Azure Synapse Analytics

Azure Synapse Analytics

- **Azure Synapse is an enterprise analytics service that accelerates,**
  - Time to insight across data warehouses
  - Big data systems

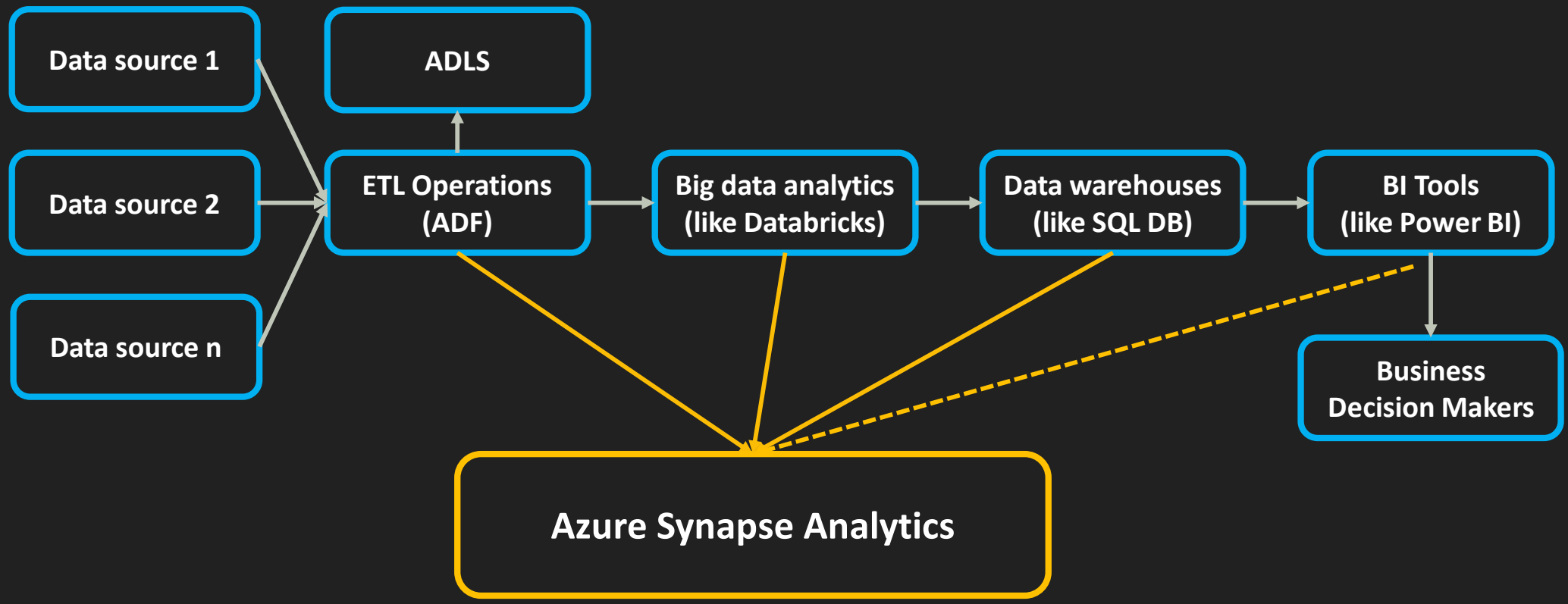
Azure Synapse Analytics

- **Azure Synapse brings together,**
  - The best of SQL technologies used in enterprise data warehousing
  - Spark technologies used for big data
  - Data Explorer for log and time series analytics
  - Pipelines for data integration and ETL/ELT
  - Deep integration with other Azure services such as Power BI, CosmosDB, and AzureML

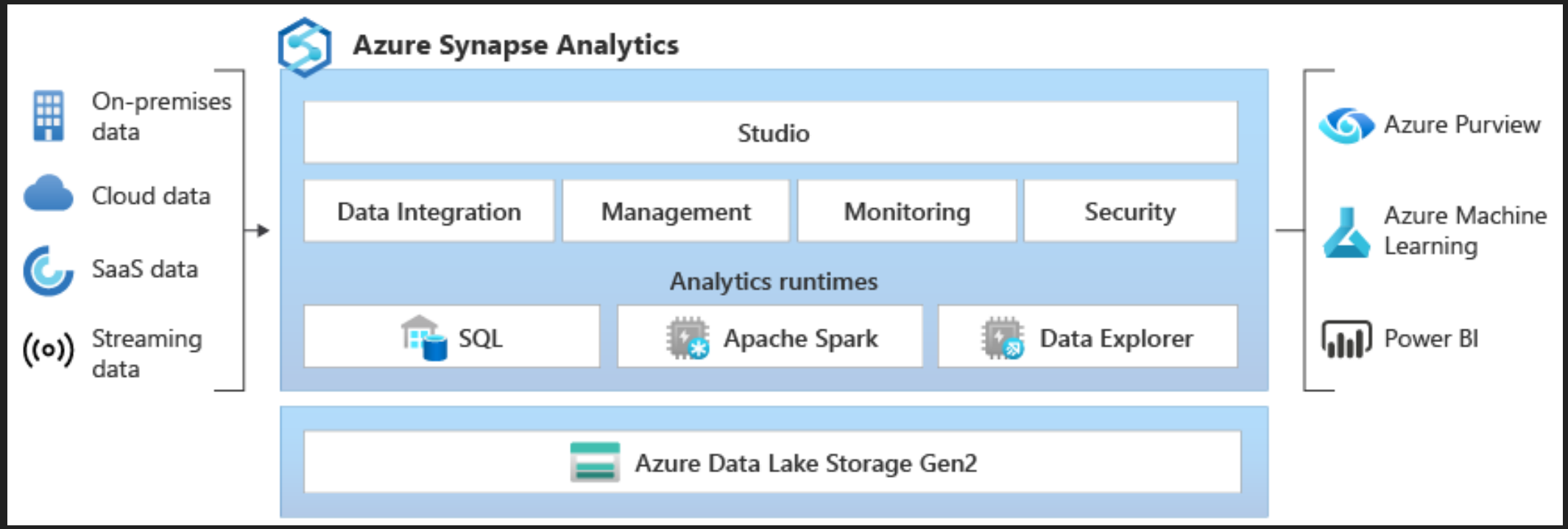
Azure Synapse Analytics

- **Azure Synapse brings together,**
  - The best of **SQL technologies** used in enterprise data warehousing
  - **Spark technologies** used for big data
  - **Data Explorer** for log and time series analytics
  - **Pipelines** for data integration and ETL/ELT
  - **Deep integration** with other Azure services such as Power BI, CosmosDB, and AzureML

Azure Synapse Analytics



Azure Synapse Analytics



Credit: Azure Cloud

Basic Concepts in Azure Synapse Analytics

Basic Concepts

- Synapse Workspace & Studio
- Linked Services
- SQL pool
- Apache Spark pool
- Pipelines
- Data flow
- Integration with Power BI

Basic Concepts

- **Synapse Workspace:** Central environment where you manage and work with various data and analytics resources, **logical container for your analytics projects**
- **Inside a Synapse Workspace you can create and manage,**
  - SQL pools
  - Apache Spark pools
  - Data flows
  - Linked services
  - Pipelines
  - Other resources related to data integration, analytics, and reporting

Basic Concepts

- **Synapse Studio:** Web-based integrated development environment (IDE) provided by Azure Synapse Analytics

- **Serves as the primary tool for,**
  - Designing
  - Developing
  - Monitoring
  - Managing data analytics solutions

- **Enables collaboration on,**
  - Data integration
  - Data preparation
  - Data transformation
  - Data visualization tasks

Basic Concepts

- **Linked Services:** Configurations that define connections to external data sources or destinations
- **These connections can be,**
  - Databases
  - Storage accounts
  - Other services you need to access within your data analytics workflows
- Linked Services store connection information, credentials, and other settings needed to interact with external data sources
- Linked services are used in various components like pipelines and data flows

Basic Concepts

- **SQL Pool (formerly known as SQL Data Warehouse):** Component of Azure Synapse Analytics used for high-performance data warehousing and structured data analytics
- Provides a massively parallel processing (MPP) architecture for query execution
- **Apache Spark Pool:** Used for big data analytics
- With Apache Spark Pools, you can use Spark notebooks and libraries to perform a wide range of data processing tasks including,
  - Data preparation
  - Exploration
  - Advanced analytics

Basic Concepts

- **Pipelines:** Used to create, schedule, and manage data workflows
- Allows you to orchestrate the data movement & data transformation from various sources to destinations
- **Pipelines can include activities like,**
  - Data copying
  - Data transformation
  - Data loading
- Azure Data Factory is often used to design and manage these pipelines

Basic Concepts

- **Data Flow:** Refers to a data transformation activity within pipelines
- It provides a visual interface for designing ETL processes (Extract, Transform, Load)
- Data Flows are used to clean, enrich, and shape data as it moves through the pipeline, making them ideal for data preparation and transformation tasks

Basic Concepts

- **Integration with Power BI:** Azure Synapse Analytics can be integrated with Power BI, a powerful BI tool
- This integration allows you to **create interactive reports & dashboards**
- You can connect Power BI directly to your Synapse Analytics data sources, enabling real-time data analysis & visualization for BI, reporting purposes
- **These concepts collectively provide a comprehensive environment for,**
  - Data integration
  - Data analytics
  - Reporting in Azure Synapse Analytics

Basic Concepts

- Synapse Workspace & Studio
- Linked Services
- SQL pool
- Apache Spark pool
- Pipelines
- Data flow
- Integration with Power BI

Analyze Data with a Serverless SQL Pool

Analyze Data with a Serverless SQL Pool

- **Step 1:** Upload the datasets in Gen2Storage (employee.csv, NYCTripSmall.parquet)
- **Step 2:** The Built-in serverless SQL pool
- **Step 3:** Analyze employee data with a serverless SQL pool (employee.csv dataset)
- **Step 4:** Analyze NYC Taxi data with a serverless SQL pool (NYCTripSmall.parquet dataset)
- **Step 5:** Select top 100 rows on both the datasets (UI)

Analyze Data with Dedicated SQL Pool

Analyze Data with Dedicated SQL Pool

- **Step 1:** Upload the dataset in Gen2Storage (NYCTripSmall.parquet)
- **Step 2:** Create a dedicated SQL pool
- **Step 3:** Load the NYC Taxi Data into Dedicated SQL Pool
- **Step 4:** Explore the NYC Taxi data in the dedicated SQL pool

Analyze Data with Serverless Spark Pool

Analyze Data with Serverless Spark Pool

- **Step 1:** Create a serverless Apache Spark pool
- **Step 2:** Understanding serverless Apache Spark pools
- **Step 3:** Analyze NYC Taxi data with a Spark pool
- **Step 4:** Load the NYC Taxi data into the Spark nyctaxi database
- **Step 5:** Analyze the NYC Taxi data using Spark and notebooks

Understanding Serverless Spark Pool

- A serverless Spark pool is a way of indicating how a user wants to work with Spark
- When you start using a pool, a Spark session is created if needed
- **The pool controls,**
  - How many Spark resources will be used by that session
  - How long the session will last before it automatically pauses
- **You pay for spark resources used during that session and not for the pool itself**
  - This way a Spark pool lets you use Apache Spark without managing clusters
  - This is similar to how a serverless SQL pool works

Integrate with Pipelines

Integrate with Pipelines

- **Step 1:** Create a pipeline and add a notebook activity
- **Step 2:** Forcing a pipeline to run immediately
- **Step 3:** Schedule the pipeline to run every hour
- **Step 4:** Monitor pipeline execution

Monitor Synapse Workspace

Monitor Synapse Workspace

- Introduction to the Monitor Hub
- Integration
- Data Explorer activities
- Apache Spark activities
- SQL activities

Administrative accounts in Synapse SQL

Administrative accounts in Synapse SQL

- When we create the synapse analytics workspace, then we have 2 admin IDs created,

1. SQL admin
2. Microsoft Entra admin

- Synapse Analytics utilizes the combination of Microsoft Entra & SQL authentication methods to manage admin access to various components

Administrative accounts in Synapse SQL

- **SQL admin username:** Refers to the traditional SQL authentication username that grants access to dedicated SQL pools within an Azure Synapse workspace
- This account possesses full administrative privileges over the SQL pool, including creating, reading, updating, and deleting data, as well as managing database objects and user permissions
- **SQL Microsoft Entra admin:** More secure & centralized approach to managing SQL pool administration
- It leverages Microsoft Entra (a unified identity management platform) to authenticate & authorize access to the SQL pool
- This method eliminates the need for managing local SQL passwords & provides a more robust authentication mechanism

Administrative accounts in Synapse SQL

Feature	SQL admin username	SQL Microsoft Entra admin
Authentication method	Local SQL authentication	Microsoft Entra
Scope	Individual dedicated SQL pool	Entire Azure Synapse workspace
Security	Less secure due to local passwords	More secure with centralized identity management
Management	Requires managing local passwords	Simplifies administration and reduces password exposure

Temporary tables in Synapse SQL

Temporary tables in Synapse SQL

- Temporary tables are useful when processing data, especially during transformation where the intermediate results are transient
- With Synapse SQL, temporary tables exist at the session level. They're only visible to the session in which they were created. They are automatically dropped when the session ends

Temporary table limitations

- Dedicated SQL pool does have a few implementation limitations for temporary tables:

1. Only session scoped temporary tables are supported. Global Temporary Tables aren't supported
2. Views can't be created on temporary tables
3. Temporary tables can only be created with hash or round robin distribution. Replicated temporary table distribution isn't supported

Temporary tables in serverless SQL pool

- Temporary tables in serverless SQL pool are supported but their usage is limited. They can't be used in queries which target files
  - For example, you can't join a temporary table with data from files in storage
- The number of temporary tables is limited to 100, and their total size is limited to 100 MB

Identity

Agenda

- Surrogate key
- Creating a table with an IDENTITY column
- Allocation of values
- Skewed data
- SELECT..INTO
- CREATE TABLE AS SELECT
- Explicitly inserting values into an IDENTITY column
- Loading data
- System views
- Limitations
- **Common tasks:** Find the highest allocated value for a table, Find the seed and increment for the IDENTITY property

What is a surrogate key?

- A surrogate key on a table is **a column with a unique identifier for each row**
- The key is not generated from the table data
- Data modelers like to create surrogate keys on their tables when they design data warehouse models
- You can use the IDENTITY property to achieve this goal simply & effectively without affecting load performance

Skewed data

- The range of values for the data type are spread evenly across the distributions
- If a distributed table suffers from skewed data, then the range of values available to the datatype can be exhausted prematurely
  - **For example,** if all the data ends up in a single distribution, then effectively the table has access to only one-sixtieth of the values of the data type
  - For this reason, the IDENTITY property is limited to INT and BIGINT data types only.

SELECT..INTO

- When an existing IDENTITY column is selected into a new table, the new column inherits the IDENTITY property, unless one of the following conditions is true:
  - The SELECT statement contains a join
  - Multiple SELECT statements are joined by using UNION
  - The IDENTITY column is listed more than one time in the SELECT list
  - The IDENTITY column is part of an expression.
- If any one of these conditions is true, the column is created NOT NULL instead of inheriting the IDENTITY property

CREATE TABLE AS SELECT

- CREATE TABLE AS SELECT (CTAS) follows the same SQL Server behavior that's documented for SELECT..INTO
- However,
  - you can't specify an IDENTITY property in the column definition of the CREATE TABLE part of the statement
  - You also can't use the IDENTITY function in the SELECT part of the CTAS
- To populate a table, you need to use CREATE TABLE to define the table followed by INSERT..SELECT to populate it

Limitations

- The IDENTITY property can't be used:
  - When the column data type is not INT or BIGINT
  - When the column is also the distribution key
  - When the table is an external table

Limitations

- The following related functions are not supported in dedicated SQL pool:
  - IDENTITY()
  - @@IDENTITY
  - SCOPE_IDENTITY
  - IDENT_CURRENT
  - IDENT_INCR
  - IDENT_SEED

OPENROWSET

OPENROWSET

- The OPENROWSET function **allows you to access files in Azure Storage**
- OPENROWSET function,
  - reads content of remote data source
  - returns content as a set of rows
- Within the serverless SQL pool resource, the OPENROWSET bulk rowset provider is accessed,
  - by calling the OPENROWSET function
  - specifying the BULK option

OPENROWSET

- **The OPENROWSET function can be referenced in the FROM clause of a query like a table name**
  - example,

```
SELECT *
FROM OPENROWSET(
   BULK 'https://gen2storageaccount257.dfs.core.windows.net/synapsecontainer/data/employee.csv',
   FORMAT = 'CSV',
   PARSER_VERSION = '2.0',
   HEADER_ROW = TRUE
) as output
```

Indexes on dedicated SQL pool tables

Index types

- Dedicated SQL pool offers several indexing options including,
  - clustered columnstore indexes
  - clustered indexes and nonclustered indexes
  - a non-index option also known as heap

Clustered columnstore indexes

- By default, dedicated SQL pool creates a clustered columnstore index when no index options are specified on a table
- Clustered columnstore tables offer both,
  - the highest level of data compression
  - the best overall query performance
- Clustered columnstore tables will generally outperform clustered index or heap tables and are usually the best choice for large tables
- For these reasons, clustered columnstore is the best place to start when you are unsure of how to index your table

Clustered columnstore indexes

- To create a clustered columnstore table, simply specify CLUSTERED COLUMNSTORE INDEX in the WITH clause, or leave the WITH clause off

SQL Copy

```
CREATE TABLE myTable
  (
    id int NOT NULL,
    lastName varchar(20),
    zipCode varchar(6)
  )
WITH ( CLUSTERED COLUMNSTORE INDEX );
```

Credit: Azure Cloud

Clustered columnstore indexes

- **Few scenarios where clustered columnstore may not be a good option:**

1. Columnstore tables do not support varchar(max), nvarchar(max), and varbinary(max). Consider heap or clustered index instead
2. Columnstore tables may be less efficient for transient data. Consider heap and perhaps even temporary tables
3. Small tables with less than 60 million rows, consider heap tables

Heap tables

- When you are temporarily landing data in dedicated SQL pool, you may find that using a heap table makes the overall process faster
- This is because loads to heaps are faster than to index tables and, in some cases the subsequent read can be done from cache
- If you are loading data only to stage it before running more transformations, loading the table to heap table is much faster than loading the data to a clustered columnstore table
- In addition, loading data to a temporary table loads faster than loading a table to permanent storage. After data loading, you can create indexes in the table for faster query performance

Heap tables

- Cluster columnstore tables begin to achieve optimal compression once there are more than 60 million rows
- For small lookup tables, less than 60 million rows, consider using HEAP or clustered index for faster query performance

Heap tables

- **To create a heap table, simply specify HEAP in the WITH clause**

SQL Copy

```
CREATE TABLE myTable
  (
    id int NOT NULL,
    lastName varchar(20),
    zipCode varchar(6)
  )
WITH ( HEAP );
```

Credit: Azure Cloud

Clustered and nonclustered indexes

- Clustered indexes may outperform clustered columnstore tables when a single row needs to be quickly retrieved
- For queries where a single or very few row lookup is required to perform with extreme speed, consider a clustered index or nonclustered secondary index
- The disadvantage to using a clustered index, only queries that benefit are the ones that use a highly selective filter on the clustered index column
- To improve filter on other columns, a nonclustered index can be added to other columns
- However, each index that is added to a table adds both space and processing time to loads

Clustered and nonclustered indexes

- **To create a clustered index table, simply specify CLUSTERED INDEX in the WITH clause**

SQL Copy

```
CREATE TABLE myTable
  (
    id int NOT NULL,
    lastName varchar(20),
    zipCode varchar(6)
  )
WITH ( CLUSTERED INDEX (id) );
```

Credit: Azure Cloud

Clustered and nonclustered indexes

- **To add a non-clustered index on a table, use the following syntax:**

SQL Copy

```
CREATE INDEX zipCodeIndex ON myTable (zipCode);
```

Credit: Azure Cloud

Partitioning tables in dedicated SQL pool

Agenda

- What are table partitions?
- Partition sizing

Table partitions

- **Table partitions enable you to divide your data into smaller groups of data.** In most cases, table partitions are created on a date column
- Partitioning is supported on all dedicated SQL pool table types,
  - clustered columnstore
  - clustered index
  - heap
- Partitioning is also supported on all distribution types,
  - Hash
  - round robin distributed

Table partitions

- Partitioning can benefit,
  - data maintenance
  - query performance
- Whether it benefits both or just one is dependent on how data is loaded and whether the same column can be used for both purposes, since partitioning can only be done on one column

Table partitions

- **Benefits for Data Loading**

1. **Turbocharge Your Data Pipeline:** Load data at warp speed with partition deletion, switching, and merging
2. **Log Less, Do More:** Skip time-consuming transaction logs - focus on what matters
3. **Archive Like a Pro:** Archive old data effortlessly by dropping partitions (e.g., say goodbye to month-long deletes)

Table partitions

- **Benefits for Your Queries**

1. **Laser-Focused Searches:** Find what you need faster. Queries skip unwanted partitions, scanning only relevant data
2. **Smarter Scans, Even Smart Results:** Clustering and indexing amplify the benefits, optimizing even more searches

Table partitions

- **Remember:** Choose a key date column for partitioning that matches your data flow for maximum impact

Partition sizing

- **Balancing Act:** Partitioning enhances performance but excessive partitions can harm it, especially for clustered columnstore tables
- **Key Considerations:**
  - Understand when to use partitioning and the optimal number of partitions
  - Successful schemes typically involve tens to hundreds of partitions, not thousands
- **Clustered Columnstore Tables:**
  - Optimal compression and performance require a minimum of 1 million rows per distribution and partition
  - Distributions are pre-set to 60; additional partitions should align with this distribution

Partition sizing

- **Example Scenario:**
  - For a sales fact table with 36 monthly partitions and 60 distributions, aim for 60 million rows per month or 2.1 billion rows in total.
  - Adjust partitioning based on the recommended minimum rows per partition for optimal performance.

Syntax differences Between Dedicated SQL Pool and SQL Server

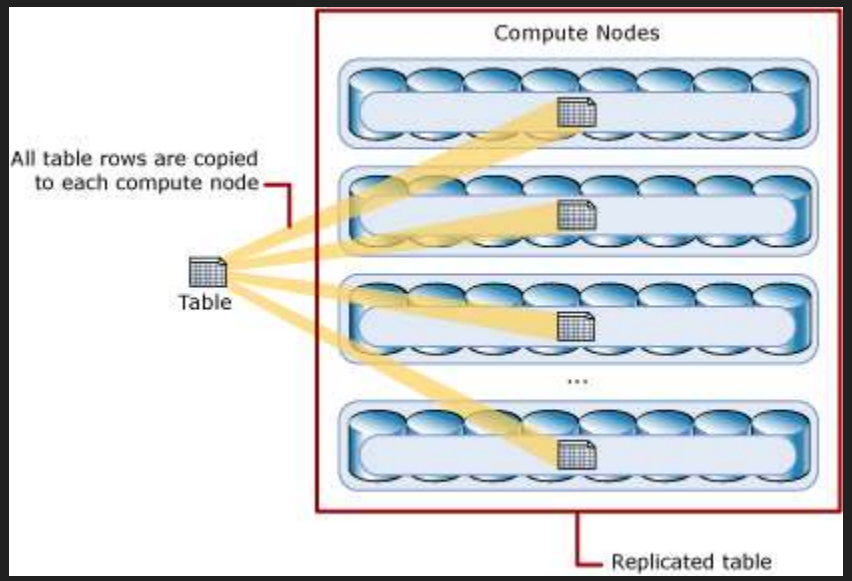
- Dedicated SQL pool introduces a way to define partitions that is simpler than SQL Server
- **Partitioning functions and schemes are not used** in dedicated SQL pool as they are in SQL Server
- Instead, all you need to do is **identify partitioned column and the boundary points**
- While the syntax of partitioning may be slightly different from SQL Server, the basic concepts are the same. SQL Server and dedicated SQL pool support one partition column per table, which can be ranged partition

Replicated Tables

Replicated Table

- A replicated table has a full copy of the table accessible on each Compute node
- Replicating a table removes the need to transfer data among Compute nodes before a join or aggregation
- Since the table has multiple copies, replicated tables work best when the table size is less than 2 GB compressed
- 2 GB is not a hard limit. If the data is static and does not change, you can replicate larger tables

Replicated Table



Credit: Azure Cloud

Replicated Table

- Replicated tables are ideal for dimension tables in a star schema, particularly when dealing with slowly changing descriptive data like customer names and addresses
- This approach minimizes maintenance due to the stable nature of dimension data

Replicated Table

- **Consider using replicated tables when:**
  - Table size is less than 2 GB, irrespective of row count
  - Tables are frequently joined, avoiding data movement in the process
  - Query plans reveal BroadcastMoveOperation, indicating potential benefits from eliminating data movement

Replicated Table

- **Replicated tables may not offer optimal performance in scenarios involving:**
  - Frequent insert, update, or delete operations, necessitating frequent rebuilding
  - Frequent scaling of the SQL pool, leading to replicated table rebuilding
  - Tables with many columns, where only a subset is frequently accessed. Consider distribution with selective indexing for improved efficiency in such cases

Table constraints

Table constraints

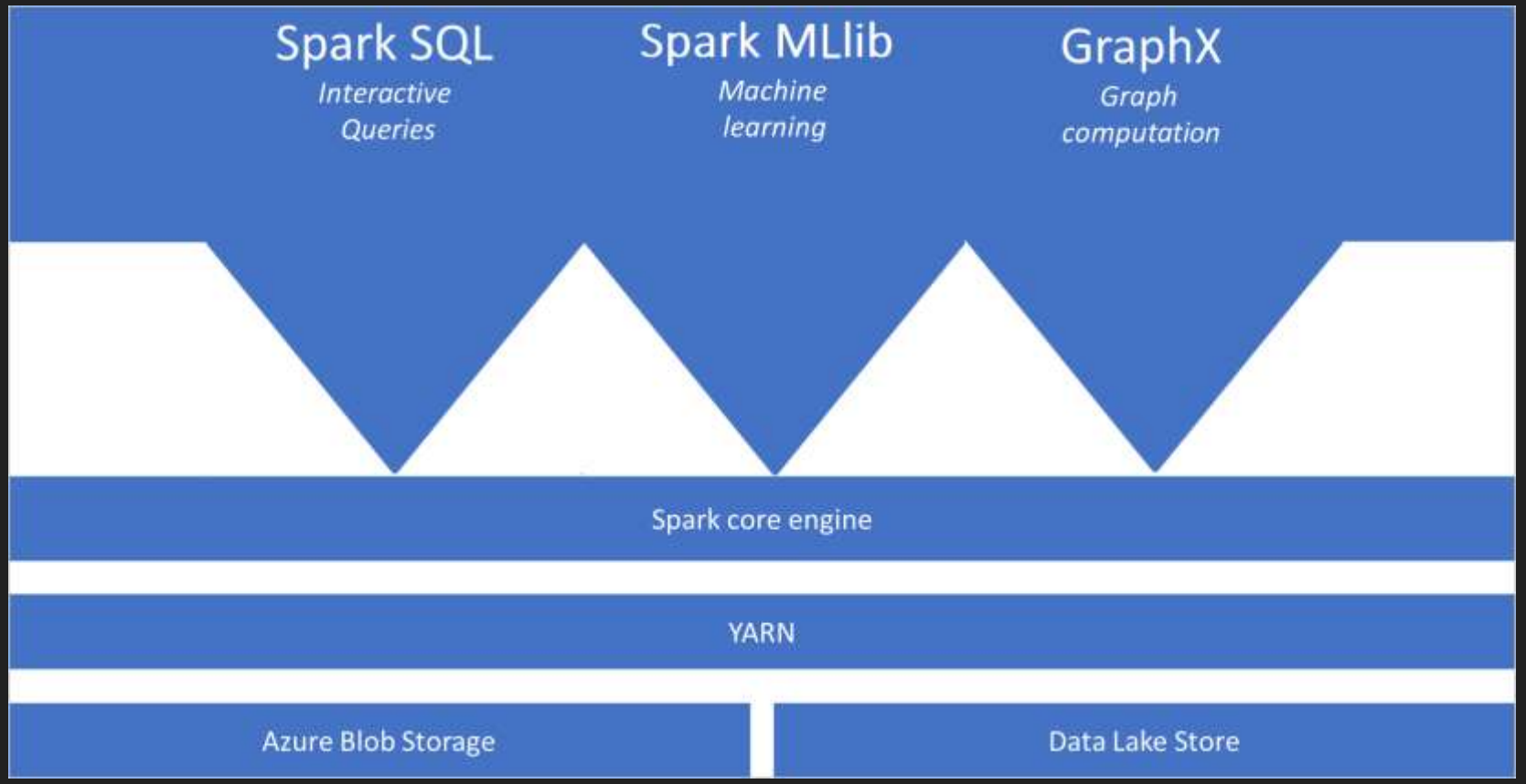
- **Dedicated SQL pool supports these table constraints:**
  - PRIMARY KEY is only supported when NONCLUSTERED and NOT ENFORCED are both used
  - UNIQUE constraint is only supported when NOT ENFORCED is used
- FOREIGN KEY constraint is not supported in dedicated SQL pool
- Having primary key and/or unique key allows dedicated SQL pool engine to generate an optimal execution plan for a query
  - All values in a primary key column or a unique constraint column should be unique

Apache Spark in Azure Synapse Analytics

Apache Spark in Azure Synapse Analytics

- **Apache Spark:** Parallel processing framework, supports in-memory processing to boost the performance of big data analytic applications
- **Apache Spark in Azure Synapse Analytics:** One of Microsoft's implementations of Apache Spark in the cloud
- Azure Synapse makes it easy to create & configure a serverless Apache Spark pool in Azure
- Spark pools in Azure Synapse are compatible with Azure Storage & Azure Data Lake Generation 2 Storage, So you can use Spark pools to process your data stored in Azure

Apache Spark in Azure Synapse Analytics

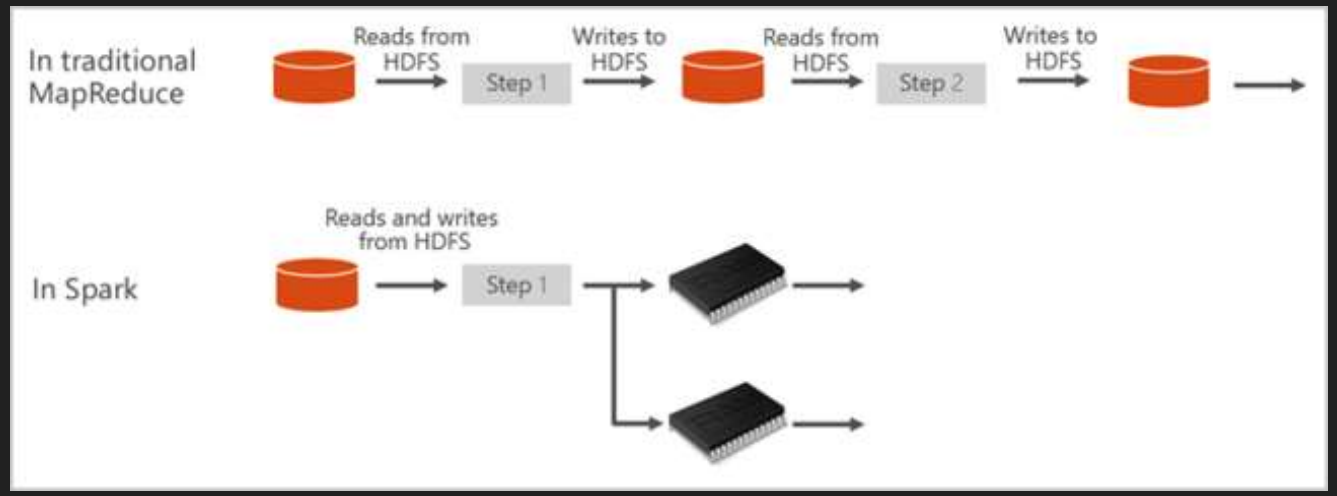


Credit: Azure Cloud

Apache Spark in Azure Synapse Analytics

- Apache Spark provides primitives for in-memory cluster computing
- A Spark job can load & cache data into memory and query it repeatedly
- In-memory computing is much faster than disk-based applications
- Spark also integrates with multiple programming languages to let you manipulate distributed data sets like local collections
- There's no need to structure everything as map and reduce operations

Apache Spark in Azure Synapse Analytics



Credit: Azure Cloud

Apache Spark in Azure Synapse Analytics

Feature	Traditional MapReduce	Spark MapReduce
Data storage	HDFS only	HDFS or other sources
Intermediate results	On disk	In memory
Shuffle	Less efficient	More efficient
Fault tolerance	Less fault-tolerant	More fault-tolerant

Benefits of creating a Spark pool in Azure Synapse Analytics

- **Azure documentation link:** https://learn.microsoft.com/en-us/azure/synapse-analytics/spark/apache-spark-overview#what-is-apache-spark

Spark Pool Architecture

- **SparkContext Coordinates:** The SparkContext object, within your main program (driver), coordinates the application's processes on the pool
- **Resource Allocation:** It connects to the cluster manager (YARN) to allocate resources and acquire executors on nodes
- **Code Distribution:** Your application code (JAR or Python files) is sent to the executors
- **Task Execution:** SparkContext sends tasks to the executors for running
- **Parallel Operations:** Executors execute parallel operations on nodes
- **Result Collection:** SparkContext gathers the results

Spark Pool Architecture

- **Data Reading/Writing:** Nodes handle data reading/writing from/to the file system
- **Data Caching:** Nodes cache transformed data in memory as RDDs
- **DAG Creation:** SparkContext converts the application into a DAG (directed acyclic graph)
- **Task Execution:** Individual tasks run within executor processes on nodes
- **Dedicated Executors:** Each application gets its own executor processes, which persist throughout the application and run tasks in multiple threads

Use Cases

- **Data Engineering/Data Preparation:** Apache Spark empowers large-scale data preparation and processing with multiple languages (C#, Scala, PySpark, Spark SQL) and extensive libraries
  - Makes data more valuable for other Azure Synapse Analytics services
- **Machine Learning:** Azure Synapse Analytics provides a complete machine learning environment:
  - MLlib: Built-in machine learning library for Spark, ready for large-scale tasks
  - Anaconda: Pre-installed Python distribution with extensive data science packages
  - Notebooks: Built-in support for creating and running machine learning workflows

Use Cases

- **Streaming Data:** Synapse Spark supports structured streaming with these key points:
  - Limited support: Only with specific runtime versions
  - 7-day lifespan: All jobs (batch and streaming) expire after 7 days
  - Restart automation: Typically managed using Azure Functions