

Data Analyst

Presentation by **Maheshkumar Paik**



Strictly private and confidential

Data Analyst

Agenda

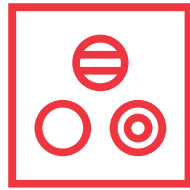
Date: Tuesday, 5 October 2021	
	Agenda
1	SQL Database
2	Break
3	SQL Lab Work
4	Break
5	Introduction to Big Data

Types of Databases

There are two type of database normally data engineers deals with:



Relational / SQL databases



Distributed / NoSQL databases



SQL Databases

SQL databases have a **structure** that allows us to identify and access **data in a relation** to another part of data in the database. Often, data is organized into **tables**, **row** and **columns**

Advantages:

- **Easy to use:** have rows and columns like Excel
- **Portable:** the same query can run on different flavours of SQL DB with minimal changes
- **Have well defined standards:** ANSI (American National Standard Institutes)
- Intuitive query language

Disadvantages:

- **Performance:** vertical scaling
- Not suitable for modern applications e.g. IoT, Big Data

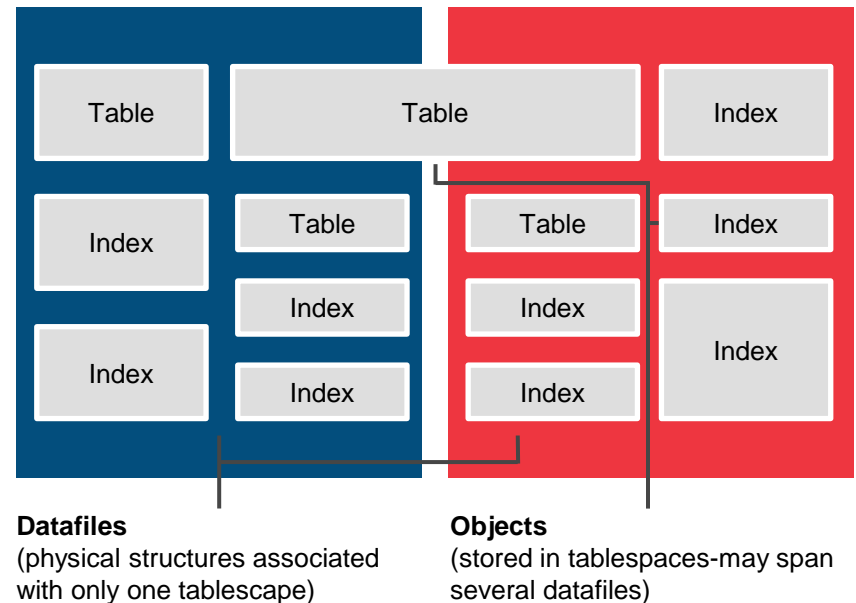
Some familiar SQL databases:



Relational database terminologies

- **Data file: Physical file on disk** created by the database application which contains the data
- **Schema: Collection of objects** in a database, including logical structures such as tables, views, sequences, stored procedures, synonyms, indexes, clusters, and database links
- **Table: Basic unit** of data storage. Data in tables is stored in **rows** and **columns** just like a tab in Excel
- **View:** A custom-tailored **presentation** of the data from **one or more tables**. Views do **not** actually store **data**, but derive it from the tables referenced in the view definition
- **Index:** An object that is used for **fast** and **efficient access** to stored information. Much like a table of contents in a book or a hyperlink

Tablespace
(one or more datafiles)

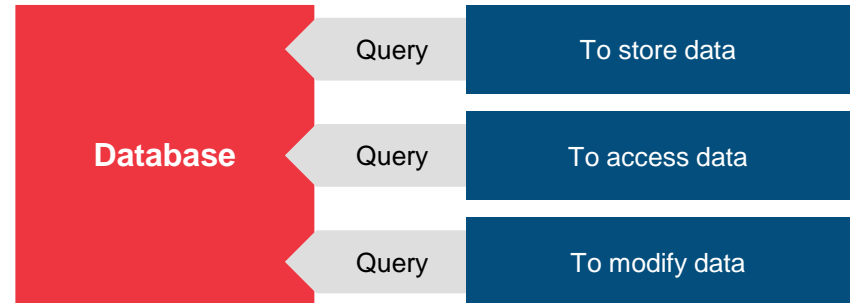


SQL Language

- SQL stands for **Structured Query Language**
- SQL is a standard language for **accessing** and **manipulating** databases
- Most database systems use SQL, some also have their own **additional proprietary extensions**
- SQL allows a **less-technical user** to use “**English-like**” language to “query” databases

Type of SQL statement

- **Data Definition Language (DDL)** – CREATE, DROP, RENAME, TRUNCATE
- **Data Manipulation Language (DML)** – SELECT, INSERT, UPDATE, DELETE
- **Data Control Language (DCL)** – GRANT, REVOKE
- **Transaction Control Language (TCL)** – COMMIT, ROLLBACK



SQL CRUD Operations

As a data engineer, you will be **frequently performing operations** that requires you to **read** from the database, **write** into the database, **update** the set of records in the tables and **delete** the records/transactions which are not required from the database.

CRUD is the acronym for the four basic commands in SQL -

C- Create command to create tables.

R- Read command to read data from tables.

U- Update command to update values in tables.

D- Delete command to delete rows from tables.

SQL Create

Create command is used to **create a new table** in the specified **database**.

There are multiple ways to create a table.

Syntax:

```
CREATE TABLE tableName (  
    column1 datatype,  
    column2 datatype,  
    column3 datatype,  
    ....  
)
```

```
SELECT column1, column2, column3, ...  
INTO newtable  
FROM oldtable  
WHERE condition;
```

Example:

```
CREATE TABLE Persons (  
    PersonID int,  
    LastName varchar(255),  
    FirstName varchar(255),  
    Address varchar(255),  
    City varchar(255)  
);
```

```
SELECT AddressID, AddressLine1, AddressLine2, City,  
PostalCode INTO Address  
FROM OLTP_Dataset.Person.Address  
WHERE AddressID < 100;
```

Each **row** in a table represents an **entity** and each **column** stores the **attributes defining an entity**. The above commands creates a tables named 'Persons' and 'Address', which stores data related to an entity in each row

SQL Read

Read operation in SQL is done using **SELECT** statements

Syntax:

```
SELECT [DISTINCT] column
FROM mytable [JOIN another_table
ON mytable.column = another_table.column
WHERE constraint_expression]
[GROUP BY column]
[ORDER BY column ASC/DESC]
[LIMIT count OFFSET COUNT];
```

```
//select all the columns/fields from table 'Address' in 'Person' database
SELECT * FROM Person.Address;
```

Output:

	AddressID	AddressLine1	AddressLin...	City	StateProvinc...	PostalCo...	SpatialLocation
1	1	1970 Napa Ct.	NULL	Bothell	79	98011	0xE6100000010CAE8BFC28BCE4474067A8918989
2	2	9833 Mt. Dias Blv.	NULL	Bothell	79	98011	0xE6100000010CD6FA851AE8D74740BC262A0A0
3	3	7484 Roundtree Drive	NULL	Bothell	79	98011	0xE6100000010C18E304C4ADE14740DA930C789:
4	4	9539 Glenside Dr	NULL	Bothell	79	98011	0xE6100000010C813A0D5F9FDE474011A5C28A70
5	5	1226 Shoe St.	NULL	Bothell	79	98011	0xE6100000010C61C64D8ABBD94740C460EA3FD

SQL Read contd.

Using WHERE clause example

//Get all the employee who are 'Design Engineer'

```
SELECT BusinessEntityID, JobTitle, LoginID FROM HumanResources.Employee WHERE JobTitle = 'Design Engineer';
```

//Get all the employee who are not 'Design Engineer'

```
SELECT BusinessEntityID, JobTitle, LoginID FROM HumanResources.Employee WHERE JobTitle <> 'Design Engineer';
```

//Display details of the persons whose record modified in data range

```
SELECT BusinessEntityID, FirstName, MiddleName, LastName, ModifiedDate FROM Person.Person  
WHERE ModifiedDate BETWEEN '2009-01-01' AND '2013-12-31';
```

//Display Product Id and Its Name where name of the product has 'Bike' string

```
SELECT ProductID, Name FROM Production.Product WHERE Name LIKE '%Bike%';
```

Using GROUP BY clause example

//Get the total number of item ordered for each product

```
SELECT SUM(OrderQty) AS Total, ProductID FROM Sales.SalesOrderDetail GROUP BY ProductID;
```

//Display count of orders placed by year for each customer

```
SELECT CustomerID, COUNT(*) AS SalesCount, YEAR(OrderDate) AS OrderYear FROM Sales.SalesOrderHeader GROUP BY CustomerID,  
YEAR(OrderDate);
```

//Get the product Id ordered more than 5000 times

```
SELECT SUM(OrderQty) AS TotalOrdered, ProductID FROM Sales.SalesOrderDetail GROUP BY ProductID HAVING SUM(OrderQty) > 5000 ;
```

SQL Read contd.

Using ORDER BY examples

//Sort the record by last name Ascending order

```
SELECT BusinessEntityID, LastName, FirstName, MiddleName FROM Person.Person  
ORDER BY LastName ASC
```

//Sort the record by last name Descending order

```
SELECT BusinessEntityID, LastName, FirstName, MiddleName FROM Person.Person  
ORDER BY LastName DESC
```

Using JOIN examples

To **Join** any **two or more tables** there should be **relationship** between them which is defined by below keys

- **Primary Key**- A field in the table that **uniquely identify** a record in the table.
- **Foreign Key**- A field in the table that is **primary key in another table**.

//Get personal information of the person working in a company as an employee

```
SELECT JobTitle, BirthDate, FirstName, LastName FROM HumanResources.Employee AS E  
INNER JOIN Person.Person AS P ON E.BusinessEntityID = P.BusinessEntityID;
```

//Displays all the products along with the SalesOrderID even if an order has never been placed for that product

```
SELECT SalesOrderID, P.ProductID, P.Name FROM Production.Product AS P LEFT JOIN Sales.SalesOrderDetail AS SO ON P.ProductID =  
SO.ProductID;
```

SQL Read contd.

Using all the clauses we learnt..

Select city, postal code and modified date from table 'Address' and AddressTypeID from table 'BusinessEntityAddress', where AddressTypeID is equal to 3, grouping the result by city, postal code, modified date, address type id. The final result is sorted by postal code in descending order.

```
SELECT A.City, A.PostalCode, A.ModifiedDate, B.AddressTypeID
FROM Person.Address as A
JOIN Person.BusinessEntityAddress as B
ON A.AddressID=B.AddressID
WHERE B.AddressTypeID=3
GROUP BY A.City, A.PostalCode, A.ModifiedDate, B.AddressTypeID
ORDER BY A.PostalCode DESC;
```

SQL Update

SQL **UPDATE** command **modifies** the **existing one** or **more records** in a table based on condition stated in WHERE clause.

Syntax:

```
UPDATE table_name SET column1 = value1, column2 = value2..., columnN = valueN WHERE [condition];
```

Example:

Update the column named 'City' for 'AddressID' = 2

	AddressID	AddressLine1	AddressLin...	City	StateProvinc...	PostalCo...	SpatialLocat
1	1	1970 Napa Ct.	NULL	Bothell	79	98011	0xE610000010
2	2	9833 Mt. Dias Blv.	NULL	Bothell	79	98011	0xE610000010

```
UPDATE Person.Address SET City='Boston' WHERE AddressID=2;
```

Messages							
(1 row(s) affected)							
	Address...	AddressLine1	AddressLin...	City	StateProvinc...	PostalCo...	SpatialLocat
1	1	1970 Napa Ct.	NULL	Bothell	79	98011	0xE6100000
2	2	9833 Mt. Dias Blv.	NULL	Boston	79	98011	0xE6100000
3	3	7484 Roundtree Drive	NULL	Bothell	79	98011	0xE6100000

SQL Delete

The SQL **DELETE** command is used to **delete the existing one or more records** from a table based on a condition.

Syntax:

```
DELETE FROM table_name WHERE [condition];
```

Example:

```
SELECT * FROM Person.Address;
```

	Address...	AddressLine1	AddressLin...	City	StateProvinc...	PostalCo...	SpatialLocation
1	1	1970 Napa Ct.	NULL	Bothell	79	98011	0xE6100000010CAE8BFC28E
2	2	9833 Mt. Dias Blv.	NULL	Boston	79	98011	0xE6100000010CD6FA851AE
3	3	7484 Roundtree Drive	NULL	Bothell	79	98011	0xE6100000010C18E304C4A
4	4	9539 Glenside Dr	NULL	Bothell	79	98011	0xE6100000010C813A0D5FE

```
DELETE FROM Person.Address WHERE City='Boston';
```

	Address...	AddressLine1	AddressLin...	City	StateProvinc...	PostalCo...	SpatialLocation
1	1	1970 Napa Ct.	NULL	Bothell	79	98011	0xE6100000010CAE8BF
2	3	7484 Roundtree Drive	NULL	Bothell	79	98011	0xE6100000010C18E30
3	4	9539 Glenside Dr	NULL	Bothell	79	98011	0xE6100000010C813A0
4	5	1226 Shoe St.	NULL	Bothell	79	98011	0xE6100000010C61C64

Workshop on Relational Databases

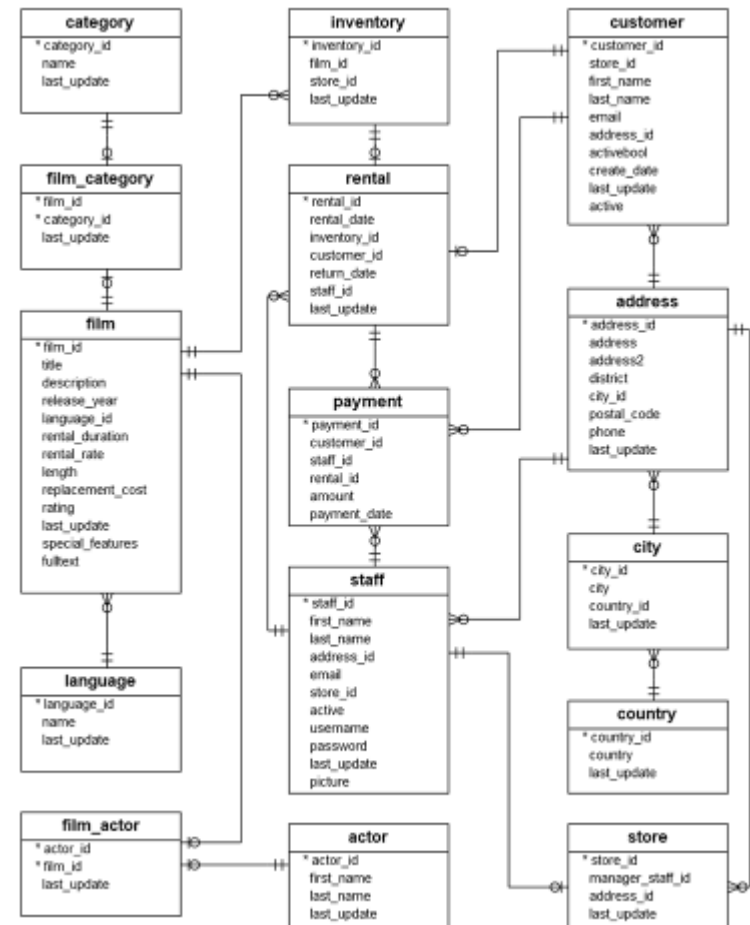
Case Scenario:

The database DvdRental has 15 tables. Below are the different tables and a brief description of them.

Questions:

- What are the top and least rented (in-demand) genres and what are their total sales?
- Can we know how many distinct users have rented each genre?
- What is the average rental rate for each genre? (from the highest to the lowest)
- How many rented films were returned late, early, and on time?
- In which countries does Rent A Film have a presence and what is the customer base in each country? What are the total sales in each country? (from most to least)
- Who are the top 5 customers per total sales and can we get their details just in case Rent A Film wants to reward them?

DVD Rental ER Model



NoSQL Databases

Are **non-relational** databases with **specific data models**, **flexible schemas** and **scale horizontally**

Advantages:

- **Schema-less**: provide a higher level of flexibility with newer data models
- **Open Source & Low-Cost**: go-to solution for organizations with limited budgets
- **Elastic scalability**: NoSQL databases are designed to function on **full throttle even with low-cost hardware**
- **Less development time**: create a database without needing to develop a **detailed** (fine-grained) **database model**

Disadvantages:

- ✓ community-driven, with enterprise support
- ✓ **Lack of standardization** like SQL
- ✓ Different database for different business cases

Familiar NoSQL Databases:




CAP Theorem

CAP theorem states **3 basic requirements** when designing application in distributed architecture



C

Data in the database will **remain consistent** after the execution of an **operation**. E.g. All client sees the same data after an update operation is performed



A

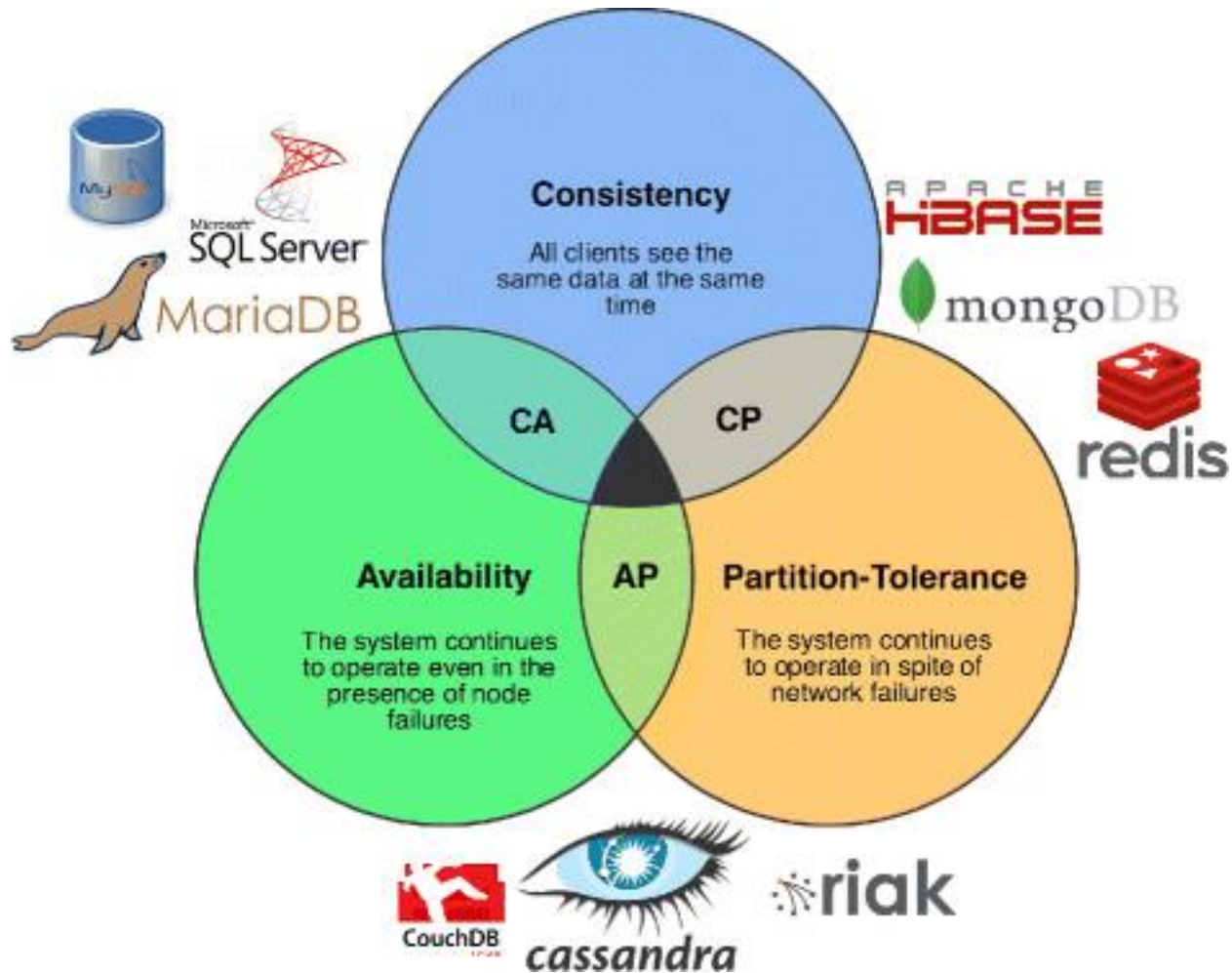
No downtime. This means the system will be always available to server the request from the client applications



P

Data will **be partitioned into multiple group** and **replicated across multiple system**. System **continues to function** even though any one of the system is unreliable or loss communication among system

CAP Theorem contd.



Deciding factors to choose SQL vs NoSQL Databases

Parameters	SQL	NoSQL
Definition	Relation database	Non-relational or distributed database
Type	Table	document, key-value pair, graph, columnar
Schema	Pre-defined Schema	Dynamic schema, schema-less
Ability to Scale	Vertically scalable	Horizontally scalable
Query Language	SQL	UnQL- unstructured query language varies from DB to DB
Standards	Emphasis on ACID (Atomicity, Consistency, Isolation, Durability)	Follow CAP theorem (Consistency, Availability, Partition)
Storage type	High available storage	Commodity drive storage
Open-source	Mix	All
Application/Use cases	<ul style="list-style-type: none">• Online Banking• Order and sales management• Data warehousing	<ul style="list-style-type: none">• Web applications• Social networks application• Internet of Things

Quick Quiz



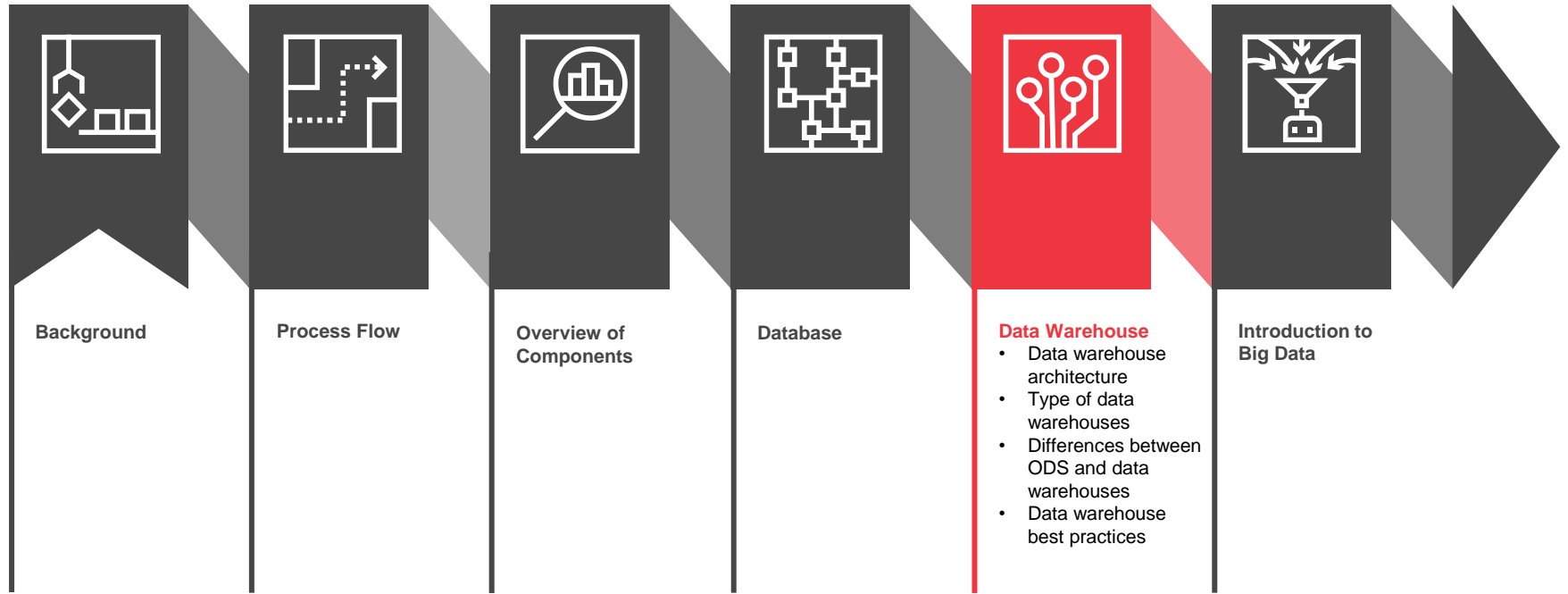
Please describe a circumstance in which an organization would choose a non-relational database over a relational database



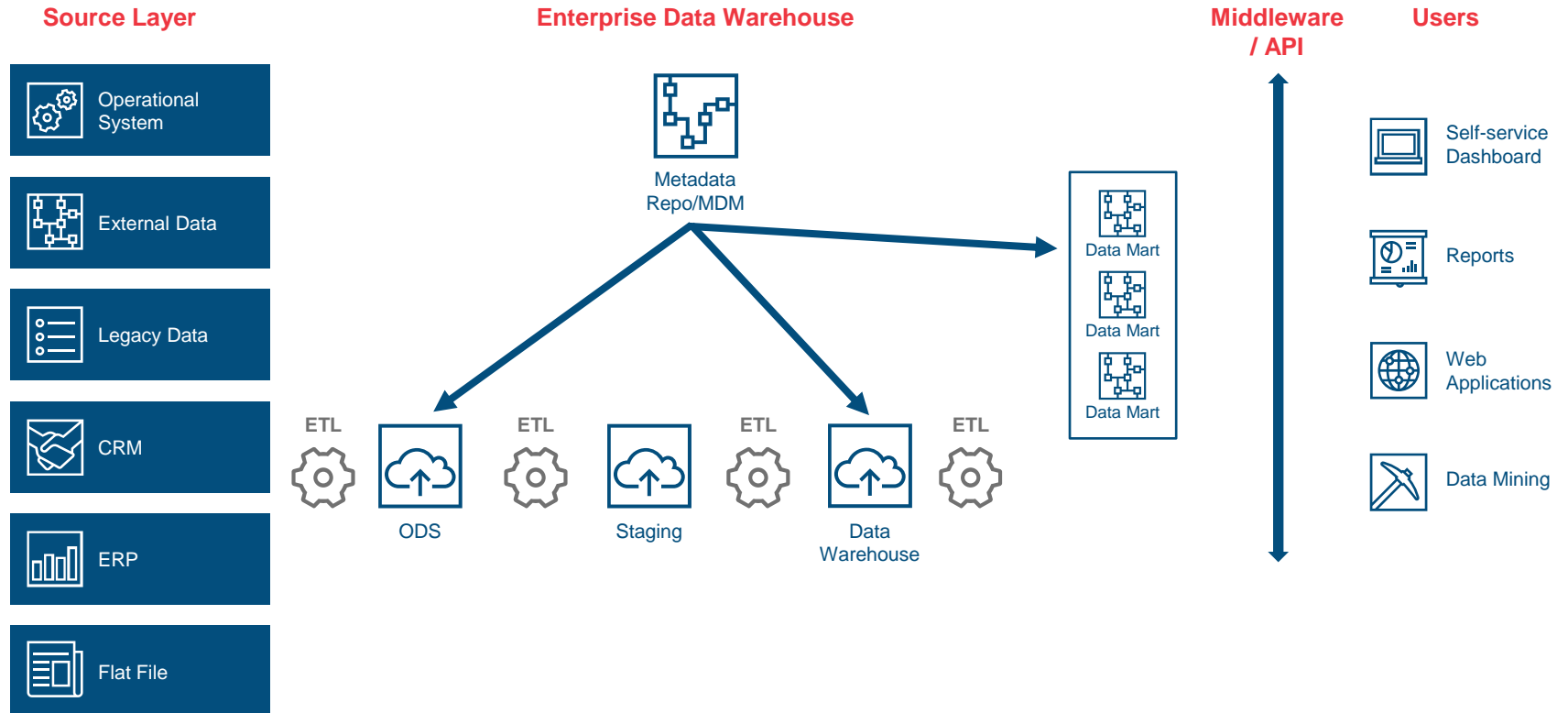
What are some advantages of SQL databases?



Day 2



Data Warehouse architecture



Data warehousing best practices

Define standards before starting – Templates for mapping, coding guidelines, documentation should be decided ahead. Timelines for status reports, release deliverables should be decided upfront before the development starts

Data Governance – Ensure data integrity and quality before the data is pushed into the data warehouse

User need based design – Primary focus of the data warehouse is to provide trusted information to the customers and address issues which are not fully articulated by the customers, this results in increased business

Faster delivery – Markets change rapidly so requirements gathering and delivery of the product should be done in an agile manner

Performance – Storage optimization and query performance tuning to account for the rapid growth of data

Adopt **agile data warehouse methodology** which breaks projects into smaller, faster deliverables

Automation – Automate the data pipeline/ETL process to leverage IT resource fully and iterate faster project execution

Introduction to Big Data

Big Data is **high-volume**, **high-velocity** and/or **high-variety** information assets that demand **cost-effective**, **innovative** forms of information processing that enable enhanced insight, decision making, and process automation.

Source: Gartner

Is a 100GB data set a “Big Data”?

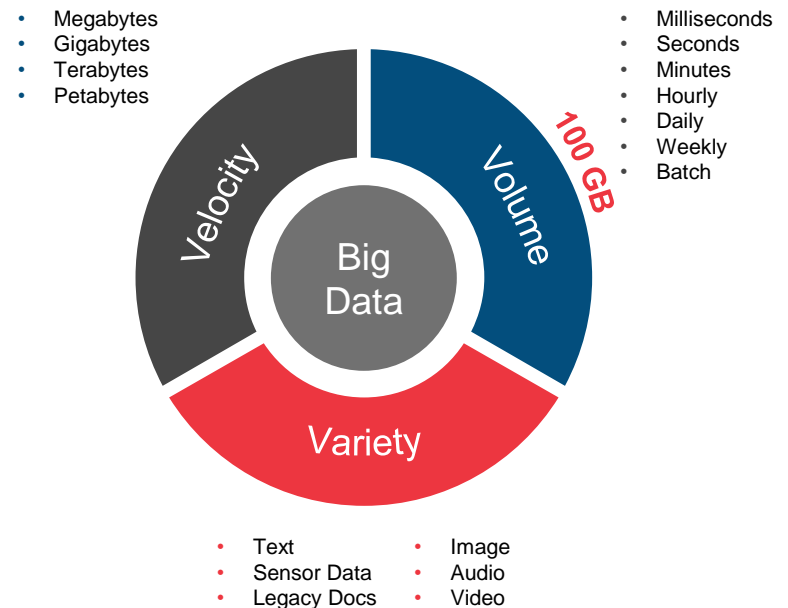
If we quickly analyse the definition, we will see 3 characteristic

- Volume
- Velocity
- Variety

How to do it?

- 100GB of structured data is easily managed by traditional data storage, so it would not be considered a **Big Data Volume**. When we start talking about 100TB or PB, it becomes Big Data
- Processing 100GB of data per minute (**Velocity**), or analysing 100GB of unstructured (social media, image, etc.) data (**Variety**) at the same speed could be Big Data
- **Cost-effective solution** for the problem created by a **combination of 3Vs**
- Requires **innovative thinking** and **use of innovative tools and techniques**.

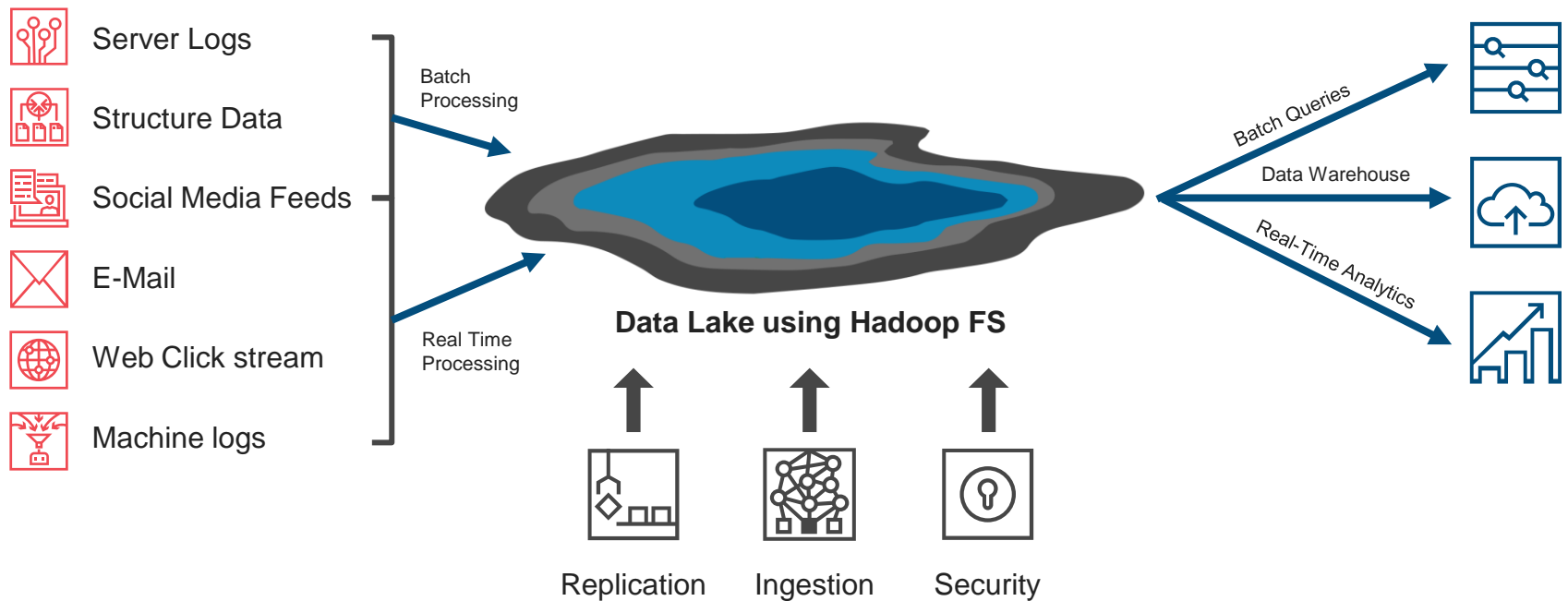
If your question fits into this definition, you have a **big data problem**



What is a Data Lake?

A data lake is a **centralized repository** that allows you to store all our **structured and unstructured data at any scale**. It stores data in a raw format, and runs different types of analytics; from dashboards and visualizations to **big data processing, real-time analytics, and machine learning** to guide better decisions.

Source: AWS



Difference between data warehouse and data lake

Basis of Differences	Data Warehouse	Data Lake
Types of data	Stores data in the tables, row and columns	Stores raw data (Structured/Unstructured/Semi-Structured) in its native format.
User	Business professionals	Data scientists
Processing	Schema-on-write, cleansed data, structured	Schema-on-Read, raw data in native format
Agility	Less agile, fixed configuration	Configuration and reconfiguration are done when required, highly agile
Reporting and analysis	Slow and expensive	Low storage, economical
Cost	Expensive storage	Low-cost storage
Security	Mature	Maturing

End of Day

