SQL : Basic to Advance

History of SQL:

SQL, short for Structured Query Language, has a history rooted in the 1970s when IBM developed the System R project. It aimed to create a database management system that could handle and manipulate structured data efficiently. As part of System R, IBM introduced a language called Structured English Query Language (SEQUEL), which later evolved into SQL The name "SEQUEL" was already taken by another company. The creators of the language didn't want to get sued, so they changed the name to 'SQL'. SQL is a backronym for "Structured Query Language." The language was influenced by Edgar F. Codd's relational model, which introduced the concept of organizing data into tables with relationships. Over time, SQL became the standard language for interacting with relational databases and was standardized by ANSI and ISO. Today, SQL is widely adopted and utilized in various database systems, enabling organizations to manage and analyze structured data effectively.

Let's first understand the Data modeling and database:

What is Data Modeling?

Data Modeling is a critical step in defining data structure, creating data models to describe associations and constraints for reuse. Data model is the conceptual design or plan for organizing data. It visually represents data with diagrams, symbols, or text to visualize relationships.
Enhancing data analytics, data modeling assures uniformity in nomenclature, rules, semantics, and security. Regardless of the application, the emphasis is on the arrangement and accessibility of the data.

Advantages of Data Modelling

The following are the essential advantages of Data Modelling:

- The data model helps us choose the right data sources to populate the model.
- The Data Model improves communication throughout the company.
- The data model aids in the ETL process's documentation of the data mapping.
- We can use data modeling to query the database's data and generate various reports based on the data. Data modeling supports data analysis through reports.

Data Modeling Terminology:

- **Entity:** Entities are the objects or items in the business environment for which we need to store data. Defines what the table is about. For example, in an e-commerce system, entities can be Customers, Orders, or Products.
- **Attribute:** Attributes provide a way to structure and organize the data within an entity. They represent the specific characteristics or properties of an entity. For instance, attributes of a Customer entity can include Name, Address, and Email.
- **Relationship:** Relationships define how entities are connected or associated with each other. They explain the interactions and dependencies between entities. For example, the relationship between Customers and Orders represents that a customer can place multiple orders.
- **Reference Table:** A reference table is used to resolve many-to-many relationships between entities. It transforms the relationship into one-to-many or many-to-one relationships. For instance, in a system with Customers and Products, a reference table called OrderDetails can be used to link customers with the products they have ordered.

- **Database Logical Design:** It refers to designing the database within a specific data model of a database management system. It involves creating the structure, relationships, and rules of the database at a conceptual level.
- **Logical Design:** Logical design involves the creation of keys, tables, rules, and constraints within the database. It focuses on the logical structure and organization of data without considering specific implementation details.
- **Database Physical Design:** It encompasses the decisions related to file organization, storage design, and indexing techniques for efficient data storage and retrieval within the database.
- **Physical Model:** The physical model is a representation of the database that considers implementation-specific details such as file formats, storage mechanisms, and index structures. It translates the logical design into an actual database implementation.
- **Schema:** A schema refers to the complete description or blueprint of the database. It defines the structure, relationships, constraints, and permissions associated with the database objects.
- **Logical Schema:** A logical schema represents the theoretical design of a database. It is typically created during the initial stages of database design, similar to drawing a structural diagram of a house. It helps visualize the relationships and organization of

The database entities and attributes

Levels of Data Abstraction:

Data modeling typically involves several levels of abstraction, including:

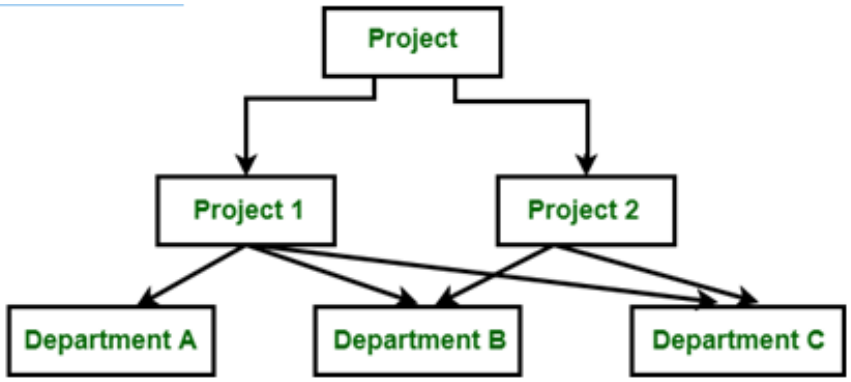
Conceptual level: This is the highest level of abstraction. It's about what data you need to store, and how it relates to each other. You can use diagrams or other visual representations to show this.
Example: You might decide that you need to store data about customers, products, and orders. You might also decide that there is a relationship between customers and orders, and between products and orders.
Logical level: This is the middle level of abstraction. It's about how the data will be stored and organized. You can use data modeling languages like SQL or ER diagrams to show this.
Example: You might decide to store the data in a relational database. You might create tables for customers, products, and orders, and define relationships between the tables.
Physical level: The physical level is the most basic or lowest abstraction. It concerns the particulars of how the data will be kept on disc. Data types, indexes, and other technical information fall under this category.
Example: You might decide to store the data in a specific database server, and use a specific data type for each column. You might also decide to create indexes to improve the performance of queries.

Important perspectives of a data model:

1. **Network Model:**

   • The Network Model represents data as interconnected records with predefined relationships. It allows for many-to-many relationships and uses a graph-like structure. For example, in a company's database, employees can work on multiple projects, and each project can have multiple employees assigned to it. The Network Model connects employee records to project records through relationship pointers, enabling flexible relationships.

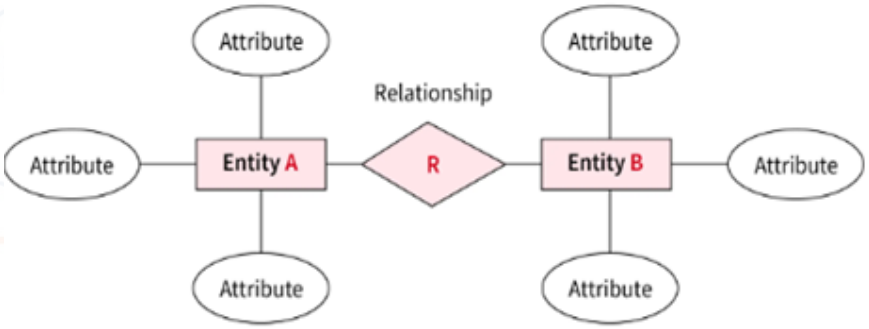
Check the below image:



2. Entity-Relationship (ER) Model:

- The ER Model represents data using entities, attributes, and relationships. Entities are real-world objects, attributes describe their properties, and relationships depict connections between entities. For instance, in a university database, entities could be Students and Courses, with attributes like student ID and course name. Relationships, such as "Student enrolls in Course," illustrate the associations between entities.

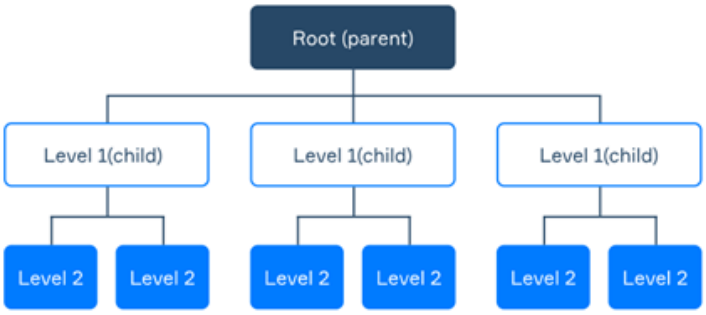
Check the below image:



3. Hierarchical Model:

- Data is arranged in a tree-like structure with parent-child relationships using the hierarchical model. There is one parent and several children per record. Consider the organizational hierarchy, where the CEO is at the top and is followed by the managers, employees, and department heads. These hierarchical links are graphically represented by the hierarchical model, allowing top-to-bottom or bottom-to-top navigation.

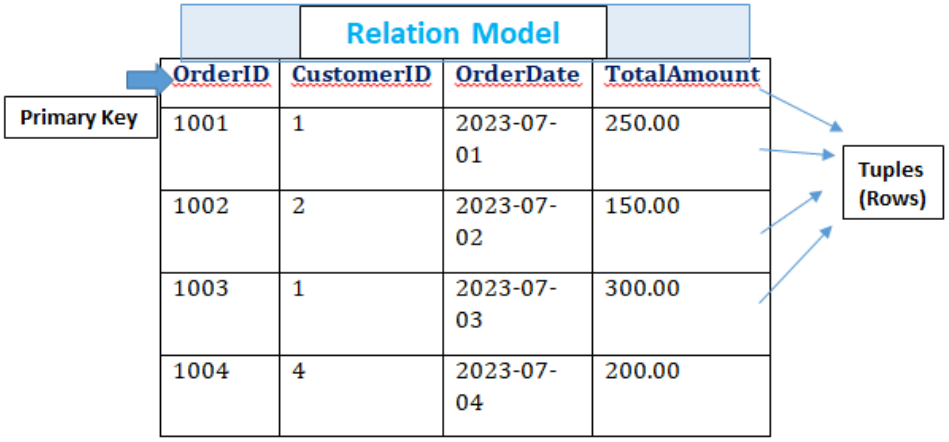
Check the below image:



4. Relational Model:

- The Relational Model organizes data into tables consisting of rows and columns. It create relationships between tables using primary and foreign keys. For example, in a customer and orders scenario, customer information is stored in one table, while order details are stored in another. The Relational Model connects the tables using a shared key, like a customer ID, to link relevant records.

Check the below image:



OrderID	CustomerID	OrderDate	TotalAmount
1001	1	2023-07-01	250.00
1002	2	2023-07-02	150.00
1003	1	2023-07-03	300.00
1004	4	2023-07-04	200.00

What is Database?

A database is like a digital warehouse where data is stored, organized, and managed efficiently. Database is a physical or digital storage system that implements a specific data model. Database is the actual implementation of that design, where the data is stored and managed. It acts as a central hub for information, making it easier to access and analyze data.

Key Components of a Database:

- **Data:** Data is the raw information stored in a database, such as customer details, product information, or financial records. It can be in different formats like text, numbers, dates, or images.
- **Tables:** Tables are like virtual spreadsheets within a database. They have rows and columns, where each row represents a specific record or instance, and each column represents a particular piece of data. For example, a table for customers may have columns like ID, Name, Address, and Contact.
- **Relationships:** Relationships define how tables are connected within a database. They establish associations based on shared data elements. For instance, a customer's ID in one table can be linked to their orders in another table. This helps maintain data consistency and enables efficient data retrieval.
- **Queries:** Queries are like search commands that allow users to extract specific data from the database. Users can search, filter, and sort data based on the criteria they specify. For example, a query can be used to find all customers who made a purchase in the last month.

There are various types of databases used in different scenarios based on their design and functionality. Here are some common types of databases:

- **Relational Databases (RDBMS):** Relational databases separate data into rows and columns and tables, and they build associations between tables using keys. They are frequently used in business applications that manipulate data using Structured Query Language (SQL). Oracle, MySQL, and Microsoft SQL Server are a few examples.
- **NoSQL Databases:** NoSQL databases are Non-relational databases that offer flexible data models without using a fixed schema. They can manage enormous amounts of semi- or unstructured data. MongoDB, Cassandra, and Couchbase are a few examples.
- **Object-Oriented Databases:** Similar to object-oriented programming, OODBs store data as objects. They are helpful for programs that deal with intricate interactions and data structures. Examples are ObjectDB and db4o.
- **Hierarchical Databases:** Hierarchical databases organize data in a tree-like structure, where each record has a parent-child relationship. They are suitable for representing hierarchical relationships, such as organization structures. IMS (Information Management System) is an example of a hierarchical database.
- **Network Databases:** Network databases are similar to hierarchical databases but allow for more complex relationships. They use a network model where records can have multiple parent and child records. CODASYL DBMS (Conference on Data Systems Languages) is an example of a network database.
- **Graph Databases:** Graph databases store data in a graph structure with nodes and edges. They are designed to represent and process relationships between data points efficiently. They are commonly used for social networks, recommendation engines, and network analysis. Examples include Neo4j and Amazon Neptune.
- **In-Memory Databases:** In-memory databases store data primarily in memory, resulting in faster data access and processing compared to disk-based databases. They are suitable for applications that require high-speed data operations. Examples include Redis and Apache Ignite.
- **Time-Series Databases:** Time-series databases are optimized for storing and retrieving time-stamped data, such as sensor data, financial data, or log files. They provide efficient storage and retrieval of time-series data for analysis. Examples include InfluxDB and Prometheus.

Let's Understand DBMS and RDBMS:

Although they are both software system used to manage databases, DBMS (Database Management System) and RDBMS (Relational Database Management System) have different qualities.

Why required DBMS?

A DBMS is required to efficiently manage the flow of data within an organization. It handles tasks such as inserting data into the database and retrieving data from it. The DBMS ensures the consistency and integrity of the data, as well as the speed at which data can be accessed.

Why required RDMS?

Similarly, an RDBMS is required when we want to manage data in a relational manner, using tables and relationships. It helps in reducing data duplication and maintaining the integrity of the database. RDBMS ensures that data is stored in a structured manner, allowing for efficient querying and retrieval.

Differences between DBMS and RDBMS:

DBMS	RDBMS
Applications using DBMS save data in files.	RDBMS applications store data in a tabular form.
No relationship between data.	Related data stored in the form of table.
Normalization is not present.	**Normalization** is present.
Distributed databases are not supported by DBMS.	RDBMS supports distributed database.
It works with small quantity of data.	It works with large amount of data.
Security is less	More security measures provided.
Low software and hardware necessities	Higher software and hardware necessities.
Examples: XML Window Registry, Forxpro, dbaseIIIplus etc.	Examples: PostgreSQL, MySQL, Oracle, Microsoft Access, SQL Server etc.

What is Normalization?

The process of normalization data in a database ensures data integrity by removing duplication. A database must be split up into various tables, and linkages must be established between them. Different levels of normalization exist, such as 1NF (First Normal Form), 2NF (Second Normal Form), 3NF (Third Normal Form), and BCNF (Boyce-Codd Normal Form)

1NF (First Normal Form):

In 1NF, each column in a table contains only atomic values, meaning it cannot be further divided. There should be no repeating groups or arrays of values within a single column. Each row in the table should be uniquely identifiable. Here's an example:

Original Table:

CustomerID	Name	Phone no.
1	Jia ria	8978847383
2	John Doe	7899748899, 8899278299
3	Smith tie	9877382892

1NF Table:

CustomerID	Name	Phone no.
1	Jia ria	8978847383
2	John Doe	7899748899
3	Smith tie	9877382892
4	John Doe	8899278299

2NF (Second Normal Form):

In 2NF, the table is already in 1NF, and each non-key column is dependent on the entire primary key. If there are partial dependencies, those columns should be moved to a separate table. Here's an example:

Original Table:

OrderID	ProductID	ProductName	Category	Price
1	1	Laptop	Electronics	1000
2	2	Smartphone	Electronics	800
3	3	Laptop	Electronics	900

2NF Tables:

Table 1: Products

ProductID	ProductName	Category
1	Laptop	Electronics
2	Smartphone	Electronics

Table 2: Orders

OrderID	ProductID	Price
1	1	1000
2	2	800
3	1	900

3NF (Third Normal Form):

In 3NF, the table is already in 2NF, and there are no transitive dependencies. Non-key columns should not depend on other non-key columns. If there are such dependencies, those columns should be moved to a separate table. Here's an example:

Original Table:

CustomerID	OrderID	ProductID	Price	CustomerName	CustomerEmail
1	1	1	1000	John Doe	john@example.com
2	2	2	800	Jane Smith	jane@example.com
3	3	1	900	John Doe	johndoe@example.com

3NF Tables:

Table 1: Customers

CustomerID	CustomerName	CustomerEmail
1	John Doe	john@example.com
2	Jane Smith	jane@example.com
3	John Doe	johndoe@example.com

Table 2: Product

ProductID	ProductName
1	Laptop
2	Smartphone

BCNF (Boyce-Codd Normal Form):

BCNF is an advanced form of normalization that addresses certain anomalies that can occur in 3NF. It ensures that there are no non-trivial functional dependencies of non-key attributes on a candidate key. Achieving BCNF involves decomposing tables further if necessary.

Binary Relationships:

A binary relationship exists when two different relationship are involved. Accordingly, every entity in the connection has a unique association with one entity in the other entity. For instance, a passport can only be issued to one individual, and a person is only allowed to have one passport at a time. An illustration of a one-to-one relationship might be this.

Cardinality:

Cardinality refers to the number of instances of an entity that can be associated with an instance of another entity. There are four types of cardinality:

One-to-One

Each instance of one entity can only be linked to one instance of the other, and vice versa, in a one-to-one relationship. It is commonly used to represent one-to-one things in the actual world, such a person and their passport, and is the strictest kind of relationship.

Example: A person can have only one passport, and a passport can be issued to only one person.

One-to-Many

A one-to-many relationship means that each instance of one entity can be associated with multiple instances of the other entity, but each instance of the other entity can only be associated with one instance of the first entity. This is a common type of relationship, and it is often used to represent hierarchical relationships in the real world, such as a parent and their children.

Example: A customer can place multiple orders, but each order can only be placed by one customer.

Many-to-One

An entity in A is associated to no more than one entity in B in this particular cardinality mapping. Or, we may say that any number (zero or more) of entities or things in A can be connected to a unit or thing in B.

Example: A single surgeon performs many operations in a specific institution. A many-to-one relationship is one of these relationships.

Many-to-Many

A many-to-many relationship means that each instance of one entity can be associated with multiple instances of the other entity, and each instance of the other entity can also be associated with multiple instances of the first entity. This is the most common type of relationship, and it is often used to represent relationships where the order of the entities does matter, such as a student and their courses.

Example: A student can take multiple courses, and each course can be taken by multiple students.

Introduction to SQL:

The computer programming language SQL (Structured Query Language) was developed especially for managing and changing relational databases. It provides commands and statements to connect to databases, retrieve and modify data, construct database structures, and perform numerous data tasks. More details are provided below:

Definition and Purpose of SQL:

According to its definition and intended application, SQL is a declarative language utilized for relational database management. By constructing queries, it enables users to interact with databases to access, alter, and manage structured data. No matter what database management system is used underneath, SQL provides a standardized and efficient method for working with databases.

Why SQL?

Due to its adaptability and efficiency in maintaining relational databases, SQL is frequently used in data science and analytics. The main justifications for SQL's high value are as follows:

- The core activities of inserting, updating, and deleting data in relational databases are made available to data professionals via SQ. It gives a simple and effective method for changing data.
- SQL gives customers the ability to get particular data from relational database management systems. Users can provide criteria and conditions to retrieve the desired information by creating SQL queries.
- SQL is useful for expressing the structure of stored data. Users can define the structure, data types, and relationships of database tables as well as add, change, and delete them.
- SQL gives users the ability to handle databases and their tables efficiently. In order to increase the functionality and automation of database operations, it facilitates the construction of views, stored procedures, and functions.
- SQL gives users the ability to define and edit data that is kept in a relational database Data constraints can be specified by users, preserving the integrity and consistency of the data.
- Data Security and Permissions: SQL has tools for granting access to and imposing restrictions on table fields, views, and stored procedures. Giving users the proper access rights promotes data security.