Unit_02 - Database Architecture

Introduction to Databases

What is Data?

- Data is a collection of a distinct small unit of information. It can be used in a variety of forms like text, numbers, media, bytes, etc. it can be stored in pieces of paper or electronic memory, etc.
- In computing, Data is information that can be translated into a form for efficient movement and processing. Data is interchangeable.

What is Database?

- The database is a collection of inter-related data which is used to retrieve, insert and delete the data efficiently. It is also used to organize the data in the form of a table, schema, views, and reports, etc.
- For example: The college Database organizes the data about the admin, staff, students and faculty etc.

What is DBMS?

- Database management system is a software which is used to manage the database.
- It consists of a group of programs which manipulate the database.
- DBMS provides an interface to perform various operations like database creation, storing data in it, updating data, creating a table in the database.
- It provides protection and security to the database. In the case of multiple users, it also maintains data consistency.

Characteristics of DBMS

- **Real-world entity:** Modern DBMS are more realistic and uses real world entities to design its architecture. It uses the behavior and attributes too. For example, a school database may use student as entity and their age as their attribute.
- **Relation-based tables:** DBMS allows entities and relations among them to form as tables. This eases the concept of data saving. A user can understand the architecture of database just by looking at table names etc.
- **Isolation of data and application:** A database system is entirely different than its data. Where database is said to active entity, data is said to be passive one on which the database works and organizes. DBMS also stores metadata which is data about data, to ease its own process.
- **Less redundancy:** DBMS follows rules of normalization, which splits a relation when any of its attributes is having redundancy in values. Following normalization, which itself is a mathematically rich and scientific process, make the entire database to contain as less redundancy as possible.
- **Consistency:** DBMS always enjoy the state on consistency where the previous form of data storing applications like file processing does not guarantee this. Consistency

is a state where every relation in database remains consistent. There exist methods and techniques, which can detect attempt of leaving database in inconsistent state.

- **Query Language:** DBMS is equipped with query language, which makes it more efficient to retrieve and manipulate data. A user can apply as many and different filtering options, as he or she wants. Traditionally it was not possible where file-processing system was used.
- **ACID Properties:** DBMS follows the concepts for ACID properties, which stands for Atomicity, Consistency, Isolation and Durability. These concepts are applied on transactions, which manipulate data in database. ACID properties maintains database in healthy state in multi-transactional environment and in case of failure.
- **Multiuser and Concurrent Access:** DBMS support multi-user environment and allows them to access and manipulate data in parallel. Though there are restrictions on transactions when they attempt to handle same data item, but users are always unaware of them.
- **Multiple views:** DBMS offers multiples views for different users. A user who is in sales department will have a different view of database than a person working in production department. This enables user to have a concentrate view of database according to their requirements.
- **Security:**
    - The Database should be accessible to the users in a limited way.
    - The access to make changes to a database by the user should be limited, and the users must not be given complete access to the entire Database.
    - Unauthorized users should not be allowed to access the Database.
    - Authentication: The DBMS has authentication for various users that directly refers to the limit to which the user can access the Database.

Advantages of DBMS

- ✓ Redundancy problem can be solved.
- ✓ Has a very high security level.
- ✓ Presence of Data integrity.
- ✓ Support multiple users.
- ✓ Avoidance of inconsistency.
- ✓ Shared data
- ✓ Any unauthorized access is restricted
- ✓ Provide backup of data

Disadvantages of DBMS

- ✓ Complexity
- ✓ Size
- ✓ Performance
- ✓ Higher impact of a failure
- ✓ Cost of DBMS

File System vs DBMS

- A file system is a technique of arranging the files in a storage medium like a hard disk, pen drive, DVD, etc. It helps user to organize the data and allows easy retrieval of files when they are required. It mostly consists of different types of files like mp3, mp4, txt, doc, etc. that are grouped into directories.
- A file system enables you to handle the way of reading and writing data to the storage medium. It is directly installed into the computer with the Operating systems such as Windows and Linux.
- Database Management System (DBMS) is a software for storing and retrieving user's data while considering appropriate security measures. It consists of a group of programs that manipulate the database. The DBMS accepts the request for data from an application and instructs the DBMS engine to provide the specific data. In large systems, a DBMS helps users and other third-party software to store and retrieve data.

File System	DBMS
A file system is a software that manages and organizes the files in a storage medium. It controls how data is stored and retrieved.	DBMS or Database Management System is a software application. It is used for accessing, creating, and managing databases.
The file system provides the details of data representation and storage of data.	DBMS gives an abstract view of data that hides the details
Storing and retrieving of data can't be done efficiently in a file system.	DBMS is efficient to use as there are a wide variety of methods to store and retrieve data.
It does not offer data recovery processes.	There is a backup recovery for data in DBMS.
The file system doesn't have a crash recovery mechanism.	DBMS provides a crash recovery mechanism
Protecting a file system is very difficult.	DBMS offers good protection mechanism.
In a file management system, the redundancy of data is greater.	The redundancy of data is low in the DBMS system.
Data inconsistency is higher in the file system.	Data inconsistency is low in a database management system.
The file system offers lesser security.	Database Management System offers high security.
File System allows you to stores the data as isolated data files and entities.	Database Management System stores data as well as defined constraints and interrelation.
Not provide support for complicated transactions.	Easy to implement complicated transactions.
The centralization process is hard in File Management System.	Centralization is easy to achieve in the DBMS system.

It doesn't offer backup and recovery of data if it is lost.	DBMS system provides backup and recovery of data even if it is lost.
There is no efficient query processing in the file system.	You can easily query data in a database using the SQL language.
These system doesn't offer concurrency.	DBMS system provides a concurrency facility.

Database Schema

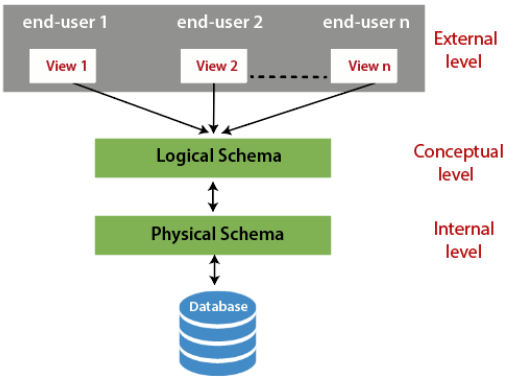
- A database schema is a logical representation of data that shows how the data in a database should be stored logically. It shows how the data is organized and the relationship between the tables.
- Database schema contains table, field, views and relation between different keys like primary key, foreign key.
- Data are stored in the form of files which is unstructured in nature which makes accessing the data difficult. Thus to resolve the issue the data are organized in structured way with the help of database schema.
- Database schema provides the organization of data and the relationship between the stored data.
- Database schema defines a set of guidelines that control the database along with that it provides information about the way of accessing and modifying the data.

There are three levels of the database schema are defined according to the three levels of data abstraction.

- Data Abstraction refers to the process of hiding irrelevant details from the user.

Levels of the schema

- Physical Schema
- Logical Schema
- View Schema



Physical Schema (or) internal level

- A Physical schema defines, how the data or information is stored physically in the storage systems in the form of files & indices.
- The Database administrator chooses where and how to store the data in the different blocks of storage.

- The physical level is used to describe complex low-level data structures in detail.
- The internal level is generally is concerned with the following activities:
    - Storage space allocations.
      For Example: B-Trees, Hashing etc.
    - Access paths.
      For Example: Specification of primary and secondary keys, indexes, pointers and sequencing.
    - Data compression and encryption techniques.
    - Optimization of internal structures.
    - Representation of stored fields.

Logical Schema (or) Conceptual Level

- The conceptual schema describes the design of a database at the conceptual level.
- The conceptual schema describes the structure of the whole database.
- The conceptual level describes what data are to be stored in the database and also describes what relationship exists among those data.
- In the conceptual level, internal details such as an implementation of the data structure are hidden.
- Programmers and database administrators work at this level.

View Schema (or) External Level

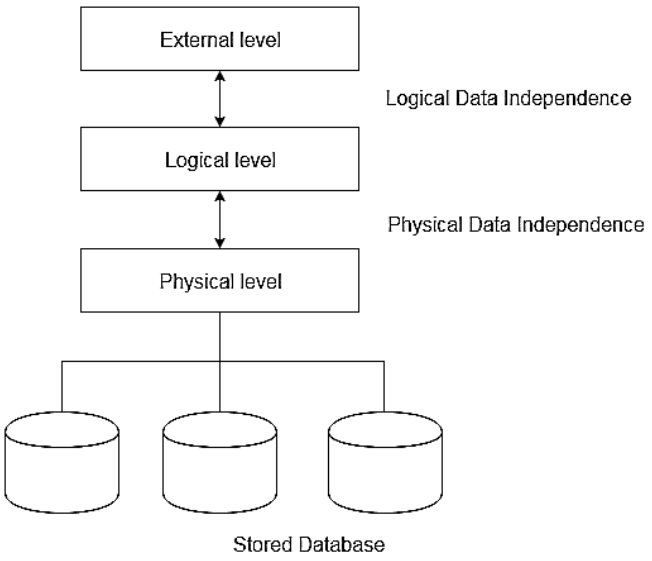
- At the external level, a database contains several schemas that sometimes called as subschema. The subschema is used to describe the different view of the database.
- Each view schema describes the database part that a particular user group is interested and hides the remaining database from that user group.
- The view schema describes the end user interaction with database systems.

Data Independence

- Data Independence is defined as a property of DBMS that helps you to change the Database schema at one level of a database system without requiring to change the schema at the next higher level. Data independence helps you to keep data separated from all programs that make use of it.
- User can use this stored data for computing and presentation. In many systems, data independence is an essential function for components of the system.

Types of Data Independence
In DBMS there are two types of data independence,

- Physical data independence
- Logical data independence



Physical Data Independence

- Physical data independence helps you to separate conceptual levels from the internal/physical levels. It allows you to provide a logical description of the database without the need to specify physical structures. Compared to Logical Independence, it is easy to achieve physical data independence.
- With Physical independence, you can easily change the physical storage structures or devices with an effect on the conceptual schema. Any change done would be absorbed by the mapping between the conceptual and internal levels. Physical data independence is achieved by the presence of the internal level of the database and then the transformation from the conceptual level of the database to the internal level.

Examples of changes under Physical Data Independence

Due to Physical independence, any of the below change will not affect the conceptual layer.

- Using a new storage device like Hard Drive or Magnetic Tapes
- Modifying the file organization technique in the Database
- Switching to different data structures.
- Changing the access method & Modifying indexes.
- Changes to compression techniques or hashing algorithms.
- Change of Location of Database from say C drive to D Drive

Logical Data Independence

- Logical Data Independence is the ability to change the conceptual scheme without changing External views

- Any change made will be absorbed by the mapping between external and conceptual levels.

- When compared to Physical Data independence, it is challenging to achieve logical data independence.

Examples of changes under Logical Data Independence

Due to Logical independence, any of the below change will not affect the external layer.

- Add/Modify/Delete a new attribute, entity or relationship is possible without a rewrite of existing application programs
- Merging two records into one
- Breaking an existing record into two or more records

Database Architecture

- The architecture of a database system is greatly influenced by the underlying computer system on which the database system runs. Database systems can be centralized, or client-server, where one server machine executes work on behalf of multiple client machines.
- The DBMS accepts SQL commands generated from a variety of user interfaces, produces query evaluation plans, executes these plans against the database, and returns the answers. (This is a simplification: SQL commands can be embedded in host-language application programs, e.g., Java or COBOL programs. We ignore these issues to concentrate on the core DBMS functionality.)
- When a user issues a query, the parsed query is presented to a query optimizer, which uses information about how the data is stored to produce an efficient execution plan for evaluating the query.
- An execution plan is a blueprint for evaluating a query, usually represented as a tree of relational operators (with annotations that contain additional detailed information about which access methods to use, etc.). Relational operators serve as the building blocks for evaluating queries posed against the data.
- The code that implements relational operators sits on top of the file and access methods layer. This layer supports the concept of a file, which, in a DBMS, is a collection of pages or a collection of records.
- Heap files, or files of unordered pages, as well as indexes are supported. In addition to keeping track of the pages in a file, this layer organizes the information within a page.
- The files and access methods layer code sits on top of the buffer manager, which brings pages in from disk to main memory as needed in response to read requests.
- The lowest layer of the DBMS software deals with management of space on disk, where the data is stored. Higher layers allocate, deallocate, read, and write pages through (routines provided by) this layer, called the disk space manager.

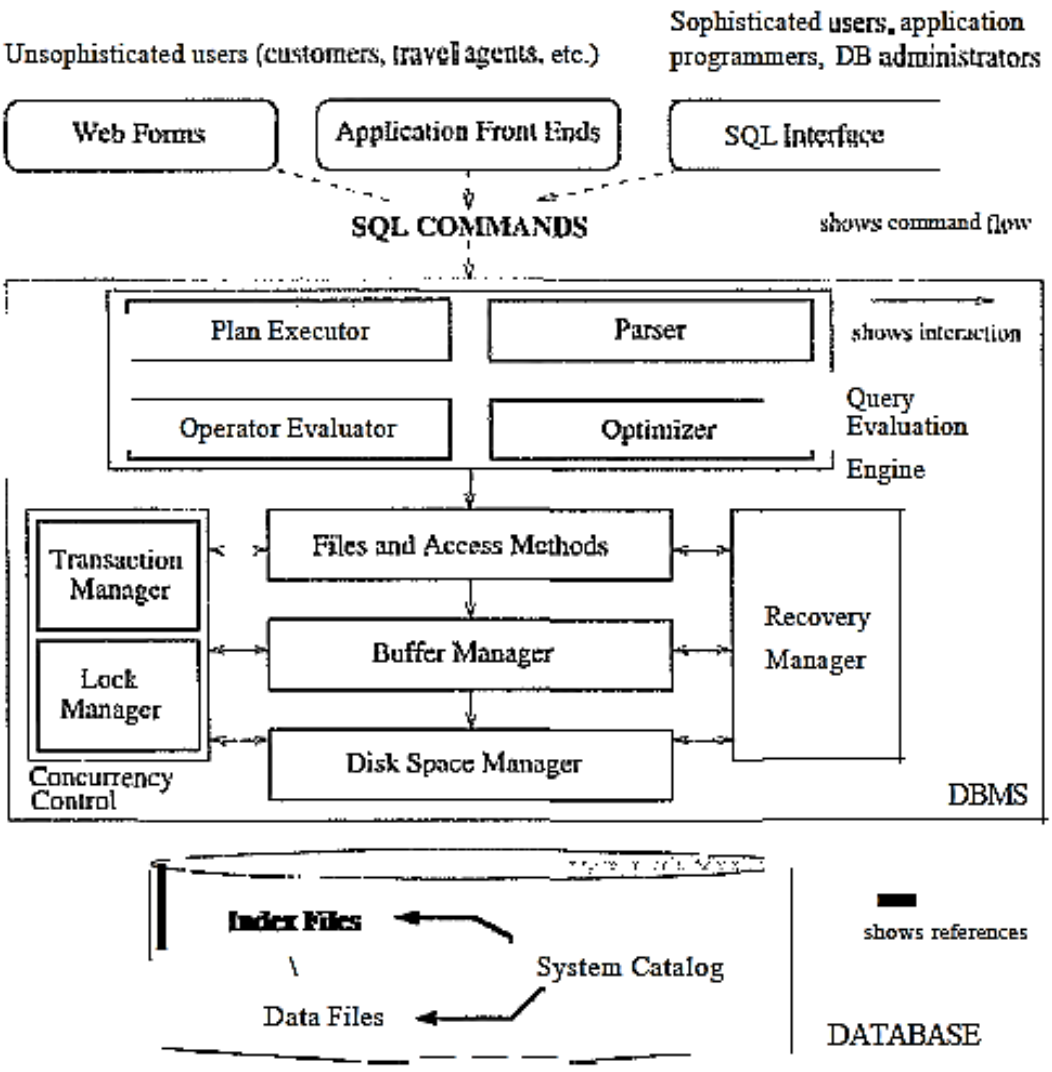


Fig: System Architecture

- The DBMS supports concurrency and crash recovery by carefully scheduling user requests and maintaining a log of all changes to the database.
- DBMS components associated with concurrency control and recovery include the transaction manager, which ensures that transactions request and release locks according to a suitable locking protocol and schedules the execution transactions;
- The lock manager, which keeps track of requests for locks and grants locks on database objects when they become available; and the recovery manager, which is responsible for maintaining a log and restoring the system to a consistent state after a crash.
- The disk space manager, buffer manager, and file and access method layers must interact with these components.

Database Models

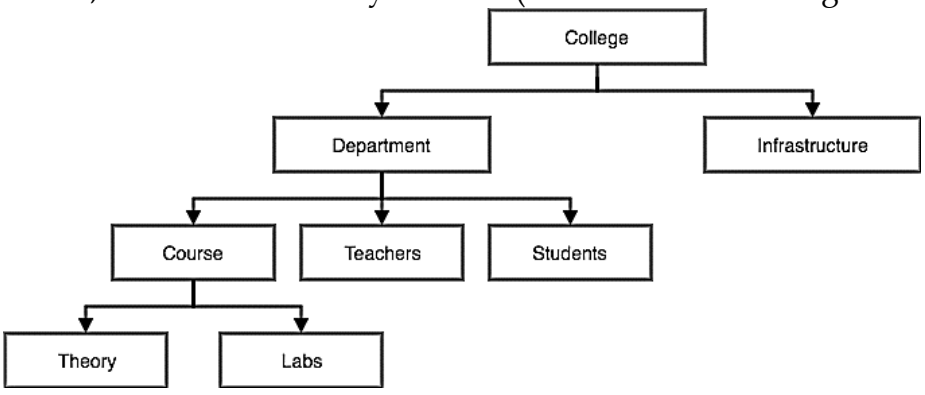
Data models define how the logical structure of a database is modeled. Data Models are fundamental entities to introduce abstraction in a DBMS. Data models define how data is connected to each other and how they are processed and stored inside the system.
The very first data model could be flat data-models, where all the data used are to be kept in the same plane. Earlier data models were not so scientific, hence they were prone to introduce lots of duplication and update anomalies.

Type of Database models

1. Hierarchical Model
2. Network Model
3. Entity-relationship Model
4. Relational Model
5. Object-oriented Model
6. NoSQL Model
7. Graph Model

1. Hierarchical Model

- The hierarchical database model organizes data into a tree-like structure, with a single root, to which all the other data is linked.
- The hierarchy starts from the Root data, and expands like a tree, adding child nodes to the parent nodes.
- In this model, a child node will only have a single parent node.
- This model efficiently describes many real-world relationships like the index of a book, etc.
- IBM's Information Management System (IMS) is based on this model.
- Data is organized into a tree-like structure with a one-to-many relationship between two different types of data, for example, one department can have many courses, many teachers, and of course many students(like shown in the diagram below).



<u>Advantages/Disadvantages of the Hierarchical Model</u>

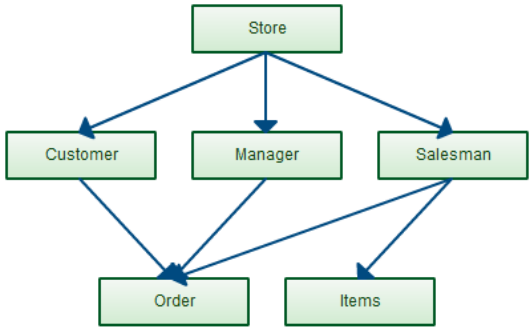
- Because it has one-to-many relationships between different types of data so it is easier and fast to fetch the data.

- Hierarchical model is less flexible.
- It doesn't support many-to-many relationships.

2. Network Model

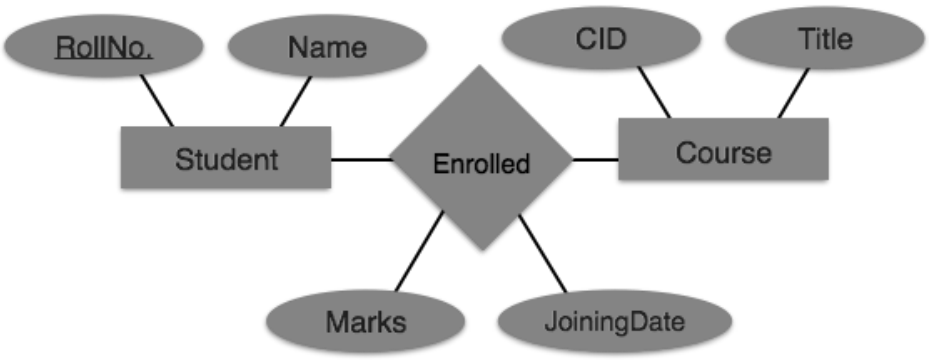
- The Network Model is an extension of the Hierarchical model.
- In this model, data is organized more like a graph, and allowed to have more than one parent node.
- In the network database model, data is more related as more relationships are established in this database model.
- As the data is more related, hence accessing the data is also easier and fast.
- This database model uses many-to-many data relationships.
- Integrated Data Store (IDS) is based on this database model.

- The implementation of the Network model is complex, and it's very difficult to maintain it.
- The Network model is difficult to modify also.



3. Entity-relationship Model

- In this database model, relationships are created by dividing objects of interest into entities and their characteristics into attributes.
- Different entities are related using relationships.
- ER Models are defined to represent the relationships in pictorial form to make it easier for different stakeholders to understand.
- This model is good to design a database, which can then be turned into tables in a relational model.
- Let's take an example, if we have to design a School Database, then the Student will be an entity with attributes name, age, address, etc. As an Address is generally complex, it can be another entity with attributes street, pincode, city, etc, and there will be a relationship between them.

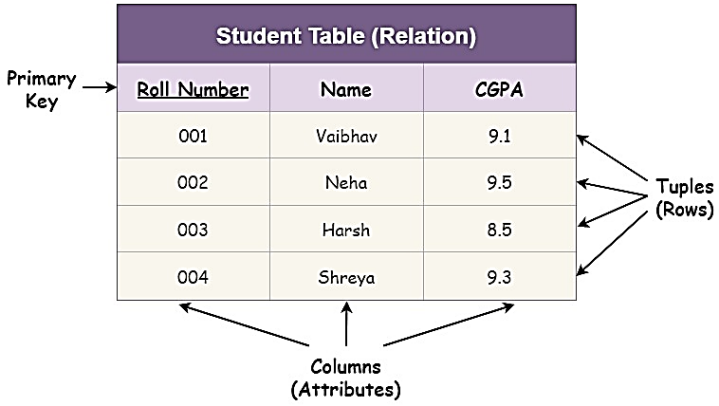


Advantages of the ER Model

- It is easy to understand and design.
- Using the ER model we can represent data structures easily.
- As the ER model cannot be directly implemented into a database model, it is just a step toward designing the relational database model.

4. Relational Model

- In this model, data is organized in two-dimensional tables and the relationship is maintained by storing a common field.
- This model was introduced by E.F Codd in 1970, and since then it has been the most widely used database model, infact, we can say the only database model used around the world.
- The basic structure of data in the relational model is tables. All the information related to a particular type is stored in rows of that table.
- Hence, tables are also known as relations in relational model.



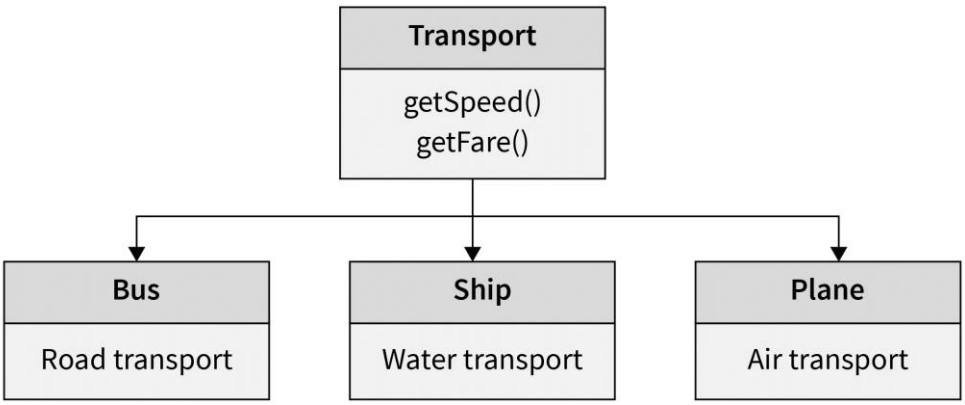
Advantages of the Relational Model

- It's simple and easy to implement.
- Poplar database software is available for this database model.
- It supports SQL using which you can easily query the data.

5. Object-oriented Model

- In this model, data is stored in the form of objects.

- The behavior of the object-oriented database model is just like object-oriented programming.
- A very popular example of an Object Database management system **or ODBMS is MongoDB which is also a NoSQL database.**
- This database model is not mature enough as compared to the relational database model.



Advantages of the Object-oriented Model

- It can easily support complex data structures, with relationships.
- It also supports features like Inheritance, Encapsulation, etc.

6. NoSQL Model

- The NoSQL database model supports an **unstructured** style of storing data.
- Data is stored as **documents**.
- The documents look more like JSON strings or **Key-value** based object representations.
- It provides a **flexible schema**.
- It does provide features like **indexing**, relationships between data, etc.
- The support for data querying is limited in the NoSQL database model.
- This database model is well-suited for Big data applications, real-time analytics, CMS (Content Management systems), etc.

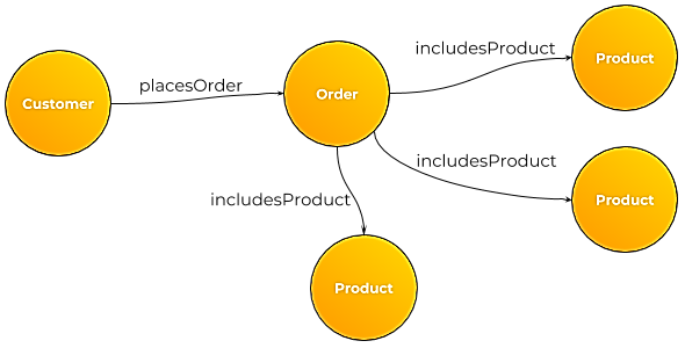
Advantages of the NoSQL Model

- NoSQL database model is scalable.
- This database model functions with high performance.
- The NoSQL database model can handle large volumes of data.

7. Graph Model

- The Graph database model is based on more real-world like relationships.
- Data is represented using Nodes or entities.
- The nodes are related using edges.
- The popular database Neo4j is based on the Graph database model.

- If your application has simple data requirements, then you should not use the graph database model.
- In modern applications like social networks, recommendation systems, etc. the graph database model is well-suited.



Advantages of the Graph Model

- It handles complex relationships very well.
- In the modern world where there is so much data and the data has to be related in different ways, the graph database model is very useful.

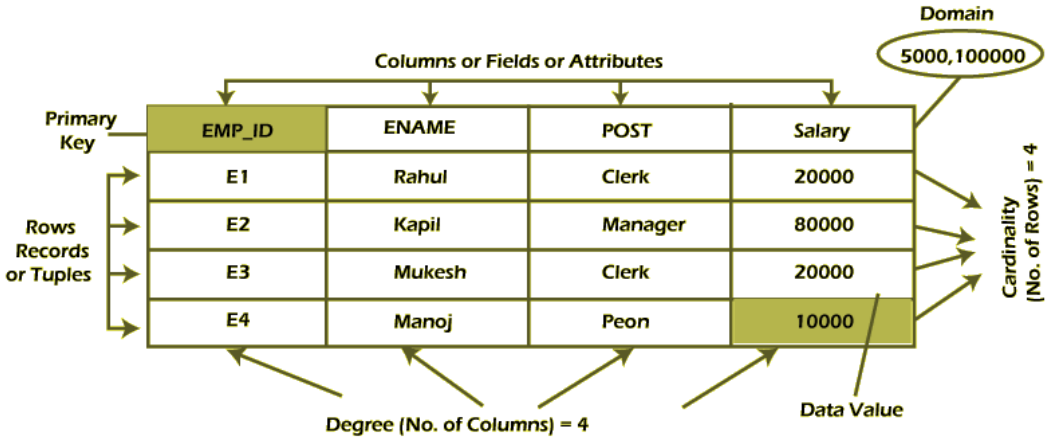
RDBMS

- RDBMS stands for Relational Database Management System.
- The relational model in DBMS is an abstract model used to organize and manage the data stored in a database. It stores data in two-dimensional inter-related tables, also known as relations in which each row represents an entity and each column represents the properties of the entity.
- All modern database management systems like SQL, MS SQL Server, IBM DB2, ORACLE, My-SQL, and Microsoft Access are based on RDBMS.
- It is called Relational Database Management System (RDBMS) because it is based on the relational model introduced by E.F. Codd.

➢ RDBMS Terminology

- **Attribute:** Each column in a Table. Attributes are the properties which define a relation. e.g., Student_Rollno, NAME,etc.
- **Tables:** In the Relational model the, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes.
- **Tuple:** It is nothing but a single row of a table, which contains a single record.
- **Relation Schema**: A relation schema represents the name of the relation with its attributes.
- **Degree:** The total number of attributes which in the relation is called the degree of the relation.
- **Cardinality:** Total number of rows present in the Table.
- **Column:** The column represents the set of values for a specific attribute.

- **Relation instance**: Relation instance is a finite set of tuples in the RDBMS system. Relation instances never have duplicate tuples.
- **Relation key:** Every row has one, two or multiple attributes, which is called relation key.
- **Attribute domain:** Every attribute has some pre-defined value and scope which is known as attribute domain.



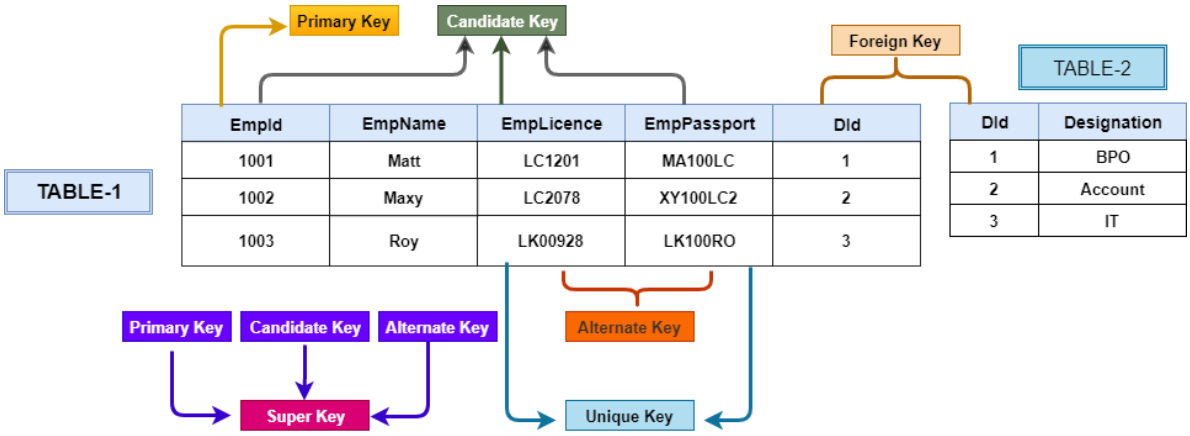
➢ RDBMS Keys

KEYS in DBMS is an attribute or set of attributes which helps you to identify a row (tuple) in a relation (table). They allow you to find the relation between two tables. Keys help you uniquely identify a row in a table by a combination of one or more columns in that table. Key is also helpful for finding unique record or row from the table. Database key is also helpful for finding unique record or row from the table.

Types of Keys in Database Management System

There are mainly seven different types of Keys in DBMS and each key has it's different functionality:

- **Super Key:** A super key is a group of single or multiple keys which identifies rows in a table.
- **Primary Key:** A column or group of columns in a table that uniquely identify every row in that table.



- **Candidate Key:** A set of attributes that uniquely identify tuples in a table. Candidate Key is a super key with no repeated attributes.
- **Alternate Key:** A column or group of columns in a table that uniquely identify every row in that table.
- **Foreign Key:** A column that creates a relationship between two tables. The purpose of foreign keys is to maintain data integrity and allow navigation between two different instances of an entity.
- **Compound Key:** Two or more attributes that allow you to uniquely recognize a specific record. It is possible that each column may not be unique by itself within the database.

➢ Integrity Constraints

Integrity constraints are rules that help to maintain the accuracy and consistency of data in a database.
Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.
Integrity constraints can also be used to enforce relationships between tables.

Types of Integrity Constraints

- Key constraint
- Domain constraint
- Entity Integrity constraint
- Referential Integrity constraint

1. Key Constraint

- Keys are the entity set that is used to identify an entity within its entity set uniquely.
- An entity set can have multiple keys, but out of which one key will be the primary key. A primary key can contain a unique and null value in the relational table.

ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1002	Morgan	8th	22

Not allowed. Because all row must be unique

2. Domain Constraint

- Domain constraint defines the domain or set of values for an attribute.
- It specifies that the value taken by the attribute must be the atomic value from its domain.
- The data type of domain includes string, character, integer, time, date, currency, etc. The value of the attribute must be available in the corresponding domain

ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1004	Morgan	8th	A

Not allowed. Because AGE is an integer attribute

3. Entity integrity constraints

- The entity integrity constraint states that primary key value can't be null.
- This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.
- A table can contain a null value other than the primary key field.

EMPLOYEE

EMP_ID	EMP_NAME	SALARY
123	Jack	30000
142	Harry	60000
164	John	20000
	Jackson	27000

Not allowed as primary key can't contain a NULL value

4. Referential Integrity Constraints

- A referential integrity constraint is specified between two tables.
- In the Referential integrity constraints, if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be null or be available in Table 2.

Example:

(Table 1)

EMP_NAME	NAME	AGE	D_No
1	Jack	20	11
2	Harry	40	24
3	John	27	18
4	Devil	38	13

Foreign key

Not allowed as D_No 18 is not defined as a Primary key of table 2 and In table 1, D_No is a foreign key defined

Relationships

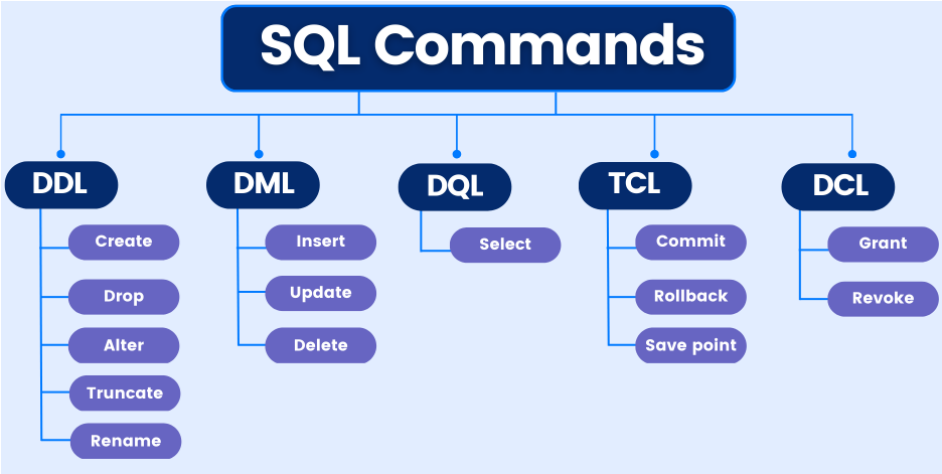
(Table 2)

Primary Key

D_No	D_Location
11	Mumbai
24	Delhi
13	Noida

SQL

- SQL stands for Structured Query Language. It is used for storing and managing data in relational database management system (RDMS).
- It is a standard language for Relational Database System. It enables a user to create, read, update and delete relational databases and tables.
- All the RDBMS like MySQL, Informix, Oracle, MS Access and SQL Server use SQL as their standard database language.



DDL

- Data Definition Language (DDL) or Schema Definition Language, statements are used to define the database structure or schema.
- These statements handle the design and storage of database objects.

➢ **CREATE**

✓ ***CREATE DATABASE:***

The CREATE DATABASE statement is used to create a new SQL database.

Syntax

```
CREATE DATABASE databasename;
```

Example

The following SQL statement creates a database called "testDB":

```
CREATE DATABASE testDB;
```

✓ ***CREATE TABLE:***

The CREATE TABLE statement is used to create a new table in a database.

Syntax

```
CREATE TABLE table_name (column1 datatype, column2 datatype,
column3 datatype,  .... );
```

- The column parameters specify the names of the columns of the table.

- The datatype parameter specifies the type of data the column can hold (e.g. varchar, integer, date, etc.).

Example

The following example creates a table called "Persons" that contains five columns: PersonID, LastName, FirstName, Address, and City:

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

- **DESC Statement (Describe Table):**

SQL DESC statement use for describe the list of column definitions for specified table. You can use either DESC or DESCRIBE statement.

Syntax:

```
Describe table  name;
```

Example:

```
Describe Persons;
```

```
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| PersonID | int         | YES  |     | NULL    |       |
| LastName | varchar(255)| YES  |     | NULL    |       |
| Firstname| varchar(255)| YES  |     | NULL    |       |
| Address  | varchar(255)| YES  |     | NULL    |       |
| City     | varchar(25) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
```

Field indicates the column name, Type is the data type for the column, NULL indicates whether the column can contain NULL values, Key indicates whether the column is indexed, and Default specifies the column's default value. Extra displays special information about columns:

- **ALTER TABLE:**
  - The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.
  - The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

✓ **ALTER TABLE - ADD Column**

To add a column in a table.

Syntax:

```
ALTER TABLE table_name ADD column_name datatype;
```

Example:

The following SQL adds an "Email" column to the "Persons" table:

```
ALTER TABLE Persons ADD Email varchar(255);datatype;
```

Add Multiple Column in table:

```
ALTER TABLE table_name ADD ( column_name1
datatype[(size)],column_name2 datatype[(size)], ... );
```

- ✓ **ALTER TABLE - DROP COLUMN**

  It is used to delete a column in a table

  Syntax:

```
ALTER TABLE table_name DROP COLUMN column_name;
```

  Example:

  The following SQL deletes the "Email" column from the "Persons" table:

```
ALTER TABLE Persons DROP COLUMN Email;
```

- ✓ **MODIFY EXISTING COLUMN IN TABLE**

  To change the data type of a column in a table

  Syntax:

```
ALTER TABLE table_name MODIFY COLUMN column_name datatype;
```

- ✓ **RENAME THE EXISTING COLUMN IN A TABLE**

  Syntax:

```
ALTER table table_name change old_column_name new_column_name datatype;
```

  Example:

```
ALTER table persons change city district varchar(50);
```

- ➢ **RENAME:**

  RENAME command is used to set a new name for any existing table.

  Syntax (Rename Table):

```
RENAME TABLE old_table_name to new_table_name;
```

  Example:

```
RENAME TABLE persons to students_info;
```

  The above query will rename the table student to students_info.

- ➢ **TRUNCATE:**

  It is used to delete all the rows from the table and free the space containing the table.

  Syntax:

```
TRUNCATE TABLE table_name;
```

  Example:

```
TRUNCATE TABLE Persons;
```

- ➢ **DROP TABLE / DATABASE**

  The DROP TABLE / DATABASE statement is used to drop an existing table in a database or a database.

Syntax

```
DROP TABLE table_name;
```

Example

The following SQL statement drops the existing table "Person":

```
DROP TABLE Person;
```

DML

- The SQL commands that deals with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements.
- DML is responsible for all forms data modification in a database.

➢ **INSERT:**

The insert statement is used to add new row to a table.

Syntax

```
INSERT INTO TABLE_NAME (col1, col2, col3,.... col N)
VALUES (value1, value2, value3, .... valueN);
```

Or

```
INSERT INTO TABLE_NAME VALUES (value1, value2, value3, .... valueN);
```

Example

Insert the values into the following table (person).

```
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| ID       | int         | NO   |     | NULL    |       |
| NAME     | varchar(20) | YES  |     | NULL    |       |
| AGE      | int         | YES  |     | NULL    |       |
| ADDRESS  | varchar(20) | YES  |     | NULL    |       |
| SALARY   | int         | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
```

```
INSERT into person values (1,"ram",32,"Chennai",2000),(2,"sam",28,"Hyd",2500),
(3,"Henry",30,"Bangalore",5000),(4,"Adam",29,"Pune",6500),(5,"Alice",25,"Delhi",8500);
```

➢ **UPDATE**

UPDATE Query is used to modify the existing records in a table. The WHERE clause with the UPDATE query to update the selected rows, otherwise all the rows would be affected.

Syntax

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

Example

Update the ADDRESS for a customer whose ID number is 5 in the table.

```
UPDATE person SET ADDRESS = "LONDON" WHERE ID = 5;
```

Now, the PERSON table would have the following records,

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | ram      |  32 | Chennai   |  2000    |
|  2 | sam      |  28 | Hyd       |  2500    |
|  3 | Henry    |  30 | Bangalore |  5000    |
|  4 | Adam     |  29 | Pune      |  6500    |
|  5 | Alice    |  25 | LONDON    |  8500    |
+----+----------+-----+-----------+----------+
```

To modify all the ADDRESS and the SALARY column values in the CUSTOMERS table, not need to use the WHERE clause as the UPDATE query would be enough as shown in the following code block.

```
UPDATE person SET ADDRESS = "LONDON" , SALARY = 5000;
```

Now, the PERSON table would have the following records,

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | ram      |  32 | LONDON    |  5000    |
|  2 | sam      |  28 | LONDON    |  5000    |
|  3 | Henry    |  30 | LONDON    |  5000    |
|  4 | Adam     |  29 | LONDON    |  5000    |
|  5 | Alice    |  25 | LONDON    |  5000    |
+----+----------+-----+-----------+----------+
```

- **DELETE**

DELETE Query is used to delete the existing records from a table.
You can use the WHERE clause with a DELETE query to delete the selected rows, otherwise all the records would be deleted.

Syntax

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

Example

The following code has a query, which will DELETE a customer, whose ID is 5.

```
DELETE FROM person
WHERE ID = 5;
```

Now, the person table would have the following records.

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | ram      |  32 | LONDON    |  5000    |
|  2 | sam      |  28 | LONDON    |  5000    |
|  3 | Henry    |  30 | LONDON    |  5000    |
|  4 | Adam     |  29 | LONDON    |  5000    |
+----+----------+-----+-----------+----------+
```

To DELETE all the records from the CUSTOMERS table, no need to use the WHERE clause and the DELETE query would be as follows –

```
DELETE FROM person;
```

DQL

SELECT:

SELECT statement is used to fetch the data from a database table which returns this data in the form of a result table. These result tables are called result-sets.

Syntax

```
SELECT column1, column2, columnN FROM table_name;
```

Example

To fetch all the fields of the table,

```
SELECT * FROM person
```

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | ram      |  32 | Chennai   |  2000    |
|  2 | sam      |  28 | Hyd       |  2500    |
|  3 | Henry    |  30 | Bangalore |  5000    |
|  4 | Adam     |  29 | Pune      |  6500    |
|  5 | Alice    |  25 | Delhi     |  8500    |
+----+----------+-----+-----------+----------+
```

To fetch selected fields (ID, NAME) of the table,

```
SELECT ID, NAME FROM person
```

```
+----+----------+
| ID | NAME     |
+----+----------+
|  1 | ram      |
|  2 | sam      |
|  3 | Henry    |
|  4 | Adam     |
|  5 | Alice    |
+----+----------+
```

TCL

- TCL stands for Transaction control language.
- TCL commands help the user manage the transactions that take place in a database.
- COMMIT. ROLLBACK and SAVEPOINT are the most commonly used TCL commands in SQL.

✓ **COMMIT**

COMMIT command in SQL is used to save all the transaction-related changes permanently to the disk.

Syntax:
```
COMMIT
```

✓ **Auto-commit**

- Auto-commit mode means that when a statement is completed, the method commit is called on that statement automatically.
- Turn off/on the auto-commit using the **SET autocommit** statement.
- To enable the auto-commit:
```
SET autocommit = 1;
```
- To turn-off the auto-commit:
```
SET autocommit = 0;
```

✓ **SAVEPOINT**

- Savepoint helps to save the transaction temporarily.
- The SAVEPOINT statement is used to set a save point for the transaction with the specified name.
- The changes done till savpoint will be unchanged and all the transactions after savepoint will be rolled back.
- **Syntax:**
```
SAVEPOINT savepoint_name;
```

✓ **RELEASE SAVEPOINT**

- The RELEASE SAVEPOINT statement deletes the specified savepoint.
- **Syntax:**
```
RELEASE SAVEPOINT savepoint_name;
```

✓ **ROLLBACK**

- Rollback is used to restore the database to that state which was last committed.
- **Syntax:**
```
ROLLBACK;
```

✓ **Rollback the transaction to the savepoint.**

- **Syntax:**
```
ROLLBACK TO SAVEPOINT mysavepoint;
```

DCL

- Data control language (DCL) is used to access the stored data. It helps in controlling access to information stored in a database.
- It provides the administrators, to remove and set database permissions to desired users as needed.
- These commands are employed to grant, remove and deny permissions to users for retrieving and manipulating a database.
- The different DCL commands are GRANT & REVOKE.

✓ **GRANT**

- It is employed to grant a privilege to a user.
- GRANT command allows specified users to perform specified tasks.
- The privilege names are ***SELECT, UPDATE, DELETE, INSERT, ALTER, ALL.***

- Syntax:

```
GRANT privilege_list
      ON Object_name
      TO user  name;
```

✓ **REVOKE**

- It is employed to remove a privilege from a user.
- REVOKE helps the owner to cancel previously granted permissions.

- Syntax:

```
REVOKE privilege_list
       ON Object_name
       TO user  name;
```

Joins

- SQL Join statement is used to combine data or rows from two or more tables based on a common field between them.
- When the related data is stored across multiple tables, joins help you to retrieve records combining the fields from these tables using their foreign keys.

Types of joins in SQL

SQL provides various types of Joins that are categorized based on the way data across multiple tables are joined together. They are listed as follows −

✓ **Inner Join**

INNER JOIN is the default join which retrieves the intersection of two tables. It compares each row of the first table with each row of the second table. If the pairs of these rows satisfy the join-predicate, they are joined together.

```
SELECT columns_from_both_tables
FROM table1
INNER JOIN table2
ON table1.column1 = table2.column2
```

✓ **Natural Join**

Natural join is an SQL join operation that creates a join on the base of the common columns in the tables. To perform natural join there must be one common attribute (Column) between two tables. Natural join will retrieve from multiple relations.

```
SELECT [column_names | *]
FROM table_name1
NATURAL JOIN table_name2;
```

✓ **Self-Join**

Self-Join is used to join a table to itself. This means that each row in a table is joined to itself and every other row in that table. However, referencing the same table more than once within a single query will result in an error. To avoid this, SQL SELF JOIN aliases are used.

```
SELECT column_name(s)
FROM table1 T1, table1 T2
WHERE condition;
```

✓ **Cross Join**

An SQL Cross Join is a basic type of inner join that is used to retrieve the Cartesian product (or cross product) of two individual tables. That means, this join will combine each row of the first table with each row of second table (i.e. permutations).

```
SELECT column_name(s)
FROM table1
CROSS JOIN table2;
```

✓ **Outer Join**

SQL Outer joins give both matched and unmatched rows of data depending on the type of outer joins. These types are outer joins are sub-divided into the following types:

✓ **Left Outer Join:**

The SQL LEFT JOIN joins two tables based on a common column. It selects records that have matching values in these columns and the remaining rows from the left table.

```
SELECT column-name(s)
FROM table1 LEFT OUTER JOIN table2
ON table1.column-name = table2.column-name;
```

✓ **Right Outer Join**

The SQL RIGHT JOIN statement joins two tables based on a common column. It selects records that have matching values in these columns and the remaining rows from the right table.

```
SELECT columns_from_both_tables
FROM table1
RIGHT JOIN table2
ON table1.column1 = table2.column2
```

✓ **Full Outer Join**

The SQL FULL OUTER JOIN statement joins two tables based on a common column. It selects records that have matching values in these columns and the remaining rows from both of the tables.

```
SELECT columns
FROM table1
FULL OUTER JOIN table2
ON table1.column1 = table2.column2;
```

Triggers

- A trigger is a stored procedure in a database that automatically invokes whenever a special event in the database occurs.
- A database that has a set of associated triggers is called an active database. A trigger description contains three parts:
  - ***Event:*** A change to the database that activates the trigger.
  - ***Condition:*** A query or test that is run when the trigger is activated.
  - ***Action:*** A procedure that is executed when the trigger is activated and its condition is true.
- Triggers are written to be executed in response to any of the following events.
  - ✓ A database manipulation (DML) statement (DELETE, INSERT, or UPDATE).
  - ✓ A database definition (DDL) statement (CREATE, ALTER, or DROP).
  - ✓ A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).
- Triggers could be defined on the table, view, schema, or database with which the event is associated.

✓ **CREATE TRIGGER**

Create a trigger in SQL Server by using the CREATE TRIGGER statement as follows:

```
CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF }
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
DECLARE
   Declaration-statements
BEGIN
   Executable-statements
EXCEPTION
   Exception-handling-statements
END;
```

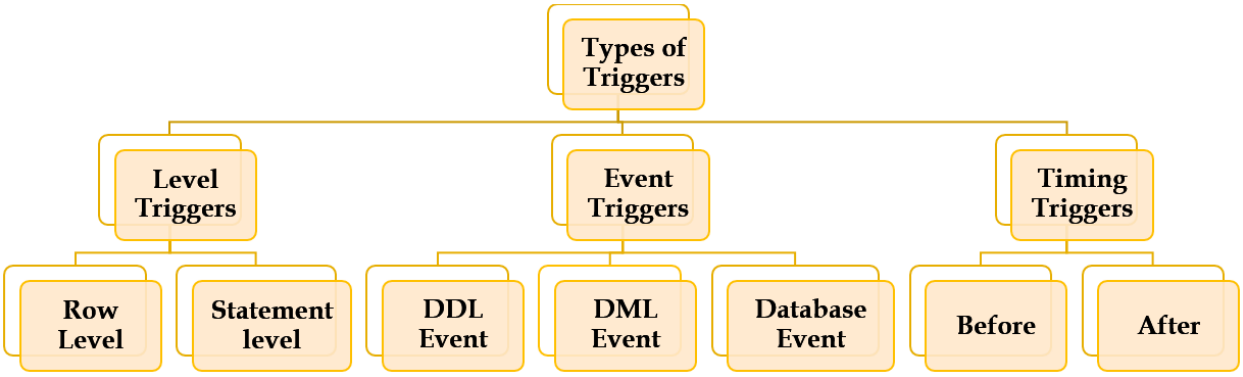
Where,

- CREATE [OR REPLACE] TRIGGER trigger_name − Creates or replaces an existing trigger with the *trigger_name*.
- {BEFORE | AFTER | INSTEAD OF} − This specifies when the trigger will be executed. The INSTEAD OF clause is used for creating trigger on a view.
- {INSERT [OR] | UPDATE [OR] | DELETE} − This specifies the DML operation.
- [OF col_name] − This specifies the column name that will be updated.
- [ON table_name] − This specifies the name of the table associated with the trigger.
- [REFERENCING OLD AS o NEW AS n] − This allows you to refer new and old values for various DML statements, such as INSERT, UPDATE, and DELETE.
- [FOR EACH ROW] − This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.
- WHEN (condition) − This provides a condition for rows for which the trigger would fire. This clause is valid only for row-level triggers.

➢ **Types of Triggers**

There are three types of Triggers:

- Level Triggers
- Event Triggers
- Timing Triggers



✓ **Level Triggers**

Level triggers are divided into two parts:

✓ **Row Level Triggers**

- It fires for each record that was affected by the execution of DML statements such as INSERT, UPDATE, DELETE, and so on.
- A FOR EACH ROW clause is always used in a triggering statement.

✓ **Statement Level Triggers**

- It fires once for each executed statement.

✓ **Event Triggers**

Event triggers are divided into three parts:

- ✓ **DDL Event Trigger**
  - It is triggered by the execution of every DDL statement (CREATE, ALTER, DROP, TRUNCATE).
- ✓ **DML Event Trigger**
  - It is triggered by the execution of every DML statement (INSERT, UPDATE, DELETE).
- ✓ **Database Event Trigger**
  - It is triggered by the execution of any database operation, such as LOGON, LOGOFF, SHUTDOWN, SERVERERROR, and so on.

✓ **Timing Triggers**

Timing triggers are divided into two parts:

- ✓ **Before Trigger**
  - It fires before the DML statement is executed.
- ✓ **After Trigger**
  - It fires after the DML statement has been executed.

➢ **Benefits of Triggers**

Triggers can be written for the following purposes −

- Generating some derived column values automatically
- Enforcing referential integrity
- Event logging and storing information on table access
- Auditing
- Synchronous replication of tables
- Imposing security authorizations
- Preventing invalid transactions

Index

- SQL Indexes are special lookup tables that are used to speed up the process of data retrieval.
- They hold pointers that refer to the data stored in a database, which makes it easier to locate the required data records in a database table.
- SQL Indexes work similar to the index of a book or a journal.
- SQL Indexes need their own storage space within the database. Despite that, the users cannot view them physically as they are just performance tools.

✓ CREATE INDEX

- An index in SQL can be created using the CREATE INDEX statement. This statement allows you to name the index, to specify the table and which column or columns to index, and to indicate whether the index is in an ascending or descending order.

- Preferably, an index must be created on column(s) of a large table that are frequently queried for data retrieval.

- Syntax
  The basic syntax of a CREATE INDEX is as follows −

```
CREATE INDEX index_name ON table_name;
```

✓ Rename an INDEX

```
ALTER INDEX old_Index_Name RENAME TO new_Index_Name;
```

✓ Remove an INDEX

```
DROP INDEX Index_Name;
```

✓ Types of Indexes
There are various types of indexes that can be created using the CREATE INDEX statement. They are:
- Unique Index
- Single-Column Index
- Composite Index
- Implicit Index

✓ Unique Indexes
Unique indexes are used not only for performance, but also for data integrity. A unique index does not allow any duplicate values to be inserted into the table. It is automatically created by PRIMARY and UNIQUE constraints when they are applied on a database table, in order to prevent the user from inserting duplicate values into the indexed table column(s). The basic syntax is as follows.

```
CREATE UNIQUE INDEX index_name on table_name (column_name);
```

✓ Single-Column Indexes
A single-column index is created only on one table column. The syntax is as follows.

```
CREATE INDEX index_name ON table_name (column_name);
```

✓ Composite Indexes
A composite index is an index that can be created on two or more columns of a table. Its basic syntax is as follows.

```
CREATE INDEX index_name on table_name (column1, column2);
```

✓ Implicit Indexes
Implicit indexes are indexes that are automatically created by the database server when an object is created. For example, indexes are automatically created when primary key and unique constraints are created on a table in MySQL database.

Views

- A view is a virtual or logical table that allows to view or manipulate parts of the tables.
- A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depends on the written SQL query to create a view.
- Views allow users to do the following −
  - Structure data in a way that users or classes of users find natural or intuitive.
  - Restrict access to the data in such a way that a user can see and (sometimes) modify exactly what they need and no more.
  - Summarize data from various tables which can be used to generate reports.

✓ Creating Views
- Database views are created using the CREATE VIEW statement. Views can be created from a single table, multiple tables or another view.
- The basic **CREATE VIEW** syntax is as follows −

```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE [condition];
```

Example:

```
CREATE VIEW studentview AS SELECT
NAME, ADDRESS FROM StudentDetails
WHERE S_ID < 5;
```

✓ Syntax to fetch the view table

```
select column_name from view_name;
```

✓ Syntax to delete the view

```
DROP view view_name;
```

✓ Syntax to update the view

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

Uses of a View: A good database should contain views due to the given reasons:

1. **Restricting data access –** Views provide an additional level of table security by restricting access to a predetermined set of rows and columns of a table.
2. **Hiding data complexity –** A view can hide the complexity that exists in multiple tables join.
3. **Simplify commands for the user –** Views allow the user to select information from multiple tables without requiring the users to actually know how to perform a join.
4. **Store complex queries –** Views can be used to store complex queries.
5. **Multiple view facility –** Different views can be created on the same table for different users.

Review Questions

1. Explain in detail about different levels of schema in DBMS.
2. Discuss in detail about different types of data models?
3. Define database. Give an overview of database architecture.
4. Explain various integrity constraint in SQL with example.
5. List and explain various DDL commands in SQL with examples.
6. List and explain various DML commands in SQL with examples.
7. What do you mean by JOINs? Explain different types of JOINs?
8. What is trigger? Explain how to implement triggers in SQL?
9. What is a view in SQL? Write a syntax for creating views with an example.
10. Write about SQL Indexes and explain various types of Indexes.