Azure Synapse SQL Architecture

Synapse SQL offers two resource models called **Dedicated SQL pool and Serverless SQL pool.**

Dedicated SQL pool:

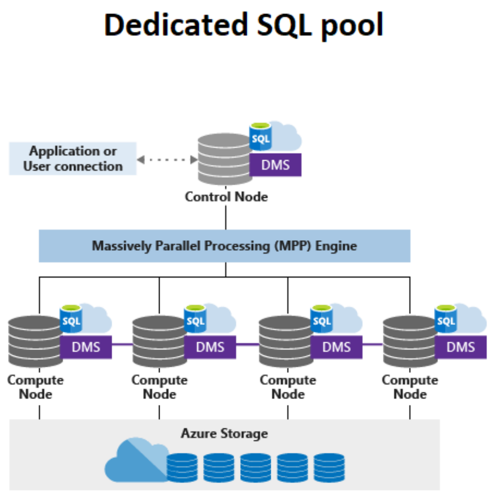
Dedicated SQL pool (formerly SQL DW) refers to the enterprise data warehousing features that are available in Azure Synapse Analytics. This SQL pool is charged based on Data Warehouse Units which is the compute unit. Compute is decided by us while creating the dedicated SQL pool.

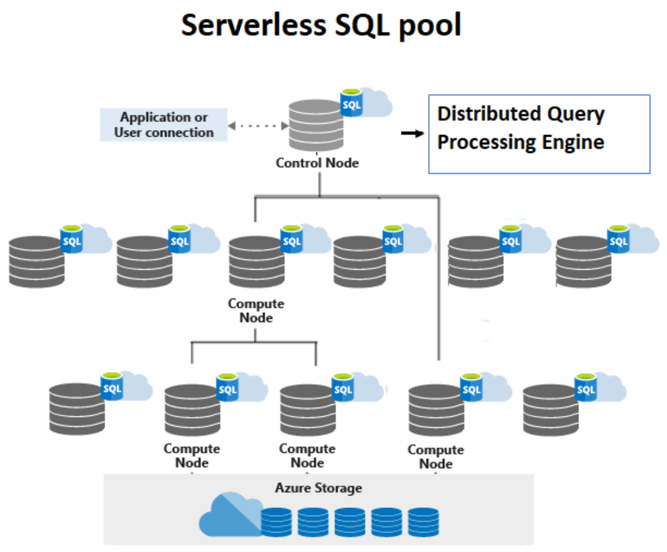
Serverless SQL pool:

A Serverless SQL pool is to analyze or query data stored in storage like Azure Datalake (parquet, delimited file, Delta lake) and Azure Cosmos DB or Dataverse. This pool is charged only for the amount of data getting processed.

Architecture :

Synapse SQL uses a node-based architecture. It distributes the computational processing of data across multiple nodes. Compute is separate from storage, which enables you to scale compute independently of the data in your system.


Dedicated SQL pool
Application or User connection
SQL
DMS
Control Node
Massively Parallel Processing (MPP) Engine
Compute Node
Azure Storage


Serverless SQL pool
Application or User connection
SQL
Control Node
Distributed Query Processing Engine
Compute Node
Azure Storage

Dedicated SQL:

It partitions and stores the data for optimized performance. We can define the sharding pattern to use to distribute the data when we create a table. The sharding patterns are **Hash, Roundrobin, and Replicate**.

When you submit a T-SQL query to a dedicated SQL pool, the **Control node** transforms it into queries that run against each distribution in parallel.

Data Movement Service (DMS) is the data transport technology in the dedicated SQL pool that coordinates data movement between the Compute nodes. Some queries require data movement to ensure that parallel queries return accurate results. When data movement is required, DMS ensures the right data gets to the right location.

Serverless SQL Pool:

For a **serverless SQL pool**, applications connect and issue T-SQL commands to a Control node, which is the single point of entry for Synapse SQL. The Azure Synapse SQL Control node utilizes a **distributed query engine** to optimize queries for parallel processing by splitting it into smaller queries that will be executed on Compute nodes. It also assigns sets of files to be processed by each Compute node.

Each small query is called a task and represents a distributed execution unit. It reads file(s) from storage, and joins results from other tasks, groups, or orders data retrieved from other tasks.

Azure Storage:

Synapse SQL uses Azure Storage to keep your user data safe. Since your data is stored and managed by Azure Storage, there's a separate charge for your storage consumption.

Control Node:

The Control node is the brain of the architecture. It's the front end that interacts with all applications and connections.

Compute Node:

The Compute nodes provide the computational power.

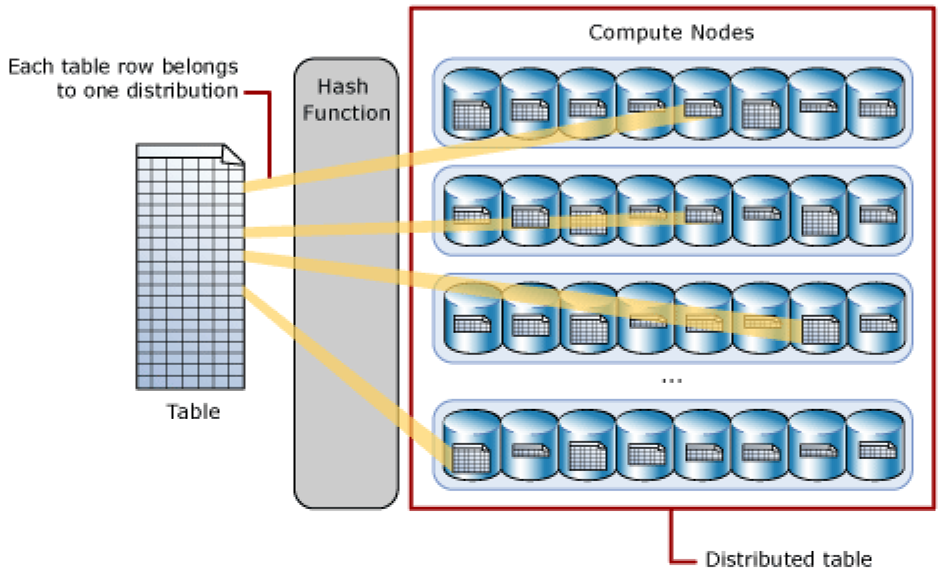
Distributions:

Distribution is the basic unit of storage and processing for parallel queries that run on distributed data in a dedicated SQL pool. When a dedicated SQL pool runs a query, the work is divided into 60 smaller queries that run in parallel.

Hash-distributed tables:

A hash distributed table can deliver the highest query performance for joins and aggregations on large tables.

To shard data into a hash-distributed table, dedicated SQL pool uses a hash function to deterministically assign each row to one distribution. In the table definition, one of the columns is designated as the distribution column. The hash function uses the values in the distribution column to assign each row to a distribution.



- Each row belongs to one distribution.
- A deterministic hash algorithm assigns each row to one distribution.
- The number of table rows per distribution varies as shown by the different sizes of tables.

There are performance considerations for the selection of a distribution column, such as distinctness, data skew, and the types of queries that run on the system.

Round-robin distributed tables

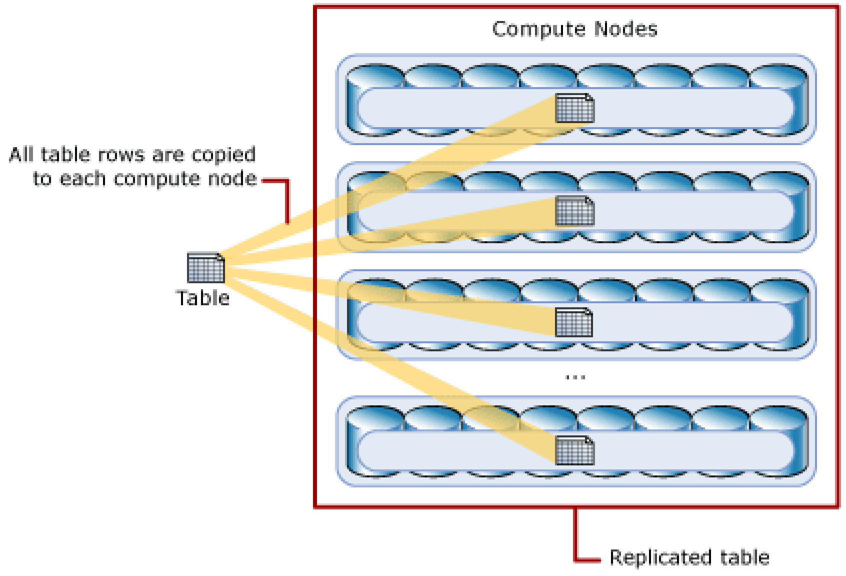
A round-robin table is the simplest table to create and delivers fast performance when used as a staging table for loads.

A round-robin distributed table distributes data evenly across the table but without any further optimization. A distribution is first chosen at random and then buffers of rows are assigned to distributions sequentially. It's quick to load data into a round-robin table, but query performance can often be better with hash-distributed tables. Joins on round-robin tables require reshuffling data, which takes extra time.

Replicated tables

A replicated table provides the fastest query performance for small tables.

A table that is replicated caches a full copy of the table on each compute node. So, replicating a table removes the need to transfer data among compute nodes before a join or aggregation. Replicated tables are best utilized with small tables. Extra storage is required and there's extra overhead that is incurred when writing data, which make large tables impractical.



> Images from Microsoft official documentation