Pre-requisites

13 February 2023 10:04

CPU i5/i6,
RAM - at least 12 GB,
Disk Storage - 500GB

Windows 10, 11 Professional / Enterprise
SQL Server 2017 / 2019 Developer / Enterprise
MS Office
PowerBI
Alteryx

username	email	Password	invitation_link
db01@mml.local	db01@mml.local	uavdxoa5yg	https://laas.makemylabs.in/wsm/KPMG-Databrick-skip20230213170942?invitation_id=35acb6a3-341b-4330-b87a-8a810472703c&context=True

Database Fundamentals and SQL Server

13 February 2023 10:07

Database types & Models

1. **Flat file database** - kind of text database where each line of the plain text file holds only a single record (e.g. MS access)
2. **Hierarchical database** - based on hierarchical data model, it's viewed as a collection of tables, data is designed into a tree like structure where each record consists of one parent record and many child record. (e.g. IBM DB2 - IBM information Management system (IMS), Windows Registry, XML data storage.
3. **Network Model database** - it can consists of parent segments and this segment can be grouped together as levels but there always exists a logical association between the segments belonging to any level.
4. **Relational database** - consists of tables and columns, rows.
5. **Object-oriented database** - information can be represented in the form of object - oriented programming inclined towards the objects like e.g. multimedia records in a relational database can be definable data object.
6. **Distributed database** - consists of two or more files located in different sites / location.
7. **NoSQL database** - non-relational db which has support for unstructured, semi-structured data as well as it can include dynamic schema, flexible data model for faster data retrieval
   e.g. Mongo db, Cassandra, Azure Cosmos db, Couch db etc.

8. **Graph database** - node-entity (rows in the table), attributes (relationship/columns) . e.g. Neo4j, Azure Cosmos db Graph API etc.

ACID properties in RDBMS

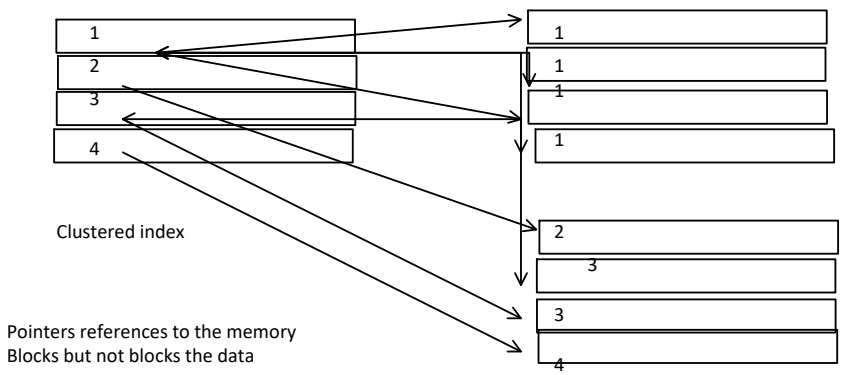
A = Atomicity	The entire transaction should take place at once or doesn't happen at all
C = Consistency	The database must be consistent before and after the transaction
I = Isolation	Multiple transactions can occur independently without interference
D = Durability	The changes of a successful transaction occurs even if a system failure occurs.

Clustered & Non-clustered Index

1. In a table, there can be one clustered index, can be multiple non-clustered index.
2. Clustered index is much more faster, non-clustered index is slower.
3. Clustered index requires less memory for operations, non-clustered indexes requires more memory for operations.
4. In clustered index, index is the main data. In case of non-clustered index, index is the copy of the data.
5. Clustered index store pointers to blocks, not the data. Non-clustered index store both the value and a pointer to actual row which holds the data.
6. Primary Keys of the table by default is considered as a clustered index. Composite key used with unique keys of the table defines the non-clustered index.
7. A clustered index is a type of index in which table records are physically recorded to match the index. A non-clustered index is a special type of index in which the logical order of the index doesn't match physical stored order of the rows on disk.
8. Clustered index size is larger, non-clustered index size is smaller.

Features of Indexes

1. A index can speed up the data retrieval and query execution very quickly by optimization
2. Indexes can be created or dropped with no effect on the data
3. When an index is created, it includes a column containing a wide range of values.



Primary Key	Unique Key
A table can have only one primary key	A table can have more than one unique key unlike the primary key
A primary key cant accept null values	Unique key constraint can accept null values for a column
	Unique key constraints are also referenced by the foreign key of another table, it can be used when developer wants to enforce a unique constraints on a column or group of columns which is not a primary key
Primary key has the support of auto-increment values.	A unique key does not support auto-increment value
We cant change or delete values stored in primary key	We can change the unique key values

Surrogate Keys

Surrogate keys are called synthetic primary keys which are generated when a new record is inserted into the table automatically by the database which can be declared as the primary key of the table.

Features of surrogate key

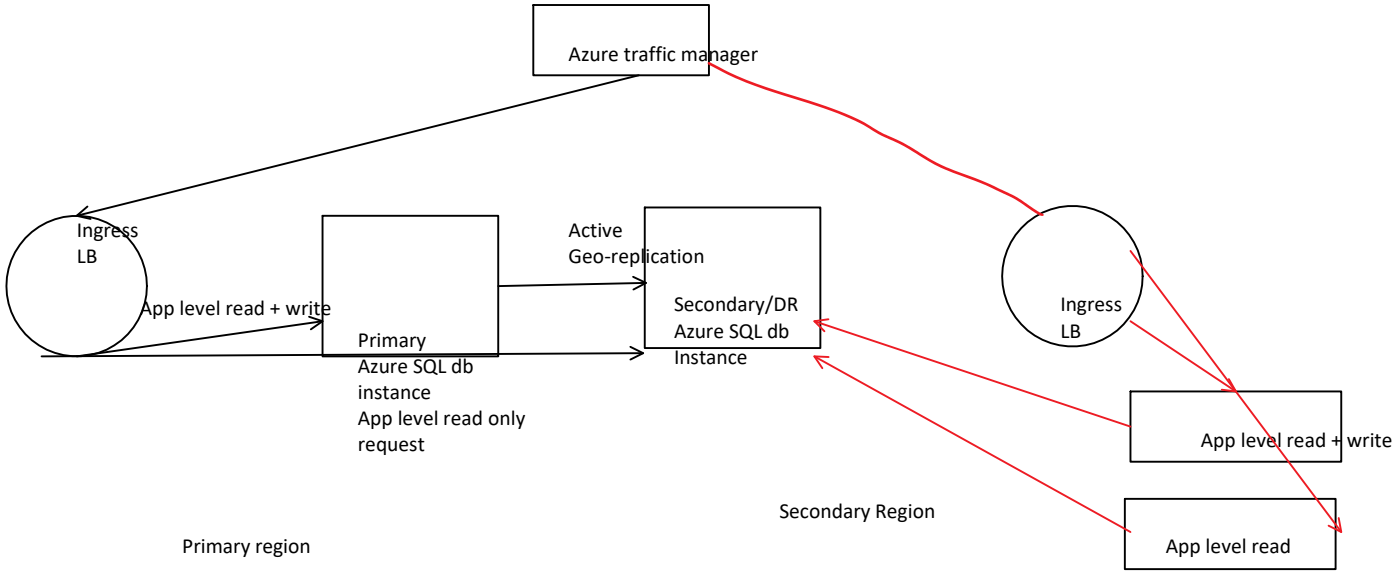
1. It's a sequential number outside the database which's made available to the user and application or it just acts as an object which's present in the database but not visible to the user/application.
2. It's automatically generated by the system
3. It holds an anonymous integer
4. It contains unique values for all records of the table
5. The value can never be modified by the user or application
6. Surrogate keys are called the factless key which is added just for the case of identifying unique values and contains no relevant fact which is useful for the table.

Surr_no	Reg_no	Name	Marks
1	21101	Mark	50
2	32281	Henry	70
3	43353	Alan	60
4	CS101	Maria	80
5	CS201	John	67

Azure SQL

14 February 2023 09:31

SQL Server on Azure VM (IaaS)	Azure SQL Managed Instance (PaaS)	Azure SQL database (PaaS)
Azure SQL VM is preferred for the scenario of simple lift-shift or rehost of existing full SQL server database engine with core administration services like e.g. service broker, SQL Server Agent, SQL mirrors, DTC, .NET and CLR data types & functions, SQL server BI support.	Lift - shift of database migration including core SQL db features like service broker, SQL server agent, distributed transactions and .net/CLR data types etc. without any management overhead.	Purely managed SQL database offering with no licensing requirement, no underlying administration of server, SQL database or network is required. End to end database migration is feasible with a just few clicks into the core sql development features.
License is required. Either through BYOL with AHB or pay-as-you-go model, license has to be procured.	No license is required, entire SQL on-premise feature can be availed with the managed platform service.	No license is required. Only SQL db dev specific features are available. There's a limitation of db sizes (~100TB)
SQL Server BI (SSRS, SSAS, SSIS), SQL Server broker, .net framework runtime, CLR integration, distributed transactions, database mail etc. all features are supported	.net framework, functions, distributed transactions, ACID , automated backups, HA are also supported along with no administration required.	No support for .net framework runtime, distributed transactions, CLR related functions, stored procedures, windows runtime etc. are not supported
SLA - 99.95% Automated backups and business continuity through FCI and availability groups. Migration can be implemented through Azure migrate (Azure site recovery).	SLA - 99.99% automated backups, point-in time restore, geo-replication, high availability, automated patching etc. features are available.	SLA - 99.99% , automated backups, active geo-replication, high availability, and disaster recovery, replication on both availability zone and regional level.



Migration to Azure SQL database

We can migrate SQL Server database running on-premises or to :

a) SQL Server on Azure VM
b) SQL Server Managed instance
c) Azure SQL db

We can migrate the following db types -

1. SQL Server running on-prem or SQL server running on VM (vmware/hyper-v etc.)
2. SQL server running on AWS EC2, Google compute engine
3. SQL server on AWS RDS
4. Cloud SQL for SQL server - GCP

1. Azure Migrate - Discovery and assess single database or at scale from different env
2. Azure SQL migration extension for Azure Data Studio - migrate to Single databases at scale, it can run in both online and offline modes.

3. Import-export service / BACPAC - migrate individual LOB apps databases , suited for smaller databases and doesn't require a separate migration service or tool.
4. Bulk Copy - migrate / transform data from source SQL server db to Azure SQL db. There's a downtime for exporting data at source and importing at the target.
5. Azure Data Factory (ADF) - source SQL server db to target Azure SQL db. Cost is important consideration and is based on factors like pipeline triggers, activity runs and duration of data movements.
6. SQL data sync - synchronize data between source and target databases. Suitable to run continuous sync between Azure SQL db and on-premise / Azure SQL db. It can have higher performance impact depending on the workload.

Migration Steps

1. Discover -
2. Assessment
3. Migrate
4. Cutover
5. Optimize

A) Data Migration Assistant (DMA)

Scenario to choose for Azure SQL Managed Instance

Pools are well suited for a large number of databases with specific utilization patterns, for a particular database, the pattern is characterized by low average utilization with infrequent utilization spikes.

Conversely, multiple databases with persistent medium-high utilization shouldn't be placed in the same elastic pool.

The more databases, we can add into a pool, the greater the savings become. Depending on the app utilization pattern, it's possible, to see savings as few as two AWS S3 db.

Overutilization and underutilization of DTU usage can be overcome through Azure SQL elastic db.
A elastic pool allows these unused DTUs to be shared across multiple databases. A pool reduces the DTUs needed and the overall cost.

The best size for a pool depends on the aggregate resources required for all databases in the pool

1. Maximum compute resources utilized by all databases in the pool. Compute resources are indexed by either eDTUs or vCores depending on the purchasing model.
2. Maximum storage bytes utilized by all databases in the pool.

Business Continuity for Azure SQL Elastic db

1. Point-in time restore - point-in-time restore uses automatic database backups to recover a database in a pool to a specific point in time.
2. Geo-restore - Geo-restore provides the default recovery option when a database is unavailable because of a incident in the region where the db is hosted.
3. Active geo-replication - For applications, which has more aggressive recovery requirements, than geo-restore can offer, we can configure active geo-replication or auto-failover group.

Azure SQL Managed Instance

vCore based purchasing model

A vCore represents a logical CPU and provides the option to choose the physical characteristics of the hardware (the no of cores, the memory, the storage sizes). The vCore based purchasing model gives us the flexibility, control, transparency of individual resource consumption and a straight forward way to translate on-prem workload requirements to the Azure platform.

vCore based purchasing model depends on -

- Service tier
- Hardware configuration
- Compute resources (the no of vcores and amount of memory)
- Reserved database storage
- Actual backup storage

Benefits of vCore based purchasing model used by Azure SQL managed instance

- Control over hardware configuration to better match the compute and memory requirements of the workload
- Pricing discounts for AHB and reserved instance
- Greater transparency in the hardware details which empowers compute, helping facilitate planning for migrations from on-prem deployments
- Higher scaling granularity with multiple compute sizes available.

Backup Storage

- Point in time restore - The storage consumption depends on the rate of the change of database and retention period configured for backups. We can configure a separate retention period of each database between 0 to 35 days for SQL managed instance. A backup storage amount is equal to the configured max data size is provided at no extra charge.

- Long term retention (LTR) - customers have the option to configure the long term retention of full backups for upto 10 years,

Feature	General Purpose	Business Critical
Scenario	Most standard business workloads. Offers budget-oriented, balanced and scalable compute and storage options	Offers business applications with highest resilience to failures by using several isolated replicas, and provides the highest I/O performance.
Read-only Replicas	0	1
HA replica	One replica is available	Three HA replica is available & one read-scale replica
Read-only replicas with failover groups enabled	One additional read-only replica and two total readable replicas, which includes the primary replica	Two additional read-only replicas, three total read-only replicas, four total readable replicas which includes the primary replica.

SQL Server on Azure VM (HADR configurations)

A Windows Server Failover Cluster is used for high availability and disaster recovery (HADR) with SQL Server on Azure VM.

Best Practices

- Deploy the SQL Server VM to multiple subnets to avoid the dependency on the Azure LB or a distributed network to route traffic to HADR solution.
- Change the cluster to less aggressive parameters to avoid unexpected outages from transient network failure to Azure platform maintenance.
- Place the SQL Server VM in a AS or different Azs.
- Use a single NIC per cluster node
- Configure the cluster quorum voting to use 3 or more odd numbers of votes.

a) Cloud witness - it's ideal for deployments in multiple sites, multiple zones and multiple regions. Use a cloud witness for disk quorum whenever possible unless using a shared-storage cluster solution.

b) Disk witness - it's the most resilient quorum option and is preferred for any cluster which uses Azure shared disks (like shared SCSI, iSCSI of fiber SAN). A clustered shared volume can be used for disk witness.
c) Fileshare witness - is suitable when the disk witness and cloud witness are unavailable

SQL database auditing

- Retain - an audit trail of selected events. We can define categories of database actions to be audited.
- Report - on database activities. We can use pre-configured reports and a dashboard to get started quickly with activity and event reporting.
- Analyse - reports, we can find suspicious events, unusual activity and trends.

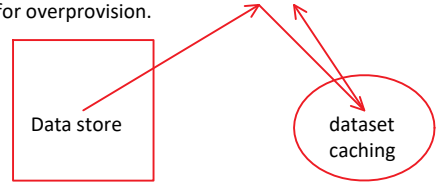
Server level vs database level audit policy

- An audit policy can be defined for a specific database or as a default server policy in Azure SQL db.

- A server policy applies to all existing and newly created databases on the server.
- If server auditing is enabled, it always applies to the database. The database will be audited, regardless of the database auditing settings.
- When auditing policy is defined at the database level to a log analytics workspace or an event hub subscription destination, these operations will not keep the source database level auditing policy like
- Database copy
- Point -in time restore
- Geo - replication

Performance improvement measures for Azure SQL db

1. Caching can improve app performance - database queries can be stored over the cache and return the queries much faster than the db when the app requests them. Not only this approach, helps for significant reduction for latency but also reduces the load on the database, lowering the need for overprovision.



- Determine whether the item is held in cache
- If the item is not available, currently in cache, read them from the data store
- Store a copy of the item in the cache

2. Caching are better than db at handling high throughput of requests - enabling the app to handle more simultaneous users.
3. Caches are typically most popular beneficial for read-heavy workloads where the same data is being accessed again and again. Caching pricing, inventory, session state or financial data are some of the examples.

4. Resolve the server index fragmentation with automatic tuning - keep the fast query performance is paramount for app with rdbms, one of the common cause for degraded performance is index fragmentation. With the indices, The SQL server can quickly locate the row with the data which user is requesting for, the first step is to identify the degree of fragmentation.

5. Resolve the fragmentation - there're three primary options for automatic tuning with Azure SQL db

a) CREATE INDEX - create new indices which can improve the performance
b) DROP INDEX - Drops redundant and unused indices (>90 days)
c) FORCE LAST GOOD PLAN - identifies queries using the last known good execution plan

MAXDOP configuration

a) In Azure SQL database (both for single and elastic pool db), the default MAXDOP setting for each new single database and elastic pool database is 8.

b) For Azure SQL managed instance, the max degree of parallelism instance option will be set to 8 by default.

This default prevents unnecessary resource utilization, while still allowing the database engine to execute queries faster using parallel threads.

- Azure SQL db level, MAXDOP can be controlled at the db level using MAXDOP database-scoped configuration.
- For Azure SQL managed instance, customers can also set the server 'max degree of parallelism' configuration option & can control MAXDOP at the resource governor workload group level.
- For all of Azure SQL deployment options, MAXDOP can additionally be controlled at the individual query level by using OPTION (MAXDOP) query hint where it actually overrides MAXDOP configurations set in the database or instance scope.

Azure Data Factory

14 February 2023 09:31

ETL is the process for integrating and loading of the data for computation and analysis. It's also the primary method to process data for traditional data warehousing and BI applications.

Benefits

1. Extract the data from the legacy systems
2. Cleanse the data to improve data quality and establish consistency
3. Load the data into the target database.

Azure Data Factory is a managed data orchestration and integration platform which helps to build complex Extract, Transform and Load (ETL) or ELT projects with data integration features.

Data Orchestration is the practice of acquiring, cleaning and matching, enriching and making data accessible across the technology teams. The effective data orchestration captures data from several sources and unifies it within a centralized system, making it organized and ready to use for getting insights from data.

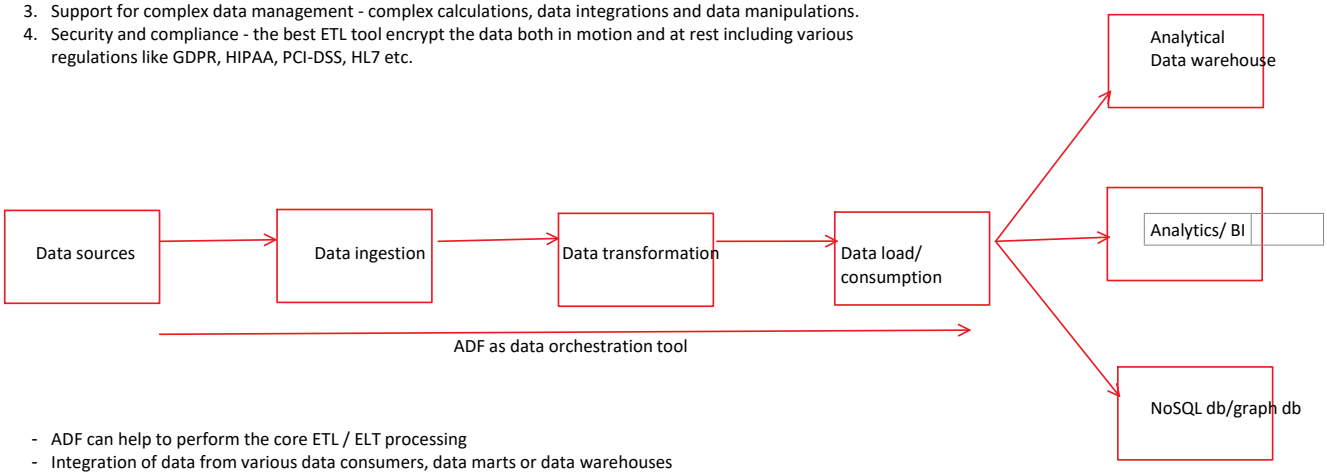
In ETL perspective, orchestration means automated management, co-ordination and management of complex data pipelines. ETL tools run on a schedule.

ETL (Extract, Transform & Load)	ELT (Extract, Load and Transform)
ETL can perform the end to end data transformation after loading the raw data into Transformation layer and captures the insights from the data.	ELT copies or exports the data from the data source locations, but instead of loading it to a staging area for transformation, it loads the raw data to the target data store to be transformed as required.
The ETL process involves more structured datasets and data has to be rational in nature, should have proper keys (PK, FK) to integrate the relationships between multiple tables	ELT is useful for high-volume, unstructured datasets as loading can occur directly from the data source. ELT is more ideal for Big Data Analytics pipeline because it can clean, parse the unstructured, semi-structured data and can load the data directly into the native format. It's ideal for big data use cases.
In on-premise, ETL data pipeline is more common	On Cloud, Azure ELT pipeline is more popular

Azure Data Factory (ETL/ ELT tool)

1. Comprehensive automation support - leading ETL tool (ADF) can automate the entire data flow, from data sources to the target data warehouse. Many tools recommend rules for extracting, transforming and loading of the data.

2. Visual drag-drop support - the functionality can be used for specifying rules and data flows.
3. Support for complex data management - complex calculations, data integrations and data manipulations.
4. Security and compliance - the best ETL tool encrypt the data both in motion and at rest including various regulations like GDPR, HIPAA, PCI-DSS, HL7 etc.



- ADF can help to perform the core ETL / ELT processing
- Integration of data from various data consumers, data marts or data warehouses
- Allows us to create the data driven workflows through ingestion, transformation and consumption
- Code free ETL tool, it helps to integration of data through ingestion, transformation and consumption through the data flows, control flows and scheduling of the ADF pipeline.

ETL process

1. Extract

Raw data gets copied into / exported from source location to a staging area. Data management can extract the data from the variety of data sources that can be structured or unstructured.

- SQL / noSQL
- CRM/ERP/MDM
- Flat files

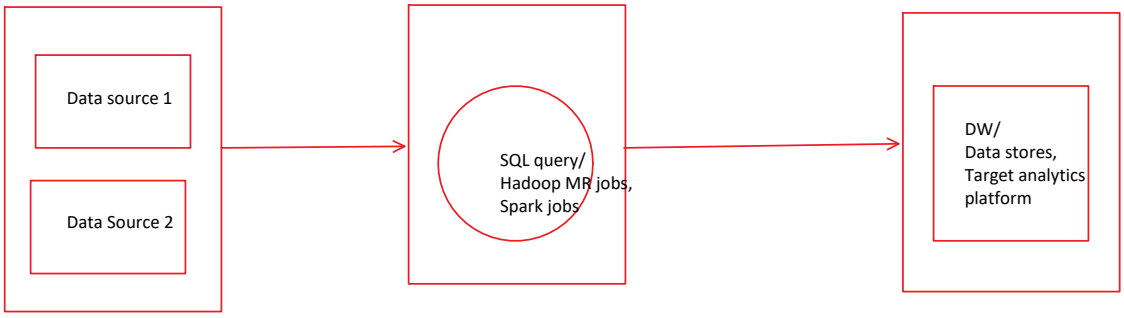
- Web pages

2. Transformation

In the staging area, the raw data undergoes the data processing, data has to be transformed and consolidated for its intended analytical use cases -

- Filtering, cleansing and de-duplicating, validating and authenticating the data
- Performing calculations, translations, summarizations of raw data
- Conducting audits to ensure data quality and compliance
- Removing, encrypting or protecting data governed by regulators
- Formatting the data into tables, joined to match the schema of the target data warehouse.

3. Load
   In the last phase, the transformed data is moved from the staging area to a target data warehouse. Which involves loading of the data , followed by the periodic loading of incremental data changes and full refreshes to erase and replace the data in the data warehouse.



- The data transformation can take place usually involves various operations, such as filtering, sorting, aggregating, joining of the data, cleaning, deduplicating and validating the data.

- As part of offering, these three ETL processes can execute in parallel to save time. e.g. while the data is being extracted, and a loading processing can begin working on the prepared data. Rather than just waiting for the entire extraction process to complete.

Core Components of Azure Data Factory

1. **Linked Service** - Creates a linking connection between the data source and the ADF pipeline.

   We must create the linked service to link up the data source with the data factory.

   e.g. db connection string which defines the connection information required for the service to connect to the external resource.

2. **Dataset** - It's the named view of the data which simply points to or references the data which required to be used in the ADF activities as inputs and outputs.

e.g. SQL server db, files

3. **Activities** - The activities refers to the actions, jobs / tasks can be performed on the data. Activities also can produce a dataset & the datasets consume the activities.

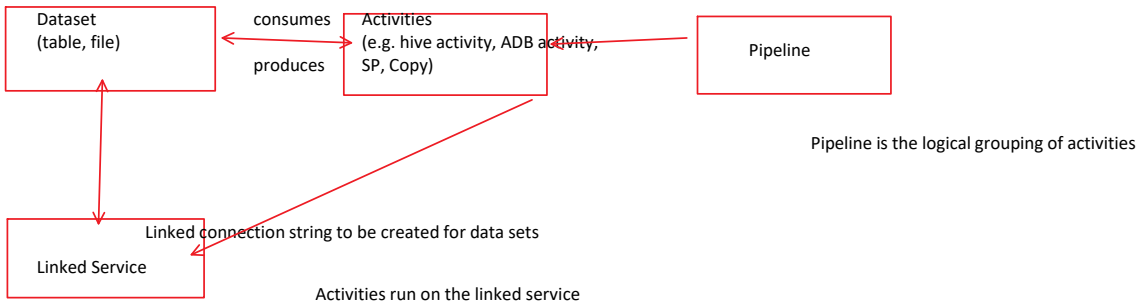
while copying/moving the data from Azure Blob storage to Azure SQL db, the storage and the linked service connection string need to be created.

- Activity can help us to copy the data
- Deduplicating the data
- Formatting the data removing all nulls, NANs
- Applying normalization, remove redundancies
- Adding column headers and formatting

4. **Pipeline** - It's logical grouping of activities, which can be monitored, managed and scheduled. The activities in a pipeline define actions to perform on the data.

e.g. an ADF pipeline could contain the set of activities which can ingest and clean log data, then transform the mapping data flow to analyse the log data.

- The pipeline allows to manage the activities as a set instead of each one individually. We can deploy and schedule the pipeline instead of activities independently.
- The activities in a pipeline defines the action to perform on the data.



Hands-on Lab 01

1. Create Azure Data factory
2. Explore the Data Factory Studio
3. Explore the ADF activities, pipelines and datasets, linked services
4. Monitoring and management of ADF

Hands-on Lab 02

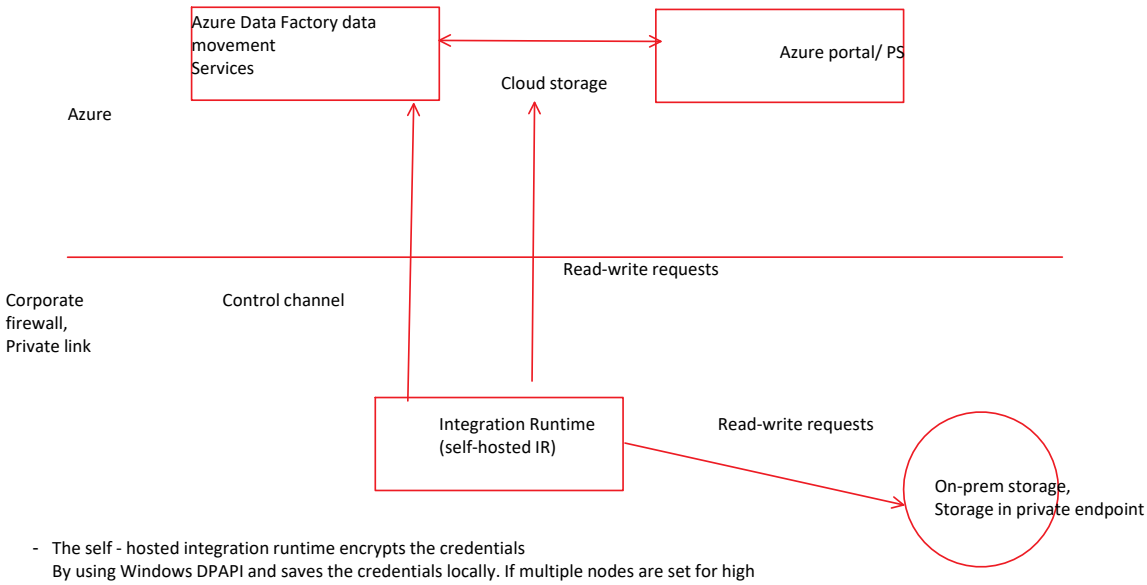
1. Transferring data from Azure SQL db to Azure Blob storage using Copy activity and Copy data tool.

a) Provision Azure SQL db with sample db
b) Provision Azure blob storage
c) Use the copy data tool to copy data from Azure SQL db to Azure blob storage/ADLS gen2
d) Create the ADF pipeline
e) Monitor the pipeline

The Integration Runtime (IR) is the compute infrastructure used by ADF to provide the data integration capabilities across the different environments.

- Data Flow - Execute a data flow in a managed Azure compute environment.
- Data movement - Copy data across the data stores in a public or private networks (for both on-prem or Vnets). This service provides support for built-in connectors, format conversion, column mapping and performant / scalable data transfer.
- Activity dispatch - Dispatch and monitor transformation activities running on a variety of compute services like ADB, Azure HDI, ML Studio, Azure SQL db and SQL server.
- SSIS package extension - natively execute SSIS packages in a managed Azure compute environment.
- So, the integration runtime provides the bridge between the activities and linked services. It specifies the compute environment where the activity is to be performed in the closest region to the target data store or compute service to maximize the performance while allowing flexibility to meet security and compliance requirements.
-
-

a) Azure - Data Flow, Data movement, Activity Dispatch (public and private endpoints)
b) Self-hosted - Data movement, Activity dispatch (public and private endpoints)
c) Azure-SSIS - SSIS package extension (public and private endpoint)



- The self - hosted integration runtime encrypts the credentials
  By using Windows DPAPI and saves the credentials locally. If multiple nodes are set for high

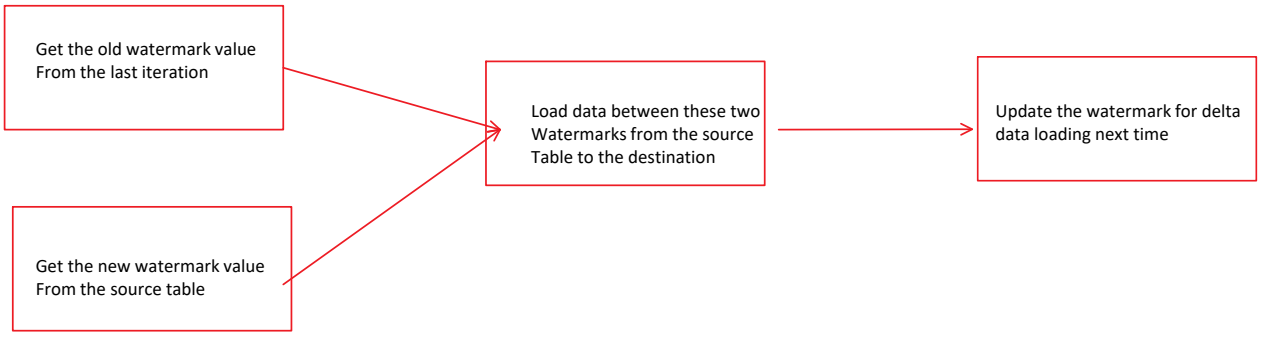
availability, the credentials are further synchronized across other nodes.
- The self hosted integration runtime can directly communicate over cloud based storage service like Azure blob storage over a secure HTTPS channel.

Hands-on Lab 03

1. Create SQL Server db (on-premise)
2. Create a Blob storage
3. Provision over self - hosted integration runtime on ADF
4. Create the pipeline

Incrementally (or delta) loading of data after an initial full data load is a widely used context.

1. **Delta data loading from database/data store by using watermark**

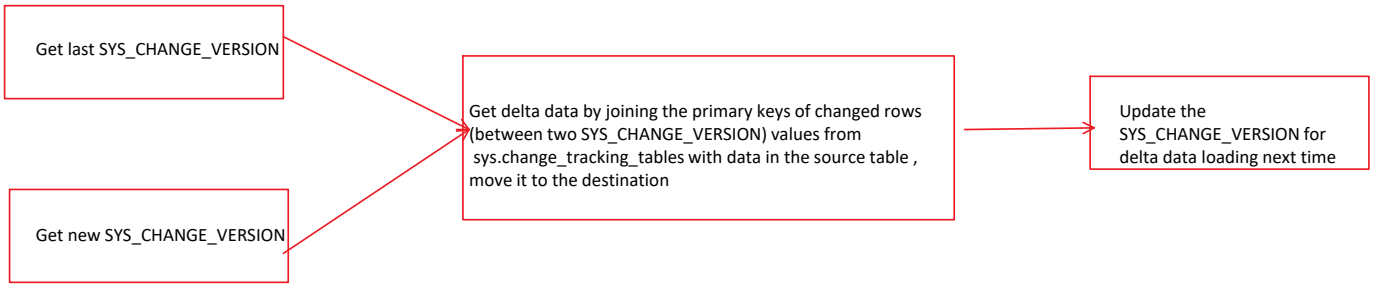


We can define a watermark in the source database. A watermark is a column which has the last updated timestamp or an incrementing key. The delta loading solution loads the changed data between the old watermark and a new watermark.

- Create two lookup activities - first lookup activity is to retrieve the last watermark value. Use the second Lookup activity to retrieve the new watermark value. These watermark values are passed to the Copy activity.
- Create a Copy activity which copies the rows from the source data store with the value of the watermark column > old watermark value and < the new watermark value. It also copies the the delta data from the source data store to Blob storage as a new file.
- Create a StoredProcedure activity which updates the watermark value for the pipeline which runs the next time.

2. **Delta table data loading from SQL db using The Change Tracking Technology**

Change tracking technology is the lightweight solution in SQL/ Azure SQL db which provides efficient change tracking mechanism for apps. It enables an application to easily identify the data which was inserted, updated or deleted.



1. Initial loading of historical data (run once)
2. Incremental loading of delta data on a schedule
   - Get old and new SYS_CHANGE_VERSION values
   - Load the delta data by joining the primary keys of the changed rows (between two SYS_CHANGE_VERSION values) from sys.change_tracking_tables with data in the source table

then move the delta table to destination.
- Update the sys_change_version for the delta loading next time.

Incremental Load

- Create two lookup activities to get old and new SYS_CHANGE_VERSION from Azure SQL db/ SQL db to pass it to copy activity.
- Create one copy activity to copy the inserted / updated/ deleted data between the two SYS_CHANGE_VERSION values from Azure SQL db to Blob storage
- Create one Stored Procedure activity to update the value of SYS_CHANGE_VERSION for next pipeline run.

3. **Loading of new & changed files by using LastModifiedDate & by using time partitioned folder or file name**

   ADF will scan all of the files from the data source store, can apply the filters over by their last modified data and to copy only the new and updated file since the last time to the destination store.

4. **Loading of new & changed files by using time partitioned folder or file name**

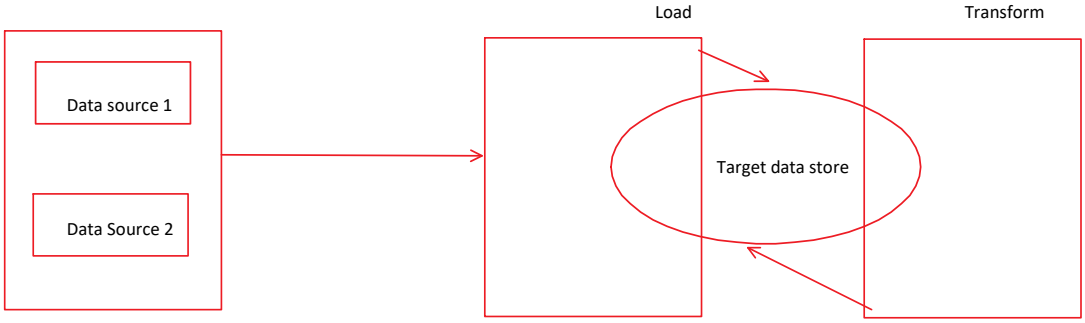
Copy of new files/ folders can be possible through ADF, when they're partitioned with timeslice information as part of file o r folder name

ELT in ADF Pipeline (Extract, Load and Transform)

Extract, Load and transform differs from the ETL mainly in where the transformation takes place. In the ELT pipeline, the transformation can occur in the target data store instead of using a separate transformation engine.

In the ELT pipeline, the transformation takes place in the target data store, the processing capabilities of the target data store are used to transform the data. This simplifies the ELT architecture by removing the transformation engine from the pipeline.

Extract



1. In Azure, the source flat files in scalable storage, such as HDFS, Azure Blob storage and ADLS gen2 are all can be used as data storage layer for extraction.
2. For in-memory based data processing platform like Spark, data warehouse tool hive, polybase, SQL query, pig/latin scripts , MR jobs etc can be used to query the source data.
3. In the ELT pipeline, the key difference is that the data store used to perform the transformation & is the same data store where the data is ultimately getting consumed. The data store reads directly from the scalable storage , instead of loading the data into its own proprietary storage. This approach skips the data copy present in ETL which often just can be time consuming operation for large data sets.
4. The data store only manages the schema of the data and applies the schema on read.
5. The final phase of ELT pipeline, is to transform the source data into a final format which is more efficient for the types of queries which need to be supported.

ADF (is a code - free ETL/ ELT tool)

1. **Ingest**

- **The data can ingested on multi-cloud and on-premise based hybrid storage using Copy data**
- **There're 100+ native connectors**
- **Serverless autoscale**
- **Use wizard for quick copy of data**

2. **Control Flow**
- **Design code free data pipeline**
- **Generate pipelines via SDK**
- **Utilize the workflow constructs - loops, branches, conditional execution, variables and**

parameters.

3. **Data Flow**
   - Code free data transformations which execute on Spark cluster
   - Scaled out (add more cluster instances) in Azure integration runtime
   - Generate the data flows via SDK
   - Designers UI is provided for data engineers and analysts

4. **Schedule**
   - **Build and maintain operational schedules for all of the data pipelines**
   - **We can execute the ADF pipeline manually, programmatically, tumbling window and schedule basis.**

5. **Monitor**
   - **View active execution and pipeline history**
   - **Detail activity and data flow execution status**
   - **Establish the alerts and notifications**

Mapping data flows are visually designed data transformations in ADF. Data flows allow data engineers to develop data transformation logic without writing code. The resulting data flows are executed as activities within the ADF pipeline which uses scaled-out Apache spark clusters. Data Flow activities can be operationalized using existing Azure data factory scheduling, control flow and monitoring capabilities.

ADF Data flow data types

- Array
- Binary
- Boolean
- Complex
- Decimal
- Date
- Float
- Integer
- Long
- Map
- Short
- String
- Timestamp

In ADF, the mapping data flow are operationalized within the ADF pipelines using the data flow activity. All a user has to do is specify which integration runtime to use and pass in parameter values.

Schema Drift in ADF data flow activities

Schema drift is the case where the data sources often change metadata, fields, columns and types which can be added, removed, or changed on the fly.

Without handling for schema drift, the data flow becomes vulnerable to the upstream data source changes. Typical ETL patterns fail when the incoming columns and fields change because they tend to be tied with the source names.

To protect against the schema drift

- Define the sources which has mutable field names, data types , values and sizes.
- Define the transformation parameters which can work with data patterns instead of hard-coded fields and values.
- Define expressions which can understand patterns to match the incoming fields, instead of using the named fields.
- Schema drift can be applied for both data source and sinks.

Transformation of the drifted columns

When the data flow in ADF has drifted columns, we can access the transformations with the methods like

- Use the 'byPosition' and 'byName' expressions to explicitly reference a column by name or position number.
- Add a column pattern in a derived column or aggregate transformation to match any combination of name, stream, position and origin or type.
- Add rule-based mapping in a SELECT or Sink transformation to match drifted columns or column aliases via the pattern.

Control Flow Activity

In ADF, Control Flow activity involves the orchestration of pipeline activities including the chaining of activities in a sequence , branching, defining the parameters at the pipeline level.

- The Control Flow activity also involves passing arguments while invoking the pipeline.
- It also includes custom-state passing and looping containers.
- The Control Flow activity defines the how control / sequence activities can pass through from one task to another task.

Features

1. Control Flow activity is an orchestration of pipeline activities in ADF.
2. The orchestration includes the chaining of activities in a sequence, branching and defining the parameters at the pipeline level.
3. It also helps to pass the arguments while invoking the pipeline on demand or from a trigger.

Example - Lookup activity in ADF.

1. Lookup activity can return upto 5000 rows, if the result set contains more records so then the first 5000 rows will be returned.
2. The lookup activity output supports up to 4 mb in size. Activity will fail if the size exceeds the limit.
3. The longest duration for lookup activity before timeout is 24 hours.

Foreach Activity

It's a control flow activity in ADF. The foreach activity can define the repeating control flow in ADF pipeline. This activity can be used to iterate over a collection and executes specified pipeline activities in a loop.

Control Flow Activities in ADF

Control Flow Activity Name	Purpose / Definition
Append Variable	Append variable activity could be used to add a value to an existing array variable defined on the ADF pipeline.
Set Variable	Set variable activity can be used to set the value of an existing variable of type string, boolean or array defined on a ADF pipeline
Execute variable	The execute pipeline activity allows ADF pipeline to invoke another pipeline
If condition	If condition activity allows directing pipeline execution, based on evaluation of certain expressions. The If condition activity provides the same functionality that an If statement provides in programming. It executes a set of activities when the condition evaluates to true and another set of activities when the condition evaluates to false.
Get Metadata	Get Metadata activity can be used to retrieve the metadata of any data in ADF
The Foreach activity	This activity defines the repeating control flow in ADF pipeline. This activity is used to iterate over a collection and executes the specified activities in a loop. The loop implementation of this activity is similar to Foreach looping structure in programming.
Lookup	Lookup activity can retrieve a dataset from any ADF supported data sources.
Filter	Filter activity can be used in a pipeline to apply a filter expression to an input array
Until	Executes the set of activities in a loop until the conditions associated with activity set to True.
Wait	Wait activity allows pausing pipeline execution for the specified time period.
Web Activity	Web Activity can be used to call custom REST endpoint from an ADF pipeline
Azure Function	Allows to run Azure Function in the ADF pipeline
Validation activity	We can use the validation activity in a pipeline to ensure the pipeline only continues execution once it has validated the attached dataset reference exists, that it meets the specified criteria or timeout has reached.
Webhook activity	It can control the execution of the pipelines through custom code, with the webhook activity, code can call an endpoint and pass it to a callback URL. The pipeline run waits for the callback invocation before it proceeds to the next activity.
Switch Activity	This switch activity provides the same functionality which is a switch statement in programming. It evaluates a set of activities corresponding to a case which matches the condition evaluation.

Webhooks are automated messages sent from the apps when an event occurs. Webhooks have a message, or payload or sent to the unique URI.

There're two kinds of Custom activities are available in ADF.

- Data movement activities to move data between the supported data source and sink data source.
- Data transformation activities to transform data using compute services (Azure HDInsight, Databricks)
- To move the data to/from the data store which the service doesn’t support or to transform /

process data in a way which's not supported by the service. We can create a custom activity with the data movement or transformation logic and use the activity in the pipeline.

Azure Data Lake

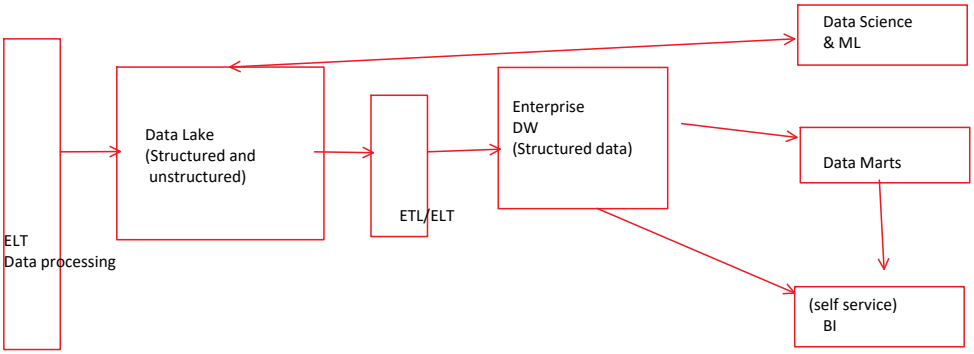
14 February 2023 09:31

A data lake provides the ability to store the data irrespective of volume, variety, veracity of the data. This data is able to be extra flexible means it provides a lot more latitude to do different kinds of analysis.

Benefits of Data lake

- Data lake supports the growing thrust of data analysis and data science models as well as critical requirement of data governance.
- It facilitates the modern data management platform built on enterprise scale platform which provides easy access for business users.
- Provides granular level access control based on the role based permission level.

Data Swamp - A data swamp happens when a data lake consists of miscellaneous data which no longer has any sort of structure. A data swamp can occur when adequate data quality and data governance measures are not implemented. Sometimes, a data swamp can also arise from a data warehouse due to hybrid models.



- If a data lake holds too much of data in a poorly organized manner without suitable metadata management and reliable data governance, relevant data becomes increasingly difficult to find.
- The information content of the data lake decreases, even though new data is constantly being added. A lack of lifecycle management of the data also leads to the silting up of a data lake.
- Data loses its relevance, if the data still remains in the data lake, more and more data with a lack of relevance accumulates over long periods of time.
- Incorrect time stamps of data set also leads to information which cant be found or evaluated.

Features of Data Swamp

1. Big Data without any organization and documentation through a data catalog or role concept.
2. Missing metadata information of the structured and unstructured data
3. Outdated and faulty data
4. No CDO or product management platform users
5. Missing or broken relationships between the datasets.

Prevention aspects

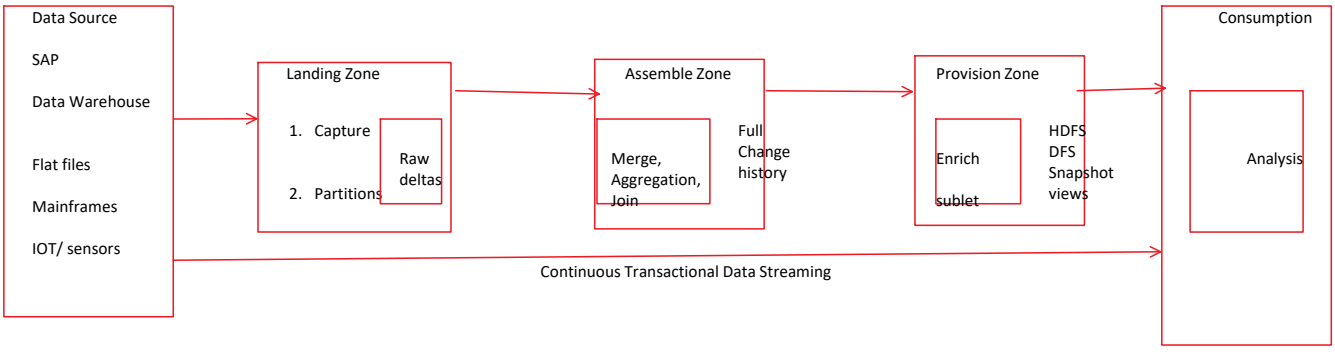
1. Create data catalog which builds the actual clarity of the data. It ensures that data reaches to the right personas based on RBAC.
2. Faulty/old data should be archived
3. Labelling of data origin, metadata labelling and meaningful nomenclature.

Data Marts

A data mart is curated subset of data which's generated for analytics and BI users. Data Marts are often created as repository of pertinent information for a subgroup of workers or a particular use case.

Difference between Data Mart and DW

- Slow and overloaded data warehouses are often the underlying reason for the creation of data marts and frequently serve as their underlying data source.
- Often, the data volumes and analytics use cases increase, enterprises cant serve every analytics use case without degrading the performance of their data warehouse, so they export a subset of data to the mart for analytics.



a) Massive volumes of structured and unstructured data like ERP transactions, and weblogs can be stored in a cost effective manner.
b) Data is available for use for faster transactions by keeping it in a raw state
c) A broader range of data can be analysed in new ways to gain unexpected and previously unavailable insights.

	Data Lake	Data Warehouse
Data Storage	A data lake contains all of an organization's data In a raw, unstructured format and can store the data for indefinite period of time. Even can be accessible for immediate or future use	A data warehouse contains the structured data such that it can be cleaned and processed, ready for strategic analysis based on predefined business needs.
Users / personas	Data from data lake with its huge volume of unstructured data can be used by data engineers, data scientists who prefer to study data in its raw form to gain new business insights	Data from DW is typically being accessed by managers, business stakeholders looking for the quick insights from the business KPI and as the data has been structured to provide answers to the pre-determined queries for analysis
Schema	Schema is defined after the data is stored in the data lake, unlike data warehouse making the process of capturing and storing the data faster.	In DW, the schema is defined before the data is stored, the lengthens the time it takes to process the data, but once it's completed, the data which made in DW, is ready as consistent, confident to use across the org.
Processing	ELT (Extract, load, transform), in this process the data is extracted from the source for storage in the data lake, and it's structured only while required	ETL (Extract, Transform and Load). In this process, data is extracted from its sources, scrubbed, parsed and then make it structured for business end analysis
Cost	Storage costs for fairly cheaper in a data lake, also less time consuming to manage which reduces the operational costs	Data warehouse cost much more than the data lakes, and required more time to manage, resulting in additional operational cost.

Scenario	Azure Storage Solution
Massively scalable and secure objects for storing into cloud and utilize for cloud-native workloads, archives , data lakes , high performance computing and to store massive volumes of data in the form of objects	Azure Blob storage
Massively scalable and secure data lake solution for the high performance analytical workloads which requires the support for both blob and file storage together and can provide the granular level access control and policy	Azure Data Lake storage
Simple, secure and serverless enterprise-grade cloud based file share, migrate on-premise file system or connect on-premise file share with Azure	Azure File Share
Store high performance real time streaming messages in pub-sub mechanism with fault tolerance and scalability support	Azure Queue storage
Store the random dataset non-compliant to RDBMS standard (no primary key/ unique key, no consistency and redundancy), non-compliant to Codd's rule. The storage required proper data partitioning for consistency	Azure Table Storage
High performance, durable block storage in the scenario of business critical apps	Azure Disk storage
Synchronization of on-premise file share with Azure file share including caching support for on-prem data as a hybrid cloud file share	Azure file sync
Appliances and solutions for transferring data into and out of Azure quickly and effectively	Azure Data Box (batch processing and real time stream processing)

ADLS Gen2

1. Hierarchical namespace - It organizes the objects/ files into the hierarchy of directories for efficient data access. There's a common object store naming convention used which slashes in the name to define the hierarchy directory structure.

- This hierarchy structure becomes real with ADLS gen2, operations such as renaming or deleting of a directory, there's no need to enumerate and process all objects which share the name prefix of the directory.
-

a) Performance wise it improves the directory management operations, which overall makes efficient job performance.

b) Management wise earlier, we can organize, manipulate the files through directories and subdirectories.
c) Security wise, it's enforceable to apply POSIX permission on the directories and individual files.
d) ABFS driver is used to access the data in the ADLS gen2, it's available within all Apache hadoop environment. The env includes Azure HDInsight, Azure Databricks and Azure Synapse analytics.

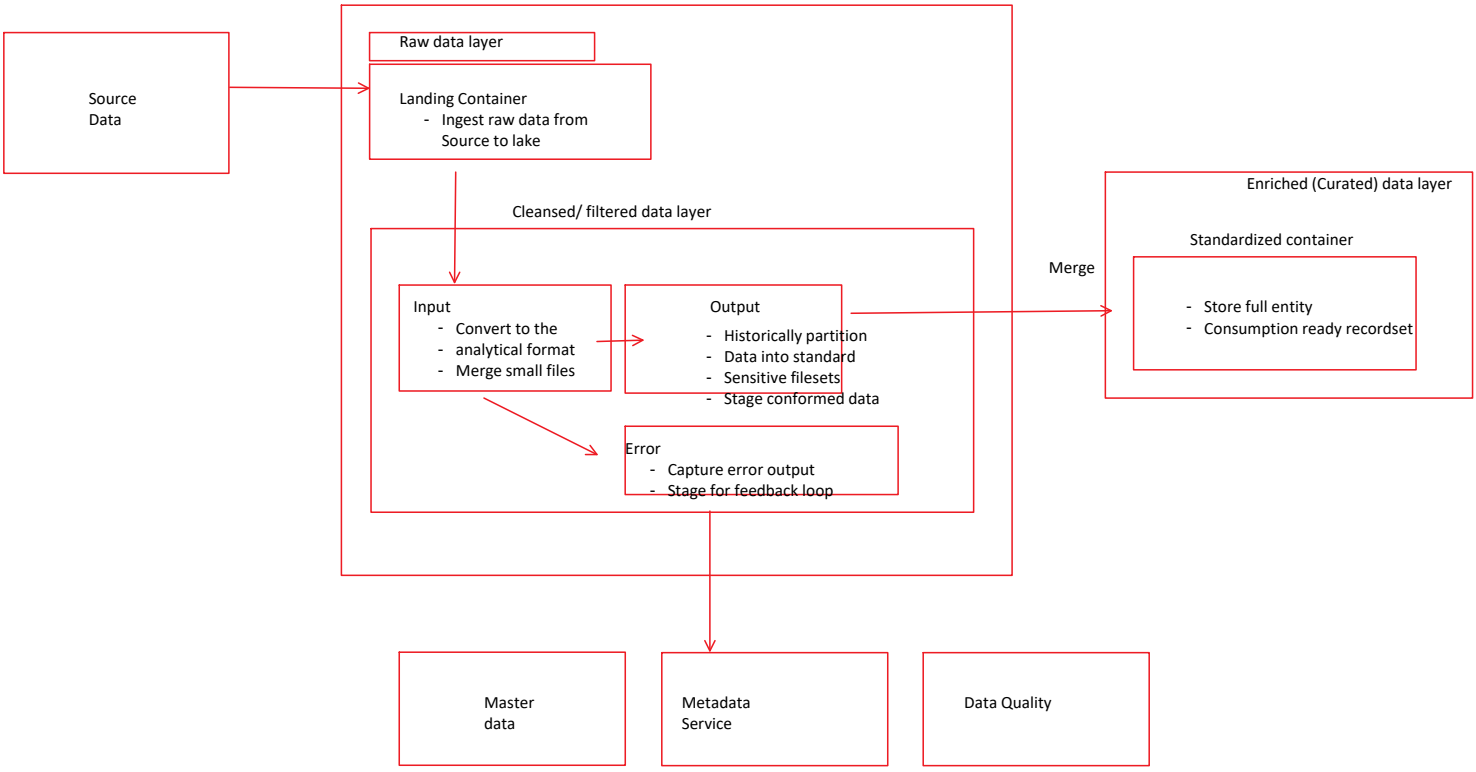
1. User delegation SAS - A user delegation SAS is secured with Azure AD credentials, also by the permission specified for the SAS. A user delegation SAS applies to Blob storage only.

2. Service SAS - A service SAS is secured with the storage account key. A service SAS delegates access to a resource in only one of the Azure storage services. Blob storage, Queue storage, Table storage or Azure files.

3. Account SAS - An account SAS is secured with the Storage account key. An Account SAS delegates access to resources in one more of the storage services. All of the operations available via a service or user delegation SAS are available via an account SAS.

From an user perspective

- Delegate access to service level operations
- Read, write and delete operations which aren't permitted with a service SAS.



Difference between ADLS Gen1 and ADLS Gen2

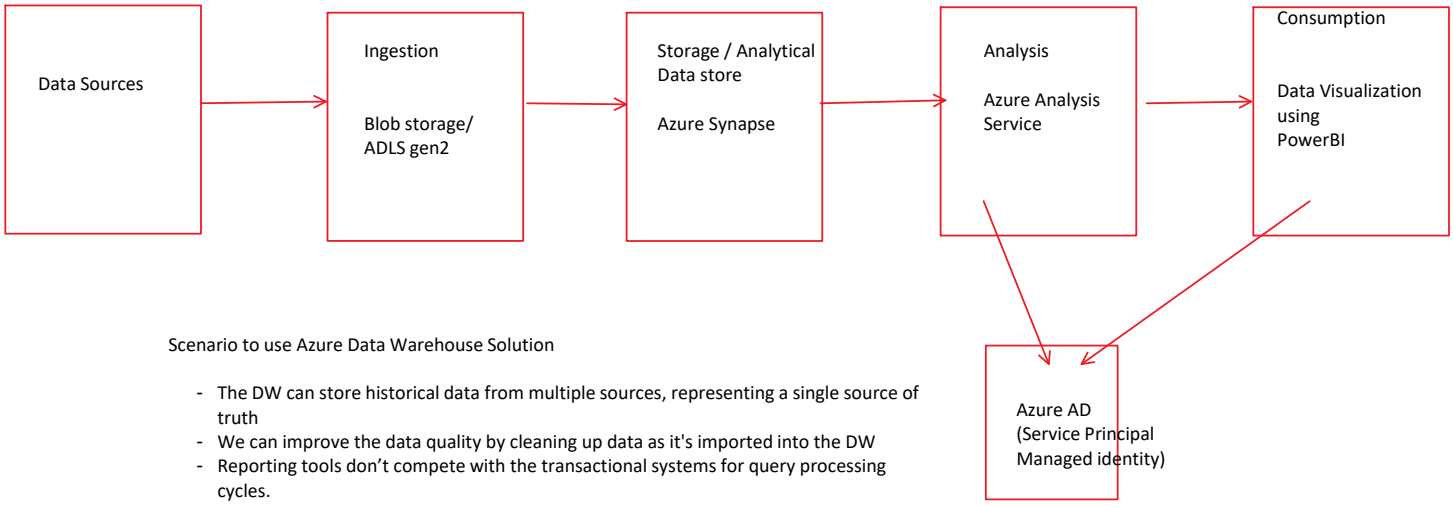
Difference	ADLS Gen1	ADLS gen2
Account Root Permissions	Permissions required for account - root - RX (minimum) , read only / read-execute or RWX (read-write-execute) to get an account root content view	An user with or without permission on root container can view the account content
RBAC roles and ACLs	All users in RBAC owner role are superusers. All other users (non-superusers) need to have permissions which should abide by the file/folder level ACL.	All users in RBAC storage blob data owner role are superusers. Rest of other users can be provided with different roles (contributor, reader) etc. which can govern their read, write and delete permission.
Store default permission	Permission for an item (file/directory) cant be inherited from the parent directory to child	File store permissions can be inherited if the default permission

	directory.	is set, on the parent directory / items before the child directory/items are being created. The file store permissions can be inherited from parent to child directory/folder.
Nested file / directory	Check whether the write-execute (wx) permissions for owner is imposed in the sub directory	Does not add wx permissions in the subdirectory
User provided permission	When a file/directory is created, the final permission will be same as the user provided permission.	File/directory is created, the final permission will be calculated based on the [user provided permission + umask] value.
Umask	Clients can apply umask on the permission for the new file/directory before the request sent	Clients can provide umask as the requested query params during the file and directory creation. The default umask will be applied 027 on the file/directory level.

Umask is used to modify the default ACLs are set on the child item when creating a file/folder. Umask is a 9-bit value on parent folders which contains an RWX value for owning user, owning group and other.

Azure SQL Data warehouse

14 February 2023 09:31



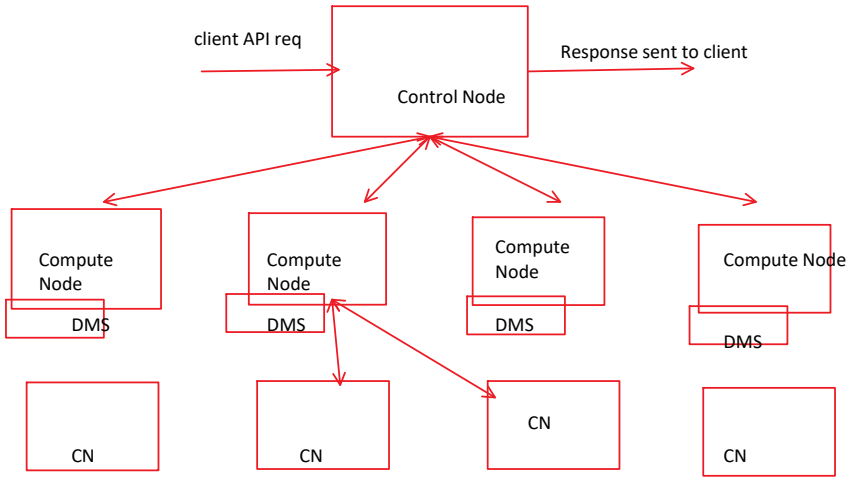
Scenario to use Azure Data Warehouse Solution

- The DW can store historical data from multiple sources, representing a single source of truth
- We can improve the data quality by cleaning up data as it's imported into the DW
- Reporting tools don't compete with the transactional systems for query processing cycles.
- A DW allows transactional system to focus on handling writes, while the DW also satisfies the majority of read requests.
- Data Warehouse can also consolidate data from different sources post transformation
- Data mining tools can find the hidden patterns in the data using automatic methodologies
- Data warehouse make it easier to provide secure access to authorized users, while restricting access to others. Business users don't need to access to the source data.
- DW makes it easier to create BI solution like OLAP cubes.

Scenario to choose for MPP architecture (Azure Synapse Analytics (formerly Azure SQL DW)

- If the data size exceed over 1 TB and are expected to continuously grow .
- If the data sizes are smaller, but the workloads are exceeding the available resources of the SMP multiprocessing (i.e. Azure SQL db etc.) then consider SQL DW (MPP architecture)
- MPP multiprocessing architecture (SQL DW) can be scaled out by adding more compute nodes (which have their own CPU, memory and I/O subsystems). There're physical limitations to scaling up a server which point scaling out is more flexible.
- In terms of querying, modelling and data partitioning, MPP solution as a complete decoupled compute and storage layer provides efficient data warehouse model.

Dedicated SQL Pool Architecture



- In the dedicated SQL pool, the control node works as the brain & orchestrator of the MPP engine
- As client API request, sent to dedicated sQL pool, the control node processes the query and converts the code based on distributed SQL plan. It's being executed on cost based optimization engine.
- After the DSQL plan has been generated, for each subsequent step, the control node sends the command to execute in each of the compute resources.
- The compute nodes are the worker nodes. They run commands as provided to them from the Control node.
- Compute node is measured using the SQL data warehouse units (DWU).
- The smallest compute unit (100 DWU) consists of control node and a compute node.
- Within the control & compute nodes, the data movement service (DMS) component can handle the movement of data between the Compute nodes themselves or from Compute nodes to the control node.
- DMS also includes the Polybase stack. An HDFS bridge is implemented within DMS to communicate with the HDFS file system.
- Polybase for SQL DW supports both Azure blob & ADLS gen2 storage.

Polybase in Azure SQL DW

Polybase is the fastest and most scalable SQL DW loading method which uses tSQL to combine and bridge the data across the RDBMS, Azure blob storage, ADLS gen2, hadoop distributed file system

- Polybase is one of the recommended option to load data into SQL DW.
- Polybase can load data from gzip, bzip2, snappy compressed files.
- Data loading onto SQL DW via polybase is not limited by the control node and as to scale out the DWU. The data transfer throughput also increases.
- We can use INSERT INTO clause in order to load incremental data into SQL DW. For full logging operation, when inserting into a populated partition which will impact on the load performance. The roll back operation on a large transaction can be expensive. Recommended to split over a big transaction into smaller batches.

Best Practices for Polybase for data loading into SQL DW

- A single polybase load operation provides the best performance.
- The load performance scales as we increase DWUs.
- Polybase automatically parallelizes the data load process, so we don't need to explicitly break the input data into multiple files and issue concurrent loads, unlike some traditional data loading methods. Each data reader automatically can read 512 MB data for each file of Azure blob storage and 256 MB for ADLS gen2.
- Multiple readers will not work against the compressed files - gzip files. Only a single reader can be used per gzip compressed file since uncompressing the file in the buffer is single threaded.

Partitioning as effective data loading methodology

1. Partitioning can also be used to improve the query performance.
2. A query which applies a filter to partitioned data can limit the scan to only qualified partitions. This method of filtering can avoid a full table scan and can only scan a smaller subset of data.
3. With the introduction of clustered columnstore indexes, the predicate elimination performance benefits are less beneficial, for e.g. if the sales fact table is partitioned into 36 months using the sales date field, then querying over the filter on the sale date can skip searching for partitions which don't match the filter.

Distributed table

A distributed table appears a single table but the rows are actually stored across 60 distributions. The rows are distributed with a hash or round-robin algorithm.

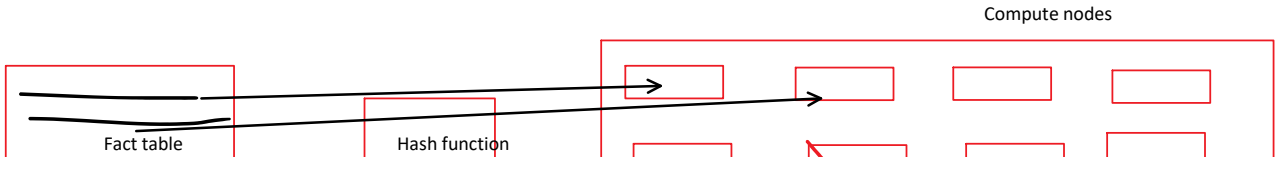
Hash-distribution improves the query performance on large fact tables.

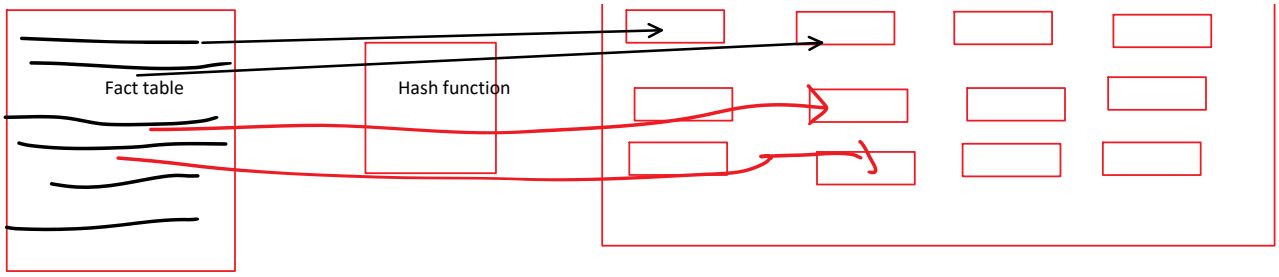
Round-robin distribution is useful for improving the data loading speed.

Another table storage option is to replicate small tables across all the compute nodes.

1. Hash distributed table (for Fact table)

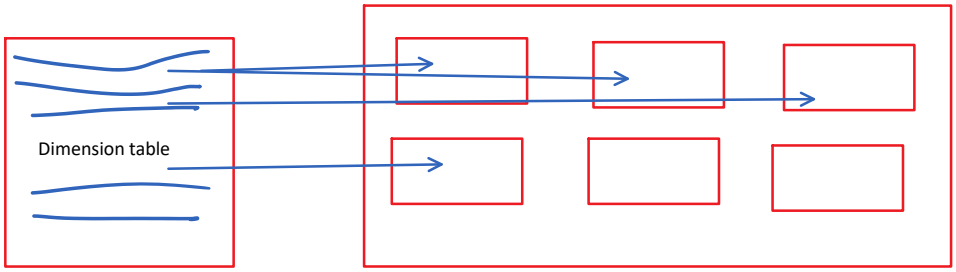
A hash-distributed table distributes table rows across the compute nodes by using a deterministic hash function to assign each row to one distribution.





- A hash-distribution table rows maps across the Compute nodes by using a deterministic hash function to assign each row to one direction.
- Identical values always hash to the same distribution unit.
- SQL DW analytics, has a built-in knowledge of the row location of tables.
- In dedicated SQL pool, the knowledge is utilized to minimize the data movement between the queries, which also improves the query performance.
- For star schema, large fact tables are designed with the hash distributed table format.
- When the table size on disk 2-5 GB, table has large no of rows and frequent insert, update and delete operations.

2. Round-robin distribution (for dimension tables)



- Distributes the table rows evenly across all distributions
- The assignment of rows is used for distribution is random
- Rows with equal values are not guaranteed to be assigned in the same direction.

Scenarios

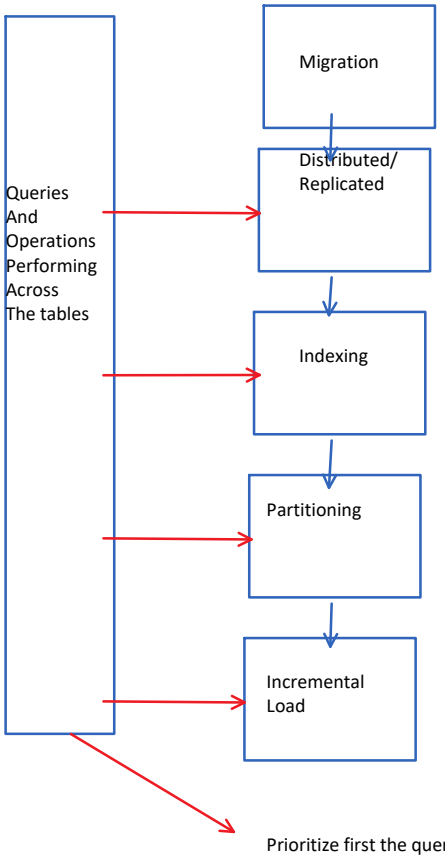
1. Design the dimension table with large no of columns
2. There's no specific joining key
3. When the table is temporarily stored staging table
4. There's no common joining key available with other key tables.

Table Distribution	Hash-distributed	Round-robin	Replicated
Scenario	Large fact tables consisting of large no of Rows	Dimension tables with large no of columns	Dimension tables with full copy option from round-robin value
Size	Size of the fact table is at least 5 GB or more	Dimension table size is at least of 2 GB	Replicated tables 2 GB or less
Performance efficiency	Most performance efficient since hash distribution key is imposed using the distribution column	Less performance efficient since the table rows are distributed randomly over the compute nodes of the cluster, not matching same order the compute nodes while distribution of rows.	Full table copy is performed over the same compute node, less data movement (DMS) , maximize the query throughput and performance

Designing of dedicated SQL pool & tables

- Objective (best practices)

1. Load the data as fast as possible to a staging table
2. Limit the data movement when using "Group By" and "Join" tables.
3. Optimize the table for read performance with the right indexing
4. Improved performance when applying filter on the partition key and can help manage the data lifecycle.
5. Minimize the disruption for the business users.



Queries and Operations performed across the tables

- Joining one or two fact tables with dimension tables, filtering of the combined table then appending the results into a data mart
- Making large or small updates into the fact tables (sales table)
- Appending only data into the tables

Prioritize first the queries and transformations in dedicated SQL pool, second the incremental load.

Polybase in ELT based data processing platform for dedicated SQL pool

Polybase is actually the data loading strategy through which we can load the data into Azure SQL datawarehouse (dedicated SQL pool) for Synapse analytics.

So, the SQL pool supports various loading methods including BCP(bulk copy utility), SQL BulkCopy API for loading data through Polybase.

1. Polybase works as an intermediate blob storage through which data after migrating from the data source can be stored for temporary purpose.
2. Once the migrated data from the source db is held through polybase and loaded successfully, then it's being transferred to the dedicated SQL pool/ Azure SQL DW db.

Process to implement a Polybase ELT for dedicated SQL pool

1. Extract the source data into text files
2. Land the data into Azure blob storage or data lake store
3. Prepare the data for loading
4. Load the data into dedicated SQL pool staging tables using Polybase
5. Transform the data
6. Insert the data into production tables.

Define External tables recommended for Polybase

We should define external tables in the DW. Polybase uses external tables to define and access the data in Azure storage. An external table is similar to a database view. The external table contains the table schema and points to data stored outside of data warehouse.

Defining the external tables involves specifying the data source, the format of text files and the table definitions.

- **CREATE EXTERNAL DATA SOURCE**
- **CREATE EXTERNAL FILE FORMAT**
- **CREATE EXTERNAL TABLE**

Loading the data into dedicated SQL pool staging tables using Polybase

It's recommended as best practice to load data into a staging table. Staging tables allow us to handle errors without interfering with the production tables. A staging table also gives the opportunity to use SQL pool built-in distributed query processing capabilities for data transformations before inserting the data into production tables.

a) Polybase with TSQL
b) Polybase with SSIS
c) Polybase with ADF
d) Polybase with Azure Databricks

While inserting the data into production tables, use INSERT INTO.. SELECT statement moves the data from the staging table to the permanent table.

As we design an ETL process, we can extract 1000 rows from the table to a file then move it to Azure, then loading it into a staging table.

Singleton updates refers to the smaller transaction batch loads should be grouped together into large batches to optimize the Synapse SQL pools processing capabilities. A one-off load to a small table with an INSERT statement may be the best approach, if it's one off.

However, if it's required to load thousands or millions of rows throughout the day, then singleton INSERTS aren't optimal solution for MPP processing SQL DW platform. One way to solve the issue is to develop one process which should write the outputs of an INSERT statement to a file, and then another process to periodically load the file to take advantage of parallelism of Azure Synapse Analytics.

Hands - on Lab

Load the data into Azure Synapse Analytics from Azure SQL db using Azure Data Factory / Synapse Pipeline

Pre-requisites

1. Azure SQL db (with sample db)
2. Azure Synapse Analytics workspace (dedicated SQL pool)
3. Azure Data factory / Synapse Pipeline
4. Azure blob storage (Staging storage i.e. Polybase)

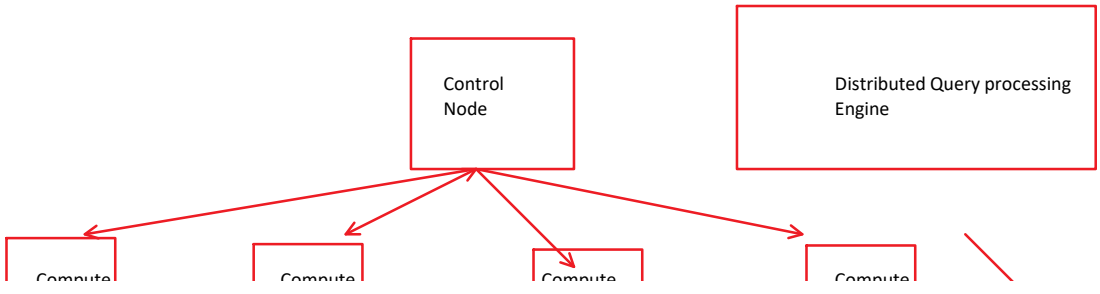
Serverless SQL Pool Architecture components of Azure Synapse

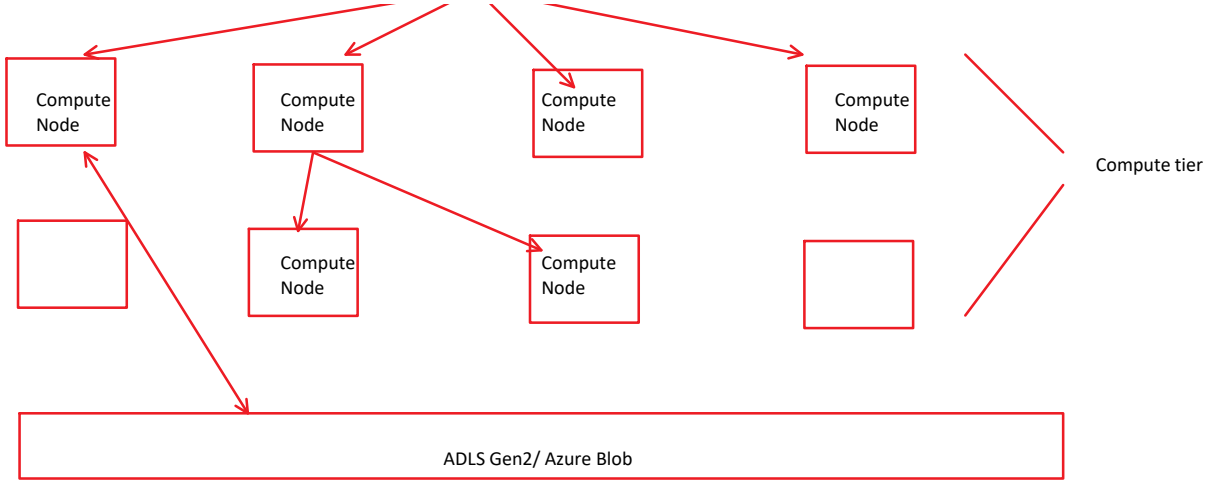
1. The Serverless SQL pool uses the distributed query engine (DQP) to optimize the queries for parallel processing and then pass operations to the compute node to perform the work in parallel.

2. The Serverless SQL pool Control Node utilizes the distributed Query processing engine to optimize, orchestrate distributed execution of user query by splitting it into smaller queries. Then this smaller queries will be executed on Compute nodes.

3. Each small query is called task and represented as distributed execution unit. It reads the files from storage, can join the results from other tasks, groups and orders data retrieved from other tasks.

4. The compute nodes store all user data in Azure storage and execute the parallel queries. The DMS is a system level internal service which moves the data across the nodes as necessary to execute the queries in parallel and return accurate results.





Benefits of Serverless SQL Pool in Synapse Workspace

1. With the decoupled storage and compute, when using Synapse SQL, enterprise can get benefitted from the independent sizing of compute power irrespective of storage requirements.
2. For serverless SQL pool, scaling is done automatically.
3. We can scale out or scale in the compute power within a dedicated SQL pool without moving the data.
4. We can pause the compute capacity while leaving the data intact so that we can pay for storage.
5. Resume back the compute capacity during the operational hours.

Benefits of Control Node in Serverless SQL pool

1. For Serverless SQL pool, the DQP engine runs the control node to optimize and co-ordinate the distributed execution of user query by splitting it into smaller sub-queries which will be executed on Compute nodes.

2. It can also assign the sets of file/storage to be processed by each node.
3. Whenever the TSQL query submitted, the control node can optimize and co-ordinate the parallel queries.

Best Practices for data loading scenario for serverless SQL pool in Synapse Workspace

Prepare the data to load into Azure storage / ADLS gen2

a) To minimize the latency, collocate the storage layer and the dedicated SQL pool.
b) In order to avoid out-of memory error, while exporting data into ORC, RC or gZIP format.
c) All of this file formats having different performance characteristics, for fastest load, use compressed delimited text files. Better to use file formats like Parquet, (columnar file format with compression support).

Compute Nodes benefits in Synapse Serverless SQL pool

a) Serverless SQL pool is a distributed query processing system, built for large scale data and computational functions. Serverless SQL pool enables to analyse the big data in seconds to minutes depending on the workload.

b) Serverless SQL pool is serverless, hence there's no infrastructure requirement to setup clusters to maintain. A default endpoint for this service is provided within every Azure Synapse workspace, so that we can start querying the data as soon as the workspace is created.

c) For serverless SQL pool, each compute node is assigned task and set the files with filters to execute the tasks on, the task is distributed on query execution unit which's actually being part of query submitted by the user.
d) Automatic scaling is in effect to define the Compute nodes are being utilized to execute the user query.

Table Persistency Concepts in SQL DW

Data for dedicated and serverless SQL pool for Azure synapse analytics, is stored in ADLS gen2/ Azure storage. There're three kinds of tables can be created in SQL DW while retrieving the data from underlying ADLS gen2 / blob storage.

a) **Regular table** - A regular table can store the data in Azure storage as part of dedicated SQL pool. The table and the data persist regardless of whether the session is open.

b) Temporary table - A temporary table is only exists for the duration of the session. We can use a temp table to prevent other users from seeing temporary results and also to reduce the requirement of cleanup.

c) **External table** - An external table points the data located in Azure Blob storage/ ADLS gen2. When used with the CREATE TABLE AS SELECT (CTAS), selecting from an external table imports data into the dedicated SQL pool.

The CREATE TABLE AS SELECT (CTAS) statement

It's one of important SQL feature which's fully parallelized operations that creates a new table based on output of select statement. CTAS is most simplest and fastest way to create a copy of the table.

Benefits of CTAS statement

1. Recreate a table with hash-distributed column
2. Recreate a table as replicated
3. Create a cluster columnstore index/columnstore index on just columns in the table.
4. Query or import the external data.

Supported Data types for Azure Synapse / SQL DW

- Int32
- Int16
- String
- Decimal
- Float
- Datetime
- Boolean
- Guid
- Binary

Types of External Tables

1. Hadoop External Tables - We can use to read and export data in various formats such as CSV, Parquet and ORC. Hadoop external tables are available in dedicated SQL pools. But they're not available in Serverless SQL pools of Synapse.

2. Native external tables - We can use to read and export data in various data formats like CSV, parquet. Native external tables are available in serverless SQL pools, and they're in public preview in dedicated SQL pools. Writing and exporting data using CETAS and native external tables is available in the serverless SQL pool.

Partitioning Features in dedicated SQL pool of Synapse Workspace

Table partitioning allows us to divide the data into smaller groups of data. In most of scenarios, table partitions in dedicated SQL pool are created based on date column.

Partitioning is supported on all dedicated SQL pool table types, including the clustered columnstore indexes and clustered indexes and heap.

Partitioning is also supported on all distribution types - hash distribution, round-robin distribution and replicated.

Partitioning can benefit data maintenance and query performance. Whether it benefits both of just one is dependent on how data is loaded and whether the same column can be used for both of the purpose.

Benefits of Partitioning in dedicated SQL pool

1. Improve the efficiency and query performance of loading data by using partition deletion, merging and switching
- A partition switch can be used quickly to remove / replace a section of table.
- Using partitioning during the overall load process can substantially improve the performance
- Recommended approach while deleting the rows over a million of rows of table, just to optimize the time and reduce the risk of large transactions which can take a long time to rollback, optimized approach is to drop the oldest partition of data.
- Applied a filter to the partitioned data can limit the scan to the qualifying partitions while overall can improve the query performance.

2. Ensure the appropriate number of partitions are getting created and use partitions to be carefully chosen.
3. Creating partitions on clustered columnstore index tables.

Statistics of Synapse SQL

The dedicated SQL pool query optimizer works as a cost based optimizer. It compares the cost of various query plans, then chooses the plan with the lowest cost. In most of time, the plan with the fastest query execution is being selected.

AUTO_CREATE_STATISTICS flag is used to configure on dedicated SQL pool, then it'll trigger the automated statistics creation.

- UPDATE
- DELETE
- EXPLAIN

Azure Databricks

14 February 2023 09:31

Scenario	Apache Hadoop	Apache Spark
Context	Apache hadoop is the open source big data distribution Which allows the users to manage the big data sets over the nodes to solve the Vast and intricate data problems	Apache Spark is also an open source distributed stream processing engine which's used for in-memory based big data analytics and transformation
Core features	Highly scalable, cost effective solution for storing of structured and unstructured data but it requires good amount of infra on bare metal servers in order to execute the data pipeline faster	Spark can split up the large tasks across the different nodes. It tends to perform faster than hadoop and uses the memory (RAM) to cache the data and process the data instead of filesystem. It enables Spark to handle the use cases which hadoop cant do.
Benefits	a) Data protection is available even there's hardware failure b) Vast scalability from a single server to thousands of machines c) Real time analytics for historical data and decision making processes	A unified in-memory based real time data processing engine which supports SQL queries, streaming data , ML and graph processing
Components	a) HDFS - primary data storage platform which manages the large datasets running on the commodity hardware. b) It also provides high-throughput data across and with high fault tolerance. c) Hadoop Mapreduce - splits the input data processing tasks into smaller onces, distributes the smaller tasks across the different nodes which executes each tasks across the different nodes. d) YARN - Cluster resource manager which scheduled the tasks and allocates the resources (CPU, memory) to applications. e) Hadoop core - set of common libraries and utilities like of modules HDFS, YARN, MR etc.	Apache Spark consists of the components a) Spark Core - underlying execution engine which schedules and dispatches tasks and co-ordinates the I/O operations. b) Spark SQL - gathers information about the structured data to enable users to optimize data processing c) Spark Streaming / Structured streaming - Both of streams could add data stream processing capability. Spark streaming takes data from the different streaming sources and divides it into micro-batches for a continuous stream. Structured streaming is built on top of Spark SQL which reduces latency and simplifies programming. d) GraphX - user friendly graph based data processing engine which enables query building capability, it can used for data modification, analysis of scalable and graph based structured data.
Core data processing and job execution	Hadoop mapreduce process the on disk. Mapreduce jobs are comparatively slower than Apache spark.	Spark is the hadoop enhancement to MapReduce. The primary difference between MapReduce and Spark is that Spark processes and retains the data in-memory for the subsequent steps.
	Data is being processing in two stages based on key-value pairs. - Map - Reduce	Spark creates the directed acyclic graph (DAG) to schedule tasks and orchestrates the cluster nodes across the hadoop cluster.

Spark Components Overview

a) Spark applications run as an independent set of processes on a cluster, which can be co-ordinated by the SparkContext object in the main program(driver program).
b) Specifically, while running on the cluster, the SparkContext can connect to several types of cluster managers (wither Spark's own standalone cluster manager or YARN, mesos) which allocates resources across the applications.
c) Once connected, Spark acquires executors on nodes of the cluster. Which are the processes that can run the computations on nodes of the cluster.
d) These processes can execute the core computations and store data for applications.
e) Next, it can send the application code (Jar, python files) passes over to the SparkContext to the executors.
f) Finally, SparkContext send the tasks to the executor to run.

Spark Driver Program (SparkContext)	This process running as the main() method for the application and Is responsible to create the SparkContext class object. Once the SparkContext class is being initialized, it can start interacting with the executors for job processing
Cluster Manager	An external Service for acquiring the resources on the Spark cluster (e.g. standalone mode, Mesos and Kubernetes)
Worker nodes	Any node which can run the application code for the cluster
Executor	A process launched for an application on worker node. It runs the tasks and keeps the data on memory or disk storage across them. Each application has their own executors.
Job	A parallel computation unit consists of multiple tasks which gets spawned over in response to Spark action (e.g. Save, Collect etc)
Tasks	Core unit of work which will be transferred to one executor
Stage	Each job passes over different stages. It gets divided into the smaller set of tasks called as stages which depends on each other.
Application	User program built on Spark. It consists of a driver program and executor on the cluster

Since, Spark 2.0 SparkSession has become the entry point to Spark in order to work with RDD, Dataframe and dataset.
Prior to Spark 2.x, SparkContext was used to be an entry point.

SparkSession -

SparkSession is introduced in version Spark 2.x and it's an entry point to underlying Spark functionality in order to programmatically create Spark RDD, dataframe and dataset.
SparkSession's object spark is the default variable available in spark-shell and it can be created programmatically using the SparkSession builder pattern.

- SparkSession includes the APIs in different contexts -
- SparkContext
- SQLContext
- StreamingContext
- HiveContext

SparkSession instance would be first statement when writing the programs on RDD, dataframe and dataset. SparkSession will be created using the SparkSession.builder() method. We can also use SparkSession.newSession() in order to create a new session for spark.

- With Spark 2.0, a new class SparkSession (pyspark.sql import SparkSession) has been introduced. SparkSession is a combined class for all different contexts like SQLcontext, HiveContext etc.

RDD and Lazy Evaluation

Spark revolves around the concepts of RDDs which 's a fault tolerant, distributed collection of elements that can be operated in parallel. There're two ways to create over RDD.

a) Parallelize() function - an existing collection of dataset can be executed in parallel in the Spark driver program
b) Referencing a dataset in an external storage program - HDFS, HBase of any other data source offering a hadoop inputformat.
c) Existing RDDs.

Operations on RDD

1. Transformation
2. Action

Transformation s in RDD

In Spark, the role of transformation is to create a new dataset from an existing one. The transformation are being considered as **lazy evaluation** as they're being computed when an action requires a result to be returned to the driver program.

Transformation Function	Description
Map(func)	It returns a new distributed dataset formed by passing each element of the source through a function func
Filter(func)	It returns a new dataset formed by selecting those elements of the source on which the func returns true.
MapPartitions(func)	It's similar to Map, but it runs separately on each partition block of the RDD. So that the func should return a sequence rather than a single item.
Union(otherDataset)	It returns a new dataset which contains the union of elements in the source dataset and arguments.
Distinct(numPartitions)	It returns a new dataset which contains the distinct elements in the source dataset and arguments
GroupByKey(NumParitions)	It returns a dataset(K, iterable) pairs which when called on a dataset of (key, value) pairs.
ReduceByKey(NumPartitions)	When called on a dataset of (K,V) pairs, returns a dataset (K,V) pairs, returns a key/value pairs where the value of each key are aggregated using the given reduce function func, which must be the type of (v,v) => v
SortByKey(numPartitions).ascending	Returns the dataset of key/value pairs sorted over the keys in ascending/descending order.

Actions in RDD

In Spark, the role of Action is to return the value to driver program after running the computation on the dataset/dataframe.

Actions	Description
Reduce(func)	It aggregates the elements of the dataset using a function(which takes two arguments and returns over one argument).
Collect()	It returns all the elements of the dataset as an array at the driver program. This is usually useful after a filter or other operations which returns sufficiently the smaller subset of data
Count()	It returns the number of elements on the dataset
First()	Returns the first element, row of the dataset
Take(n)	It returns an array with the n number of elements in the dataset
saveAsTextFile(Path)	It's used to write the elements of the dataset as a text file in a given directory of the local filesystem. HDFS/S3, Spark calls toString() on each element to convert it to a line of text in the file.
saveAsSequenceFile(Path)	It's used to write the elements of the dataset in a simple format such as hadoop sequence file in a given path of the local filesystem
CountByKey()	It's only available RDDs of type (K,V) pairs, thus it can return the hashmap of (K, int) pairs with the count of each key.
Foreach(func)	It returns the function func on each element of the dataset for side transformation like updating the input dataset or changing the values of the variables.
takeSample/takeOrdered	It returns an array with random sample of num elements in the dataset with/without replacement

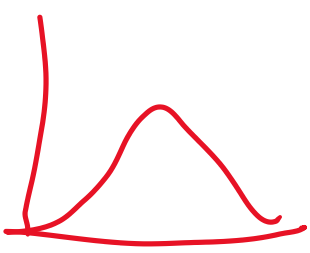
RDD in Spark is the fundamental data structure which consists of immutable collection of objects and which computes on the different nodes of the cluster.

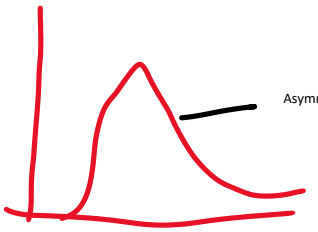
Since RDD is resilient , as fault tolerant, there's a lineage graph (DAG) is being created and it can recompute any missing or damaged dataset partitions due to the node failure.

DAG Scheduler operation

In Spark RDD, the DAG works on every edge which directs the job to operate from earlier to the next job execution in the sequence. On calling of an Action, the created DAG submits the job to the DAG scheduler so that scheduler can further split the graph into stages of the task.

Spark directed acyclic graph (DAG) allows the user to dive into the stage and expand every details of the job execution stage.





Asymmetry distribution

In Pyspark, Skewness is referred to as statistical measure of asymmetric distribution of data, while kurtosis helps to determine if the distribution is heavy-tailed compared to a normal distribution.

PySpark Date & Time Functions

Date and Time functions are very important functions for pyspark SQL which can perform ETL/ELT operations.
PySpark SQL provides different kinds of Date and Time functions like

a) Date functions
b) Timestamp functions
c) Date & timestamp window function

Current_date() - retrieve the current system date
Date_format() - to retrieve the date and convert the from yyyy-mm-dd to mm-dd-yyyy format
To_date() - converts a string to date datatype. Date datatype can be converted to any specific type.
Datediff() - return the difference between two dates
Months_between() - returns the months between two dates
Trunc() - truncate the date at specified unit
Add_months(), date_add(), date_sub() - add or substract date or month of given input
Year(), month(), next_day(), week_of_year(), dayofweek(), dayofmonth(), dayofyear()

Scenario to choose for Azure HDInsight & Azure Databricks

Azure HDInsight	Azure Databricks
It's a Apache hadoop managed platform on Azure Including other hadoop core components like Hive, Pig, Spark, Kafka etc.	The Databricks platform provides around five times more performance than the open source Apache Spark
Cost effective and provides the benefit to optimize and process the big data since we can choose the type of services based cluster (for e.g. provision Spark type HDI cluster, Kafka based HDI cluster, Hbase/Storm based HDI cluster etc.)	Azure Databricks is the best choice for enterprise running Azure cloud services as the Spark based analytics platform specially optimized for Azure
Integrates with Azure AD, AD domain services (ADDS) are available	Works with premium Spark cluster, the premium spark cluster is faster than the open source spark cluster. - Azure Databricks is the PaaS solution, it doesn't require a lot of administration efforts after the initial deployment. It provides the core security through Azure AD without any requirement of the custom configuration.
Azure HDInsight provides the most popular OSS frameworks (hadoop, pig hive, kafka, sqoop, storm etc.)	Azure Databricks can offer various data integration capabilities - Data Engineering platform - Data streaming platform - Data Science and AI platform - Delta Lake - Lakehouse
Per cluster based time pricing	Billing is calculated per cluster time VM (VM cost + DBU (Databricks unit - infrastructure compute capacity)
Apache Spark is available through Spark type of HDI cluster	Apache Spark is the core component which provides 5 times more performance than Azure HDI
We can work with Jupyter notebook, execute spark jobs and also through cluster shell	In Databricks, we can work with Jupyter notebooks, pyspark for Spark SQL and Spark streaming jobs
In Azure HDInsight, the spark/hadoop cluster cant be paused	In Azure Databricks interactive cluster, Spark cluster can be paused, resumed and terminated.

Azure Synapse Analytics (DW)	Azure Databricks
Azure Synapse Analytics provides limitless analytics services and it's the managed Data warehouse solution on Azure	Azure Databricks is the managed Apache Spark analytics platform on Azure
Synapse is based on analytical data warehouse (OLAP based models, interacts with data mines, data marts, data warehouse)	Integrates with Azure to help to build optimized ML algorithms (tensorflow, Pytorch, Keras) etc. Mlflow can also be executed on Databricks Spark cluster along with the benefits of pyspark.
In Synapse, we can perform end to end SQL% based data analysis as offering MPP multiprocessing and designing fact, dimension and integrated tables. Since it's managed DW platform, it integrates with the full capacity of SQL and DW capability with top down approach. In Azure Synapse/DW, the pools (dedicated/serverless) are being used for enterprise data warehousing.	It's NOT a data warehouse solution, it's not possible to implement the full capacity of SQL and data warehouse capacity like building fact / dimension tables with star/snowflake schema. In Azure Databricks, we've full data transformation (map(), flatmap(), groupBy(), orderBy(), sort(), etc.) and actions (collect(), reduce(), count() etc.) various operations are available. But it doesn't have full support of TSQL (limited in Spark SQL).

Benefits/ Utilities of Azure Databricks

- Customers can fully embrace the Databricks platform to get the benefits of unified platform to build and deploy the data engineering, ML workflows and ML models and analytical dashboards which empowers the innovations and insights across the organization.

a) Azure Databricks workspace provides the UI for the tasks like interactive notebooks (Jupyter, Zeppelin notebooks)
b) SQL editors and dashboards
c) Data ingestion and data governance support (integration with Azure purview)
d) Data discovery and classification
e) ML based model development and deployment
f) Source control integration with Git

The Components of Azure Databricks

For other three personas of Azure Databricks, the following three environments, there're few common resources across the cluster to be used -

- Databricks Data Science and Engineering
- Databricks ML
- Databricks SQL

a) Workspaces - In Azure Databricks, the workspaces are two types

- An Azure Databricks deployment in the cloud, which's the unified environment for accessing all of the databricks resources.
- The organization can choose to have multiple workspaces or just one
- The UI for the databricks is the persona based environments, for e.g. the workspace we can implement includes the spark cluster, we can browse the notebooks and can access the libraries.

Resources in Azure Databricks workspace - Azure Databricks consists of the following resources

1. **Notebooks** - A web based interface to documents which contains the runnable commands, visualizations and narrative texts.
2. **Dashboards** - An interface which provides organized access to visualizations.
3. **Library** - A package of code available to the notebook or job running on the cluster. Databricks runtime includes many libraries and we can add our own libraries as well.
4. **Experiment** - It's the collection of MLFlow service which runs a training for ML model.
5. **Repo** - A folder whose contents are co-versioned together by syncing them to a remote Git repo.
6. **Databricks file system (DBFS)** - A filesystem abstraction layer on Azure Blob storage/ ADLS gen2 & it can contain (data files, libraries and images) and other directories. DBFS is automatically populated with some datasets which we can use it to perform the Spark data transformation.
7. **Table** - A representation of structured data in Azure databricks. We can query over the databricks tables with Apache Spark SQL and Apache Spark APIs.
8. **Metastore** - This component stores all the structured information of various tables and partitions in the DW including columns, column type information, the serializers and deserializers which're necessary to read the data and the corresponding files where the data is being stored.

   Each ADB deployment has a central Hive metastore accessible by all clusters to persist the table metadata.

Resources in Azure Databricks for Computation Management

1. **Cluster -** A set of computation resources and configurations on which we can run the notebooks and jobs.

There're two kinds of Azure Databricks cluster

1. All-purpose cluster - We can create all-purpose cluster using Azure portal UI, CLI, REST API. We can manually terminate and pause/resume all purpose cluster. Multiple users can share the all-purpose cluster to perform the interactive data analysis.

2. Job cluster - The Azure databricks job cluster creates we run the job on a new job cluster and terminates the cluster when the job is completed. We cant restart the job cluster.

3. **Databricks pool** - A set of idle, ready to use instances which can reduce the cluster start and autoscaling times. When attached to a pool, a cluster allocates the driver and worker nodes from the pool. If the pool doesn't have sufficient idle resources to accommodate the cluster's request. The pool can then expand by allocating new sources from the instance provider. When an attached cluster is terminated, the instance is used are returned to the pool and can be reused by different cluster.

4. Databricks runtime - Databricks runtime includes Apache spark but also adds a number of components and updates the substantially improvement of usability, performance and security of big data analytics.

5. Workload - Azure databricks can include two kinds of workloads like data engineering (job) and data analytics (all purpose) cluster type.

i) Data engineering - An automated workload runs on a job cluster which the Azure databricks job scheduler creates for each workload.

Ii) Data Analytics - An interactive workload runs on all-purpose cluster, Interactive workload typically can run commands within the Azure databricks notebook. However, for running a job on an existing all-purpose cluster is being treated as interactive workload.

6. Billing - DBU (Databricks Billing Unit) - Azure Databricks billing unit is the unit of data processing capability per hour based on VM instance type.

7. **Authentication and authorization** - Azure databricks uses the following resources for authentication and authorization.

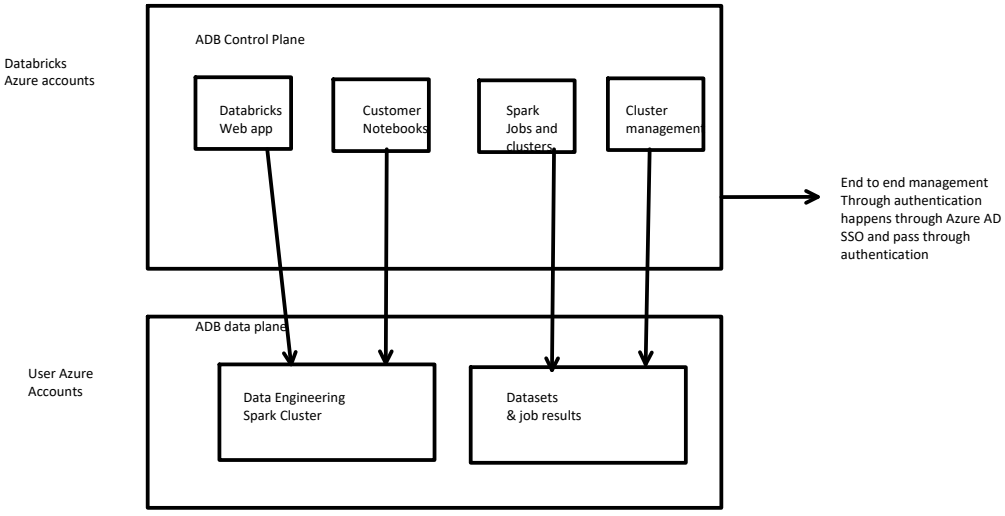
- **User** - A individual user who has access to the system and user identities are represented by the email address.
- **Service principal** - managed Azure AD based identity for use with jobs, automated tools and systems like scripts , apps and CI/CD pipeline integration. Service principals are represented with clientID, clientSecret, tenantID and subscriptionID.
-
-
- Azure AD Group - An Azure AD group is the collection of identities. Groups can simplify the identity management, assign access to the workspace, data and other scalable objects. All databricks identities can be assigned as members of groups.

- **Access Control List (ACL)** - a list of permissions attached to the workspace, cluster, job, table or experiment. An ACL can specify which users or system processes are granted access to the objects.

- **Personal Access Token (PAT) token** - A opaque string in Azure Databricks, is used to authenticate to the REST API and by tools in the Databricks integration to connect to SQL data warehouses.

Azure Databricks Architecture

Azure Databricks consists of Control Plane and Data Plane.



Features of Azure Databricks Cluster Components

a) Azure Databricks is structured to enable secure cross functional team collaboration while keeping a significant amount of backend services managed by Azure databricks.
b) Users can focus onto the data analysis, data engineering and data science tasks.

Azure Databricks Control Plane

1. The Databricks control plane includes the backend services which Azure databricks manages through into its own Azure account. The Jupyter notebook commands and many other workspace configurations are stored in the control plane and encrypted at rest.

2. The end user's / customers Azure account manages the ADB data plane and can never be able to access the control plane since it's managed by Databricks own Azure account. The customer's Azure account itself the data is being stored.

3. We can use the Azure Databricks connectors to connect clusters to external data sources outside of Azure account to ingest the data, or for storage. We can also ingest the data from external streaming data sources such as events data, streaming data and IoT data.

Azure Databricks Data Plane

1. The data is stored at rest in the Azure account of the data plane using own data sources. The data for the data analysis/engineering can never being stored in control plane. So that, the customers are responsible to maintain the control and ownership of the data.

2. The job results can reside within the Azure storage account.

3. The interactive/jupyter notebook results are stored in a combination of the control plane and the Azure storage account.

Azure Databricks Authentication and authorization components

1. **Azure AD User** - A unique individual who has access to the system. User identities are being represented by Azure AD groups, access policies and identities.

2. **Service Principa**l - A service identity for use with jobs, automated jobs/tools and systems such as scripts, apps and CI/CD pipelines. The Service principals are represented by the appId.

3. **Azure AD group** - A collection of identities in Azure AD. Groups can simplify identity management, making it easier assign access to workspaces, data and other scalable object. All Databricks identities can be assigned as members of groups.

4. **Access Control List(ACL)** - A list of permissions attached to the workspace, jobs, clusters, tables and experiment. An ACL specifies which users or system processes are granted access to the objects, as well as what operations are allowed on the assets.
   - Each entry in a typical ACL specifies a subject and an operation

5. **Personal Access Token (PAT)** - a opaque string is used to authenticate to REST API tools for integration with Databricks workspace to connect to Azure SQL DW / Synapse Analytics.

Data management Components in Azure Databricks

a) **Databricks File system (DBFS)** - A filesystem abstraction layer over the blob storage / ADLS gen2 storage. It contains directories, files (data files, libraries and images) and other directories. DBFS is automatically populated with some datasets which can be explored.

b) **Database** - A collection of information which can be organized so that it can be easily accessed, managed and updated.

c) **Table** - A representation of structured data, we can query the tables with Spark SQL and Spark APIs.

d) **Hive metastore** - This component which stores all the structures of information like various tables, and their partitions information in the data warehouse including columns, column type information, the serializers, the deseralizers to read and write the data. The corresponding files where the data is being stored. Each Azure Databricks deployment has a central hive metastore repo accessible by all spark clusters to persist over the table metadata.

SQLContext Class in PySpark SQL / Spark SQL

SQLContext is a class which's used for initializing the functionalities of Spark SQL. SparkContext class object (sc) is required for initializing SQLContext object.

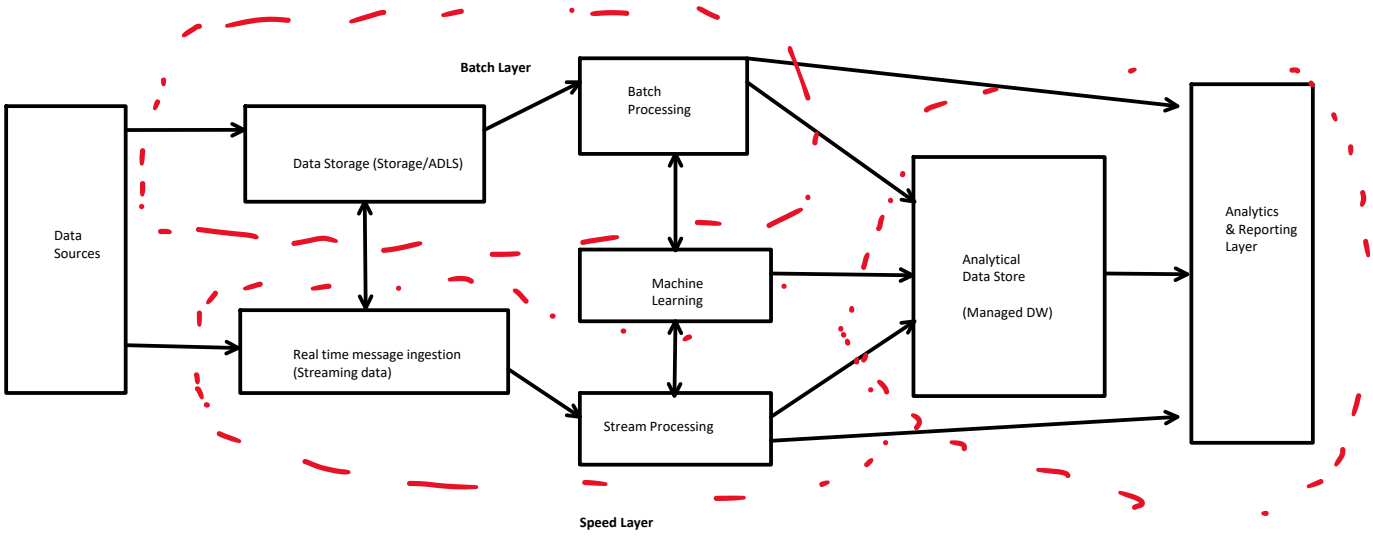
Data Sources

Spark SQL supports operating on variety of data sources through the dataframe interfaces. A dataframe can be operated on using relational transformation and also can be used to create a temporary view/table.

Registering a dataframe as temporary view/table allows to SQL queries on the data.

Caching / Performance Tuning for PySpark SQL

a) PySpark / Spark SQL can cache tables using an in-memory based columnar format by invoking spark.catalog.cacheTable("tableName") or dataframe.cache().
b) Spark SQL/ Pyspark SQL will scan only the required columns and automatically tune the compression to minimize the memory usage and garbage collection(GC) pressure.
c) We can invoke spark.catalog.uncacheTable("tableName") or dataframe.unpersist() to remove the table from memory.



1. Data Sources - Applications based data sources , static files, weblogs, MDM, ERP, CRM, IOT data, Sensors

2. Data Storage - Data for batch processing operations is typically stored in a distributed file store which can hold high volumes of large files in various formats. The kind of storage is called as data lake.

3. Batch Processing - Because the data sets are too large, often a big data solution requires data files for long running batch jobs to filter, aggregate and prepare for data analysis. ADB pyspark/spark SQL transformation, HDI Spark cluster for running scala/pyspark based data processing.

4. Real time message ingestion - solution can include real time data sources, incoming messages can be dropped into a folder for processing. As a solution, it requires message ingestion store to act as a buffer for message, support for scale out processing, reliable delivery and other message queuing semantics. This part of real time message ingestion is referred as stream buffering.

5. Stream processing - As a managed stream processing service where the processed data to be written to the sink, Azure Stream Analytics/ Databricks structured streaming based on perpetually running SQL queries which can operate on unbounded streams.

6. Analytical data store - The analytical data store can be used to serve these queries in rdbms, BI solution, data could be presented in a low latency noSQL db, Azure Synapse as managed DW solution provides large scale, cloud based DW.

7. Analysis, reporting and orchestration -

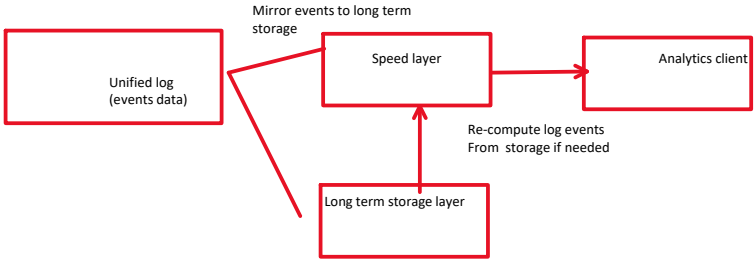
Lambda Architecture

1. Batch layer - (cold path) stores all of the incoming data in its raw form and performs batch processing on the data. The result of the processing is stored as batch view.

2. Speed layer (hot path) analyses data in real time. This layer is designed for low latency at the expense of accuracy.

The batch layer feeds into a serving layer which indexes the batch view for efficient querying. The speed layer updates the serving layer with incremental updates based on most recent data.

Kappa Architecture



Delta Lake in Azure Databricks

Delta Lake is the optimized storage layer which provides the foundation for storing data and tables in the Databricks Lakehouse platform. Delta lake is open source distributed platform which extends the parquet data files with a file based transaction log for ACID transactions and scalable metadata handling.

Delta lake is fully compatible with Apache Spark APIs, was developed for tight integration with Structured streaming, allowing is to use a single copy of data for both batch and streaming operations and providing incremental processing at scale.

Delta lake is the default storage format for all operations on Azure Databricks. Unless otherwise specified, all tables in Azure databricks are delta tables. Databricks originally developed Delta lake protocol and continues to actively contribute to Delta lake platform.

All tables on Azure Databricks are Delta tables by default. As using Spark Dataframes or SQL, Delta lake can support by saving the data to the lakehouse with default settings.

Updating and modifying Delta Lake tables

Atomic transactions with Delta lake provide many options for updating data and metadata. Databricks recommends to avoid interacting directly with data and transaction log files in Delta lake file directories to avoid corrupting the tables.

- Delta lake supports upserts using the merge operation
- Delta lake provides numerous options for selective overwrites based on filters and partitions
- We can manually or automatically update the table schema without rewriting data.
- Column mapping enables the columns to be renamed or deleted without rewriting data.
-

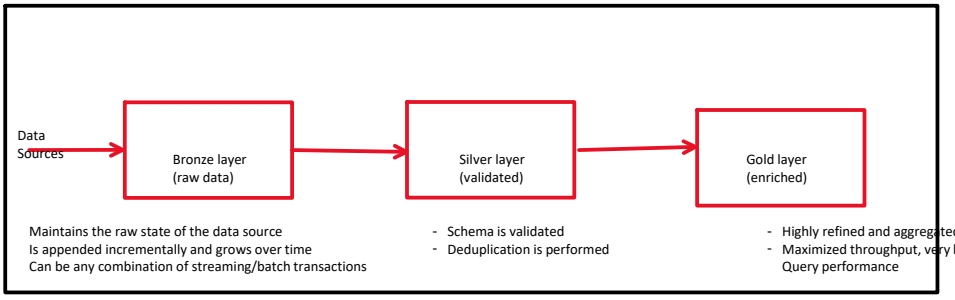
Incremental updates of Streaming workloads

- Since Delta lake is optimized for Structured streaming on Azure Databricks. Delta Live tables can extend native capabilities with simplified infra deployment, enhanced scaling and managed data dependency.
- Query over previous versions of table since each write to delta table creates a new table version. We can use the transaction log to review modifications of the table and query the prev version.

- Azure Databricks stores all data and metadata for Delta lake tables in ADLS storage, many configurations can be set either on table level or within the Spark session.

- Azure Databricks helps to process medallion architecture to process data in a series of tables as the data is cleaned and enriched. Delta Live tables simplifies ETL workloads through optimized execution and automated infra deployment and scaling.

Delta Lake of ADB



Access Control Enablement in ADB

1. In Azure Databricks, the ACLs can be used to configure permission to access data tables, clusters, pools, jobs and workspace objects like notebooks, experiments, and folders.

2. All admin users can manage the access control lists, as users who have been given delegated permission can manage the access control list

3. An Databricks cluster admin also manage the Azure Databricks REST API by giving or denying users the ability to generate access token,

4. An Azure administrator with the proper permissions can configure Azure AD conditional access to control where and when users are permitted to sign in to Azure Databricks and enable Azure Data lake storage credential pass through authentication. Which allows users to authenticate to ADLS gen2 from Azure Databricks clusters using the same Azure AD identity as it's being used to login to Azure Databricks.

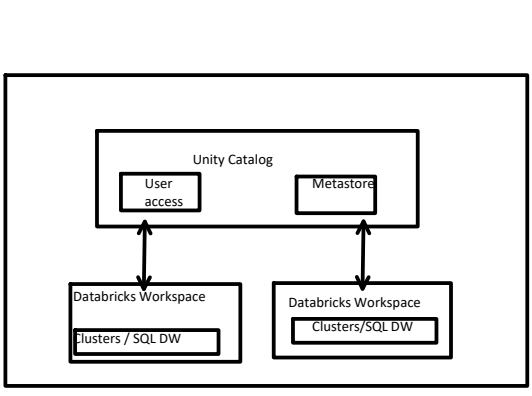
- All admin users can manage the access control list and also users who have been given delegated permission to manage the access control list.
- Workspace object access control
- Cluster access control
- Pool access control
- Jobs access control
- Dashboard access control
- Databricks SQL data access control
- Query access control

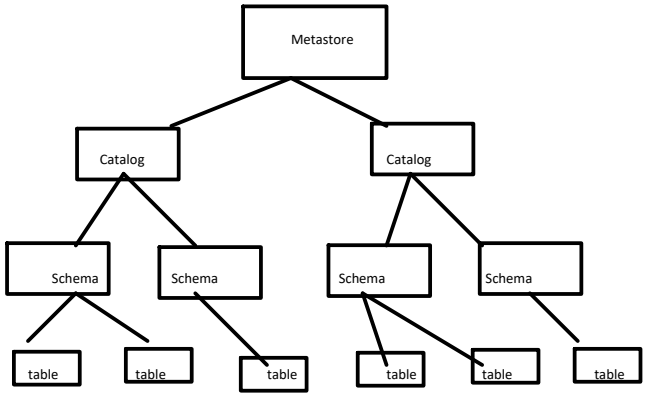
Workspace object Access Control	By default, all users can create and modify the workspace objects - including folders, notebooks, experiments and models - unless an administrator enables workspace access control. With workspace object access control, individual permissions determine a user's ability. 1. All users have **Can Manage** permission for items in the Databricks Workspace -> Shared folder. 2. We can grant **Can Manage** permission to notebooks and folders by moving them to the shared folder. 3. All users have **Can Manage** permission for objects the user creates. 4. With workspace object access control enabled - a) On the workspace level, only administrators can create new items in the Workspace folder.

	b) Existing items in the workspace folder - **Can Manage** c) User home directory - The user has **Can Manage** permission
Cluster Level Access Control	By default, all users can create and modify clusters unless an administrator enables cluster access control. With cluster access control, permissions determine a user's ability. 1. The allow unrestricted cluster creation entitlement controls the ability to create clusters. 2. Cluster level permissions control the ability to use and modify a specific cluster 3. When the cluster access control is enabled - a) An administrator can configure whether a user can create cluster b) Any user with **Can Manage** permission for a cluster can configure whether a user can attach to, restart, resize and manage that cluster.
Pool based Access control	By default, all users can create and modify pools unless an administrator can enable pool access control. With pool access control, permissions determine a user's ability. 1. Pool permissions are No permission, **Can attach to** and **Can Manage**.
Jobs Access Control	Enabling the access control for jobs allows job owners to control who can view job results or manage runs of a job. 1. There're five permission levels for the job - No permission - Can View - Can manage run - Is Owner - Can manage
Dashboard access control	There're four permissions for a dashboard - No permission - Can Run - Can edit - Can manage
Query Access Control	The default permission for query access control is to create the Databricks SQL query and manage the SQL queries.
Table access control	Lets us to programmatically grant and revoke access to data using ADB view based access control model. Table access control requires the premium tier. - SQL only table access control - Which restricts users to SQL commands. User's should be restricted to the Apache Spark SQL APIs, therefore cant use Python, scala, R, RDD APIs or clients who can directly read data from cloud storage such dbutils. - Python and SQL table access control - Which allows users to run SQL, Python and pyspark commands. Clients are restricted to the Spark SQL APIs, Dataframe API and hence cant use Scala, R, RDD APIs or clients who can directly read the data from cloud storage i.e. DBUtils. Before enablement of Python and SQL table access control, Azure Databricks admin must - Enable table level access control for the Azure Databricks workspace. - Deny users access to clusters which are not enabled for table access control. In practice, it means denying most users permissions to create clusters and denying users the **Can Attach To** permission for clusters which are not enabled for table access control.

Unity Catalog in ADB

In Unity catalog, admins and data engineers can manage users and access to the data centrally across all of the workspaces in Azure Databricks account. With Unity catalog, it provides centralized access control, auditing, lineage, data discovery capability across the Databricks workspace.





The hierarchy of primary data objects flows from metastore to table

- Metastore - The top level container for metadata. Each metastore exposes a three level namespace (catalog, schema, table) which organizes the data.

- Catalog - The first layer of the object hierarchy, used to organize the data assets.
- Schema - Also known as the databases, schemas are the second layer of the object hierarchy and contains tables and view
- Table - At the lowest level of the object hierarchy are tables and views.

1. Users in different workspace can share access to the same data, depending on the priviledges granted centrally in Unity catalog.

a) Define once and secure everywhere - Unity catalog in ADB offers a single place to administer the data access policies which apply across all workspaces and personas.

b) Standards - Compliant security model - Unity catalog security model is based on standard ANSI SQL allows administrators to grant permissions based on their existing data lake, at the level of catalogs, databases and schemas/views.

c) Built-in auditing - Unity catalog automatically can capture database, catalogs, tables and views. It can capture user level audit logs which record access to the data.

A table resides on the third layer of unity catalog. Rows and attributes of the data.

- Managed table - stored in the root storage location (":/dbfs") when we create the metastore. We can use the Delta table format.
- External table - External tables are the tables whose data can be stored after the deletion. Data is stored outside of the root storage location. We require direct access to the data using other tools

- Delta
- CSV
- JSON
- AVRO
- PARQUET
- ORC
- TEXT

Configure Unity Catalog in ADB

- Configure the storage container and Azure managed identity that unity catalog uses to store and access the data in the Azure account.
- Create a metastore for each region in which each org operates, attach the workspace to the metastore. Each workspace will have the same view of the data that can manage the unity catalog.
- If there's a new account. Users, groups, and service principals can be added to the databricks account.

Hands on Lab

Executing Production ready tools on Azure Databricks to develop, deploy the ETL workloads pipeline for data orchestration.

- Launching Spark cluster
- Create pyspark notebook and configure incremental data ingestion to Delta lake of ADB with **Autoloader**
- Execute the pyspark notebook to process, query and preview data
- Schedule the notebook as databricks job

Data Analytics

14 February 2023 09:32

Alteryx

14 February 2023 09:32