Pre-requisites

12 December 2022 19:53

Lab Setup Requirement	Hardware CPU - Intel Core i3/i5/i7 processor, RAM - at least 8 GB HDD- 512 GB / 1 TB, OS - Windows 10 /8.1/11, MS Word/excel, PowerBI desktop (optional)

Pre-requisites
Software - 1. SQL Server 2016, 2017 or 2019 Enterprise/Developer edition, 2. SQL Server Management Studio/Azure Data Studio, 3. Visual Studio 2019/2022/VS code, (IDE) 4. A Valid Azure Subscription, Azure CLI, Storage explorer, Microsoft Azure Subscription, Git tools and GitHub account, Azure PowerShell, PowerShell ISE, Note: Linux environment VMs (Apache hadoop/big data) (Linux OS) Tool: Putty.exe Azure based Linux servers, Vmware player/ virtual box AWS Tools for VS code, GCP Tools for VS code.

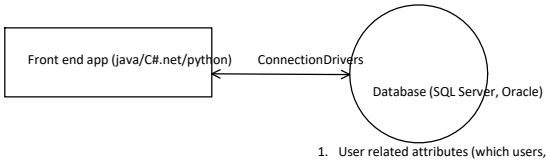
Azure Subscription (trial)
-- 12 months of free service + 30 days of 200 USD credit
-- enterprise subscription

Database Fundamentals and SQL Server BI

12 December 2022 19:54

Application metadata

MSSQL - 1433
MySQL - 3306



1. User related attributes (which users, port, MSSQL - 1433, encryption, protocol (TCP)
2. SQLConnection
3. ADOConnection (ADO.Net)

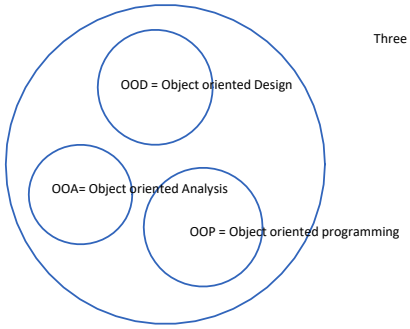
1960 (IBM) - developed the integrated Management system which is based on hierarchical database model.

1970 (IBM) - the relational database model was developed by E.F Codd.

1980 (IBM) - developed the Structured Query Language (SQL). It is declared as the standard language for the queries by ISO (International Standard Organization) and ANSI (American National Standards Institute).

OOPs principles

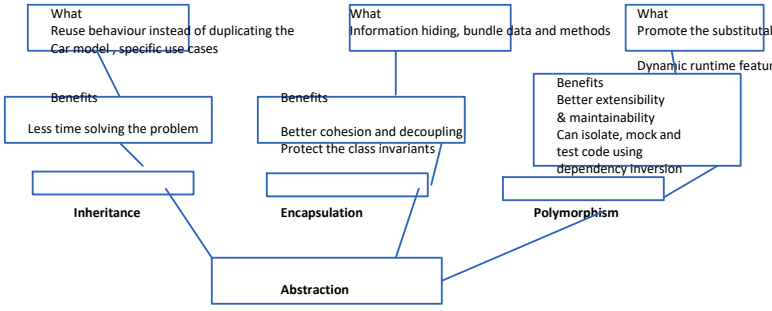
- Abstraction
- Encapsulation
- Inheritance
- Polymorphism



Three pillars of object design

Type Car = {
Types: 'small' | 'medium' | 'heavy'
Color: 'white' | 'black' | 'red'
Full_efficiency: 'efficient' | 'non-efficient'
Price: 'chaper' | 'moderate' | 'expensive'
}

Data Abstraction is the root principle of design.



1. Cohesion - represents a relationship within the module where a single class, has its well-defined purpose.

- It refers to what the class (a module) can do.

a) Low cohesion - low cohesion means a class which can perform a great variety of actions - broad, involves multiple operations.

Contracts - interface, abstract class, types

What
Data Modelling concepts using the contracts and concretions. (objects)

Benefits
Simply design API, simply data design, should
Separate how the data/ abstract can be designed
Declarative vs imperative

Staff Class
checkEmail()
sendEmail()
emailValidate()
PrintLetter()

b) High cohesion - means the class is to be focused on what it should be implementing. It includes only the methods related to the intension of the class.

Dependency Inversion - forms the pillar of object oriented programming to define as couple the software modules loosely so that one form of objects/ class while changing, doesn't affect others.

Staff Class
salary
emailaddr
setSalary (newsalary)
getSalary()
setEmailAddr(newEmail)
getEmailAddr()

Polymorphism - Compile time / Static polymorphism (method overloading, operator overloading)
Runtime / dynamic polymorphism (function overriding)

2. Coupling - refers to the principle of how related or dependent two classes / modules are toward each other. For low coupled classes, changing something major in one class should not affect other.

Coupling definition

In OOD, the coupling refers to the degree of the direct knowledge that one element/ object has of another element/object. How often the changes in class A forces the changes in class B.

1. Tight coupling - means when two classes often change together, when class A cant be changed directly because it's going to make changes in class B.

```
Class Subject {
Topic t = new Topic();        // subject class is tightly coupled the topic class
Public void letsRead()
{
 t.understand();
}
}


Class Topic {
 public void undertand()
{
 System.out.println("Tight coupling scenario");
}

}
```

2. Loose coupling - when two classes are just aware of each other , like class A and class B. Also, class B is expose through its interface, then class A and class B are loosely coupled.

e.g. Microservices application

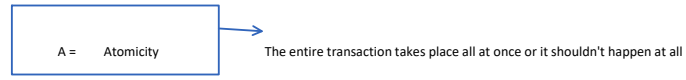
```
Public interface Topic
{
 void understand();
}

Class Topic1 implements Topic {
Public void undertand()
{
 System.out.println("This is loose coupling example");
}
}
Class Topic2  implements Topic {
Public void understand()
{
System.out.println("This is another loose coupling class");
}
}
Public class Subject{
Public static void main(String[] a)
{
 Topic a  = new Topic1();
t.understand();
}
}
```

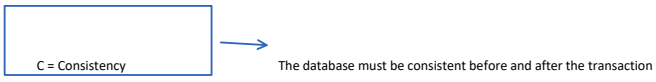
e.g. microservices application design

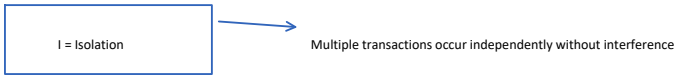
Empaddr and emp classes are separate classes where change of a simple address of a emp entity / object does not affect to the emp class and its related methods.

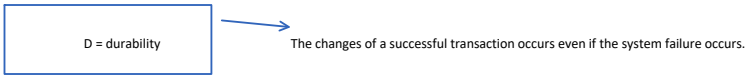
High coupling scenario - it would make difficult to change or maintain the code, since classes are closely knit together , making a change could require an entire system revamp.

Best practice - good software design should always has **high cohesion and low coupling**.









Different Types of Databases

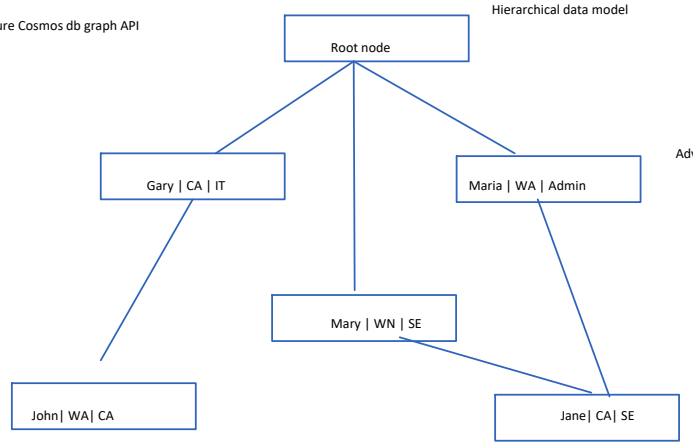
1. **Flat file database** -- kind of text database where each line of the plain text file holds only a single record. (e.g. MS Access, MS Excel)
2. **Hierarchical database** -- based on hierarchical data model, where the data is viewed as a collection of tables, data is designed into a tree like structure 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** - can consists of multiple parent segments and this segments can be grouped together as levels but there's always exists a logical association between the segments belonging to any level.
4. **Relational database** - consists of set of tables with columns and rows
5. **Object-oriented database** - information can be represented in the form of object-oriented programming. Inclined towards the objects e.g. multimedia records in a relational database can be definable data object.
   Mongo db is a object-oriented database.
6. **Distributed Database** - Consists of two or more files located in different sites/ location.

e.g. SQL Server mirror databases,

7. **NoSQL databases** - non-relational db has support for unstructured, semi-structured data and it can include dynamic schema, flexible data model for faster data retrieval
   e.g. Mongo db, Cassandra, Couch db, Azure Cosmos db
8. **Graph database** - node - entity (rows in the table), attributes (relationship/columns)

e.g. Neo4j, Azure Cosmos db graph API

Hierarchical data model

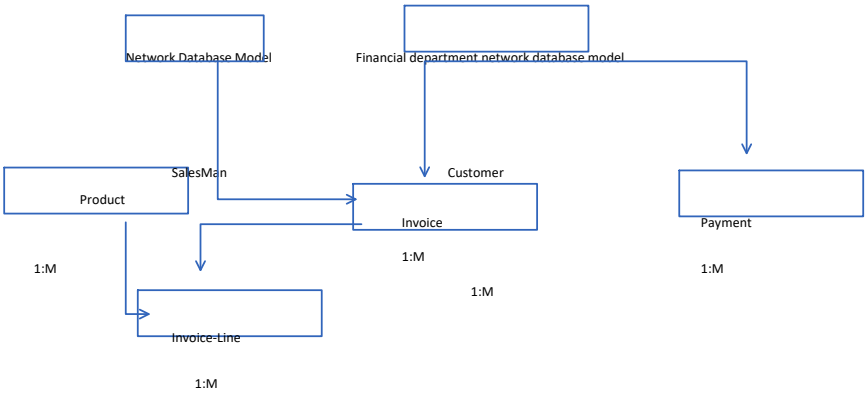


Advantages

- As the database is based on the hierarchical data models, the relationship between various layers are logically simple , it has a very simple hierarchical database structure.
- It has the data sharing facility since all data are held in a common database data and sharing of data becomes practically feasible
- It offers data security and integrity since it's based on parent-child relationship.

Disadvantages:

- Design is simple but implementation is complex
- This model also lacks flexibility as the changes to the new tables or segments often leads to very complex system management tasks.
- It has no standards as the implementation of the model does not provide any specific standard and limited to the relationship s which does not conform to 1:N format.



Advantages

- This model is simple and easy to design compared to hierarchical data model
- This model is capable of handling multiple types of data easily and we can access data easily, we can implement 1:1, 1:M, M:N relationships.

Disadvantages

- The schema of the network database mode is quite complex and records are maintained through pointers.
- The design of the network database mode is not user-friendly
- The model does not have any scope of automated query optimization

Data Dictionary or metadata in DBMS

A data dictionary is an integral part of a database. It holds the information about the database and the data which it stores named as metadata.

Characteristics of data dictionary

- A metadata is defined as the data about the data
- It's the self-describing in nature of databases
- It holds the information about each data elements in the database
- Such as names, types, ranges of values, access authorization, include the application details / program which uses the data
- Metadata is being used by programmers to develop the application, queries to manage and manipulate the data.

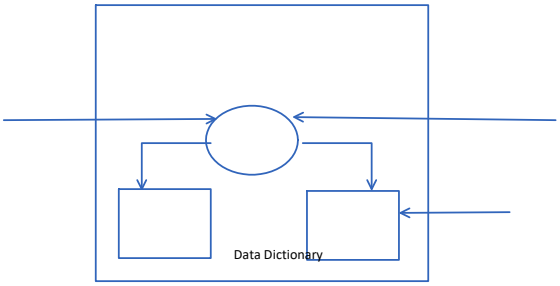
Types of Data Dictionary

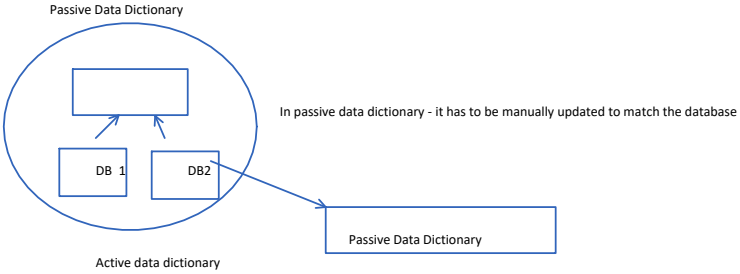
1. Active Data Dictionary -- self updating

a) managed automatically by the data management software
b) It's always consistent with the current structure of the database (e.g. RDBMS)

2. Passive Data Dictionary - mainly for documentation purpose
- Managed by user of the system and is modified manually by the user. (e.g. Dataedo by IBM IMS)

Active Data Dictionary





Data considered in DBMS for data Dictionary

Schema
- Tables

-Columns
- Constraints
- Foreign keys
- Indexes
- Sequences

Programs in SQL
- Views
- Stored Procedures
- User defined Functions (UDFs)
- Triggers

Storage
- Size of tables and indexes stored inside the db
- Number of rows in table

Benefits of data dictionary

1. Improves the data quality
2. Spot the data anomalies
3. Implement transparency and collaboration
4. Get access to the good data
5. Involve regulatory compliance
6. Enables fast and accurate data analysis

OLTP - Online Transaction Processing (SQL server database engine / MSSQL, mysql, oracle)
OLAP - Online Analytical Processing (SQL server datawarehouse)

Employee Table

Fields	Columns/Attributes Emp ID	Columns/Attributes Emp Name	Columns/Attributes Emp Hire date
Rows1 (records/ tuples)	1001	John	10/12/2021
Row 2	1002	Nash	11/12/2021

Foreign Key

Primary Key - Emp_id which uniquely defines the each record in the employee table & help to distinctly identify each employee.

Employee_Address table

Fields	Attributes (emp_id)	Emp_primary_address	Emp_secondary_address
Row1	1001		
Row2	1002		

Primary Key = emp_id

Primary Key definition

A primary key is a column or a set of columns in a table whose values uniquely identifies a row in the table. A relational database is designed to enforce the uniqueness of primary keys by allowing only one row with a given primary key value in a table.

Foreign Key definition

A foreign Key (FK) is a column or combination of columns which is used to establish and enforce a relationship between the data in two tables.

- A Foreign key is a column whose value corresponds to the values of the primary key in another table
- A foreign key is a column or a set of column in table whose values corresponds to the values of the primary key in another table.

- In order to add a row with a given foreign key value, there must exist a row in the related table with the same primary key value.

- Emp_id the foreign key column of employee_address table whose value corresponds to values of the primary key (emp_id) of employee table

Integrity Constraints in SQL

- Constraints are the set of rules which enforce on the data to be entered into the database table. Basically, constraints are used to restrict the type of data which can be inserted into a database table.

Feature of Integrity Constraints

- The integrity constraints are one of the protocols which should be followed by the table's data columns. The constraints are generally established to restrict multiple types of information which can be entered into a table to ensure the integrity of data.

- Achieving the complete configuration of the constraints ensures that the data in the db is accurate and reliable to be consumed in the application.
- We can apply the integrity constraints at the column level or table level.
- The table level integrity constraints are applied on the entire table
- While the column level integrity constraints apply to the only one column.

Constraints can be defined in two ways:

a) Column level: The constraints can be defined immediately after the column definition with the CREATE TABLE statement. So, these are called as the column-level constraints.
b) Table level: The constraints can be defined after all the columns specified, with the ALTER TABLE statement. This is referred as the table level constraints.

SQL Server has the support for six types of constraints

1. **Primary Key constraint -** The **primary key** is a set of one or more fields / columns of a table which helps to identify the records in the db table. It can not accept any null or duplicate values.

Features of Primary Key Constraint

- The primary key must have distinct values which means one of the columns of the table should have a unique value from the other.
- By default with the primary key, null values are not allowed in a primary key column of a table.
- Any table may have only a single primary key which can be made up of multiple fields.

Primary key constraint defined at the column level:

```
Create table table_name
(
Column1 datatype [CONSTRAINT constaints_name] PRIMARY KEY,
Column2 datatype
);
```

2. Unique Key Constraints - A unique key is a set of one or more columns/fields which uniquely identifies each row / record in a table.

Features of Unique Key Constraint

- A table can have more than one unique key unlike the primary key
- Unique key constraint can also accept null values for a column
- Unique constraints are also referenced by the foreign key of another table. It can be used when developer wants to enforce unique constraints on a column and a group of columns which is not a primary key.

Students table - 1

Roll_no	Student_Name	Batch	Phone_no	Citizen_ID	Country_of_origin
001	Alan	01	212-334-2234	AB01	US
002	Matt	02	202-440-2335	CD04	Canada
003	Hans	03	304-223-2337		

Roll_no is a primary key

Citizen_ID , Country_of_origin are the unique key for the Students table

Student table -2

Roll_no	Student_Name	Batch	Phone_no	Citizen_ID	Country_of_origin
001	Alan	01	212-334-2234	AB01	US
002	Matt	02	202-440-2335	CD04	US
003	Hans	03	304-223-2337		Holland

Primary Key - Roll_no
Unique Key - Citizen_ID

Difference between Primary Key and Unique Key

Features	Primary Key	Unique Key
Basic	Used to serve as a unique identifier for each row in a table	Uniquely identifies a row which is not a primary key
NULL value acceptance	Cannot accept any NULL values	Can accept NULL values
Number of Keys can be defined in the table	Only one primary Key in a table	More than one unique key
Auto-increment	A Primary Key has the support for auto-increment value.	A unique Key does not support the auto-increment value
Modification	We cannot change or delete values stored in primary key	We can change the unique key values

IDENTITY (2,1)
Seed - initialization value (2)
Increment - increment value (3, 4, 5)

Indexes in SQL Server

Definition - An index is a schema object. It's used by the db server to speed up the data retrieval or rows/records by using a pointer. It can reduce disk (I/O) by using a rapid path access method to locate data quickly.

Features of Indexes

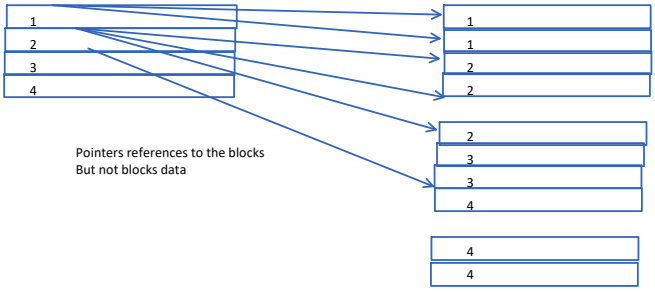
1. An Index can help to speed up the select queries and where clauses.
2. Indexes can be created or dropped with no effect on the data.
3. When an index is created, it includes a column contains a wide range of values.

Create index index_name on table_name column_name;

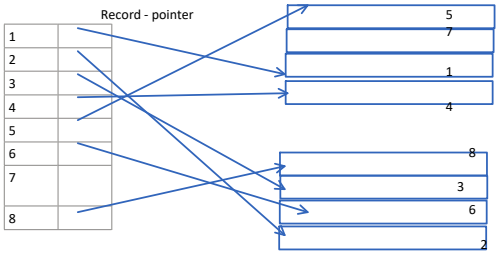
Table - table_name
Column - column_name

Create index index_name on table_name (col1, col2...)

a) Clustered Index - Clustered index is a kind of index which should satisfy the conditions
- The data or file, which moving to the secondary memory should be in sequential or sorted order,
- There should be a key value, means it cant have any repeated value.
- When applied clustered index on a table, it should perform sorting in that table only.
- We can create only one clustered index in a table like primary key
- Clustered index is as same as the dictionary where the data is arranged by alphabetical order.
- In clustered index, index contains pointer to block but not direct data gets blocked.
- We can have one clustered index on multiple columns and this kind of index is called composite clustered index.



b) Non-Clustered index - index contains the corresponding pointer to the data



Difference between Clustered and Non-clustered index

Clustered index	Non-Clustered Index
Faster	Is slower
Required less memory for operations	Required more memory for operations
In clustered index, index is the main data	Index is the copy of the data
A table can have only one clustered index	A table can have multiple non-clustered index
Clustered index store pointers to block not data	Non-clustered index store both the value and a pointer to actual row which holds the data
It has the ability to store data on disk	Non-clustered index does not have inherent ability to store data on disk
Primary Keys of the table by default is considered as clustered index	Composite key / used with Unique key of the table defines the non-clustered index
A clustered index is 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 logical order of the index doesn't match physical stored order of the rows on disk
Clustered index size is larger	Non-clustered index size is smaller.

Surrogate Keys

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

Features of Surrogate Key

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

Student_reg_A

Reg_no	Name	% obtained
210101	Harry	80

210102	Matt	70
210103	Chris	84
210104	Lee	85

Student_reg_B

Reg_no	Name	% obtained
CS101	Maria	70
CS102	Simon	60
CS203	Sam	90

Surr_no	Reg_no	Name	% obtained
1	210101	Harry	80
2	210102	Matt	70
3	210103	Chris	84
4	210104	Lee	85
5	CS101	Maria	70
6	CS102	Simon	60
7	CS203	Sam	90

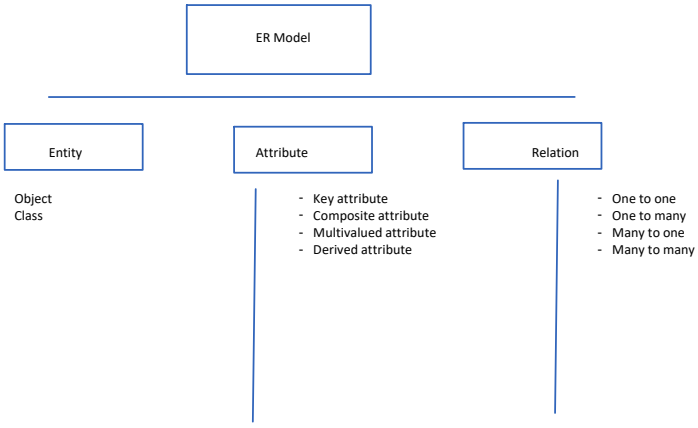
Item1
Item2
Item3
Item4
Item5
Item6
Item7
Item8

OFFSET 3 ROWS

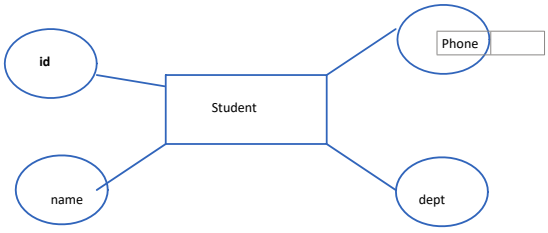
FETCH 5 ROWS

Benefits

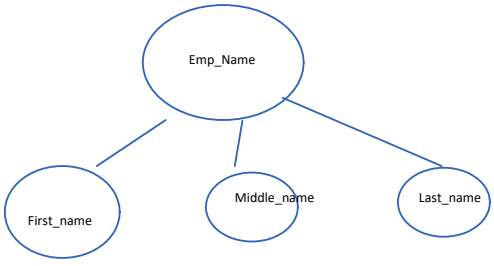
1. There's no direct information related with the table, so the changes are only based on the requirements of the application
2. Performance enhanced as the value of the key is relatively smaller
3. The key value is guaranteed to contain unique information
4. As it holds smaller constant values, the integration of the table easier
5. Enables to execute fast queries.



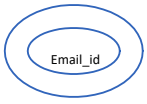
a) Key Attribute - key attribute is used to represent the main characteristics of an entity. It represents the primary key.



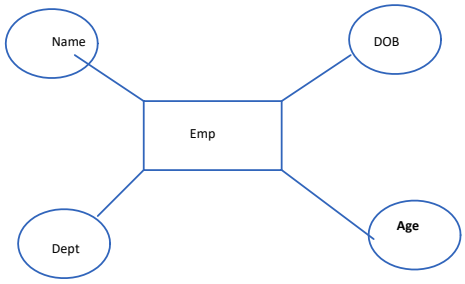
b) Composite attribute - composed of many other attributes altogether is called as composite attribute. The composite attribute is multiple simple attributes joined together.

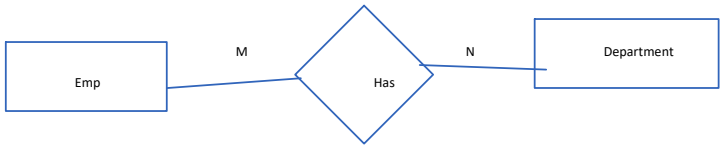


c) Multi-valued attribute - this attribute can have more than one value. These attributes are called as multivalued attributed.

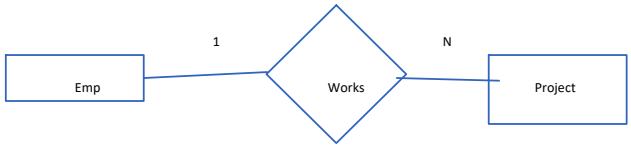


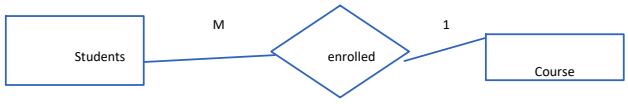
d) Derived attribute - An attribute can be derived from other attribute,





E1 : E2





Data types in SQL Server

Numeric data type in SQL server

Numeric	Storage space		
Tinyint	1 byte		
smallint	2 byte		
int	4 byte		
bigint	8 bytes		
Decimal(p,s) Numeric(p,s)	5-17 byte		
smallmoney	4 byte		
money	8 byte		
real	4 byte		
Float(n)	4-8 bytes		

P = precision (total number of digits can be stored both to the left and right of the decimal)
S = scale (max no of digits to the right of the decimal)

Decimal(8,3) allow the storage of total 8 digits and 3 of the digits to the right of the decimal or values.

Character data type

Char(n)	1 byte per character defined upto n
Varchar(n)	1 byte per character stored upto max 8000 bytes
text	1 byte per character upto 2 GB
Nchar(n)	2 bytes per character
Nvarchar(n)	2 bytes per character upto max 4000 bytes
ntext	2 bytes per character stored upto 2 gb

Nvarchar(max)

Date & time data type

datetime	0.00333 sec	8 bytes
Datetime2	100 nanosec	6-8 bytes
Date	1 day	3 bytes
time	00:00:00 to 23:59:59	3-5 bytes

Binary Data type

binary	Fixed with binary data	Upto 8000 bytes
Varbinary	Variable length binary data	Upto 8000 bytes

Column Property in SQL Server

IDENTITY property in SQL server

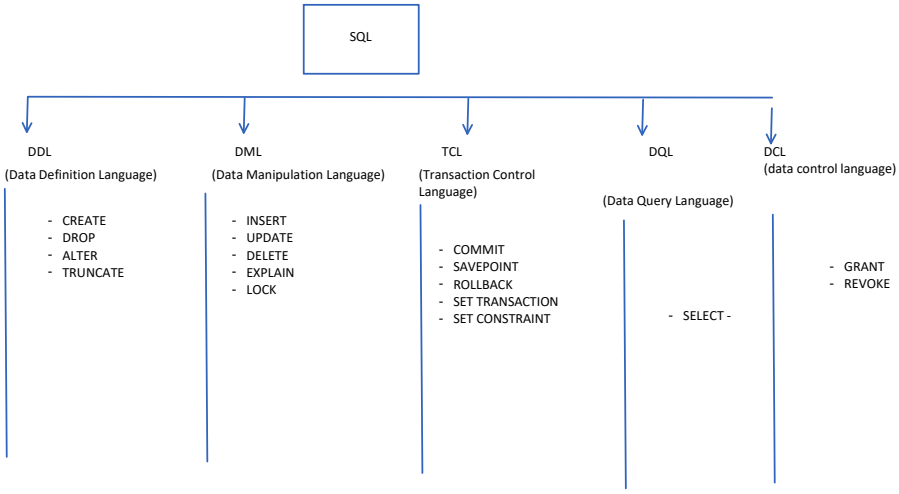
Identity property on columns can be defined to have a numeric data type. When the IDENTITY property is defined, SQL server manages the values in the columns on behalf of user.

IDENTITY consists of two parameters

- Seed - seed parameter defines the first number which will be assigned when the data is inserted into the table
- Increment - defines the number which will be added to the previous value for every subsequent row inserted into the table.

IDENTITY(1,2) - it'll start the value with 1 and will increment by 2 everytime when a new row is inserted into the table.

Nullability constraint in SQL server
Nullability is the most common property assigned to a column, whenever the column is defined as NOT NULL, we're required to assign a value to the column,
Whereas, when a column is defined as NULL, you don't have to assign a value to the column.



TCL

The transactions group a set of tasks into a single execution unit. Each transaction begins with a specific task and ends when all the tasks in the group successfully complete.

If any of the tasks fail, the transactions fail. Therefore, a transaction has only two results,

- Success
- Failure

MARS in SQL Server

The multiple active result sets is a feature with SQL server to allow the execution of multiple batches on a single connection. When MARS is enabled for use in SQL Server, each command object used adds a session to the connection.

- Applications can have multiple default result sets open and can interleave reading from them.
- MARS enables the execution of multiple db connection requests within a single connection. It allows batches to run. The database applications could not maintain multiple active statements in a connection.
- Applications can execute others statements (INSERT, UPDATE, DELETE, stored procedure calls) while the default result sets are open.

Benefit of schema

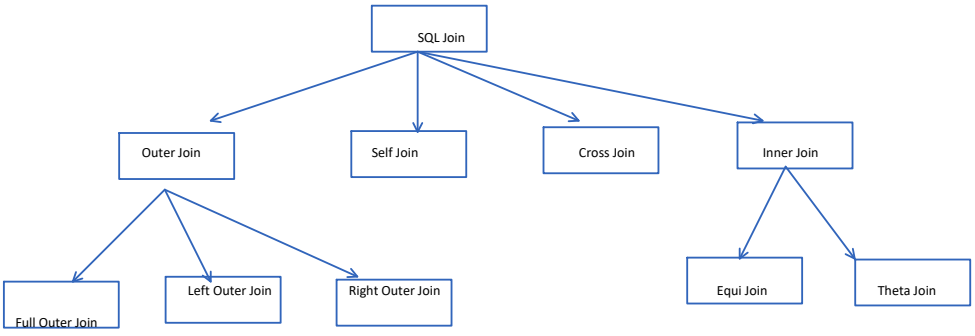
A database schema is considered the blueprint of a database object which describes how the data may relate to other tables or other data models.

The default schema for sql server is dbo.

But it's always recommended to create a customized schema to define how the different types of data (product, customer, customerAddress, employee) have been related to other tables / data models.

SQL Server Joins

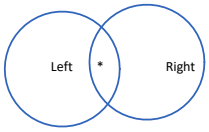
A SQL Join is a special form of generating a meaningful data by combining multiple tables relate to each other using a 'Key'. Typically, relational tables, must be designed with a unique column and this column is used to create relationships with one or more other tables. When you require a result-set that includes related rows from multiple tables, then the SQL joins are required on the column.



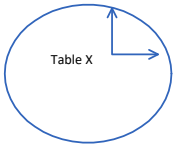
SQL Inner Join - The most simple and common form of join which is default join type.

- An Inner join combines two tables based on the criteria in the IN clause while also eliminating any rows from both tables which do not meet the criteria.

Cross - Join



Self join



Joining the table itself , is referred as self join

Normalization

- Benefits

1. Normalization is the process of organizing data in the database
2. It's used to eliminate undesirable characteristics from a set of tables. Used to eliminate the errors/ anomalies from insertion, update or delete.
3. Normalization divides the larger table into smaller tables linking them with relationships.
4. The normal form is used to reduce redundancy from the database table.

First Normal Form

- A relation will be 1NF if it contains an atomic value
- It can include a single value only in an attribute/field
- An attribute of a table cannot hold multiple values. It must hold single-valued attribute.

Second Normal Form

- All of the tables should be 1NF
- In the second normal form, all non-key attributes are fully functional dependent on the primary key

A B C - three variables

A -> B A is dependent on B
B -> C B is dependent on C

A -> C (A is dependent on C), (transitive dependency)

Third Normal Form

- A relation will be called in 3NF, if it is already in 2NF, and not contains transitive dependency
- 3NF is used to reduce the data duplication.
- It's also used to achieve data integrity
- If there's no transitive dependency exists for non-prime attributes for a table, then the relation/table must be in third normal form (3NF).

A relation is in third normal form if it holds at least one of the conditions for dependency X-> Y

1. X is super key
2. Y is prime attribute/ primary key, each element of Y is dependent on to some part of candidate/super key.

SQL Data Warehouse

12 December 2022 19:54

OLAP - Online Analytical Processing platform (Data Warehouse)

1. Datawarehouse definition
2. Concepts on Data Marts
3. What are the Data Lakes? Why we should design a Data Lake?
4. Examples of data warehouse
5. Examples of Data Lake & Data Mart
6. Data Warehouse Architecture
7. Tables design in Datawarehouse
8. Star schema design in SQL Server db through SSMS (SQL Server Management Studio)
9. - Fact table
   - Dimension table
   - Benefits of Data Warehouse
10. Define the concepts on Data Modelling
    - Star Schema (Demo/lab)
    - Snowflake Schema
    -
11. Definition of data integration
12. OLAP (Online Analytical Processing) , benefits, use cases
13. Difference between OLAP(SQL Data warehouse) and OLTP (SQL database) database
14. Data Mining concepts

Tools -

1. SQL Server Management Studio (SSMS)

A Data-warehouse is a process of collecting and managing data from varied sources to provide a meaning business insights.

e-Commerce use case

Customer sample dataset

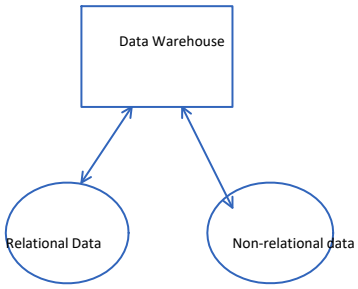
ID	Name	Location	Items Purchased	Time of purchase	IP address	Items kept in Cart
1	Matt	NJ	Laptop, Monitor	2022-9-6 17:00:00	120.34.23.5	Printer
2	Joe	NY	Headset	2021-12-01	192.168.1.1	Book

Problem Statement

a) Define which customers have made maximum purchases over the last three months
b) Define which customers have made minimum purchases (sales) over the last three months
c) Define which locations customers visited maximum to the ecommerce website?
d) Define how to make a recommendation of a product to customer which he/she can purchase next?
e) Define which of items has been kept in shopping cart of the customers but never made purchase?

- Sales Analytics
- Sentiment Analytics
- Weblog Analytics
- Recommendation Engine

Retail domain
Healthcare domain
Banking and Financial domain
Manufacturing domain



To ingest the relational / non-relational dataset into the SQL Data warehouse, the following analytical steps has to be performed.

Non-relational data - json, xml format
Real-time devices where data is extracted (IoT devices, weather sensors etc.)

ETL - Extract - Transform - Load model in Data Analytics

a) Data Extraction

Tasks-

- Data Cleaning (remove all duplicate data)
- Data parsing (sort the data as per the analytical requirement)
- Identify the structuredness/ formatting of the data (into proper format, in terms of rows/columns design)
- Impose the data into the tabular structure (table format with specific attributes and records)

b) Data Transformation

- Sql queries to start the analytical references (joins, functions, indexes, complex queries, views/stored procedures)

- Total no of customers who made the max purchase for products
- Total no of customers who made the min purchase of products
- What're the location(city/country) where customers visited max to the ecommerce website

c) Data Load

The processed data can be stored into various data warehouses for future analytical purposes. Load the data into the SQL datawarehouse. Design further the schema of the dataset with Star and snowflake schema.

<xml>

<node1>
<node2> Name </node2>
<node3> Address </node3>

A XML data for Customer

file: Customer.xml (Extensible Markup Language)

```
<Customer>
 <Name> Matt </Name>
<address> New York </address>
<phone> 202-122-1222 </phone>
</Customer>
```

Customer.json (javascript object notation)

```
Customer:
"name": "matt"
"address": "New York"
"phone": "212-122-1222"
```

file: Customer.json

Requirements of Data Warehouse

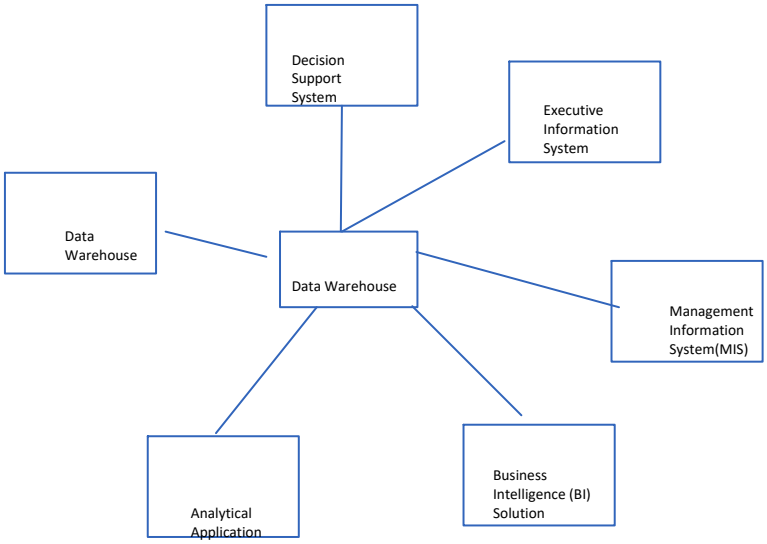
1. A Data warehouse (DW) is process of collecting and managing data from various sources to provide meaningful business insights.
2. A Data warehouse is typically used to connect and analyse business data from heterogenous sources. (data sources - traditional file system data, .csv, excel data, json/xml data, social media data, ERP/CRM dataset)
3. The data warehouse is the core of the Business Intelligence (BI) system which is built for data analysis and reporting.

Features of Data warehouse

- The data warehouse is maintained separately from the organization's operational database.
- We can call a data warehouse in some other names as well.

Examples of Data warehouse

- Informatica
- Vertica
- SQL Server Analysis Services (SSAS)
- Oracle Data warehouse
- Azure SQL Datawarehouse
- AWS Redshift
- GCP BigQuery



- How the data warehouse works

A data warehouse works as a central repository where the information arrives from one or more data sources. Data flows into the data warehouse from the transactional system and other relational databases.

Data may be

1. Structured
2. Semi-structured
3. Unstructured

- The data is processed, transformed and ingested so that users can access the processed data in the DW through the BI tools, SQL client tools and excel sheets.
- A DW also merges data coming from different sources into one comprehensive database.

- Types of Data Warehouse

1. Enterprise Data Warehouse (EDW)

Enterprise data warehouse (EDW) is a centralized warehouse. It provides decision support service across the enterprise. It offers a unified approach for organizing and representing data. It also provides the ability to classify the data according to the subject and gives access according to the division of the data.

e.g. SAP successfactor (employee organizational data)

2. Operational Data Store

Operational data store (ODS) is nothing but the data store required when neither the data warehouse nor the OLTP systems support organizations reporting requirements. In ODS, Data warehouse is refreshed in real time. It is widely preferred for routine activities like storing employee activities.

e.g. Salesforce CRM

3. Data Mart

A data mart is a subset of data warehouse. It specifically designed for a particular line of business such as Sales, Finance, Accounting, HR etc.

A data mart can collect data directly from various different data sources.

Limitation of DBMS

1. Data volume , there 's a limitation. In Azure SQL db, maximum volume/size of the db can be 5 TB.
2. Traditional DBMS cant fulfill the requirements of Big Data (>100 TB, 500 TB, 100 PB, 500 PB)
3. A data warehouse s separate from DBMS, it stores a huge amount of data which is typically collected from multiple heterogenous sources like files, DBMS etc.
4. The goal is to produce statistical results which can help in decision making

e.g. a college might see the exams results, student performance analysis results.

Need for Data warehouse

1. An ordinary RDBMS can store only in MB/GB amount of data and too specific purpose.
2. For storing in TB size, the data storage should be used Data warehouse,
3. A transactional database (e.g. RDBMS - SQL server db, mysql) doesn't offer to analytics.
4. To perform effective analytics, an organization needs to keep a centralized data warehouse to study its business by organizing , and using its historic data for taking strategies, decisions and analyse the trends.

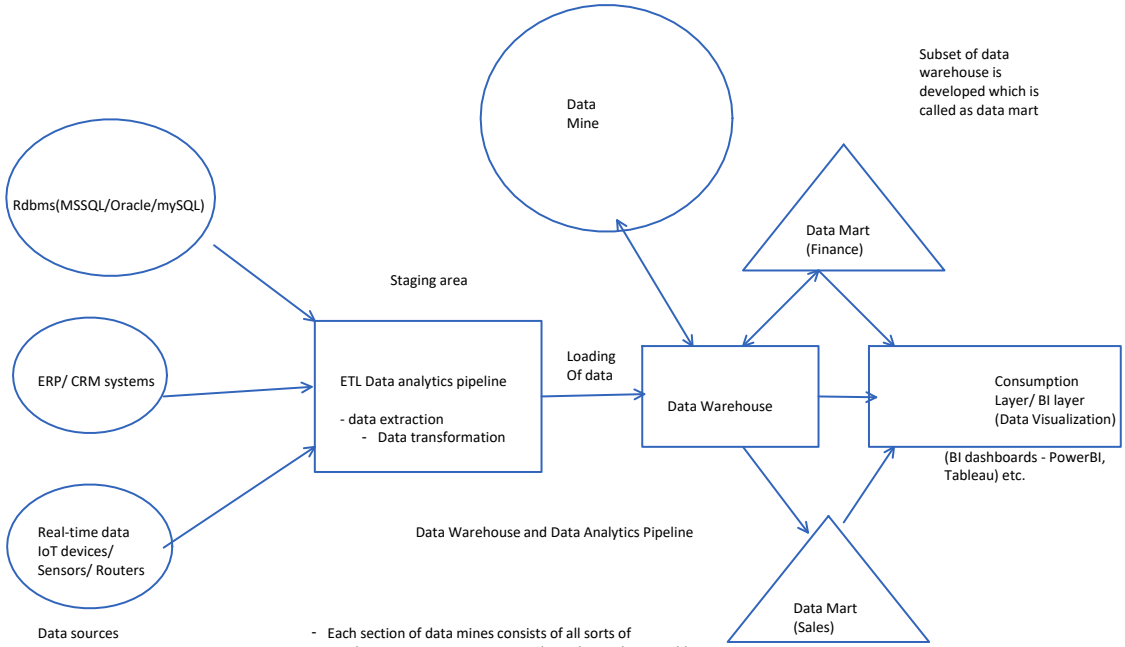
Difference Between RDBMS and Data Warehouse

Features	RDBMS (OLTP)	Data warehouse (OLAP)
Purpose	A common transactional database on operational or transactional processing. Each operation is an indivisible transaction.	A data warehouse is based on analytical processing of data. We can build data pipeline through ETL jobs. (Extract - Transform - Load)
analysis	The RDBMS system stores the current and upto-date data which is been used for daily operations.	A Data warehouse is integrated generally at the organization level by combining data from different databases. Example - A data warehouse integrates data from one or more databases. So that analysis can be done to get the results such as top sales of the month of an ecommerce website.
	A database is generally application specific. e.g. more data includes for app specific detail Customer database contains various tables customer, customerAddress, customerPurchase , customerOrders etc.	A data warehouse is integrated generally at the organization level , by combining data from various databases. e.g. A data warehouse integrates data from one or more multiple databases, so that the analysis can be implemented to get the results. "top product purchased by customer over last six months" in case of retail/ecommerce domain.
	Construction of the database is not so expensive.	Construction of data warehouse can be extensive.

Examples of Data warehousing

1. Social Media Websites - The social networking sites - Linkedin, Twitter are based on large data sets.
2. Banking - Credit card holders information, Spending/purchase pattern of customers, stored with the centralized DW of the banks.
3. Governments - data warehouse can be used to store and analyse tax payments which used to detect the tax liabilities.

e-commerce, Retail, healthcare, marketing.



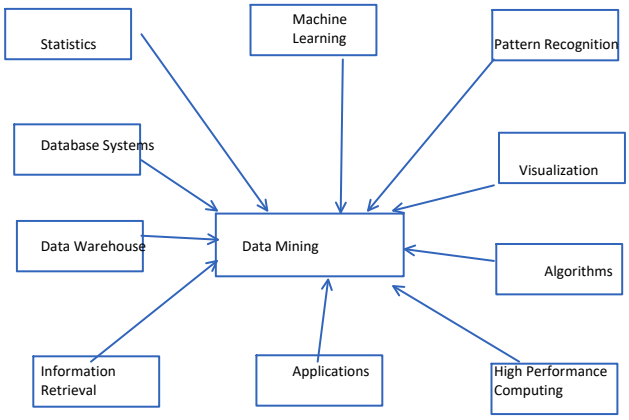
- Each section of data mines consists of all sorts of Employee, project, customer, products data, schema, tables, Attributes, relations etc.
- The data warehouse which is the centralized analytics data repository of an organization gets full set of data over extended period of time(> 10 yrs, > 20 yrs, > 30 yrs etc.) stored within the data mines.

Definition of Data Mining

The Data Mining refers to the knowledge mining from data, knowledge extraction, data / pattern analysis, data insights analysis. It's basically the process carried out for the extraction of useful information from the bulk of data or data warehouses.

The data mining is the result of extraction of the data patterns and knowledge after the data is processed through ETL jobs.

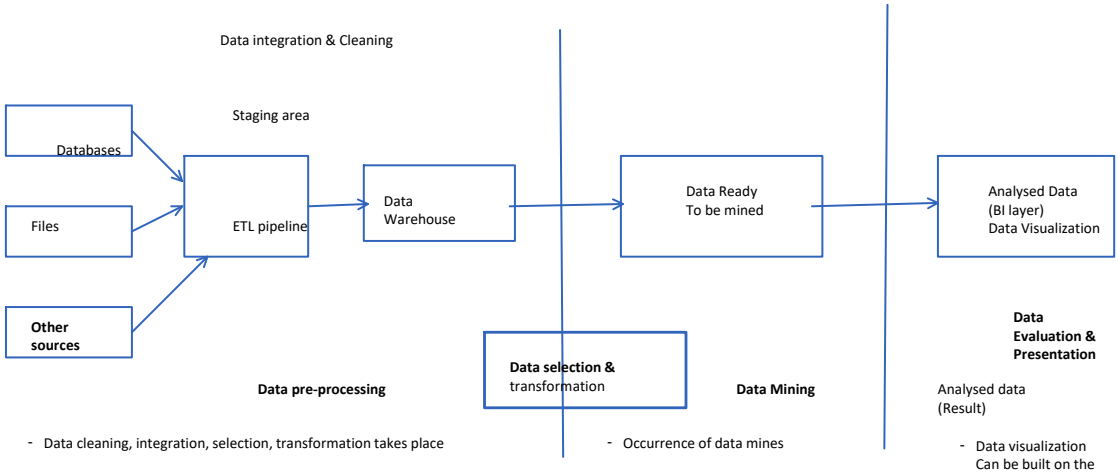
Technical data mining is the computation process of analysing data from different perspectives, dimensions, angles, categorizing/ summarizing it into meaningful information.



Data Mining can be applied to any type of data

- Data warehouses
- Transactional databases
- Relational databases
- Multimedia databases
- WWW (web apps)

Data mining is a while process involving into data pre-processing, data mining and data evaluation and presentation.

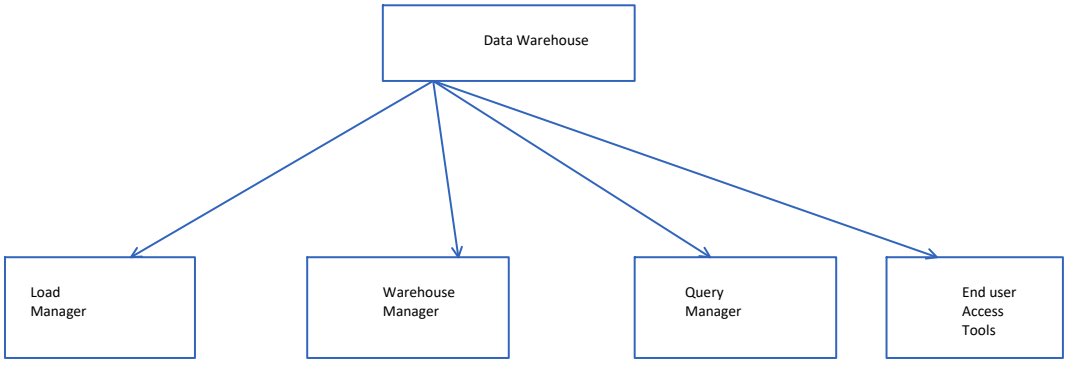


- Data cleaning, integration, selection, transformation takes place
- Occurrence of data mines
- Data visualization Can be built on the analysed data

Apps for data mining

- Financial Analysis
- Biological analysis
- Scientific analysis
- Fraud detection analysis
- Research analysis

- Components of Data Warehouse



- Load manager is called the front-end Component.
- It performs operations associated With the management of data into the data warehouse
- Query Manager is the backend
- Kinds of tools can be used

- Load manager is called the front-end Component.
- It performs all of the operations associated with the extraction and loading of the data Into the data warehouse.

- It performs operations associated With the management of data into the data warehouse
- It performs the analysis of the data to ensure consistency , creation

Of index, views,

- Generation of normalization, aggregation, transformation & merging

Of all source and backing up data.

- Query Manager is the backend

Component of SQL DW.

- Performs all the operations related to the management

Of user queries.

- Operations of the DW components are direct

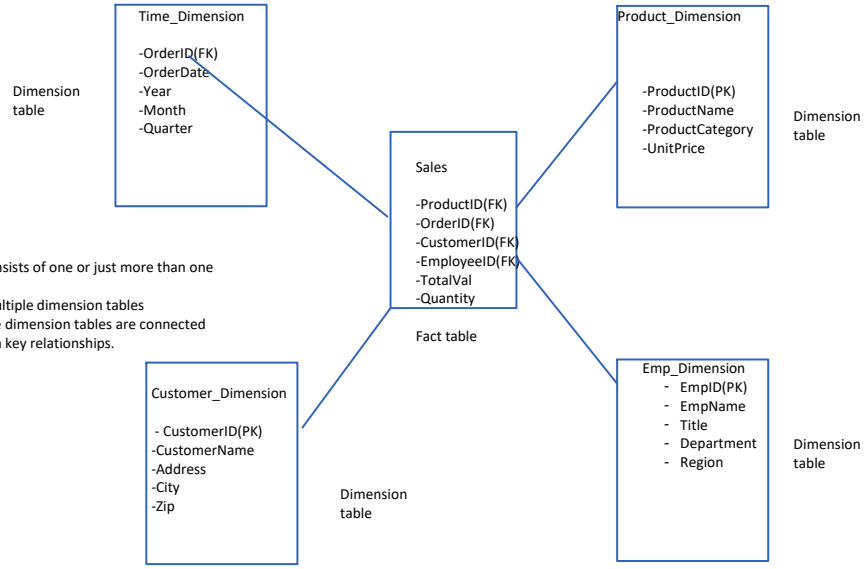
Queries to the appropriate tables for scheduling of the query execution.

- Kinds of tools can be used

a) Data Reporting tools (SSRS, Excel, PowerBI, Tableau etc.)
b) Query tools (SQL client tools)
c) App development tools (Visual studio)
d) OLAP tools / data mining tools (DB2, informatica)

Star Schema in Data Warehouse modelling

Star schema is the fundamental schema among the data mart and warehouse. It's simplest.
This schema is widely used to develop and build a data warehouse and dimensional data marts.
It includes one or more fact tables indexing any number of dimension tables.



- Star schema consists of one or just more than one Fact table
- It consists of multiple dimension tables
- The fact and the dimension tables are connected

on the basis of foreign key relationships.

Features of Star Schema on data modelling

- In star schema, the business process data, which holds the quantitative data about the a business which is distributed in fact and dimension tables.
- The fact table consists of more number of columns / attributes
- The dimension are smaller in size compared to fact tables, contains more number of rows/records.

Advantages of Star schema

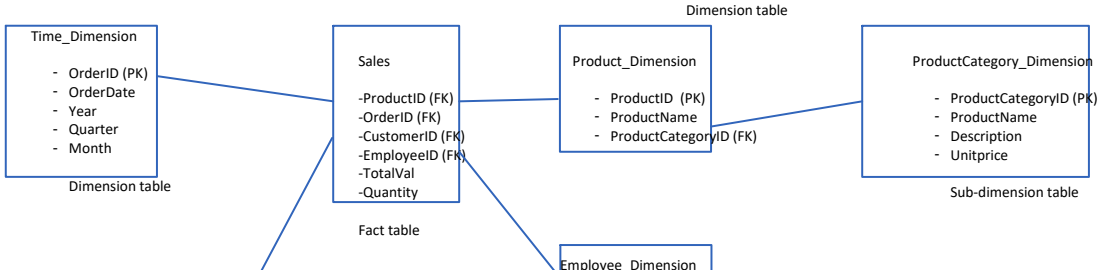
1. Simpler Queries - Join logic of star schema is quite simple and in comparison other SQL joining logic can be seen to fetch data from a transactional schema. The tables are highly normalized.
2. Simplified Business Reporting logic - In comparison to a transactional schema, which is highly normalized, the star schema makes simpler common business reporting logic like as reporting and ad-hoc analysis on data warehouse.
3. Build OLAP systems - Star schema is widely used in data warehouses (OLAP systems). In fact, the Star schema can help to deliver the major Relational OLAP model which can use star schema as a source of data.

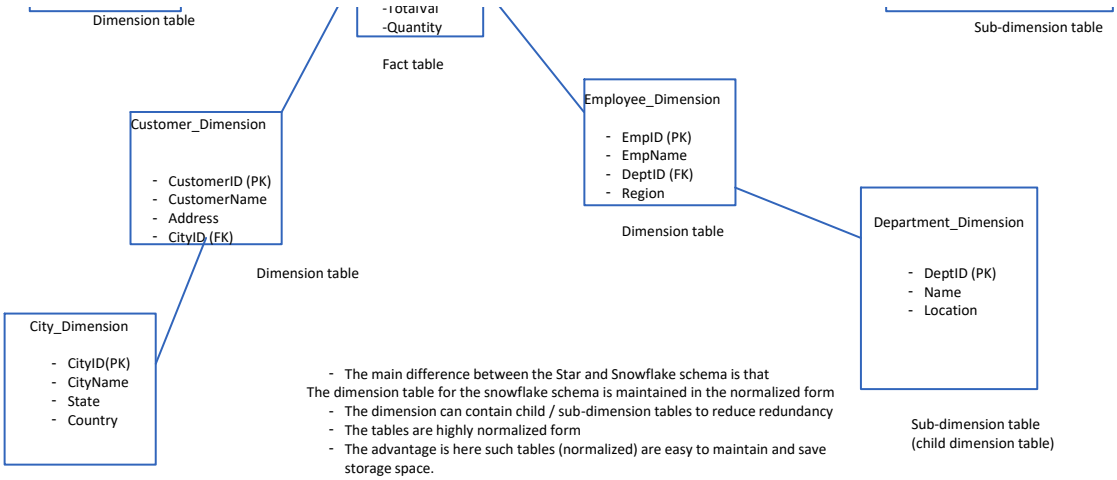
Snowflake Schema

The snowflake schema is a variant of the star schema. The centralized fact table is connected to multiple dimensions. In the snowflake schema, dimensions are presented in a normalized form in multiple tables.

The snowflake structure is materialized when the dimensions of a star schema are detailed and highly structured, also having several levels of relationship, the child tables have multiple parent tables.

The snowflake schema effect affects only the dimension tables and does not affect the fact tables.





- The main difference between the Star and Snowflake schema is that The dimension table for the snowflake schema is maintained in the normalized form
- The dimension can contain child / sub-dimension tables to reduce redundancy
- The tables are highly normalized form
- The advantage is here such tables (normalized) are easy to maintain and save storage space.
- It also means more joins will be required to execute the query.

Sub-dimension table
(child dimension table)

Sub-dimension table
(Child dimension table)

Characteristics of Snowflake schema

- The tables are highly normalized compared to star schema, they take less disk storage space .
- It's easy to implement dimension which is added to the schema
- They are multiple dimension tables, so performance is sometimes reduced, querying is bit more complex.
- Structured data which reduces the complexity around the problem of data integrity
- It uses small disk space because the data is highly structured.

Difference between Star and Snowflake schema

Star Schema	Snowflake schema
In Star schema, the fact and the dimension tables are contained.	While in snowflake schema, The fact tables, the dimension tables as the sub-dimension tables are contained.
Star schema is top-down approach	While it's a bottom-up approach
Star schema uses more space	While, it uses less space because all of the dimension tables are normalized upto one or more sub-dimension tables.
It takes less time for the execution of queries.	While for snowflake schema, it takes more time for the execution of queries.
In star schema, normalization is not used	In snowflake schema, both normalization and denormalization is used
It's quite simple to design	While, design is complex compared to star schema
The query execution & complexity for star schema is quite low.	Query execution and complexity of snowflake schema is higher than star schema
It has less number of foreign keys because of less number of dimension tables	While, snowflake schema has more of foreign keys because of higher number of dimension tables
Star schema has high data redundancy	While, it has less data redundancy because all of the dimension tables are normalized to sub-dimension tables.

Difference between Fact and Dimension table

Fact table	Dimension table
Fact table contains the measuring of the attributes Of a dimension table.	Dimension table contains the attributes on that truth table which calculates the metric.
In fact table, there's less attributes than dimension table. It means fact table contains more number of Rows/ records.	While in dimension table, there's more attributes than the fact table. More number of columns are available in the dimension tables.
In fact table, there' more records than dimension Table,	There's less records, more columns than fact table
Fact table forms a vertical table	While, dimension table forms a horizontal table.
The attribute format of fact table is in numerical format and text format.	While, the number of dimension is more than fact table in a schema.
It's used mainly for analysis purpose and decision Making systems	While the main task of the dimension table is to store the information about a business and its process.

Hands-on Lab

Create Star Schema with SQL Server Management Studio

Tasks

1. **Create a dedicated db for Star Schema**
2. **Create Database diagram support for the db**
3. **Create new database diagram**
4. **Create the data types for Fact and dimension tables (Under Database -> Programmability -> Types -> User-Defined-data-type)**
   1. **dbo.udt_surrogate_key (int, not null)**
   2. **dbo.udt_business_key (nvarchar(20), not null)**
   3. **dbo.udt_dollar_amount (money, not null)**
   4. **dbo.udt_quantity(int, not null)**

5. **Create Dimension tables**
6. **Create Fact tables**
7. **Create Build out relationships between Fact and dimension tables.**

1. Click on empty space & select "Select All"
2. Right click on any table & click "Table view" & choose the option "Name only"
3. Right click on the any of the table and click on option "Authorize selected tables"
4. Right Click on the empty space & click on "arrange tables"

Dimension tables

1.dim_date primary_key- (date_key)
2. dim_individual_customer PK - (individual_customer_key)
3. dim_territory PK - (territory_key)
4. dim_store PK - (store_key)
5. dim_employee PK - (employee_key)

Fact table (fact_sales_order)

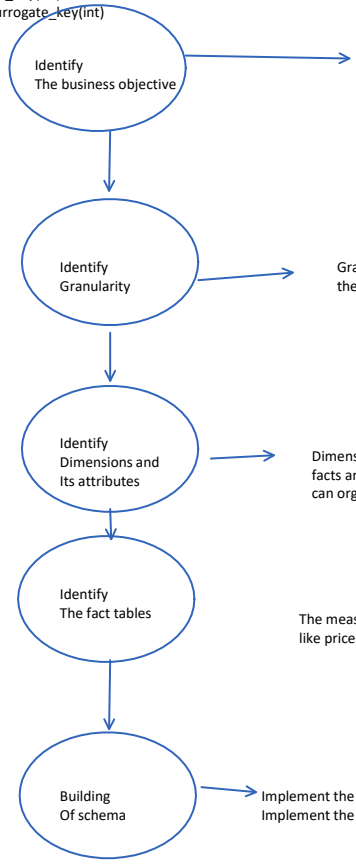
date_key udt_surrogate_key(int)
territory_key udt_surrogate_key(int)
employee_key udt_surrogate_key(int)
store_key udt_surrogate_key(int)
individual_customer_key udt_surrogate_key(int)

Dimension Data Modelling

Dimension data modelling is comprised of fact and dimension tables. The main objective of the dimension data modelling is to improve the data retrieval so it is optimized for SELECT operation.

Advantage - we can store the data in such a way so it's becomes easier to retrieve the data once stored in the data warehouse.
Dimensional model is the data model used by many OLAP systems.

Steps for creating the dimension data model



Sales, marketing, HR are some of the examples as per the requirement
Of the org. Since the most important step of data modelling is the selection of business
Objective which also depends on the quality of data available for that process

Granularity refers to the lowest level of information stored in the table. The level of detail for the business problem and its solution is described by the context.

Dimensions are the objects or things. Dimensions can categorize and describe the data warehouse facts and measures in a way which supports answers to the business problems. A data warehouse can organize the descriptive attributes as columns in dimension tables.

The measurable data is stored within the fact tables. Most of the fact table rows are numerical values like price, cost per unit etc.

Implement the relationship between the fact and dimension table through foreign key constraints.
Implement the dimensional model in this step.

Measures

Measures in Data warehouse are the set of aggregates which we can calculate, such as sum of total orders placed by the customer.

Measures example

Qualitative - productID
Quantative - like the price of the product

The source of all OLAP structures is a table within a data source. OLAP has two terms to refer to the

source tables.

- Fact table - is used to define the measures
- Dimension table - stores the data used to define dimensions

In data warehouse, the dimensions are pieces of data which allows us to understand and index measures in the data models. Dimensions are either characteristics of a measure or pieces of data which helps to contextualize the fact.

- Dimensions example
-

a) Product which was sold online
b) Product color
c) Product name
d) Name of the customer purchased the product
e) Name of the employee sold the product
f) Store location
g) Warehouse location

- **Dimensions**
- **Attributes**
- **Measures**
- **Hierarchies**

1. Dimensions define the basic business attributes like we want to analyse. It refers to customers, products, and time.

2. The attributes within a dimension define the columns within a dimension which are used for analysis like CustomerName, ProductName, City, PortalCode and OrderDate.

3. Measures are the set of aggregates which we can calculate such as sum of total orders, number of orders.

4. Hierarchies allows us to define a navigation structure within a dimension such as
   - Customers within cities within states within countries
   -Calendar months within the calendar quarter within the calendar years.
- Fiscal months within the fiscal quarter within fiscal year

Cube - A cube contains of the analysis objects you define with the highest level of security that can be assigned. Within SQL Server Analysis Services, we can multiple cubes for the users within the SQL Server analysis services database.

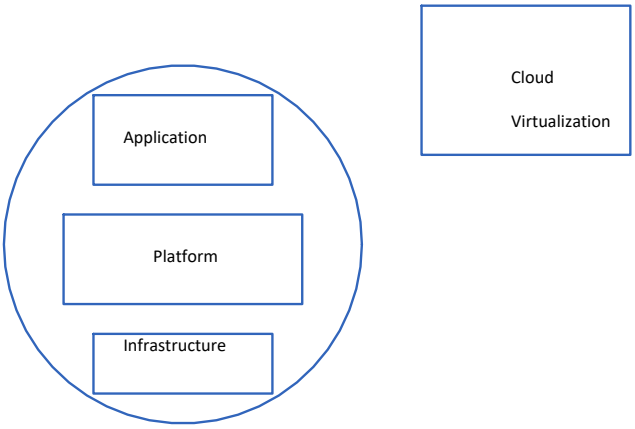
Cloud Fundamentals

12 December 2022 19:54

1. Cost efficiency
2. Security
3. Flexibility
4. Mobility
5. Insights
6. Increased collaboration
7. Quality Control
8. Disaster Recovery (infrastructure, platform, application, database, network)
9. Loss Prevention (any data deleted should be able to recover it)
10. Automated Software Updates
11. Sustainability

By definition, the Cloud computing refers to the internet based computing. Cloud computing refers to the components and sub-components required

- Front-end (thin client (web apps), thick client (desktop apps))
- Backend (servers, storage)
- Cloud based network (internet, intranet & intercloud)



Benefits of Cloud Computing

1. Cloud is easy to access for everyone. All the resources servers, disk, network, IP address, storage, databases, data warehouses etc. all accessible to everyone globally through internet.
2. Developing new applications, services, storage, databases are easier to build onto cloud.
3. It's useful to migrate the existing applications, databases, storage to the cloud as per supportability of the cloud vendor.
4. Software on demand.
5. Cost effectiveness - **Pay as you go**. (pay by hour)
6. Analysis of data
7. Streaming of high quality OTT resources (audio/video streaming)

Cloud Migration - Capex (capital expenditure) -> Opex (operational expenditure)
(on -prem env) (cloud env)

Features of Cloud

a) Scalability - increase the number of instances or sizes of server, storage or databases on cloud.
   1. - **horizontal scalability (add more server instances)**
      1.1 Scale out - add more server instances
      1.2 Scale in - reduce the number of server instances
   2.Vertical scalability (add more compute capability - more CPU core, memory, disk storage etc.)
      1.1 scale up - add more compute capacity to the server instances
      1.2 scale down - reduce the compute capacity to the server instances

As per the Cloud design principles, **horizontal scalability (scale out) is recommended to go for in business.**

b) **SLA - Service Level Agreements -** 99.99% of the availability will be there all the time of the Azure SQL db instance.
   - there will be only 0.001sec of downtime per sec/week/month basis will be there.

- Associated with each and every cloud provider (Amazon web services, Microsoft Azure, Google Cloud platform, IBM cloud, Oracle Cloud)
- Cloud provider can compensate for any downtime for any missed SLA / guaranteed uptime.
- This is the part of cloud based business model for the cloud provider to their customer

Different types of Cloud

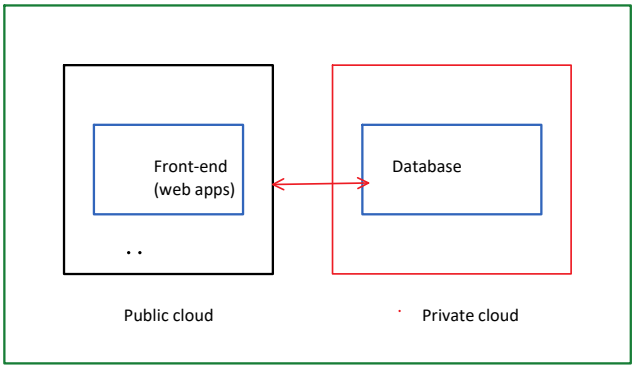
Public Cloud
Private Cloud
Hybrid Cloud

Cloud Models

IaaS (Infrastructure as Service)
PaaS (Platform as Service)
SaaS (Software as Service)

Different types of Cloud Computing Platform

	Public Cloud	Private Cloud	Hybrid Cloud
Definitions	Public cloud is open to all to store and access information via the internet, using pay-as-you-go model. In Public cloud, computing resources are managed and operated by the Cloud Service provider (i.e. Microsoft Azure, Amazon Web Services, Google Cloud platform etc.)	Private cloud is known the organization's internal cloud. The applications deployed over the private cloud are accessible only within the private network.	It's the combination of both public and private cloud. Hybrid cloud = public cloud + private cloud
Features	Public cloud is secure with the cloud controls (policy, Certificates, keys/passwords)	Private cloud is the most secure cloud platform since the applications are being deployed into the organization's own server and datacenters.	Hybrid cloud is partially secure because the services which are running on the public cloud can be accessed by anyone. While the services which are running on a private cloud can be accessed only by the org's users.
Benefits	Public cloud can be owned at a lower cost than the private and hybrid cloud	Private Cloud provides high level of security and privacy to the users	Hybrid cloud is suitable for organizations which require more security than the public cloud
Benefits	Public cloud is maintained by the cloud service provider (CSP) and users do not need to worry about the maintenance of the cloud	Private cloud offers better performance with improved speed and space capacity.	Hybrid cloud helps to deliver new products and services more quickly
Benefits	Public cloud is easier to integrate, offers a better flexibility approach to the customers, public cloud is delivered through internet, apps, databases, virtual machines are accessible over the internet.	Each organization has their own private cloud where they have full control over the cloud because it's managed by the organzation itself.	Hybrid cloud offers flexible resources because public cloud and secure resources because of private cloud.
Cons	Public cloud is less secure because resources are shared publicly	Private cloud is accessible only within the organization level, the area of operations in private cloud is limited.	Security feature is not good as private cloud. Managing a hybrid cloud is complex because it's difficult to manage more than one type of deployment model.
cons	Performance of the public cloud resources Depends upon the high-speed network linked to the cloud provider. The client no control to the data underlying to the infrastructure	Private cloud is not suitable for organizations which has a high user base and organizations that do not have the pre-built infra, sufficient developers and can manage the cloud platform	In the Hybrid Cloud, the reliability of the services depends on the cloud providers
examples	Microsoft Azure, AWS, GCP, IBM bluemix, Oracle Cloud	HP data centers, Azure Stack, Vmware private cloud, Ubuntu canonical private cloud	Openstack, Azure hybrid cloud



Hybrid Cloud Platform

Different types of Cloud Models

	Cloud Model	Benefits	Examples
IaaS (Infrastructure as Service)	Rent for respective infrastructure required for business. End users are responsible to manage the underlying infrastructure - Compute - Cpu	Less complex Less development cycle	Azure Virtual Machine Azure Virtual Network Azure Disk

	- Memory - Disk storage - Containers - network		
PaaS (Platform as a Service)	Managed services (Compute, storage, database, data warehouse, analysis services) -- end users are not responsible to manage infrastructure in PaaS To manage - Compute - Disk storage - Network - OS - OS update/patch management - Security of the infra managed by the cloud providers Only responsible to manage their application on Azure	Less burden on infra provisioning Developers can focus more into their application or business logic Development language agnostic (.net, java, springboot, react/angular, python, ruby etc.)	Azure App Service (Web apps) Azure Storage Azure SQL database Azure Hadoop Services Azure Analysis Services Azure Data Lake Services etc.
Software as Service (SaaS)	Managed services, End users/customers are not responsible for the application development and deployment.	Almost zero burden on the end user in terms of app development. No burden on app scalability, reliability, disaster recovery, app backup or restore	Office 365 Gsuite gmail, Google drive Salesforce

Hands-on Lab

Creation of Azure Storage Account

1. Login to Azure portal (https://portal.azure.com)

Azure IaaS, PaaS & SaaS

Azure IaaS

IaaS - Infrastructure as Service (IaaS) is a type of cloud computing service which offers compute, storage and networking resources on demand on a pay-as-you-go basis.

Features

1. Migrating the organization's infrastructure to an IaaS solution helps to reduce the maintenance of on-premise data centers. It helps to save money on hardware costs, and gain real - time business insights.
2. IaaS solutions gives us the flexibility to scale the IT resources up and down with demand.
3. They can help to quickly provision new applications and increase the reliability of the underlying infrastructure.
4. IaaS lets us to bypass the cost and complexity of buying and managing physical servers and datacenter infrastructure.
5. A cloud computing service provider like Microsoft Azure when manages entirely the infrastructure, when you can purchase, install, configure and manage your own software - operating systems, middleware and applications.

Advantages of Azure IaaS -

1. **Reduce the capital expenditure and optimizes the cost** - IaaS can eliminate the cost of configuration and managing a physical datacenter, which is cost-effective choice of migrating to the cloud. The pay-as-you-go subscription model used by IaaS providers helps to reduce hardware costs and maintenance. It enables the IT team to focus on the business.

2. **Increases scale and performance of IT workloads** - IaaS helps to scale globally and deliver the applications globally.

3. Increases stability, reliability and supportability - With IaaS, there's no need to maintain and upgrade software and hardware or troubleshoot equipment issues. With the appropriate SLA, portal service, we can get 99.99% of uptime of cloud infra resources (VM, Azure Virtual Network etc.)
4. Enhance security - appropriate SLA (99.99%) in Microsoft Azure offers better security for the applications and data we can achieve.

Azure IaaS components - Azure Virtual Machine, Azure Virtual Network, Azure Disk Storage etc.

Azure PaaS (Platform as a Service)

Platform as a service (PaaS) in Azure is a complete development and deployment environment in the

cloud. With resources, it enables us to deliver the applications from single cloud based apps to cloud enabled enterprise applications.

When we purchase the resources from a cloud service provider (CSP), on a pay-as-you-go model, can access them over a secure internet connection.

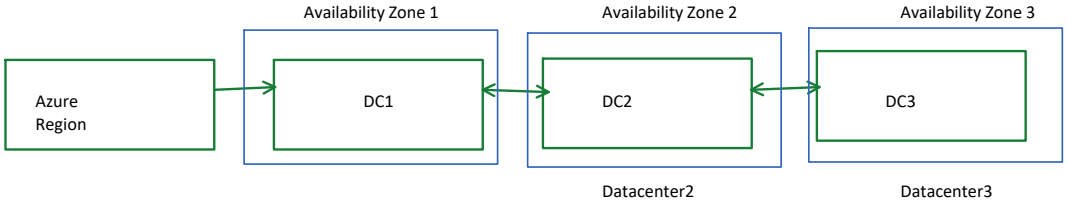
In PaaS, we focus only on the application development and deployment. Azure PaaS is designed to support the complete web application lifecycle - building(development), testing, deploying, managing and updating.

PaaS also allows us to avoid the expense and complexity of buying and managing software licenses.

Advantages of Azure PaaS

1. Cut the coding time - PaaS development tools can cut the time it takes to code new apps with pre-coded app components built into the platform - workflow, directory services or security features etc.

2. App development on Azure PaaS - PaaS can give the development team new capabilities without requiring new staff & having new skills.
3. Develop for web , mobile apps and develop new apps cross - platform - develop and deploy applications (web and mobile) apps much faster & deploy on Azure PaaS services.



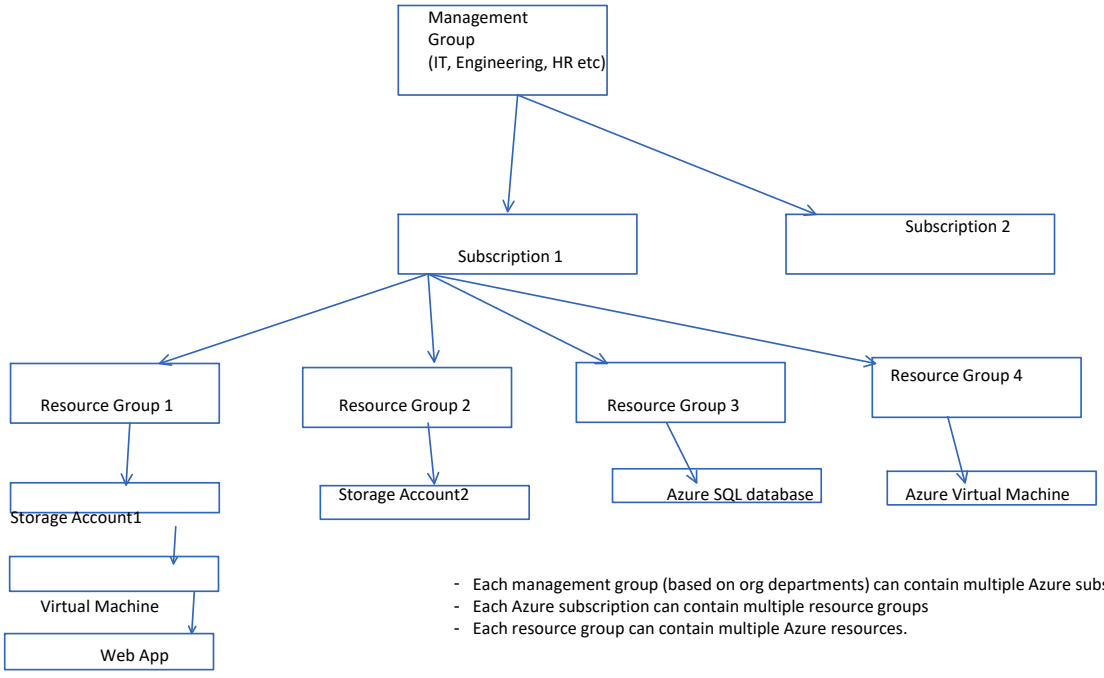
Each datacenters are separated from each other by 100 KM away & comprises of independent power, cooling and network speed.

All of these physical buildings (datacenters) are comprises of Azure region.

e.g. Central India (Pune) , South India (Chennai), Southeast Asia, East Asia

Resource : A resource in Azure is a manageable item which is available through Azure. Virtual Machines, Storage accounts, web apps, databases and virtual networks are examples of resources. Resource groups, subscriptions, management groups are also examples of resources.

Resource Group: A logical container which holds the related resources for an Azure solution. The resource group includes those resources which we can manage as a group. We can decide which resource belongs in a resource group based on what is required as per org policy.



- Each management group (based on org departments) can contain multiple Azure subscriptions
- Each Azure subscription can contain multiple resource groups
- Each resource group can contain multiple Azure resources.

Features of Azure resource group

- A resource group can be created at a specific Azure region

- Resource groups can be created, deleted, managed within the Azure portal.
- Azure resource groups can be used for administrative actions to manage the Azure resources.

Resource provider - A service which supplies the Azure resources.

All Azure Virtual Machine are part of "Microsoft.Compute" resource provider
All storage account resources are part of "Microsoft.Storage" resource provider

Azure Fundamentals

12 December 2022 19:55

Blob storage types -

Block Blob	Mainly used for storing of any objects data, like flat files, audio/video, media streaming, batch files, excel/csv/xml/json files
Page blob	Contains the virtual hard disk images
Append blob	Add new blob, made up blocks like block blobs. Append blobs are optimized for append operations.

Azure Storage types

Blob storage	Store any kind of object based storage
Queue storage	Store randomized messages
Table storage	Store the data files which are not RDBMS compliant
File Share	Store any on-premise windows/linux compatible file systems
Disk storage	Store any OS compatible disk drives (VHD, VHDX)

Basics of PowerShell Scripting

12 December 2022 19:55

Features	Programming Language	Scripting Language
Mode of execution	Programming language need to transform the high level Programming code into machine languages(byte code) through a compiler.	Compilation step is not required for the scripting language because they use an interpreter
	Compiler used in the programming language compiles the whole bunch of code at a time	Interpreter executes code line by line
Basic Difference	Objects are consisting of data and code,	Scripting languages are of functions & actions. The scripts are designed for a specific runtime environments.
Interpretation mechanism	Objects Oriented (OOPS) employ a compilation step.	Scripting language executes scripts inside the another parent language while OOPS languages like C, C++ are executed without a parent program.
Platform	OOPS can be executed in various platform (cross-platform)	Scripting languages are specific to the platform
Translation	In OOPS, a complete chunk of code is compiled and converted into machine level byte code	Scripting language are not self-executable and need to be encapsulated in a host to execute the scripts.
examples	Java, C#, Go	PowerShell, Shell scripting, Python, javascript, PHP

Features of PowerShell

1. PowerShell Remoting - PowerShell remoting allows scripts and cmdlets to be invoked on a remote machine.
2. Background jobs - it helps to invoke script or the pipeline asynchronously, we can run the jobs either on the local machine or multiple remotely operated machines.
3. Transactions - enable the cmdlets and allows the developers to perform transactions
4. Network file transfer - PowerShell offers native support for prioritized, asynchronous, throttled, transfer of files between the machines using the background intelligent transfer (BITS) technology
5. Evening - This command helps us to listen, forwarding, and acting on management and system events.

PowerShell cmdlets

A cmdlet is called Command let, it's a lightweight command used in the Windows base PowerShell environment. PowerShell invokes these cmdlets in the command prompt. We can create and invoke cmdlets command using PowerShell APIs.

Cmdlets	Command
Cmdlets in PowerShell are .NET framework class objects, so It can't be executed separately.	

Cmdlets can be constructed from a few lines of code	
Parsing, output formatting, and error presentation are not handled by the cmdlets	
Cmdlets are record based so that it processes a single object at a time	
Get - to get action Start- to run action/execution Out - to output something Stop - to stop something Set - to define something New - to create something	Get-Help

Concepts on PowerShell

Cmdlets	Cmdlets are the built-in commands written in .net language like C#, VB. It allows the developers to extend the set of cmdlets by loading and write powershell scripts
Functions	Functions are commands which's written in the PowerShell language. It can be developed without using any other IDE like VS or VS code
Scripts	Scripts are the text files on disk with .ps1 extension
Applications	Applications are existing windows programs
What if	Tells the cmdlet not to execute. But to define what would happen if the cmdlet is being executed
Debug	Instructs the cmdlet to provide the debugging information
ErrorAction	Instructs the cmdlet to perform a specific action when an error occurs.
ErrorVariable	It specifies the variable which holds the error information
OutVariable	Tells the cmdlets to use a specific variable to hold the output information

PowerShell datatypes

Boolean	True or false condition
Char	An 8-bit unsigned whole number from 0 to 255
Date	A 16 bit unsigned number from 0 to 65535
Decimal	A 128 bit decimal value
Double	A double precision 64 bit floating point number.
Integer	A 32 bit signed whole number
Long	A 64 bit signed whole number
Object	Description
Short	A 16 bit unsigned number
Single	A single precision 32 bit floating point number
String	A grouping of characters which is called text

Intro to Big Data

12 December 2022 19:55

Traditional Constraints & Reasons for evolving in Big Data

1. **Data volume** aspects - KB, GB, TB, PB, ZB (10 to the power 9-11)

2. **Data Variety** aspects - unstructured, semi-structured, quasi-structured format , structured

RDBMS , weblog, clickstream, BI -> consumption layer (semi-structured data, unstructured data or quasi-structured data)

3. **Data Velocity** - IoT devices, sensors (latitude, longitude, temperature), fast rate the data should be processed and transformed.

4. **Data Veracity** - measurement of different aspects of data. Twitter feed data, Clickstream from various ecommerce websites, weblog data comes from various web servers, crime data, weather data, social events data

Big Data Processing

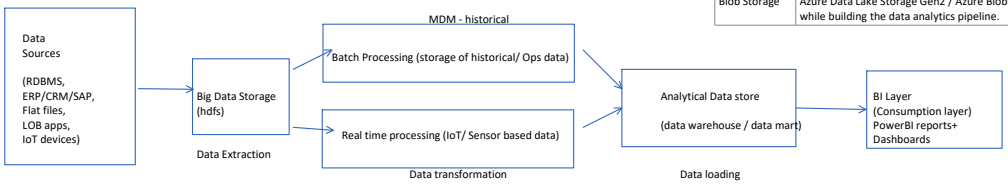
- Batch Data processing
- Real time streaming data processing

e.g. Social Sentiment Analytics (Twitter feed data)

Core Concepts on Big Data framework

- Data computation happens where the data is being stored (big data storage systems).
- Data is never moved to anywhere for computation

Big Data Analytics pipeline architecture



(Social media,
Website - Clickstream, Weblog
Weather data from sensors in case of real Time data)

Extract - Transform - Load (ETL) jobs in data analytics

Resource	Purpose	Utility
Apache Hadoop	Big Data Solution & provides platform Through which we can store, transform and manage the big data solutions. Apache Hadoop is the Big Data cluster which contains the HDFS and the MapReduce component.	Big Data Analytics use cases we use hadoop cluster for storing and computation of Big Data.
HDFS	Distributed file based storage layer through which we can store any specific data file format (csv, txt, parquet, avro etc.)	It's the distributed storage layer of apache hadoop cluster
MapReduce	It's the distributed data processing/transformation layer through which we can perform the computation on the data.	The Mapreduce job can help us to perform the actual data computation/transformation through which we retrieve the big data analytics job results
YARN	It's the next gen Mapreduce platform which helps for the advanced data processing for complex file formats. It provides better throughput and low latency while processing the data computation. It's also called Mapreduce V2.	It's more advanced mapreduce job processing framework which provides guaranteed data computation with maximized throughput and low latency.
HDInsight	It's the managed Apache Hadoop cluster on Azure platform. In HDInsight, the HDFS layer consists of by Azure Data Lake Gen2/ Azure Blob storage. The core data computation is performed by the Mapreduce layer	It's the hadoop based data computation / transformation platform where the core data processing takes place of the ETL/ELT pipeline.
Apache Pig	It's the scripting based ETL / ELT analytics component through which we can perform data transformation on hadoop cluster.	It's the ETL based data computation platform
Apache Hive	It's the data warehousing solution on Apache Hadoop cluster. In hive, we can create the tables and upload the data within the tables & run the queries through Hive QL language.	We can perform the core data transformation through ETL/ELT pipeline for Big data Hadoop clusters through Hive.
Azure Data Factory	Azure Data Factory is the cloud based data orchestration and data integration resource which helps to move/copy data through data source to sink through the ETL/ELT data analytics pipeline.	It's the managed data orchestration and integration resource on Azure which helps to copy the data from data source to destination for data analytics pipeline with scale.
Azure Data Lake Storage Gen2 & Azure Blob Storage	It's the managed azure based data storage platform through which we can store the object based data. Azure Data Lake Storage Gen2 / Azure Blob storage is the data ingestion layer while building the data analytics pipeline.	It's the data ingestion layer.

Use cases of Big Data

Big Data involves the data produced by different devices and applications.

- Social Media apps - Social media like Twitter, Linkedin which holds the versatile volume of data, consists of more petabytes to zetabytes level of data leads to the way of using Big Data platform

- Stock Exchange data - It holds the information related buyers and sellers decisions based on stocks made by different companies,

- Power Grid data - Power grid data which holds the information consumed by a particular node with the respect to the electrical base station of substation.

- Transform data - data includes model, capacity, distance and available of the vehicle
- Search engine data - search engines retrieve data from different databases

Big Data examples according to the data variety

a) **Structured data** - Relational data
b) **Semi- Structured data** - XML, Json data
c) **Quasi-Structured data** - Json, sensors data, flat file data
d) **Unstructured data** - Word, PDF, text, media logs, weblogs

Features of Big Data

1. Big Data Analytics is the use of advanced analytics technique against very large, diverse data sets include structured, semi-structured and unstructured data from different sources and in different sizes from terabytes to zettabytes.

2. Big Data is a term applied to data sets whose size or type is beyond the ability of traditional relational databases to capture, manage and process the data with low latency.

3. Big Data has characteristics - high volume, high velocity and high variety
4. Artificial Intelligence (AI), mobile, social and IoT are driving data complexity through which new forms and sources of data are considered.

5. Example - Big Data comes from Sensors, devices, audio/video, networks, log files, transactional applications, web and social media data. Much of it generates in real time and at a very large scale.

Characteristics of Big Data

1. Big Data analytics is the advanced analytics techniques against very large, diverse data sets which include structured, semi-structured and unstructured data from different sources and in different sizes from terabytes to petabytes.

2. Big Data can be defined as data sets whose size or type is beyond the ability to traditional relational databases to capture, manage and process the data with low latency.

3. The characteristics of Big Data include high volume, high velocity and high variety.

4. Sources of data are becoming very complex than those for traditional data because they are driven by AI, mobile apps, social media data and IoT device data.

5. With Big Data analytics, we can ultimately get the real time data insights and leads the way to decision making, data modelling and prediction of data for enhanced business intelligence.

6. The Core Big Data frameworks like open source - Apache Hadoop, Apache Spark and the entire hadoop ecosystem tools which are cost-effective, flexible data processing, and storage tools designed to handle the volume of data being generated today.

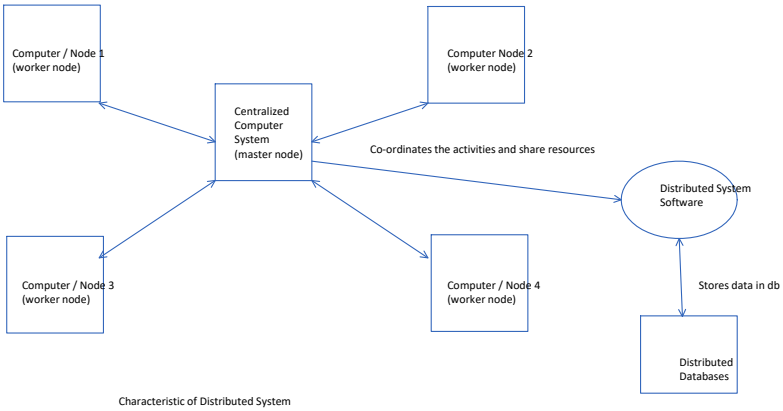
Benefits of Big Data

1. Social media data helps to gather about the marketing campaigns, product promotions are analytics evolvements
2. Enterprises can use market basket analysis, customer churn analytics, predictive analytics, helps plan their new products, new campaigns & revenue model
3. Using Big Data in healthcare industry, the health records of the patients can be analysed in more details and can provide better insights
4. Banking industry, Big Data can provide more and more insights on customer purchase pattern, credit card usage, demographics which can lead to better sales profit margin analysis and recommend new products to customer.

Distributed System

It's a collection of autonomous computer systems which are typically separated but are connected by a centralized computer network which is equipped with distributed system software.

The autonomous computers will communicate among other system by sharing resources and files and performing the tasks assigned to them.



Characteristic of Distributed System

1. Scalable
2. Concurrency
3. Fault tolerance
4. Transparency
5. Heterogeneity

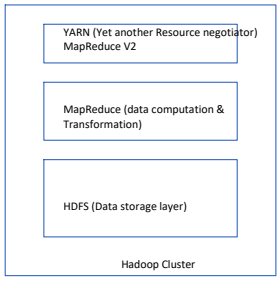
Apache Hadoop

Apache Hadoop is the distributed, fault tolerant open source framework dedicated for Big Data analytics platform.

Apache Hadoop is the official Big Data framework / tool released from ASF (Apache Software Foundation).

Feature of Apache Hadoop

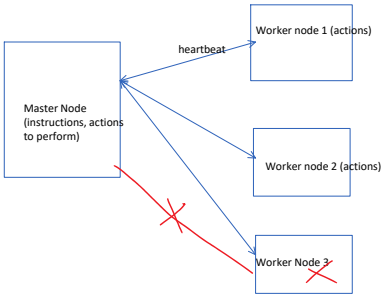
1. Apache Hadoop consists of two layers - Storage layer (HDFS - hadoop distributed file system)
2. MapReduce -> core data computation layer



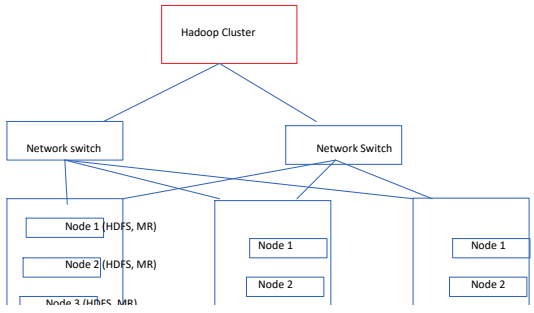
Hadoop v2.x onwards

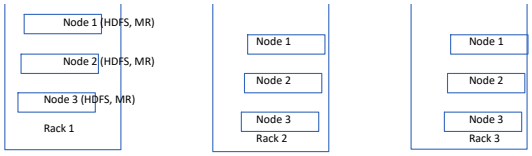
Rest of other Hadoop ecosystem components -

1. Apache Hive - Data warehouse and analytics tool in hadoop framework
2. Apache Pig - data querying tool with scripting language (ETL jobs)
3. Apache Kafka - real time data processing framework built on top of hadoop



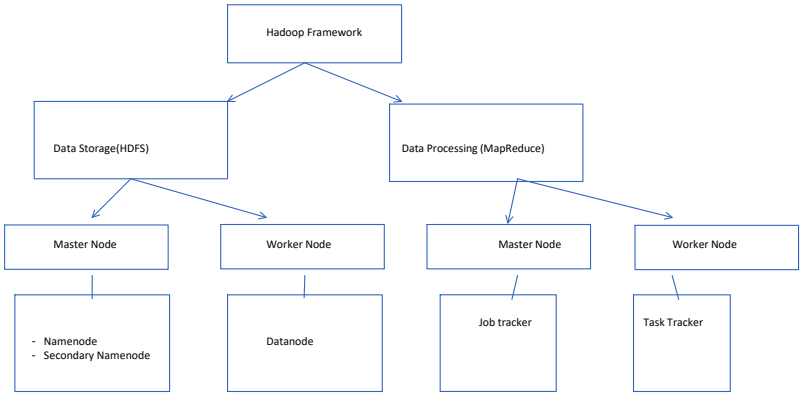
- Apache hadoop is a distributed, fault tolerant, highly available Big data ecosystem cluster which works with master node & worker node design.
- The master node is responsible for sending the instructions and job details to the worker node(S).
- The worker nodes processes the jobs and produces the output as per the instructions provided the master node
- Any point of time if any worker node failed to respond to the master on time, that worker node gets discarded, new worker nodes are being assigned as per the horizontal scaling feature.
- A collection of nodes is called cluster in hadoop
- A node is a point of intersection/ connection within a network, i.e. server.





Data Storage (HDFS)

Master Node - HDFS Namenode, Secondary Namenode
Worker Node - HDFS Datanode



HDFS Operations

1. HDFS has master and worker node architecture.
2. An HDFS cluster consists of a single Namenode, a master node that manages the file system namespace and regulates access to the files by clients.
3. In addition, there're number of datanodes, usually, one per node in the cluster which manages storage attached to the nodes that they run on.
4. HDFS exposes a file system namespace and allows user data to be stored in files.
5. Internally, a data file is being splitted into one or more blocks and these blocks are stored in a set of datanodes.
6. The namenode executes file system namespace operations like opening, closing, and renaming files and directories.
7. It also determines the mapping of hdfs datanode blocks to individual datanodes.
8. The datanodes are responsible for serving read and write requests from the file system's directories from the clients.
9. The datanodes also can perform block creation, deletion, and replication upon instructions given by Namenode.

Starting from Hadoop V2.x, each & every datablocks in HDFS datanode should be size of 128 mb.

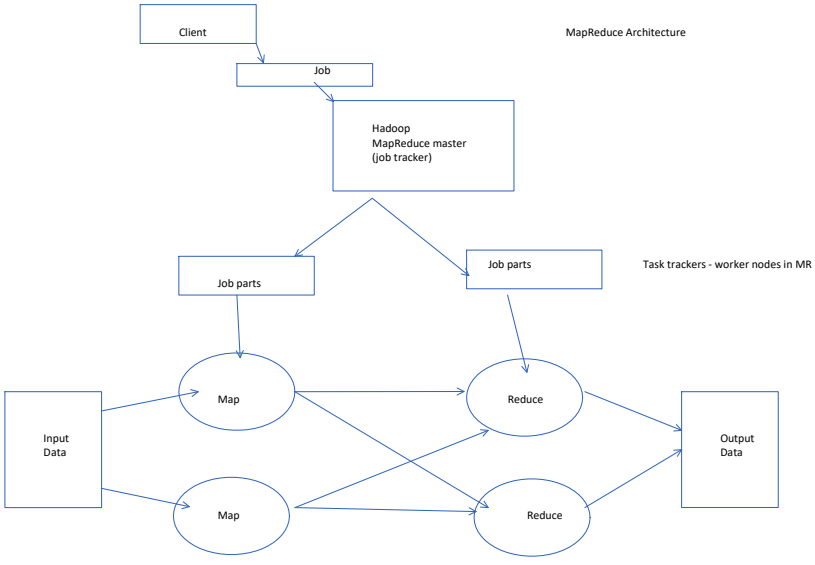
Input data file = 1GB = 1024 MB / 128 mb = 8 blocks of data would be stored in data node

Features of MapReduce

1. MapReduce and HDFS are the two major components of Hadoop which makes it so powerful and efficient to use.
2. MapReduce is programming model used for efficient processing in parallel over large data sets in a distributed manner.
3. The input which is first stored within datanode blocks of HDFS, is first split and then combined to produce the final result.
4. The libraries for MapReduce is written in so many programming languages with various different optimizations.

Benefits of MapReduce

1. The purpose of MapReduce is to Map each of the jobs and then it will reduce the processing power.
2. The MapReduce task is mainly divided into two phases Map phase and Reduce phase.



1. Client - The MR client is one which brings the job for data computation for processing. There can be multiple clients available that can continuously send jobs for processing to the hadoop MR manager
2. Job - The MapReduce job is the actual work that the client performs which consists of so smaller tasks that client executes to process the data.
3. Hadoop MR master - it divides the particular job into subsequent job parts.

4. Job parts - The task of job parts which are obtained after dividing the main job. The result of all the job parts are combined to produce the final output
5. Input data - the data which is fed into MR for processing and transformation
6. Output data - refers to the final data i.e. processed and transformed data.

Job Tracker - the purpose of job tracker is to manage all the resources and all the jobs across the cluster and also to schedule each map on the task tracker running on the same data node. Since there can be hundreds of data nodes available in the cluster.

Task Tracker - The task tracker can be considered as the actual slaves which are working on the instruction given by the job tracker. This task tracker is deployed on each of the nodes available in the cluster which executes the Map and Reduce tasks as instructed by the job tracker.

Job history server - the job history server is a daemon process which saves and stores historical information about the task or application.

Big Data use cases

1. Product development - social media to develop social sentiment analysis
2. Predictive maintenance - Aerodynamics,
3. Customer experience - Telecom, Banking, Retail
4. Fraud Detection and Compliance - Banking, Financial
5. Drive innovation - BFSI, Healthcare and marketing domains

Apache Pig

1. Apache Pig can handle structured, semi-structured or unstructured data & stores the data into HDFS.
2. Every task or query running on pig is executed through MapReduce.

Benefits

a) Ease of programming. We can write the queries in Apache Pig through scripting.
b) Optimization of complex data processing (Load, Transform & DUMP)
c) Flexible and with built-in operators (sort, filter, join, foreach)
d) Supported for Avro based file formats (row wise)

Features of Apache Pig

- For performing several operations, apache pig provides rich sets of operators like the filters, join, sort etc.
- Joins operations are easier in Pig.
- Apache pig allows splits in the pipeline, data structure is multivalued, nested and richer.
- Pig can handle for analysis for both structured and unstructured data.
-

Data types in Pig

1. Tuple - it's an ordered set of the fields
2. Bag - It's a collection of the tuples
3. Map - It's a collection of key/value pairs.
4. Atom - It's an atomic data value which's used to store as a string. It can be used as a number and string.

MapReduce	Apache Pig
It’s a low level data processing	It's a high level data flow tool
Complex programs can execute through java/python	Apache pig we can execute simple pig/latin scripts which are simpler also consists of less no of lines.
Data operators are difficult compared to pig	Pig latin these built-in operators are available to work with
It does not allow nested data types	It provides the nested data types like tuple, bag and map

Apache Hive

It's data warehouse tool on Apache hadoop framework. Hive is used for data analysis in Big Data platform.

a) Tools to enable easy access to data via SQL like query interface which is called HQL
b) Enables us to perform data warehouse tasks such (extract, transform and load) ETL operations, analysis and reporting.
c) Query processing , execution is done on hive through Mapreduce.
d) In hive, we can query the data through SQL like query interface called as Hive QL.
e) Hive supports various file formats like CSV/TSV, apache parquet (columnar file format which is performant), Apache ORC (row-columnar file).
f) Hive queries can be executed based on query retrieval support when using Hive LLAP, Apache Yarn etc.
g) Hive can operate with declarative SQL support.

Apache Pig	Apache Hive
Pig operates on the client side, since it's client side Scripting language	Hive operates on the server side of the cluster
Pig uses the pig-latin script for analysis	Hive uses the hive-ql language
Pig/latin is purely procedural language	Hive is declarative SQL language
Pig has support of AVRO file format (row - oriented)	Apache Hive is suitable for Parquet and ORC file format
Apache Pig is suitable for complex and nested data structure	Hive is suitable for match processing and OLAP system
Apache Pig does not support schema to store data	Apache Hive has the support for JDBC and ODBC
Pig does not have any metastore(metadata store)	Apache hive has the metastore support for DDL language (SQL, mySQL, postgreSQL) etc.
Pig operates quickly and data loads quickly	Hive loads data slowly
.pig file is extension for pig-latin scripts	Any file formats(.hql) applicable for hive data query files.

Hands-on Lab

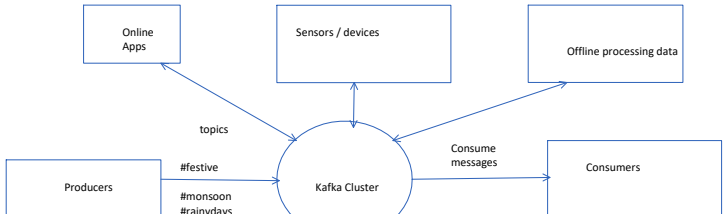
Create Azure Virtual Machine (on Linux)

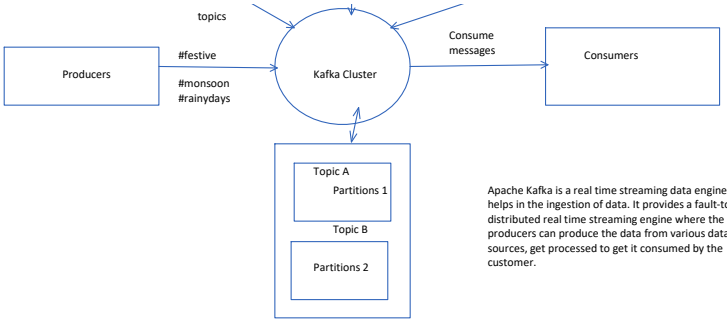
Pre-requisites

- Need to have the Azure Subscription (free trial)
- Resource provider, we've to register the resource "Microsoft.Compute"

Hadoop ecosystem components

- HDFS (data storage)
- Mapreduce (data computation)
- YARN (resource management / MR v2)
- Pig (data analysis)
- Hive (data warehouse)
- Kafka (real time data streaming)
- Spark (real time data transformation and processing)
- Sqoop (bidirectional transfer of data from hadoop / HDFS to any RDBMS (MSSQL mysql etc.)
- Flume (data ingestion tool)
- Oozie (workflow management tool)

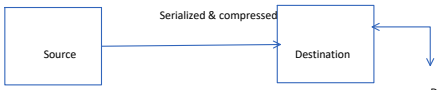




Apache Kafka is a real time streaming data engine which helps in the ingestion of data. It provides a fault-tolerant, distributed real time streaming engine where the producers can produce the data from various data sources, get processed to get it consumed by the customer.

Serialization of Data

In HDFS for hadoop cluster



Data gets Deserialized & stored
In its original format

HDFS federation & High availability

1. HDFS federation is the process through which the namenode failure can overcome in case of namenode is failed and the single point of failure happens.
2. HDFS has primarily two components

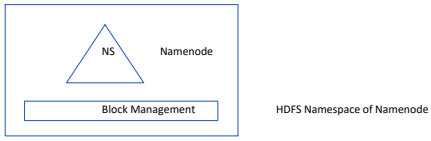
- Namespace - consists of files, directories and blocks
- Block storage service - which has two parts

a) Block management

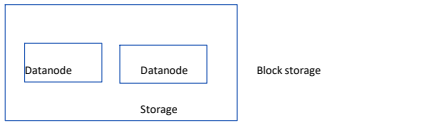
b)

	Provides Datanode cluster membership by handling registration, also sends periodic heartbeats.
	Provides the block report and maintains the location of blocks.

Manage replica place, block replication for under replicated blocks, deletion of blocks which are replicated.



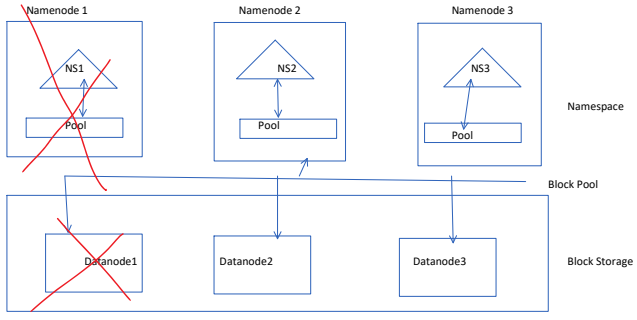
HDFS Namespace of Namenode



Block storage

Multiple Namespaces / Namenodes

As name service/ namespace gets scaled horizontally, HDFS federation uses multiple namenode/ namespaces. The namespaces are federated, the namenodes are independent and do not require any co-ordination. The Datanodes are used as common storage for blocks by all namenodes. Each datanode gets registered with all of the namenodes in the cluster. Data nodes to send periodic heartbeats and send block reports. They can handle commands from the namenodes.



Block Pool -

It's a set of blocks which belongs to a single namespace. Datanodes store the blocks for all block pools in the cluster. Each block pool is managed independently. This allows a namespace to generate Block Ids for new blocks without the need for co-ordination with other namespaces.

A namenode failure does not prevent the datanode from serving other namenodes in the cluster.

A namenode and its block pool together are called namespace volume. A self contained unit of management. When a namenode/namespace is deleted, the corresponding block pool at the datanode is deleted. Each namespace volume is upgraded as unit during the cluster upgrade.

Benefits of Namenode HA & Federation

1. Namespace Scalability - Federation adds namespace horizontal scalability, large deployments or deployment using lot of small files benefit from namespace scaling by allowing more namenode to be added to the cluster
2. Performance - File system throughput is not limited by a single namenode. Adding more namenodes to the cluster scales the file system read/write throughput.
3. Isolation - A single namenode can offer no isolation in a multi user environment. It can overcome by using HDFS federation, where different applications can be isolated by a different namespace by using multiple namenode.

Apache Hadoop Overview

12 December 2022 19:55

HDFS Caching

Centralized Cache management in HDFS is an explicit caching mechanism which allows users to specify paths to be cached by HDFS. The namenode will communicate with datanode which has the desired blocks on disk, and instruct them to cache the blocks in off-heap cache.

Centralized cache management in HDFS

1. Explicit caching of data, helps to evict the old data from memory. This is particularly important when the size of working set exceeds the size of main memory.

2. Because, datanode caches are managed by the namenode. Applications can query on the set of cached block of data when making task placement decisions. Co-locating a task with a cached block replica improves the read performance.
3. When a block is cached by the datanode, clients can use a new, more-efficient, zero-copy read API, since the checksum verification of cached data is done by the datanode.
4. Centralized cache can improve the overall cluster memory utilization, when relying on the OS buffer at each datanode, repeated blocks will result all n replicas of the block being pull down into the buffer cache. With the centralized caching, a user can explicitly specify the n replicas saving memory.

Use cases

Centralized cache management is useful for files that being read/accessed repeatedly. For example, a small fact table in hive is often joins is good candidate for caching.

Centralized caching,

For mixed workloads, with performance SLA. Caching is a working ser of high priority workload ensures that it does not content for disk I/O with a low priority workload.

MapReduce Framework

1. A Mapreduce job usually splits the input dataset into the independent chunks of data which are being processed by the map tasks in a completely parallel manner.
2. This Framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically both the input and output of the jobs are stored in a file system.
3. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks.
4. Typically, the compute nodes and storage nodes are the same, MR framework & HDFS are running on the same set of nodes. This configuration allows the framework to effectively schedule tasks on the nodes where data is already being present resulting in very high aggregate bandwidth across the cluster.
5. The MR framework consists of single master ResourceManager, one worker NodeManager per cluster-node and MRAppMaster per app.
6. Minimally, apps can specify the input and output locations and supply map and reduce functions via implementation of appropriate interfaces and abstract classes. These and other job parameters are referred as the job configuration.
7. During mapreduce job execution, the hadoop client submits the job and shares the configuration with the ResourceManager which then assumes the responsibility of distributing the software/configuration to the workers.
8. Scheduling tasks and monitoring them provides the status and diagnostics information to the job-client.

MapReduce Input and Output

The MR framework operates exclusively on <key, value> pairs, this framework to the job as a set of <key, value> and produces a set of <key, value> pairs as the output of the job.

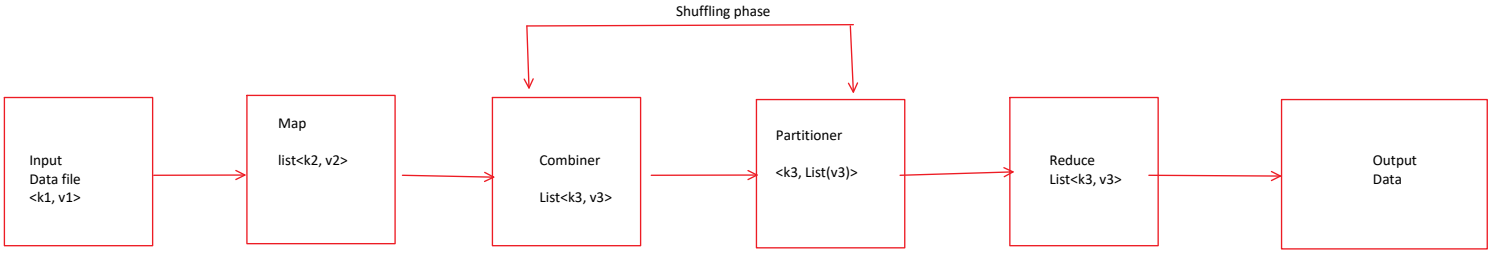
The key and value classes have to be serializable by the framework and hence need to implement the Writable interface.

Input and Output operation types of MR job

(input) <k1, v1> -> **map** -> <k2, v2>-> **combine** -> <k2, v2> -> **reduce** -> <k3, v3> (output)

Map Reduce job execution life cycle

Mapper -> combiner -> Reducer



The input data is
Initially processed by all the
Mappers/ map jobs and the
Intermediate output is created.

1. Partitioning of keys of the intermediate map output
Is created which is controlled by partitioner.
2. On the basis of key/value pair, each map output is partitioned.
3. The data records having same key value goes into same partition(within each mapper), then each partition is mapped sent to the reducer.
4. Partition phase runs between mapper and reducer.

Apache Hadoop (Deep Dive)

12 December 2022 19:56

Partitioning of Hive - Hive Partition is a way to organize large tables into smaller logical tables based on values of columns. One logical table (partition) for each distinct value. In Hive, tables are created as a directory in HDFS. A table can have one more partitions which corresponds to a sub-directory for each partition inside the table directory.

Example: A student details table in Hive can be partitioned based on Departments.

The partitions can create the sub-directories which leads to the faster data processing.

Bucketing of Hive - It's technique to split the data into more managable files.

Features of Partitioning and Bucketing

Hive Partitioning	Hive Bucketing
It's the process of organizing tables into partitions for grouping same type of data together Based on a column or partition key. Each table in the hive can have one or more partition keys to identify a particular partition. Using partition, we can make it faster to execute queries on slices of data.	In Hive tables or partition are subdivided into buckets based on hash function. It's more efficient process because it gives extra structure to the data to be used for more efficient queries.
Pros - 1. It distributes the execution load horizontally 2. In partition faster execution of queries with the low volume of data takes place.	Pros - 1. It provides faster query response like partitioning 2. In bucketing, due to equal volumes of data in each partition, joins at Map side will be quicker.
Cons: 1. Theirs is the possibility of too many small partitions creations - too many directories 2. Partition is effective for low volume of data. There're few queries which takes long time to execute. 3. There's no need for searching entire table column for a single record	Cons: We can define a number of buckets during the table creation. But loading of an equal volume of data which has to be done manually by programmers.

HDFS commands

1. Create a directory:

hdfs dfs -mkdir /user

2. Check the contents of the directory within HDFS

Hdfs dfs -ls /user

3. Copy the files from local drive to hdfs

Hdfs dfs -put /home/user/file.txt /user
Hdfs dfs -CopyFromLocal <local_file_path> <hdfs_file_path>

4. Create a new empty file

Hdfs dfs -touchz /user/username

5. Delete a file from the HDFS directory

Hdfs dfs -rm /demo/sample.txt <hdfs_directory_path>

6. Copy the file back to local drive from hdfs

Hdfs dfs -copyToLocal <local_file_path> <hdfs_file_path>

Apache Sqoop

Sqoop is a tool which is designed to transfer the data between hadoop and relational databases. We can use sqoop to import data from a RDBMS such as mysql or Oracle, postgreSQL, sqlite into the HDFS, transform the data in hadoop mapreduce, then export the data back to RDBMS.

Sqoop automates most of the process, relying on the database to describe the schema for the data to be imported. Sqoop uses MapReduce jobs to import and export the data, which provides the parallel operation as well as fault tolerance.

Features of Sqoop :

1. With Sqoop, we can import data from a relational database system into HDFS. The input to the import process is a database table. Sqoop will read the table row-by-row basis into HDFS.
2. The output of this import process is a set of files containing a copy of the imported table. The import process is performed in parallel. The output will be in multiple files. These files may be in delimited text files (commas, tabs separated fields)
3. Sqoop uses jdbc driver in order to connect to the hive table or sql table in rdbms.
4. Using Conditional import, can specify which data to import like --table argument, to select the table to import. By default, all columns within a table can be selected for import.
5. Imported data is written to HDFS in its natural order, so that once the table containing the columns like A, B, C, it would result import of data like
   A1, B1, C1
   A2, B2, C2 ..

   Incremental Import in Apache Sqoop

   Sqoop provides an incremental import mode which can be used to retrieve only the rows newer than the some previously-imported set of rows.

   Arguments in incremental import of Sqoop

   -- check-column -- specifies the column to be examined when determining which rows to import
   -- incremental (mode) -- specifies how sqoop determine which rows are new,
   --last-value (value) - specifies the max value of the check column from the previous import.

a) Sqoop supports two types of incremental imports: append and lastmodified.
b) We can use the -- incremental argument to specify the type of incremental import to perform.
c) We should specify append mode when importing a table where new rows are continually being added with increasing row id values. The columns containing the row's id with --check-column.
d) Sqoop actually can import the rows where the check column has a value greater than the one specified with --last-value.
e) At the end of the incremental import, the value which should be specified as --last-value for subsequent import is printed on the screen. While running a subsequent import, we should specify --last-value to ensure only the new or updated data can be imported.

File Formats in Sqoop

We can import data through sqoop in one of the two file formats - delimited text and SequenceFiles.

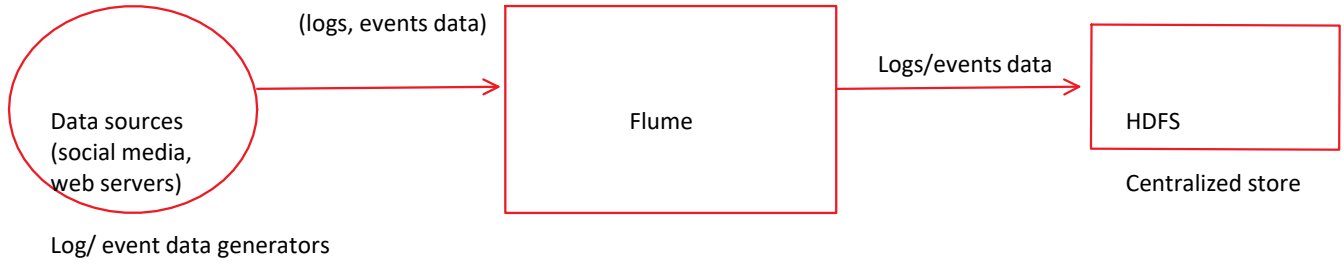
Delimited text is default import format. We can also specify it explicitly by using the --as-textfile argument. This argument will write string-based representations of each record to the output files, with delimiter characters between the individual columns and rows.

- The delimiters may be commas, tabs, or other characters.
- Delimited text is appropriate for most of non-binary data types. It also readily supports further manipulation by other tools (i.e. hive).
- Sequence files are binary format which store individual records in custom record-specific data types. These data types are manifested as java classes. Sqoop will automatically generate these data types. The format supports exact storage of all data in binary representations. It's appropriate for storing binary data, or data which will be manipulated by the custom MapReduce programs.

Apache Flume

Apache Flume is a tool/service/data ingestion mechanism for collecting, aggregating and transporting large amounts of streaming data such as log files, events from various sources to a centralized data store.

Flume is highly reliable, distributed and configurable tool. It's designed to copy streaming data from various web servers to HDFS.



Benefits of Flume

1. Using Apache Flume, we can store the data in any of the centralized stores (HDFS).
2. When the rate of incoming data exceeds the rate, at which the data can be written to the destination, Flume acts as a mediator between the data producers and centralized stores. Provides a steady flow of data between them.
3. The transactions in Flume are channel based where two transactions (one sender, one receiver) are maintained for each message. It also guarantees reliable message delivery.
4. Flume is reliable, fault tolerant, scalable, manageable and customizable.

Flume Interceptors

1. Flume Interceptors are those who has the capability to modify/drop events in-flight.
2. There're different kinds of interceptors in flume -

a) host-flume interceptors - it inserts the hostname/IP address of the host where the agent is running on.

b) Regex-filtering interceptors

c) Remove-header interceptors

d) Static interceptors

e) Timestamp interceptors - it inserts the event headers , the time in which it processes the event.

Azure Data Factory

12 December 2022 19:56

ETL Definition

ETL is a data integration process which combines data from multiple data sources into a single, consistent data store which's loaded into a data warehouse / consumption layer or other BI systems.

ETL is the process for integrating and loading the data for computation and analysis, it's also the primary method to process data for data warehousing projects.

Benefits

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

Azure Data Factory is a managed cloud service which's built for these complex extract-transform-load (ETL) , extract-load-transform(ELT) and data integration projects.

Data orchestration is the practice of acquiring, cleaning and matching, enriching and making data accessible across the technology systems. 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 a schedule.

ETL vs ELT

ETL	ELT
ETL (extract, transform, load) can perform the end to end data transformation after loading the raw data into the transformation layer and captures the insights from the data.	ELT (extract, load, transform) 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 directly to the target data store to be transformed as required.
The ETL process, involves more structured datasets and data has to be relational 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 dataset and ideal for big data use cases.
In on-premise, ETL data pipeline is more common	On Cloud , Azure ELT pipeline is more popular

ETL process

Extract

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

- SQL / noSQL
- CRM / ERP
- Flat files
- Web pages

Transform

In the staging area, the raw data undergoes the data processing, data is transformed and consolidated for its intended analytical use case.

- Filtering, cleansing, de-duplicating, validating, and authenticating the data
- Performing calculations, translations, summarizations of the 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.

Load

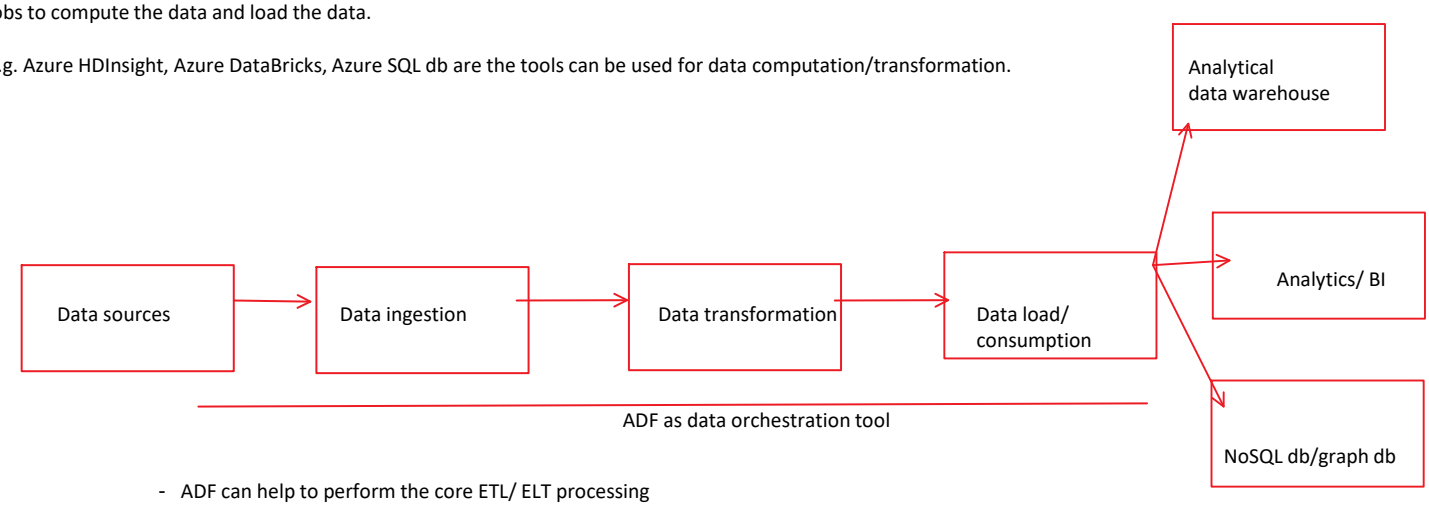
In the last step, the transformed data is moved from the staging area to a target data warehouse. Which involves loading of all the data, followed by the periodic loading of incremental data changes and full refreshes to erase and replace the data in the data warehouse.

ETL Tool (Azure Data Factory)

1. Comprehensive automation support - leading ETL tools like 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 interface - the functionality can be used for specifying rules and data flows.
3. Support for complex data management - complex calculations, data integrations and string manipulations.
4. Security and compliance - the best ETL tool encrypt the data both in motion and at rest includes various regulations like GDPR, HIPAA etc.

Data factory is the cloud based ETL and data integration service allows to create data-driven workflows for orchestrating data movement and transforming the data at scale. Using data factory (ADF), we can orchestrate, create and schedule the data driven workflows (pipeline) which can ingest the data from disperate data sources and can build complex ETL jobs to compute the data and load the data.

e.g. Azure HDInsight, Azure DataBricks, Azure SQL db are the tools can be used for data computation/transformation.



- ADF can help to perform the core ETL/ ELT processing
- Integration of data from various data consumers, data marts or warehouses
- Allows us to create the data driven workflows through ingestion, transformation and consumption
- Code free data analytics service which can help us data ingestion, data pipeline and consumption through the data flows, control flows. We can monitor and schedule the ADF pipeline

Azure Data Factory components

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 to the data factory.

   e.g. db connection strings which define the connection information required for the service to connect to the external resource.

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

2. Activity - It's the resource defines us which action / operation to be performed on the data.
   - Copy data,
   - Deduplicating the data
   - Formatting the data to remove all nulls, NANs
   - Applying normalization, remove redundancies
   - Adding column headers and formatting
   -

3. Dataset - dataset represents the data source (sql table, flat files)

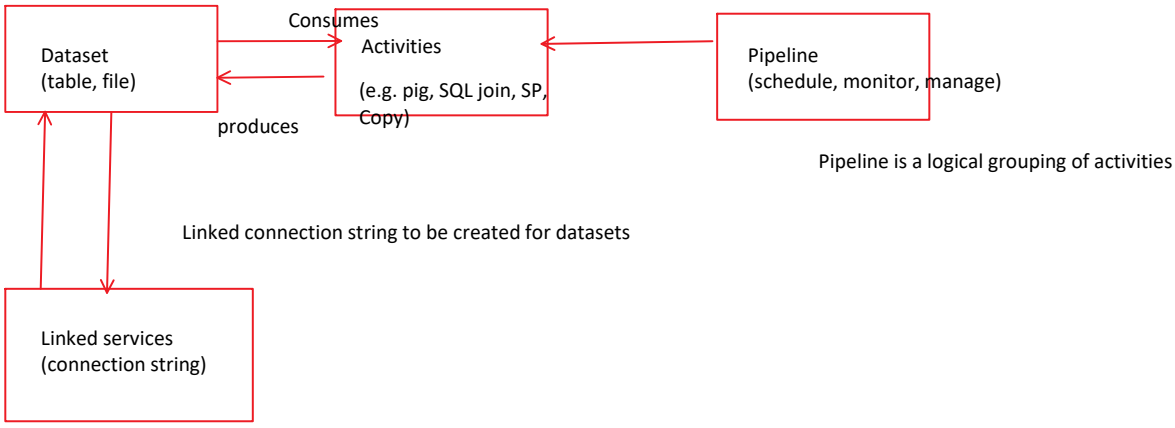
4. Pipeline - an integrated set of ADF activities

A data factory can contain one or more pipelines.

a) A pipeline is the logical grouping of activities which together perform a task.
For e.g. the 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.

b) 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.

c) The activities in a pipeline defines the actions to perform on the data.



Hands-on Lab: 1

Pre-requisites : Azure Portal

Tasks

1. Create Azure Data Factory
2. Explore Azure Data Factory Studio
3. Explore ADF activities, data sources, pipelines
4. Monitoring and management capacity of ADF

Hands-on Lab : 2

Transferring data from Azure SQL db to Azure Blob storage
through ADF Copy Data Tool

Tasks:

1. Provision an Azure SQL db with sample AdventureWorks db
2. Provision an Azure Blob storage
3. Use Copy data tool to copy the data from Azure SQL db to Blob storage
4. Create the ADF pipeline
5. Monitor the pipeline

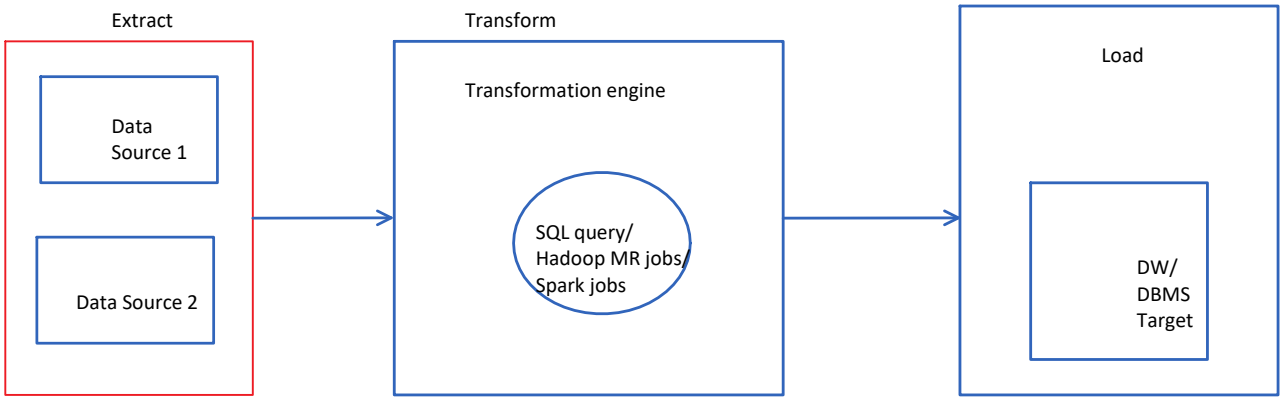
Extract, Transform, and Load (ETL) process

Extract, Transform and load (ETL) is a data pipeline used to collect data from various sources. It then transforms the data according to business rules, and loads the data into the destination data store.

The transformation work in ETL takes place in a specialized engine, which often involves using staging tables to temporary hold data as being transformed and ultimately loaded to its destination.

ETL process in ADF

Load



1. The data transformation which takes place usually involves various operations, such as filtering, sorting, aggregating, joining data, cleaning data, deduplicating data and validating data.

2. Often, these three ETL processes can execute in parallel to save time.

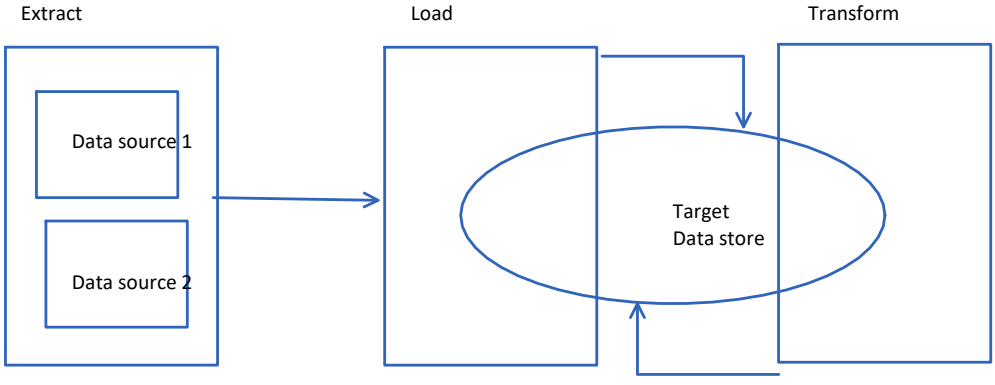
e.g. while the data is being extracted, a transformation process could be working on data already received and prepare it for loading, and a loading process can begin working on the prepared data. Rather than just wating for the entire extraction process to complete.

ELT in ADF pipeline (Extract, Load and Transform)

Extract, load and transform differs from ETL mainly in where the transformation takes place. In the ELT pipeline, the transformation occurs 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.

1. Scaling of the target data store also scales the ELT pipeline performance. However, the ELT only works well then the target system is powerful to transform the data efficiently.



1. In Azure, the source flat files in scalable storage, such as HDFS, Azure blob storage or ADLS gen2 are all can be used as data storage layer for extraction.
2. For Spark, Hive, polybase, SQL query, Pig/latin scripts, MR jobs etc. can be used to query the source data.
3. In 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 step present in ETL which often just can be a 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.

Azure Data Factory (Code free ETL as a service)

1. **Ingest**

- The data can ingest on multi-cloud and on-premise based on hybrid copy data.
- 100+ native connectors
- Serverless and auto scale
- Use wizard for quick copy jobs

2. **Control Flow**

- Design code free data pipelines.
- Generate pipelines via SDK
- Utilize workflow constructs - loops, branches, conditional execution, variables and parameters.

3. **Data Flow**
- Code free data transformations which execute in Spark
- Scaled out (add more instances) in Azure Integration Runtime
- Generate data flows via SDK
- Designers for data engineers and data 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 executions and pipeline history
- Detail activity and data flow execution status
- Establish alerts and notifications

Schema Drift in ADF data flow

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 to the source names.

- In order 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 if hard-coded fields and values.
- Define expressions which can understand patterns to match incoming fields, instead of using named fields.
- Schema drift can be applied for both data source and sinks.

Transform the drifted columns

When the data flow has drifted columns, we can access in 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 , origin or type.**
- **Add rule-based mapping in a Select or Sink transformation to match drifted columns to columns aliases via the pattern.**

Control Flow Activity

Control Flow activities in ADF involves 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 also how control / sequence activities can pass through from one task to another.

Features:

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

Examples:

1. **Lookup Activity in ADF**

Lookup activity can retrieve a dataset from any of the data sources supported by data factory. We can use it to dynamically determine which objects to operate on in a subsequent activity, instead of hard coding the object name, some objects examples like files and tables.

Lookup activity reads and returns the content of a configuration file or table.

- It also returns the result of executing a query or stored procedure.
- The output can be a singleton value or an array of attributes, which can be consumed in a subsequent copy, transformation or control flow activities like ForEach activity.

Features of Lookup activity

1. The Lookup activity can return upto 5000 rows, if the result set contains more records, 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.

Type properties in Lookup Activities

a)Dataset - provides the dataset reference for the lookup. Get details from the dataset properties section in each of the corresponding connector,

b) Source - Contains the dataset-specific properties. The same as the Copy activity source.

c) firstRowOnly - indicates whether to return only the first row or all rows .

2. **Foreach Activity** - It's a control flow activity type 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.

The loop implementation for the activity is similar to Foreach looping structure in programming.

Type properties

- Name
- Type
- isSequential (specifies whether the loop should be executed sequentially or in parallel. Max of 50 loop iterations can be executed at once in parallel.) If we have a ForEach activity iterating over a copy activity with 10 different source and sink dataset, with **isSequential** set to false, all copies are executed at once.

If 'isSequential' is set to false, ensure it that there's a correct configuration is being executed to run multiple executables. Otherwise, this property should be used with caution to avoid incurring write conflicts.

- batchCount
- Items - expression which returns a JSON array to be iterated over
- Activities - the activities are to executed.

Hands-on Lab 3

1. Create the ADF
2. Create the ADLS gen2 storage
3. Create Azure SQL db (with AdventureWorks db)
4. Providing the Linked Service connection through managed identity from ADLS to ADF
5. Copy the data through Copy data activity in ADF pipeline
6. Test, Validate and Monitor the pipeline

Hands-on Lab 4:

Mapping Data Flow

Hands-on Lab 5:

Move/Copy data through ADF from on-premise SQL Server database to Azure Blob storage

Pre-requisites - SQL authentication is required to be enabled for on-premise SQL server database

Hands-on lab 6:

Building a reusable pipeline in ADF

Hands-on Lab 7:

Monitoring, validating the ADF pipeline

3. 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.

4. Execute Pipeline Activity - This activity allows the data factory to invoke another pipeline.

property

Name	Name of the execute pipeline activity
Type	Must be set to : Execute Pipeline
pipeline	Pipeline reference to the dependent pipeline which the pipeline invokes. A pipeline reference object has two properties, a) referenceName - the referenceName property specifies the name of the reference pipeline b) Type - the type property should be set to pipeline reference.
parameters	Parameters to be passed to that invoked pipeline
waitOnCompletion	Defines whether activity execution waits for the dependent pipeline execution to finish. Default is true.

5. The Custom Activity - There are two types of activities in ADF, to use in Data Factory.

- Data movement activities to move data between the supported data source and sink data source
- Data transformation activities to transform the data using compute services (such as Azure HDInsight)
- To move the data to/from the data store which the service does not support, or to transform/process data in a way 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.

Overview of Azure Data Factory Control Flow Activities

Control Flow Activity Name	Purpose / Definition
Append Variable	Append variable activity could be used to add a value to an existing array variable defined in ADF pipeline
Set variable	Set variable activity can be used to set the value of an existing variable of type string, bool, or array defined in a ADF pipeline
Execute Pipeline	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	The activity defines the repeating control flow in the ADF pipeline. This activity is used to iterate over a collection and executes 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 of the 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 condition associated with the 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 the Azure Function in the ADF pipeline
Validation Activity	We can use the Validation 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 pipelines through the custom code. With the webhook activity, code can call an endpoint and pass it a callback URL. The pipeline run waits for the callback invocation before it proceeds to the next activity.

Webhooks are automated messages sent from apps when something happens. Webhooks have a message, or payload and sent to the unique URL,

- We've to get the webhook URL from the application to send the data to.
- Use the URL in the webhook section of the application we want to retrieve the data from
- Choose the type of events we want the application to notify about

Hands-on Lab 4:

Pre-requisites

1. **Create the ADF resource**
2. **Create the Azure SQL db (with sample AdventureWorks db) (input data source)**
3. **Create the ADLS gen2 storage (data sink)**

4. **Create the linked services**

4.1. Linked Service for Azure SQL db
4.2. Linked Service for ADLS gen2

5. **Create the datasets**

5.1. ADLS dataset

5.4. Azure SQL for product table
5.5 Azure SQL for ProductCategory table

6. **Three pipelines**

6.1. copy_product
6.2 copy_productCategory

1. Start the Lab 4 for copy data and use the data flows.

Mapping Data Flow Activities

1. Mapping Data flow activities are visually designed data transformations in Azure Data Factory. Data Flows allow data engineers to develop data transformation logic without writing code.

2. The resulting data flows are executed as activities within ADF pipelines which use scaled-out Apache Spark cluster, Data flow activities can be operationalized using existing Azure Data Factory scheduling, control flow and monitoring capabilities.

3. Mapping data flows provide an entirely visual experience with no coding required. The data flows run on ADF-managed clusters for scaled out data processing. Azure Data factory handles all the code translation, path optimization and execution of the data flow jobs.

Data flow data types

1. Array
2. Binary
3. Boolean
4. Complex
5. Decimal (includes precision)
6. Date

7. Float
8. Integer
9. Long
10. Map
11. Short
12. String
13. Timestamp

Mapping data flow allows to interactively see the results of each transformation step applied while building and debugging the data flows, the debug session can be used both in when building the data flow logic and running pipeline debug runs with data flow activities.

Schema Drift

Schema drift allows us to define data sources which should have mutable field names, data types, values and sizes

- Define transformation parameters which can work with data patterns instead of hard-coded fields and values
- Define expressions which understand patterns to match incoming fields, instead of using naming fields.

  Azure Data Factory natively supports flexible schema which change from execution to execution which we can use to build generic data transformation logic without the need to recompile the data flows.

- There's an architectural decision in the data flow to accept the schema drift throughout the data flow.
- With schema drift, can protect against the schema changes from the sources.
- Through ADF, it treats the schema drift flows as late-binding flows, so while building the transformations, the drifted column names aren't being available in the schema views throughout the flow.
- Columns which are coming to the data flow from the source definition are defined as 'drifted' when they're not present in the source projection.
- We can view the source projection from the projection tab in the source transformation.
- when we select a dataset from the source, the service will automatically take the schema from the dataset and create the projection from the dataset schema definition.
- When the schema drift is enabled, all incoming fields are read from the source during execution and passed through the entire flow to the Sink.
- By default, all newly detected columns, known as drifted columns which arrive as a string data type, of we wish the data flow to automatically infer data types of drifted columns , check the drifted column types in the source settings.

Transforming drifted columns

When the data flow has the drifted columns, we can access it in the transformations with the method,

- 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 on any combination of name, stream, position, origin or type.
- Add rule based mapping in a select or sink transformation to match drifted columns to columns aliases via a pattern.

Integration Runtime in ADF:

Integration runtime provides the compute capacity the ADF resources. Cpu, memory and disk utility for the infrastructure compute unit within the ADF integration runtime.

1. Azure IR (AutoResolveIntegrationRuntime) - all of the data copy, data movement, core Azure data transformation can be executed.

2. Self-hosted IR (Self-hosted Integration Runtime) - whenever, on-premise data sources or data sinks are required to be orchestrated in the ADF pipeline.

e.g. data copy from on-premise SQL db to Azure SQL db
Data copy from Azure Blob storage to on-premise SQL db

3. Azure - SSIS - used while transformation, copying data from on-premise SSIS db to Azure SQL db.

Hands-on Lab 5:

Create reusable pipeline in ADF

Pre-requisites

1. Create the ADF resource
2. Create Azure SQL db (with AdventureWorks sample db)
3. Create Azure Data Lake Storage (ADLS Gen2)
4. Create Linked Service for Azure SQL db and ADLs gen2

Tasks:

4. Create data flows using dynamic contents with reusable datasets
5. Explore the options to build the ADF reusable pipeline with dynamic contents to retrieve the data from SQL dataset (10 different tables) and sink to ADLS storage.
6. Used the generic pipeline to extract ten different Azure SQL tables with a single pipeline containing only three activities (Lookup, Foreach -> Execute Pipeline activity)

Azure

Azure Data Lake Gen2

12 December 2022 19:56

Azure Data Lake Storage Gen2 is the extended modified version of Azure Blob storage which includes the core capabilities of Azure Blob + File share. It includes the driver as abfss:// (azure blob file share driver) , for azure blob storage, the driver is referred as wasb:// (windows azure blob storage).

While provisioning ADLS gen2, the hierarchical namespace is to be enabled.

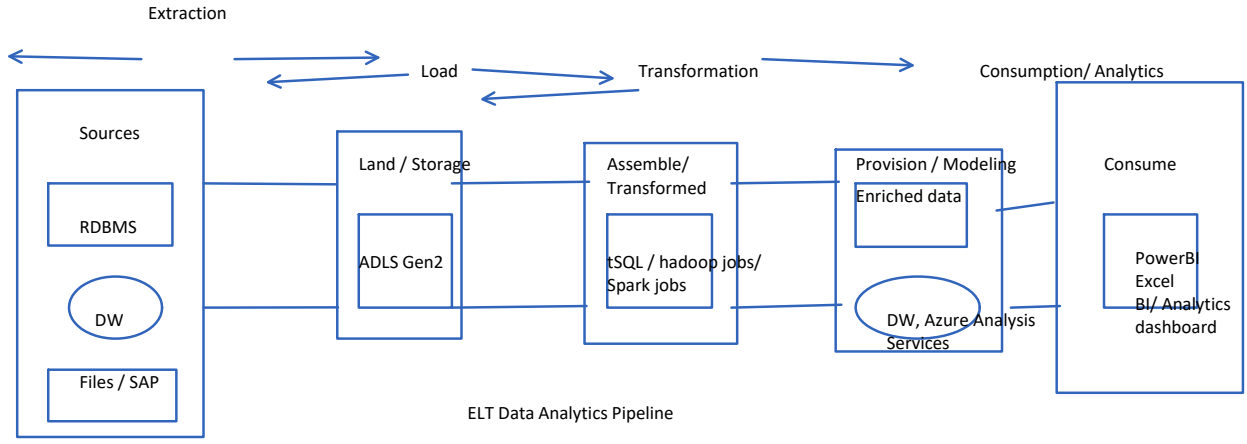
- The Data Lake Storage gen2 converges the capabilities of Big Data Analytics which is built on Azure blob storage.
- The Azure Data Lake storage includes better access control (RBAC), access control policies (ACL) as part of security compared to Azure blob storage.
- Azure data lake storage gen2 also includes the support of file system semantics, file-level security and scale.
- For all of these capabilities built on Blob storage, we can get low cost, tiered storage with high availability / DR capability.

Comparison between ADLS gen2 and Blob storage

Feature	ADLS gen2	Azure Blob storage
Performance	Performance in ADLS gen2 is optimized since we don't need to copy or transform data as a pre-requisite for analysis. Compared to the flat namespace on blob storage, the hierarchical namespace on ADLS gen2 offers the performance improvement of directory management operations, which overall improves the job performance.	Performance is not optimized compared ADLS gen2 since it consists of flat namespace of object based storage
Management	Management of ADLS gen2 is easier because we can organize and manipulate files through directories and subdirectories	Management is complex compared to ADLs gen2, files are managed through blob containers
Security	Security is enhanced since we can define POSIX permissions on directories or individual files	Security is less enforceable compared to ADLS gen2 due to lack of support of file system permissions, file share capabilities

A Data Lake is a repository which stores all of the organization's data - both structured and unstructured. The Data lake as a massive storage pool for data in its natural raw state (like a lake).

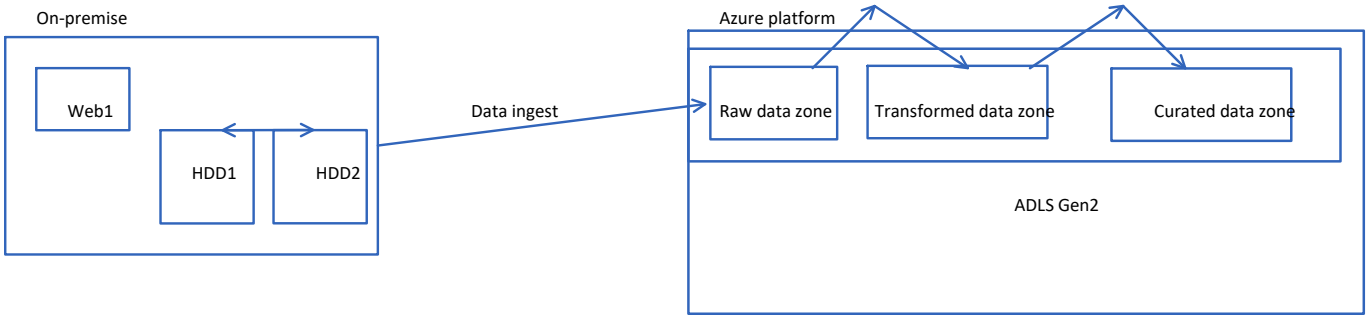
A data lake architecture can handle the huge volumes of data that most organizations produce without the need to structure it first. Data stored in a data lake can be used to build data pipelines to make it available for data analytics to ols to find insights which informs the key business decisions.



Data Lake Benefits

- Because the large volumes of data in a data lake are not structured before being stored, skilled data scientists or data engineers can perform end to end data analysis using self-service BI tools to gain access to the broader range of data much faster than in a data warehouse.

1. Massive volumes of structured and unstructured data like transactions, sales data, logs can be stored cost effectively.
2. Data is available for use far faster by keeping in a raw state
3. A broader range of data can be analysed in new ways to gain unexpected and previously unavailable insights.

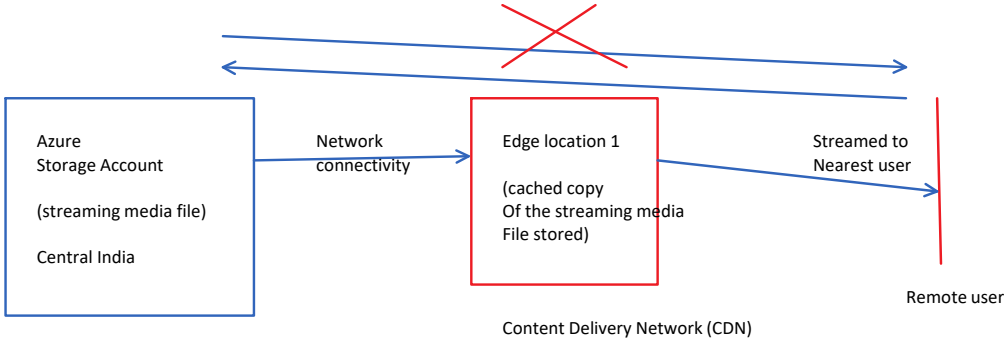


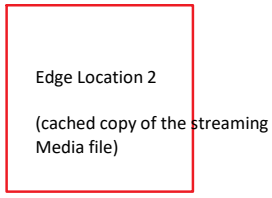
The layers of Azure Data Lake

1. Raw data zone - which is responsible for the storage of raw data irrespective of file size, formats from on-premise data sources to Azure data lake storage.

2. Transformed data zone - this layer in ADLS is responsible to store transformed data through SQL queries, hadoop/spark jobs.

3. Curated data zone - this layer in ADLS is responsible to store the final curated data which can be made available to consumption layer for BI/ Analytics purpose.





- Streaming media files broadcasting
- Static websites accessibility
-
-

We use Azure CDN with blob storage to deliver high quality content to the remote users through Edge location/ zone.

ADLS Gen1 & Gen2 Difference

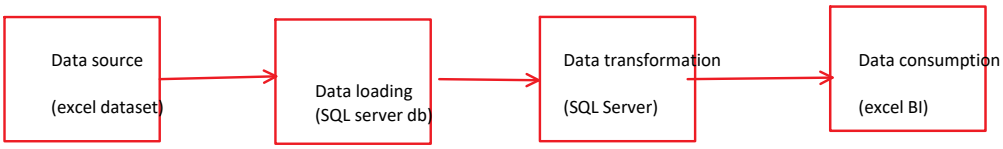
1. ADLS gen2 is the superset of ADLS gen1 including the support of blob storage , tiers , HDFS and object store APIS.
2. ADLS Gen2 also extends Azure blob storage capabilities and it's best optimized for big data analytics workloads. It includes HDFS compatible file system interfaces and data copy support.

Sprint 1 Presentation

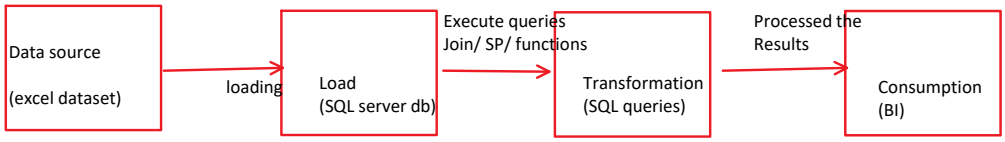
1. **Powerpoint slide deck** (your use case topic, business functionality, technical details (design, pre-requisites tools/services, screenshots of the results of sql queries of the analytical queries) (4-5 slides) (15-20 mins)
2. **Project document**
   - introduction
   - business process
   - technical process
   - high level design (HLD)
   - low level design (LLD)
   -- technical component / resource details (excel , SQL server db, python, pandas, SSMS, azure storage, azure data factory, azure sql db etc.)
   -- end to end implementation process (data cleaning, parsing, loading & transformation details)

3. Conclusion

High level design



Low level design



1. SQL Server Analysis Services
2. Visual Studio community edition 2019/ 2017
3. SQL Server data tools (SSDT)

Azure SQL Database

12 December 2022 19:57

SQL Server on Azure VM	Azure SQL Managed Instance	Azure SQL database
Simple migration of application Where the core SQL server on-prem features are required. (e.g. .net framework runtime, CLR, SQL Server Agent, SQL Server broker, log shipping) etc.	We can get all the core SQL db features like on-premise SQL database but it's a managed service. We don't have to configure the underlying hardware/infrastructure compute, storage, networking for the Azure SQL Managed instance.	Purely managed SQL database offering where no license is required. No underlying administration of server, sql database, network is required. End to end database migration is feasible with a just a click focusing into the core SQL development features.
Includes the benefits of distributed databases, distributed transactions, advanced data types - spatial, geography, geometry, SQL Server broker etc.	It's includes the benefits of SQL Server on Azure VM as well as includes the benefits of Azure SQL database.	Azure SQL database is a managed database service which does not contain the support of distributed transaction, distributed databases, advanced data types - spatial, geography, geometry , broker etc.
License is required, either through BYOL, License has to be procured	No license required, entire SQL db on-premise feature can be availed with the managed platform service.	No license is required, only SQL db development specific features are available. There's also a limitation on the database size (5 TB -> 100 TB)
SQL Server Analysis Service (SSAS), SQL Server Reporting Services (SSRS), SQL Server Integration Services (SSIS), SQL server broker, .net framework runtime, CLR integration, distributed transactions, database mail etc. All these features are supported.	.net framework, CLR related SQL functions, distributed transactions, ACID semantics, automated backups, high availability etc. all are supported along with no administration requirement.	No support for Microsoft .Net framework runtime, distributed transaction , ACID (atomicity, consistency, isolation, durability), SQL Server broker, CLR related SQL functions, distributed stored procedures etc based on Windows runtime are not yet supported.
SLA - 99.9% guaranteed SLA. Automated backups, Azure Site recovery for disk level backup of database	SLA - 99.99%, 99.95%, automated backups, point-in time restore, geo-replication, high availability, automated patching	SLA - 99.99% of high availability, automated backups, active geo-replication, disaster recovery benefits. The data replication benefits are available on both availability zones and region basis.

1. Application Migration - The scenario evolves when the application is moved to Azure platform.

e.g. asp.net mvc application can be migrated to Azure Web app

2. Database Migration- The scenario involves when the relational database (RDBMS) is migrated to Azure platform.

e.g. on-premise SQL server db is migrated to Azure SQL db.

Azure SQL Database

1. Azure SQL database is a fully managed platform as service (PaaS) database engine handles most of the database management functions such as upgrading, patching, backups and monitoring without the user involvement.

2. Azure SQL database is always running on the latest stable version of the SQL server database engine and patched OS with 99.99% availability.

3. PaaS capabilities can built into Azure SQL database enable us to focus on the domain specific database administration and optimization activities which are critical for the business.

4. With Azure SQL database, we can create a highly available and highly performance data storage layer for the applications and solutions in Azure. SQL database is the right choice for a variety of modern apps since it enables to process both relational data and non-relational structures like graphs, JSON, spatial and XML data.

5. Azure SQL database is based on the latest stable version of the Microsoft SQL server database engine. We can use the advanced query processing features like high-performance in-memory technologies and intelligent query processing.

6. SQL database also enables us to define and scale performance within two different purchasing

model
- vCore-based purchasing model
- DTU-based purchasing model

SQL database is the fully managed service which has built-in high availability, backups and other common maintenance operations. Microsoft handles all the patching and updating of the SQL and OS code. As a customer, we don't have to manage the underlying infrastructure.

RPO = Recovery Point Objective

The amount of data loss can be tolerated during the regional failure

RTO = Recovery Time Objective

The amount of time it should take to recover the database in the secondary region

Deployment Models

There are two types of deployment models of Azure SQL database.

1. Single database - represents a fully managed, isolated database. We can use this Azure SQL single database if we can have the modern cloud applications and microservices which need a single reliable data source. A single database is similar to a contained database in the SQL server database engine.

2. Elastic Pool - Elastic pool is a collection of single databases with a shared set of resources, such as CPU or memory. Single databases can be moved into and out of an elastic pool.

Purchasing Models

SQL database offers the following purchasing model.

a) **vCore based purchasing model** - The lets us to choose the number of vCore, the amount of memory, the amount and speed of storage. The vCore - based purchasing model also allows us to use Azure Hybrid benefit (AHB) for SQL server.

b) **The DTU based purchasing model** - offers a blend of compute, memory and I/O resources to three service tiers, to support light to heavy database workloads.

Compute sizes within each tier provide a different mix of these resources for which we can add these additional storage resources.

Service Tiers

The v-core & DTU based purchasing model offers three service tiers.

a) **General-purpose / Standard service tier** - It allows us to choose the number of vCores, the amount of memory, the amount and speed of storage. Service tier is designed for common workloads, it offers the budget oriented balanced compute and storage options.

b) **Business critical / premium service tier** - service tier is designed for OLTP applications with high transaction rates and low latency I/O requirements. It offers the highest resilience to failures by using several isolated replicas.

c) **The hyperscale tier** - it's designed for most business workloads, hyperscale provides great flexibility and high performance with independently scalable compute and storage resources. It also offers higher resilience to failures by allowing configurations of more than one isolated db replica.

Hands-on Lab 01:

1. Create The Azure Resource Group
2. Create Azure SQL server
3. Create a single Azure SQL database
4. Create an elastic pool of SQL database

Azure SQL Elastic Pool database

Azure SQL elastic pool database are simple, cost-effective solution for managing, scaling multiple databases which has varying and unpredictable usage demands. The databases in an elastic pool are on a single server and share a set number of resources at a set price.

Elastic pools in the SQL database enable developers to optimize the price performance for a group of databases within a prescribed budget while delivering performance elasticity for each database.

Definition of SQL elastic pool

Elastic pools provide a simple resource allocation mechanism within a predictable budget. Elastic pools enable us to purchase resources for a pool shared by multiple databases to accommodate unpredictable periods of usage by individual databases. We can configure the resources for the elastic pool based on either the DTU-based purchasing model or the v-Core based purchasing model.

1. Add databases to the sql elastic pool
2. Optionally we can set the minimum and maximum resources for the databases.
3. We can set the resources of the pool based on the budget

Within the pool, individual databases are given the flexibility to use resources within set parameters, Under heavy load, a database can consume more resources to meet demand. Databases under light loads, can consume less, and databases under no load consume no resources.
Henceforth, provisioning resources for the entire pool rather than for a single databases simplifies the resource management tasks, also, there should be a predictable budget for the pool.

Scenario to consider SQL elastic pool

Pools are well suited for the large number of databases with the specific utilization patterns. For a given 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 can be added to the pool, the greater the cost effective. Depending on the application utilization pattern, it's possible to see the cost effectiveness with two or multiple S3 databases.

Correct Pool size for Azure SQL elastic pool db

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

Estimate whether a pool is merely cost-effective than a single database

1. **For the DTU-based purchasing model**

MAX(<total number of DBs X Average DTU utilization per DB>, <Number of concurrently peaking dbs X Peak DTU utilization per DB>)

2. **For the vCore based purchasing model**

MAX(<Total number of DBs X Average vCore utilization per DB>, <Number of concurrently peaking db X Peak vCore utilization per DB>)

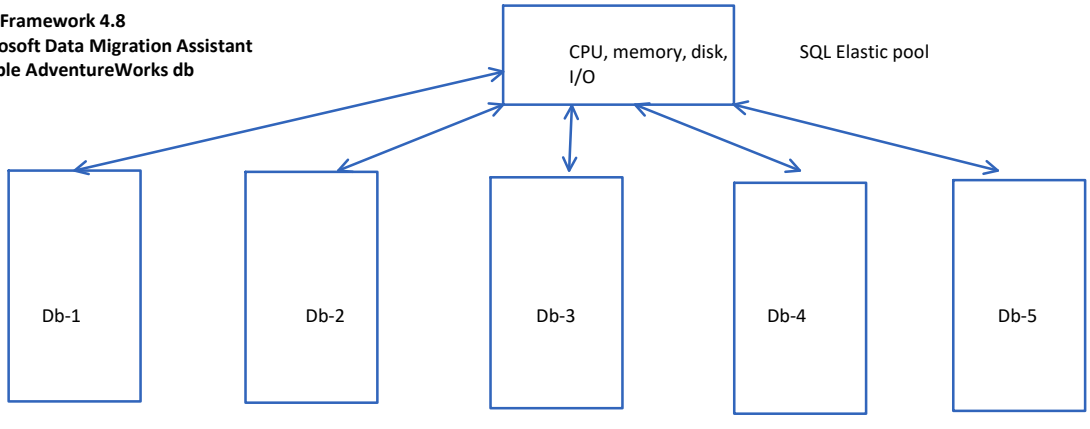
Hands-on Lab 02

1. Provision a new deployment of Azure SQL db
2. Configuring Security on Azure SQL db - Auditing, ledger, data discovery and classification, data masking of sensitive data fields & TDE
3. Monitoring of Azure SQL db - metrics & build visuals on CPU usage, memory and disk space

Hands-on lab 03

1. Provision an Azure SQL db (blank database)
2. Provisioning the SQL server on Azure VM database
3. Migrating the database from SQL Server on Azure VM (on-premise SQL db) to Azure SQL db using Azure Database Migration tool

1. **.Net Framework 4.8**
2. **Microsoft Data Migration Assistant**
3. **Sample AdventureWorks db**



1. Core infrastructure compute, network, storage allocated within the elastic pool of the Azure SQL elastic pool resource.
2. The elastic pool dbs share their cpu, memory, disk, I/O, from the core elastic pool.

DTU - Database transaction Unit (Core infrastructure compute unit allocated for the Azure SQL db)
e-DTU - elastic database transaction unit (Core infrastructure compute unit allocated for Azure SQL elastic db)

Data Migration Tool

Assessment of the on-premise sql server database and schemas, tables, objects and assists for the migration to Azure sql db.

- To migrate the on-premise SQL server database to Azure SQL db
- To migrate the SQL server on Azure VM db to Azure SQL db
- To migrate AWS EC2 SQL db to Azure SQL db
- To migrate AWS RDS SQL db to Azure SQL

Data Protection

1. **Encryption of data in Azure SQL at rest** - while the data stored in Azure SQL db, it's being encrypted. So that, no unintended user should be able to access the data.

2. **Encryption of data in transit for Azure SQL** - while the data is traversing over the network i.e. from the application layer to database layer & vice versa, so the end to end data connectivity should be encrypted. So that, no malicious user can tamper the data / steal the data.

Azure Blob Storage

12 December 2022 19:57

Benefits of Azure Storage

1. Durable and highly available - Redundancy ensures that the data is safe in the event of transient hardware failure. We can opt for LRS, GRS or ZRS kind of redundancy options for zonal or regional resiliency.

2. Secure - All data written to Azure storage is encrypted by the service. Azure storage provides fine grained access control over who has access to the data.

3. Scalable - Azure storage is designed to be massively scalable to meet the data storage and performance needs for today's applications.

4. Accessible - Data in Azure Storage is accessible from anywhere over HTTP/ HTTPS. Microsoft Azure provides client libraries for Azure storage in a variety of languages including java, C#, Python, Node.js, Ruby etc.

Azure Storage Services

Azure Blobs	A massively scalable object store for text and binary data. Also, it includes support for big data analytics workloads through ADLS gen2.
Azure Files	Managed file share for cloud and on-premise deployments
Azure Queues	A messaging store for reliable messaging between application components.
Azure Tables	A NoSQL store for schemaless storage for structured data
Azure Disks (Azure Page blobs)	Block level storage (Page blob) for randomized read-write access which stores disk space volumes (vhd, vhdx) files over Azure storage.

Block Blob	Page Blob	Append Blob
Store text and binary data file, media files etc. Block blobs are made up of blocks of data which can be managed Individually. Block blobs can store upto 190.7 TB	Page blobs are store virtual hard drives & serve the disks for the Azure VMs. It can store & access files up to 8 TB in size.	Made up of blocks like block blobs , but are optimized for append operations. Append blobs are ideal for logging data from Azure VMs.

Access Tiers for Blob

Tier	Accessibility	Cost
Hot	We can access the data files from hot access tier In just milliseconds of time (1-2 sec)	Most costlier data storage solution. Data read/accessibility cost is least compared to cool / archive tier.
Cool	We can store the data files in Cool access tier for more than a month of few months. Accessibility of data takes around few mins of time. (5-10 mins). Subsequent latency is observed while retrieving the data.	Less costlier than hot access tier
Archive	We can store the data files from 3 months to 6 months to few years. High latency is observed in case of data retrieval time. (around few hours Depending the data volumes (TB/PB)	Least cost compared to hot and cool access tier . For Archive access tier, the data read cost is highest compared to Hot / Cool tier

Hands-on Lab 02

1. Create Azure File share
2. Create an Azure VM (Windows)
3. Mount the file share on the Windows local drive

Benefits of Azure File share

1. Replace or supplement on-premise file servers
2. Lift - shift applications
3. Simplify the cloud development
4. Managed file share support
5. Resiliency
6. Scripting and tooling support

Azure Analysis Services

12 December 2022 19:57

The SQL Server Analysis Services (SSAS) is the analytics services engine for SQL Server.

There are two kinds of models available for SQL Server Analysis Services.

a) Tabular Model
b) Multidimensional model

Tabular Model (SSAS)

Tabular model of SQL Server Analysis Service was introduced with SQL Server 2012. Tabular model in SQL Server uses a different engine (xVelocity), which is designed to build queries much faster based on columns, because it uses the columnar storage in addition to better data compression.

In Tabular model, the data is stored in-memory, it's very important to have a lot of memory allocated for the server and for the faster data processing of queries as very fast CPU.

The disks are not important in tabular model.

Multi-dimensional model (SSAS)

The multi-dimensional model is very different structure than a relational database and allows us to generate reports very fast. The multi-dimensional model was the only solution used in the past to create multi-dimensional databases.

The multi-dimensional model are supported in SSAS. It's useful to build OLAP modelling structure like (cubes, dimensions and measures).

The multi-dimensional model is also supported to import the data from external sources.

The amount and the type of data imported which is required can be a primary consideration when deciding which model is best suited for the data.

Feature	Tabular Model	Multi-dimensional Model
Hardware	It's very important to clarify, the hardware used for multi-dimensional model can't be used for tabular model. Tabular model is a memory dependent solution. The tabular model is optimized with more memory, the better performance can be achieved with more memory. Without sufficient RAM, tabular model will fail. CPU speed is also very important for tabular database	For multi-dimensional model, where the database requires a lot of disk space like > 5 TB. There we've to build multi-dimensional model.
Advantage	Tabular model is faster for some queries and compresses the data even more than the multi-dimensional solutions	Multi-dimensional model is not that much faster unlike tabular model
Data Import	Tabular model can import data from relational data sources, data feeds, some document formats etc. We can use OLE db, ODBC providers for building tabular model.	Multi-dimensional model, it can import data from relational data sources using OLE db native and managed drivers.
Data sources	SQL Server relational db, access databases, Oracle db, Teradata, Sybase, DB2	Most of the similar data sources through OLE db provider, .net framework data providers, OLEDB providers etc.
Query	DAX calculations, DAX queries, MDX queries	Supports the MDX queries, supports and calculates MDX queries, DAX queries etc.
PowerShell Support	SQL Server Analysis Services, powershell cmdlets are supported for both tabular and multi-dimensional model and databases.	

Azure Analysis Service is a fully managed platform as service (PaaS) which provides the enterprise - grade data models in the cloud. Use advanced mashup and modelling features to combine data from multiple data sources, define the metrics, secure the data in a single, trusted tabular semantics data model. The data model provides an easier and faster way for users to perform ad hoc data analysis using the PowerBI and excel.

There're three tiers are available for Azure Analysis Services

1. Developer tier

(Recommended for evaluation, dev and testing purpose, a single plan includes the same functionality of the standard tier, but it's not limited in processing power, query

processing unit (QPU) and memory size).

-- Limitations (Query Replica scale out is not available in this tier, this tier doesn't offer an SLA).

2. Basic tier

This tier is recommended for production , the solutions with smaller tabular models, limited user concurrency and simple data refresh requirements. Query replica can scale out & it's not available in the basic tier.

(Perspective, Partitions, and direct Query is not supported)

3. Standard tier

This tier is recommended for business critical and mission critical purpose, for applications which require elastic user-concurrency and have rapidly growing data models, it supports the advanced data refresh for near real time model updates and supports all tabular model features.

Hands-on Lab 01

Create Azure Analysis Service Server

1. **Add Azure Analysis services resource provider in the Azure subscription**
2. **Create Azure Analysis Service Server**
3. **Configure the Analysis Services Server Firewall**
4. **Add a sample Analysis Services Model from the portal**
5. **Create Azure AD User account and add as Azure Analysis Services Server Admin**
6. **Add an Azure AD user account to the Server Admin Role from SSMS**

Scale out (Flexible Scaling) resources in Azure Analysis Services for faster query response

With the scale out approach, client libraries are distributed among the multiple query replicas in a query pool, query replicas have synchronized copies of the tabular models. By spreading the query workload, response times during the high query workloads can be reduced. Model processing operations can be separated from the query pool, ensuring the client queries are not adversely affected by the query processing.

We can create the query pool with upto seven query pool replica, the number of query replica that can include in the pool depends on the chosen plan and region. Query replicas cant be spread outside of the server's region, whereas the query replicas are billed the same rate of the server.

Tabular Model Core Features in Azure / SQL Server Analysis Services (SSAS)

a) Tabular models in both in-memory and direct query modes are supported. In-memory mode tabular model is supported by multiple data sources, because the model data is highly compressed and cached in-memory. This mode provides the fastest query response over large amounts of data. It also provides the greatest flexibility for complex datasets and queries.

b) Partitioning enables incremental loads, increases the parallelization and reduces memory consumption with other data advanced data modelling features like calculated tables, fields, columns, measures and all DAX functions are supported.

c) In-memory models must be refreshed or processed to update cached data from data sources.

d) Tabular model also supports the direct query mode, it leverages the relational db for storage and query execution purpose. The extremely large datasets like SQL Server, SQL Server DW, Azure SQL db, Azure Synapse Analytics, Oracle db, Teradata etc.. Various other data sources being supported.

e) Complex data models refresh scenarios in Azure Analysis Services (AAS) tabular model direct query mode is not required. Exceptions like limited data source types, DAX formula limitations or unsupported features.

DAX - Data Analysis Expression is a formula language used to create custom calculations in Azure Analysis Services, SSAS, PBI, powerpivot in excel. DAX formula can include functions, operators, values to perform advanced calculations on the data in tables and columns.

- Calculated columns - is a column which we can add to an existing table & then can create a DAX formula which defines the column's value.

- Calculated tables - computed objects based on the DAX query or expressions and it can be derived from a full part of other tables in the same model.

- Measures - Measures are the dynamic formulas where the results can change depending on the context. Measures are used in reporting formats which support combining and filtering model data by using multiple attributes such as PowerBI report, excel powerpivot or PowerBI dashboard.

Total Sales = SUM([Sales Amount])

A measure is simply a calculation using DAX formula. But however, unlike, calculated columns, measures are evaluated based on selected filter.

For example - a particular column or slicer added to the Row labels field in a pivottable, a value for each cell in the filter is then calculated by the applied measure.

Measures are powerful, flexible calculations which we can include in almost every tabular models to perform dynamic calculations on numerical data.

A formula in a measure can be used to implement standard aggregation like COUNT, SUM or even custom formula in DAX.

- Row Filters - defines which rows in a table should be visible to the members of the particular role/attribute. It is being evaluated which rows can be returned by the results of a query by members of a particular role. In DAX, the row filters are evaluated as true/false.

  For writing Measures in tabular model.

  Unlike calculated columns, with measure formulas we can type by the measure name, followed by a colon and then followed by the formula expression.

  Perspective

  A perspective in tabular model defines a viewable subset of a model which provides the focused, business-specific, or application specific viewpoints.
  When a user connects to a model by using a perspective, they see only model objects (tables, columns, measures, hierarchies and KPIs) as fields defined in that perspective.

  Hierarchies

  Hierarchies are the group of columns arranged in levels. For example, a Geography hierarchy means it includes the sublevels of Country, state, District and City.

  Hands-on Lab 02

  Tabular Model Development & Deployment to AAS/SSAS Services

  Goals-

1. Create a new tabular model project
2. How to import data from SQL Server db/ Azure SQL DW db into the tabular model project
3. Mark the **DimDate** table as date table
4. Rename the column of specific table (dimDate) table.
5. Set the Mark as Date table
6. Create and manage relationships between the tables in the model (PK/FK constraints)
7. Create and manage the calculated columns using DAX
8. Measures
9. KPIs (Knowledge performance Indicators) for analysis
10. Create the hierarchies
11. Create the partitions dividing the table data into the smaller logical parts.
12. Apply security model objects
13. Deploy the tabular model in AAS.

Pre-requisites of the lab

1. Visual Studio 2017/2019
2. Compatible SQL Server DB engine (2017/2019)
3. SSMS
4. AdventureWorksDW edition
5. Azure SQL Data warehouse instance with sample AdventureWorks DW db
6. PBI desktop
7. An Azure Analysis Service instance (Azure subscription) , SSAS

Tasks for today as pre-requisites for hands-on lab 02

1. Provision the Azure VM (with the same SKU as previous)
2. Provision the Azure SQL DW with sample adventureworks db
3. Login into the Azure VM & then install Visual Studio
4. Create the tabular model project with analysis service extension

https://developerinsider.co/download-visual-studio-2019-web-installer-iso-community-professional-enterprise/#downloadvisualstudio2019iso

Links:

https://developerinsider.co/download-visual-studio-2019-web-installer-iso-community-professional-enterprise/#downloadvisualstudio2019iso

https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16

https://download.microsoft.com/download/4/8/6/486005eb-7aa8-4128-aac0-6569782b37b0/SQL2019-SSEI-Eval.exe

Steps for this lab

1. Provision an Azure VM (sufficient memory required)
2. Provision SQL Server 2017 with SSDT
3. Install SSMS and PBI desktop
4. Download the AdventureWorks DW db
5. Visual Studio community edition 2017/2019.

DimCustomer, DimDate, DimGeography, DimProduct, DimProductCategory, DimProductSubCategory, FactInternetSales

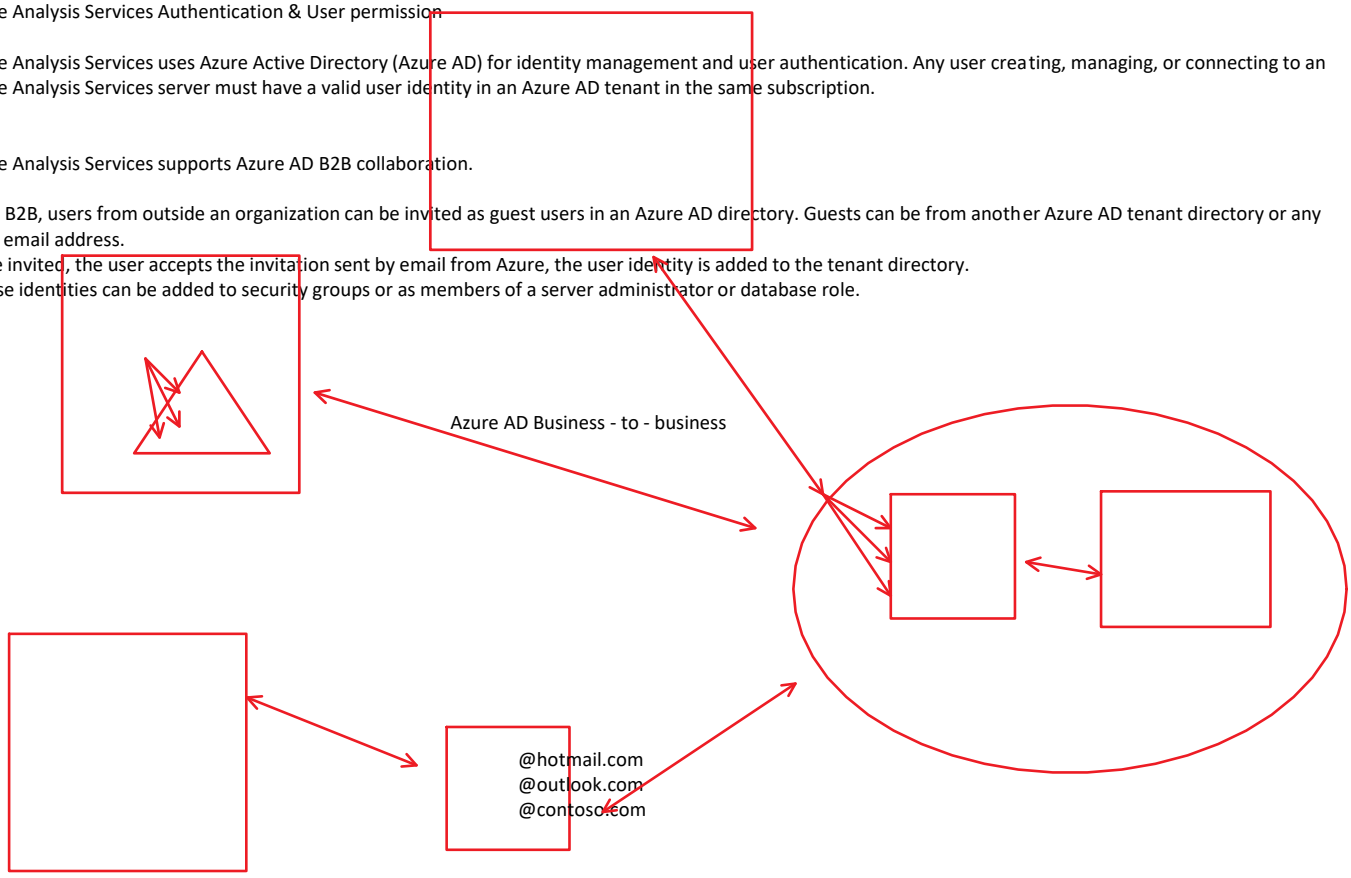
DimCustomer - Remove Columns - SpanishEducation, FrenchEducation, SpanishOccupation, FrenchEducation
DimDate - Remove Columns - SpanishDayNameOfWeek, FrenchDayNameOfWeek, SpanishMonthName, FrenchMonthName
DimGeoGraphy - Remove Columns - SpanishCountryRegionName, FrenchCountryRegionName
DimProduct - Remove Columns - SpanishProductName, FrenchProductName, FrenchDescription, ChineseDescription, ArabicDescription, HebrewDescription, ThaiDescription, GermanDescription, JapaneseDescription, TurkishDescription
DimProductCategory - SpanishProductCategoryName, FrenchProductCategoryName
DimProductSubCategory - SpanishProductSubCategoryName, FrenchProductSubCategoryName
FactInternetSales - no need to remove any columns

Azure Analysis Services Authentication & User permission

Azure Analysis Services uses Azure Active Directory (Azure AD) for identity management and user authentication. Any user creating, managing, or connecting to an Azure Analysis Services server must have a valid user identity in an Azure AD tenant in the same subscription.

Azure Analysis Services supports Azure AD B2B collaboration.

- With B2B, users from outside an organization can be invited as guest users in an Azure AD directory. Guests can be from another Azure AD tenant directory or any valid email address.
- Once invited, the user accepts the invitation sent by email from Azure, the user identity is added to the tenant directory.
- Those identities can be added to security groups or as members of a server administrator or database role.



On-premise AD

User1

User2

User3

ADFS

Your organization AD tenant

User1

User2
User3

Azure AD

Azure Analysis Service

Tools and Client apps

- SSMS
- PowerBI Desktop
- Excel
- Custom

ADOMD
MSOLAP

Authentication Feature of Azure Analysis Services

1. All client apps and tools use one or more Analysis services client libraries (AMO, MSOLAP, ADOMD) to connect to the server.

2. All these client libraries support both Azure AD interactive flow , non-interactive authentication methods. Two non-interactive methods - AD password and AD integrated authentication methods can be used in applications utilizing AMOMD and MSOLAP.

3. Client apps like excel, PowerBI desktop and tools like SSMS and analysis services extension for Visual Studio can install the latest version of the client libraries with regular updates.

4. The tools like PBI desktop, VS, SSMS support active directory universal authentication, an interactive method supports Azure AD multi-factor authentication (MFA), Azure AD MFA helps to safeguard access to the data and applications while providing a simple sign-on process. Interactive MFA with Azure AD can result in a pop-up for validation. Universal authentication is recommended.

5. SSMS supports the authentication with Azure Analysis Services through windows authentication, Active directory password authentication and Active directory universal authentication. Supports both interactive and non-interactive authentication methods.
6. Supports Azure AD B2B users invited into the Azure Analysis services tenant, while connecting to the server. The guest users must select Active Directory Universal Authentication when connecting to the server.
7. With MFA, Azure AD MFA can provide safeguard access to data and applications with a range of verification options - phone call, text, smart cards or app verification.

User Permissions in Azure Analysis Services

a) Server Administrators - specific to the Azure Analysis Services server instance. They connect with tools like SSMS, Azure portal, VS to perform the tasks like configuration settings, managing the user roles. The AAS server administrator must have an account in the Azure AD tenant of the same subscription.

b) Database users - they can connect to model databases by using client apps like excel/PowerBI. Users must be added to the database roles, database roles define administrator, process or read permissions for a database. It's important to understand the database users in a role with administrator permissions is different than the server administrators. However, by default, server administrators are also database administrators.

- Roles define for a tabular model are database roles. The roles contain the members consists of Azure AD users and security groups having specific permissions define the action of those members can take on a model database.

- A database role is created as a separate object in the database and applies only to the database in which the role is created.

c) Azure resource owners - Resource owners manage the resources for an Azure subscription. Resource owners can add Azure AD identities to Owner or contributor roles with a subscription by using access control in the Azure portal or with Azure resource manager templates.

Service Principals in Azure Analysis Services

- Service Principals are an Azure Active Directory application resource which you can create within the Active Directory tenant to perform unattended resource and service level operations.

- Service principals are the unique type of user identity with an application ID and password or certificate. A service principal has only those permissions necessary to perform tasks defined by the roles and permissions for which it's assigned.
- In Analysis services, service principals are used with Azure automation, PowerShell unattended mode, custom client applications and web apps to automate common tasks. Like, for example - provisioning servers, deploying models, data refresh, scale up/down and pause / resume can all be automated by using service principals.
- Permissions are assigned to service principals through role membership, much like regular Azure

mode, custom client applications and web apps to automate common tasks. Like, for example - provisioning servers, deploying models, data refresh, scale up/down and pause / resume can all be automated by using service principals.

- Permissions are assigned to service principals through role membership, much like regular Azure AD UPN accounts.

Add Service Principals to server admin role

Before we can use a service principal for Analysis services server management operations, we must add it to the server administrators role. Service Principals must be added directly to the server administrator role. Adding a service principal to the security group, and then adding that to the security group to the server administrator role is not supported.

Hands-on Lab 03

Pre-requisite - Azure subscription is required with Azure AD access

1. Create Azure Analysis Service server instance
2. Create Service Principal in Azure AD
3. How to add a Service Principal to the Server Administrator role in AAS.

client secret - EZ_8Q~sIbcG1DHOUA75ofS.CMjI2J3t_L0xZbcuA

client ID - fbc5431e-b93e-447f-9d5e-ed75c6b78471

tenant ID - 4d387489-ee46-428c-a284-31fd15314024

Service Principal

Subscription ID - 07ee4d7d-00ae-4ebd-b4fd-ff11533f5667

Resolution Steps for connecting to Azure Analysis Services through SSMS

1. Create a new Azure AD user account
2. Add that newly created Azure AD user account to the AAS service admin
3. Try login again through SSMS with the newly created AAS service admin user

Group By Function in DAX

=GROUPBY(Table, Columns_GroupBy, Name1, expression1,.....)

CURRENTGROUP function in DAX can also return a set of rows from the table argument of GROUPBY which belongs to the current row of the GROUP BY result.

The CURRENTGROUP function takes no arguments and is only supported as the first argument of the functions - AVERAGEX, COUNTX, MAXX, MINX

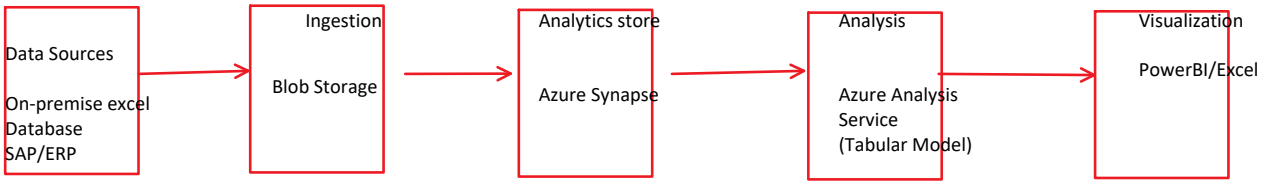
- GROUPBY function in DAX is useful to group by columns with no lineage
- Each column added by GROUPBY must iterate CURRENTGROUP()
- We cant use CALCULATE inside a GROUPBY iteration of DAX

=GROUPBY(Customers, [Customer Category], "#Customers", COUNTX(CURRENTGROUP(), 1))

= GROUPBY(SalesByCountryAndCategory, GeoGraphy[Country], "Max Sales", MAXX(CURRENTGROUP(), [Total Sales])

Azure Synapse Analytics

12 December 2022 19:57



Data warehouse

A data warehouse is a centralized repository of integrated data from one or more disparate sources. Data warehouses store current and historical data and are used for reporting and analysis of the data.

Features of Data warehouse

1. To move data into the data warehouse, data is periodically extracted from various data sources which contains important business information.
2. As the data moves, it can be formatted, cleaned, validated, summarized and reorganized.
3. Alternatively, the data can be stored in the lowest level of detail with aggregated views provided for reporting.
4. The data warehouse becomes a permanent data store for reporting, analysis and BI.

Scenario to use Data warehouse & Benefits

1. The data warehouse can store historical data from multiple sources representing a single source of truth.
2. We can improve the data quality by cleaning up data as it's imported into the data warehouse
3. Reporting tools can be used with the data warehouse platform. A data warehouse allows the transactional system to focus on handling writes, while the data warehouse satisfies the majority of read requests.
4. A data warehouse can consolidate data from different software and data mines can find the hidden patterns in the data using automatic methodologies.
5. Data warehouse helps to provide secure access to the authorized users while restricting access to others. Data warehouses make it easier to create BI solutions like OLAP cubes.

Requirement of Data Warehousing in Azure Platform

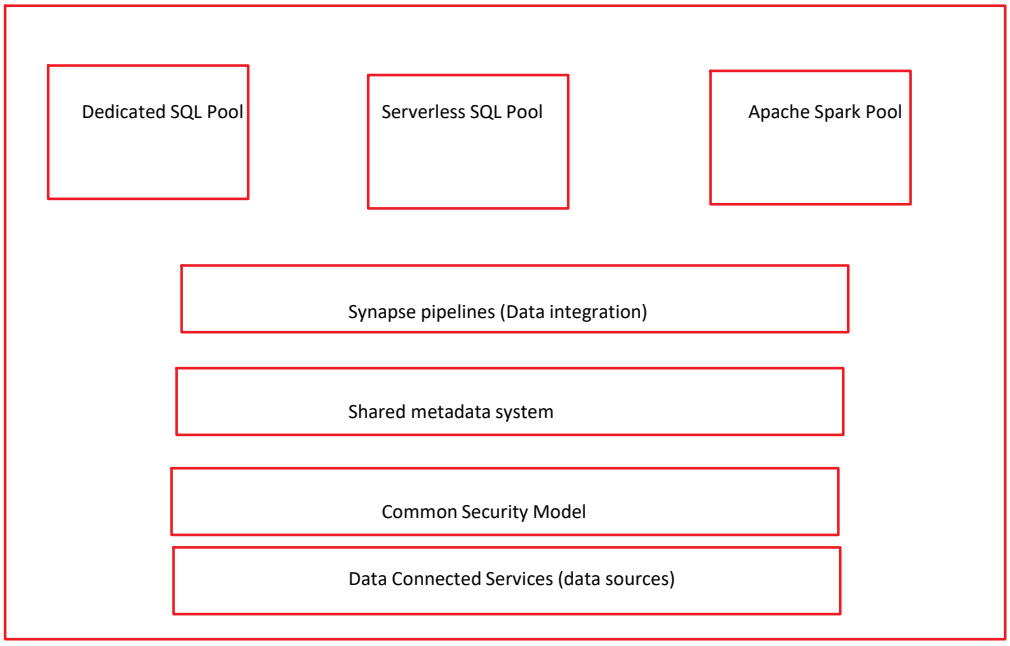
1. The data traditionally being store in one or more OLTP databases. The data could be persisted in other storage mediums like network share, azure blob storage or data lake.
2. The data can be stored by the data warehouse itself or can be stored in relational database like Azure SQL db.
3. The purpose of the analytical data store layer is to satisfy the queries issued by analytics and reporting tools against the data warehouse.
4. In Azure, this analytical store capability can be met with Azure Synapse or formerly Azure SQL DW.

Azure Synapse Analytics (Formerly SQL DW)

Azure Synapse Analytics is an analytics service which brings together the enterprise data warehousing and Big Data Analytics platform. Dedicated SQL Pools (earlier SQL DW) refers to the enterprise data warehousing features available to Azure Synapse Analytics.

1. Azure Synapse (dedicated SQL Pool) represents a collection of analytics resources which are provisioned using Synapse SQL. The size of a dedicated SQL pool is determined by the data warehouse units (DWU).
2. Once the dedicated SQL pool is provisioned, we can import big data with simple Polybase sql queries. Then we can use the power of SQL DW query engine to execute the high performance analytics.

Architecture of Azure Synapse Analytics



Synapse Workspace

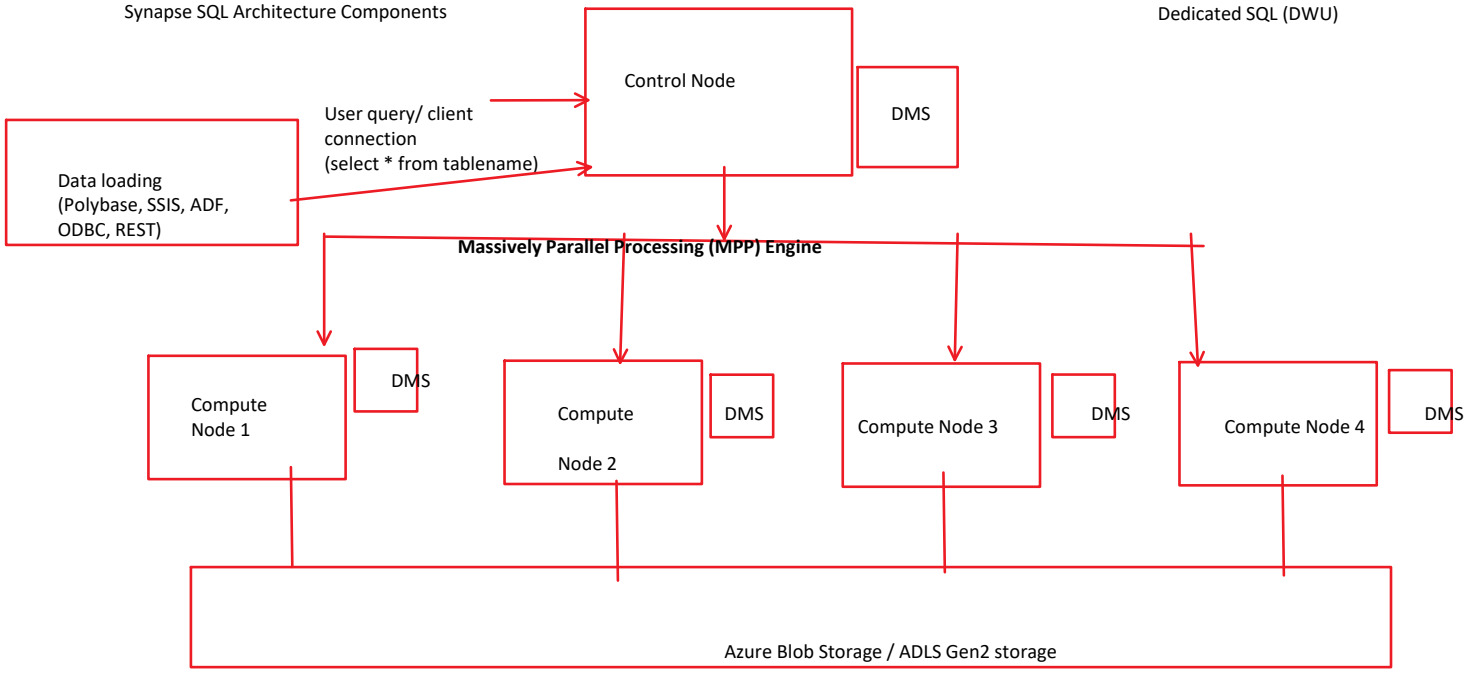
Benefits of dedicated SQL Pool

1. Dedicated SQL Pool uses a node based architecture, applications which can connect and issue tSQL commands to a control node.

2. The Control Node is the brain of the data ware house, which hosts the distributed queries in the query engine. It also optimizes the queries for parallel processing. Passes the operations to compute nodes to execute the work in parallel.

Synapse SQL Architecture Components

Dedicated SQL (DWU)



Features

1. The dedicated SQL pool (SQL DW) uses a scale out architecture to distribute computation processing of data across the multiple nodes.

2. The unit of scale is an abstraction of compute power which is known as the data warehouse unit (DWU).
3. In Data warehouse platform, compute engine is separate from the storage layer, which enables us to scale compute independently of the data in the system.

4. With decoupled storage and compute, when using a dedicated SQL pool, we can independently size the compute power irrespective of the storage needs,
5. Grow or shrink compute power, within a dedicated SQL pool without moving the data.
6. Pause compute capacity while leaving the data intact so that we can pay only for storage.
7. Resume compute capacity during the operational hours.
8. The compute nodes store all user data in Azure storage and execute the parallel queries.
9. The data movement service (DMS) is a system level internal service which moves the data across the data as necessary to run the queries in parallel and return the accurate results.
10. All applications and client connections interacts with the control node, each interacts with a multitude of compute nodes.
11. The Control Node receives the input request and then analyzes it before sending it to the compute nodes.
12. Calculation nodes execute the query on their databases and return the results to the control node which collects the results.
13. The data is stored in Azure Data Lake Storage (ADLS gen2) is not attached to the compute nodes. Hence, we can have the decoupled architecture of compute and storage layer in Azure Synapse platform.
14. The compute layer can be scaled out or in without affecting the underlying storage layer.

Data Movement Service (DMS) in SQL DW

The data movement service is a system - level internal service which moves the data across the nodes as required to execute the queries in parallel and return the appropriate results.

Azure Data Lake Storage Gen2 (ADLS Gen2)

Dedicated SQL pool / SQL DW uses the ADLS Gen2 to keep the data as safe and scalable. The cost is two-fold. Compute and storage altogether creates the Data Warehouse Units (DWU). The data is shared into distributions to optimize the performance of the systems.

Based on the partitions of the data, the tables in the SQL DW/ Dedicated SQL pool / Serverless SQL pool is being designed.

Massively Parallel Processing (MPP)

Hands-on Lab 01

1. Create a dedicated SQL Pool in Azure Synapse Analytics
2. Create the server level firewall rules
3. Connect to the SQL data DW from SSMS
4. Run queries on Azure SQL data warehouse db.

Hands-on Lab 02

1. Create Azure Synapse Analytics workspace
2. Load data into Synapse Analytics into ADLS gen2
3. Loading the SQL pool (SQL DW) & execute the queries
4. Visualize the sql query results in charts / reports.

Distribution Table in Azure Synapse SQL DW database

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

1. **Hash-distribution** improves the query performance on large fact tables, it's the focus of the star schema.

2. **Round-robin** distribution table in SQL DW is useful for data loading speed. These table design has a significant effect on improving the query and loading the performance.

3. **Replicated tables** in SQL DW are useful to replicate small tables across the Compute nodes.

As part of table design in SQL DW, we've to consider the following scenarios

1. How large is the table?
2. How often is the table refreshed?
3. Do we need to have fact and dimension tables in a dedicated SQL pool?

a) Hash distributed table in Azure Synapse SQL DW

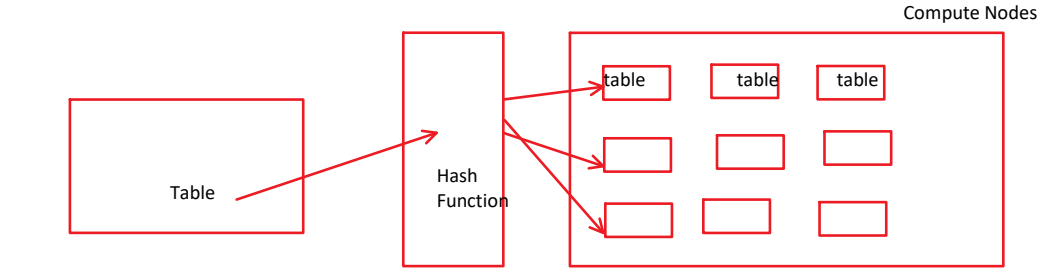
A hash distributed table distributes table rows across the compute nodes by using the a deterministic hash function to assign each row to one distribution.

Since identical values always hash to the same distribution, SQL Analytics has built-in knowledge of row locations. In dedicated SQL pool, the knowledge is used to minimize the data movement during queries, which improves the query performance.

Hash distributed tables work well for large fact tables in a star schema. They can have very large numbers of rows and still achieve high performance. There're some design considerations which can help the performance of the distributed system is designed to provide.

Hash distributed table has to be considered when-

1. The table size on disk is more than 2 GB
2. The table has frequent insert, update, and delete operations.



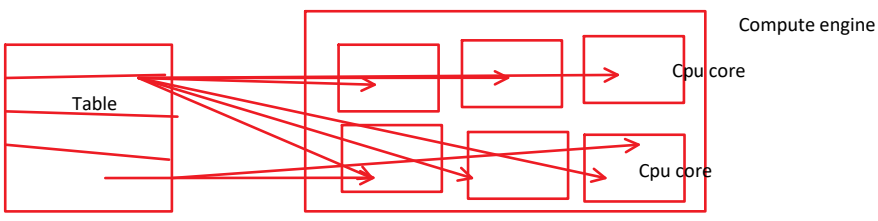
Each table row belongs to one distribution

Round-robin distribution

A round-robin distribution is kind of distribution which distributes the table rows evenly across all partitions. The assignment of rows to distribution is random. Unlike the hash-distributed tables, the rows with equal values are not guaranteed to be assigned to the same distribution.

The system can invoke the data movement operations to better organize the data before it can resolve the query.

- Dimension tables of the Star schema can be used for Round-robin distribution.



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

Scenarios to use Round-Robin distribution

1. Designing of Dimension tables in Star schema
2. There's no specific joining key
3. When the table is temporarily stored as staging table
4. There's no common joining key available with other key tables.

Table Distribution Type	**Hash-Distributed**	**Round-Robin**	**Replicated**
Scenario	Large Fact table consists of large number of rows.	For Dimension tables, the Round-robin distribution is used. The Round-robin distribution is used for dimension tables with large number of columns.	For developing sub-dimension tables with full-copy option from round-robin table.
Size	Size of fact table is at least of 2 GB or more, for those of the scenarios, we use the hash distributed tables.	For dimension tables, the size of the dimension tables are less than 2 GB	Replicated table is actually sub-dimension tables, they're less than 1 GB of size
Performance Efficiency	Most performance efficient, since the hash distribution key is imposed by using the distribution column.	Less performance efficient, since the table rows distributed randomly over the compute nodes of the cluster. Not being matching the same order of the compute nodes while distribution of rows.	Full table copy is performed over the same compute node, less data movement (DMS), maximizes the throughput performance.

Hands-on Lab 03

Task 1:

Load Data into Azure SQL DW/Azure Synapse from Azure SQL database using Azure Data Factory

Pre-requisites

1. **Azure SQL database (with AdventureWorks db)**
2. **Azure Synapse Analytics Workspace with dedicated SQL pool (SQL DW)**
3. **Azure Data Factory**
4. **Azure Blob Storage (Staging storage or Polybase)**

(same resource group + Region) = cost effective

Polybase

Polybase is actually the data loading strategy through which we can load the data into Azure SQL datawarehouse (dedicated SQL pool) of 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 the data after migrating from the source data source can be stored for temporary purpose.**

2. **Once, the migrated data from the source db is hold though polybase and loaded successfully, then it's being transferred to the dedicated SQL pool / Azure SQL DW db.**

Task 2: (Assignment)

Load data into Azure SQL DW/Synapse Dedicated SQL pool by using your project db.

Hands-on Lab 04

1. **Create a Synapse Analytics Workspace**
2. **Create the external data source (CREATE EXTERNAL DATA SOURCE) command**
3. **CREATE the External table in SQL data warehouse db**
4. **Load the data into the external tables using the CTAS (CREATE TABLE AS SELECT) statement.**
5. **Sample queries to test**

6. **Apply the performance tuning and optimization techniques of the SQL DW external tables.**

Hands-on Lab 05

1. **Create the external tables**
2. **Load the data into the external tables of Azure SQL DW db**
3. **Monitor the performance & query optimization techniques.**

Synapse SQL Architecture Components

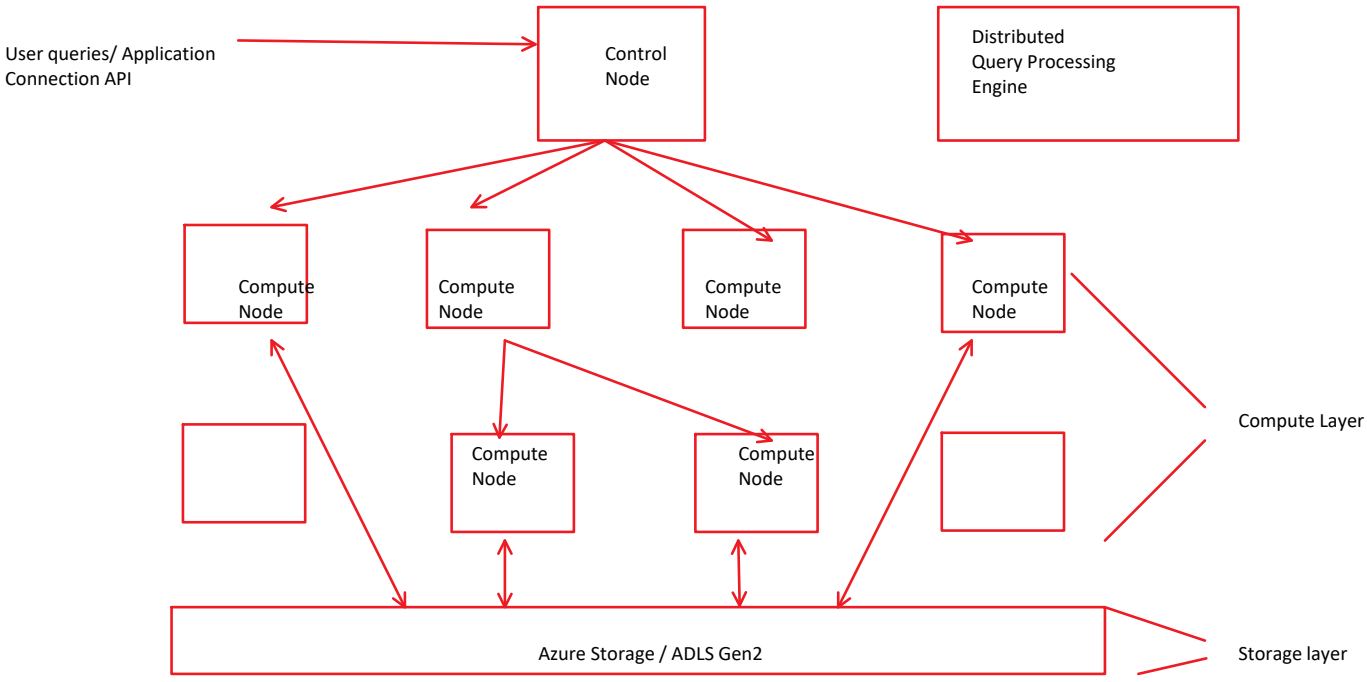
1. Dedicated SQL Pool (SQL DW) - the unit of scale is considered as an abstraction of compute power which is known as the data warehouse unit. (DWU = 100c)
2. Serverless SQL Pool - being serverless, scaling is implemented automatically to accommodate query resource requirements. As topology changes over the time, removing of nodes or failovers, it adapts the change and make sure the query has enough resources and finishes successfully.

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

b) The serverless SQL pool Control node utilizes the distributed query processing (DQP) engine to optimize and orchestrate distributed execution of the user query by splitting it into smaller queries. Then this smaller queries will be executed on Compute nodes.

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

d) 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 Azure Synapse Analytics

1. With decoupled storage and compute, when using Synapse SQL, enterprise can get benefitted from the independent sizing of compute power irrespective of the 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 only we can pay for storage.
5. Resume back the compute capacity during the operational hours.

Benefits of the Control Node in Serverless SQL Pool

1. For Serverless SQL Pool, the DQP engine (distributed query processing) engine runs the Control Node to optimize and co-ordinate the distributed execution of user query by splitting into smaller queries which will be executed on Compute nodes.
2. It also assigns the sets of files/storage to be processed by each node.
3. Whenever the tsql query gets submitted, the control node can optimize and co-ordinate the parallel queries.

Best Practices for data loading into Serverless SQL pool of Synapse Analytics

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, gZIP format.**
c) **All the file formats having different performance characteristics, for fastest load, use compressed delimited text files. Better to use the file formats like Parquet file formats (columnar file format with compression support).**

Compute Nodes benefits in 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 is required to setup or 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) In Serverless SQL Pool, each compute node is assigned task and set the files with filters to execute the task on, the task is distributed on query execution unit which is actually being part of query submitted by the user.
d) Automatic scaling is in effect to define Compute nodes are utilized to execute user query.

In Azure SQL DW, there're two kinds of schemas can be created

1. Star Schema
2. Snowflake Schema

Table	Fact table	Dimension table	Integration Table
	Contains the quantitative data Which are commonly generated by a transactional system. Then the data gets loaded into the dedicated SQL pool. e.g. Sales fact table	Contains the attribute data which might change but usually change infrequently. e.g. CustomerID can be sharable by both Fact and Dimension table. Instead of fact and dimension table, can be joined query over the two tables to associate a customer's profile and transactions.	Provide a place for integrating or staging data. We can create an integration table as a regular table, as an external table or temporary table. e.g. loading the data into staging table & perform the transformation on the staging data and insert the data into production table.
	Contains more number of rows hence, it's vertical table	Contains more number of attributes/ columns, hence it's called horizontal table	Staging table , before loading the data into final production fact table, we load into the staging/integration table first, perform the transformation & then load the data into the final production fact table.

Table Persistence Concepts

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

a) Regular Table - A regular table 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 only exists for the duration of the session. We can use a temporary 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 statement (CTAS), selecting from an external table imports data into the dedicated sql pool.

The **CREATE TABLE AS SELECT (CTAS)** statement

It's one of the important SQL feature which is fully parallelized operation which creates a new table based on the output of select statement. CTAS is the 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 columnstore index or just some columns in the table
4. Query or import the external data

Supported Data types of Azure Synapse / SQL DW

- Int32
- Int16
- String
- Decimal
- Float
- Date
- Datetime

Index considerations on SQL pool tables

Dedicated SQL pool offers several indexing options like

- Clustered Columnstore indexes - created by SQL pool table when no index options are specified.
- Clustered Indexes - Indexes are applied on the list of columns

TSQL features available in dedicated / Serverless SQL pool of Synapse

1. Synapse SQL supports the WHILE Loop for repeatedly executing the statement blocks. This WHILE loop continues for as long as the specified conditions are true or until the code specifically terminates the loop using the BREAK keyword.

2. Loops in Synapse SQL are useful for replacing the cursors defined in SQL. Always almost the cursors are written in SQL code are of the fast forward, read-only variety. So, WHILE loops are great alternatives for replacing the CURSORS.

3. We can replace the SQL cursors with a looping construct while writing query in Synapse SQL.

Update Column name / rename column

EXEC sp_rename 'dbo.tablename' , 'column1', 'column2';

External Tables in dedicated SQL Pool and serverless SQL pool of Synapse SQL

1. Query Azure Blob storage and Azure Data Lake Storage Gen2 with tSQL statements.
2. Store the query results to files in Azure Blob storage or Azure Data Lake storage using CETAS. (Create table as SELECT)
3. Import data from Azure Blob storage and Azure Data lake storage and store it in a dedicated SQL pool.

- CR**EATE EXTERNAL DAT**A SOURCE - to reference an external Azure storage and specify the credential which should be used to access the storage.
- CR**EATE EXTERNAL FILE FO**RMAT - to describe format of CSV or Parquet file
- C**REATE EXTERNAL TA**BLE - on top of the files placed on the data source with the same file format.

```
CREATE EXTERNAL DATA SOURCE <data_source_name>
WITH
(
LOCATION = '<prefix>://<path>'
[,CREDENTIAL = <database_scoped_credential>]
, TYPE = HADOOP
)
[;]

CREATE EXTERNAL FILE FORMAT census_file_format
WITH
(
FORMAT_TYPE = PARQUET,
DATA_COMPRESSION = 'org.apache.hadoop.io.compress.snappycodec'
```

Case Study

12 December 2022 19:57

Sprint 1 Contents

1. Apache Hadoop
2. Azure Data Factory
3. Azure SQL
4. Azure Data Lake Storage Gen2 & Blob storage
5. Azure Analysis Service
6. Azure Synapse Analytics

Select convert (int, table_name.amount_in_usd) from table_name;

Select cast(table_name.amount_in_usd as int) from table_name;

Click here to join the meeting

Apache Spark

12 December 2022 19:58

Difference between Apache Hadoop and Apache Spark

Features	Apache Hadoop	Apache Spark
Definition	Hadoop is the open source distribution 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 is used for big data analytics and transformation
Core Features	Highly scalable, cost-effective solution which stores and processes the structured, unstructured data but it requires good amount of infrastructure capacity to execute the data pipeline faster	Spark splits up the large tasks across the different nodes. However, it tends to perform faster than hadoop and uses the memory (RAM) to cache the data and process the data instead of the filesystem. Which enables Spark to handle the use cases which hadoop cant do.
Benefits	a) Data protection even there's a 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 processing engine which supports SQL queries, streaming data, ML and graph processing
Ecosystem	Hadoop supports advanced analytics for stored data (e.g. predictive data analytics, data mining, ML etc.). It also enables big data processing tasks to be split into smaller tasks. The smaller tasks are performed in parallel by using an algorithm (e.g. MR) and then distributed across the hadoop cluster.	Apache Spark is the largest open source project can be used for data processing. It combines the feature of both AI and ML. It enables the users to perform large-scale data transformations and analysis.
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 the smaller ones, distributes the smaller tasks across the different nodes which executes each tasks across the different nodes. d) YARN - Cluster resource manager which schedules the tasks and allocates the resources (e.g. CPU and memory) to applications. e) Hadoop Core - set of common libraries and utilities on which the modules like HDFS, YARN, MR depends on	a) Spark Core - underlying execution engine which schedules and dispatches tasks and coordinates I/O operations. b) Spark SQL - gathers information about the structured data to enable users to optimize the data processing. c) Spark Streaming/ Structured streaming - Both of the streams could add data stream processing capability. Spark streaming takes data from the different streaming sources and divides it into the micro-batches for a continuous stream. Structured streaming is built on top of Spark SQL, reduces the latency and simplifies the programming. d) GraphX - user-friendly computation engine which enables the interactive query building capability, which can be used for data modification, analysis of scalable and graph-structured data.
Core processing	Hadoop Mapreduce process the data on disk	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.
Benefits	Mapreduce jobs are comparatively slower than Spark	Spark's data processing capacity is 100x times faster than Mapreduce.
Job Execution	Data is being processed two stages in hadoop mapreduce. - Map - Reduce	Spark creates a directed acyclic graph (DAG) to schedule tasks and orchestrates the cluster nodes across the hadoop cluster.
Data processing	In hadoop mapreduce, mapper and reducers are being executed including partitioners and combiners while processing the data.	The task-tracking process enables fault tolerance which reapplies recorded operations to data from a previous state.

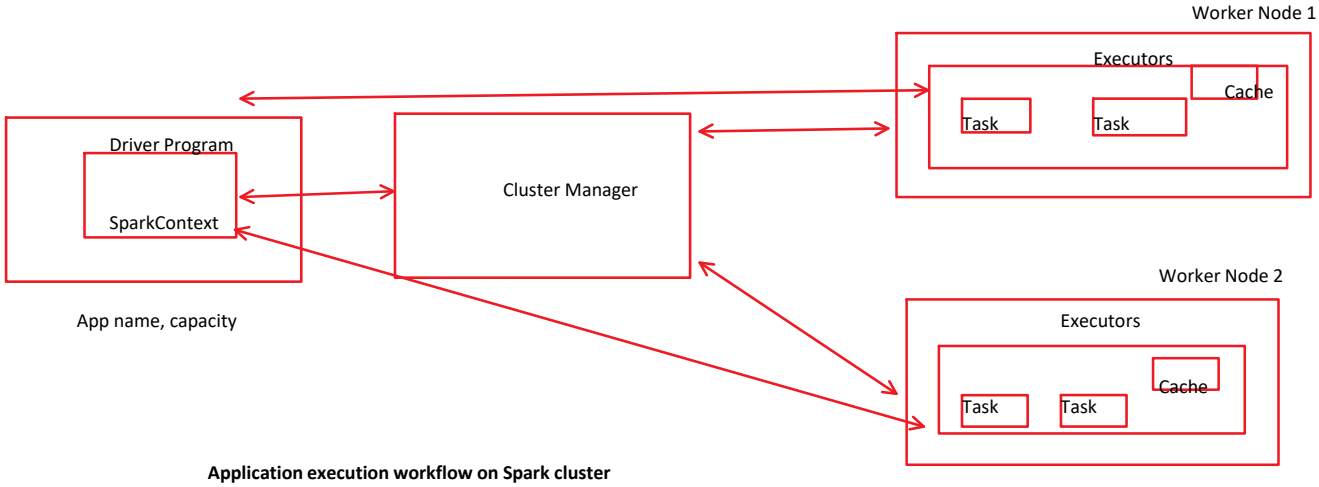
Spark Components Overview

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 a cluster, the SparkContext can connect to several types of cluster managers (either Spark's own standalone cluster manager or YARN or 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) Which are the processes that can execute the computations and store the data for the application.
e) Next, it can send the application code (JAR, python files passes to SparkContext) to the executors.
f) Finally, SparkContext send the tasks to the executors to run.

Spark Cluster Architecture Components



Application execution workflow on Spark cluster

1. Each application in Spark gets its own executor processes. Which stay up to date during the whole application execution duration and can run tasks in multiple threads.
2. This has the benefits of isolating applications from each other, or both of the scheduling side (each driver schedules its own tasks) and executor side, tasks from different applications can run in different JVMs.
3. However, it also means data can be shared across the different spark applications (instances of SparkContext) without writing to an external storage system.
4. Spark is agnostic to the underlying cluster manager, as long as it can acquire the executor processes, these can communicate with each other, its relatively easy to run it even on a cluster manager which supports the other apps. (e.g. Mesos, YARN etc.)
5. The driver program must listen for and accept incoming connections from its executors throughout its lifetime. As such the driver program must be network addressable from the worker nodes.
6. Because, the driver schedules tasks over the cluster, it should be close to the worker nodes, preferably on the same LAN, if it's required to send requests to the cluster remotely.
7. It's better to open an RPC (remote procedure call) to the driver and have it submit the operations from the nearby then to the execution of driver far away from the worker nodes.

Driver Program	The process running the main() method for the application and responsible to create the SparkContext class object. Once the SparkContext is 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 tasks and keeps the data on memory or disk storage across them. Each application has its 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 the 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 an entry point to Spark to work with RDD, Dataframe and dataset. Prior to 2.0, SparkContext used to be an entry point.

Definition of SparkSession

SparkSession is introduced in version Spark 2.0 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 created programmatically using SparkSession builder pattern.

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

- SparkSession instance would be first statement when we're writing programs on RDD, Dataframe and dataset. SparkSession will be created using the SparkSession.builder() method. We can also use SparkSession.newSession() 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 we used to have prior to 2.0 release. Since, 2.0, SparkSession can be used in replace with SQLContext, HiveContext and other contexts prior to 2.0.

Definition of Dataframe

Dataframe is a distributed collection of data organized into named columns. It's conceptually equivalent to a table in a relational database or a dataframe in R/Python, but also includes richer optimizations under the hood. So, Dataframes can be constructed from a wide array of sources such as structured data files, tables in hive, external databases or existing RDDs.

Dataframe in PySpark

The dataframe is a distributed collection of data organized into named columns.

Definition of Resilient Distributed Dataset (RDD)

At a high level, every spark application consists of a driver program which runs the user's main function and executes various parallel operations on a cluster. The main abstraction Spark provides a resilient distributed dataset (RDD) which is a collection of elements partitioned across the nodes of the cluster that can be operated in parallel. RDDs can be created by starting with a file in the hadoop file system (or any other hadoop-supported file system hdfs) or an existing Scala collection of the driver program, and transforming it.

Users may also ask Spark to persist an RDD in memory, allowing it to reused efficiently across parallel operations. Finally, RDDs automatically recover from the node failure.

Shared variables in Spark

- In Spark, the shared variables are used that can be utilized for parallel operations. By default, while Spark runs a function in parallel as a set of tasks on different nodes, it ships a copy of each variable used in the function to each task.
- Sometimes, the variable needs to be shared across the tasks or between the tasks and the driver program.
- Spark supports two types of shared variables

a) Broadcast variables - these variables can be used to cache a value in memory on all nodes.

Broadcast variables also allows the developers to keep a read only variable cached on each machine rather than the shipping copy of it with tasks. They can be reused. For e.g. to provide every node a copy of large input dataset in an efficient manner.

1. Spark also attempts to distribute the broadcast variables using the efficient broadcast algorithms to reduce the communication cost.
2. Spark actions are executed through the set of stages, separated by distributed "shuffle" operations.
3. Spark automatically broadcasts the common data required by tasks within each stage. The data broadcasted this way cached in serialized form and deserialized before running each task.

a) Accumulators - these are the variables which are only "added to" such as counters and sums.

Broadcast variables	Accumulators
It can be used to cache a value in memory on all nodes	Accumulators are the variables which can be only added to such counters and sums.
Allows the developers to keep a read-only variable cached on each machine/node rather than shipping a copy of it with tasks.	Added to through an associated operation and can therefore be efficiently supported in parallel.
Broadcast variable in spark are cached on executor side. e.g. if there're 10 executors are running on spark worker nodes, and in your app execute 100 tasks in total. The broadcast variable of Spark will sent to the 10 executors	Accumulators are variables which are used for aggregating info across the executors.

as opposed to 100 times.	

Demo - 3:

Create an Empty RDD and Dataframe with PySpark

a) To create an empty RDD in Pyspark, we can use the emptyRDD() of SparkContext class.

Resilient Distributed Dataset (RDD)

Spark revolves around the concept of resilient distributed dataset(RDD), which is a fault -tolerant collection of elements that can be operated in parallel. There are two ways to create a 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 RDD.

Operations on RDD

1. Transformation
2. Action

Transformation in RDD -

In Spark, the role of the transformation is to create a new dataset from an existing one. The transformation are being considered as **lazy evaluation** as they are only 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 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 the argument.
GroupByKey(NumPartitions)	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 of (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 of type (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 the driver program after running the computation on the dataset.

Actions	Description
Reduce(func)	It aggregates the elements of the dataset using a function func(which takes two arguments and returns 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 operation which returns sufficiently the small subset of data.
Count()	It returns the number of elements on the dataset
First()	Returns the first element of the dataset (similar to take(1))
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 on 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 a 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 the random sample of num elements in the dataset with or 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.

Benefits of Spark RDD

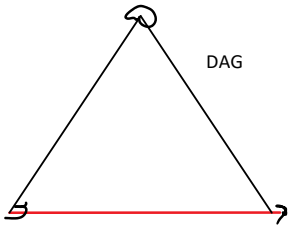
1. RDD is useful to implement iterative algorithms
2. Interactive data mining tools
3. Each and every dataset in Spark RDD is logically partitioned across many servers so that they can be computed on different nodes of the cluster.

- Resilient - since as fault tolerant , a lineage graph (DAG) is being created and it can recompute any missing or damaged dataset partitions due to node failure.

DAG Scheduler -

In Spark RDD directed Acyclic graph, every edge directs from earlier to later in the sequence. On the calling of an Action, the created DAG submits the job to the DAG scheduler so that the scheduler can further splits the graph into the stages of the task.

Apache Spark directed acyclic graph (DAG) allows the user to dive into the stage and expand every detail on any job execution stage.



Edges of the graph represents - RDD operations
Vertices represents - RDDs

Caching in RDD

The Spark RDD can also be cached and manually partitioned. Caching is useful when we require to use RDD multiple times. Since, manual partitioning is important to correctly balance the partitions.

Generally , the smaller partitions allow distributing RDD data more equally among many executors. Hence, fewer partitions make the work easier.

Persistance in RDD

Apache Spark has the capability to provide a convenient way to work on the dataset by persisting it in memory across operations. While persisting an RDD, each node stores any partitions on it so that it can compute in memory, Now, we can reuse the RDDs on other tasks of the dataset.

We can use either persist() or cache() method to mark an RDD is to persisted. Spark's cache is fault-tolerant.

Benefits of Persistance and Caching in RDD

1. In case of missing partitions, lost partition, the respective RDD partition will automatically be recomputed using the transformation which originally creates it.
2. There're different storage levels can be used to store the persisted RDDs. These storage Levels can be used by passing StorageLevel object to persist().
3. The cache() method can be used for the default storage level. For default storage level, RDD caching, the storage level type is StorageLevel.MEMORY_ONLY.

StorageLevel in RDD for persistance and Caching

Memory_ONLY	Stores the RDD as deserialized java objects on JVM. This is the default storage level of persisted RDD. If the RDD doesn't fit into the memory, some partitions will not be created and recomputed each time when required.
MEMORY_AND_DISK	Stores the RDD as deserialized java objects on JVM. If the RDD doesn’t fit into the memory, store the partitions if it doesn't fit it in the disk, it'll read from there when it's required
MEMORY_ONLY_SER	Stores RDD as serialized java objects. This is generally space-efficient than deserialized object.
MEMORY_AND_DISK_SER	Spills over the partitions which doesn't fit in memory to disk instead of recomputing them

Core Features of Spark RDD

1. In-memory computation

2. Lazy evaluation
3. Immutability
4. Fault tolerance
5. Persistence - users can choose which RDD will be reused and based on that choose the appropriate storage level
6. Partitioning - fundamental unit of parallelism in Spark RDD. Each partition is one logical division of data which is mutable. One can create a partition through some transformation on existing partitions.
7. Location-stickiness - RDDs are capable of defining placement preference to compute partitions. Placement preference the location of RDD. The DAGScheduler places the partitions in such a way that tasks in close to the data as much as possible. Thus, it can speed up the computation.

Hands-on Lab 04

1. PySpark Dataframe operations with show(), StructType(), StructField(), ColumnClass, select(), collect(), withColumn(), withColumnRenamed(), where(), filter(), drop(), dropDuplicates(), orderBy(), sort(), join(), union() and unionAll()

2. PySpark dataframe with transform(), apply(), foreach(), fillna(), fill(), partitionBy() functions

Hands-on Lab 05 PySpark SQL functions

1. Pyspark - SQL aggregate functions
2. Pyspark - SQL Window functions
3. Pyspark - SQL Date and timestamp functions

Hands-on Lab 06 PySpark Datasources

1. Read & write with CSV file
2. Read and write with json file etc.

Hands-on Lab 07 PySpark in-built functions

1. Data transformations with functions like when(), expr(), lit(), split(), substring(), translate() , to_date(), datediff(), array() etc.

Spark SQL / PySpark SQL

1. Spark SQL is a Spark module used for structured data processing.
2. Interaction with Spark SQL can take place SQL and Dataset API.

Benefits

a) The major benefits of Spark SQL is to execute SQL queries. Spark SQL can also be used to read data from an existing hive table.
b) When running SQL query on Spark SQL from any programming language - the results will be returned as dataset/dataframe.
c) We can interact with the SQL interface using the CLI and JDBC/ODBC
d) Through Spark SQL, we can build the relational data tables over RDD.
e) We can import relational data from the files like parquet, avro, csv, json and hive tables.
f) We can execute SQL queries over imported data and existing RDDs.
g) Easily write RDDs out to hive tables or parquet/avro/csv files.
h) Spark SQL includes a cost based optimizer, columnar storage and code generation to make queries faster.
i) At the same time, it scales to thousands of nodes to multi-hour queries using the spark engine.
j) It provides full mid-query fault tolerance, without having to using any different engine for historical data.

Datasets and DataFrame

A dataset is a distributed collection of data.

- Dataset is a new interface added in Spark which provides the benefits of RDD, (strong typing, ability to use powerful lambda functions can be used in pyspark) with the benefits of Spark SQL's optimized execution engine.

- A dataset can be constructed from JVM objects and manipulated using the functional transformations(map, flatmap, filter etc.).
- A dataset API is available in Scala, java. In Pyspark, we don't have support of dataset API. Due to dynamic nature of python, many beneficial feature of datasets are available in pyspark.

DataFrame

A dataframe is a dataset organized into the named columns. It's conceptually equivalent to a relational database or a dataframe in python. Dataframes can be constructed from a wide array of data sources like structured data files, tables in hive, external databases or existing RDDs.

- The Dataframe API is available in Python, Scala, java etc.
-
-

SQLContext Class in PySpark SQL / Spark SQL

SQLContext is a class which is 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 interface. A Dataframe can be operated on using relational transformation and also be used to create a temporary view/table.

Registering a dataframe as a 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 columnar format by invoking spark.catalog,cacheTable("tableName") or dataFrame.cache().
b) Spark SQL/PySpark SQL will scan only the required columns and will 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.

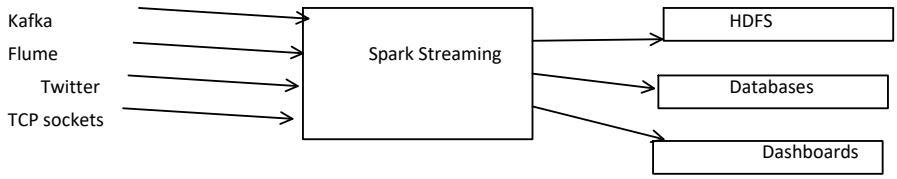
PySpark SQL Functions

PySpark SQL provides several built-in functions which comes from org.apache.pyspark.sql.functions to work with Dataframe and SQL queries. All these Spark SQL functions can return org.apache.pyspark.sql.Column type.

- String functions (concat, ltrim, upper, lower)
- Data and time functions (current_date, to_date, add_months, date_add, year, quarter, datediff etc.)
- Math functions (Pi, Sin, Cos, Tan, Log)
- Aggregate functions (avg, min, max, median, count, collect_list etc.)
- Window functions (window, dense_rank, rank, ntile, row_number, lag, lead etc.)

Spark Streaming

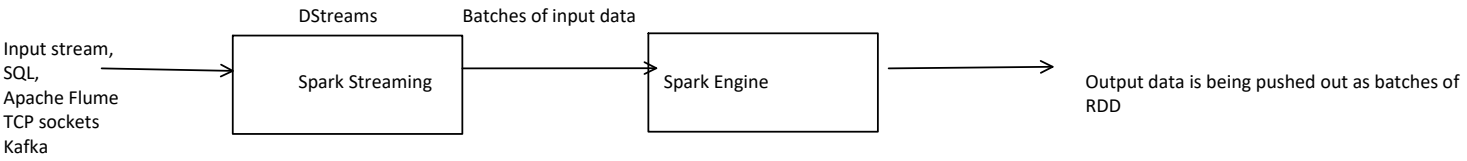
Spark Streaming works on the principles of micro-batches.



Spark Streaming is being able to process real time data from the various real time data streams as discretized streams (D-Streams) which are fundamental abstractions as they represent streams of data into small chunks.

a) Spark streaming has gained adoption because of its disparate data processing capabilities making it easy for developers to rely on the single framework to satisfy all processing requirements.
b) The data models can be trained offline using Mlib and then can be used for streaming data scoring using Spark streaming.
c) Some data models can learn and score simultaneously while the streaming data is getting collected.
d) Spark SQL makes it possible to combine the streaming data with a wide range of static data sources.

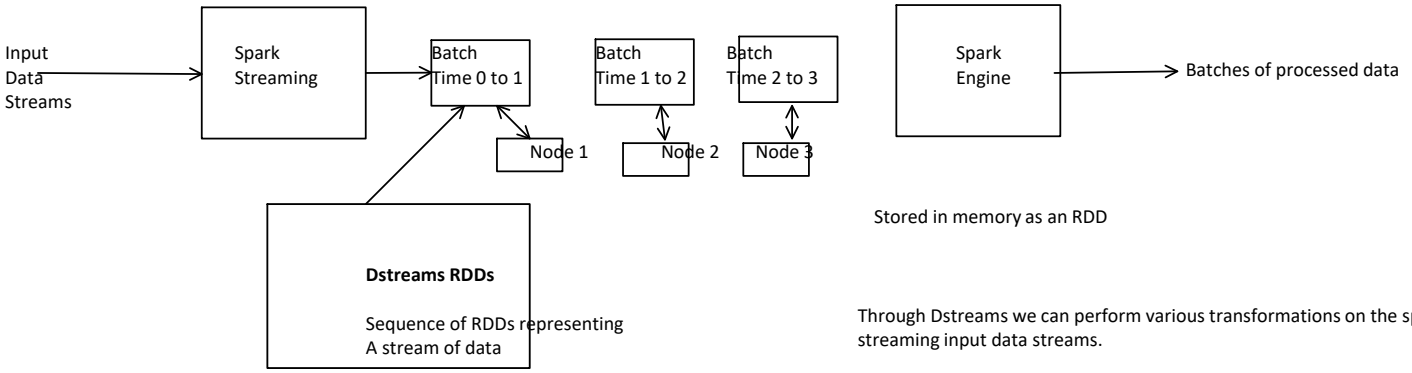
Workflow of Spark Streaming Engine



Discretized Stream or Dstream is the basic abstraction provided by the Spark Streaming. It represents a continuous stream of data, either the input data stream received from the source or the processed data stream generated by transforming the input stream.

- Dstream is a sequence of RDDs. Spark receives the real time data streams from the various input data sources so it divides it into the smaller batches for the execution engine.

- In Spark Structured streaming, the structured streaming is built on Spark SQL API for the data stream processing.
- Dstreams are represented as a continuous streams of RDDs which's Spark's abstraction of an immutable, distributed dataset.



Data processing in Spark Streaming

- **Spark** streaming is a separate library in Spark which continuously streams data
- It provides the discretized Streams which is called Dstreams empowered by Spark RDDs.
- Dstreams provide the data divided into smaller chunks as RDDs received from the source of streaming to be processed.
- After processing, sends it to the destination.

Through Dstreams we can perform various transformations on the spark streaming input data streams.

There're two types of transformations can be performed.

1. **Stateless transformation -**

a) The processing of each batch has no dependency on the data of previous batch.
b) Stateless transformations of Dstreams are simple RDD transformations.
c) It applies on every batch means every RDD in a streams.
d) It includes RDD transformations like map(), filter(), reduceByKey(), etc.
e) Each Dstream is a collection of many RDDs (batches).

2. **Stateful transformation**

It uses the data or the intermediate results from the previous batches and computes the result of the current batch. Stateful transformations are the operations on Dstreams which can track data across time.

- Thus it makes use of some data from the previous batches to generate the results for a new batch.
- Two kinds of windows operations which can act over a sliding window of time periods like updateStateByKey() which is used to track the state across the events for each key.

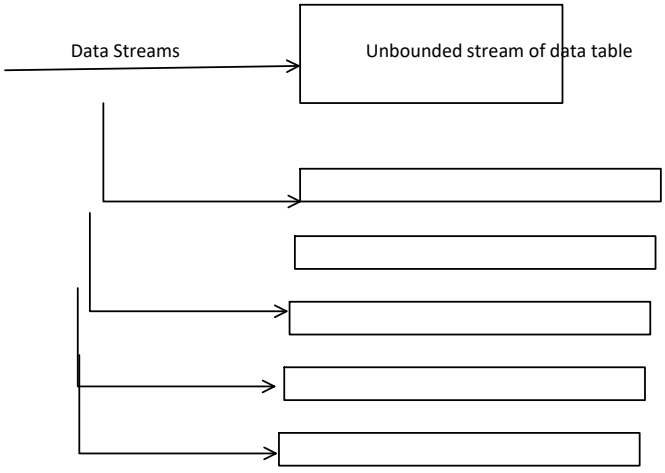
Spark Structured Streaming

Spark Structured Streaming is a scalable, fault-tolerant stream processing engine built-on top of the Spark SQL engine. We can express the streaming computation on the same way as we could express the batch computation on the static data.

- Spark Structured streaming is another way to handle structured data with Spark. This model of streaming data is based on dataset and DataFrame API.
- We can also utilize any SQL query (using the Dataframe API in Spark Structured Streaming) on the streaming data.

- The computation on Spark Structured streaming is executed on the same optimized Spark SQL engine.
- Finally, the system ensures end-to-end exactly once fault tolerant guaranteed data processing through checkpointing and Write-ahead logs
- Spark Structured streaming provides fast, scalable , fault-tolerant, guaranteed streaming data processing without the user having to reason about streaming.
-
- Internally, Spark Structured streaming queries are processed by using a micro-batch processing engine which processes data streams as a series of small batch jobs and thereby achieving end-to-end latencies as low as 100 milliseconds and exactly-once fault tolerant guarantees.
-
-

Workflow architecture of Spark Structured Streaming



New rows in the data streams are automatically appended to a unbounded table

Spark Streaming (Dstreams)	**Spark Structured Streaming**
Works on micro-batches. The stream pipeline is registered with some operations and Spark actually polls the source after every batch, duration. Then the batch is created of the received data. Each incoming record belongs to a batch of Dstreams. Each Batch represents an RDD.	Works also on the same architecture of polling the data after some duration, based on trigger interval. There's no concept of batches, the received data in a trigger is appended into the continuously flowing data streams. Each row of data stream is processed and the result is updated into the unbounded result table. New data in the data stream, automatically whenever new rows are available are appended to a unbounded stream of data table.
Spark Streaming is based on Dstreams API	Spark Structured Streaming uses dataset and dataframe API to perform the streaming operations
Spark Streaming only works with the timestamp when the data is received by the Spark. Based on the ingestion timestamp, Spark streaming put the data in a batch even if the event is generated early and belonged to the earlier batch. It may cause into less accurate information leading to data loss.	Structured streaming provides the functionality to process the data on the basis of event-time when the timestamp of the event is included in the data received. Structured Streaming provides a different way of processing the data according to the exact time of the data generation in real time. We can handle data coming on late events and get more accurate results/
Spark Streaming (Unstructured) also gets benefit from checkpointing. There's no guarantee of end to end exactly once semantics	In Spark Structured streaming, though the utility of checkpointing, it can provide end to end exactly once semantics. Avoiding the data loss. The destination / sink must support idempotent operations
There's no restriction on the type of sink, RDD caching and performance tuning is possible. Streaming returns an RDD created by each batch one-by-one and can perform any actions over them like saving or storage or performing some computations.	Spark Structured streaming is more flexible and gives an edge over the spark streaming and over the other flexible sinks.

	In Spark Structured Streaming, the queries are processed using a micro-batch processing engine which processes the data streams as a series of small batch jobs and thereby achieving end to end latencies as low as 100 ms and exactly once guarantees through the checkpointing and write-ahead logs. Structured streaming provides fast, scalable, fault-tolerant exactly once stream processing without the user having to reason about streaming.

Data Protection Feature of Spark Structured Streaming

- Structured Streaming has the data protection feature against node level failure, a logs of journals are managed called as write ahead logs (WAL).
- Structured streaming enforces fault-tolerant capacity by saving all data received by the receivers to log files located in the checkpoint directory.
- It can be enabled through spark streaming receiver write ahead logs enable property.
- Once the write-ahead-log (WAL) is activated, cache level should make a replication.
- Additional condition is the reliability of receiver, it should acknowledge data reception only after to be sure to save it into the write ahead log. (WAL)

Hands-on Lab 01

SparkStreaming class -
StreamingContext class to explore and perform a wordcount operation to count the number of words in text data stream.

Hands-on Lab 02

Output modes for Spark Streaming
- Append mode
- Complete mode
- Update mode

a) Append output mode in which only the new rows of the processed data streams will be written in the sink.
b) Complete output mode in which all the rows of the processed data streams will be written in the sink.

Hands-on Lab 03

Reading the streaming data from the directory utilizing the Spark Structured streaming

- In Spark Structured streaming, a new low latency processing mode called continuous processing which can achieve end to end latencies as low 100 ms with at least of once guaranteed processing.

Hands -on Lab 04

Pandas API with PySpark

a) Object creation - creating a pandas on pyspark series by passing a list of values, letting pandas API on PySpark to create a default integer index.

Python Programming

12 December 2022 19:58

1. Python is currently most widely used programming language, high level scripting language.
2. Python allows programmers to use object oriented and procedural paradigms.
3. Python programs are generally consisting of less lines of code, smaller than other programs like java, C++.
4. Python language is being widely used by almost all technical products like - GCP Bigquery, AWS S3, Uber, Zomato.. Etc.
5. Python is widely used for machine learning and AI applications.
6. GUI applications (PyQt, Tkinter etc.)
7. Web frameworks like Django and Flask (used by Youtube, Dropbox)
8. Web scraping, web development, big data transformation and analytics pipeline etc. (with Spark)
9. Test frameworks (test automation tools like Qt, Selenium etc.)
10. Text processing, neural networks and scientific computing purpose.

1) Get an Interpreter

We need to have an interpreter to interpret and execute the programs.

- For windows env , the many interpreters are available to run scripts like IDLE (integrated Development Environment).
- Linux
- macOS

2) Features of Python

a) Interpreted language

- There are no separate compilation and execution steps like C, C++/java
- Directly the python programs can be executed from the source code
- Internally, Python converts the source code into an intermediate form called as bytecodes which then translated into the native language of specific computer to run it.
- There's no requirement of linking and loading the code with libraries etc.

b) Platform-independent

- Python programs can be developed and executed on multiple operating system platforms.
- Python can be used on Linux, Windows, OSX, Chrome os etc.

c) Python is free and open source

d) Python is high-level language - In python, there's no need to take care of low-level details like managing the memory used by the program.

e) Python is simple programming language

- In python, the code is written is seamlessly with indentation
- More emphasis is given on the solution rather than syntax.

f) Robust

- Exceptional memory handling features
- Memory management techniques in built

g) Rich library support

- The python library is very vast.
- Python can be for implementation of regular expressions, document generations, multi-threading, unit-testing, databases, web browsers, XML, HTML, media files, cryptography, GUI programming and scientific computing.
- Besides the standard library, python is also used for high quality libraries like for image processing, Machine learning and AI applications.

h) Python dynamically typed

- There no requirement to declare anything in python.
- An assignment statement binds a name to an object.
- The object could be of any type.

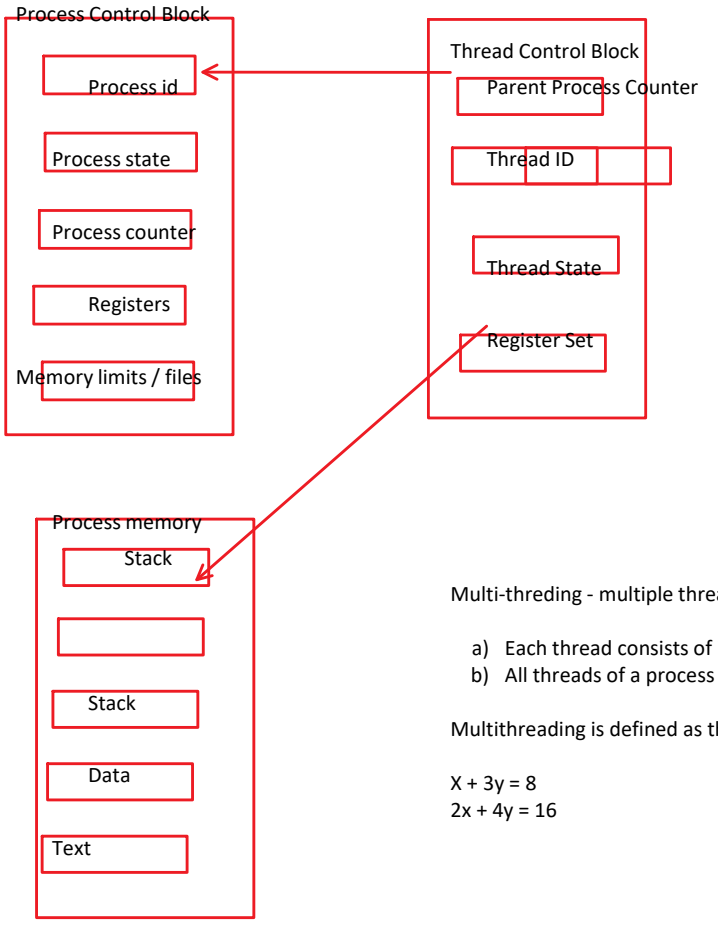
- No type casting is required when using container objects.
- Express the statements concisely in limited words.
- Use indentation for structuring the code

i) Open library support
- Numpy, scipy, matplotlib, plotly, pandas for data analytics etc.
- User-friendly data structures
- Python is a dynamically typed language (no requirement to mention the data types based on the value assigned, it takes the data type)
- Object oriented language, portable and interactive.
- Highly efficient (python's clean object oriented design provides enhanced process control, and the language is equipped with excellent text processing and integration capabilities, as well as its own unit testing framework).
- IoT programming (devices, sensors collect data for real time data streaming and analysis)
- Portable across the OS

Installation

Multi-threading

1. Install Python SDK 3.10/3.11
2. Install Python extension for VS code



Relationship between the process and thread

Multi-threding - multiple threads can exist in one process.

a) Each thread consists of its own register set and local variables(stored in stack).
b) All threads of a process share a global variable(stored in heap) and the program code.

Multithreading is defined as the ability of a process to execute multiple threads concurrently.

X + 3y = 8
2x + 4y = 16

Azure Databricks

12 December 2022 19:58

The Azure Databricks provides a unified set of tools for building, deploying, sharing and maintaining enterprise grade data solutions at scale. Azure Databricks guarantees the cloud data storage and security on Azure account. Azure Databricks manages and deploys Azure infrastructure on behalf of customers.

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 benefits to optimize the data	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 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 admin 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 framework (hadoop, pig, hive, kafka, sqoop etc.) which are easily accessible from the Azure portal.	Databricks can offer the various data integration capabilities - Data engineering platform - Data Streaming platform - Data Science and AI platform - Delta Lake - Lakehouse
Per cluster time based pricing	Billing is calculated per cluster time (VM cost + DBU (databricks unit - infrastructure compute capacity))
Apache Spark is available through Spark type of Azure HDI cluster	Apache Spark is core components and provides 5 times more performance than Azure HDI
We can work with Jupyter notebook, we can perform the Spark jobs also through cluster shell.	In Databricks, we can work with jupyter notebooks, pyspark for Spark SQL and Spark Streaming jobs
Azure Hdinsight, the spark /hadoop cluster can't be paused.	Azure Databricks cluster is easy to manage machines and allows auto-scaling. In Azure Databricks, the Spark cluster can be paused, resumed and deleted

Azure Synapse Analytics with Databricks

Azure Synapse Analytics	Azure Databricks
Azure Synapse provides limitless analytics 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 the data mines, data marts and Data warehouses	Integrates with Azure to help to build the optimized ML algorithms (tensorflow, PyTorch, Keras). Mlflow can also be executed on Databricks Spark cluster along with the benefits of pyspark.
In Synapse, we can perform end to end SQL% data analysis. Since as managed data warehouse platform, it integrates the full capacity of SQL and DW capability with top down approach. In Azure Synapse/ SQL DW, the sql 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 the data warehousing capacity like building of fact and dimension tables with star/snowflake schema Databricks we have the full data transformation(map(), flatmap(), groupBy(), orderBy(), sort() etc.) and actions(collect(), reduce(), count()) operations available. But doesn't have full support of TSQL (limited in Spark SQL).

Benefits / utility of Azure Databricks

- Customers can fully embrace the Databricks platform to get the benefits of its unified platform to build and deploy data engineering workflows, machine learning 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, zeppeline notebooks)
b) SQL editors and dashboards
c) Data ingestion and data governance
d) Data discovery and data exploration
e) Machine learning based model development & deployment
f) Source control with Git

The Components of Azure Databricks

For three personas of Azure Databricks, the following three environments, there're few common resources across the cluster

- Databricks data science and engineering
- Databricks machine learning
- Databricks SQL

a) **Workspaces** - In Azure Databricks the workspace has two types

- An Azure databricks deployment in the cloud which's the unified environment used 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. for the workspace we can implement the spark cluster, we can browse the notebooks, we can access the libraries.

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

1. Notebooks - A web based interface to documents which contain 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 include many libraries and we can add out own libraries as well.
4. Experiment - It's the collection of Mlflow runs for training a machine learning 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 over a Azure blob storage. It can contain files(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 the tables with Apache Spark SQL and Apache Spark APIs.
8. Metastore - This component stores all the structured information of the various tables and partitions in the DW including columns and column type information, the serializers and deserializers necessary to read the data and the corresponding files where the data is stored.

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

   Resources in Azure Databricks for Computation Managements

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 clusters

i) **All-purpose cluster** - we can create all-purpose cluster using the Azure portal UI, CLI or REST API. We can manually terminate and restart the all-purpose cluster. Multiple users can share the all-purpose cluster to perform the interactive data analysis

Ii) **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 can't restart the job cluster.

2. **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 its driver and worker nodes from the pool. If the pool does not have sufficient idle resources to accommodate the cluster's request. The pool can expand by allocating new resources 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.

3. **Databricks Runtime** - Databricks runtime includes Apache Spark but also adds a number of components and updates the substantially improve the usability, performance and security of big data analytics.

4. **Workload** - Azure databricks can have two types of workloads like data engineering (job) and data analytics (all - purpose)

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 also being treated as an interactive workload.

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

c) **Authentication and authorization** - Azure Databricks uses the following components 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 - a service identity for use with jobs, automated tools and systems such as scripts, apps and CI/CD platforms. Service principals are represented by the applicationID.

d) **Group** - An Azure AD group is the collection of identities. Groups can simplify the identity management, assign access to workspaces, data and other scalable objects. All databricks identities can be assigned as members of groups.

e) **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.

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

Hands-on Lab 01

1. Provision the Azure Databricks Workspace through portal
2. Create the Spark cluster in Databricks
3. Import the sample data from public blob storage account
4. Execute the Spark SQL job in the PySpark Jupyter notebook.

Azure Databricks Architecture

Hands-on Lab 02

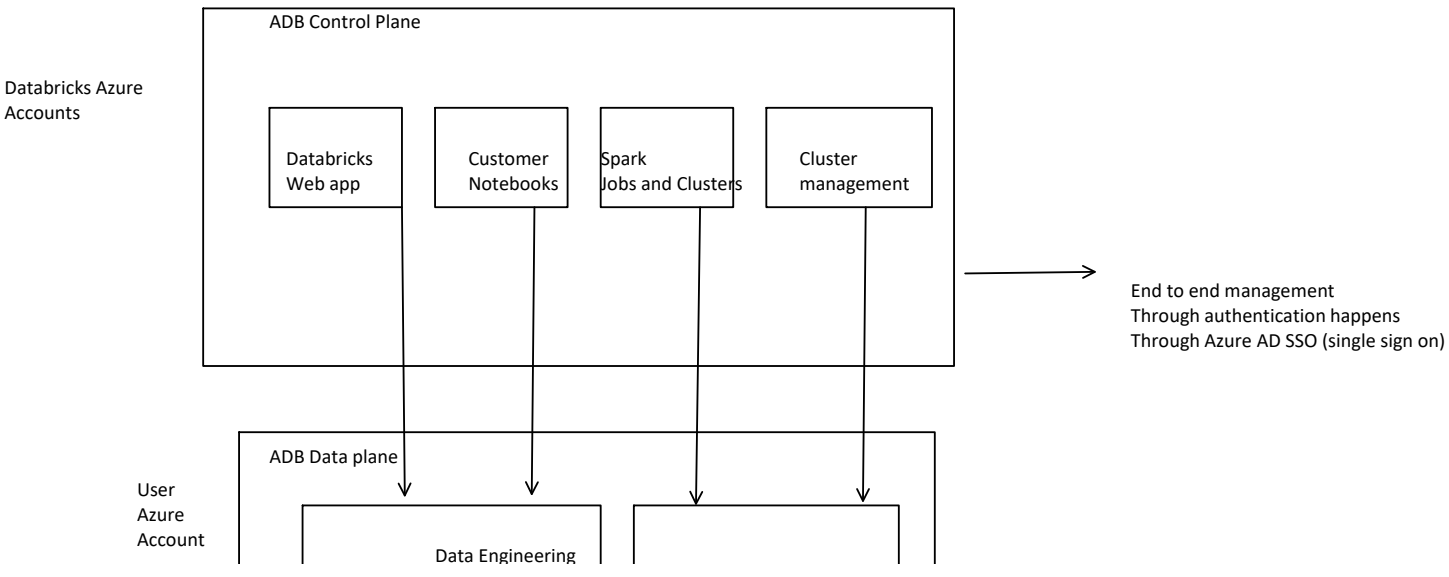
1. Write another Jupyter notebook, we'll perform city safety response analysis over Seattle fire departments data (import from public Azure blob storage account)
2. Perform the analysis through PySpark SQL

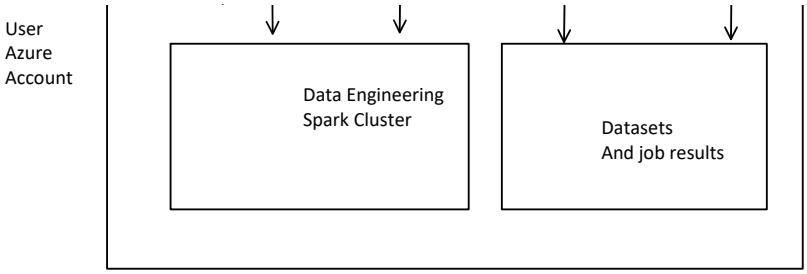
Hands-on Lab 03

1. Write a Graduate student income data analysis through Jupyter notebooks using the various python lib packages (matlotlib, plotly, numpy, scipy , pandas etc.)
2. Perform it using Jupyter notebook on Anaconda

Azure Databricks Architecture

Azure Databricks consists of Control Plane and the 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 or data science tasks.

Azure Databricks Control Plane

1. The control plane includes the backend services which Azure databricks manages 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 / customer's Azure account manages the ADB data plane, and there's the location where the data resides.
3. We can use the Azure Databricks connectors to connect clusters to external data sources outside the 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 in the data plane using own data sources. The data for the data analysis/engineering never being stored in the control plane. So that, customers are responsible to maintain the control and ownership of the data

2. Job results reside within the Azure storage account.

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

Overview of AWS

12 December 2022 19:58

Overview of GCP

12 December 2022 19:58

Case Study

12 December 2022 19:58

L1 Preparation

12 December 2022 19:59