AZURE SYNAPSE PLAYBOOK

Author: Marcelo Silva

Contents

[Introduction 3](#_Toc107384212)

[History 3](#_Toc107384213)

[Components 4](#_Toc107384214)

[Storage Options 7](#_Toc107384215)

[Feature Summary 9](#_Toc107384216)

[Dedicated SQL Pools 10](#_Toc107384217)

[Architecture 10](#_Toc107384218)

[Scaling 11](#_Toc107384219)

[Data Distribution 12](#_Toc107384220)

[Distributed Tables 13](#_Toc107384221)

[Replicated Tables 14](#_Toc107384222)

[Management Tools 18](#_Toc107384223)

[Load Performance Factors 18](#_Toc107384224)

[a. Where the data comes from. 18](#_Toc107384225)

[b. The indexes on the target table. 18](#_Toc107384226)

[Row Stores 19](#_Toc107384227)

[Column Stores 22](#_Toc107384228)

[c. The distribution strategy of the target table. 25](#_Toc107384229)

[d. The data load method. 25](#_Toc107384230)

[e. Whether the operation is minimally or fully logged. 27](#_Toc107384231)

[f. Processing capacity of the pool. 27](#_Toc107384232)

[g. Processing capacity of account doing the load. 27](#_Toc107384233)

[h. Concurrent operations at the destination. 30](#_Toc107384234)

[Things to pay attention to after the load. 30](#_Toc107384235)

[Ensure Statistics are created. 30](#_Toc107384236)

[Ensuring compression quality. 31](#_Toc107384237)

[BACKUPS and DR 32](#_Toc107384238)

[Summary of best practices 32](#_Toc107384239)

[Setting up access for dedicated pools 33](#_Toc107384240)

[Authentication Options 33](#_Toc107384241)

[Database Users 33](#_Toc107384242)

[Row-level Security 35](#_Toc107384243)

[Column-level Security 35](#_Toc107384244)

[Dynamic Data Masking 35](#_Toc107384245)

[Column-level Encryption 36](#_Toc107384246)

[Workspace-level Encryption 36](#_Toc107384247)

[Managing your Dedicated Pool 36](#_Toc107384248)

[Managing Activity 38](#_Toc107384249)

[Managing pool size 39](#_Toc107384250)

[Advanced Features 40](#_Toc107384251)

[Materialized Views 40](#_Toc107384252)

[Resultset Caching 41](#_Toc107384253)

[Workload Isolation 42](#_Toc107384254)

[Ordered Clustered Columnstore Indexes 42](#_Toc107384255)

[Unsupported features in dedicated pools 44](#_Toc107384256)

[Creating a Workspace 45](#_Toc107384257)

[Optional pre-requisite resources: 45](#_Toc107384258)

[User Input Parameters: 45](#_Toc107384259)

[Additional Settings 45](#_Toc107384260)

[Secondary Level Encryption 46](#_Toc107384261)

[Create a Dedicated Pool 47](#_Toc107384262)

[Firewall Configuration 48](#_Toc107384263)

[Private Endpoint 49](#_Toc107384264)

[Managed Private Endpoint 51](#_Toc107384265)

[Git Configuration 54](#_Toc107384266)

[SQL Serverless Pools 55](#_Toc107384267)

[Apache Spark Pools 56](#_Toc107384268)

[Audit 59](#_Toc107384269)

[Azure Defender 60](#_Toc107384270)

[Diagnostics 61](#_Toc107384271)

# Introduction

## History

This document provides information to help you utilize Azure Synapse Analytics resources effectively. It is not intended to be a comprehensive guide, our goal is to help readers understand key concepts, share useful tips, and provide references for further education.

Traditionally SQL Server has been a SMP (symmetric multi-processing) database engine, which is characterized by all work being done by a single compute node, although it may have multiple read-only nodes. In 2010 Microsoft released SQL Server PDW (Parallel Data Warehouse), introducing SQL to the MPP (massively parallel processing) space. MPPs are characterized for being truly distributed engines with a control node and multiple worker nodes which can run updates and reads and store different parts of data. PDW was sold as an appliance, or a group of servers, storage, power distribution, networking, and software (Windows + SQL) all combined into a rack.

In 2015 Microsoft started offering ADW (Azure Data Warehouse) as a PDW Paas offering in Azure. Microsoft was essentially hosting and managing the appliances along with running patching and backups. Microsoft also fixed the number of distributions as 60, meaning that every ADW database is distributed in 60 parts.

ADW sizes are measured in DWUs (data warehouse units). The smallest offering is 100 DWUs, which contains part of a control node and part of a compute node. The largest offering is 30,000 DWUs, which contains a control node and 60 compute nodes. Microsoft does not provide hardware details, but each compute node usually has 64+ vCores, 400+ gbs of memory and 1.5 TB local ultra-fast SSD cache.   
  
As a typical MPP, ADW was designed to handle a small number of large queries. Some of its main limitations were low limit of concurrent requests (128) and relatively high storage cost. One of its main competitors was SPARK, since it works natively with Data Lake Storage, offering more flexibility and cheaper storage. Data Lake Storage also offers interesting features not available in ADW such as automatic archival, automatic tiering, geo-redundancy, support for semi-structured data (JSON, XML), support for high compression formats (such a parquet) and ability to serve multiple ecosystems from a single copy (Synapse, HDInsight, Machine Learning, Databricks, Log Analytics and Data Explorer).

In 2020 ADW evolved into Synapse Analytics. At this point Synapse became a combination of some of the best big data technologies, including classic Azure Data Warehouse, pay per query Serverless SQL Pools, Data Lake, Data Factory and SPARK. The best term to define Synapse is hence a platform.

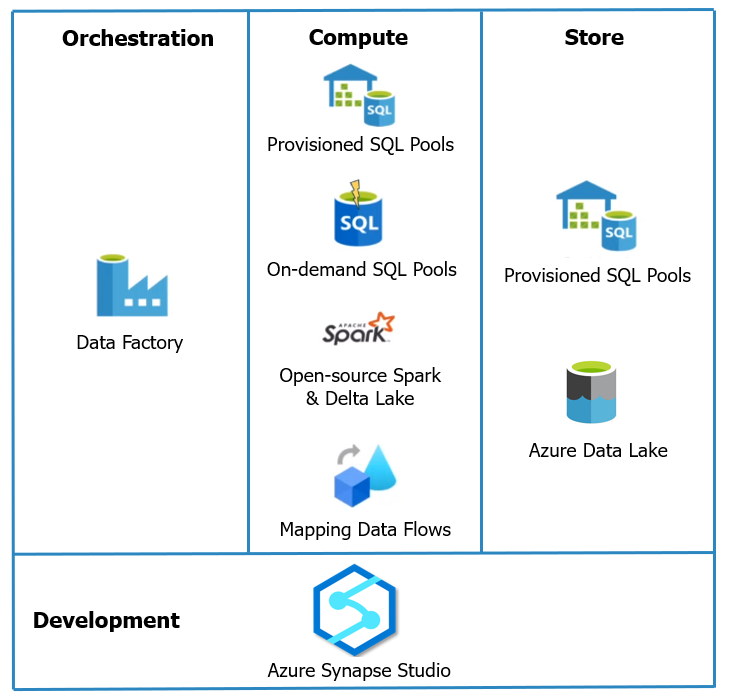
Timeline

Description automatically generated

## Components

The main components of Synapse are:

* **Workspace** – this is a placeholder in azure portal which contains all resources for a Synapse platform.
* **Synapse Studio** – this is a development/management tool in the azure portal which allows working with most features in the synapse platform.
* **Dedicated SQL Pool –** this is the azure data warehouse part of Synapse, which can vary from 1/5th of a node to 61 nodes, runs the distributed SQL engine, and stores data internally for best performance.
* **Data Lake Gen2** – this is where data may be stored before loading, or in the form or external tables.
* **Serverless SQL Pool** – this is like SQL pool, but can access only data from the lake, and is intended for exploratory or bursty workloads.
* **Spark Cluster** – this can run SPARK jobs and Jupiter notebooks. It can auto scale and auto terminate. It can access data in the lake directly and data in a dedicated pool indirectly (meaning it must go through the dedicated pool engine).
* **Pipelines** – Allow creating, running, and monitoring azure data factory pipelines to import and export data.
* **Mapping Data Flows** – Allow developing pipelines that use scale out Spark clusters.

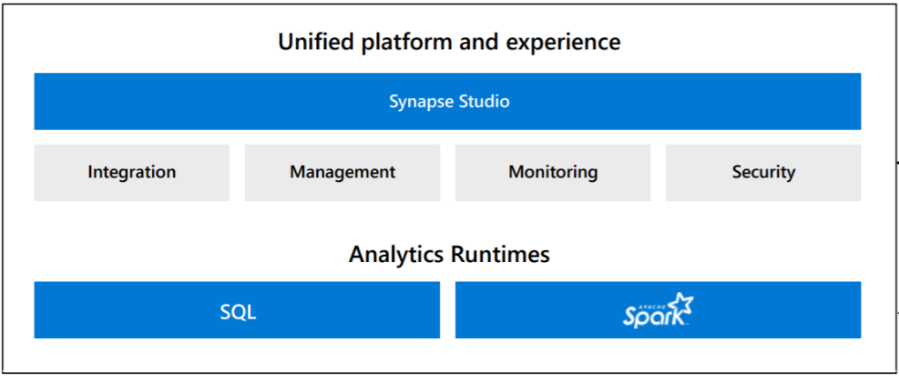


Connected Services:  
Graphical user interface, application

Description automatically generated with medium confidence

Keep in mind that each service is billed separately, serverless resources are charged by the time used. Storage is charged by TBs. Cool and archive storage tiers also charge for reads.

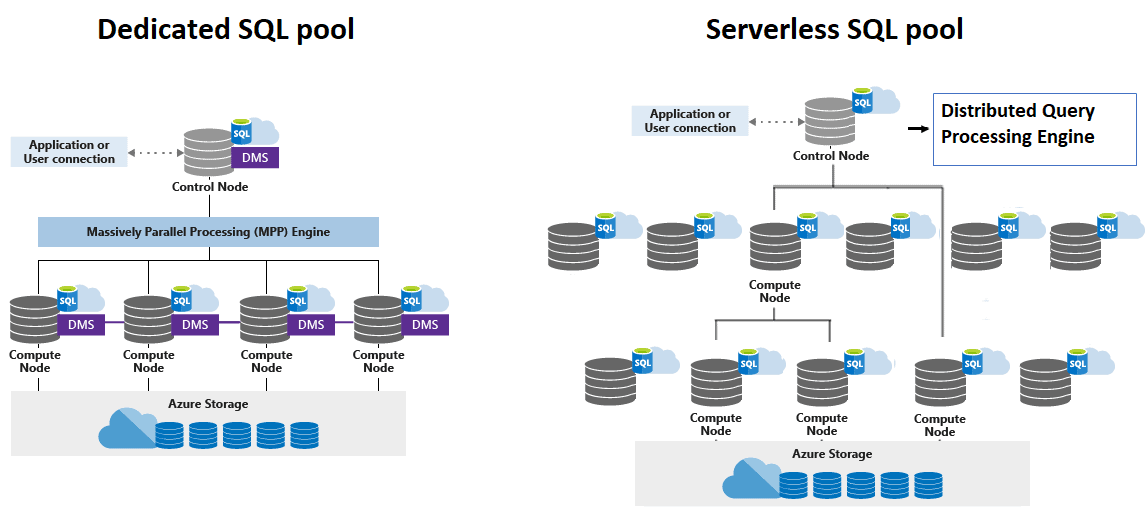
Synapse Studio is free and provides a unified platform experience:



Diagram

Description automatically generated

Dedicated pools are fixed (until scaled), Serverless pools allow auto scaling dynamically.



References:

[What is Azure Synapse Analytics? - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-gb/azure/synapse-analytics/overview-what-is)

[Analytics end-to-end with Azure Synapse - Azure Example Scenarios | Microsoft Docs](https://docs.microsoft.com/en-us/azure/architecture/example-scenario/dataplate2e/data-platform-end-to-end?tabs=portal)

## Storage Options

Synapse allows you to store data in 2 places: Azure Data Lake Gen 2 and Dedicated SQL Pool. The decision of where to store depends on the nature of the data and how it will be processed and consumed. SQL pools store data in relational tables much like SQL. The main challenges with those stores are inflexibility and scalability. Adding tables and columns usually requires more planning, coding, and testing. Also scaling is more limited and can get more expensive. In SMP SQL one is limited to a single writeable node and a few read only nodes. In Synapse SQL one can have up to 60 read/write nodes, while with data lake one could have thousands of nodes.

Data lake works over blob storage, and it supports files in any format (csv, json, parguet, avro, xml to name a few). This increases flexibility of client tools, but there is generally more work involved accessing the data. There is no such thing as schema free, one either defines the structure before writing or after reading. If the data is read many times in repeated ways, it’s preferable to store it in a well-defined schema. If one isn’t sure of how the data will be consumed, and cares mostly about saving it fast and cheap, open formats and blob storage are better suited.

For structured data and typical data warehouse workloads, we recommend storing it in Dedicated SQL Pools. The distributed SQL engine is best for frequent commands which may combine data from multiple tables (joins). If you plan to read the data from multiple environments (serverless synapse pool, machine learning, python, spark), or if your data is semi-structured, it makes more sense to store it in the lake. Beware that consuming data from the lake is typically slower, especially as external tables.

Even with recent delta lake enhancements, which include transaction support, time travel, schema evolution, processing data in the lake is typically harder than in database environments. For instance, delta supports transactions with only 1 table, schema evolution allows adding columns, but not changing types of removing columns. Time travel only goes as far as compaction, and if one does not compact storage requirements may increase drastically. Other challenges are the delta lake does not support constraints (primary, foreign, check), and that usually translates to more ETL code to ensure the data is consistent.

Keep in mind hadoop/spark/delta were created to support big data workloads with high velocity, volume and variety of data. If one is coming from SQL centric ETL environment, and rely heavily in stored procedures, constraints and traditional ETL tools like SSIS, reflect whether your workload has any of the high V needs, and be clear about what challenges your initiative is trying to mitigate, because refactoring the ETL code to SPARK, or simply converting routines from SQL to Synapse SQL is usually harder than one predicts. Finally take into consideration concurrency needs. SQL environments can handle thousands of simultaneous requests. A Synapse dedicated pool can handle at most 128 concurrent requests (with a large enough cluster). When it comes to data lake and spark, even a high concurrency cluster is limited to a few users, and when we have many clusters running at the same time we can run into bandwidth issues with storage accounts. There are fundamental differences between designing environments to handle a few large jobs, or to handle many small jobs. For instance, when increasing frequency of ETL from once a day to every few minutes, thus reducing the sizes of incremental jobs but increasing the number and frequency of commands, the nature of the ETL workload could shift from OLAP to OLTP, making is less suited for MPPs or Data Lakes. Columnar storage formats such as parquet were designed to store large volumes of static data. These formats rely on compression for performance, it turns out compression does not work well when the data gets updated/deleted. So columnar stores and near real time ETL have inherent incompatibilities.

In many situations hybrid approaches make more sense, combining SMP SQL, MPP SQL, Serverless SQL, data lake and Spark, levering the best that each stack offers. Synapse offers ease of integration of these technologies like no other product. Keep in mind you can also add Azure SQL, Managed Instance and SQL VMs to the mix, especially when you want to lift and shift and minimize refactoring.

A picture containing chart

Description automatically generated

## Feature Summary

A picture containing calendar

Description automatically generated

# Dedicated SQL Pools

## Architecture

Traditional relational database engines (such as SQL Server, Oracle, MySql, PostGress, DB2) use the SMP (symmetric multi-processing) architecture. In this architecture you may have readable secondary nodes, but all updates must go through the primary node. Also, any given query can be processed only 1 by node.

Example of a multi node SMP:

Diagram

Description automatically generated

Distributed data warehouse engines (such as Synapse Dedicated Pools, Teradata, Netezza, Snowflake and Vertica) use the MPP (massive parallel processing) Architecture, which is composed of a Control node and multiple Compute nodes. In MPPs all queries still must be sent to the control node, which does authentication (who is connecting), authorization (permissions evaluation), and optimization (build query plan). But the request gets split and sent to compute nodes for distributed execution. Also, updates may be sent directly to the compute nodes (in Synapse this is called Polybase).

Diagram

Description automatically generated

## Scaling

Clusters are sized in DWUs and every 500 DWUs add a compute node.

Graphical user interface

Description automatically generated

Every database is split into 60 distributions identical in schema. Distributions get evenly attached to compute nodes when the cluster starts. The higher the DWUs the more compute nodes are available and the fewer distributions each node will contain.

Timeline

Description automatically generated

The above cluster has 2000 DWUs, 4 compute nodes, and each node holds 15 distributions.

Each compute node has 64+ vCores, 400+ GB of RAM, and local NVme cache:  
A picture containing text, businesscard

Description automatically generated  
  
Timeline

Description automatically generated

## Data Distribution

The power of a Synapse pool comes from its distributed nature and its capacity to leverage multiple nodes to process a query (divide and conquer). Conversely the root of Synapse’s problems also comes from its distributed nature. When you send a query to Synapse, the control node does the following steps:

* 1. Parsing – separating the query elements.
  2. Binding – Linking tables, columns and routines to database schema.
  3. Security Evaluation – making sure the user has permissions on the objects.
  4. Optimization – picking a good execution plan based on the available indexes, filters, compute nodes and statistics.

If all goes well the control node sends smaller queries to the compute nodes. Problems arise when the compute nodes don’t have all the information needed to process the queries. For instance, when tables are joined Synapse checks if the data is properly aligned in the distributions based on the join condition. If the operation is not aligned Synapse usually copies the smaller table into all distributions. This operation is called a “Broadcast”, and it is managed by the DMS (Data Movement Service).

Aggregates may also trigger DMS operations, for instance when the hash distribution key is not contained in the group by keys.

Table

Description automatically generated with medium confidence

DMS operations can significantly increase execution times. There are 2 ways to prevent DMS:

1. Replicate a table.
2. Make the operation “distribution aligned” by hashing tables on right columns.

When you insert data Synapse has to decide how to split the rows in the distributions. A table can be: Distributed or Replicated.

### Distributed Tables

In a distributed table each row is stored into one distribution. The distribution may be on Round\_Robin or Hash basis. With Round\_Robin the rows are spread in a circular fashion, first row goes into partition 1, then partition 2, all the way to 60, then back to 1.

With Hash distribution you need to specify a column in which a hash function gets applied. The result determines which distribution gets the record. Hashes on a given value always yield the same result.

Round\_Robin tables allow for faster inserts because there is no hash function overhead, however they yield to more frequent DMS operations during reads. Round\_Robin is often used for staging tables, but if you are using a staging table to join another table (for instance with MERGE) then you are probably better off hashing the staging table on the same column as the target table.

The most important advice in this document is: **Design your tables in a way that reduce DMS operations for frequent large commands**. If necessary, denormalize data by repeating a column in many tables so that they all can be distributed on the same column, and make sure to use that column on join conditions.

The downside of hash distribution is to potentially split the rows unevenly, causing some distributions to have more rows than others, leading to a problem called “Distribution Skew”. To mitigate this issue chose a distribution column which has a lot of distinct values. It’s impractical to split the rows completely evenly. Even round robin tables end up with some level of skew.   
Icon

Description automatically generated with low confidence

-- Find data skew for a distributed table

DBCC PDW\_SHOWSPACEUSED('dbo.FactInternetSales');

Another option is to create the view vTableSizes from the link:

<https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-tables-overview#table-size-queries>

And run:

select \* from dbo.vTableSizes

where two\_part\_name in (

select two\_part\_name from dbo.vTableSizes where row\_count > 0 group by two\_part\_name

having (max(row\_count \* 1.000) - min(row\_count \* 1.000))/max(row\_count \* 1.000) >= .10

) order by two\_part\_name, row\_count

<https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-tables-distribute>

### Replicated Tables

This means the entire table is copied into the first distribution of each compute node. Theoretically all their joins would be distribution aligned and they would never cause DMS operations. However, when a replicated table is created, it is created as a round robin table at first. This round robin table is the source of truth for the table. A replicated table state is marked in the system catalog as ready or not. When you first create a table, or when you modify any record in it, the status is set to not-ready. Then first time you query the table, Synapse will use the round robin copy, and issue a broadcast operation in the background. Once broadcast is done the table is considered "ready", but is read-only. If any change is made the cached version is invalidated, Synapse falls back to the round robin version, and the whole process re-starts.

To prevent this issue, it is recommended to run “select top 1 from table” every time you modify a replicated table. **Replicated tables are well suited for small dimension tables frequently used in joins, which do not get updated very frequently, and which size does not exceed 2 GB.**

This query lists the replicated tables pending synchronization:

SELECT s.name [schema], t.name [table]

FROM sys.tables t

JOIN sys.schemas s on s.schema\_id = t.schema\_id

JOIN sys.pdw\_replicated\_table\_cache\_state c ON c.object\_id = t.object\_id

JOIN sys.pdw\_table\_distribution\_properties p ON p.object\_id = t.object\_id

WHERE p.[distribution\_policy\_desc] = 'REPLICATE' and c.[state] = 'NotReady'

<https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/design-guidance-for-replicated-tables>

Graphical user interface

Description automatically generated with low confidence

Graphical user interface, application

Description automatically generated

Diagram

Description automatically generated

Graphical user interface, application

Description automatically generated

Text

Description automatically generated with low confidence

<https://docs.microsoft.com/en-us/azure/synapse-analytics/sql/overview-architecture>

## Management Tools

* Azure Synapse Studio  
  <https://docs.microsoft.com/en-us/azure/synapse-analytics/sql/author-sql-script>
* SSMS   
  [Download SQL Server Management Studio (SSMS) - SQL Server Management Studio (SSMS) | Microsoft Docs](https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?redirectedfrom=MSDN&view=sql-server-ver15)
* Azure Data Studio.   
  [Azure Data Studio | Microsoft Azure](https://azure.microsoft.com/en-in/services/developer-tools/data-studio/)

## Load Performance Factors

The performance of loading data into a dedicated pool is affected by the following:

## Where the data comes from.

Small amounts of data may be imported directly from the from source systems (whether they come from on-premise or other cloud providers). As the amount of data grows, and bandwidth becomes a bottleneck, we recommend copying into azure first, typically into a storage account in the same region as Synapse. This approach is also useful when you need to import data multiple times.   
   
There are several tools for migrating your data to an Azure Storage blob. The easiest way is to use Azure Storage Explorer and copy files into the storage container.  
[Azure Storage Explorer – cloud storage management | Microsoft Azure](https://azure.microsoft.com/en-us/features/storage-explorer/)

Another option is to use the AzCopy command line utility, which is available for Windows and Linux.

https://docs.microsoft.com/en-us/azure/storage/common/storage-use-azcopy-v10

For very large data imports we recommend a data box, which is a highly secure, encrypted and tough external drive provided by Azure.   
  
<https://docs.microsoft.com/en-us/azure/databox/data-box-overview>

## ****The**** indexes on the target table.

Synapse dedicated pool allows multiple indexes per table. During runtime Synapse will decide which index to use depending on the query. Each index is stored independently and can be understood as a separate copy of the data. The more indexes you add to a table the slower ETL will be, but the more options Synapse will have to build faster execution plans.

There are 2 types of indexes in Synapse: Row Stores and Column Stores:  
Graphical user interface

Description automatically generated

### Row Stores

Row stores are the traditional storage mode for transactional databases. Rows are organized in 8kb pages. A page contains all columns of a record, and may contain multiple records depending on size. If a record has more than 8k bytes it gets stored in overflow or blob pages. When you select from a rowstore, even if you need only 1 column, the entire pages with all columns in them will be loaded from disk into memory.

Row stores can have 2 representations: Heaps and Btrees.

#### Heaps

This is when pages are stored unsorted and uncompressed. During a load synapse allocates empty pages, fills them up, then allocates more pages.   
Heaps can be useful for:  
 -Tables which do not require ordering of results.  
 -Small tables which require only a few data pages.  
 -Tables which do not require direct access to a row or a subset.

Example to create a heap:  
CREATE TABLE myTable ( id int NOT NULL, name varchar(20))   
WITH ( HEAP );

#### Btrees

This is when pages get stored sorted and following a tree like structure. Branch pages contain the key columns and pointers to other pages with other key ranges.

By default, Btree indexes use Page compression in Synapse, which is somewhat equivalent to doing pkzip at the page level. If a page contains a lot of repeated values then page compression will likely yield so good compression.

**Btrees are well suited for transactional workloads**, in which updates are frequent and/or most operations retrieve a small range of values. Btrees are also well suited for data extraction workloads in which most operations do “select \* from table” with no aggregates. Finally, btrees are well suited for key lookups such as “select \* from table where key=@val”. If your query has a selective filter, consider creating a rowstore index on the filtered column/s.

A btree index may of 2 types: Clustered and Non-Clustered.

Clustered Indexes  
The data is sorted by the index keys and all other columns are contained in the leaf level.  
Example to create a table with a clustered index:  
CREATE TABLE myTable (id int NOT NULL, name varchar(20), zipCode varchar(6))   
WITH ( CLUSTERED INDEX (id) );

Non-Clustered Indexes  
The data is sorted by the index keys. The leaf level contains the clustered index keys and the included columns.  
Example to create a non-clustered index:  
CREATE INDEX zipCodeIndex ON myTable (zipCode) INCLUDE (name);  
--id column is implicitly included as the clustered index

#### Clustered Index Design Principles:

**You can have only 1 clustered index per table so chose it wisely**. A CI is essentially the table itself and its columns define how the data is sorted. Moreover, your CI columns will be included in the leaf level of EVERY non clustered index, so keep it small!

1) Make it as narrow as possible, both in number of columns and in number of bytes.

2) Make it unique if possible. If a clustered index is not unique SQL will automatically add a hidden Row Identifier column with 6 bytes.

3) Avoid using columns that get updated.

4) Avoid using variable length columns (var\_anything) and give preference to fixed length (int, char, date).

Remember that strings carry collation overhead (case and accent sensitivity) for comparison and sorting, so give preference to numbers or dates.

5) Guids are fixed length and don't need collation, so they are a "happy medium". The problem with guids is that they are not sequential, you never know where in the btree your record will land, leading to high fragmentation. Assume your PK is on a guid, and you get 10 thousand records per day, in a few months your btree is wide and deep. Now if you need to get the new rows from yesterday, they will probably be spread all over the index, so SQL will need to bring a lot more pages to memory than if you CI was sequential. Give preference to INT/BIGINT, if you must use GUIDs use Sequential/Comb Guids.

6) A popular choice is using an ever-increasing column (for example identity), this helps keep fragmentation low, and reduce "hotspots" disk in disk.

7) Design your clustered index for the most frequent reads, so if your table has a column used very frequently in WHERE or GROUP BY, and this column is not updated, consider putting it on the left of your clustered index.

#### Non-Clustered Index Design Principles

**Attention:** Indexes are a double-edged sword, and while they may dramatically speed up some selects they also:

* Slow down insert, update, delete operations.
* Increase storage and memory utilization.
* Increase risk of logical contention including deadlocks.

**Symptoms of duplicated indexes**:

* Multiple indexes starting with the same columns.
* Non clustered indexes starting with the same column as the clustered index.

These are typically a consequence of following blindly missing index stats recommendations, or a poor clustered index choice. If you have a table with duplicated indexes, consider consolidating them into fewer indexes by combining some of them together, or consider changing the clustered index and deleting all duplicates.

If you have multiple columns in an index start with:

* The columns which are more frequently used in filters.
* The more selective columns, or columns with more distinct values.

### Column Stores

Columnstore is the default storage form in Synapse dedicated pools. In this form columns are stored separately in data structures called segments. Columnstores have some major advantages for some types of requests. To start the data can be highly compressed since all values on the same column have the same data type. Typically, columnstore reduces storage requirements 10 times.

The separation of storage into column segments allows for useful tricks during selects:

* Threads can be issued at the column level (higher parallelism).
* Operations may be done in several values at the same time instead of row by row (batch mode execution).
* Segments have headers (with min, max, sum depending on type), so Synapse could get the total value from the header rather than adding values individually (aggregate pushdown), and if we want to filter a value that is out of that min/max range Synapse can skip the entire segment (predicate pushdown).
* Returned columns which are not used in filters can be fetched late in the execution plan, after filters have been applied or keys passed join predicates (late materialization).
* Fast joins using Positions, which are essentially row ids at the column level which are later used to assemble the rows.

These features make columnstores well suited for analytical workloads, in which most queries fetch a subset of columns and aggregate results. Consider columnstores for fact or large dimension tables in a data warehouse.

Columnstores have some drawbacks:

* Good compression is achieved only with a high number of rows. Avoid columnstores in tables with less than 6 million rows, ideally, they should have 60 million and over. If you partition a table these numbers apply to each partition.
* Compression suffers when inserting few rows, so when using column stores perform inserts in BULK with batch sizes >= 100k.
* Updates and deletes have a strong negative impact on compression, avoid columnstores in tables with frequent updates. If you need to modify or delete a large number of rows, consider creating a new table and swapping it with the old.
* When multiple columns from a table are selected Synapse must reassemble the rows. The more columns returned the higher the reassembly cost.

**Note:** Synapse allows combining both column and rowstore indexes on the same table! If a table has various access patterns you may keep rowstore indexes to support seeks and columnstore indexes to support scans. During runtime Synapse will decide which structure to use depending on the query and the statistics available. Beware this will increase the data load times and storage cost!

Graphical user interface, application

Description automatically generated

Diagram

Description automatically generated

<https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-tables-index>

## The distribution strategy of the target table.

Please refer to the data distribution section in this document.

## The data load method.

In this section we will cover the different methods to load data for purposes of:

* **Initial Load** refers to importing data for the first time with INSERTS.
* **Incremental Load** refers to contiguous loads with INSERTS, UPDATES, DELETES.

**Initial Load**

The main methods to insert data are:

* Individual INSERTs
* BULK INSERT
* POLYBASE/COPY

The best method depends on:

* Whether you are loading into a column or row store.
* How much data you are loading.

Choose what makes sense for you your needs.

If you are loading a very small number of rows, it is perfectly fine to insert one row at the time with classic INSERT INTO VALUES commands. These are simple to build and allow for a single script to contain all the data and code. I use this method often when I import from excel, by creating a formula with the INSERT commands, then copy and paste into a SQL script.

Next, we have BULK INSERT, which should be the preferred method for small and medium tables. This method can be implemented easily with SSIS, Data Factory, Synapse Pipelines, among other ETL tools. Run the wizard, connect to source and target databases, pick your source tables, chose automatic column mapping, and let the engine do the work. This method runs fast enough for most needs and is less prone type conversion issues. If you are inserting a few million rows into a COLUM STORE you will benefit from using a large batch size, such 100 thousand, which will allow for better compression. If you know other users don’t need to read the table while you insert, and your operation is eligible for minimal logging (such as when the table is empty and has only 1 index, details on next section), then you will also benefit from using the TABLOCK hint.

COPY and POLYBASE are preferred for large tables with hundreds of millions of rows. These methods work by saving data to storage accounts first, typically in parquet, then bypassing the cluster control node and sending data directly into the compute nodes, allowing more parallelism. These methods can be orders of magnitude faster but are also more prone to conversion issues. To leverage these methods effectively a cluster should have at least 2 compute nodes (1000DWUs+).

Timeline

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated

Diagram

Description automatically generated with medium confidence

There are multiple ways to leverage Polybase:

* [PolyBase with T-SQL](https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/load-data-from-azure-blob-storage-using-copy) requires you to define external data objects.
* [Load data into Azure Synapse Analytics - Azure Data Factory & Azure Synapse | Microsoft Learn](https://learn.microsoft.com/en-us/azure/data-factory/load-azure-sql-data-warehouse?toc=%2Fazure%2Fsynapse-analytics%2Fsql-data-warehouse%2Ftoc.json&bc=%2Fazure%2Fsynapse-analytics%2Fsql-data-warehouse%2Fbreadcrumb%2Ftoc.json&tabs=data-factory)
* [PolyBase with SSIS](https://docs.microsoft.com/en-us/sql/integration-services/load-data-to-sql-data-warehouse?toc=/azure/synapse-analytics/sql-data-warehouse/toc.json&bc=/azure/synapse-analytics/sql-data-warehouse/breadcrumb/toc.json&view=azure-sqldw-latest&preserve-view=true) works well when your source is SQL Server. SSIS defines the source to destination table mappings, and orchestrates the load. If you already have SSIS packages, you can modify the packages to work with the new data warehouse destination.
* [PolyBase with Azure Databricks](https://docs.microsoft.com/en-us/azure/databricks/scenarios/databricks-extract-load-sql-data-warehouse?bc=%2fazure%2fsynapse-analytics%2fsql-data-warehouse%2fbreadcrumb%2ftoc.json&toc=%2fazure%2fsynapse-analytics%2fsql-data-warehouse%2ftoc.json) transfers data from a table to a Databricks dataframe and/or writes data from a Databricks dataframe to a table using PolyBase.

Despite PolyBase’ s popularity the [COPY statement](https://docs.microsoft.com/en-us/sql/t-sql/statements/copy-into-transact-sql?view=azure-sqldw-latest&preserve-view=true) is the preferred loading utility for large tables as it has additional loading capabilities that PolyBase does not provide.

Here is an example of a COPY INTO statement:

COPY INTO dbo.test\_1 (Col\_one default 'myStringDefault' 1, Col\_two default 1 3)

FROM 'https://myaccount.blob.core.windows.net/myblobcontainer/folder1/'

WITH ( FILE\_TYPE = 'CSV', CREDENTIAL = (IDENTITY= 'Storage Account Key', SECRET=''),   
FIELDQUOTE = '”', FIELDTERMINATOR = ',', ROWTERMINATOR = '0x0A', ENCODING = 'UTF8', FIRSTROW = 2 )

**Attention**: remember that column store tables have compression issues with small inserts. If you have a table with need for frequent small updates consider making it a clustered index (rowstore).

**Incremental Loads**Incremental loads tend to be smaller in volume but greater in complexity. The key to a successful incremental ETL strategy is to have a reliable and efficient change tracking mechanism at the source. Warehouses often need to read from live systems, and these reads are subject to transaction anomalies such as skipping rows or reading them more than once. Other common problems are maxing out IO/network/CPU at the source, connection drops and locking tables for too long. When a reliable change tracking mechanism is not available, we may need to resort to full truncate and reloads. This is often beyond the control of a business intelligence engineer and has a huge impact on the success of the data warehouse.

Data Factory has “change data capture” functionality, but it is limited to tracking pointers of the last successful load, and does not capture changes at the source by itself. If your source is a SQL Database you have a lot of options including cdc, triggers, rowversion and transactional replication.

Assuming you have a change tracking mechanism on the source system, a typical ETL case is to load recent changes into staging tables using bulk insert, then MERGE (insert, update, delete) with another target table.

If the target table is hash distributed, it’s good practice to distribute the stating table on the same column, to prevent shuffle.

If a target table is a column store, and a staging table has a lot of rows (such as 30% or more of total on target), it’s good practice to replace the existing table instead of merging, to prevent compression and fragmentation issues. A replacement consists of creating a new table, populating, dropping the original and renaming the new. If the target table is partitioned this can be accomplished with a partition swap. The same concept applies when you need to update a lot of rows.   
Example:



When the target table is large the join with staging can be slow, tips to mitigate:

* Consider using an ordered column store on the hash key (details on the advanced features of this document).
* Ensure that statistics are up to date on both source and staging tables. If you are doing proper maintenance stats on the target are likely accurate, but if you have auto\_update\_stats disabled, stats on the staging table could be inaccurate, so run “UPDATE STATISTICS STG\_Table” before merge.
* On the update make sure at least one column has a new value, to prevent unnecessary logging.
* Consider using specific scalar where predicates in addition to the join, so that synapse has more information to filter rows.

Example:  
 

Note: When you create surrogate keys during ETL and you don’t have the hash keys on the source data, this is often the case when adopting the medallion architecture (bronze, silver, gold). In such cases consider using non clustered indexes and/or replicated tables to support the joins.  
  
Synapse Merge has some limitations, for full details please refer to:  
[MERGE (Transact-SQL) - SQL Server | Microsoft Learn](https://learn.microsoft.com/en-us/sql/t-sql/statements/merge-transact-sql?view=azure-sqldw-latest&preserve-view=true#remarks)

## Whether the operation is minimally or fully logged.

Synapse dedicated pools are ACID compliant. In the event an error happens partially completed operations must be rolled back. When an operation is fully logged every record inserted/modified is stored in the transaction logs of each distribution. When an operation is minimally logged only the allocations of new extents are stored in the transaction log, making them much faster. Whenever possible use minimally logged constructs.

For example, this command is likely minimally logged:

Create table MyInternalTable as  
Select \* from MyExternalTable

While this command is likely fully logged:

Insert into MyInternalTable   
Select \* from MyExternalTable

**For more details please refer to:**<https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-develop-best-practices-transactions>

## Processing capacity of the pool.

Large imports may get throttled by hardware resources. As your jobs execute monitor resource utilization in the azure portal. If capacity limits are reached, consider scaling your cluster up during heavy load.

The overview page of the dedicated pool in the azure portal contains 2 very important graphs: DWU usage and Active and queued queries.

Graphical user interface, application

Description automatically generated

The graph on the left relates to resource utilization. The blue line shows the cluster size, and the orange line shows utilization. If the orange line is flattening at the top, it means you are reaching max capacity and you can benefit from scaling the cluster up.

The graph on the right relates to running queries. The blue column represents the queries running, while the orange column represents the queries waiting for resources to start executing. Ideally the orange column should be zero or low. **You can reduce the queued queries by scaling up the cluster or by reducing the resource class of the logins. The next section elaborates on this.**  
<https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-manage-compute-overview>

## Processing capacity of account doing the load.

SMP database engines are designed to handle hundreds if not thousands of concurrent small queries. During the optimization process the database engine estimates how much memory a query will require and gets it from the operating system (or from the db engine OS). Conversely MPP engines are designed to handle a low number of large queries. The engines lack an accurate way to estimate memory requirements, so limits are predefined by resource classes. Queries are run by database logins, and logins belong to resources classes. The amount of memory available for a process will depend on the largest resource class the login running the query belongs to.

Resource classes may be dynamic or static. Dynamic resource classes scale the limits according to the cluster size (DWUs). As the cluster grows/shrinks Synapse adjusts the memory limits. Dynamic resource classes sizes vary from smallrc to xlargerc. Static resource classes on the other hand always allow the same amount of memory regardless of cluster size. Static resource class sizes vary from staticrc10 to staticrc80.

Synapse can handle 1024 open sessions, but the number of queries Synapse can run in parallel (at the same time) is limited by the number of concurrency slots. With a 100 DWUs Synapse has 4 concurrency slots, and with 6,000 DWUs Synapse allows 128 concurrency slots, which is the maximum. Even if you scale up your cluster to 30,000 DWUs you still get 128 slots, but each slot will have more memory and CPUs. These numbers may seem low compared to SMP engines, but are actually high compared to Spark engines with similar compute capacity.

Finally, the number of concurrency slots a query takes depends on the resource class the login belongs to. The bigger the class the more slots a query consumes and the fewer the number of simultaneous queries synapse can handle. A cluster with 1000 DWUs can handle 32 queries (in parallel) from users in the small resource class, but only 4 if the users belong to the large resource class.

Chart, histogram

Description automatically generated

By knowing the ETL pattern one can estimate how much data will be loaded and how much memory will be needed and assign an adequate resource class for the data load user. Beware that running large inserts with low memory yields poor compression quality and poor read performance until the compression issues are mitigated.

Here is an example on how to create a user in the staticrc20 resource class, which is well suited for small to medium loads.

-- Connect to master

CREATE LOGIN LoaderRC20 WITH PASSWORD = 'a123STRONGpassword!';

-- Connect to the database

CREATE USER LoaderRC20 FOR LOGIN LoaderRC20;

GRANT CONTROL ON DATABASE::[cmspool] to LoaderRC20;

EXEC sp\_addrolemember 'staticrc20', 'LoaderRC20';

Table

Description automatically generated

Table

Description automatically generated

[Memory and concurrency limits - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/memory-concurrency-limits)

[Resource classes for workload management - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/resource-classes-for-workload-management)

## Concurrent operations at the destination.

On the previous topic we discussed the memory allocations implications of running multiple processes at the same time in Synapse. If possible, run your ETL during hours of low business activity.

Aside from memory and concurrency slots, running large update processes in parallel may lead to IO contention, as well as logical blocking contention between the processes. Even though your data is distributed, Synapse needs to ensure it is consistent, so all update operations are ACID compliant.

When multiple processes are loading data into the same table, they may get blocked by one another, taking longer to execute, and holding concurrency slots longer. Schedule your loads so that they don’t compete for locks in the destination tables.  
  
[Transactions (Azure Synapse Analytics) - SQL Server | Microsoft Docs](https://docs.microsoft.com/en-us/sql/t-sql/language-elements/transactions-sql-data-warehouse?view=aps-pdw-2016-au7)

# Things to pay attention to after the load.

## Ensure Statistics are created.

Synapse uses a cost-based query optimizer, so table statistics are essential to determine efficient query plans. Automatic statistics creation is enabled by default, but you may confirm with this query:  
SELECT name, is\_auto\_create\_stats\_on FROM sys.databases;

If disabled, you may reenable with:  
ALTER DATABASE <yourDb> SET AUTO\_CREATE\_STATISTICS ON;

However automatic statistics are created with unintuitive names with prefix WA\_Sys followed by hexadecimal digits. Some customers prefer creating statistics explicitly so they can have control of the names and how often to update.

To view existing statistics, you may run:

SELECT sm.[name] AS [schema\_name],

tb.[name] AS [table\_name],

co.[name] AS [stats\_column\_name],

st.[name] AS [stats\_name],

STATS\_DATE(st.[object\_id],st.[stats\_id]) AS [stats\_last\_updated\_date]

FROM sys.objects ob

JOIN sys.stats st ON ob.[object\_id] = st.[object\_id]

JOIN sys.stats\_columns sc ON st.[stats\_id] = sc.[stats\_id] AND st.[object\_id] = sc.[object\_id]

JOIN sys.columns co ON sc.[column\_id] = co.[column\_id] AND sc.[object\_id] = co.[object\_id]

JOIN sys.types ty ON co.[user\_type\_id] = ty.[user\_type\_id]

JOIN sys.tables tb ON co.[object\_id] = tb.[object\_id]

JOIN sys.schemas sm ON tb.[schema\_id] = sm.[schema\_id]

where st.user\_created=1

The syntax to create statistics is:  
create statistics [stats\_name] on [schema].[table].[column];

Or you may create statistics for all columns missing with this procedure.

  
execute dbo.[prc\_sqldw\_create\_stats] 1, null

It is a good practice to update statistics after significant changes, the syntax is:

--for all stats in a table

update statistics [schema].[table];

--for a specific statistic

update statistics [schema].[table]([statistic]);

Or you may update all statistics with this procedure.



execute dbo.[prc\_sqldw\_update\_stats]

<https://docs.microsoft.com/en-us/azure/synapse-analytics/sql/develop-tables-statistics>

## Ensuring compression quality.

For good performance it is essential that your column stores are well compressed. The below view can be created and used on your system to compute the average rows per row group and identify any sub-optimal compression.   


After creating the view this query returns poor quality segments.   
SELECT \*  
FROM [dbo].[vColumnstoreDensity]  
WHERE COMPRESSED\_rowgroup\_rows\_AVG < 100000  
OR INVISIBLE\_rowgroup\_rows\_AVG < 100000

The last column generates a SQL statement that can be used to rebuild your indexes, which syntax is:  
ALTER INDEX ALL ON [schema].[table] REBUILD;

For large tables you may replace the keyword ALL with an index name for more granular control, and if your table is partitioned, you may also specify a partition with:  
ALTER INDEX ALL ON [schema].[table] REBUILD PARTITION = X;

**Attention:** When you run ETL you need memory to compress the new data, but when you rebuild indexes you may need memory for the entire table, so **we recommend using a login with a resource class even higher than the one for ETL**. Keep in mind that only compressed rowgroups leverage the local cache at the compute nodes, so good compression is key for performance!  
  
[Adaptive caching powers Azure SQL Data Warehouse performance gains | Azure Blog and Updates | Microsoft Azure](https://azure.microsoft.com/en-us/blog/adaptive-caching-powers-azure-sql-data-warehouse-performance-gains/?ranMID=24542&ranEAID=TnL5HPStwNw&ranSiteID=TnL5HPStwNw-aJKdejxWSeLYkpLB3fpYYw&epi=TnL5HPStwNw-aJKdejxWSeLYkpLB3fpYYw&irgwc=1&OCID=AID2200057_aff_7593_1243925&tduid=%28ir__jgceodhmogkf62yav909ujn2mf2xt69r2ncprwp000%29%287593%29%281243925%29%28TnL5HPStwNw-aJKdejxWSeLYkpLB3fpYYw%29%28%29&irclickid=_jgceodhmogkf62yav909ujn2mf2xt69r2ncprwp000)

## BACKUPS and DR

Graphical user interface, text, table

Description automatically generated with medium confidence

Synapse performs a standard geo-backup once per day to a paired datacenter. The RPO for a geo-restore is 24 hours.

<https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/backup-and-restore>

## Summary of best practices

[Best practices for dedicated SQL pools - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/sql/best-practices-dedicated-sql-pool)

[Cheat sheet for dedicated SQL pool (formerly SQL DW) - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/cheat-sheet)

# Setting up access for dedicated pools

## Authentication Options

Authentication can be done with SQL or Active Directory (AD) accounts. A pre-requisite for AD Authentication is to configure and AD Administrator for the Synapse Workspace.

The admin can be any AD account or group.

<https://docs.microsoft.com/en-us/sql/relational-databases/security/authentication-access/create-a-login?view=sql-server-ver15>

<https://docs.microsoft.com/en-us/azure/azure-sql/database/authentication-aad-configure?tabs=azure-powershell>

## Database Users

To create a database user in a **dedicated pool**, we need 3 steps:

1. Create login at the server level, which may be an individual account or AD Group.
2. Create user in the databases from the login.
3. Grant Permissions

Example:

--PART 1 - server level logins / connect to master

--Example to create an AD user

CREATE LOGIN [login\_name@domain.com] FROM EXTERNAL PROVIDER

--Example to create an AD group

CREATE LOGIN [AD\_Group] FROM EXTERNAL PROVIDER

--PART 2 – database user / connect to your dedicated pool

create user [AD\_Group\_Or\_User] from login [AD\_Group\_Or\_User]

--PART 3 – grant access

--To grant permissions at the entire database level use database roles

--Grant admin permission on the database:

EXEC sp\_addrolemember 'db\_owner', 'AD\_Group\_Or\_User'

--Grant select permission on all user tables and views:

EXEC sp\_addrolemember 'db\_datareader', 'AD\_Group\_Or\_User'

--Grant insert/update/delete permission on all user tables:

EXEC sp\_addrolemember 'db\_datawriter', 'AD\_Group\_Or\_User'

For more granular access the major commands are:

Grant - assigns permissions.

Revoke - removes permissions.

Deny - Prohibits access, which takes precedence over any grant, so use carefully.

These commands are applied to a [principal], which can be a SQL User, AD User or AD Group.

The major operations are select, insert, delete, update, execute. You may also use the keyword "all", which will grant all operations applicable to the objects.

--To grant permissions at the schema level use

grant all on schema::[schema\_name] to [principal];

--To remove you may use

revoke all on schema::[schema\_name] to [principal];

--Deny example

deny update, delete on [schema].[table] to [principal];

**Note:** There is a special principal called public which represents all users. When you Grant/Revoke/Deny to public this will apply to everyone. To undo a deny run a grant.

--To grant permissions at the object level use

GRANT SELECT ON OBJECT::[schema].[object] TO [principal];

For **serverless pools** connect to master database and run:

CREATE LOGIN [AD\_GROUP\_Or\_User] FROM EXTERNAL PROVIDER;

For admin users also run:  
ALTER SERVER ROLE sysadmin ADD MEMBER [AD\_GROUP\_Or\_User];

**Note:** In order to allow users run queries against the data lake assign role “**Storage Blob Data Contributor**” to the same AD\_GROUP\_Or\_User.

References:  
[How to set up access control for your Synapse workspace - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/security/how-to-set-up-access-control)

<https://docs.microsoft.com/en-us/sql/t-sql/statements/grant-object-permissions-transact-sql?view=sql-server-ver15>

<https://docs.microsoft.com/en-us/sql/t-sql/statements/deny-transact-sql?view=sql-server-ver15>  
<https://docs.microsoft.com/en-us/sql/t-sql/statements/grant-schema-permissions-transact-sql?view=sql-server-ver15>

## Row-level Security

Row level security allows users to see only certain rows in a table/view/function.

The implementation requires creating a "SECURITY POLICY" and "inline table-valued function".

Example:  
  
[Row-Level Security - SQL Server | Microsoft Docs](https://docs.microsoft.com/en-us/sql/relational-databases/security/row-level-security?toc=%2Fazure%2Fsynapse-analytics%2Fsql-data-warehouse%2Ftoc.json&bc=%2Fazure%2Fsynapse-analytics%2Fsql-data-warehouse%2Fbreadcrumb%2Ftoc.json&view=sql-server-ver15)

## Column-level Security

Column level security allows users to see only certain columns in a table/view/function.

--To grant permission at the column level use

GRANT SELECT ON [schema].[object](ColumnA, ColumnB, ...) TO [principal];

If a user attempts to run "SELECT \* FROM [schema].[object]" he or she will get an error "SELECT permission was denied" on the columns lacking permission.  
  
[Column-level security for dedicated SQL pool - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/column-level-security)

## Dynamic Data Masking

Masking works at the column level by obfuscating the data before it is returned to the user.

Once you modify a column adding a mask, non admin users will no longer be able to see the data, regardless of which method they use to query. However, members of db\_owner will still be able to see the values unmasked, so beware of this limitation in dev environments. You may also grant privileges to some non admin users to unmask the data.

There are multiple masking functions suited for different data types such as email, phone, addresses, etc.

Even with masking enabled users are still able to find records with = or like operators.

Attached follows a comprehensive example of data masking in EDW, as well as how to mask all PII columns in the database.

  
[Dynamic data masking - Azure SQL Database | Microsoft Docs](https://docs.microsoft.com/en-us/azure/azure-sql/database/dynamic-data-masking-overview?toc=%2Fazure%2Fsynapse-analytics%2Fsql-data-warehouse%2Ftoc.json&bc=%2Fazure%2Fsynapse-analytics%2Fsql-data-warehouse%2Fbreadcrumb%2Ftoc.json&view=sql-server-ver15)

## Column-level Encryption

Column-level encryption (CLE) helps customers implement fine-grained protection of sensitive data within a table (server-side encryption). With CLE, customers gain the ability to use different protection keys for columns in tables with each key having its own access permissions. The data in CLE enforced columns is encrypted on disk (and remains encrypted in memory) until the DECRYPTBYKEY function is used to decrypt it.

[Encrypt a Column of Data - SQL Server & Azure Synapse Analytics & Azure SQL Database & SQL Managed Instance | Microsoft Docs](https://docs.microsoft.com/en-us/sql/relational-databases/security/encryption/encrypt-a-column-of-data?view=azure-sqldw-latest)

## Workspace-level Encryption

Workspaces can be configured to enable double encryption with a customer-managed key at the time of workspace creation. Enable double encryption using a customer-managed key on the "Security" tab when creating your new workspace. You can choose to enter a key identifier URI or select from a list of key vaults in the same region as the workspace. The Key Vault itself needs to have purge protection enabled.  
  
[Azure Synapse Analytics encryption - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/security/workspaces-encryption#:~:text=1%20Encryption%20of%20data%20at%20rest.%20A%20complete,in%20Synapse%20workspaces.%203%20Workspace%20encryption%20configuration.%20)

Managing your Dedicated Pool   
  
The azure portal shows utilization statistics for the past 7 days:

Graphical user interface, application

Description automatically generated

<https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-manage-compute-overview>

For longer term and more granular metrics you may leverage Azure Monitor:

[How to monitor Synapse Analytics using Azure Monitor - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/monitoring/how-to-monitor-using-azure-monitor)

Observe your pool to understand usage patterns and automate scale operations to meet peak usage. During long periods of zero utilization consider pausing your dedicated pool so you will be charged only for storage.

<https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-manage-compute-overview>

A picture containing table

Description automatically generated  
  
Text, letter

Description automatically generated

Diagram

Description automatically generated

## Managing Activity

This query returns the errors in the past 24 hours:  
select \* from sys.dm\_pdw\_errors

where create\_time > getdate()-1

order by create\_time desc

This query returns the requests currently running:  
select \* from sys.dm\_pdw\_exec\_requests

where status not in ('Completed','Failed')

order by submit\_time desc

If a query has been running for a long time it could be blocked, waiting for a resource grant, or waiting for a large DMS operation, here are some queries to help identify:

--queries blocked or waiting for a resource grant

SELECT waits.session\_id, waits.request\_id, requests.command,

requests.status, requests.start\_time, waits.type,

waits.state, waits.object\_type, waits.object\_name

FROM sys.dm\_pdw\_waits waits

JOIN sys.dm\_pdw\_exec\_requests requests ON waits.request\_id=requests.request\_id

WHERE status not in ('Completed','Failed')

ORDER BY waits.object\_name, waits.object\_type, waits.state;

To get the execution plan get the query text from one of the queries above and run:

EXPLAIN

QUERYTEXT;

To terminate a query, you may issue:

KILL REQUEST\_ID;

You may use LABELS OPTION to help identify queries such as:

SELECT \* FROM sys.tables OPTION (LABEL='MyLabel')

<https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-develop-label>

Other monitoring tips:  
[Troubleshooting dedicated SQL pool (formerly SQL DW) - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-troubleshoot#performance)  
  
<https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-troubleshoot>

<https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-manage-monitor#monitor-waiting-queries>

## Managing pool size

Here is how to scale the pool using TSQL while connected to master database:  
  
--to get the current size  
SELECT db.name [Database], ds.[Edition], ds.service\_objective   
FROM sys.database\_service\_objectives ds  
JOIN sys.databases db ON ds.database\_id = db.database\_id  
WHERE db.name = 'mySampleDataWarehouse';

Graphical user interface, text, application

Description automatically generated

**Beware that changing the service level causes the pool to be re-started.**

# **Advanced Features**

## Table Partitioning

This feature allows physically splitting a table horizontally. Assume we have an Orders table partitioned by Year, the table will look and behave like a single table, but each Year will have its own data structures.

This allows for several interesting features:

a) Partition elimination when a filter specifies which partitions are needed.

b) Join Collocation. When multiple tables are partitioned by the same column and are joined together, SQL will join the underlying partitions directly, allowing for higher parallelism.

c) Locking escalation from row to partition, instead of table, allowing for more concurrent updates in the table.

d) Instant bulk updates/purging by partition switching.

e) Applying different compression settings for each partition.

f) Rebuilding indexes at the partition level.

Features a-c allow some queries to run much faster.

Features d-f facilitate maintenance.

**Attention**: **Partitioning is often implemented improperly, causing more harm than good.**

Do not confuse Partitions with Distributions. The goal of distribution planning is to prevent shuffles, so distribute by columns used in joins and aggregates. The goal of Partition planning is to allow elimination, especially on user queries, but also on index maintenance and purge operations. In most cases a business date column is the best choice for partitioning, for it allows purging old data with partition truncation, and many queries naturally use business date filters. In some cases, it may make sense to partition by business segment, like state or division, which can reduce logical locking and allow more parallelism when processes apply partition filters. Almost never a column that is good for distribution is also good for partitioning.

Another common mistake is partitioning small tables or using too many partitions, ideally each partition should have 60+ million rows for column stores, and 10+ million records for row stores.

Graphical user interface, text, application

Description automatically generated

[Partitioning tables - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-tables-partition)

## Materialized Views

In Synapse a view behaves like a CTE or named query. When you query from a view Synapse will compile the query at runtime and build an execution plan merging the view definition and the rest of the query. An indexed view however works like a table. It may be distributed or replicated independent of the base tables, and it may have column and rowstore indexes.

If you query from an indexed view Synapse will build an execution plan using the indexes on the view instead of the underlying tables. Moreover, if you run queries against the underlying tables, and Synapse detects that the indexed view can satisfy the query, then Synapse may also use the indexed view.

Indexed views offer much greater flexibility than indexes, since they can have joins, complex filters, and aggregates. Performance gains from querying indexed views may also be much greater than combining indexes in underlying tables. However, performance disadvantages on insert/update/delete/merge commands can also be much greater, and so the risk for logical contention and deadlocks.

A decision to use an indexed view should consider the frequency of updates and the complexity of the views. Indexed views work well for infrequent updates. Be very careful placing indexed views if it contains any table with frequent updates. Also be very careful if the logic in the view is complex because Synapse may not have an accurate way to assert which records in the view need to be refreshed and may end up reprocessing more than necessary.

Another drawback of indexed views is that they place schema constraints in the base tables, which block some DDL operations.  
  
[Performance tune with materialized views - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/performance-tuning-materialized-views)

## Resultset Caching

Turning this feature on essentially instructs Synapse to store the results from nearly every query in the control node, exceptions are listed in the reference below. When a new query is received Synapse checks if there is already a table in the cache for the same command, then checks if the new user has the same permissions as the one who originated the cache, if both are true then Synapse simply returns the data from the cache, saving compute effort. Queries satisfied by the cache also do not take concurrency slots, which can allow Synapse to handle thousands of queries in parallel. The maximum size of result set cache is 1 TB per database and cache eviction is managed automatically. Finally, if a table has any data or schema change then all tables in the cache which reference this table will be invalidated.   
  
[Performance tuning with result set caching - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/performance-tuning-result-set-caching)

## Workload Isolation

Workload groups allow reserving resources exclusively to certain users, preventing users from taking too many resources, and preventing queries from running for too long. The control is done by the attributes:

* MIN\_PERCENTAGE\_RESOURCE – this value is reserved for the group, even if it has no active queries, so be careful with setting this value high.
* MAX\_PERCENTAGE\_RESOURCE – this is the upper boundary all users in the group can allocate.
* QUERY\_EXECUTION\_TIMEOUT\_SEC – this is the maximum number of seconds a query is allowed to execute.

To implement this feature, we start by creating a workload group such as:

CREATE WORKLOAD GROUP DataLoads WITH (

MIN\_PERCENTAGE\_RESOURCE = 10

,CAP\_PERCENTAGE\_RESOURCE = 40

, QUERY\_EXECUTION\_TIMEOUT\_SEC =600);

Then we assign users using a classifier such as:

CREATE WORKLOAD CLASSIFIER [wgcELTLogin]

WITH (WORKLOAD\_GROUP = 'DataLoads',MEMBERNAME = 'ELTLogin');

To cleanup resources we do:

DROP WORKLOAD CLASSIFIER [wgcELTLogin]

DROP WORKLOAD GROUP [DataLoads]

And to view settings we query:

--Workload groups

SELECT \* FROM sys.workload\_management\_workload\_groups

--Workload classifiers

SELECT \* FROM sys.workload\_management\_workload\_classifiers

--Run-time values

SELECT \* FROM sys.dm\_workload\_management\_workload\_groups\_stats

[Workload isolation - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-workload-isolation)

## Ordered Clustered Columnstore Indexes

If a table uses a rowstore representation, a clustered index automatically determines the sort order of the data. However, if a table uses a column store representation, by default the index builder does not sort the data before building column segments. As consequence segments with overlapping ranges can occur, which limits the effectiveness of predicate/aggregate pushdown operations. Ordered clustered columnstore indexes come to help mitigate this issue. They do not **eliminate** overlapping ranges but help reduce them. The higher the resource class you utilize for the index build, and the more DWUs, the more memory will be available and the fewer segment overlaps will occur. Beware that turning this on not only slows down index rebuilds but also slows down ETL.

To enable this feature, add the “ORDER” keyword to the create index command, example:

**CREATE CLUSTERED COLUMNSTORE INDEX CCO\_schema\_table ON [schema].[table] ORDER (Col\_A, Col\_B);**

create table **[schema].[table]** (

id bigint,

date datetime,

)

with (distribution = hash (id)

, CLUSTERED COLUMNSTORE INDEX ORDER (id)

)

Finally, the more threads used the more overlaps will occur. To reduce overlaps, add “OPTION (MAXDOP 1)” to the command above.

**Note: This feature can greatly increase the efficiency of joins and merges when large tables are ordered by join columns. It is ok to sort and hash distribute on the same column. The concept has similarities with clustered indexing and zordering.**

[Performance tuning with ordered clustered columnstore index - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/performance-tuning-ordered-cci)

## Unsupported Features in Dedicated SQL Pools

* Linked servers
* Cross server queries
* Cross database queries
* Triggers
* Foreign Keys
* Sequences
* Global temporary tables
* Indexes with included columns
* Filtered indexes
* Values table construct
* Rowstore indexes with more than 16 columns or 900 bytes
* Most query hints
* Default constraints with system functions such as getdate(), user\_name()
* Unsupported data types: 'geography', 'geometry', 'hierarchyid', 'image', 'text', 'ntext', 'sql\_variant', 'xml'.
* Primary Keys and Unique Constraints are supported but not enforced, so duplicates can occur.  
  <https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-table-constraints>
* Identity Columns do not guarantee the order in which values are allocated.  
  <https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-tables-identity>
* Blob columns (anything(max)) are not supported in column stores, so the tables must be defined as row store. The consequences are:
  + Compression will be poor (limited to PAGE).
  + Max size of 60 TB per table and 240 TB per dedicated pool.
  + These structures do not take advantage of local cache in the compute nodes.

Warning: Frequent queries over large tables containing BLOB columns (anything(max)) can severely affect overall performance of dedicated pools. Consider architecting your application in a way that stores blobs externally.

More details:  
[T-SQL feature in Synapse SQL pool - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/sql/overview-features)  
<https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-tables-data-types>  
<https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-service-capacity-limits>

# Creating a Workspace

## Optional pre-requisite resources:

1. Storage account with hierarchical namespaces for the data lake (one may be created with Synapse).
2. If secondary level encryption is desired, we need a Keyvault store with a key.

Storage account without hierarchical namespaces for the audit logs.

1. Email account to receive alerts and scan results.
2. VNet and Subnet for Private Endpoint.

## User Input Parameters:

* Subscription
* Resource Group
* Region
* Workspace Name
* Data Lake Storage Name (this can get automatically created or use an existing one)
* File System Name (this can get automatically created or use an existing one)
* Admin Account
* Admin Password
* AAD Admin Account Group
* Tags

## Additional Settings

These are our recommendations for Synapse network setup.

|  |  |  |
| --- | --- | --- |
| **Setting** | **Value** | **Comment** |
| Allow all Azure Services to Connect | No | Setting this to Yes would allow Paas resources on other subscriptions to connect. |
| Allow Connections from all IPs | No | Setting this to Yes would allow any IP on the internet to connect (provided they had credentials). |
| Enable Managed Virtual Network | Y | This ensures your workspace network is isolated from other workspaces and saves you from having to manage subnets and NSG rules for the workspace. [Managed virtual network - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/security/synapse-workspace-managed-vnet) |
| Allow outbound traffic only to approved targets? | Y | This helps prevention against data exfiltration by validating the target before sending data. [Create a workspace with data exfiltration protection enabled - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/security/how-to-create-a-workspace-with-data-exfiltration-protection) |
| Enable Second Level Encryption? | ? | This provides additional encryption with a customer managed key. Beware this carries a compute overhead of about 5% and increases management complexity. <https://docs.microsoft.com/en-us/azure/synapse-analytics/security/workspaces-encryption#key-access-and-workspace-activation> |
| Allow access from pipelines running as system assigned identity? | Y |  |
| Enable Auditing | Y | Activity on the dedicated SQL pools will be saved in an audit log. |
| Enable Azure Defender for SQL | Y |  |
| Private Endpoint | Y | Private endpoints connect your virtual network to Azure services without a public IP address at the source or destination. By mapping private endpoints to Azure Synapse workspaces, you can reduce data leakage risks. |

More details below.

## Secondary Level Encryption

If you create the workspace with secondary encryption, you will receive a message in the portal saying the workspace must be activated.

1. The first step is to copy the Managed Entity Object ID from the Overview page.  
   Graphical user interface, text, application, email

   Description automatically generated
2. Next go into the key vault Access Policies then Add Access Policy.

Graphical user interface, application

Description automatically generated

1. Under Key Permissions add Get, Unwrap Key, Wrap Key.
2. Click “Select Principal”, then paste the managed identity id

Graphical user interface, application

Description automatically generated

1. Click “Select”, click “Add”, then “Save”.
2. Go back to Synapse overview and hit “click here” in the message.

## Create a Dedicated Pool

Once the workspace is created you can create a dedicated SQL Pool on :

Graphical user interface, text, application

Description automatically generated

Specify the pool name and initial size set initial size to DW100c. You can later scale up the pool as needed. You may create the pool as copy of an existing pool restore point. The default collation is Case Insensitive + Accent Sensitive (a = A ≠ á). Change the collation if desired.

<https://docs.microsoft.com/en-us/azure/synapse-analytics/quickstart-create-sql-pool-portal>

## Firewall Configuration

Open your Synapse workspace in the Azure portal, go on the firewall section, ensure that “Allow Azure services and resources to access this workspace” is turned off, click “+ Add client IP”, then save.

Now let’s perform 2 tests:

1. Connect to the dedicated pool using SSMS or Azure Data Studio.   
   Graphical user interface, text, application, email

   Description automatically generated

Enter pool URL, admin account and password, and click connect:

Graphical user interface, text, application, email

Description automatically generated

1. Connect to Synapse studio. On the workspace page click on SQL Pools, then click on the dedicated pool name  
   Graphical user interface, text, application

   Description automatically generated

Then click on “Launch Synapse Studio”. If you get an error copy the IP address from the message, add it to the workspace firewall, then retry.

Graphical user interface, text, application

Description automatically generated

These steps are needed only during the initial setup, further connections should give preference to private endpoints.

[Configure IP firewall rules - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/security/synapse-workspace-ip-firewall#:~:text=There%20are%20two%20ways%20IP%20firewall%20rules%20are,it%20a%20name%2C%20Start%20IP%2C%20and%20End%20IP.)

## Private Endpoint

A private endpoint is a special network interface which allows injecting an Azure Paas service (Synapse, AzureSQL, CosmosDB, Storage, etc.) in your virtual network. When you use private endpoints, the traffic between your Paas Service and other azure resources traverses entirely over the Microsoft backbone network, not going over the internet.

A private endpoint has 2 main elements: on one end we have a Paas resource and on the other end we have a private IP inside a subnet. The key concept to understand a private endpoint is that it “brings” a Paas resource into your vNet, so all resources on that vNet can connect to the Paas service as if it were in the local network. Your infra structure may have multiple resources which want to connect to Synapse, depending on where they are placed you may need to setup multiple private endpoints.

In this example we will create a private endpoint attached to a Synapse workspace and allow azure resources on our default vNet/Subnet to connect to Synapse.

Navigate to your workspace, click on Private endpoints then +:  
Graphical user interface, application

Description automatically generated

On the basics tab enter the resource group, name and region:  
Graphical user interface, text, application, email

Description automatically generated

On the resource tab pick your synapse workspace.

Graphical user interface, text, application, email

Description automatically generated

The target sub-resource has 3 options:

1. SQL – to connect to the dedicated pool
2. SQLOnDemand – to connect to the serverless pool
3. Dev – to connect to Synapse studio

On Configuration tab pick the vNet/Subnet. Make sure to enable “Integrate with private DNS zone”.  
Graphical user interface, text, application, email

Description automatically generated

On the next tab you may add tags, then finally click Create. After a few minutes, the private endpoint will be created and approved, and then the resources on the default subnet above will be able to connect to Synapse dedicated pool. To connect to Serverless and Synapse studio more endpoints will be needed.

**To allow connection TO your synapse workspace create private endpoints.**

[Connect to a Synapse workspace using private links - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/security/how-to-connect-to-workspace-with-private-links)

## Managed Private Endpoint

These are private endpoints created in the Synapse Managed Virtual Network which allow connection FROM Synapse TO OTHER Paas Services.

In this example we will create a managed private endpoint to allow spark pools (in the synapse workspace) to read files on the storage account.

First launch Synapse Studio.

Next open Settings / Managed private endpoints. There you will find 2 endpoints automatically setup during workspace creation.  
Graphical user interface, text, application, email

Description automatically generated

Next select +New, then Data Lake Storage 2, enter a name, subscription, pick the workspace name from the drop-down menu, select target sub-resource as SQL, and click create:  
Graphical user interface, text, application, email

Description automatically generated

The endpoint will show as provisioning for a few minutes, then will have pending approval status.   
Graphical user interface, text, application, email

Description automatically generated

Next click on the name of the new pending endpoint, which will open the screen below, so you can click on the manage approvals link:

Graphical user interface, text, application, email

Description automatically generated

Once in the storage account overview page click on Capabilities then Private Endpoints:  
Graphical user interface, application

Description automatically generated

Finally select the endpoint and click approve.   
Graphical user interface, application

Description automatically generated

After a few minutes the endpoint will show as approved.  
Graphical user interface, text, application

Description automatically generated

To allow Synapse to connect to other Paas services such as CosmosDB,

<https://docs.microsoft.com/en-us/azure/synapse-analytics/security/how-to-create-managed-private-endpoints>

## Git Configuration

By default, Synapse Studio authors directly against the Synapse service. We recommend using source control instead.

Open Synapse Studio, click on Manage / Git Configuration / Configure:

Graphical user interface, text, application, email

Description automatically generated

You may pick Azure DevOps or GitHub, you should have been pre-configured.  
  
[Source control in Synapse Studio - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/cicd/source-control)

## SQL Serverless Pools

Serverless SQL pools allow querying data in the Azure Data Lake, [Cosmos DB](https://docs.microsoft.com/en-us/azure/synapse-analytics/sql/query-cosmos-db-analytical-store?toc=/azure/synapse-analytics/toc.json&bc=/azure/synapse-analytics/breadcrumb/toc.json&tabs=openrowset-key), or Dataverse.

* Supports only external tables from files in storage account, formats: parquet, delta, delimited text, JSON.
* Supports views and functions to encapsulate logic and enhance security.
* Cannot access data in dedicated pools.
* It has no local storage, only metadata objects are stored in databases.
* No support for local tables, triggers, materialized views and DML.
* Supports SQL and AD authentication.
* Supports OPENROWSET with a lot of extra features compared to dedicated pools.
* Only 1 serverless pool per workspace. Create multiple workspaces for increased concurrency.
* Auto scale, charged on a per query basis. Does not charge compute, only storage used to load external files and temporary structures needed to process query. $5 per TB. Each query charges a minimum of 10 MB.

Overview  
<https://docs.microsoft.com/en-us/azure/synapse-analytics/sql/on-demand-workspace-overview>

Getting Started  
[Quickstart: Use serverless SQL pool - Azure Synapse Analytics | Microsoft Learn](https://learn.microsoft.com/en-us/azure/synapse-analytics/quickstart-serverless-sql-pool)  
<https://docs.microsoft.com/en-us/azure/synapse-analytics/get-started-analyze-sql-on-demand>

Creating views and external tables  
[Create and use views in serverless SQL pool - Azure Synapse Analytics | Microsoft Learn](https://learn.microsoft.com/en-us/azure/synapse-analytics/sql/create-use-views)  
[Create and use external tables in Synapse SQL pool - Azure Synapse Analytics | Microsoft Learn](https://learn.microsoft.com/en-us/azure/synapse-analytics/sql/create-use-external-tables)  
Configuring access  
[SQL Authentication in Azure Synapse Analytics - Azure Synapse Analytics | Microsoft Learn](https://learn.microsoft.com/en-us/azure/synapse-analytics/sql/sql-authentication?tabs=serverless)

Best Practices  
<https://docs.microsoft.com/en-us/azure/synapse-analytics/sql/best-practices-serverless-sql-pool>

Self Help  
[Serverless SQL pool self-help - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/sql/resources-self-help-sql-on-demand)

Querying  
[Query data storage with serverless SQL pool - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/sql/query-data-storage#query-multiple-files-or-folders)  
[Query Delta Lake format using serverless SQL pool (preview) - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/sql/query-delta-lake-format)

Saving Results  
[Store query results from serverless SQL pool - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/sql/create-external-table-as-select)

Troubleshooting common problems

[Serverless SQL pool self-help - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/sql/resources-self-help-sql-on-demand#delta-lake)

Cost Management  
<https://docs.microsoft.com/en-us/azure/synapse-analytics/sql/data-processed>

## Apache Spark Pools

Spark is an optional analytics engine that can provide additional functionality to your Synapse workspace. Spark offers serval rich libraries: Spark SQL, Spark Streaming, MLlib and GraphX

You can also develop solutions in Scala, Python, Spark SQL, .NET or R using language bindings. Spark pools are a provisioned service, charged by the minute basis, and can be automatically started up and paused when the pool is inactive.

Before creating a spark cluster, we need to make sure it has access to the storage account.

If the storage account was created by the workspace, then permissions should already have been set. Otherwise, we need to grant with these steps:

1. Go to the storage account, click on Access Control, then View access to this resource.
2. Search by the workspace name and make sure it has role “Storage Blob Data Contributor”  
   Graphical user interface, text, application, email

   Description automatically generated
3. In case it doesn’t then add permissions:

Graphical user interface, application

Description automatically generated   
Graphical user interface, text, application, email

Description automatically generated

Next you can go into Apache Spark pools and click “+ New”

Graphical user interface, text, application

Description automatically generated

Select a node size and max number of nodes:  
Graphical user interface, text, application, email

Description automatically generated

Next select the auto pausing interval and spark settings. Spark instances are charged per minute.

Graphical user interface, text, application, email

Description automatically generated

Add tags then click Create.

Overview  
[Apache Spark in Azure Synapse Analytics overview - Azure Synapse Analytics | Microsoft Learn](https://learn.microsoft.com/en-us/azure/synapse-analytics/spark/apache-spark-overview)  
[Apache Spark pool concepts - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/spark/apache-spark-pool-configurations)

Getting Started

[Quickstart: Create a serverless Apache Spark pool using the Azure portal - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/quickstart-create-apache-spark-pool-portal)

Spark Jobs   
[Tutorial: Create Apache Spark job definition in Synapse Studio - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/spark/apache-spark-job-definitions)

Additional libraries  
[Manage Python libraries for Apache Spark - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/spark/apache-spark-manage-python-packages)

Monitor using Spark UI and Livy   
[Monitor Apache Spark applications using Synapse Studio - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/monitoring/apache-spark-applications)

Improving performance with indexing   
[Hyperspace indexes for Apache Spark - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/spark/apache-spark-performance-hyperspace?pivots=programming-language-python)

Delta Lake   
[Overview of how to use Linux Foundation Delta Lake in Apache Spark for Azure Synapse Analytics –](https://docs.microsoft.com/en-us/azure/synapse-analytics/spark/apache-spark-delta-lake-overview?pivots=programming-language-python)

## Audit

Auditing track database events and writes them to log. This can help understand activity and meet compliance standards.

Logs can be saved at:

* Storage – allows for long term storage.
* Log Analytics – allows for storage, ad hoc querying and has built in reports.
* Event hubs – allows consuming events into other platforms such as PowerBI, stream analytics and CosmosDB.

Open your Synapse workspace, go on Azure SQL Auditing, enable auditing. We recommend logging to both storage account and log analytics. Event hubs require more intricate implementation.  
Graphical user interface, application, email

Description automatically generated

[Azure SQL Auditing for Azure SQL Database and Azure Synapse Analytics - Azure SQL Database | Microsoft Docs](https://docs.microsoft.com/en-us/azure/azure-sql/database/auditing-overview)

## Azure Defender

Azure Defender provides:

* SQL Vulnerability Assessment – a service to identify potential vulnerabilities.
* Advanced Threat Protection – a service that detects anomalous activities such as SQL injection and password scan attempts.

We highly recommend enabling this on every environment which contain sensitive data. To do so

open your Synapse workspace, go on Azure Defender for SQL, enable the option, select a storage account for logs, enable periodic scans, enter the email account for notification and click save.   
Graphical user interface, text, application, email

Description automatically generated

[Azure Defender for SQL - Azure SQL Database | Microsoft Docs](https://docs.microsoft.com/en-us/azure/azure-sql/database/azure-defender-for-sql#:~:text=Azure%20Defender%20is%20available%20for%20Azure%20SQL%20Database%2C,that%20could%20indicate%20a%20threat%20to%20your%20database.)

## Diagnostics

Diagnostic settings are used to export logs and metrics to the destination of your choice. These diagnostics can be highly useful to understand utilization patterns and troubleshoot issues. We recommend you enable all Diagnostics and send them to both storage account and log analytics.

Open your Synapse workspace, go on Diagnostic settings and click “+ Add diagnostic setting”  
Graphical user interface, application

Description automatically generated  
Enable all events, set the retention period, select the log analytics workspace and storage account for long term storage.  
Graphical user interface, text, application, email

Description automatically generated

[Create diagnostic settings to send platform logs and metrics to different destinations - Azure Monitor | Microsoft Docs](https://docs.microsoft.com/en-us/azure/azure-monitor/essentials/diagnostic-settings?WT.mc_id=Portal-Microsoft_Azure_Monitoring&tabs=CMD)