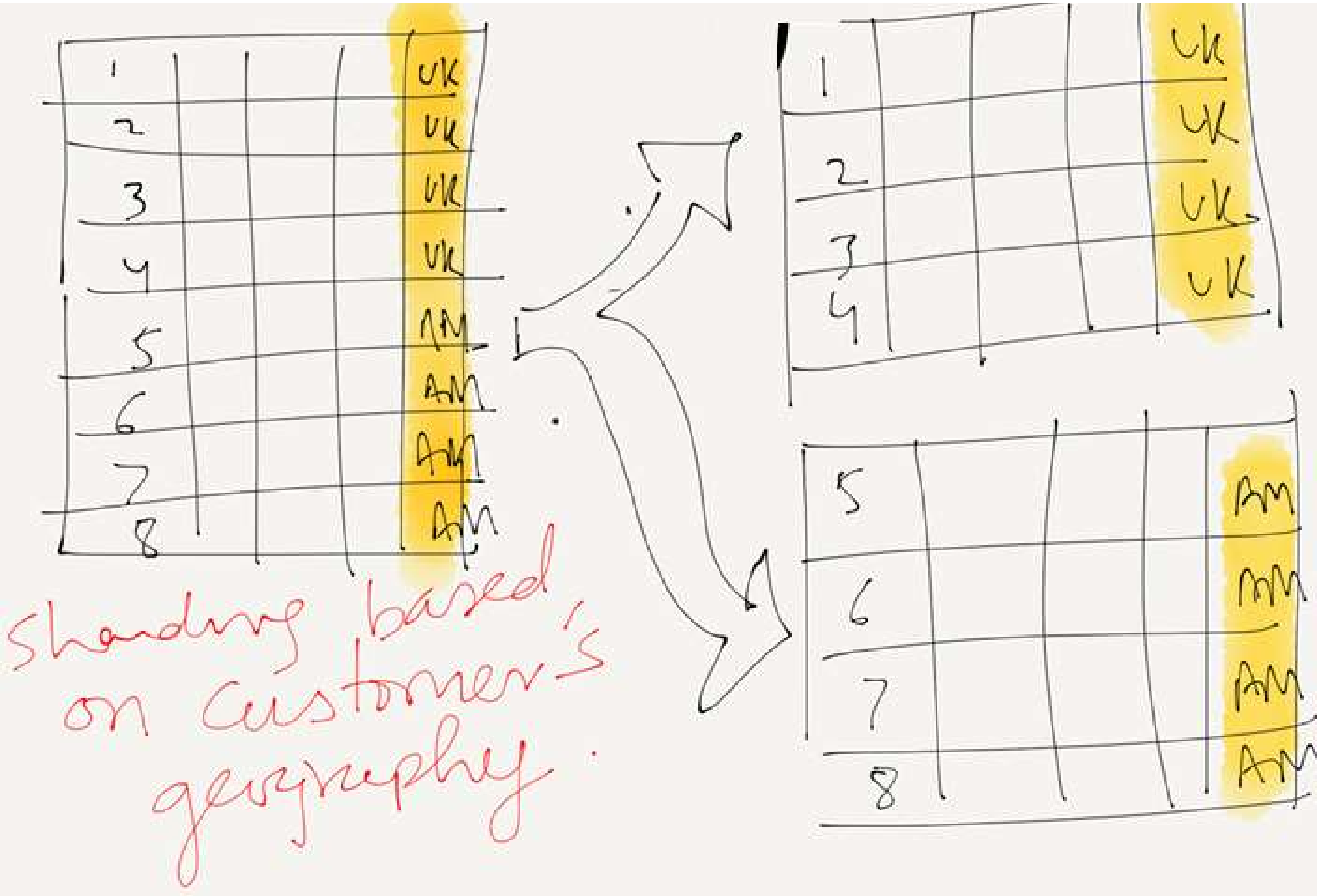


Key	Name	Description	Stock	Price	LastOrdered
ARC1	Arc welder	250 Amps	8	119.00	25-Nov-2013
BRK8	Bracket	250mm	46	5.66	18-Nov-2013
BRK9	Bracket	400mm	82	6.98	1-Jul-2013
HOS8	Hose	1/2"	27	27.50	18-Aug-2013
WGT4	Widget	Green	16	13.99	3-Feb-2013
WGT6	Widget	Purple	76	13.99	31-Mar-2013

Key	Name	Description	Stock	Price	LastOrdered
ARC1	Arc welder	250 Amps	8	119.00	25-Nov-2013
BRK8	Bracket	250mm	46	5.66	18-Nov-2013
BRK9	Bracket	400mm	82	6.98	1-Jul-2013

Key	Name	Description	Stock	Price	LastOrdered
HOS8	Hose	1/2"	27	27.50	18-Aug-2013
WGT4	Widget	Green	16	13.99	3-Feb-2013
WGT6	Widget	Purple	76	13.99	31-Mar-2013

Sharding



1				UK
2				UK
3				UK
4				UK
5				AM
6				AM
7				AM
8				AM

1				UK
2				UK
3				UK
4				UK

5				AM
6				AM
7				AM
8				AM

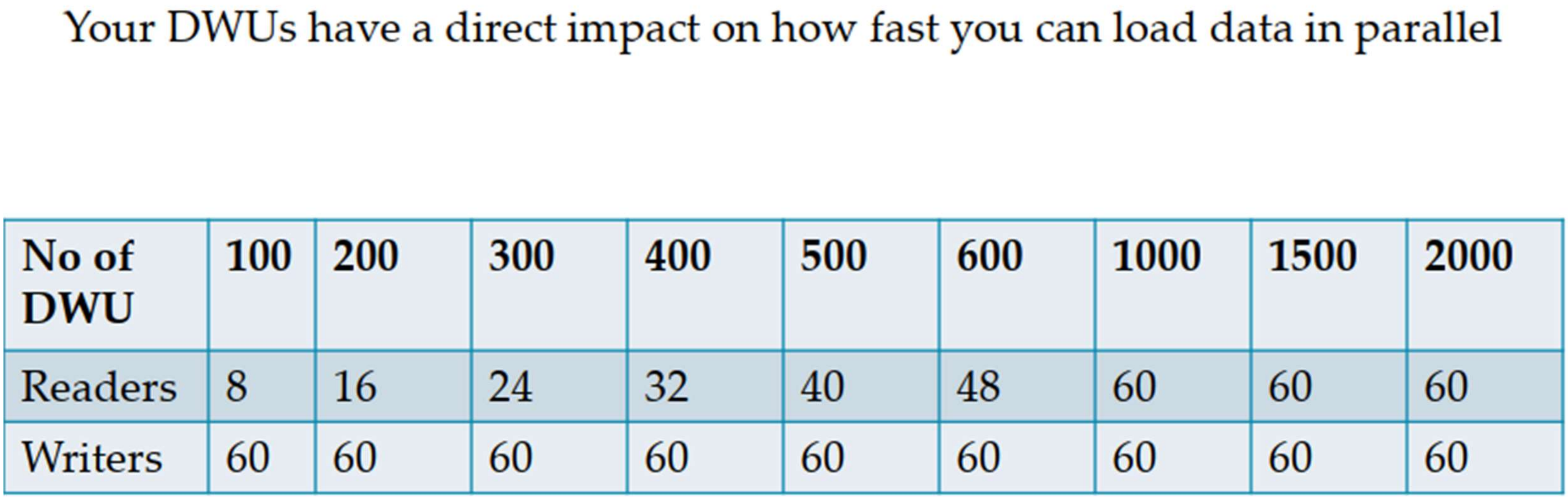
Sharding based on customer's geography.

Hands-on: Analyse data distribution

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

Best Practices

Best Practices: Data Warehouse Readers

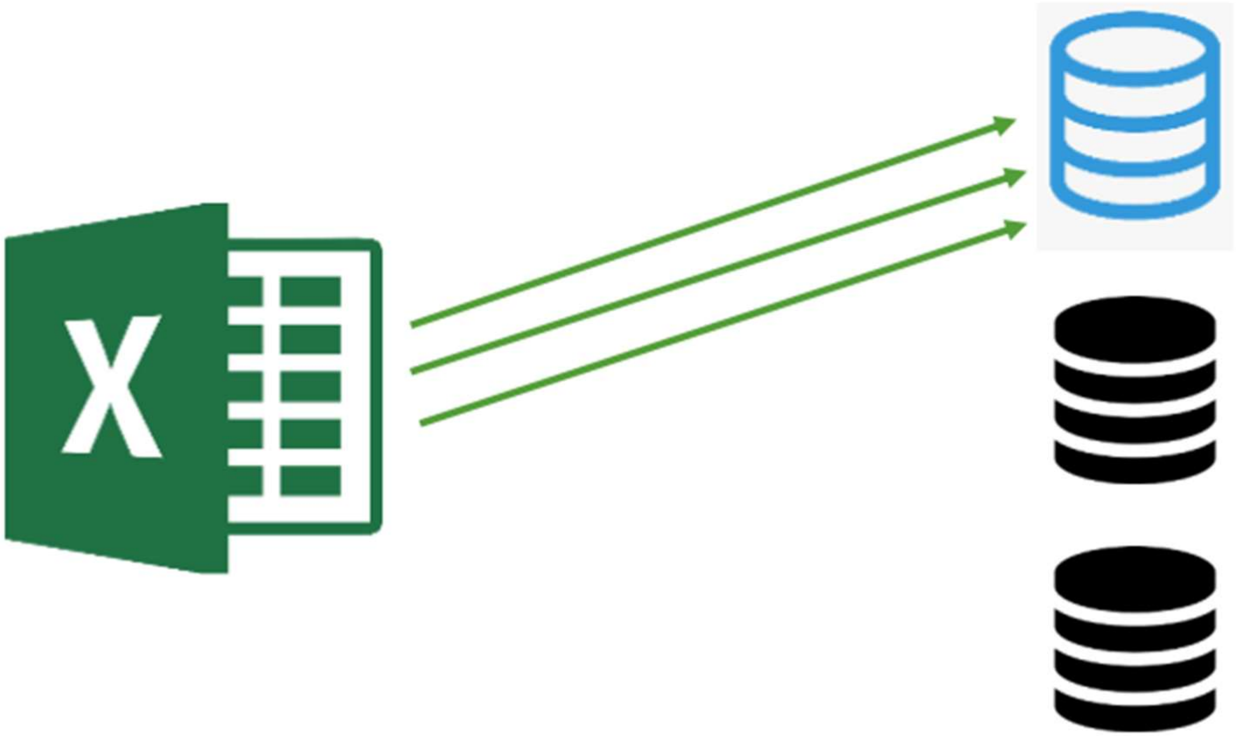


Your DWUs have a direct impact on how fast you can load data in parallel

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

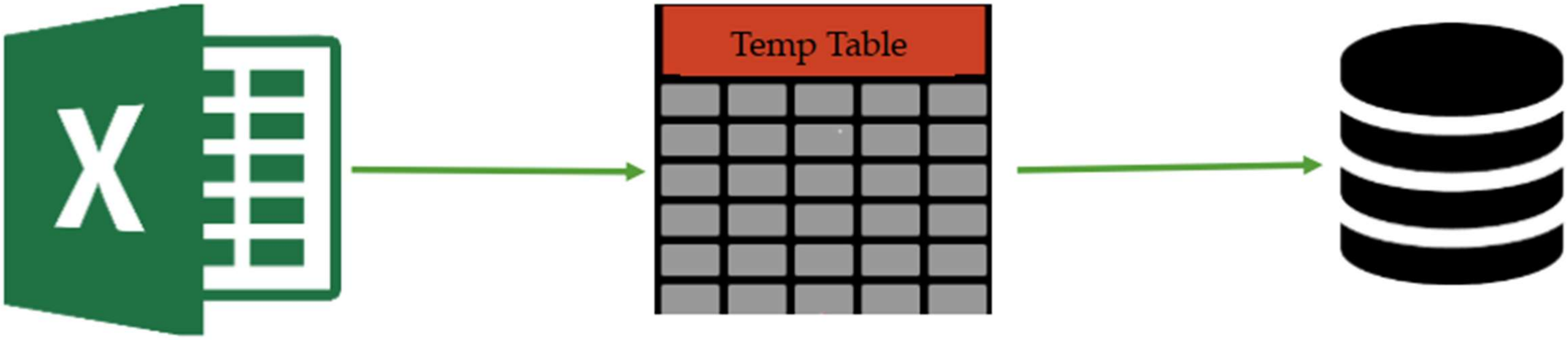
Best Practices: Avoid ordered data

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

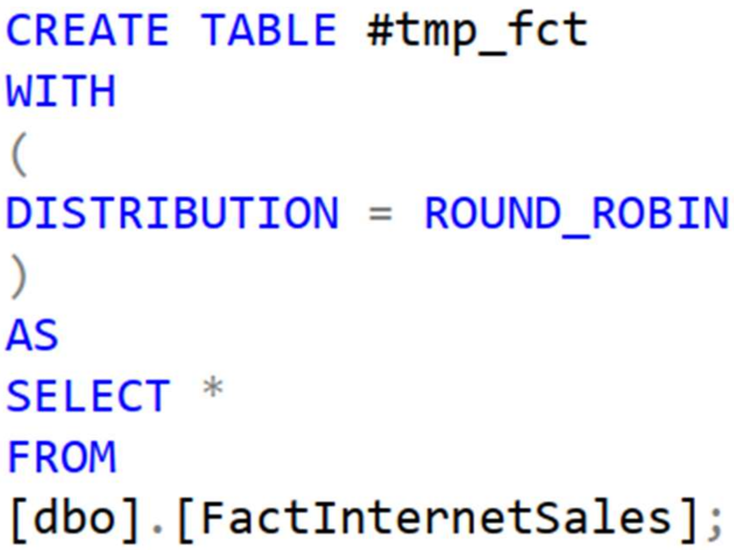


Best Practices: Using temporary tables

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



Best Practices: 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

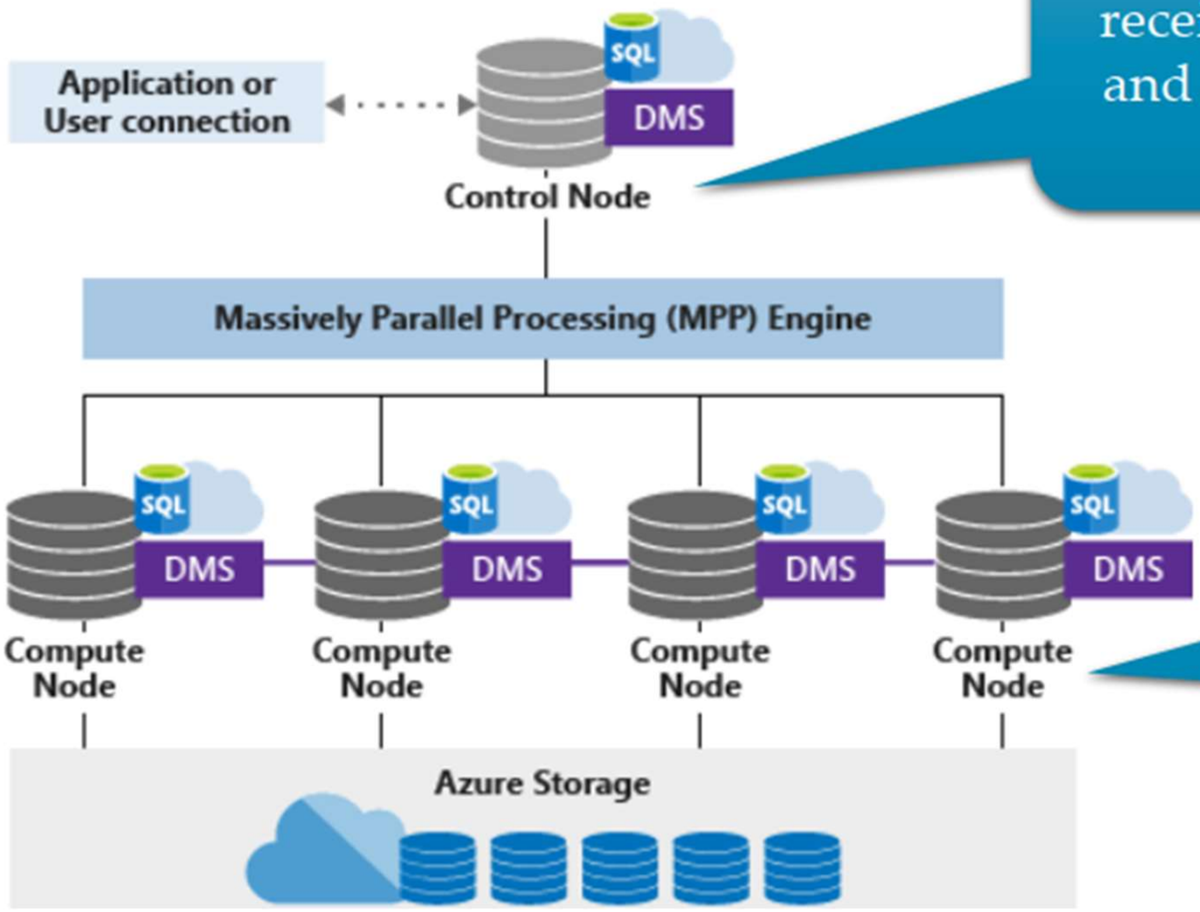
Single Client

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

Parallel Readers

- 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

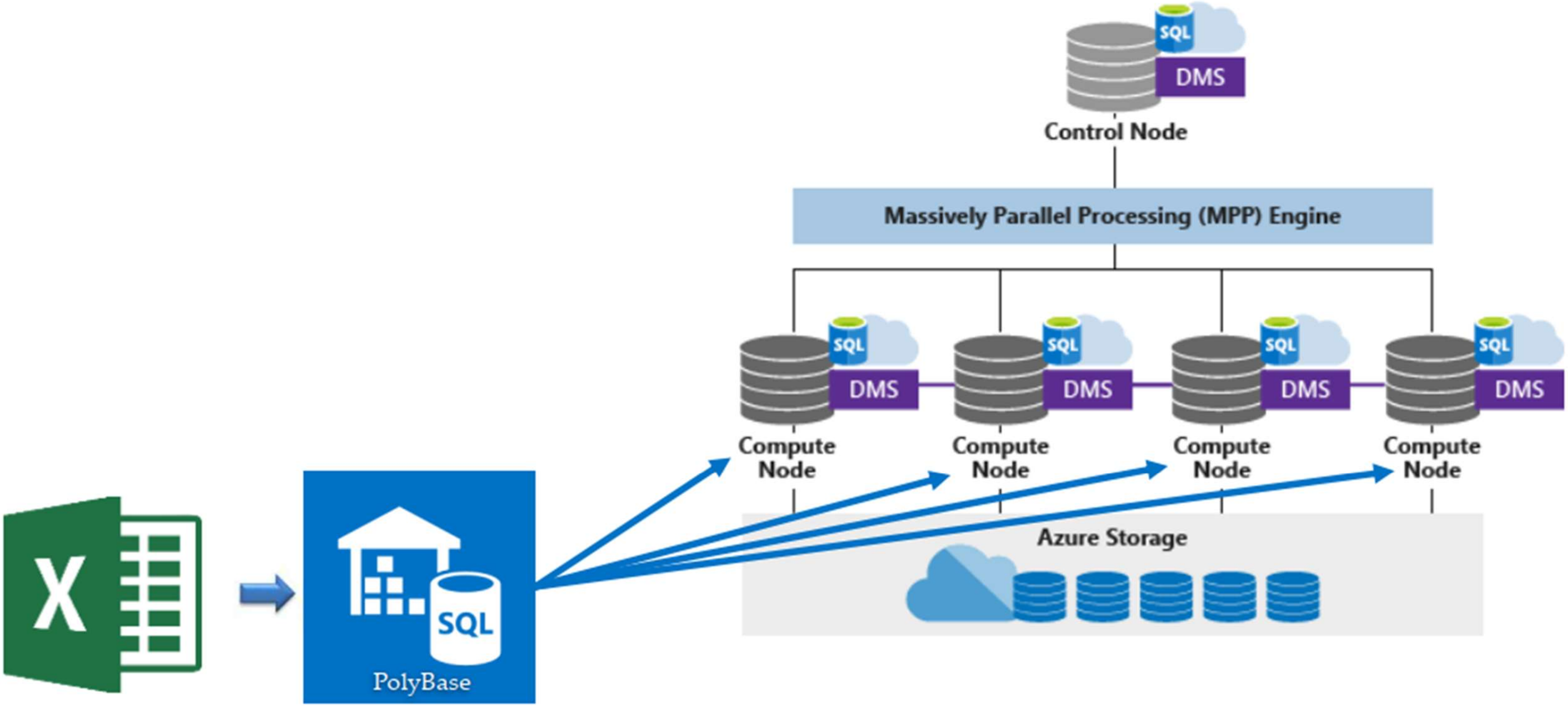
Control Node



The Control node receives connections and orchestrates the queries

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

Loading with PolyBase



Design tables in Synapse SQL pool

Determine table category

A Star Schema

- Organizes data into fact and dimension tables.

Decide if Table Data Belongs in a

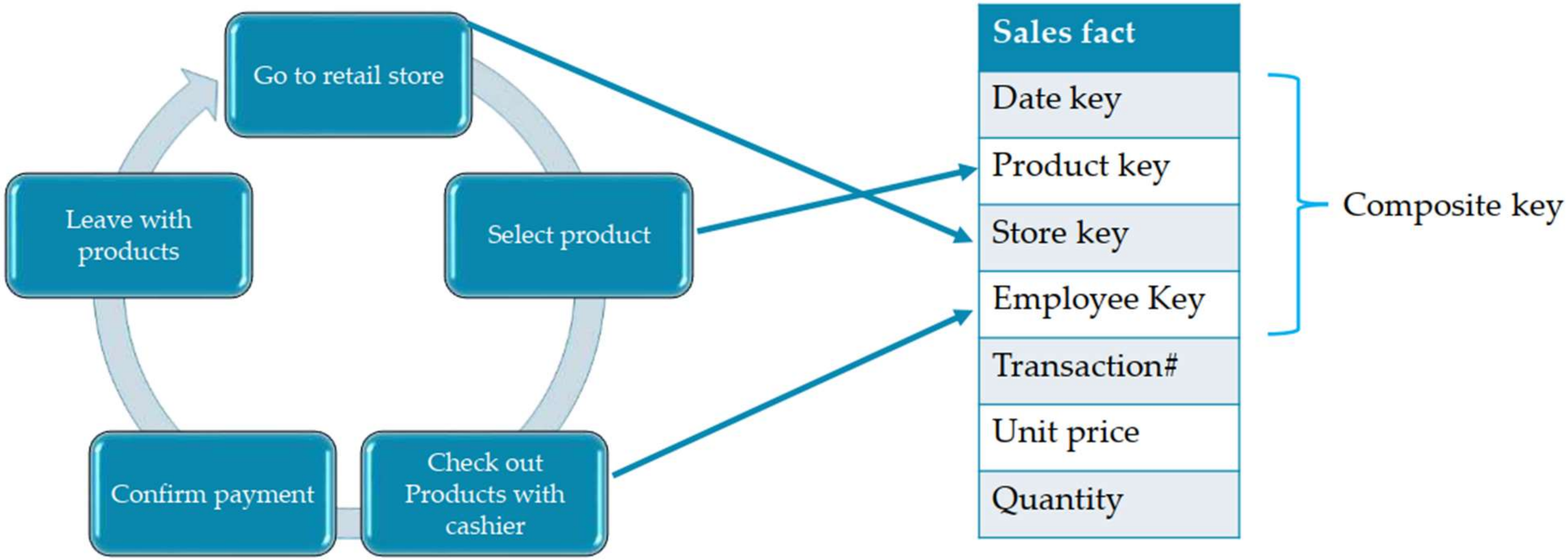
- Fact,
- Dimension, or
- Integration table

This Decision informs

- The appropriate table structure and distribution.

Fact tables

- Contain quantitative data that are generated in a transactional system
- For example
  - A retail business generates sales transactions every day, and then
  - Loads the data into a SQL pool fact table for analysis.



Go to retail store

Select product

Check out Products with cashier

Confirm payment

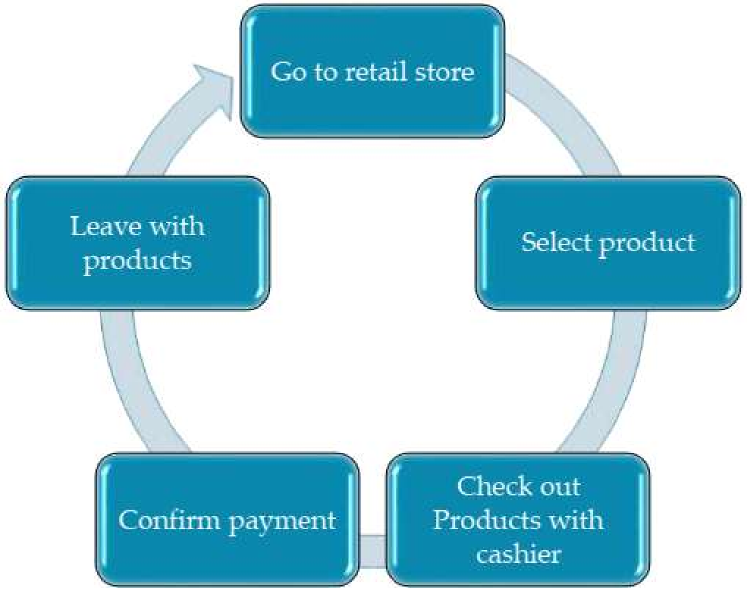
Leave with products

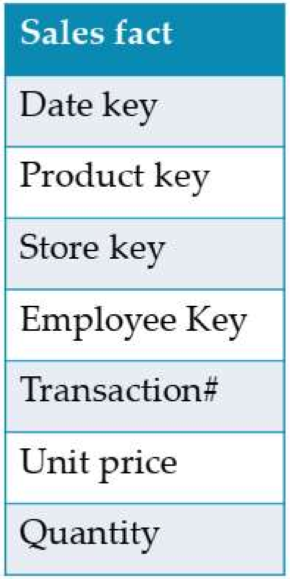
Sales fact
Date key
Product key
Store key
Employee Key
Transaction#
Unit price
Quantity

Composite key

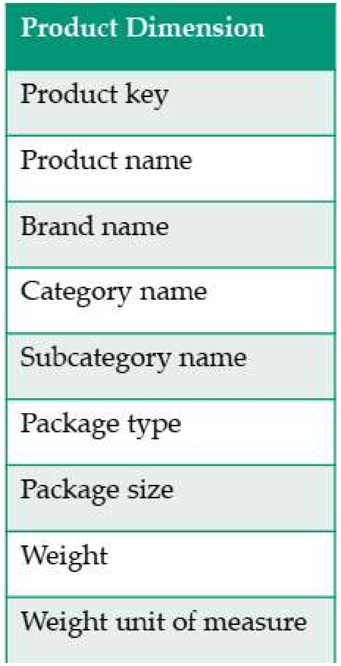
Dimension tables

- Contain attribute data that changes infrequently.
- For example
  - A product name, brand name and weight are stored in a dimension table
  - And updated only when the product details are changes





Sales fact
Date key
Product key
Store key
Employee Key
Transaction#
Unit price
Quantity



Product Dimension
Product key
Product name
Brand name
Category name
Subcategory name
Package type
Package size
Weight
Weight unit of measure

Integration tables

For integrating or staging data

Can create an Integration Table as

- Regular table
- External table or
- Temporary table

Example

- Can load data to a staging table
- Perform transformations on the data in staging, and
- Insert the data into a production table.

Table persistence

- Tables store data either
  - Permanently in Azure Storage,
  - Temporarily in Azure Storage, or
  - In a data store external to SQL pool.

Regular table

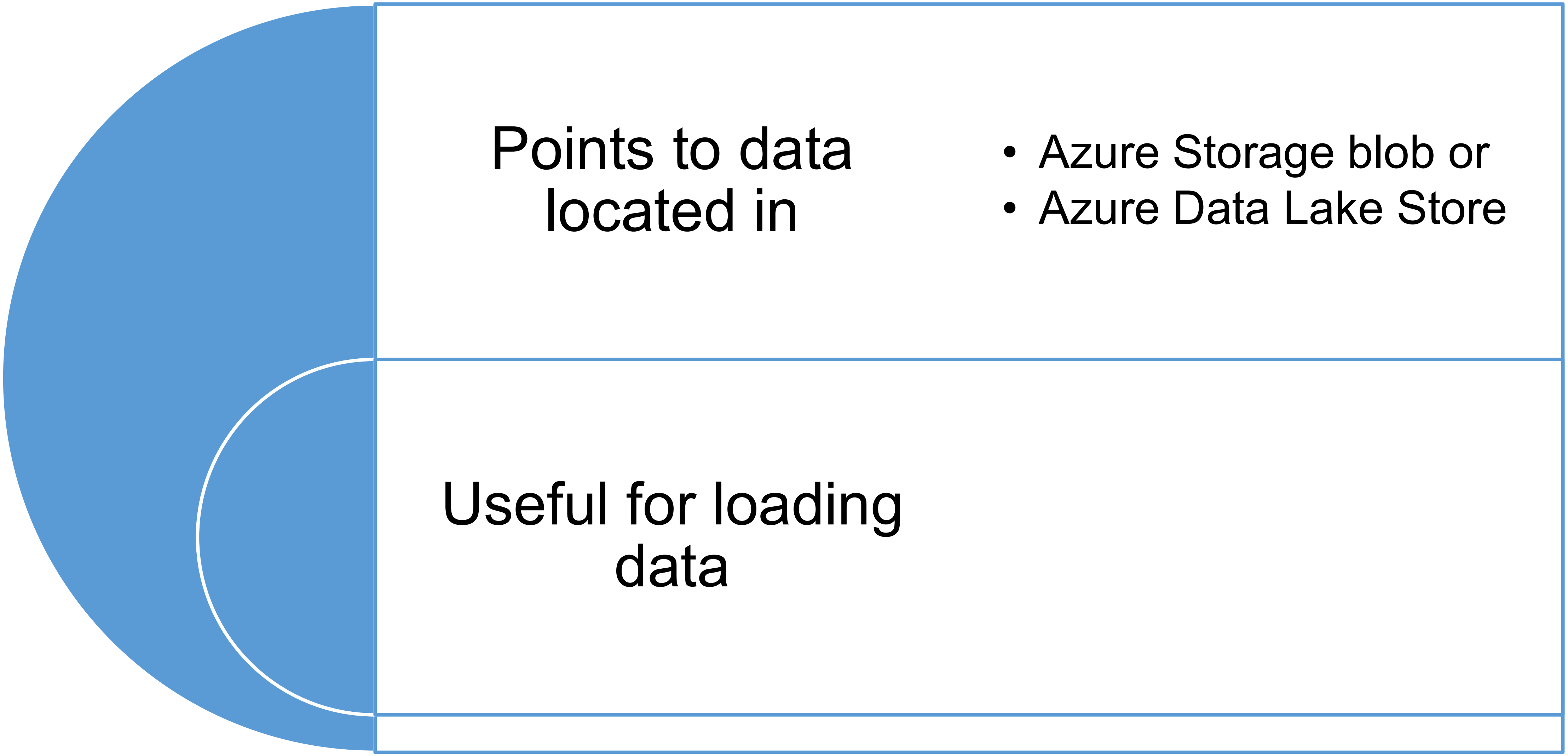
- Stores data in Azure Storage as part of SQL pool
- The table and the data persist regardless of whether a session is open

- The following example creates a regular table with two columns.
  - CREATE TABLE MyTable (col1 int, col2 int);

Temporary table

- Only exists for duration of session
  - To prevent other users from seeing temporary results and
  - To reduce the need for cleanup.
- Are created by prefixing with a #
- For example:
  - CREATE TABLE #stats_ddl
  - (
  -     [schema_name]   NVARCHAR(128) NOT NULL
  - ,   [table_name]   NVARCHAR(128) NOT NULL
  -)
  - WITH
  - (
  -     DISTRIBUTION = HASH([seq_nmbr])
  - ,   HEAP
  -)

External table



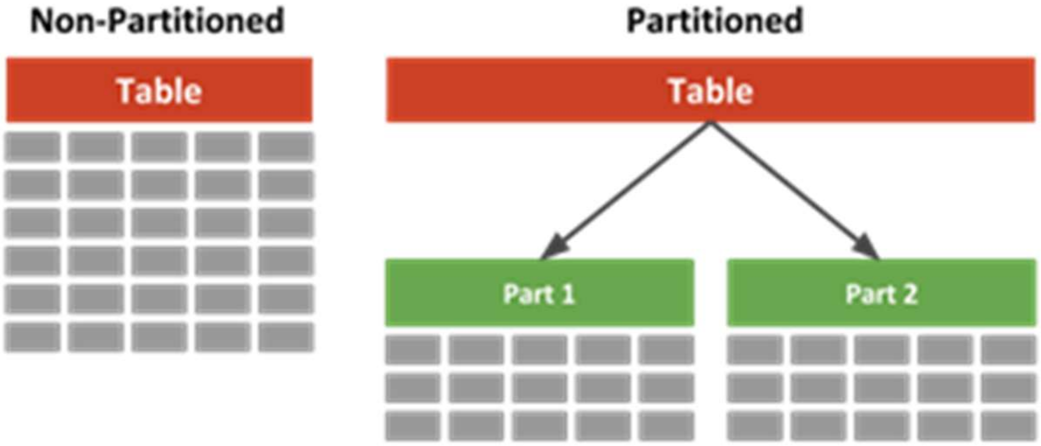
Points to data located in

- Azure Storage blob or
- Azure Data Lake Store

Useful for loading data

Table partitions

- A partitioned table stores and performs operations on the table rows according to data ranges
- For example, a table could be partitioned by day, month, or year
- You can improve query performance through partition elimination, which limits a query scan to data within a partition.

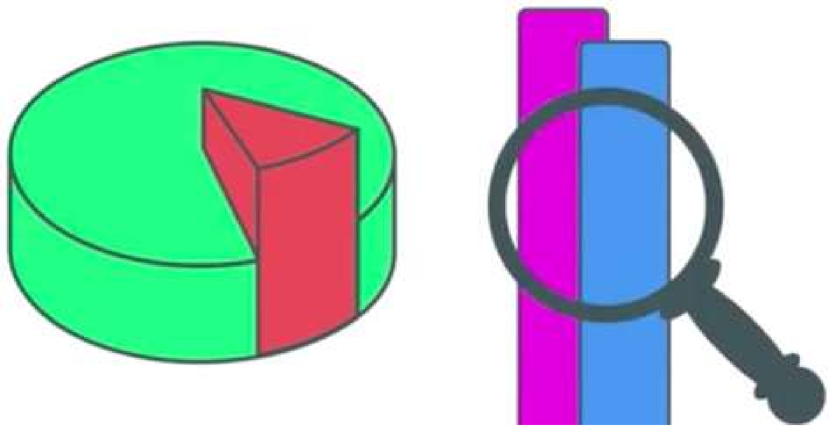


Statistics

Used by Query Optimizer

- When it creates the plan for executing a query

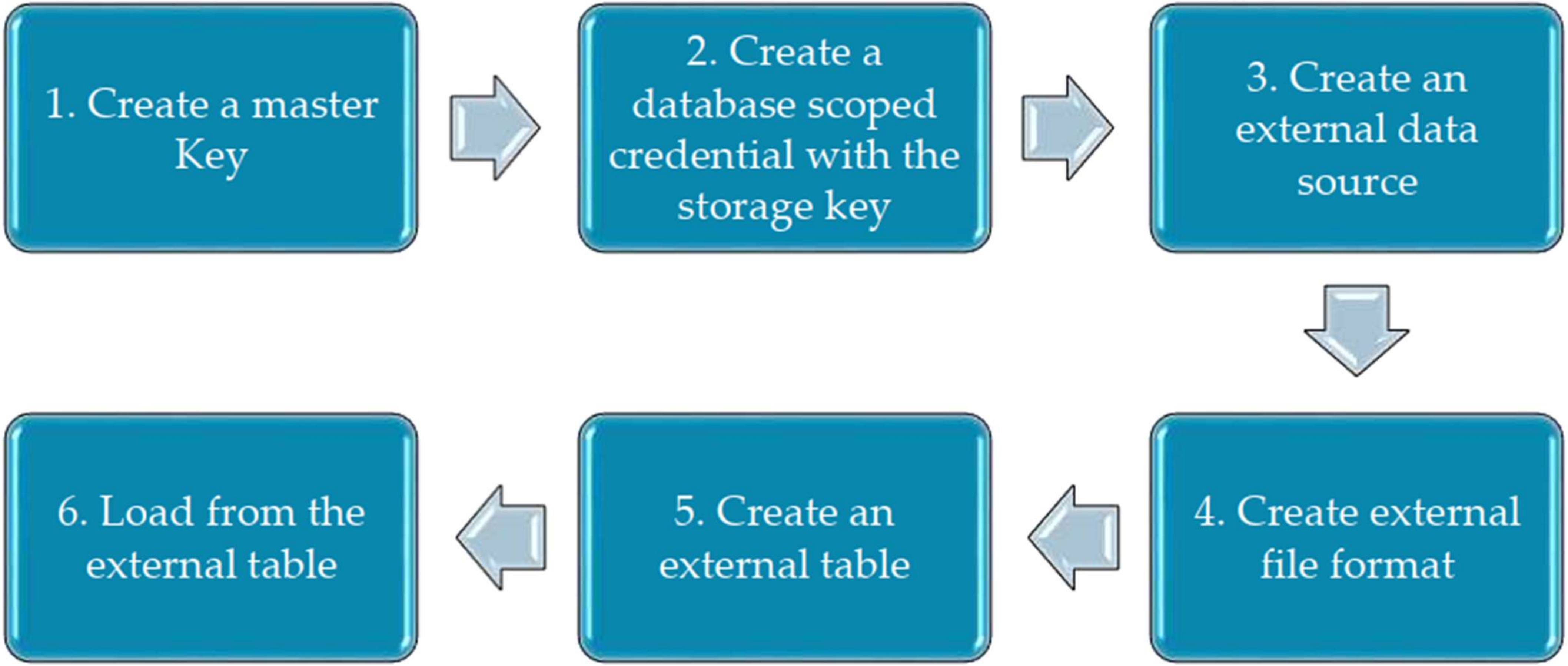
To improve query performance



- It's important to have statistics on individual colum
- Especially for columns used in query joins

Data Migration

PolyBase Setup



1. Create a master Key

2. Create a database scoped credential with the storage key

3. Create an external data source

4. Create external file format

5. Create an external table

6. Load from the external table

Hand-on: 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

Hand-on: Loading Data using Data Factory

Thanks