Designing for Azure SQL Data Warehouse

Warner Chaves
SQL MCM / MS DATA PLATFORM MVP

@warchav   sqlturbo.com

What's in This Module?

Design choices to consider for Azure SQL Data Warehouse

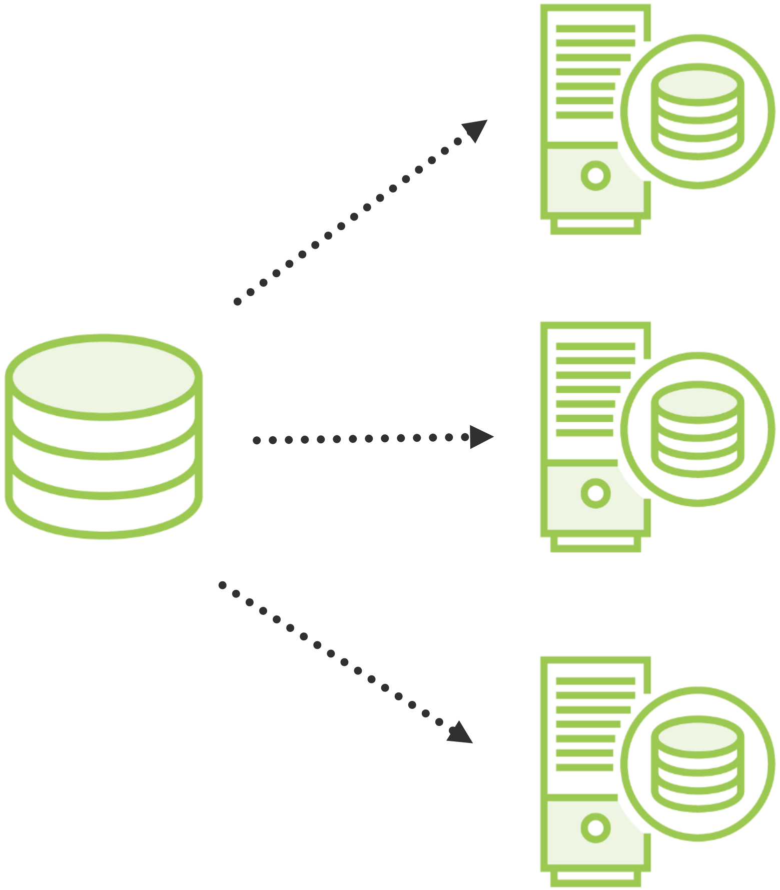
Considerations for Fact Tables

Considerations for Dimension Tables

SQL Server ≠ Azure SQL Data Warehouse

An Azure SQL DW database will require design decisions that are different from SQL Server.

Distribution Key



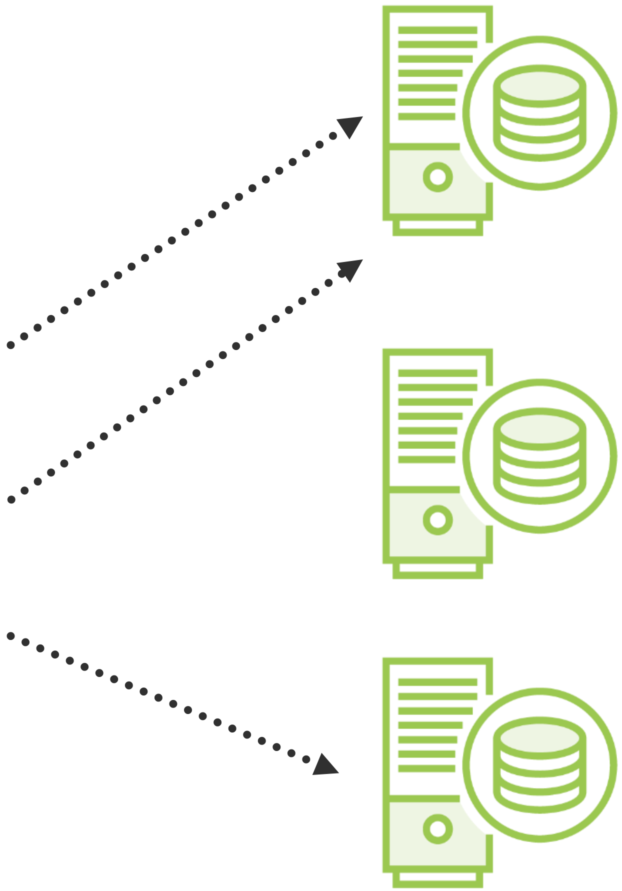
Determines the method in which Azure SQL Data Warehouse spreads the data across multiple nodes.

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

Hash Distribution

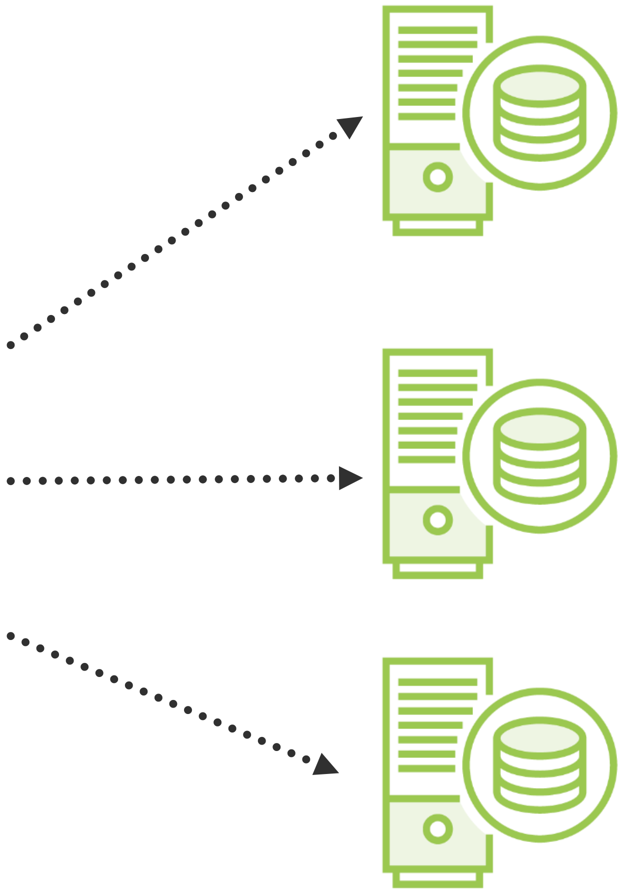
Record	Product	Store
1	Volleyball	New York
2	Volleyball	Chicago
3	Basketball	Atlanta

Hashing by Product



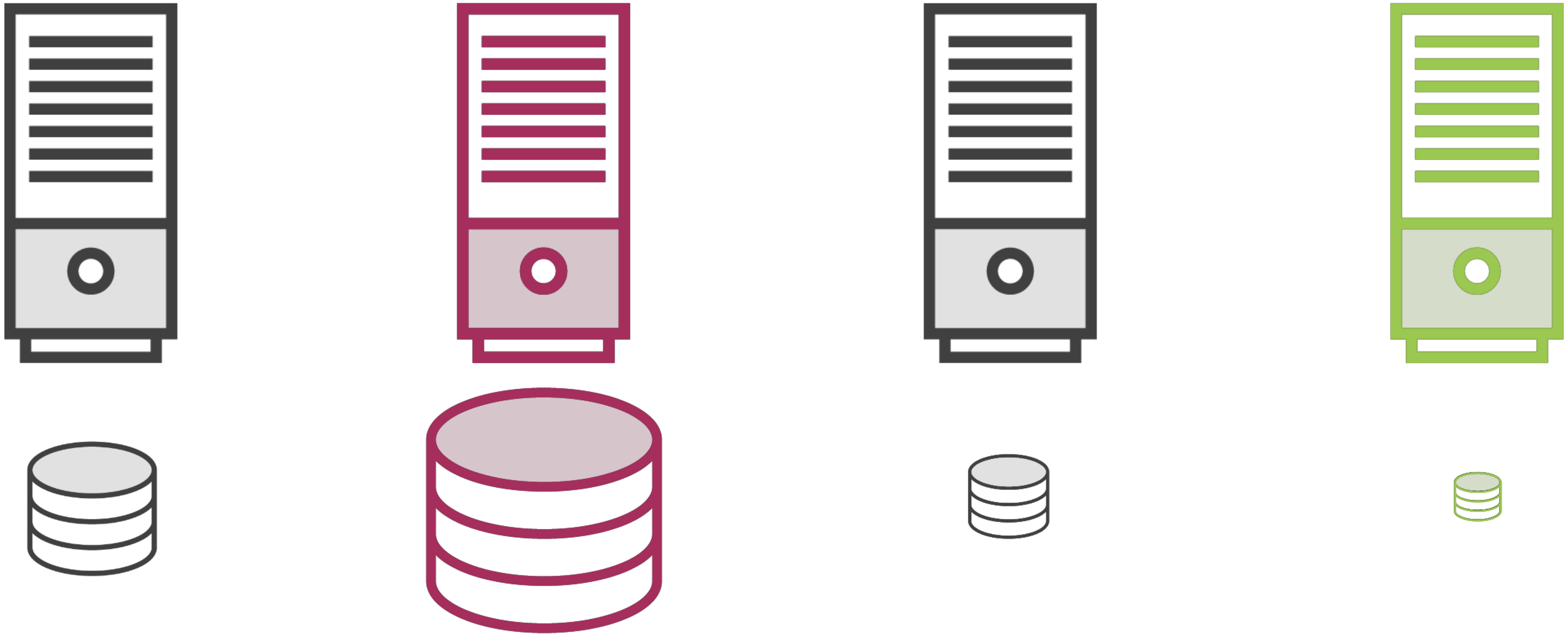
Round-Robin Distribution

Record	Product	Store
1	Volleyball	New York
2	Volleyball	Chicago
3	Basketball	Atlanta



Rows distributed to all nodes

Avoid Data Skew



Even Distribution

Good Hash Key

- **Distributes Evenly**
- Used for Grouping
- Used as Join Condition
- Is Not Updated
- Has more than 60 distinct values

Round-Robin will always provide a uniform distribution but not necessarily the best performance.

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.

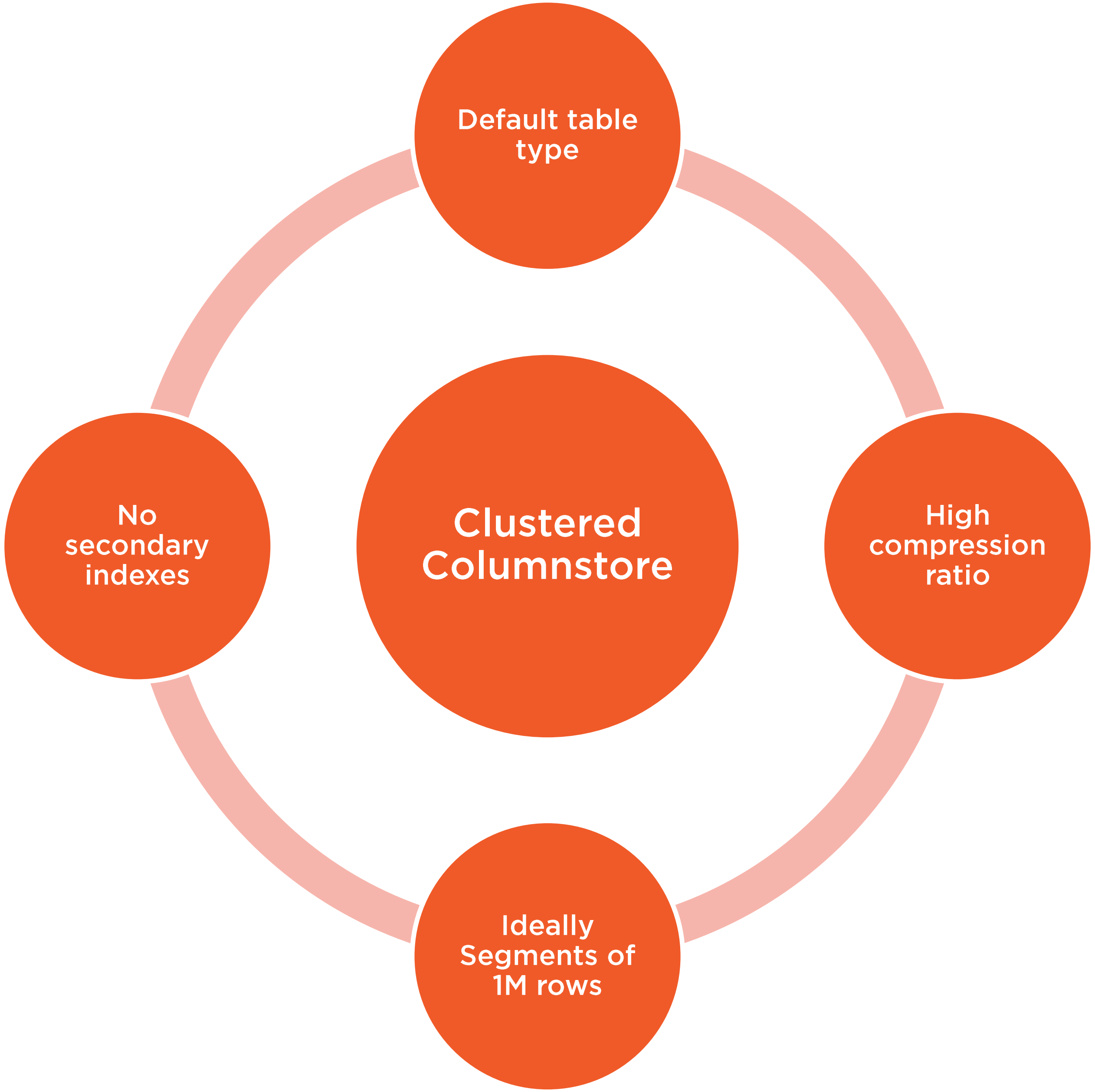
Some complex data types (xml, geography, etc) are not supported on Azure SQL Data Warehouse yet.

Table Types

Clustered Columnstore

Heap

Clustered B-Tree Index



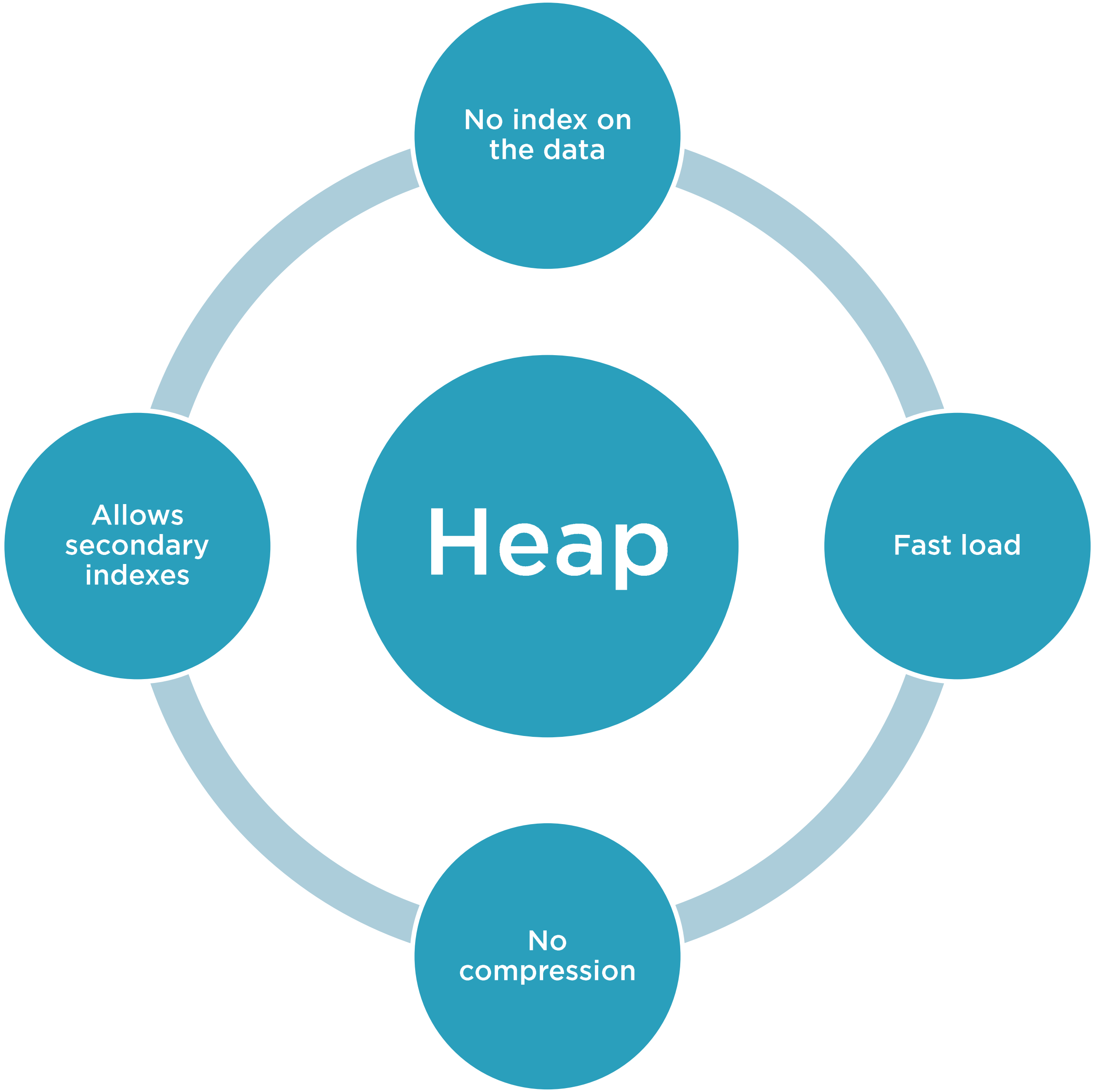
Default table type

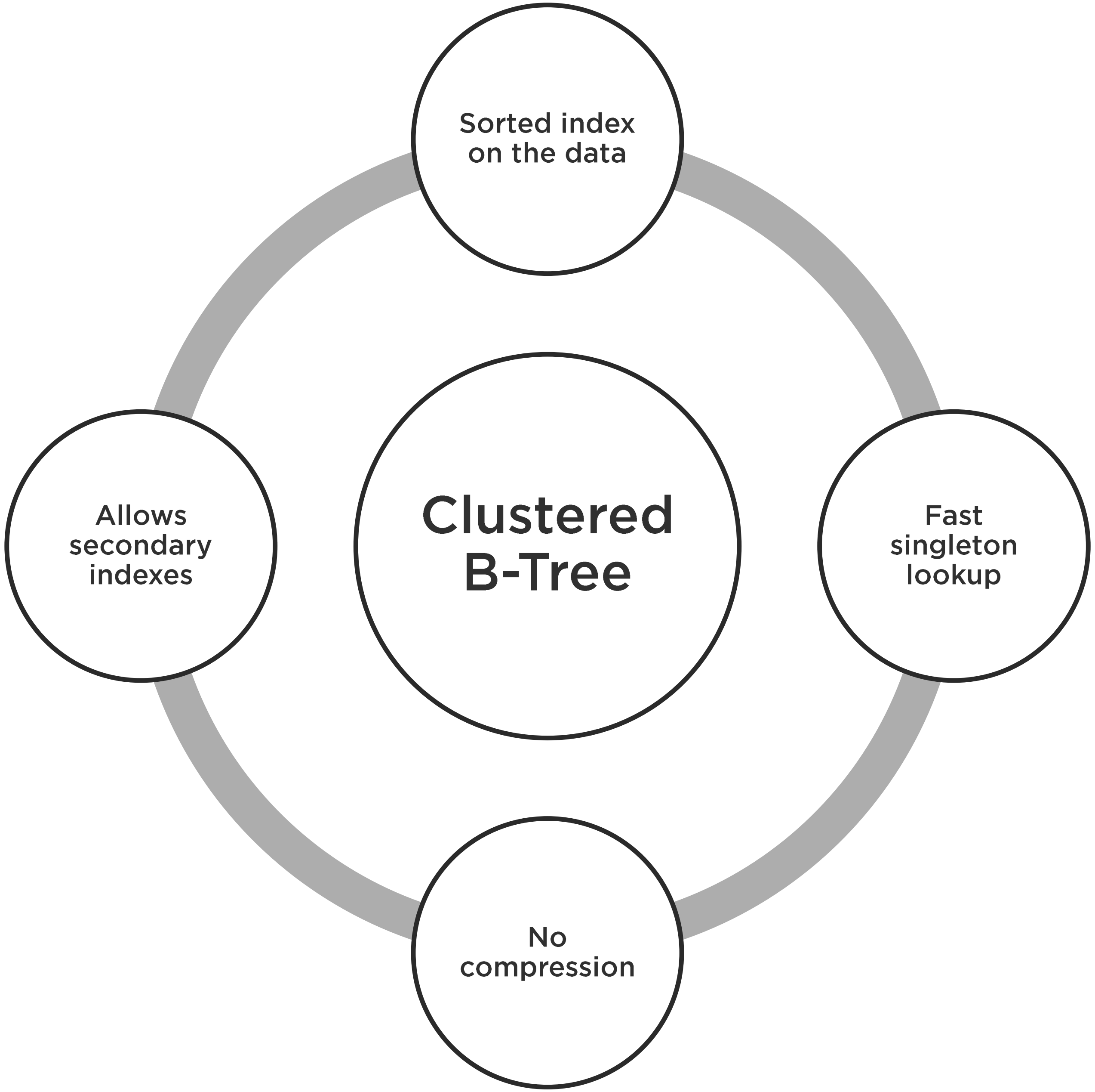
No secondary indexes

Clustered Columnstore

High compression ratio

Ideally Segments of 1M rows


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



Clustered B-Tree

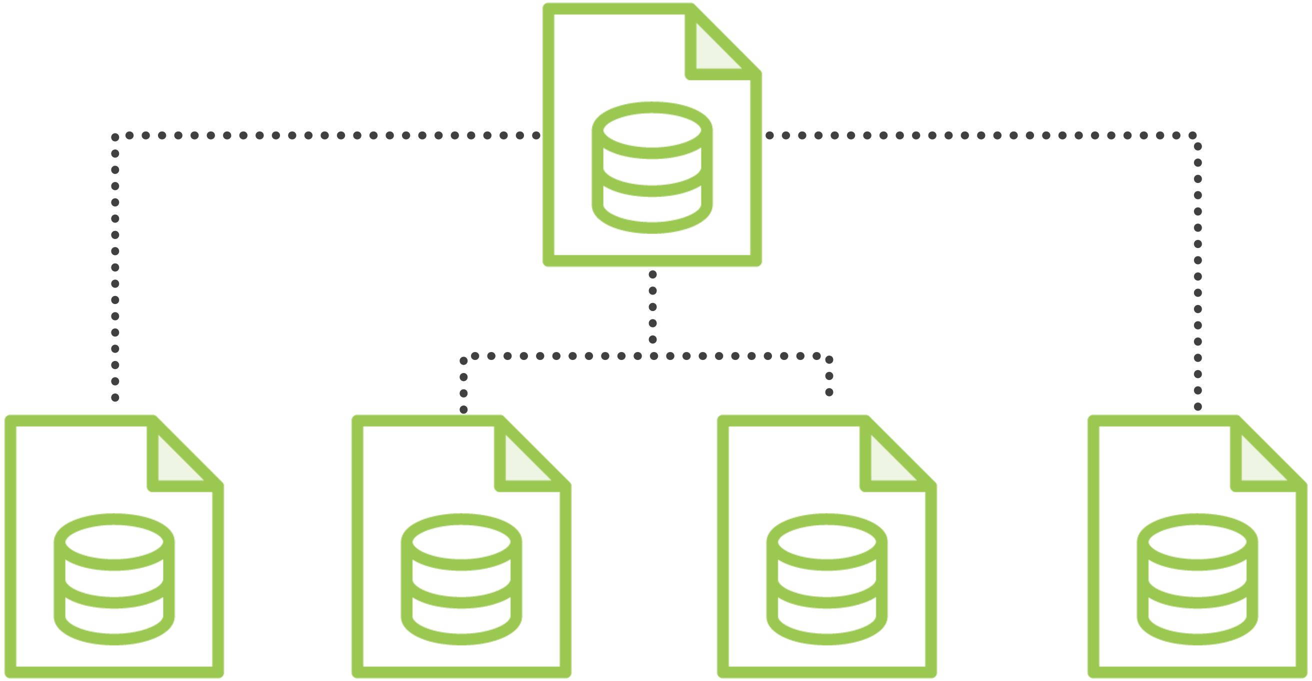
Sorted index on the data

Fast singleton lookup

No compression

Allows secondary indexes

Table Partitioning



Partitioning is very common in SQL Server Data Warehouses for three reasons:

1 - Ease of loading and removal of data from a partitioned table.

2 - Targeting specific partitions on table maintenance operations.

3 - Performance improvements due to partition elimination.

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

For 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

Can be Hash distributed or Round-Robin if there's no clear candidate join key

Columnstore for large dimensions

Heap or Clustered Index for small dimensions

Add secondary indexes for alternate join columns

Partitioning not recommended

Demo

Preparing the AdventureWorksDW database

Demo

Analyzing distribution and data types for Data Warehouse tables

Summary

Distributions are a new concept of Azure SQL Data Warehouse.

There are 3 table types: Columnstores, heaps and clustered b-tree indexes.

Partitioning has to be analyzed carefully.

Fact and Dimension tables have their own design Best Practices.

Next Module: Loading Data into Azure SQL Data Warehouse