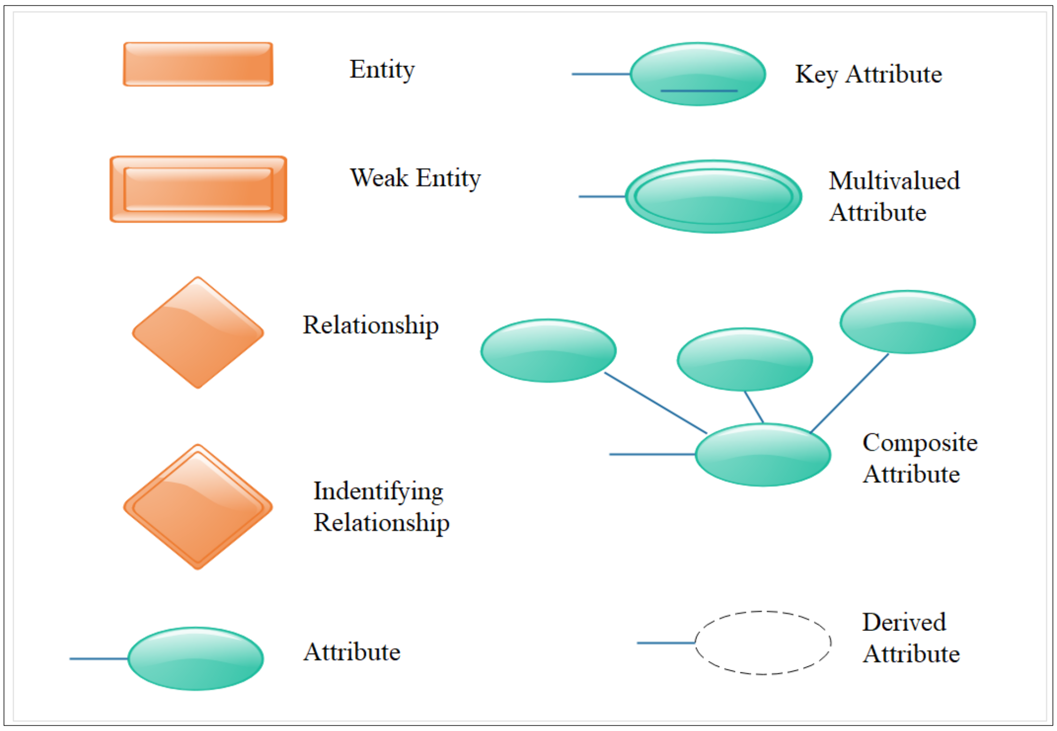
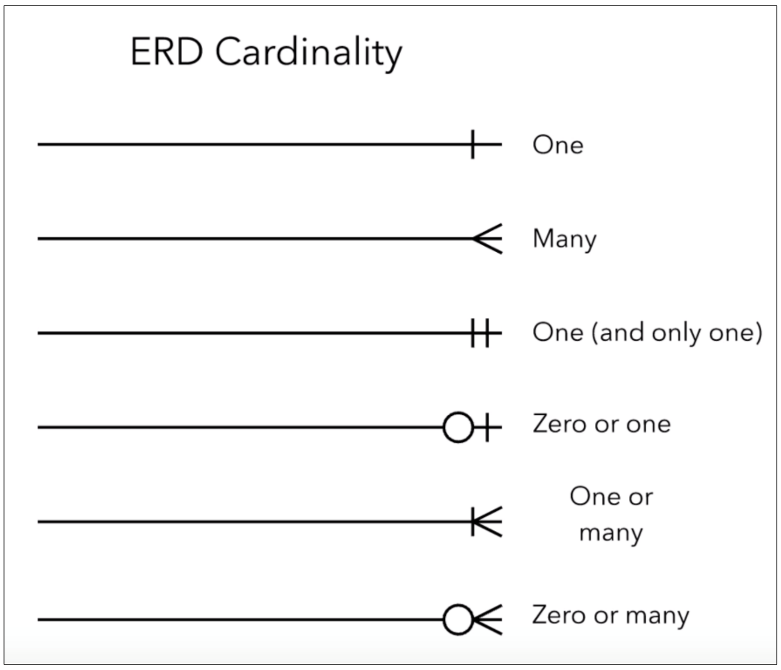
2. Concept of E-R, E-R Notation, EER, E-R to table rule, constraints.

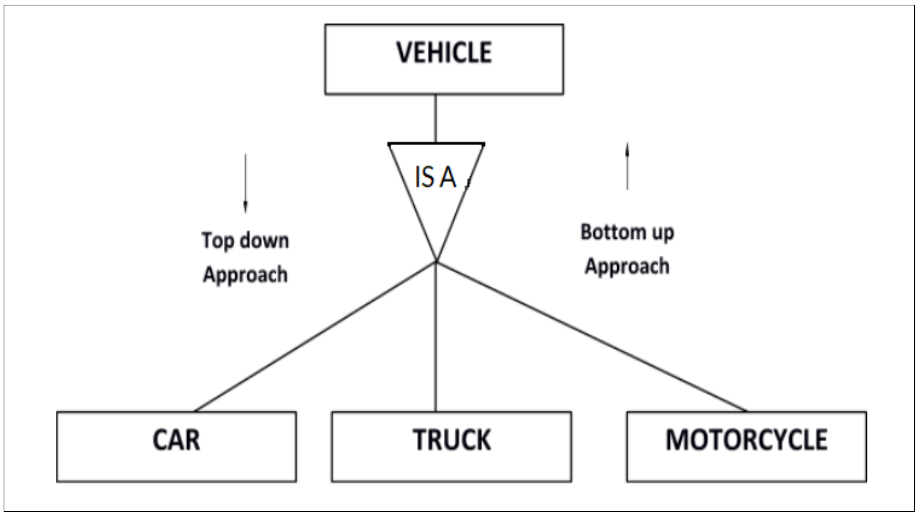
- ER model stands for an Entity-Relationship model. This model is used to define the data elements and relationship for a specified system. It develops a conceptual design for the database. In ER modelling, the database structure is portrayed as a diagram called an entity-relationship diagram.
- ER Notation:





- **More…** https://www.javatpoint.com/dbms-er-model-concept

- **EER (Extended Entity-Relationship):**
  EER is a high-level data model that incorporates the extensions to the original ER model. EERare high level models that represent the requirements and complexities of complex database.
  a. **Specialization**
     Specialization is a process of identifying subsets of an entity that share some different characteristic. It is a top-down approach in which one entity is broken down into low level entity.



b. **Generalization**

Generalization is a process of generalizing an entity which contains generalized attributes or properties of generalized entities. It is a Bottom-up process.

c. **Aggregation:**

Aggregation refers to the process by which entities are combined to form a single meaningful entity. The specific entities are combined because they do not make sense on their own. To establish a single entity, aggregation creates a relationship that combines these entities.

- **ER to table:**
- Since ER diagram gives us the good knowledge about the requirement and the mapping of the entities in it, we can easily convert them as tables and columns. i.e.; using ER diagrams one can easily create relational data model, which nothing but the logical view of the database.
- **The basic rule for converting the ER diagrams into tables is**
    - Convert all the Entities in the diagram to tables.
    - All single valued attributes of an entity are converted to a column of the table.
    - Key attribute in the ER diagram becomes the Primary key of the table.
    - Declare the foreign key column, if applicable.
    - Any multi-valued attributes are converted into new table.
    - Any composite attributes are merged into same table as different columns
    - One can ignore derived attribute, since it can be calculated at any time.
- Constraints:

▪ .There are five types of constraints:

NOT NULL	is a rule that prevents null values from being entered into one or more columns within a table.

unique	(Also referred to as a *unique key constraint*) is a rule that forbids duplicate values in one or more columns within a table. Unique and primary keys are the supported unique constraints.
primary key	is a column or combination of columns that has the same properties as a unique constraint. You can use a primary key and foreign key constraints to define relationships between tables
foreign key	(Also referred to as a *referential constraint* or a *referential integrity constraint*) is a logical rule about values in one or more columns in one or more tables.
check	(Also called a *check constraint*) sets restrictions on data added to a specific table.

3. Key Concept-Super key,candidate key,primary key,foreign key.

I. **Super key-** Super Key is the set of all the keys which help to identify rows in a table uniquely. This means that all those columns of a table than capable of identifying the other columns of that table uniquely will all be considered super keys.

II. **Candidate key-** Candidate keys are those attributes that uniquely identify rows of a table. The Primary Key of a table is selected from one of the candidate keys. So, candidate keys have the same properties as the primary keys. There can be more than one candidate keys in a table.

III. **Primary key-** A primary key is a column of a table or a set of columns that helps to identify every record present in that table uniquely. There can be only one primary Key in a table. Also, the primary Key cannot have the same values repeating for any row. Every value of the primary key has to be different with no repetitions.

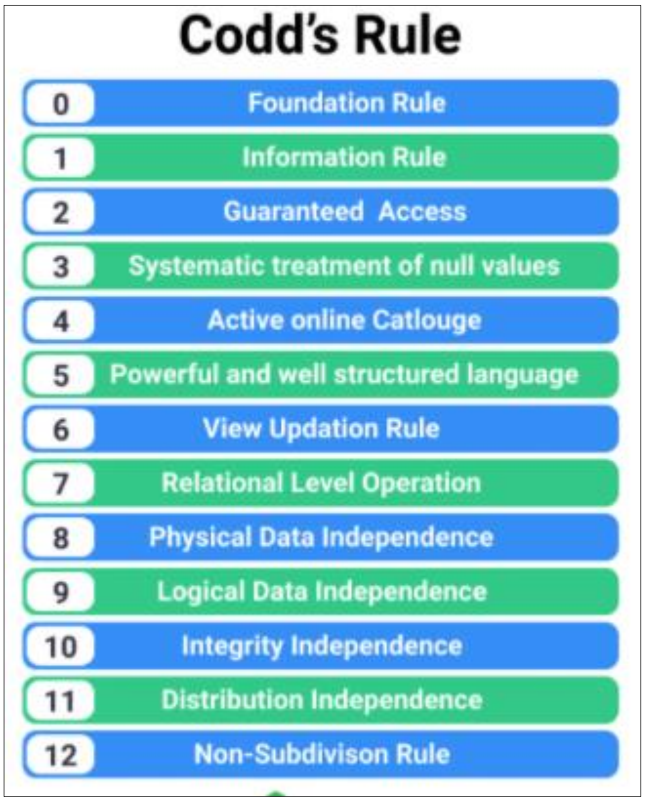
IV. **Foreign key-** Foreign Key is used to establish relationships between two tables. A foreign key will require each value in a column or set of columns to match the Primary Key of the referential table. Foreign keys help to maintain data and referential integrity.

4. Integrity Constraints types with example.

Domain Integrity Constraint	A domain integrity constraint is a set of rules that restricts the kind of attributes or values a column or relation can hold in the database table. For example, we can specify if a particular column can hold null values or not, if the values have to be unique or not, the data type or size of values that can be entered in the column, the default values for the column, etc.
Entity Integrity Constraint	Entity Integrity Constraint is used to ensure the uniqueness of each record or row in the data table. There are primarily two types of integrity constraints that help us in ensuring the uniqueness of each row, namely, UNIQUE constraint and PRIMARY KEY constraint.
Referential Integrity Constraint	Referential Integrity Constraint ensures that there always exists a valid relationship between two tables.This makes sure that if a foreign key exists in a table relationship, then it should always reference a corresponding value in the second table or it should be null. For example, "Department" table and then "Employees" where the "department" attribute references to "Department_ID" in the former table.
Key Constraints	Some of the key constraints in SQL are: 1. Primary key constraints 2. Unique key constraints 3. Foreign Key constraints 4. NOT NULL constraints

	5. Check constraints Detail: https://www.ques10.com/p/17134/explain-types-of-integrity-constraints-with-exampl/

5. Codd's rule with example.



Codd's Rule

0	Foundation Rule
1	Information Rule
2	Guaranteed Access
3	Systematic treatment of null values
4	Active online Catlouge
5	Powerful and well structured language
6	View Updation Rule
7	Relational Level Operation
8	Physical Data Independence
9	Logical Data Independence
10	Integrity Independence
11	Distribution Independence
12	Non-Subdivison Rule

Rule 0: **Foundation rule**	The database must be in relational form. So that the system can handle the database through its relational capabilities.
Rule 1: **Information Rule**	The data stored in a database, may it be user data or metadata, must be a value of some table cell. Everything in a database must be stored in a table format.
Rule 2: **Guaranteed Access Rule**	Every single data element (value) is guaranteed to be accessible logically with a combination of table-name, primary-key (row value), and attribute-name (column value). No other means, such as pointers, can be used to access data.
Rule 3: **Systematic Treatment of NULL Values**	The NULL values in a database must be given a systematic and uniform treatment. This is a very important rule because a NULL can be interpreted as one the following − data is missing, data is not known, or data is not applicable.
Rule 4: **Active Online Catalog**	The structure description of the entire database must be stored in an online catalog, known as data dictionary, which can be accessed by authorized users. Users can use the same query language to access the catalog which they use to access the database itself.
Rule 5: **Comprehensive Data Sub-Language Rule**	A database can only be accessed using a language having linear syntax that supports data definition, data manipulation, and transaction management operations. This language can be used directly or by means of some application. If the database allows

	access to data without any help of this language, then it is considered as a violation.
Rule 6: **View Updating Rule**	All the views of a database, which can theoretically be updated, must also be updatable by the system.
Rule 7: **High-Level Insert, Update, and Delete Rule**	A database must support high-level insertion, updation, and deletion. This must not be limited to a single row, that is, it must also support union, intersection and minus operations to yield sets of data records.
Rule 8: **Physical Data Independence**	The data stored in a database must be independent of the applications that access the database. Any change in the physical structure of a database must not have any impact on how the data is being accessed by external applications.
Rule 9: **Logical Data Independence**	The logical data in a database must be independent of its user's view (application). Any change in logical data must not affect the applications using it. For example, if two tables are merged or one is split into two different tables, there should be no impact or change on the user application. This is one of the most difficult rule to apply.
Rule 10: **Integrity Independence**	A database must be independent of the application that uses it. All its integrity constraints can be independently modified without the need of any change in the application. This rule makes a database independent of the front-end application and its interface.

Rule 11: **Distribution Independence**	The end-user must not be able to see that the data is distributed over various locations. Users should always get the impression that the data is located at one site only. This rule has been regarded as the foundation of distributed database systems.
Rule 12: **Non-Subversion Rule**	If a system has an interface that provides access to low-level records, then the interface must not be able to subvert the system and bypass security and integrity constraints.

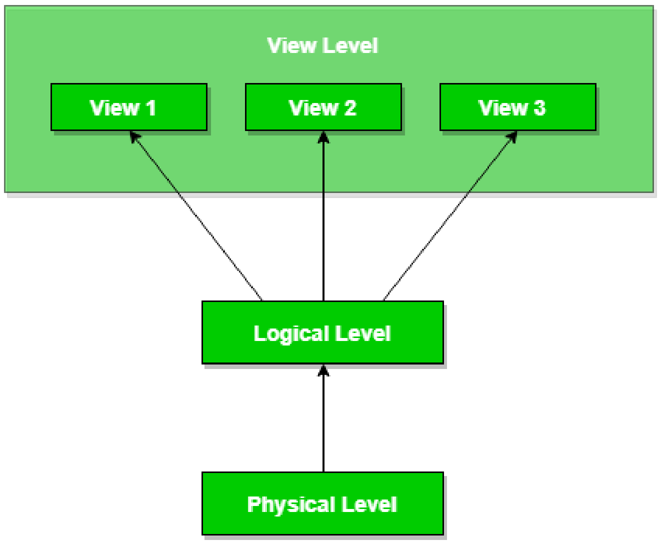
6. Data Independence and Data Abstraction

Answer :Database systems comprise complex data-structures. In order to make the system efficient in terms of retrieval of data, and reduce complexity in terms of usability of users, developers use abstraction i.e. hide irrelevant details from the users. This approach simplifies database design.

There are mainly **3** levels of data abstraction:
Physical: This is the lowest level of data abstraction. It tells us how the data is actually stored in memory. The access methods like sequential or random access and file organization methods like B+ trees, hashing used for the same. Usability, size of memory, and the number of times the records are factors that we need to know while designing the database.
eg. Suppose we need to store the details of an employee. Blocks of storage and the amount of memory used for these purposes are kept hidden from the user.

Logical: This level comprises the information that is actually stored in the database in the form of tables. It also stores the relationship among the data entities in relatively simple structures. At this level, the information available to the user at the view level is unknown.
We can store the various attributes of an employee and relationships, e.g. with the manager can also be stored.

View: This is the highest level of abstraction. Only a part of the actual database is viewed by the users. This level exists to ease the accessibility of the database by an individual user. Users view data in the form of rows and columns. Tables and relations are used to store data. Multiple views of the same database may exist. Users can just view the data and interact with the database, storage and implementation details are hidden from them.



The main purpose of data abstraction is to achieve data independence in order to save time and cost required when the database is modified or altered.

We have namely two levels of data independence arising from these levels of abstraction :

Physical level data independence: It refers to the characteristic of being able to modify the physical schema without any alterations to the conceptual or logical schema, done for optimization purposes, e.g., Conceptual structure of the database would not be affected by any change in storage size of the database system server.

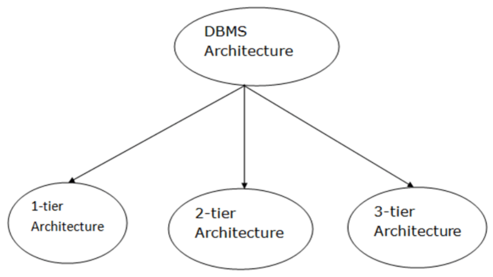
- Utilizing new storage devices.
- Modifying data structures used for storage.
- Altering indexes or using alternative file organization techniques etc.

Logical level data independence: It refers characteristic of being able to modify the logical schema without affecting the external schema or application program. The user view of the data would not be affected by any changes to the conceptual view of the data. These changes may include insertion or deletion of attributes, altering table structures entities or relationships to the logical schema, etc.

7. Database Architecture

- The DBMS design depends upon its architecture. The basic client/server architecture is used to deal with a large number of PCs, web servers, database servers and other components that are connected with networks.
- The client/server architecture consists of many PCs and a workstation which are connected via the network.
- DBMS architecture depends upon how users are connected to the database to get their request done.

Types of the Database Architecture



1-Tier Architecture

- In this architecture, the database is directly available to the user. It means the user can directly sit on the DBMS and uses it.
- Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.
- The 1-Tier architecture is used for development of the local application, where programmers can directly communicate with the database for the quick response.

2-Tier Architecture

- The 2-Tier architecture is same as basic client-server. In the two-tier architecture, applications on the client end can directly communicate with the database at the server side. For this interaction, API's like: **ODBC**, **JDBC** are used.
- The user interfaces and application programs are run on the client-side.
- The server side is responsible to provide the functionalities like: query processing and transaction management.

- To communicate with the DBMS, client-side application establishes a connection with the server side.

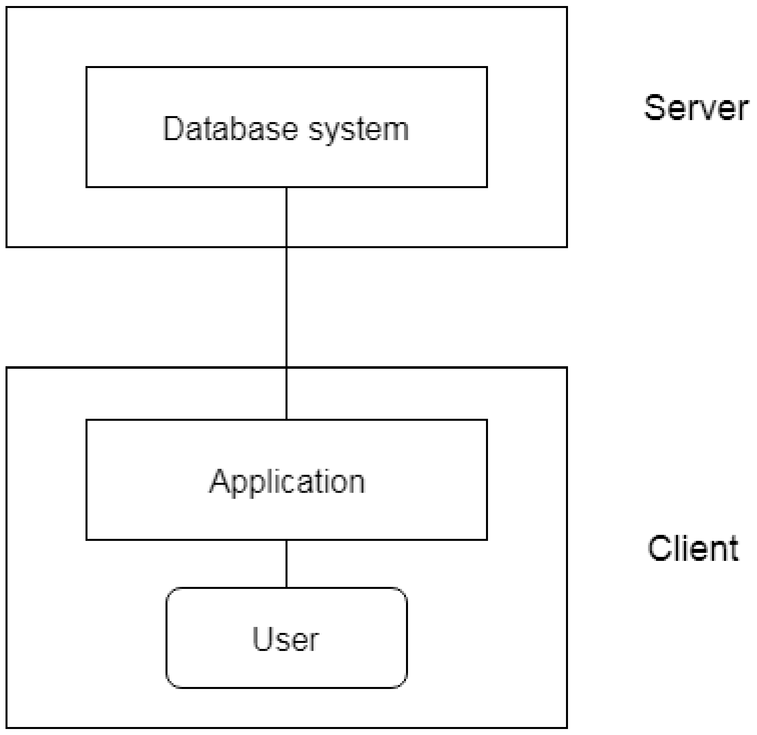


Fig: 2-tier Architecture

3-Tier Architecture

- The 3-Tier architecture contains another layer between the client and server. In this architecture, client can't directly communicate with the server.
- The application on the client-end interacts with an application server which further communicates with the database system.
- End user has no idea about the existence of the database beyond the application server. The database also has no idea about any other user beyond the application.
- The 3-Tier architecture is used in case of large web application.

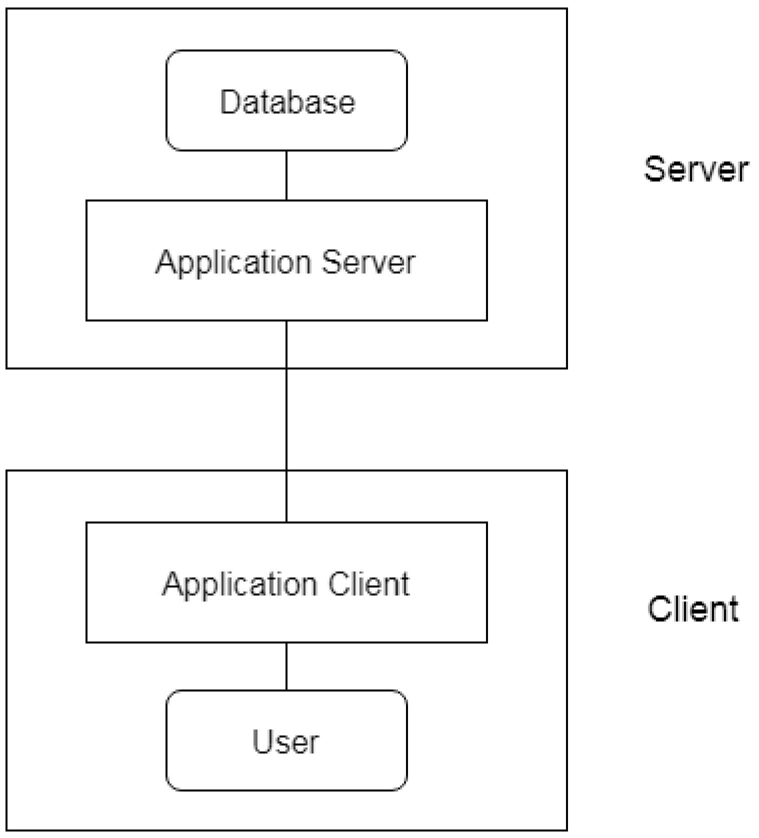


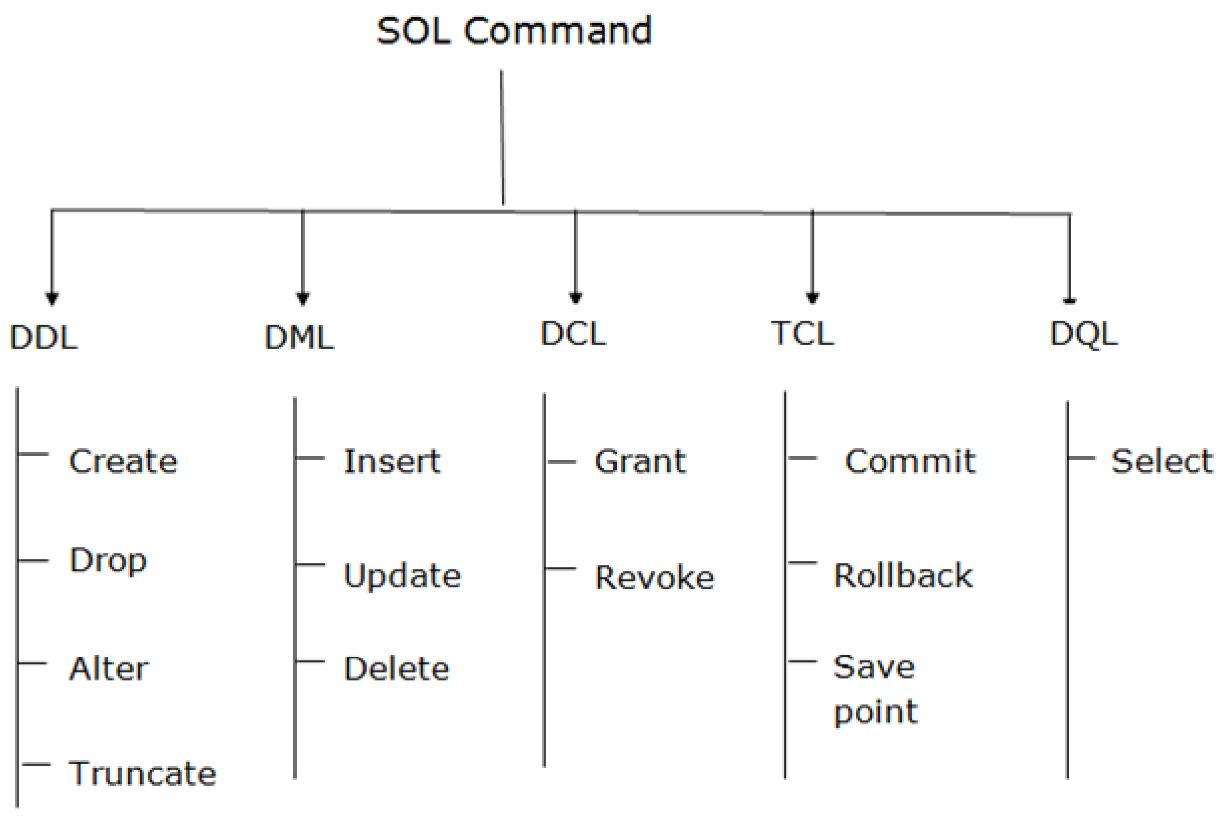
Fig: 3-tier Architecture

8. SQL Commands Types

- SQL commands are instructions. It is used to communicate with the database. It is also used to perform specific tasks, functions, and queries of data.
- SQL can perform various tasks like create a table, add data to tables, drop the table, modify the table, set permission for users.

<u>Types of the SQL Commands</u>

There are five types of SQL commands: DDL, DML, DCL, TCL, and DQL.



1. Data Definition Language(DDL)

DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.

- All the command of DDL are auto-committed that means it permanently save all the changes in the database.

Here are some commands that come under DDL:

- CREATE
- ALTER
- DROP
- TRUNCATE

a. CREATE It is used to create a new table in the database.

Syntax:

```
1. CREATE TABLE TABLE_NAME (COLUMN_NAME DATATYPES[,....]);
```

Example:

1. CREATE TABLE EMPLOYEE(Name VARCHAR2(20), Email VARCHAR2(100), DOB DATE);

b. DROP: It is used to delete both the structure and record stored in the table.

Syntax

1. DROP TABLE ;

Example

1. DROP TABLE EMPLOYEE;

c. ALTER: It is used to alter the structure of the database. This change could be either to modify the characteristics of an existing attribute or probably to add a new attribute.

Syntax:

To add a new column in the table

1. ALTER TABLE table_name ADD column_name COLUMN-definition;

To modify existing column in the table:

1. ALTER TABLE MODIFY(COLUMN DEFINITION....);

EXAMPLE

1. ALTER TABLE STU_DETAILS ADD(ADDRESS VARCHAR2(20));
2. ALTER TABLE STU_DETAILS MODIFY (NAME VARCHAR2(20));

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

Syntax:

1. TRUNCATE TABLE table_name;

Example:

1. TRUNCATE TABLE EMPLOYEE;

2. Data Manipulation Language(DML)

- DML commands are used to modify the database. It is responsible for all form of changes in the database.
- The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.

Here are some commands that come under DML:

- INSERT
- UPDATE
- DELETE

a. INSERT: The INSERT statement is a SQL query. It is used to insert data into the row of a table.

Syntax:

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

Or

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

For example:

```
1. INSERT INTO javatpoint (Author, Subject) VALUES ("Sonoo", "DBMS");
```

b. UPDATE: This command is used to update or modify the value of a column in the table.

Syntax:

```
1. UPDATE table_name SET [column_name1= value1,...column_nameN = valueN] [WHE
   RE CONDITION]
```

For example:

```
1. UPDATE students
2. SET User_Name = 'Sonoo'
3. WHERE Student_Id = '3'
```

c. DELETE: It is used to remove one or more row from a table.

Syntax:

1. DELETE FROM table_name [WHERE condition];

For example:

1. DELETE FROM javatpoint
2. WHERE Author="Sonoo";

3. Data Control Language (DCL)

DCL commands are used to grant and take back authority from any database user.

Here are some commands that come under DCL:

- Grant
- Revoke

a. Grant: It is used to give user access privileges to a database.

Example

1. GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOTHER_USER;

b. Revoke: It is used to take back permissions from the user.

Example

1. REVOKE SELECT, UPDATE ON MY_TABLE FROM USER1, USER2;

4. Transaction Control Language(TCL)

TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.

These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.

Here are some commands that come under TCL:

- COMMIT
- ROLLBACK

- SAVEPOINT

a. Commit: Commit command is used to save all the transactions to the database.

Syntax:

1. COMMIT;

Example:

1. DELETE FROM CUSTOMERS
2. WHERE AGE = 25;
3. COMMIT;

b. Rollback: Rollback command is used to undo transactions that have not already been saved to the database.

Syntax:

1. ROLLBACK;

Example:

1. DELETE FROM CUSTOMERS
2. WHERE AGE = 25;
3. ROLLBACK;

c. SAVEPOINT: It is used to roll the transaction back to a certain point without rolling back the entire transaction.

Syntax:

1. SAVEPOINT SAVEPOINT_NAME;

5.Data Query Language (DQL)

DQL is used to fetch the data from the database.

It uses only one command:

- SELECT

a. SELECT: This is the same as the projection operation of relational algebra. It is used to select the attribute based on the condition described by WHERE clause.

Syntax:

1. SELECT expressions
2. FROM TABLES
3. WHERE conditions;

For example:

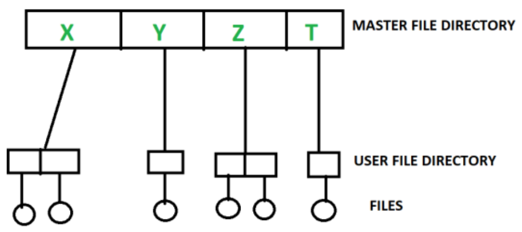
1. SELECT emp_name
2. FROM employee
3. WHERE age > 20;

9. Comparison of RDBMS and File system

1. File System :

File system is basically a way of arranging the files in a storage medium like hard disk. File system organizes the files and helps in retrieval of files when they are required. File systems consists of different files which are grouped into directories. The directories further contain other folders and files. File system performs basic operations like management, file naming, giving access rules etc.
Example:

NTFS(New Technology File System), EXT(Extended File System).

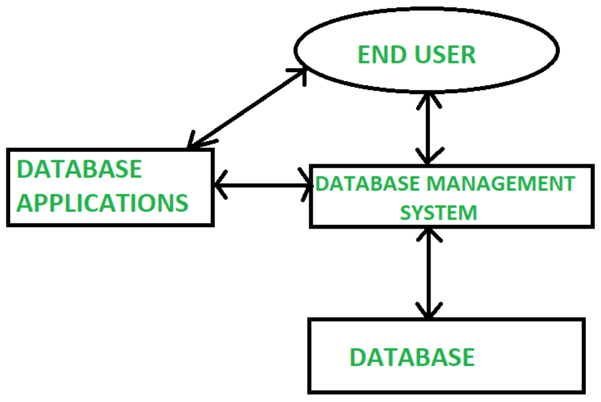


2. DBMS(Database Management System) :

Database Management System is basically a software that manages the collection of related data. It is used for storing data and retrieving the data effectively when it is needed. It also provides proper security measures for protecting the data from unauthorized access. In Database Management System the data can be fetched by SQL queries and relational algebra. It also provides mechanisms for data recovery and data backup.

Example:

Oracle, MySQL, MS SQL server.



S.NO.	**File System**	**DBMS**
1.	File system is a software that manages and organizes the files in a storage medium within a computer.	DBMS is a software for managing the database.
2.	Redundant data can be present in a file system.	In DBMS there is no redundant data.
3.	It doesn't provide backup and recovery of data if it is lost.	It provides backup and recovery of data even if it is lost.
4.	There is no efficient query processing in file system.	Efficient query processing is there in DBMS.
5.	There is less data consistency in file system.	There is more data consistency because of the process of normalization.
6.	It is less complex as compared to DBMS.	It has more complexity in handling as compared to file system.
7.	File systems provide less security in comparison to DBMS.	DBMS has more security mechanisms as compared to file system.
8.	It is less expensive than DBMS.	It has a comparatively higher cost than a file system.

10. DBMS Basic Terminologies

1. **Data:**The raw information (facts and figures) input by the user is called data.
2. **Data Type:**Datatype is used to define the values that a column can contain.
3. **Information:**Processed data is termed as information. Information means meaningful data by relational connection.
4. **Knowledge:**A processed data, i.e., Information when coming into practical use, an understanding of the information through senses, is termed as knowledge.
5. **Database:** A collection of information related to a particular topic or purpose.
6. **Database Management System (DBMS):** A database-management system (DBMS) is a collection of interrelated data and a set of programs to access those data.
7. **Database Model:**A database model shows the logical structure of a database, including the relationships and constraints that determine how data can be stored and accessed.
8. **Table:**A table is a named relational database data set that is organized by rows and columns.
9. **Row:** A row is a collection of fields that make up a record that is relevant to a specific entity.
10. **Column:** A column is a set of data values, all of a single type.
11. **Query:**A query is a request for data or information from a database table or combination of tables.
12. **Record:** A record is a group of fields within a table that reference one particular object.
13. **Relation:**A relation is defined as a set of tuples that have the same attributes.A relation is usually described as a table, which is organized into rows and columns. All the data referenced by an attribute are in the same domain and conform to the same constraints.
14. **Entity:**An entity is a real-world thing which can be distinctly identified like a person, place or a concept. It is an object which is distinguishable from others. If we cannot distinguish it from others then it is an object but not an entity.
15. **Attribute:**An attribute is a property or characteristic of an entity. An entity may contain any number of attributes. One of the attributes is considered as the primary key.
16. **Relational Database:**A relational database organizes data into tables which can be linked—or related—based on data common to each.
17. **Key:**A key is a data item that exclusively identifies a record. In other words, key is a set of column(s) that is used to uniquely identify the record in a table.
18. **Constraint:** Constraints are the rules enforced on the data columns of a table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database. Constraints could be either on a column level or a table level.
19. **Database Schema:**A database schema is the logical structure that represents the logical view of the entire database. It defines how the data is organized and how the

relations among them are associated. It formulates all the constraints that are to be applied on the data.

20. **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.
21. **Cardinality:** Cardinality refers to the relationships between the data in two database tables
22. **Mapping Cardinality:** Mapping cardinality, or cardinality ratio, express the number of entities to which another entity can be associated via a relationship set.
23. **Transaction:**A database transaction is a series of operations performed within a databasemanagement system against a database such that, once completed, the data is left in a reliable and consistent state.
24. **View:**A database view is a subset of a database and is based on a query that runs on one or more database tables.
25. **Join:**In DBMS, a join statement is mainly used to combine two tables based on a specified common field between them. If we talk in terms of Relational algebra, it is the cartesian product of two tables followed by the selection operation.

11.Application of RDBMS

1. It is used for maintaining small as well as large databases of the firms, companies and organizations.
2. It is used for managing inventory and stock, payroll etc.
3. It is used as an analysis tool for managing and processing data warehouses and data mining for better understanding and decision making.
4. It is used for recording and managing day to day activities and transactions such as production, selling, income and expenses, purchase.
5. Used in hospitals, banks, railway, school, colleges for managing their routine activities.
6. It is used for financial planning and developing business strategy.
7. It can be useful for monitoring financial and economical state of the organization.

12.Advantage RDBMS over File System

1. No redundant data: Redundancy removed by data normalization. No data duplication saves storage and improves access time.
2. Data Consistency and Integrity: The root cause of data inconsistency is data redundancy, since data normalization takes care of the data redundancy, data inconsistency also been taken care of as part of it
3. Data Security: It is easier to apply access constraints in database systems so that only authorized user is able to access the data. Each user has a different set of access thus data is secured from the issues such as identity theft, data leaks and misuse of data.

4. Privacy: Limited access means privacy of data.
5. Easy access to data – Database systems manages data in such a way so that the data is easily accessible with fast response times.
6. Easy recovery: Since database systems keeps the backup of data, it is easier to do a full recovery of data in case of a failure.
7. Flexible: Database systems are more flexible than file processing systems.

13.Data Definition Language with constraint and example

Data Definition Language (DDL) is used to define database structure or pattern. It is used to create schema, tables, indexes, constraints, etc. in the database. Using the DDL statements, you can create the skeleton of the database. Data definition language is used to store the information of metadata like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc.

Sr. No.	DDL Command	Description
1	Create	It is used to create objects in the database.
2	Alter	It is used to alter the structure of the database.
3	Drop	It is used to delete objects from the database.
4	Truncate	It is used to remove all records from a table.
5	Rename	It is used to rename an object.
6	Comment	It is used to comment on the data dictionary.

1) Create:

CREATE DATABASE

```
Syntax:             CREATE DATABASE <DatabaseName>
Example:        CREATE DATABASE ExampleDB;
```

CREATE TABLE

```
Syntax:             CREATE TABLE <TableName> (
                <Column1><DataType>,
                <Column2><DataType>,
                <Column3><DataType>,
                .
                .
                .
                <ColumnN><DataType>,
                )
Example:        CREATE TABLE Employees (
                Id INT,
                Name VARCHAR(50),
                Phone BIGINT,
```

IsContractor BIT
);

2) Alter:
ALTER TABLE
Add a column to the table.
Syntax: ALTER TABLE <TableName> ADD <ColumnName><DataType>
Example: ALTER TABLE Employees ADD Department VARCHAR(20);

Modify a column of the table.
Syntax: ALTER TABLE <TableName> MODIFY <ColumnName><DataType>
Example: ALTER TABLE Employees MODIFY Department INT;

Rename the table.
Syntax: ALTER TABLE <TableName> RENAME TO <NewTableName>
Example: ALTER TABLE Employees RENAME TO Employee;

Rename the column:
Syntax: ALTERTABLE<TableName> RENAME COLUMN <ColumnName> TO<NewColumnName>
Example: ALTER TABLE Employees RENAME COLUMN Department TO DepartmentId;

Delete the column:
Syntax: ALTER TABLE <TableName> DROP COLUMN <ColumnName>
Example: ALTER TABLE Employees DROP COLUMN DepartmentId;

3) Drop:
DROP TABLE:
Syntax: DROP TABLE <TableName>
Example: DROP TABLE Employee;

DROP DATABASE:
Syntax: DROP DATABASE <DatabaseName>
Example: DROP DATABASE ExampleDB;

4) Truncate:
Syntax: TRUNCATE TABLE <table_name>;
Example: TRUNCATE TABLEEmployee;

Constraints in SQL:

SQL constraints are used to specify rules for the data in a table. The following constraints are commonly used in SQL:

Sr. No.	Constraint	Description
1	NOT NULL	Ensures that a column cannot have a NULL value.
2	UNIQUE	Ensures that all values in a column are different.
3	PRIMARY KEY	A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table.
4	FOREIGN KEY	Prevents actions that would destroy links between tables.
5	CHECK	Ensures that the values in a column satisfies a specific condition.
6	DEFAULT	Sets a default value for a column if no value is specified.
7	CREATE INDEX	Used to create and retrieve data from the database very quickly.

1) NOT NULL on CREATE TABLE:

Example:
```
CREATE TABLE Employees (
Id INT NOT NULL,
Name VARCHAR(50),
Phone BIGINT,
IsContractor BIT
);
```

NOT NULL on ALTER TABLE:

Example:
```
ALTER TABLE Employeed

MODIFY Name VARCHAR(50) NOT NULL;
```

2) UNIQUE on CREATE TABLE:

Example:
```
CREATE TABLE Employees (
Id INT NOT NULL UNIQUE,
Name VARCHAR(50),
Phone BIGINT,
IsContractor BIT
);
```

3) PRIMARY KEY on CREATE TABLE:

Example:
```
CREATE TABLE Employees (
Id INT NOT NULL,
Name VARCHAR(50),
Phone BIGINT,
IsContractor BIT
PRIMARY KEY (Id)
);
```

4) FOREIGN KEY on CREATE TABLE:
   Example: CREATE TABLE Employees (

```
Id INT NOT NULL,
DepartmentId INT,
Name VARCHAR(50),
Phone BIGINT,
IsContractor BIT
PRIMARY KEY (Id)
FOREIGN KEY (DepartmentId) REFERENCES
Departments(DepartmentId)
);
```

14.Data Manipulation Language with example

Data Manipulation Language (DML) deals with data manipulation and includes most common SQL statements such SELECT, INSERT, UPDATE, DELETE, etc., and it is used to store, modify, retrieve, delete and update data in a database.

Sr. No.	DML Command	Description
1	Insert	It is used to insert data into a table.
2	Update	It is used to update existing data within a table.
3	Delete	It is used to delete all records from a database table.

1) Insert:
   Syntax:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);

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

   Example:

```
INSERT INTO Eomployees (Id, DepartmentId, Name, Phone,) VALUES
(111, 222,  'Tom B. Erichsen', 9876543278);
```

2) Update:
   Syntax:

```
UPDATE <table_name>
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

   Example:

```
UPDATE Employees

SET Name = 'Alfred Schmidt'

WHERE Id = 1;
```

3) Delete:

Delete single record.

Syntax: DELETE FROM table_name WHERE condition;

Example: DELETE FROM Employees WHERE Name='AlfredsFutterkiste';

Delete all records.

Syntax: DELETE FROM <table_name>;

Example: DELETE FROM Employees;

15.Data Query Language with example

Data Query Language(DQL) statements are used for performing queries on the data within schema objects. The purpose of the DQL Command is to get some schema relation based on the query passed to it.

Sr. No.	DQL Command	Description
1	Select	It is used to is used to retrieve data from the database.

1) Select:

Syntax:
```
SELECT column1, column2, ...
FROM table_name;

SELECT * FROM table_name;
```

Example:
```
SELECT Name, Phone FROM Employees;

SELECT * FROM Employees;
```

16.Transaction Query Language

Transaction Control Language (TCL). TCL commands deal with the transaction within the database. These commands are used to manage the changes made by DML statements. TCL allows the statements to be grouped together into logical transactions.

Sr. No.	TQL Command	Description
1	Commit	It commits a transaction.
2	Rollback	It is used to rollback a transaction in case of any error occurs.
3	Savepoint	It is used to set a savepoint within a transaction.
4	Set Transaction	It is used to specify characteristics for the transaction.

1) Commit:
   Syntax: COMMIT;
   Example:
```
DELETE FROM Employees
WHERE Id = 1;
COMMIT;
```

2) Rollback:
   Syntax: ROLLBACK;
   Example:
```
DELETE FROM Employees
WHERE Id = 1;
ROLLBACK;
```

3) Savepoint:
   Syntax: SAVEPOINT SAVEPOINT_NAME;
   Example:
```
DELETE FROM Employees
WHERE Id = 1;
SAVEPOINT SP1;
DELETE FROM Employees
WHERE Id = 2;
ROLLBACK TO SP1;
```

4) Set Transaction:
   Syntax: SET TRANSACTION [READ WRITE | READ ONLY];

16.Data Control Language

Data Control Language (DCL) is used to control user access in a database.These commandsare related to the security issues. It allows or restricts the user from accessing data in database schema.

Sr. No.	**DCL Command**	**Description**
1	Grant	It gives user’s access privileges to the database.
2	Revoke	It withdraws user’s access privileges given by using the GRANT command.

1) Grant:
   Syntax:
```
            GRANT <privilege list>
ON <relation name or view name>
TO <user/role list>;
```

   Example:
```
GRANT ALL

ON Employees
```

```
TO ABC;
[WITH GRANT OPTION]
```

2) Revoke:

Syntax:
```
REVOKE <privilege list>
ON <relation name or view name>
FROM <user name>;
```

Example:
```
REVOKE UPDATE
ON Employee
FROM ABC;
```

17. SQL Queries

CREATE DATABASE - creates a new database :

```
CREATE DATABASE database_name;
USE database_name;
```

CREATE TABLE - creates a new table

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

ALTER TABLE - modifies a table

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

DROP TABLE - deletes a table

```
DROP TABLE table_name;
```

SELECT - extracts data from a database

```
SELECT Name
FROM Student

SELECT Name,SurName
FROM  Student
```

INSERT INTO - inserts new data into a database
INSERT INTO *table_name* (*column1*, *column2*, *column3*, ...)
VALUES (*value1*, *value2*, *value3*, …);

UPDATE - updates data in a database
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;

DELETE - deletes data from a database
DELETE FROM *table_name* WHERE *condition*;

18. Clauses, Aggregate Function.

A. What are Clauses in SQL?

1. Clauses are in-built functions available to us in SQL. With the help of clauses, we can deal with data easily stored in the table.
2. Clauses help us filter and analyze data quickly. When we have large amounts of data stored in the database, we use Clauses to query and get data required by the user.
3. Some of the examples of clauses are – where, and, or, like, top, etc.

The following are the various SQL clauses:

1. **Group By**
2. **Having**
3. **Order By**

Student_Number	**Student_Name**	**Student_Phone**	**Student_Marks**	**Student_Major Subject**
1	Andrew	6615927284	95	Literature
2	Sara	6583654865	65	Maths
3	Harry	4647567463	48	Literature
4	Sally	6537837084	30	Literature
5	Anne	7457337732	88	Maths

1. Group by

- SQL group by statement is used to arrange identical data into groups. The group by statement is used with the SQL select statement.
- The group by statement follows the where clause in a select statement and precedes the order by clause.
- The group by statement is used with aggregation function.

Syntax:

SELECT column FROM table_name WHERE conditions   GROUP BY column ORDER BY column

Example:

Select Count (Student_Number), Student_MajorSubjectFrom StudentGroup byStudent_MajorSubject

Result:

Count(Student_number)	**Student_MajorSubject**
3	Literature
2	Maths

2. Having

- HAVING clause is used to specify a search condition for a group or an aggregate.
- Having is used in a GROUP BY clause. If you are not using GROUP BY clause then you can use HAVING function like a WHERE clause.

Syntax:

SELECT column1, column2   FROM table_name WHERE conditions   GROUP BY column1, column2   HAVING conditions ORDER BY column1, column2;

Example:

Select Count(Student_number), Student_MajorSubjectFrom StudentGroup by Student_MajorSubjectHaving Count(Student_Number) > 2

Result –

Count (Student_Number)	**Student_MajorSubject**

3	Literature

3. ORDER BY

- The ORDER BY clause sorts the result-set in ascending or descending order.
- It sorts the records in ascending order by default. DESC keyword is used to sort the records in descending order

Syntax:

```
SELECT column1, column2  FROM table_name  WHERE condition  ORDER BY column1, co
lumn2... ASC|DESC;
```

Where:

ASC: It is used to sort the result set in ascending order by expression.

DESC: It sorts the result set in descending order by expression.

Example:

Select Student_NameFrom StudentWhere Student_Marks>50Order by Student_Marks

Result –

Student_Name
Sara
Anne
Andrew

B. What are aggregate functions?

- Aggregate functions perform a calculation on a set of values and return a single value.
- Aggregate functions ignore NULL values except COUNT.
- It is used with the GROUP BY clause of the SELECT statement.

Following are the Aggregate functions:

1. AVG
2. MAX

3. MIN
4. SUM
5. COUNT()
6. COUNT(*)

Example :<Employee> Table

Eid	Ename	Age	City	Salary
E001	ABC	29	Pune	20000
E002	PQR	30	Pune	30000
E003	LMN	25	Mumbai	5000
E004	XYZ	24	Mumbai	4000
E005	STU	32	Bangalore	25000

Aggregate Functions	Description	Syntax	Example	Output
AVG	It returns the average of the data values.	SELECT AVG <column_name> FROM <table_name>;	SELECT AVG(Salary) FROM Employee;	**AVG(Salary)** 16800
MAX	It returns the maximum value for a column.	SELECT MAX <column_name> FROM <table_name>;	SELECT MAX(Salary) FROM Employee;	**MAX(Salary)** 30000
MIN	It returns the minimum value for a column.	SELECT MIN <column_name> FROM <table_name>;	SELECT MIN(Salary) FROM Employee;	**MIN(Salary)** 4000
SUM	It returns the sum (addition) of the data values.	SELECT SUM <column_name> FROM <table_name>;	SELECT SUM(Salary) FROM Employee WHERE City='Pune';	**SUM(Salary)** 50000

COUNT()	It returns total number of values in a given column.	SELECT COUNT <column_name> FROM <table_name>;	SELECT COUNT(Empid) FROM Employee;	**COUNT(Empid)** 5
COUNT(*)	It returns the number of rows in a table.	SELECT COUNT(*) FROM <table_name>;	SELECT COUNT(*) FROM Employee;	**COUNT(*)** 5

19. Join its type and example.

What is Join in DBMS?

Join in DBMS is a binary operation which allows you to combine join product and selection in one single statement. The goal of creating a join condition is that it helps you to combine the data from two or more DBMS tables. The tables in DBMS are associated using the primary key and foreign keys.

Types of JOINS are

1. **Cross JOIN or Cartesian Product**
2. **Inner Join**
3. **Outer Join**

1. Cross JOIN or Cartesian Product

This type of JOIN returns the cartesian product of rows from the tables in Join. It will return a table which consists of records which combines each row from the first table with each row of the second table.

Cross JOIN Syntax is,

SELECT column-name-listFROM table-name1 CROSS JOIN table-name2;

Example of Cross JOIN
Following is the class table,

ID	NAME
1	abhi
2	adam
4	alex

and the class_info table,

ID	Address
1	DELHI

2	MUMBAI
3	CHENNAI

Cross JOIN query will be,

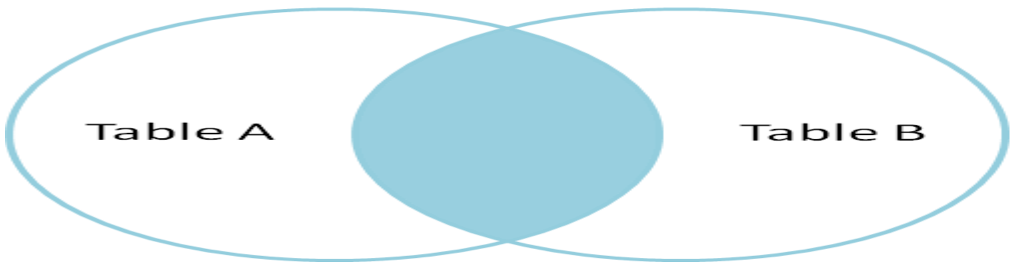
SELECT*FROMclass CROSSJOIN class_info;

The resultset table

ID	**NAME**	**ID**	**Address**
1	abhi	1	DELHI
2	adam	1	DELHI
4	alex	1	DELHI
1	abhi	2	MUMBAI
2	adam	2	MUMBAI
4	alex	2	MUMBAI
1	abhi	3	CHENNAI
2	adam	3	CHENNAI
4	alex	3	CHENNAI

2. Inner Join

The INNER JOIN keyword selects all rows from both the tables as long as the condition satisfies. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be same.



Inner Join Syntax:

SELECT column-name-list FROM table-name1 INNER JOIN table-name2 WHERE table-name1.column-name = table-name2.column-name;

Example of INNER JOIN
Consider a class table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu

and the class_info table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI

Inner JOIN query will be,

SELECT * from class INNER JOIN class_info where class.id = class_info.id;

The resultset table will look like,

ID	**NAME**	**ID**	**Address**
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI

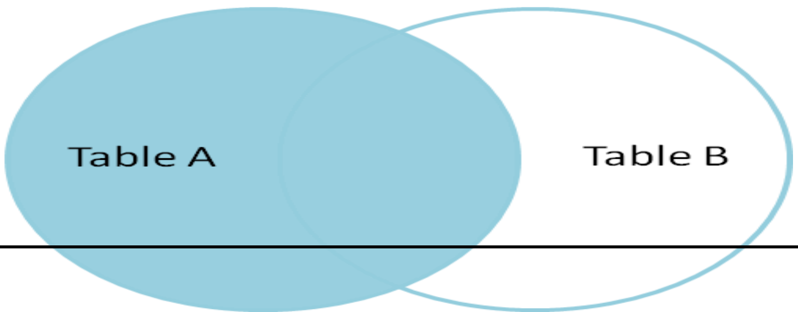
3. Outer Join

The outer join operation is an extension of the join operation. It is used to deal with missing information.

1. Left Outer Join
2. Right Outer Join
3. Full Outer Join

1. Left Outer Join

This join returns all the rows of the table on the left side of the join and matching rows for the table on the right side of join. The rows for which there is no matching row on right side, the result-set will contain *null*. LEFT JOIN is also known as LEFT OUTER JOIN.



Syntax for Left Outer Join:

SELECT column-name-list FROM table-name1 LEFT OUTER JOIN table-name2

ON table-name1.column-name = table-name2.column-name;

To specify a condition, we use the ON keyword with Outer Join.

Left outer Join Syntax for **Oracle** is,

SELECT column-name-list FROM table-name1, table-name2 on table-name1.column-name = table-name2.column-name(+);

Example of Left Outer Join

Here is the **class** table,

ID	**NAME**
1	abhi
2	adam
3	alex
4	anu
5	ashish

and the **class_info** table,

ID	**Address**
1	DELHI
2	MUMBAI
3	CHENNAI
7	NOIDA

8	PANIPAT

Left Outer Join query will be,

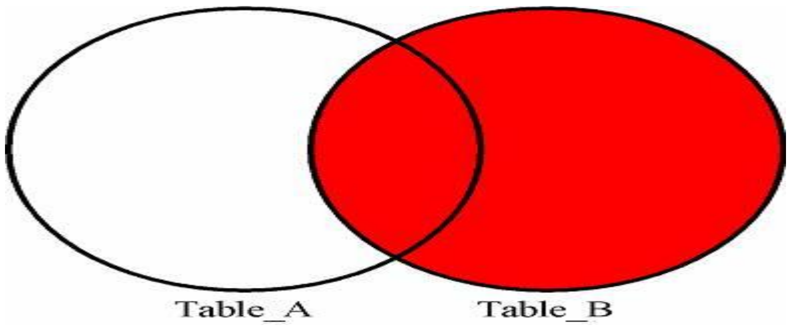
SELECT * FROM class LEFT OUTER JOIN class_info ON (class.id = class_info.id);

The resultset table will look like,

ID	**NAME**	**ID**	**Address**
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI
4	anu	null	null
5	ashish	null	null

2. RIGHT Outer Join

The right outer join returns a resultset table with the **matched data** from the two tables being joined, then the remaining rows of the **right** table and null for the remaining **left** table's columns.



Syntax for Right Outer Join is,

SELECTcolumn-name-list FROMtable-name1 RIGHTOUTERJOINtable-name2
ONtable-name1.column-name =table-name2.column-name;

Right outer Join Syntax for **Oracle** is,
SELECTcolumn-name-list FROMtable-name1,table-name2
ONtable-name1.column-name(+)=table-name2.column-name;

Example of Right Outer Join

Once again the **class** table,

ID	**NAME**
1	abhi
2	adam
3	alex
4	anu
5	ashish

and the **class_info** table,

ID	**Address**
1	DELHI
2	MUMBAI
3	CHENNAI
7	NOIDA
8	PANIPAT

Right Outer Join query will be,

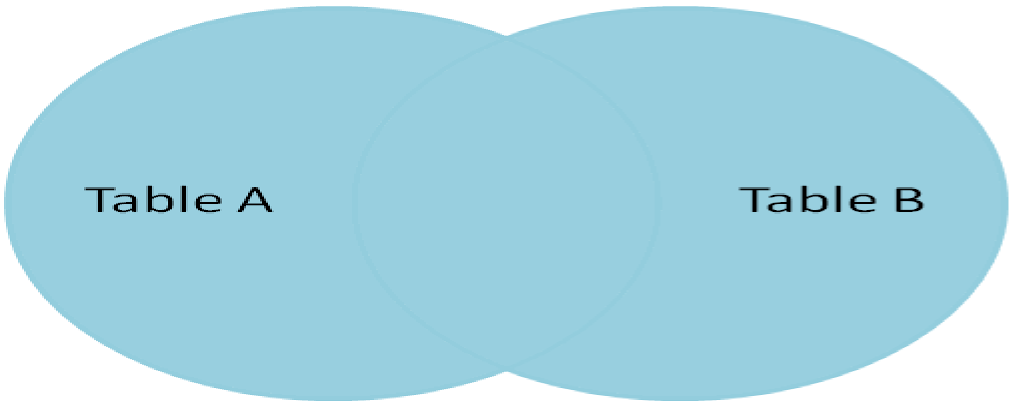
SELECT*FROM class RIGHTOUTERJOIN class_info ON(class.id = class_info.id);

The resultant table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI
null	null	7	NOIDA
null	null	8	PANIPAT

3.Full Outer Join

The full outer join returns a resultset table with the **matched data** of two table then remaining rows of both **left** table and then the **right** table.



Syntax of Full Outer Join is,

SELECTcolumn-name-list FROM
table-name1 FULLOUTERJOINtable-name2
ONtable-name1.column-name =table-name2.column-name;

Example of Full outer join is,
The **class** table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu
5	ashish

and the **class_info** table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI
7	NOIDA
8	PANIPAT

Full Outer Join query will be like,

SELECT*FROM class FULLOUTERJOIN class_info ON(class.id = class_info.id);

The resultset table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI
4	anu	null	null
5	ashish	null	null
null	null	7	NOIDA
null	null	8	PANIPAT

20.trigger concept

Trigger: A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

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.

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

Syntax:

```
create trigger [trigger_name]

[before | after]

{insert | update | delete}
```

on [table_name]

[for each row]

[trigger_body]

Explanation of syntax:

create trigger [trigger_name]: Creates or replaces an existing trigger with the trigger_name.

[before | after]: This specifies when the trigger will be executed.

{insert | update | delete}: This specifies the DML operation.

on [table_name]: This specifies the name of the table associated with the trigger.

[for each row]: This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected.

[trigger_body]: This provides the operation to be performed as trigger is fired

BEFORE and AFTER of Trigger:

BEFORE triggers run the trigger action before the triggering statement is run.

AFTER triggers run the trigger action after the triggering statement is run.

Example:

Given Student Report Database, in which student marks assessment is recorded. In such schema, create a trigger so that the total and average of specified marks is automatically inserted whenever a record is insert.

Here, as trigger will invoke before record is inserted so, BEFORE Tag can be used.

Suppose the database Schema –

```
    mysql> desc Student;

+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
| tid   | int(4)      | NO   | PRI | NULL    | auto_increment |
| name  | varchar(30) | YES  |     | NULL    |                |
| subj1 | int(2)      | YES  |     | NULL    |                |
| subj2 | int(2)      | YES  |     | NULL    |                |
| subj3 | int(2)      | YES  |     | NULL    |                |
| total | int(3)      | YES  |     | NULL    |                |
| per   | int(3)      | YES  |     | NULL    |                |
+-------+-------------+------+-----+---------+----------------+
7 rows in set (0.00 sec)
```

SQL Trigger to problem statement.

```
create trigger stud_marks
before INSERT
on
Student
for each row
set Student.total = Student.subj1 + Student.subj2 + Student.subj3, Student.per = Student.total * 60 / 100;
```

Above SQL statement will create a trigger in the student database in which whenever subjects marks are entered, before inserting this data into the database, trigger will compute those two values and insert with the entered values. i.e.,

```
mysql> insert into Student values(0, "ABCDE", 20, 20, 20, 0, 0);
Query OK, 1 row affected (0.09 sec)
mysql> select * from Student;

+-----+-------+-------+-------+-------+-------+------+
| tid | name  | subj1 | subj2 | subj3 | total | per  |
+-----+-------+-------+-------+-------+-------+------+
| 100 | ABCDE |    20 |    20 |    20 |    60 |   36 |
+-----+-------+-------+-------+-------+-------+------+
1 row in set (0.00 sec)
```

In this way trigger can be creates and executed in the databases.

21.Views in SQL

- Views in SQL are considered as a virtual table. A view also contains rows and columns.
- To create the view, we can select the fields from one or more tables present in the database.
- A view can either have specific rows based on certain condition or all the rows of a table.

Sample table:Student_Detail

STU_ID	NAME	ADDRESS
1	Stephan	Delhi
2	Kathrin	Noida
3	David	Ghaziabad
4	Alina	Gurugram

Student_Marks

STU_ID	NAME	MARKS	AGE
1	Stephan	97	19
2	Kathrin	86	21
3	David	74	18
4	Alina	90	20
5	John	96	18

1. Creating view

A view can be created using the **CREATE VIEW** statement. We can create a view from a single table or multiple tables.

Syntax:

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

2. Creating View from a single table

In this example, we create a View named DetailsView from the table Student_Detail.

Query:

```
CREATE VIEW DetailsView AS
SELECT NAME, ADDRESS
FROM Student_Details
WHERE STU_ID < 4;
```

Just like table query, we can query the view to view the data.

```
1.  SELECT * FROM DetailsView;
```

Output:

NAME	ADDRESS
Stephan	Delhi
Kathrin	Noida
David	Ghaziabad

3. Creating View from multiple tables

View from multiple tables can be created by simply include multiple tables in the SELECT statement.

In the given example, a view is created named MarksView from two tables Student_Detail and Student_Marks.

Query:

```
CREATE VIEW MarksView AS
SELECT Student_Detail.NAME, Student_Detail.ADDRESS, Student_Marks.MARKS
```

```
FROM Student_Detail, Student_Mark
WHERE Student_Detail.NAME = Student_Marks.NAME;
```

To display data of View MarksView:

```
SELECT * FROM MarksView;
```

NAME	ADDRESS	MARKS
Stephan	Delhi	97
Kathrin	Noida	86
David	Ghaziabad	74
Alina	Gurugram	90

4. Deleting View

A view can be deleted using the Drop View statement.

Syntax: DROP VIEW view_name;

Example:

If we want to delete the View **MarksView**, we can do this as:

```
DROP VIEW MarksView;
```

22. Normalization concept

Normalization is the process of minimizing **redundancy** from a relation or set of relations. Redundancy in relation may cause insertion, deletion and updation anomalies. So, it helps to minimize the redundancy in relations. **Normal forms** are used to eliminate or reduce redundancy in database tables.

Here are the most commonly used normal forms:

- First normal form(1NF)
- Second normal form(2NF)
- Third normal form(3NF)
- Boyce & Codd normal form (BCNF)

First Normal Form

First Normal Form is defined in the definition of relations (tables) itself. This rule defines that all the attributes in a relation must have atomic domains. The values in an atomic domain are indivisible units.

Course	Content
Programming	Java, c++
Web	HTML, PHP, ASP

We re-arrange the relation (table) as below, to convert it to First Normal Form.

Course	Content
Programming	Java
Programming	c++
Web	HTML
Web	PHP
Web	ASP

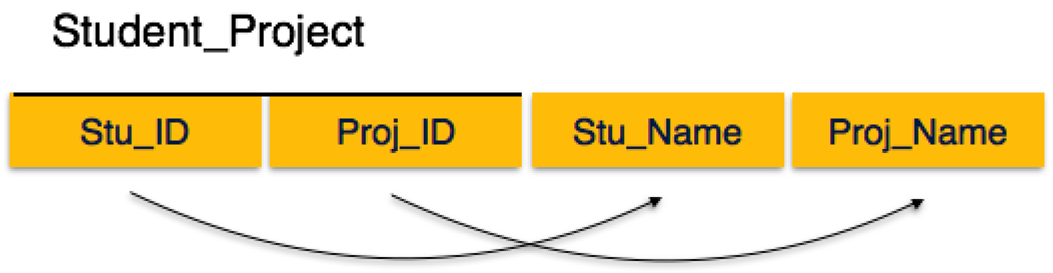
Each attribute must contain only a single value from its pre-defined domain.

Second Normal Form

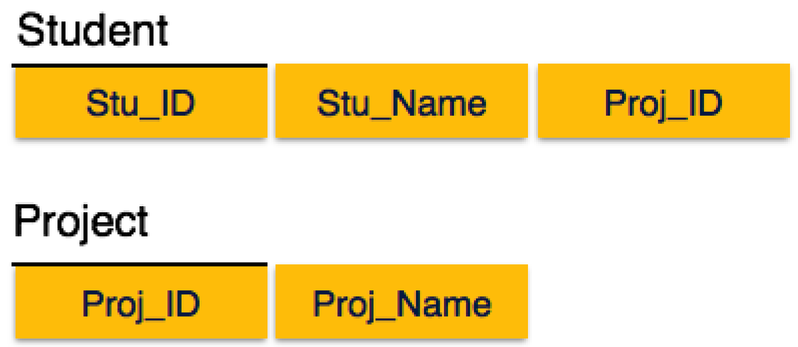
Before we learn about the second normal form, we need to understand the following −

- **Prime attribute** − An attribute, which is a part of the candidate-key, is known as a prime attribute.
- **Non-prime attribute** − An attribute, which is not a part of the prime-key, is said to be a non-prime attribute.

If we follow second normal form, then every non-prime attribute should be fully functionally dependent on prime key attribute. That is, if X → A holds, then there should not be any proper subset Y of X, for which Y → A also holds true.



We see here in Student_Project relation that the prime key attributes are Stu_ID and Proj_ID. According to the rule, non-key attributes, i.e. Stu_Name and Proj_Name must be dependent upon both and not on any of the prime key attribute individually. But we find that Stu_Name can be identified by Stu_ID and Proj_Name can be identified by Proj_ID independently. This is called **partial dependency**, which is not allowed in Second Normal Form.

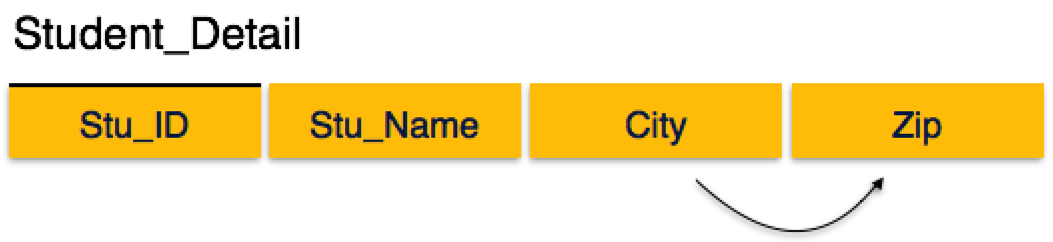


We broke the relation in two as depicted in the above picture. So there exists no partial dependency.

Third Normal Form

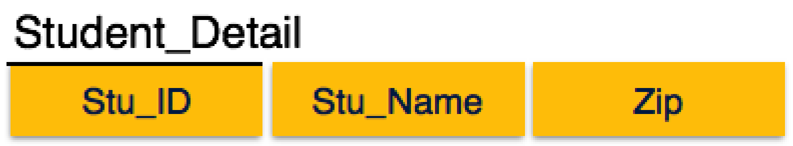
For a relation to be in Third Normal Form, it must be in Second Normal form and the following must satisfy –

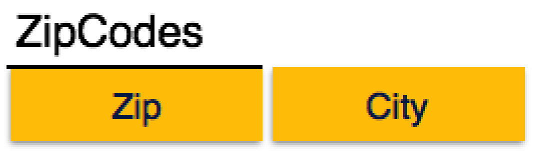
- No non-prime attribute is transitively dependent on prime key attribute.
- For any non-trivial functional dependency, X → A, then either –
    - X is a superkey or,
    - A is prime attribute.



We find that in the above Student_detail relation, Stu_ID is the key and only prime key attribute. We find that City can be identified by Stu_ID as well as Zip itself. Neither Zip is a superkey nor is City a prime attribute. Additionally, Stu_ID → Zip → City, so there exists **transitive dependency**.

To bring this relation into third normal form, we break the relation into two relations as follows −





Boyce-Codd Normal Form

Boyce-Codd Normal Form (BCNF) is an extension of Third Normal Form on strict terms. BCNF states that −

- For any non-trivial functional dependency, X → A, X must be a super-key.

In the above image, Stu_ID is the super-key in the relation Student_Detail and Zip is the super-key in the relation ZipCodes. So,

Stu_ID → Stu_Name, Zip

and

Zip → City

Which confirms that both the relations are in BCNF.

Normal	Description

Form	
1NF	A relation is in 1NF if it contains an atomic value.
2NF	A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
3NF	A relation will be in 3NF if it is in 2NF and no transition dependency exists.
4NF	A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
5NF	A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.

23. Different RDBMS system with its features

A relational database management system (RDBMS) is a program that allows you to create, update, and administer a relational database. Most relational database management systems use the SQL language to access the database.

Types of RDBMS system:

1. Oracle
2. MySQL Database
3. Microsoft SQL Server
4. PostgreSQL Database
5. DB2

1. Oracle

Oracle database came into existence in the late 70s and has many versions available for use. It is compatible with the cloud and is deployable on one or more servers.

Also, the logical data does not affect the physical data. We get an upgraded level of security as the transactions are done in different sessions thus, avoiding any possibility of a clash.

Features:

- Provides the latest innovations and features.
- Oracle DBMS tools are incredibly robust thus, capable of performing almost any
- possible task.

2. MySQL Database

One of the most popular databases for all the available technological requirements. It is freeware and thus is ideal for both small and large-scale organizations.

MySQL provides us with the choice to configure the data types to accommodate any possible data we have. Also, it is reliable and has no large resources required.

Features:

- It is freeware thus cost-efficient.
- Provides a large number of functionalities.
- It supports various user interfaces for easy use.
- Compatible with other DBMS like Oracle and DB2.

3. Microsoft SQL Server

This DBMS works on a cloud-based system or a local server.

Some of the major features include the tracing facility of any changes in the data. It also allows dynamic data masking which helps in protecting the sensitive data stored.

Features:

- This is a reliable and fast DBMS.
- Has the ability to adjust to available resources, hence is very resource-efficient.
- Provides easy visualizations for mobile devices.
- Compatible with all Microsoft products.

4. PostgreSQL Database

One of the major free popular databases and is frequently used in web applications. It supports deployment in various environments i.e. virtual, physical, and cloud-based environments.

The newer versions have support for large volumes of data. Security has also improved.

Features:

- A scalable and adjustable DBMS.
- Provide built-in support for the JSON data.
- Provides plenty of predefined functions.
- It is available in various versions of user interfaces.

5. DB2

It is the DBMS that was found by IBM for their internal use and was later released for public use. One of the most important and evident features is faster-skipping technology.

Data skipping helps in increasing the system speed and boosts the effective use of resources. It has added disaster recovery options that increase reliability and compatibility.

Features:

- Due to the high-speed, the handling of enormous data becomes easier.
- Compatible with cloud, physical server, or both at the same time.
- Task automation is available as it provides the task scheduler.
- Provides proper error codes and exit codes which make debugging easier.