AVVIARE
EDUCATIONAL HUB

Study Material

BCA

Second Semester

Database Management System

INDEX

BCA-124 DATABASE MANAGEMENT SYSTEM

Course Contents

UNIT I

Database Management System: Introduction, Definition of DBMS, File processing system Vs DBMS, Limitation of file processing system, Comparison of File processing system and DBMS, Advantages and Disadvantages of DBMS, Users of DBMS - Database Designers, Application programmers, Sophisticated Users, End Users, Capabilities of good DBMS, Overall System structure.

UNIT II

Data Models: Introduction, Data Models, Object Based Logical Model, Record Base Logical Model - a. Relational Model, b.Network Model, c. Hierarchical Model, Entity Relationship Model, Entity Set, Attribute, Relationship Set, Entity Relationship Diagram (ERD), Extended features of ERD.

UNIT III

Relational Databases: Introduction, Terms - a. Relation, b.Tuple, c. Attribute, d. Cardinality, e. Degree, f. Domain, Keys -Super Key, Candidate Key, Primary Key, Foreign Key -Relational Algebra, Operations - a. Select, b. Project, c. Union,d. Difference, e. Intersection, f. Cartesian Product, g. Natural Join.

UNIT IV

Relational Database Design: Introduction, Anomalies of normalized database, Normalization - Normal Form, 1NF, 2NF, 3 NF. Transactions and Concurrency Control: ACID Property, Serializability, time stamp, locking protocol, granularity.

Tree Structured Indexing: Introduction, Index Sequential Access Method (ISAM), Structure of index sequential File, B+ Tree: A Dynamic Index Structure, Operations on B+ Tree - a.Search, b. Insertion, c. Deletion.

UNIT V

SQL (Structured Query Language): Introduction, History of SQL, Basic Structure, DDL, DML Commands, Simple Queries, Nested Queries, Aggregate Functions, Clauses.

UNIT 1
Database Management System

What is Information?

Information is in the form of news, Knowledge. Information is a collection of facts that are used as information. The group of information is creating a useful report for giving a better decision for future. The information contains any organizational or any business fact information, it a way to express a knowledge towards that fact. Information collected from various ways like newspaper, internet, television, books etc. Information not decides a proof about that fact-that only a brief writing report about that fact. Data is collected in information and used for decision making.

What is Data?

A collection of facts is called as data. Data is in the form of not meaningful information. It is in the form of statistical report or diagrammatic report or any graphical report. Data is not a confidential report for giving better decision about organisation. Data contains numbers, statements or report. Data also in the form of table. In analysis data plays important role for better decision. Data is a not a final output.

What is Database?

Database is a collection of meaningful data that was used for specific purpose. Database support to digital storage. The meaningful data can be stored in the form of tables. Database is used for giving a decision from records. Database in the form of tables and all records in the can be stored in row and columns mainly.

Ex.-A telephone directory is a best example of database. In telephone directory all phone number represented with users name in table format. That directory used for giving a better decision to user.

Database mainly manages, deleted or manipulated by users. Database is proof information about that organization. Database mainly in the form tables and they are used for storing information from user.

Different types of database are used like-

1. Distributed databases:

In that database centralized database was divided in to two or more database. Single database file sharing to many computers.

2. Relational databases:

In that database system all database files are connected and related with each other and they share data with each other.

3. Centralized database:

The centralized database share to other all computer system.

4. Open-source databases:

That database file open to all company employees for access.

5. Graph databases:

In that database file all data are in the form of graphical.

What is DBMS-Database Management System?

DBMS stores data in to the database but in manageable format. DBMS uses managing software for database handle in easier manner. DBMS means managing database easily with different operation. DBMS used for data retrieve to user with user requirement. In DBMS data collected from user and that all data save in database and if data is save successfully then that search, update and delete operations perform on that database with use of DBMS system, without DBMS database handling gets harder for data saving, retrieving and deleting from database.

DBMS consist lots of queries in build for data operations. Views, table, schema, report lots of things are used in DBMS for managing data. In DBMS**, DDL-Data Definition Language, DML-Data Manipulation Language, DCL-Data Control Language and TCL-Transaction Control Language languages** are used for data saving, searching and deleting from database successively.

Need of DBMS-

In previous ways data can be collected and stored with paper work, which creates lot of confusion at the time of data retrieve and all process get time consuming. In traditional way records' saving process was difficult because of lots of paper work was there which was not handled properly.

In latest technology computer use is the main focus of any work. Computer manages record without paper work and digital format, so records handling gets easier in latest ways. All data about any organization is handled in database format or table format, which include day to day transactions. This entire database is managed with latest technology called DBMS. DBMS manages all databases with ease using different languages.

Features/Characteristics of DBMS –

1. In DBMS command languages are used that allow user to create, alter and delete database easily.
2. In DBMS, user can create a data dictionary in easy way and documentation about that database in simple way.
3. DBMS support to view functionality that reason the data display with different view and different formation with user requirement.
4. DBMS supports data manipulation commands then data handling get easy to way.
5. DBMS manages lots of data in single formation and at a time, thousands of record handled in DBMS at a time in smaller time duration.
6. DBMS supports Rollback property means confidentiality about any transaction is done successively if any interrupt occur.
7. DBMS manages distributed database and complicated database structure well in manner.
8. DBMS manages table connectivity with different keys formations like primary key, foreign key etc.
9. DBMS support different types of database file like SQL, MS-access database file etc.
10. DBMS also checks users' authority at the time of database handling, DBMS gives a proper security to data.
11. DBMS allow to user data back up and data recovery.

Advantages of DBMS-

- Control data redundancy means that DBMS cannot allow inserting record as a duplicate.
- Sharing data get easier.
- Maintain and manage data easily.
- Data insertion, updation and searching time required less.
- Multiple user access databases at a time.
- Data backup and recovery featured is advanced in DBMS.

Dis-Advantages of DBMS-

- DBMS software requires advanced hardware and software that cost is high.
- Size of database is day to day increased then storage problem arises.
- Need to maintain software.
- Employees require knowledge to handle DBMS software.

Popular DBMS Software-

MySQL
Microsoft Access
Oracle
SQLite
Microsoft SQL Server etc.

Functions of DBMS-
DBMS contains important functions for maintain DBMS integrity and consistency. The functions of DBMS are as fallows;

1. Multi user accessing control
2. DBMS supports data backup and recovery
3. Data integrity management
4. Data presentation and transformation
5. DBMS interface
6. Data communication advantage
7. Data storage management
8. Data dictionary system
9. DBMS security management

1. Multi user accessing control –
DBMS supports multi user accessing control. In database management system data get stored in database in that database contains one or or more database files exist. In DBMS all database files access multi user at a time

2. DBMS supports data backup and recovery-
In DBMS lots of important data saved in database that reason database require data backup and data recovery functionality so in DBMS data backup and recovery gives data security for storage

3. Data integrity management-
DBMS supports Data integrity and data redundancy problem. In DBMS data can be saved only once the data cannot be e saved repeatedly in the database.

4. Data presentation and transformation-
In DBMS data presentation and data transformation get easier.DBMS supports “view” functionality to data for presenting and show to end user.

5. DBMS interface-
DBMS supports user friendly interface to user. User can access database management system easily because of their interface.

6. Data communication advantage-
DBMS supports multiple communication methods for accessing database with use of network and internet. You can access database with use of DBMS software online at every any place and time.

7. Data storage management-
Database can work on storage well in manner at the time of database handling in DBMS system. Data can save with recovery and backup plan.

8. Data dictionary system-
In DBMS data can saved in database with data dictionary method. Data dictionary method used data relational component with table connection for fast accessing data.

9. DBMS security management-
DBMS gives a proper security to data at the time of accessing data.DBMS authenticate user firstly at the time of data accessing. Identification and authentication is strongly bound with DBMS for data accessing.

File Processing System (FPS)-

In earlier days, data was stored manually, using pen and paper but after computer was discovered, the same task could be done by using files. A computer File is a resource which uniquely records data, in a storage device in a computer. There are various formats in which data can be stored. E.g. Text files can be stored in .txt format while pictures can be stored in .png format etc.

In case of computer files, data about data (metadata) can be stored in different lines, separated by spaces, commas or tab to resemble tables. Each file is placed in relevant folders for the ease of access.

In Computer Science, File Processing System (FPS) is a way of storing, retrieving and manipulating data which is present in various files.

Files are used to store various documents. All files are grouped based on their categories. The file names are much related to each other and arranged properly to easily access the files. In file processing system, if one needs to insert, delete, modify, store or update data, one must know the entire hierarchy of the files.

Advantages of File Processing System:

- Cost friendly –

There is a very minimal to no set up and usage fee for File Processing System. (In most cases, free tools are inbuilt in computers.)

- Easy to use –

File systems require very basic learning and understanding, hence, can be easily used.

- High scalability –

One can very easily switch from smaller to larger files as per his needs.

Disadvantages of File Processing System:

- Slow access time –

Direct access of files is very difficult and one needs to know the entire hierarchy of folders to get to a specific file. This involves a lot of time.

- Presence of redundant data –

The same data can be present in two or more files which take up more disc space.

- Inconsistent Data –

Due to data redundancy, same data stored at different places might not match to each other.

- Data Integrity Problems –

The data present in the database should be consistent and correct. To achieve this, the data should must satisfy certain constraints.

- Difficulty in recovery of corrupt data –

Recovery or backup of lost and corrupt data is nearly impossible in case of File Processing System.

- Lack of Atomicity –

Operations performed in the database must be atomic i.e. either the operation takes place as a whole or does not take place at all.

- Problem in Concurrent Access –

When a number of users operate on a common data in database at the same time then anomalies arise, due to lack of concurrency control.

- Unauthorized Access –

Anyone who gets access to the file can read or modify the data.

DBMS System-

DBMS is a collection of meaningful data and that all data stored in tabular format in well manner. The data insertion, updation and deletion get easy as compare file system. DBMS give a different and formation to user.DBMS gives a data recovery and data backup facility.DBMS gives a security to data.DBMS take care of data redundancy.

Comparison between File System and DBMS System

S.No.	File System	DBMS System
1	File system contains meaningful and unused data.	DBMS contains a meaningful data.
2	File system shows data information file storage size to user.	DBMS not shows data information and storage information to user.
3	file system not support backup and recovery mechanism	DBMS supports backup and recovery mechanism.
4	File system not a proper security to data.	DBMS gives a good security to data.
5	In file system data saving and retrieve process is hard.	In DBMS data saving and retrieving process is easy and user friendly.
6	File system not caring about data redundancy.	DBMS always take of data redundancy.

Different Types of Database Users-

Data base user type depends upon who interact with DBMS system and why. The different types of database is as fallows-

1. Database Administrator (DBA)-

DBA is totally responsible for handling schema definition from database. DBA handle the 3 levels of DBMS system mainly. DBA is totally responsible for handling security of database system. DBA can work on backup a database and recovery of database if fails.

2. End User-

End user is a main part of DBMS system because all DBMS system can use that user. That user don't have any knowledge about database, He can only interact with DBMS as a handler of that software mainly. Example- Clerk of bank is an end user, Ticket booking employee is a end user.

3. System Analysis-

That is part of company employee. System analysis is a bridge between user and DBA team. If any problems occur to user then system analysis can meet to end user and understand that problem and discuss that problem to DBA team for solution.

4. Database Designer-

That user is totally responsible for design a database for DBMS system. They can create a back-end for software mainly.

5. Application Programmer-

That user work for create a front-end for DBMS system mainly. In that phase designing task and coding task completed for DBMS system. The application programmer can join front-end to back-end.

BASIC PRACTICE QUESTIONS -

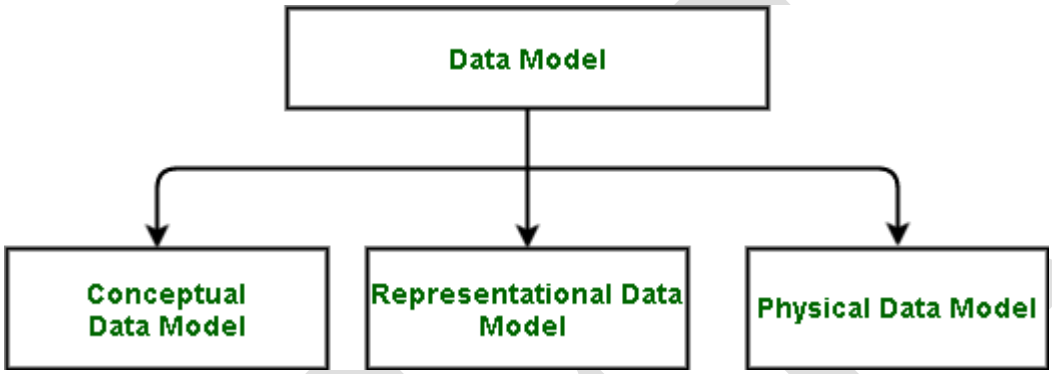
Q1 - What are the primary advantages of using a Database Management System (DBMS) over traditional file processing systems? Discuss at least three advantages and provide examples to illustrate each.
Q2 - Outline three key features of a modern Database Management System (DBMS) and explain how each contributes to improved data management and accessibility.
Q3 - Despite its numerous advantages, Database Management Systems (DBMS) also have some disadvantages. Identify and discuss three potential disadvantages of using a DBMS, and propose strategies to mitigate each drawback.
Q4 – Discuss the comparison between File System and DBMS System.
Q5 - Explain the various types of database users and how they interact with a Database Management System (DBMS).
Q6 – Describe in detail, what are the different types of database?

UNIT 2
Data Models

A **Data Model** in Database Management System (DBMS) is the concept of tools that are developed to summarize the description of the database. Data Models provide us with a transparent picture of data which helps us in creating an actual database. It shows us from the design of the data to its proper implementation of data.

Types of Relational Models

1. Conceptual Data Model
2. Representational Data Model
3. Physical Data Model



1. Conceptual Data Model

The conceptual data model describes the database at a very high level and is useful to understand the needs or requirements of the database. It is this model that is used in the requirement-gathering process i.e. before the Database Designers start making a particular database. One such popular model is the **entity/relationship model (ER model)**. The E/R model specializes in entities, relationships, and even attributes that are used by database designers.

Entity-Relationship Model (ER Model):

It is a high-level data model which is used to define the data and the relationships between them. It is basically a conceptual design of any database which is easy to design the view of data.

Components of ER Model:

Entity: An entity is referred to as a real-world object. It can be a name, place, object, class, etc. These are represented by a rectangle in an ER Diagram.

Attributes: An attribute can be defined as the description of the entity. These are represented by Eclipse in an ER Diagram. It can be Age, Roll Number, or Marks for a Student.

Relationship: Relationships are used to define relations among different entities. Diamonds and Rhombus are used to show Relationships.

Characteristics of a conceptual data model

- Offers Organization-wide coverage of the business concepts.
- This type of Data Models are designed and developed for a business audience.
- The conceptual model is developed independently of hardware specifications like data storage capacity, location or software specifications like DBMS vendor and technology. The focus is to represent data as a user will see it in the "real world."
- Conceptual data models known as Domain models create a common vocabulary for all stakeholders by establishing basic concepts and scope.

2. Representational Data Model

This type of data model is used to represent only the logical part of the database and does not represent the physical structure of the database. The representational data model allows us to focus primarily, on the design part of the database. A popular representational model is a Relational model. The relational Model consists of Relational Algebra and Relational Calculus. In the Relational Model, we basically use tables to represent our data and the relationships between them. It is a theoretical concept whose practical implementation is done in Physical Data Model.

The advantage of using a Representational data model is to provide a foundation to form the base for the Physical model.

3. Physical Data Model

The physical Data Model is used to practically implement Relational Data Model. Ultimately, all data in a database is stored physically on a secondary storage device such as discs and tapes. This is stored in the form of files, records, and certain other data structures. It has all the information on the format in which the files are present and the structure of the databases, the presence of external data structures, and their relation to each other. Here, we basically save tables in memory so they can be accessed efficiently. In order to come up with a good physical model, we have to work on the relational model in a better way. Structured Query Language (SQL) is used to practically implement Relational Algebra.

This Data Model describes HOW the system will be implemented using a specific DBMS system. This model is typically created by DBA and developers. The purpose is actual implementation of the database.

Characteristics of a physical data model:

- The physical data model describes data need for a single project or application though it may be integrated with other physical data models based on project scope.
- Data Model contains relationships between tables that which addresses cardinality and null ability of the relationships.
- Developed for a specific version of a DBMS, location, data storage or technology to be used in the project.
- Columns should have exact datatypes, lengths assigned and default values.
- Primary and Foreign keys, views, indexes, access profiles, and authorizations, etc. are defined.

Some Other Data Models

1. Hierarchical Model

The hierarchical Model is one of the oldest models in the data model which was developed by IBM, in the 1950s. In a hierarchical model, data are viewed as a collection of tables, or we can say segments that form a hierarchical relation. In this, the data is organized into a tree-like structure where each record consists of one parent record and many children. Even if the segments are connected as a chain-like structure by logical associations, then the instant structure can be a fan structure with multiple branches. We call the illogical associations as directional associations.

2. Network Model

The Network Model was formalized by the Database Task group in the 1960s. This model is the generalization of the hierarchical model. This model can consist of multiple parent segments and these segments are grouped as levels but there exists a logical association between the segments belonging to any level. Mostly, there exists a many-to-many logical association between any of the two segments.

3. Object-Oriented Data Model

In the Object-Oriented Data Model, data and their relationships are contained in a single structure which is referred to as an object in this data model. In this, real-world problems are represented as objects with different attributes. All objects have multiple relationships between them. Basically, it is a combination of Object Oriented programming and a Relational Database Model.

4. Float Data Model

The float data model basically consists of a two-dimensional array of data models that do not contain any duplicate elements in the array. This data model has one drawback it cannot store a large amount of data that is the tables cannot be of large size.

5. Context Data Model

The Context data model is simply a data model which consists of more than one data model. For example, the Context data model consists of ER Model, Object-Oriented Data Model, etc. This model allows users to do more than one thing which each individual data model can do.

6. Semi-Structured Data Model

Semi-Structured data models deal with the data in a flexible way. Some entities may have extra attributes and some entities may have some missing attributes. Basically, you can represent data here in a flexible way.

Advantages of Data Models

1. Data Models help us in representing data accurately.
2. It helps us in finding the missing data and also in minimizing Data Redundancy.
3. Data Model provides data security in a better way.
4. The data model should be detailed enough to be used for building the physical database.
5. The information in the data model can be used for defining the relationship between tables, primary and foreign keys, and stored procedures.

Disadvantages of Data Models

1. In the case of a vast database, sometimes it becomes difficult to understand the data model.
2. You must have the proper knowledge of SQL to use physical models.
3. Even smaller change made in structure requires modification in the entire application.
4. There is no set data manipulation language in DBMS.
5. To develop Data model one should know physical data stored characteristics.

Features of ER model

The basic E-R concepts can model most database features, some aspects of a database may be more aptly expressed by certain extensions to the basic E-R model. **The extended E-R features are specialization, generalization, higher- and lower-level entity sets, attribute inheritance, and aggregation.**

Specialization – An entity set broken down sub-entities that are distinct in some way from other entities in the set. For instance, a subset of entities within an entity set may have attributes that are not shared by all the entities in the entity set. The E-R model provides a means for representing these distinctive entity groupings.

Specialization is an "aTop-down approach" where a high-level entity is specialized into two or more level entities.

Example – Consider an entity set vehicle, with attributes color and no. of tires. A vehicle may be further classified as one of the following:

Car, Bike, Bus

Each of these vehicle types is described by a set of attributes that includes all the attributes of the entity set vehicle plus possibly additional attributes. For example, car entities may be described further by the attribute gear, whereas bike entities may be described further by the attributes automatic break. The process of designating subgroupings within an entity set is called **specialization.** The specialization of vehicles allows us to distinguish among vehicles according to whether they are cars, buses, or bikes.

Generalization – It is a process of extracting common properties from a set of entities and creating a generalized entity from it. Generalization is a "Bottom-up approach". In which two or more entities can be combined to form a higher-level entity if they have some attributes in common.

In **generalization,** Subclasses are combined to make a superclass.

Example: There are three entities given, car, bus, and bike. They all have some common attributes like all cars, buses, and bikes they all have no. of tires and have some colors. So they all can be grouped and make a superclass named a vehicle.

Inheritance – An entity that is a member of a subclass inherits all the attributes of the entity as the member of the superclass, the entity also inherits all the relationships that the superclass participates in. **Inheritance** is an important feature of Generalization and Specialization. It allows lower-level entities to inherit the attributes of higher-level entities.

Example – Car, bikes, and buses inherit the attributes of a vehicle. Thus, a car is described by its color and no. of tires, and additionally a gear attribute; a bike is described by its color and no. of tires attributes, and additionally automatic break attribute.

Aggregation – In aggregation, the relation between two entities is treated as a single entity. In aggregation, the relationship with its corresponding entities is aggregated into a higher-level entity.

Example- phone numbers on your mobile phone. You can refer to them individually – your mother's number, your best friend's number, etc. But it's easier to think of them collectively, as your phone number list. It is also important to realize that each member of the aggregation still has the properties of the whole. In other words, each phone number in the list remains a phone number. The process of combining them has not altered them in any way.

Conclusion

- Data modeling is the process of developing data model for the data to be stored in a Database.
- Data Models ensure consistency in naming conventions, default values, semantics, and security while ensuring quality of the data.
- Data Model structure helps to define the relational tables, primary and foreign keys and stored procedures.
- There are three types of conceptual, logical, and physical.
- The main aim of conceptual model is to establish the entities, their attributes, and their relationships.
- Logical data model defines the structure of the data elements and set the relationships between them.
- A Physical Data Model describes the database specific implementation of the data model.
- The main goal of a designing data model is to make certain that data objects offered by the functional team are represented accurately.
- The biggest drawback is that even smaller change made in structure requires modification in the entire application.

BASIC PRACTICE QUESTIONS –

Q1 - Compare and contrast the hierarchical, network, relational and object-oriented data models.

Q2 - Explain the concept of entity-relationship (ER) modeling. What are entities, attributes, and relationships in an ER diagram? Provide an example of an ER diagram and describe how it represents the structure of a database.

Q3 - Discuss in detail the types of data models: conceptual, representational, and physical data models, including their definitions, characteristics, and how they relate to each other in database design.

Q4 – Write down the features of ER Model in detail.

Q5 – Write down 5 advantages and 5 disadvantages of data models.

Q6 - Write down the difference between Generalization and Specialization.

UNIT 3
Relational Databases

What is RDBMS (Relational Database Management System)?

RDBMS stands for Relational Database Management System.

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 uses SQL queries to access the data in the database.

What is a Database Table?

A table is a collection of related data entries, and it consists of columns and rows.

A column holds specific information about every record in the table.

A record (or row) is each individual entry that exists in a table.

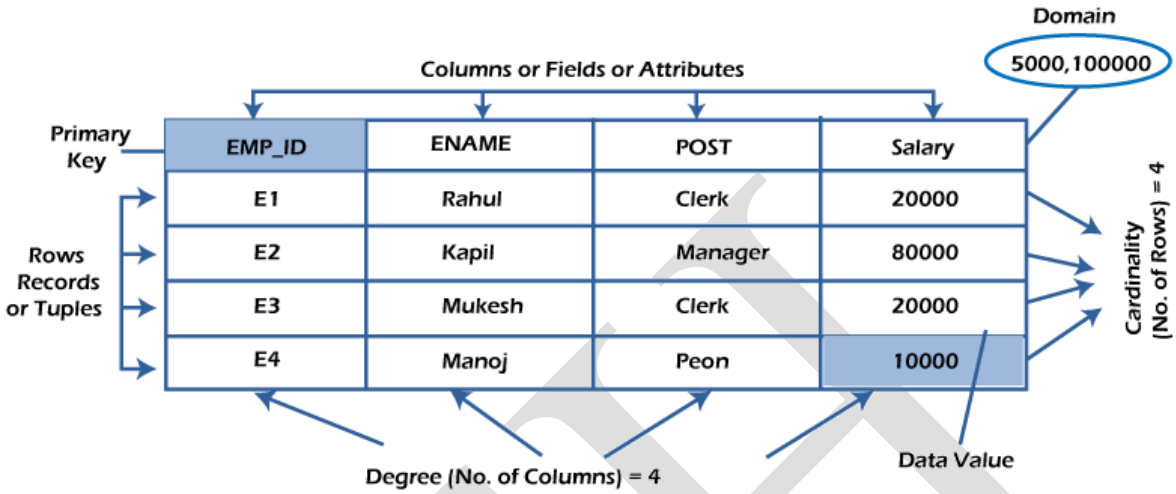
Customer ID	Customer Name	Address	City	Postal Code	Country
1	Anthony	Obere Str. 57	Berlin	12209	Germany
2	Antio	Avda. de la Cón 2222	México D.F.	05021	Mexico
3	Beljin	Matas 2312	México D.F.	05023	Mexico
4	Samy	120 Hanover	London	78965	UK

The columns in the "Customers" table above are: CustomerID, CustomerName, Address, City, PostalCode and Country. The table has 4 records (rows).

What is a Relational Database?

A relational database defines database relationships in the form of tables. The tables are related to each other - based on data common to each.

Following are the various terminologies of RDBMS:



What is table/Relation?

Everything in a relational database is stored in the form of relations. The RDBMS database uses tables to store data. **A table is a collection of related data entries and contains rows and columns to store data.** Each table represents some real-world objects such as person, place, or event about which information is collected.

Properties of a Relation:

- Each relation has a unique name by which it is identified in the database.
- Relation does not contain duplicate tuples.
- The tuples of a relation have no specific order.
- All attributes in a relation are atomic, i.e., each cell of a relation contains exactly one value.

A table is the simplest example of data stored in RDBMS.

What is a row or record?

A row of a table is also called a record or tuple. It contains the specific information of each entry in the table. It is a horizontal entity in the table. For example, the above table contains 4 records.

Properties of a row:

- No two tuples are identical to each other in all their entries.
- All tuples of the relation have the same format and the same number of entries.
- The order of the tuple is irrelevant. They are identified by their content, not by their position.

What is a column/attribute?

A column is a vertical entity in the table which contains all information associated with a specific field in a table. For example, "ENAME" is a column in the above table which contains all information about an employee's name.

Properties of an Attribute:

- Every attribute of a relation must have a name.
- Null values are permitted for the attributes.
- Default values can be specified for an attribute automatically inserted if no other value is specified for an attribute.
- Attributes that uniquely identify each tuple of a relation are the primary key.

What is data item/Cells?

The smallest unit of data in the table is the individual data item. It is stored at the intersection of tuples and attributes.

Properties of data items:

- Data items are atomic.
- The data items for an attribute should be drawn from the same domain.

Degree:

The total number of attributes that comprise a relation is known as the degree of the table.

For example, the employee table has 4 attributes, and its degree is 4.

Cardinality:

The total number of tuples at any one time in a relation is known as the table's cardinality. **The relation whose cardinality is 0 is called an empty table.**

For example, the employee table has 4 rows, and its cardinality is 4.

Domain:

The domain refers to the possible values each attribute can contain. It can be specified using standard data types such as integers, floating numbers, etc. For example, an attribute entitled Marital_Status may be limited to married or unmarried values.

NULL Values

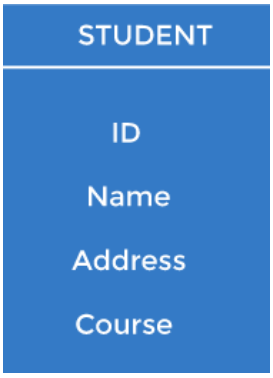
The NULL value of the table specifies that the field has been left blank during record creation. It is different from the value filled with zero or a field that contains space.

Keys

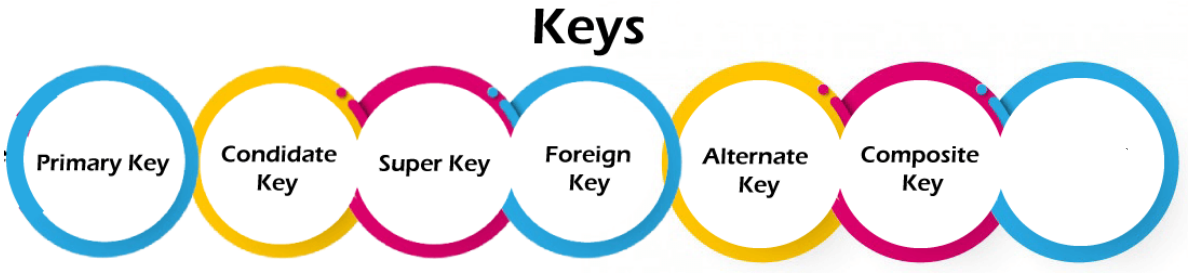
Keys play an important role in the relational database.

It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.

For example, ID is used as a key in the Student table because it is unique for each student.

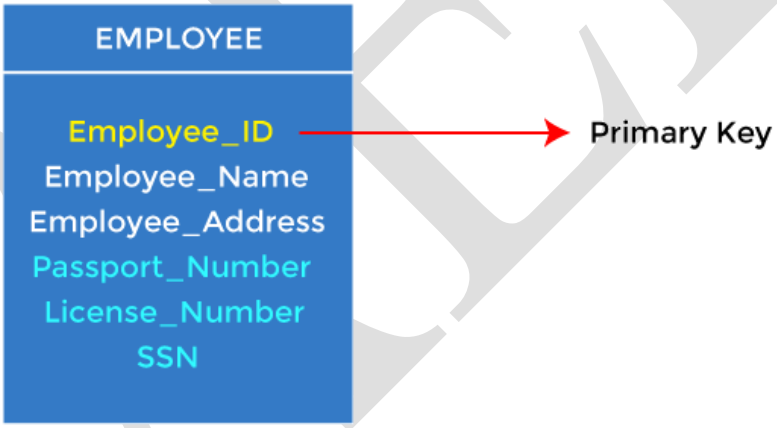


Types of keys:



1. Primary key

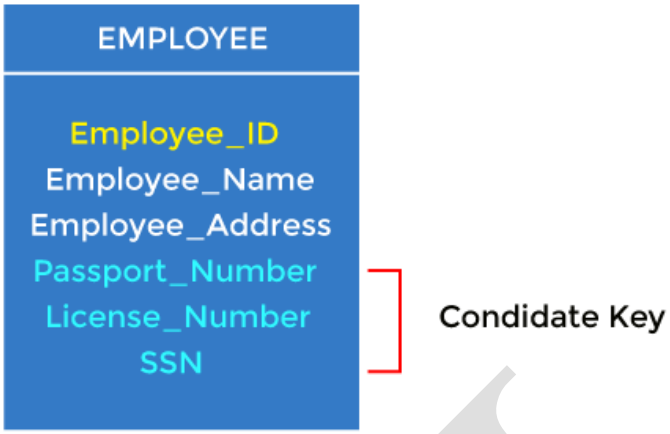
- It is the first key used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys, as we saw in the student table. The key which is most suitable from those lists becomes a primary key.
- In the EMPLOYEE table below, ID can be the primary key since it is unique for each employee. In the EMPLOYEE table, we can even select License_Number and Passport_Number as primary keys since they are also unique.
- For each entity, the primary key selection is based on requirements and developers.



2. Candidate key

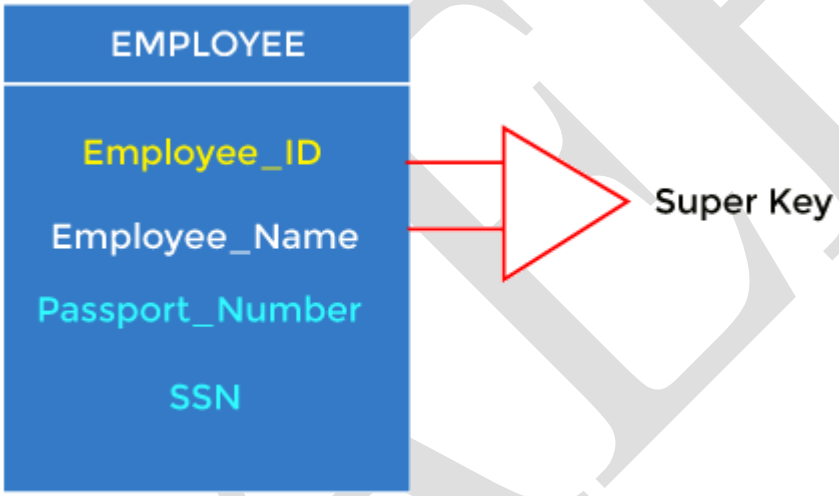
- A candidate key is an attribute or set of attributes that can uniquely identify a tuple.
- Except for the primary key, the remaining attributes are considered a candidate key. The candidate keys are as strong as the primary key.

For example: In the EMPLOYEE table, id is best suited for the primary key. The rest of the attributes, like SSN, Passport_Number, License_Number, etc., are considered a candidate key.



3. Super Key

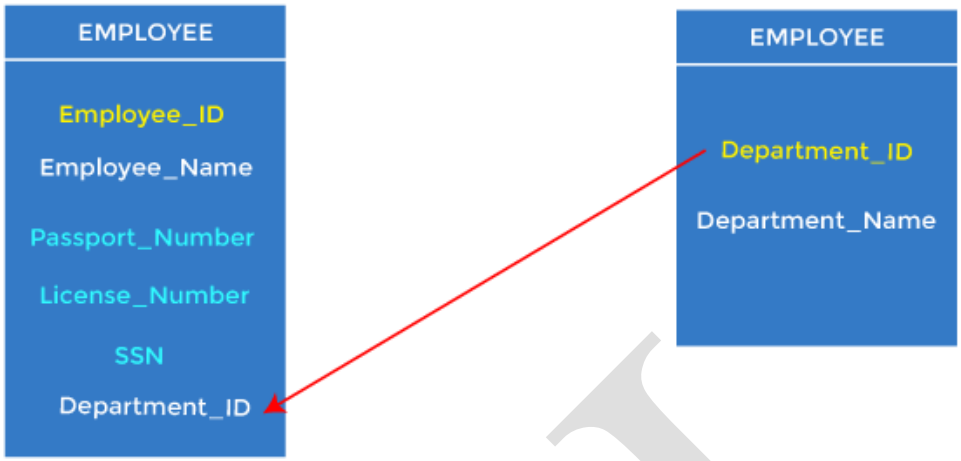
Super key is an attribute set that can uniquely identify a tuple. A super key is a superset of a candidate key.



For example: In the above EMPLOYEE table, for (EMPLOEE_ID, EMPLOYEE_NAME), the name of two employees can be the same, but their EMPLYEE_ID can't be the same. Hence, this combination can also be a key.

4. Foreign key

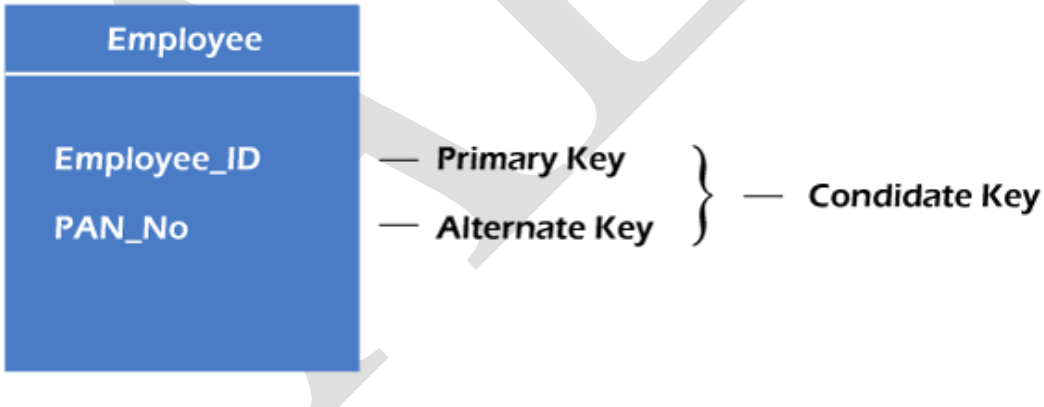
- Foreign keys are the column of the table used to point to the primary key of another table.
- Every employee works in a specific department in a company, and employee and department are two different entities. So we can't store the department's information in the employee table. That's why we link these two tables through the primary key of one table.
- We add the primary key of the DEPARTMENT table, Department_Id, as a new attribute in the EMPLOYEE table.
- In the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related.



5. Alternate key

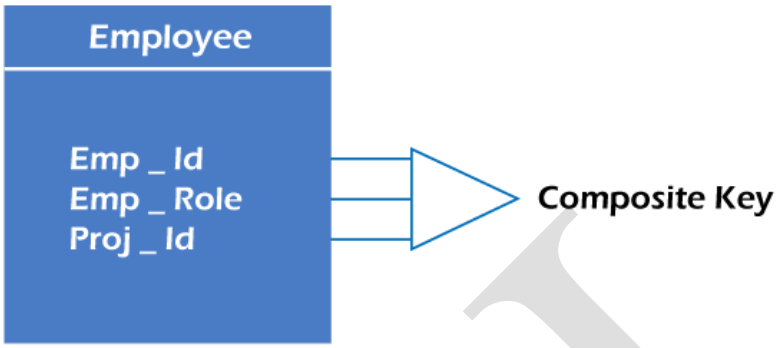
There may be one or more attributes or a combination of attributes that uniquely identify each tuple in a relation. These attributes or combinations of the attributes are called the candidate keys. One key is chosen as the primary key from these candidate keys, and the remaining candidate key, if it exists, is termed the alternate key. **In other words,** the total number of the alternate keys is the total number of candidate keys minus the primary key. The alternate key may or may not exist. If there is only one candidate key in a relation, it does not have an alternate key.

For example, employee relation has two attributes, Employee_Id and PAN_No that act as candidate keys. In this relation, Employee_Id is chosen as the primary key, so the other candidate key, PAN_No, acts as the Alternate key.



6. Composite key

Whenever a primary key consists of more than one attribute, it is known as a composite key. This key is also known as Concatenated Key.



For example, in employee relations, we assume that an employee may be assigned multiple roles, and an employee may work on multiple projects simultaneously. So the primary key will be composed of all three attributes, namely Emp_ID, Emp_role, and Proj_ID in combination. So these attributes act as a composite key since the primary key comprises more than one attribute.

Relational Algebra

Relational algebra in DBMS is a procedural query language. Queries in relational algebra are performed using operators. Relational Algebra is the fundamental block for modern language SQL and modern Database Management Systems such as Oracle Database, Microsoft SQL Server, IBM Db2, etc.

What is Relational Algebra in DBMS?

Relational Algebra came in 1970 and was given by Edgar F. Codd (Father of DBMS). It is also known as Procedural Query Language (PQL) as in PQL; a programmer/user has to mention two things, **"What to Do"** and **"How to Do"**.

Suppose our data is stored in a database, then relational algebra is used to access the data from the database.

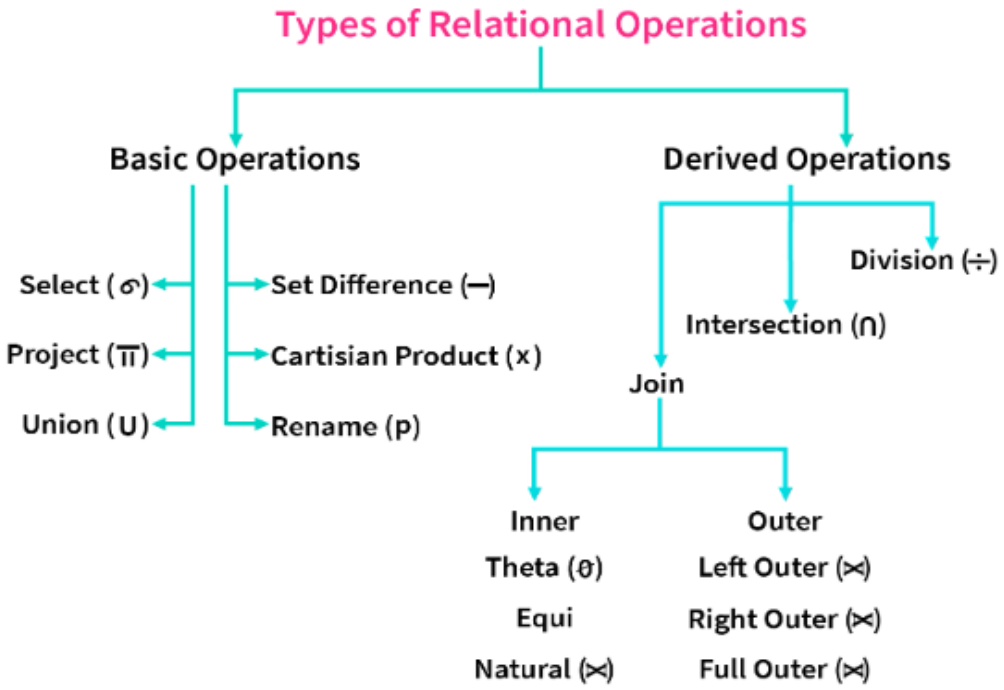
The First thing is we have to access the data, this needs to be specified in the query as **"What to Do"**, but we have to also specify the method/procedure in the query that is **"How to Do"** or how to access the data from the database.

Types of Relational Operations in DBMS

In Relational Algebra, we have two types of Operations.

- **Basic Operations**
- **Derived Operations**

Applying these operations over relations/tables will give us new relations as output.



Basic Operations

Six fundamental operations are mentioned below. The majority of data retrieval operations are carried out by these. Let's know them one by one.

But, before moving into detail, let's have two tables or we can say relations STUDENT(ROLL, NAME, AGE) and EMPLOYEE(EMPLOYEE_NO, NAME, AGE) which will be used in the below examples.

STUDENT

ROLL	NAME	AGE
1	Aman	20
2	Atul	18
3	Baljeet	19
4	Harsh	20
5	Prateek	21
6	Prateek	23

EMPLOYEE

EMPLOYEE_NO	NAME	AGE
E-1	Anant	20
E-2	Ashish	23
E-3	Baljeet	25
E-4	Harsh	20
E-5	Pranav	22

Select (σ)

Select operation is done by **Selection Operator which is represented by "sigma"(σ).** It is used to retrieve tuples (rows) from the table where the given condition is satisfied. It is a unary operator means it requires only one operand.

Notation: σ p(R)

Where σ is used to represent SELECTION

R is used to represent RELATION

p is the logic formula

Let's understand this with an example:

Suppose we want the row(s) from STUDENT Relation where "AGE" is 20:

σ AGE=20 (STUDENT)

ROLL	NAME	AGE
1	Aman	20
4	Harsh	20

Project (∏)

Project operation is done by **Projection Operator which is represented by "pi"(∏).** It is used to retrieve certain attributes (columns) from the table. It is also known as vertical partitioning as it separates the table vertically. It is also a unary operator.

Notation: ∏ a(r)

Where ∏ is used to represent PROJECTION

r is used to represent RELATION

a is the attribute list

Let's understand this with an example:

Suppose we want the names of all students from STUDENT Relation.

∏ NAME (STUDENT)

NAME
Aman
Atul
Baljeet
Harsh
Prateek

As you can see from the above output it eliminates duplicates.

For multiple attributes, we can separate them using a ",".

∏ ROLL, NAME (STUDENT)

Above code will return two columns, ROLL and NAME.

ROLL	NAME
1	Aman
2	Atul
3	Baljeet
4	Harsh
5	Prateek
6	Prateek

Union (∪)

Union operation is done by **Union Operator which is represented by "union"(∪)**. It is the same as the union operator from set theory, i.e., it selects all tuples from both relations but with the exception that for the union of two relations/tables both relations must have the same set of Attributes. It is a binary operator as it requires two operands.

Notation: R ∪ S

Where R is the first relation

S is the second relation

If relations don't have the same set of attributes, then the union of such relations will result in NULL.

Let's have an example to clarify the concept:

Suppose we want all the names from STUDENT and EMPLOYEE relation.

∏ NAME (STUDENT) ∪ ∏ NAME (EMPLOYEE)

NAME
Aman
Anant
Ashish
Atul
Baljeet
Harsh
Pranav
Prateek

As we can see from the above output it also eliminates duplicates.

Set Difference (-)

Set Difference as its name indicates is the difference between two relations (R-S). It is denoted by a "Hyphen"(-) and it returns all the tuples (rows) which are in relation R but not in relation S. It is also a binary operator.

Notation: R - S

Where R is the first relation

S is the second relation

Just like union, the set difference also comes with the exception of the same set of attributes in both relations.

Let's take an example where we would like to know the names of students who are in STUDENT Relation but not in EMPLOYEE Relation.

∏ NAME (STUDENT) - ∏ NAME (EMPLOYEE)

This will give us the following output:

NAME
Aman
Atul
Prateek

Cartesian product (X)

Cartesian product is denoted by the "X" symbol. Let's say we have two relations R and S. Cartesian product will combine every tuple (row) from R with all the tuples from S. I know it sounds complicated, but once we look at an example, you'll see what I mean.

Notation: R X S

Where R is the first relation

S is the second relation

As we can see from the notation it is also a binary operator.

Let's combine the two relations STUDENT and EMPLOYEE.

STUDENT X EMPLOYEE

ROLL	NAME	AGE	EMPLOYEE_NO	NAME	AGE
1	Aman	20	E-1	Anant	20
1	Aman	20	E-2	Ashish	23
1	Aman	20	E-3	Baljeet	25
1	Aman	20	E-4	Harsh	20
1	Aman	20	E-5	Pranav	22
2	Atul	18	E-1	Anant	20
2	Atul	18	E-2	Ashish	23
2	Atul	18	E-3	Baljeet	25
2	Atul	18	E-4	Harsh	20
2	Atul	18	E-5	Pranav	22

Derived Operations

Also known as extended operations, these operations can be derived from basic operations and hence named Derived Operations. These include three operations: Join Operations, Intersection operations, and Division operations.

Let's study them one by one.

Join Operations

Join Operation in DBMS are binary operations that allow us to combine two or more relations.

They are further classified into two types: **Inner Join, and Outer Join.**

First, let's have two relations EMPLOYEE consisting of E_NO, E_NAME, CITY and EXPERIENCE. EMPLOYEE table contains employee's information such as id, name, city, and experience of employee (In Years). The other relation is DEPARTMENT consisting of D_NO, D_NAME, E_NO and MIN_EXPERIENCE.

DEPARTMENT table defines the mapping of an employee to their department. It contains Department Number, Department Name, Employee Id of the employee working in that department, and the minimum experience required (In Years) to be in that department.

EMPLOYEE

E_NO	E_NAME	CITY	EXPERIENCE
E-1	Ram	Delhi	04
E-2	Varun	Chandigarh	09
E-3	Ravi	Noida	03
E-4	Amit	Bangalore	07

DEPARTMENT

D_NO	D_NAME	E_NO	MIN_EXPERIENCE
D-1	HR	E-1	03
D-2	IT	E-2	05
D-3	Marketing	E-3	02

Also, let's have the Cartesian Product of the above two relations. It will be much easier to understand Join Operations when we have the Cartesian Product.

E_NO	E_NAME	CITY	EXPERIENCE	D_NO	D_NAME	E_NO	MIN_EXPERIENCE
E-1	Ram	Delhi	04	D-1	HR	E-1	03
E-1	Ram	Delhi	04	D-2	IT	E-2	05
E-1	Ram	Delhi	04	D-3	Marketing	E-3	02
E-2	Varun	Chandigarh	09	D-1	HR	E-1	03
E-2	Varun	Chandigarh	09	D-2	IT	E-2	05
E-2	Varun	Chandigarh	09	D-3	Marketing	E-3	02
E-3	Ravi	Noida	03	D-1	HR	E-1	03
E-3	Ravi	Noida	03	D-2	IT	E-2	05
E-3	Ravi	Noida	03	D-3	Marketing	E-3	02

E-4	Amit	Banga lore	07	D-1	HR	E-1	03
E-4	Amit	Banga lore	07	D-2	IT	E-2	05
E-4	Amit	Banga lore	07	D-3	Marketin g	E-3	02

Natural Join (⋈)

A comparison operator is not used in a natural join. It does not concatenate like a Cartesian product. A Natural Join can be performed only if two relations share at least one common attribute. Furthermore, the attributes must share the same name and domain.

Natural join operates on matching attributes where the values of the attributes in both relations are the same and remove the duplicate ones.

Preferably Natural Join is performed on the foreign key.

Notation: R ⋈ S

Where R is the first relation

S is the second relation

Let's say we want to join EMPLOYEE and DEPARTMENT relation with E_NO as a common attribute.

Notice, here E_NO has the same name in both the relations and also consists of the same domain, i.e., in both relations E_NO is a string.

EMPLOYEE ⋈ DEPARTMENT

E_NO	E_NAME	CITY	EXPERIENCE	D_NO	D_NAME	MIN_EXPERIENCE
E-1	Ram	Delhi	04	D-1	HR	03
E-2	Varun	Chandigarh	09	D-2	IT	05
E-3	Ravi	Noida	03	D-3	Marketing	02

But unlike the above operation, where we have two columns of E_NO, here we are having only one column of E_NO. This is because **Natural Join automatically keeps a single copy of a common attribute.**

Intersection (∩)

Intersection operation is done by Intersection Operator which is represented by "intersection"(∩).It is the same as the intersection operator from set theory, i.e., it selects all the tuples which are present in both relations. It is a binary operator as it requires two operands. Also, it eliminates duplicates.

Notation: R ∩ S

Where R is the first relation

S is the second relation

Let's have an example to clarify the concept:

Suppose we want the names which are present in STUDENT as well as in EMPLOYEE relation, Relations we used in Basic Operations.

∏ NAME (STUDENT) ∩ ∏ NAME (EMPLOYEE)

NAME
Baljeet
Harsh

BASIC PRACTICE QUESTIONS –

Q1 - Define the following key terms in the context of Relational Database Management Systems (RDBMS): Relation, Tuple, Attribute, Primary key, Foreign key, and Alternate key. Explain how each term contributes to the structure and functionality of a relational database.

Q2 – Determine the degree and cardinality of the provided table.

ID	Name	Age	Gender
1	Alice	25	Female
2	Bob	30	Male
3	Charlie	28	Male
4	David	22	Male

Q3 - Define and explain the SELECT operation in Relational Algebra. Provide an example query using the SELECT operation to retrieve specific rows from a given relation. Discuss how the SELECT operation filters rows based on specified conditions.

Q4 -Discuss the concepts of UNION, INTERSECT, and DIFFERENCE operations in Relational Algebra. Explain how each operation works and provide examples to illustrate their usage in combining or comparing data from multiple relations.

Q5 - Describe the concept of JOIN operations in Relational Algebra. Provide examples of NATURAL JOIN operation and discuss their significance in combining data from multiple tables.

UNIT 4
Relational Database Design

Normalization

A large database defined as a single relation may result in data duplication. This repetition of data may result in:

- Making relations very large.
- It isn't easy to maintain and update data as it would involve searching many records in relation.
- Wastage and poor utilization of disk space and resources.
- The likelihood of errors and inconsistencies increases.

So to handle these problems, we should analyze and decompose the relations with redundant data into smaller, simpler, and well-structured relations that are satisfy desirable properties. Normalization is a process of decomposing the relations into relations with fewer attributes.

What is Normalization?

Normalization is the process of organizing the data in the database.

Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like **Insertion, Update, and Deletion Anomalies.**

Normalization divides the larger table into smaller and links them using relationships.

The normal form is used to reduce redundancy from the database table.

Why do we need Normalization?

The main reason for normalizing the relations is removing these anomalies. Failure to eliminate anomalies leads to data redundancy and can cause data integrity and other problems as the database grows. Normalization consists of a series of guidelines that helps to guide you in creating a good database structure.

Data modification anomalies can be categorized into three types:

1. **Insertion Anomaly:** Insertion Anomaly refers to when one cannot insert a new tuple into a relationship due to lack of data.

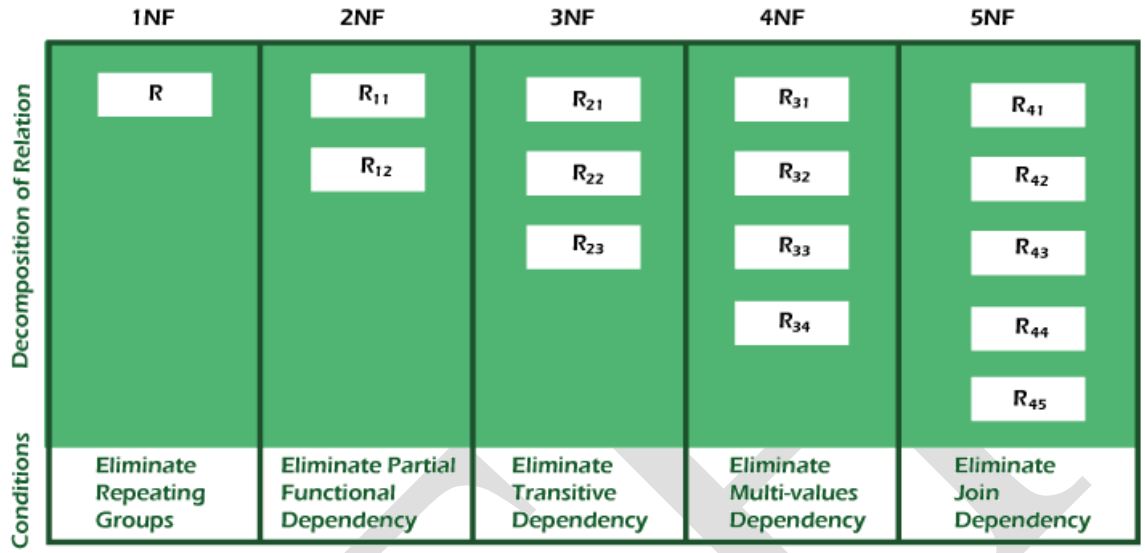
2. **Deletion Anomaly:** The delete anomaly refers to the situation where the deletion of data results in the unintended loss of some other important data.

3. **Updation Anomaly:** The update anomaly is when an update of a single data value requires multiple rows of data to be updated.

Types of Normal Forms:

Normalization works through a series of stages called Normal forms. The normal forms apply to individual relations. The relation is said to be in particular normal form if it satisfies constraints.

Following are the various types of Normal forms:



Normal Form	Description
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.
BCNF	A stronger definition of 3NF is known as Boyce Codd's normal form.

Advantages of Normalization

1. Normalization helps to minimize data redundancy.
2. Greater overall database organization.
3. Data consistency within the database.
4. Much more flexible database design.
5. Enforces the concept of relational integrity.

Disadvantages of Normalization

- You cannot start building the database before knowing what the user needs.
- The performance degrades when normalizing the relations to higher normal forms, i.e., 4NF, 5NF.
- It is very time-consuming and difficult to normalize relations of a higher degree.
- Careless decomposition may lead to a bad database design, leading to serious problems.

First Normal Form (1NF)

- A relation will be 1NF if it contains an atomic value.
- It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
- First normal form disallows the multi-valued attribute, composite attribute, and their combinations.

Example: Relation EMPLOYEE is not in 1NF because of multi-valued attribute EMP_PHONE.

EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385, 9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389, 8589830302	Punjab

The decomposition of the EMPLOYEE table into 1NF has been shown below:

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385	UP
14	John	9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389	Punjab
12	Sam	8589830302	Punjab

Second Normal Form (2NF)

- In the 2NF, relational must be in 1NF.
- In the second normal form, all non-key attributes are fully functional dependent on the primary key

Example: Let's assume, a school can store the data of teachers and the subjects they teach. In a school, a teacher can teach more than one subject.

TEACHER TABLE –

TEACHER_ID	SUBJECT	TEACHER_AGE
25	Chemistry	30
25	Biology	30
47	English	35
83	Math	38
83	Computer	38

In the given table, non-prime attribute TEACHER_AGE is dependent on TEACHER_ID which is a proper subset of a candidate key. That's why it violates the rule for 2NF.

To convert the given table into 2NF, we decompose it into two tables:

TEACHER_DETAIL table:

TEACHER_ID	TEACHER_AGE
25	30
47	35
83	38

TEACHER_SUBJECT table:

TEACHER_ID	SUBJECT
25	Chemistry
25	Biology
47	English
83	Math
83	Computer

Third Normal Form (3NF)

- A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.
- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

A relation is in third normal form if it holds atleast one of the following conditions for every non-trivial function dependency $X \rightarrow Y$.

1. X is a super key.
2. Y is a prime attribute, i.e., each element of Y is part of some candidate key.

Example:

EMPLOYEE_DETAIL table:

EMP_ID	EMP_NAME	EMP_ZIP	EMP_STATE	EMP_CITY
222	Harry	201010	UP	Noida
333	Stephan	02228	US	Boston
444	Lan	60007	US	Chicago
555	Katharine	06389	UK	Norwich
666	John	462007	MP	Bhopal

Super key in the table above:

{EMP_ID}, {EMP_ID, EMP_NAME}, {EMP_ID, EMP_NAME, EMP_ZIP}....so on

Candidate key: {EMP_ID}

Non-prime attributes: In the given table, all attributes except EMP_ID are non-prime.

Here, EMP_STATE & EMP_CITY dependent on EMP_ZIP and EMP_ZIP dependent on EMP_ID. The non-prime attributes (EMP_STATE, EMP_CITY) transitively dependent on super key (EMP_ID). It violates the rule of third normal form.

That's why we need to move the EMP_CITY and EMP_STATE to the new <EMPLOYEE_ZIP> table, with EMP_ZIP as a Primary key.

EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_ZIP
222	Harry	201010
333	Stephan	02228
444	Lan	60007
555	Katharine	06389
666	John	462007

EMPLOYEE_ZIP table:

EMP_ZIP	EMP_STATE	EMP_CITY
201010	UP	Noida
02228	US	Boston
60007	US	Chicago
06389	UK	Norwich
462007	MP	Bhopal

Transactions and Concurrency Control:

What is Transaction?

A **transaction** is a collection of operations that performs a single logical function in a database application. Each transaction is a unit of both atomicity and consistency. Thus, we require that transactions do not violate any database consistency constraints. That is, if the database was consistent when a transaction started, the database must be consistent when the transaction successfully terminates. However, during the execution of a transaction, it may be necessary temporarily to allow inconsistency, since either the debit of A or the credit of B must be done before the other. This temporary inconsistency, although necessary, may lead to difficulty if a failure occurs.

It is the programmer's responsibility to define properly the various transactions, so that each preserves the consistency of the database. For example, the transaction to transfer funds from the account of department A to the account of department B could be defined to be composed of two separate programs: one that debits account A, and another that credits account B. The execution of these two programs one after the other will indeed preserve consistency. However, each program by itself does not transform the database from a consistent state to a new consistent state. Thus, those programs are not transactions.

The concept of a transaction has been applied broadly in database systems and applications. While the initial use of transactions was in financial applications, the concept is now used in real-time applications in telecommunication, as well as in the management of long-duration activities such as product design or administrative workflows.

A set of logically related operations is known as a transaction. The main operations of a transaction are:

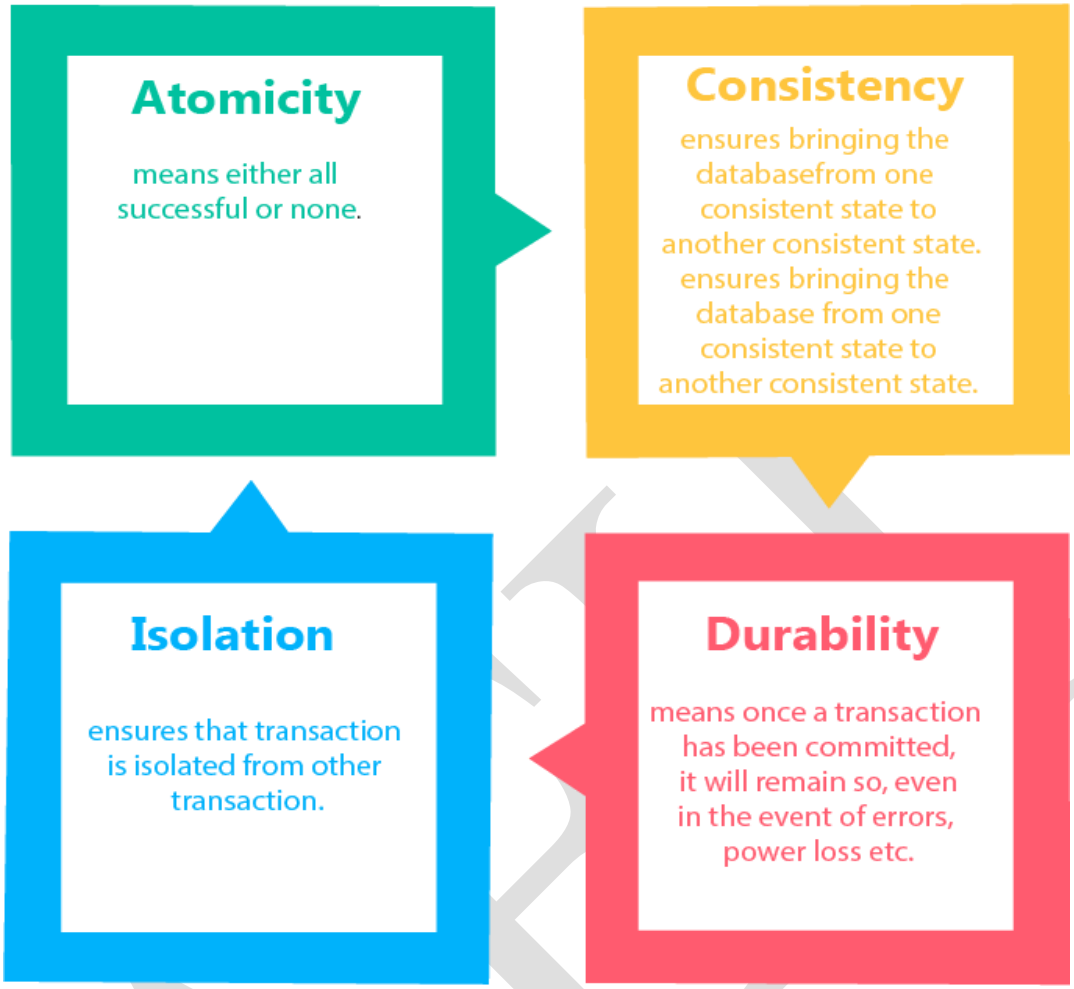
- **Read (A):** Read operations Read (A) or R (A) reads the value of A from the database and stores it in a buffer in the main memory.
- **Write (A):** Write operation Write (A) or W (A) writes the value back to the database from the buffer.

Transaction property

The transaction has the four properties. These are used to maintain consistency in a database, before and after the transaction.

Property of Transaction

1. **Atomicity**
2. **Consistency**
3. **Isolation**
4. **Durability**



Atomicity –

- It states that all operations of the transaction take place at once if not, the transaction is aborted.
- There is no midway, i.e., the transaction cannot occur partially. Each transaction is treated as one unit and either run to completion or is not executed at all.

  Atomicity involves the following two operations:

  Abort: If a transaction aborts then all the changes made are not visible.

  Commit: If a transaction commits then all the changes made are visible.

Example: Let's assume that following transaction T consisting of T1 and T2. A consists of Rs 600 and B consists of Rs 300. Transfer Rs 100 from account A to account B.

T1	T2
Read(A) A:= A-100 Write(A)	Read(B) Y:= Y+100 Write(B)

After completion of the transaction, A consists of Rs 500 and B consists of Rs 400.

If the transaction T fails after the completion of transaction T1 but before completion of transaction T2, then the amount will be deducted from A but not added to B. This shows the inconsistent database state. In order to ensure correctness of database state, the transaction must be executed in entirety.

Consistency

The integrity constraints are maintained so that the database is consistent before and after the transaction.

The execution of a transaction will leave a database in either its prior stable state or a new stable state.

The consistent property of database states that every transaction sees a consistent database instance.

The transaction is used to transform the database from one consistent state to another consistent state.

For example: The total amount must be maintained before or after the transaction.

Total before T occurs = 600+300=900

Total after T occurs= 500+400=900

Therefore, the database is consistent. In the case when T1 is completed but T2 fails, then inconsistency will occur.

Isolation

It shows that the data which is used at the time of execution of a transaction cannot be used by the second transaction until the first one is completed.

In isolation, if the transaction T1 is being executed and using the data item X, then that data item can't be accessed by any other transaction T2 until the transaction T1 ends.

The concurrency control subsystem of the DBMS enforced the isolation property.

Durability

The durability property is used to indicate the performance of the database's consistent state. It states that the transaction made the permanent changes.

They cannot be lost by the erroneous operation of a faulty transaction or by the system failure. When a transaction is completed, then the database reaches a state known as the consistent state. That consistent state cannot be lost, even in the event of a system's failure.

The recovery subsystem of the DBMS has the responsibility of Durability property.

What is Serializability in DBMS?

In computer science, serializability is a property of a system describing how different processes operate on shared data. A system is serializable if its result is the same as if the operations were executed in some sequential order, meaning there is no overlap in execution. A database management system (DBMS) can be accomplished by locking data so that no other process can access it while it is being read or written.

DBMS Concurrency Control

Concurrency Control is the management procedure that is required for controlling concurrent execution of the operations that take place on a database.

Concurrent Execution in DBMS

- In a multi-user system, multiple users can access and use the same database at one time, which is known as the concurrent execution of the database. It means that the same database is executed simultaneously on a multi-user system by different users.
- While working on the database transactions, there occurs the requirement of using the database by multiple users for performing different operations, and in that case, concurrent execution of the database is performed.
- The thing is that the simultaneous execution that is performed should be done in an interleaved manner, and no operation should affect the other executing operations, thus maintaining the consistency of the database. Thus, on making the concurrent execution of the transaction operations, there occur several challenging problems that need to be solved.

Concurrency Control

Concurrency Control is the working concept that is required for controlling and managing the concurrent execution of database operations and thus avoiding the inconsistencies in the database. Thus, for maintaining the concurrency of the database, we have the concurrency control protocols.

Concurrency Control Protocols

The concurrency control protocols ensure the atomicity, consistency, isolation, durability and serializability of the concurrent execution of the database transactions. Therefore, these protocols are categorized as:

1. Lock Based Concurrency Control Protocol
2. Time Stamp Concurrency Control Protocol
3. Validation Based Concurrency Control Protocol

Lock-Based Protocol

In this type of protocol, any transaction cannot read or write data until it acquires an appropriate lock on it. There are two types of lock:

1. Shared lock:

It is also known as a Read-only lock. In a shared lock, the data item can only read by the transaction.

It can be shared between the transactions because when the transaction holds a lock, then it can't update the data on the data item.

2. Exclusive lock:

In the exclusive lock, the data item can be both reads as well as written by the transaction.

This lock is exclusive, and in this lock, multiple transactions do not modify the same data simultaneously.

Time-stamp Protocol

- The Timestamp Ordering Protocol is used to order the transactions based on their Timestamps. The order of transaction is nothing but the ascending order of the transaction creation.
- The priority of the older transaction is higher that's why it executes first. To determine the timestamp of the transaction, this protocol uses system time or logical counter.
- The lock-based protocol is used to manage the order between conflicting pairs among transactions at the execution time. But Timestamp based protocols start working as soon as a transaction is created.
- Let's assume there are two transactions T1 and T2. Suppose the transaction T1 has entered the system at 007 times and transaction T2 has entered the system at 009 times. T1 has the higher priority, so it executes first as it is entered the system first.
- The timestamp ordering protocol also maintains the timestamp of last 'read' and 'write' operation on a data.

Validation Based Protocol

Validation phase is also known as optimistic concurrency control technique. In the validation based protocol, the transaction is executed in the following three phases:

Read phase: In this phase, the transaction T is read and executed. It is used to read the value of various data items and stores them in temporary local variables. It can perform all the write operations on temporary variables without an update to the actual database.

Validation phase: In this phase, the temporary variable value will be validated against the actual data to see if it violates the serializability.

Write phase: If the validation of the transaction is validated, then the temporary results are written to the database or system otherwise the transaction is rolled back.

Granularity: It is the size of data item allowed to lock.

Multiple Granularity:

- It can be defined as hierarchically breaking up the database into blocks which can be locked.
- The Multiple Granularity protocol enhances concurrency and reduces lock overhead.
- It maintains the track of what to lock and how to lock.
- It makes easy to decide either to lock a data item or to unlock a data item. This type of hierarchy can be graphically represented as a tree.

For example: Consider a tree which has four levels of nodes.

- The first level or higher level shows the entire database.
- The second level represents a node of type area. The higher level database consists of exactly these areas.
- The area consists of children nodes which are known as files. No file can be present in more than one area.
- Finally, each file contains child nodes known as records. The file has exactly those records that are its child nodes. No records represent in more than one file.

Hence, the levels of the tree starting from the top level are as follows:

Database

Area

File

Record

DBMS Multiple Granularity

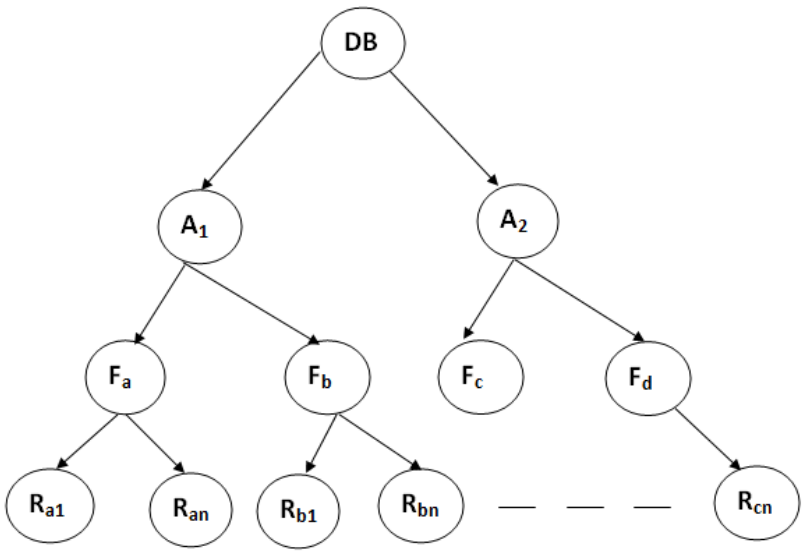


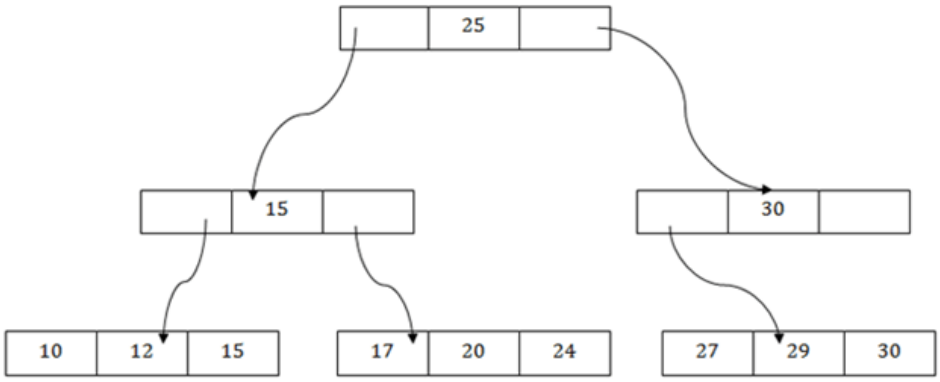
Figure: Multi Granularity tree Hierarchy

B+ File Organization

B+ tree file organization is the advanced method of an indexed sequential access method. It uses a tree-like structure to store records in File.

It uses the same concept of key-index where the primary key is used to sort the records. For each primary key, the value of the index is generated and mapped with the record.

The B+ tree is similar to a binary search tree (BST), but it can have more than two children. In this method, all the records are stored only at the leaf node. Intermediate nodes act as a pointer to the leaf nodes. They do not contain any records.



The above B+ tree shows that:

- There is one root node of the tree, i.e., 25.
- There is an intermediary layer with nodes. They do not store the actual record. They have only pointers to the leaf node.
- The nodes to the left of the root node contain the prior value of the root and nodes to the right contain next value of the root, i.e., 15 and 30 respectively.
- There is only one leaf node which has only values, i.e., 10, 12, 17, 20, 24, 27 and 29.
- Searching for any record is easier as all the leaf nodes are balanced.
- In this method, searching any record can be traversed through the single path and accessed easily.

Pros of B+ tree file organization

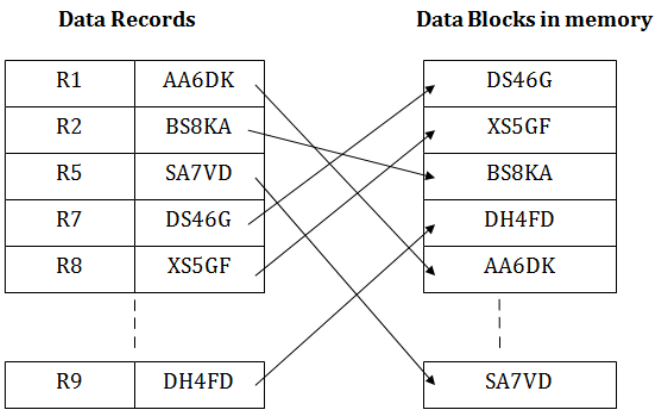
- In this method, searching becomes very easy as all the records are stored only in the leaf nodes and sorted the sequential linked list.
- Traversing through the tree structure is easier and faster.
- The size of the B+ tree has no restrictions, so the number of records can increase or decrease and the B+ tree structure can also grow or shrink.
- It is a balanced tree structure, and any insert/update/delete does not affect the performance of tree.

Cons of B+ tree file organization

This method is inefficient for the static method.

Indexed sequential access method (ISAM)

ISAM method is an advanced sequential file organization. In this method, records are stored in the file using the primary key. An index value is generated for each primary key and mapped with the record. This index contains the address of the record in the file.



If any record has to be retrieved based on its index value, then the address of the data block is fetched and the record is retrieved from the memory.

Pros of ISAM:

In this method, each record has the address of its data block, searching a record in a huge database is quick and easy.

This method supports range retrieval and partial retrieval of records. Since the index is based on the primary key values, we can retrieve the data for the given range of value. In the same way, the partial value can also be easily searched, i.e., the student name starting with 'JA' can be easily searched.

Cons of ISAM

This method requires extra space in the disk to store the index value.

When the new records are inserted, then these files have to be reconstructed to maintain the sequence.

When the record is deleted, then the space used by it needs to be released. Otherwise, the performance of the database will slow down.

BASIC PRACTICE QUESTIONS –

Q1 - What is normalization in the context of database design? Describe the purpose of normalization and provide examples of different normal forms. How does normalization improve data integrity and reduce redundancy in a relational database?

Q2 -Explain the concept of normalization in the context of database design. Why is normalization important and what are the different normal forms?

Q3 - What are the ACID properties in the context of database transactions? Describe each property and explain why they are essential for maintaining data integrity and consistency.

Q4 - Explain the concept of concurrency control in Database Management Systems (DBMS). Discuss the challenges associated with concurrent transactions and the need for concurrency control mechanisms. Provide examples of potential issues that may arise in a multi-user database environment without proper concurrency control.

Q5 - Compare and contrast the two main approaches to concurrency control in DBMS: locking-based concurrency control and timestamp-based concurrency control. Discuss the advantages and disadvantages of each approach,

Q6 – What is B+ tree organization? Explain its advantages and disadvantages.

What do you understand by Indexed sequential access method (ISAM)?

Write its pros and cons.

UNIT 5
SQL (Structured Query Language)

- 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.
- SQL allows users to query the database in a number of ways, using English-like statements.

Rules:

SQL follows the following rules:

- Structure query language is not case sensitive. Generally, keywords of SQL are written in uppercase.
- Statements of SQL are dependent on text lines. We can use a single SQL statement on one or multiple text line.
- Using the SQL statements, you can perform most of the actions in a database.
- SQL depends on tuple relational calculus and relational algebra.

History of SQL

SQL (Structured Query Language) is a programming language for storing, managing, manipulating, and processing data in relational databases. SQL has been around since the 1970s, and was standardized by the American National Standards Institute (ANSI) in 1986. Since then, SQL has become the most widely used language for managing relational databases and has undergone several revisions to keep pace with the evolving needs of the database industry.

SQL is pronounced “sequel” or sometimes “ess-cue-ell.”

What is SQL used for?

SQL is an essential part of the technology stack for many organizations and is widely used in business, government, and scientific applications across many roles including: Data Analysts, Business Intelligence Analysts, Data Scientists, Database Developers, and Data Engineers. SQL is used by organizations to interact with databases which store and manage data in a structured and efficient way.

The following are common tasks that people use SQL for:

- Creating, modifying, and deleting database tables and records.
- Inserting, updating, and deleting data in a database.
- Retrieving data from a database with SELECT statements.
- Grouping and aggregating data.
- Joining data from multiple tables.
- Creating views and stored procedures.
- Performing data analysis and data mining.
- Managing the security and permissions of a database.
- Monitoring and optimizing database performance.
- Backing up and restoring databases.

What are SQL Queries?

SQL Queries form the backbone of database interactions. They allow users to extract specific information from databases, enabling them to make informed decisions based on the data at hand. Whether you are working with financial records, user data, or inventory information, Queries empower you to retrieve the exact data you need efficiently.

Components of SQL Queries

a) **SELECT Statement:** "SELECT" is used to retrieve data from a database. It allows you to specify which particular columns you want to fetch from a particular table.

b) **FROM Clause:** The FROM clause indicates the table from which you want to retrieve data. It identifies the source of data for your Query.

c) **WHERE Clause:** The WHERE clause is used to filter data based on specific conditions. It allows you to extract only the records that meet certain criteria.

d) **ORDER BY Clause:** The ORDER BY clause is used to sort the result set in either ascending or descending order based on one or multiple columns.

e) **GROUP BY Clause:** The GROUP BY clause is employed to group rows with identical values in one or multiple columns. It is often used along with aggregate functions.

The basic structure of SQL Queries

Let's now delve into the Basic Structure of SQL Queries, including some essential operations:

Creating a database

To create a new database, you can use the following command like:

```
CREATE DATABASE database_name;
```

Creating a table

Tables are used to organise data into rows and columns. You can create a new table using the CREATE TABLE command:

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

Inserting data

To add new records to a table, you can use the INSERT INTO statement:

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

Retrieving data

To fetch data from a table, you can use the SELECT statement like:

SELECT column1, column2, ...

FROM table_name;

Updating data

To modify existing records in a table, you can use the UPDATE statement like:

UPDATE table_name

SET column1 = value1, column2 = value2, ...

WHERE condition;

Deleting data

To remove records from a table, you can use the DELETE FROM statement like:

DELETE FROM table_name

WHERE condition;

DDL Commands in SQL

DDL is an abbreviation of **Data Definition Language**.

The DDL Commands in Structured Query Language are used to create and modify the schema of the database and its objects. The syntax of DDL commands is predefined for describing the data. The commands of Data Definition Language deal with how the data should exist in the database.

Following are the five DDL commands in SQL:

1. CREATE Command
2. DROP Command
3. ALTER Command
4. TRUNCATE Command
5. RENAME Command

CREATE COMMAND

CREATE is a DDL command used to create databases, tables, triggers and other database objects.

Examples of CREATE Command in SQL

Example 1: This example describes how to create a new database using the CREATE DDL command.

Syntax to Create a Database:

CREATE Database Database_Name;

Suppose, you want to create a Books database in the SQL database. To do this, you have to

write the following DDL Command:

Create Database Books;

Example 2: This example describes how to create a new table using the CREATE DDL command.

Syntax to create a new table:

CREATE TABLE table_name

(

column_Name1 data_type (size of the column) ,

column_Name2 data_type (size of the column) ,

column_Name3 data_type (size of the column) ,

...

column_NameN data_type (size of the column)

) ;

DROP Command

DROP is a DDL command used to delete/remove the database objects from the SQL database. We can easily remove the entire table, view, or index from the database using this DDL command.

Examples of DROP Command in SQL

Example 1: This example describes how to remove a database from the SQL database.

Syntax to remove a database:

DROP DATABASE Database_Name;

Suppose, you want to delete the Books database from the SQL database. To do this, you have to write the following DDL command:

DROP DATABASE Books;

Example 2: This example describes how to remove the existing table from the SQL database.

Syntax to remove a table:

DROP TABLE Table_Name;

Suppose, you want to delete the Student table from the SQL database. To do this, you have to write the following DDL command:

DROP TABLE Student;

ALTER Command

ALTER is a DDL command which changes or modifies the existing structure of the database, and it also changes the schema of database objects.

We can also add and drop constraints of the table using the ALTER command.

Examples of ALTER Command in SQL

Example 1: This example shows how to add a new field to the existing table.

Syntax to add a newfield in the table:

ALTER TABLE name_of_table ADD column_name column_definition;

Suppose, you want to add the 'Father's_Name' column in the existing Student table. To do this, you have to write the following DDL command:

ALTER TABLE Student ADD Father's_Name Varchar(60);

Example 2: This example describes how to remove the existing column from the table.

Syntax to remove a column from the table:

ALTER TABLE name_of_table DROP Column_Name_1 , column_Name_2 , ….., column_Name_N;

Suppose, you want to remove the Age and Marks column from the existing Student table. To do this, you have to write the following DDL command:

ALTER TABLE Student DROP Age, Marks;

TRUNCATE Command

TRUNCATE is another DDL command which deletes or removes all the records from the table.

This command also removes the space allocated for storing the table records.

Syntax of TRUNCATE command

TRUNCATE TABLE Table_Name;

Example

Suppose, you want to delete the record of the Student table. To do this, you have to write the following TRUNCATE DDL command:

TRUNCATE TABLE Student;

The above query successfully removed all the records from the student table. Let's verify it by using the following SELECT statement:

SELECT * FROM Student;

RENAME Command

RENAME is a DDL command which is used to change the name of the database table.

Syntax of RENAME command

RENAME TABLE Old_Table_Name TO New_Table_Name;

Example

RENAME TABLE Student TO Student_Details ;

This query changes the name of the table from Student to Student_Details.

DML Commands in SQL

DML is an abbreviation of **Data Manipulation Language.**

The DML commands in Structured Query Language change the data present in the SQL database. We can easily access, store, modify, update and delete the existing records from the database using DML commands.

Following are the four main DML commands in SQL:

- **SELECT Command**
- **INSERT Command**
- **UPDATE Command**
- **DELETE Command**

SELECT DML Command

SELECT is the most important data manipulation command in Structured Query Language. The SELECT command shows the records of the specified table. It also shows the particular record of a particular column by using the WHERE clause.

Syntax of SELECT DML command

SELECT column_Name_1, column_Name_2, ….., column_Name_N FROM Name_of_table;

Here, column_Name_1, column_Name_2, ….., column_Name_N are the names of those columns whose data we want to retrieve from the table.

If we want to retrieve the data from all the columns of the table, we have to use the following SELECT command:

SELECT * FROM table_name;

Examples of SELECT Command

Example 1: This example shows all the values of every column from the table.

SELECT * FROM Student;

Example 2: This example shows all the values of a specific column from the table.

SELECT Emp_Id, Emp_Salary FROM Employee;

If you want to access all the records of those students whose marks is 80 from the above table, then you have to write the following DML command in SQL:

SELECT * FROM Student WHERE Stu_Marks = 80;

INSERT DML Command

INSERT is another most important data manipulation command in Structured Query Language, which allows users to insert data in database tables.

Syntax of INSERT Command

INSERT INTO TABLE_NAME (column_Name1 , column_Name2 , column_Name3 , ... c

olumn_NameN) VALUES (value_1, value_2, value_3, ... value_N) ;

Examples of INSERT Command

Example 1: This example describes how to insert the record in the database table.

Let's take the following student table, which consists of only 2 records of the student.

Stu_Id	Stu_Name	Stu_Marks	Stu_Age
101	Ramesh	92	20
201	Jatin	83	19

Suppose, you want to insert a new record into the student table. For this, you have to

write the following DML INSERT command:

INSERT INTO Student (Stu_id, Stu_Name, Stu_Marks, Stu_Age) VALUES (104, Anmol, 89, 19);

UPDATE DML Command

UPDATE is another most important data manipulation command in Structured Query Language, which allows users to update or modify the existing data in database tables.

Syntax of UPDATE Command

UPDATE Table_name SET [column_name1= value_1, ….., column_nameN = value_N] WHERE CONDITION;

Here, 'UPDATE', 'SET', and 'WHERE' are the SQL keywords, and 'Table_name' is the name of the table whose values you want to update.

Examples of the UPDATE command

Example 1: This example describes how to update the value of a single field.

Suppose, you want to update the Product_Price of the product whose Product_Id is P102. To do this, you have to write the following DML UPDATE command:

UPDATE Product SET Product_Price = 80 WHERE Product_Id = 'P102' ;

Example 2: This example describes how to update the value of multiple fields of the database table.

Suppose, you want to update Stu_Marks and Stu_Age of that student whose Stu_Id is 103 and 202. To do this, you have to write the following DML Update command:

UPDATE Student SET Stu_Marks = 80, Stu_Age = 21 WHERE Stu_Id = 103 AND Stu_Id = 202;

DELETE DML Command

DELETE is a DML command which allows SQL users to remove single or multiple existing records from the database tables.

This command of Data Manipulation Language does not delete the stored data permanently

from the database. We use the WHERE clause with the DELETE command to select specific rows from the table.

Syntax of DELETE Command

DELETE FROM Table_Name WHERE condition;

Examples of DELETE Command

Example 1: This example describes how to delete a single record from the table.

DELETE FROM Product WHERE Product_Id = 'P202' ;

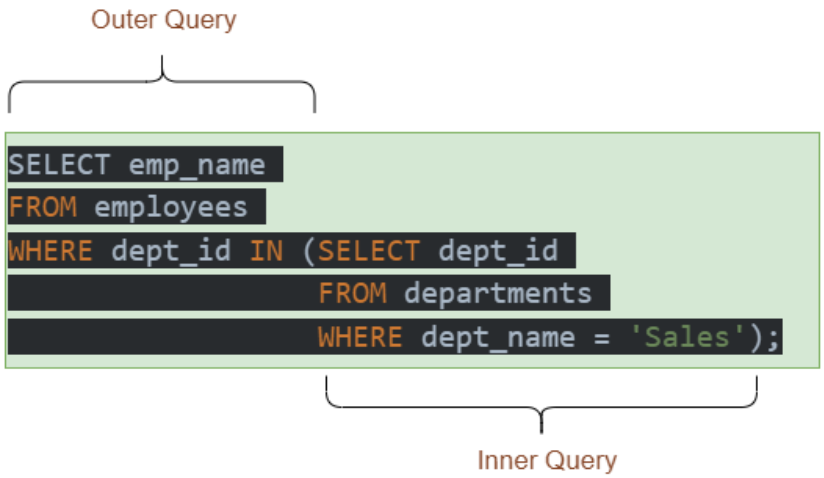
Example 2: This example describes how to delete the multiple records or rows from the database table.

Suppose, you want to delete the record of those students whose Marks is greater than 70. To do this, you have to write the following DML Update command:

DELETE FROM Student WHERE Stu_Marks > 70 ;

Nested Query

In SQL, a nested query involves a query that is placed within another query. Output of the inner query is used by the outer query. A nested query has two SELECT statements: one for the inner query and another for the outer query.



Syntax of Nested Queries

The basic syntax of a nested query involves placing one query inside of another query. Inner query or subquery is executed first and returns a set of values that are then used by the outer query. The syntax for a nested query is as follows:

SELECT column1, column2, ...

FROM table1

WHERE column1 IN (SELECT column1

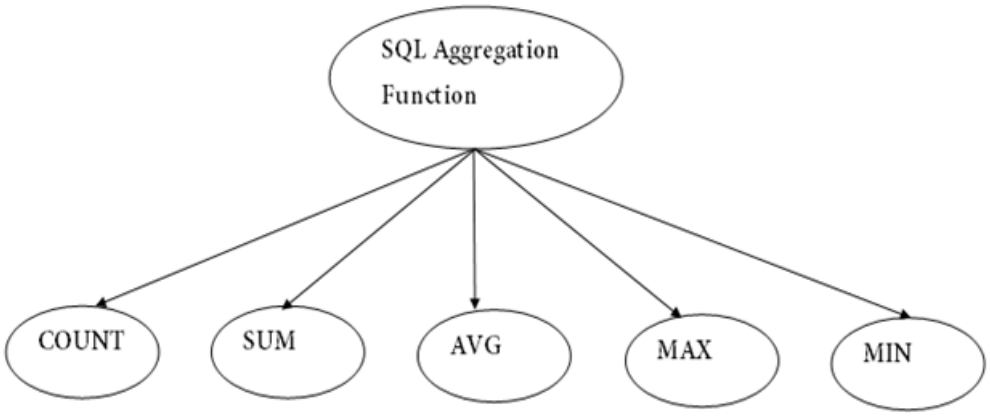
FROM table2

WHERE condition);

SQL Aggregate Functions

- SQL aggregation function is used to perform the calculations on multiple rows of a single column of a table. It returns a single value.
- It is also used to summarize the data.

Types of SQL Aggregation Function



COUNT FUNCTION

COUNT function is used to Count the number of rows in a database table. It can work on both numeric and non-numeric data types.

COUNT function uses the COUNT(*) that returns the count of all the rows in a specified table. COUNT(*) considers duplicate and Null.

Syntax

COUNT(*)

or

COUNT([ALL|DISTINCT] expression)

SELECT COUNT(*)

FROM PRODUCT_MAST;

Output:

10

Example: COUNT() with WHERE

SELECT COUNT(*)

FROM PRODUCT_MAST;

WHERE RATE>=20;

Output:

7

Example: COUNT() with DISTINCT

```
SELECT COUNT(DISTINCT COMPANY)
FROM PRODUCT_MAST;
```

Output:

3

Example: COUNT() with HAVING

```
SELECT COMPANY, COUNT(*)
FROM PRODUCT_MAST
GROUP BY COMPANY
HAVING COUNT(*)>2;
```

SUM Function

Sum function is used to calculate the sum of all selected columns. It works on numeric fields only.

Syntax

SUM()

or

SUM([ALL|DISTINCT] expression)

Example: SUM()

```
SELECT SUM(COST)
FROM PRODUCT_MAST;
```

Example: SUM() with WHERE

```
SELECT SUM(COST)
FROM PRODUCT_MAST
WHERE QTY>3;
```

Example: SUM() with HAVING

```
SELECT COMPANY, SUM(COST)
FROM PRODUCT_MAST
GROUP BY COMPANY
HAVING SUM(COST)>=170;
```

AVG function

The AVG function is used to calculate the average value of the numeric type. AVG function returns the average of all non-Null values.

Syntax

AVG()

or

AVG([ALL|DISTINCT] expression)

Example:

SELECT AVG(COST)

FROM PRODUCT_MAST;

MAX Function

MAX function is used to find the maximum value of a certain column. This function determines the largest value of all selected values of a column.

Syntax

MAX()

or

MAX([ALL|DISTINCT] expression)

Example:

SELECT MAX(RATE)

FROM PRODUCT_MAST;

MIN Function

MIN function is used to find the minimum value of a certain column. This function determines the smallest value of all selected values of a column.

Syntax

MIN()

or

MIN([ALL|DISTINCT] expression)

Example:

SELECT MIN(RATE)

FROM PRODUCT_MAST;

SQL CLAUSES

- SQL clause helps us to retrieve a set or bundles of records from the table.
- SQL clause helps us to specify a condition on the columns or the records of a table.

Different clauses available in the Structured Query Language are as follows:

- **WHERE CLAUSE**
- **GROUP BY CLAUSE**
- **HAVING CLAUSE**
- **ORDER BY CLAUSE**

WHERE CLAUSE

A WHERE clause in SQL is used with the SELECT query, which is one of the data manipulation language commands. WHERE clauses can be used to limit the number of rows to be displayed in the result set, it generally helps in filtering the records. It returns only those queries which fulfill the specific conditions of the WHERE clause. WHERE clause is used in SELECT, UPDATE, DELETE statement, etc.

WHERE clause with SELECT Query

Asterisk symbol is used with a WHERE clause in a SELECT query to retrieve all the column values for every record from a table.

Syntax of where clause with a select query to retrieve all the column values for every record from a table:

SELECT * FROM TABLENAME WHERE CONDITION;

If according to the requirement, we only want to retrieve selective columns, then we will use below syntax:

SELECT COLUMNNAME1, COLUMNNAME2 FROM TABLENAME WHERE CONDITION;

Example 1:

Write a query to retrieve all those records of an employee where employee salary is greater than 50000.

Query:

SELECT * FROM employees WHERE Salary > 50000;

The above query will display all those records of an employee where an employee's salary is greater than 50000. Below 50000 salary will not be displayed as per the conditions.

Example 2:

Write a query to update the employee's record and set the updated name as 'Harshada Sharma' where the employee's city name is Jaipur.

Query:

UPDATE employees SET Name = "Harshada Sharma" WHERE City = "Jaipur";

GROUP BY CLAUSE

The Group By clause is used to arrange similar kinds of records into the groups in the Structured Query Language. The Group by clause in the Structured Query Language is used with Select Statement. Group by clause is placed after the where clause in the SQL statement. The Group By clause is specially used with the aggregate function, i.e., max (), min (), avg (), sum (), count () to group the result based on one or more than one column.

The syntax of Group By clause:

SELECT * FROM TABLENAME GROUP BY COLUMNNAME;

The above syntax will select all the data or records from the table, but it will arrange all those data or records in the groups based on the column name given in the query.

The syntax of Group By clause with Aggregate Functions:

SELECT COLUMNNAME1, Aggregate_FUNCTION (COLUMNNAME) FROM TABLENAME GROUP BY COL

Example 1:

Write a query to display all the records of the employees table but group the results based on the age column.

Query:

SELECT * FROM employees GROUP BY Age;

Example 2:

Write a query to display all the records of the employees table grouped by the designation and salary.

Query:

SELECT * FROM employees GROUP BY Salary, Designation;

HAVING CLAUSE:

When we need to place any conditions on the table's column, we use the WHERE clause in SQL. But if we want to use any condition on a column in Group By clause at that time, we will use the HAVING clause with the Group By clause for column conditions.

Syntax:

SELECT * FROM TABLENAME GROUP BY COLUMNNAME HAVING CONDITION;

Example 1:

Write a query to display the name of employees, salary, and city where the employee's maximum salary is greater than 40000 and group the results by designation.

Query:

SELECT Name, City, MAX (Salary) AS Salary FROM employees GROUP BY Designation HAVING MAX (Salary) > 40000;

The above output shows that the employee name, salary, and city of an employee where employee salary is greater than 40000 grouped by designation. (Employees with a similar designation are placed in one group, and those with other designation are placed separately).

Example 2:

Write a query to display the name of employees and designation where the sum of an employee's salary is greater than 45000 and group the results by city.

Query:

SELECT Name, Designation, SUM (Salary) AS Salary FROM employees GROUP BY City HAVING SUM (Salary) > 45000;

The above output shows the employee name, designation, and salary of an employee. The sum of salary is greater than 45000 grouped by city. (Employees with similar cities are placed in one group and those with a different city are not similar are placed separately).

ORDER BY CLAUSE

Whenever we want to sort anything in SQL, we use the ORDER BY clause. The ORDER BY clause in SQL will help us to sort the data based on the specific column of a table. This means that all the data stored in the specific column on which we are executing the ORDER BY clause will be sorted. The corresponding column values will be displayed in the sequence in which we have obtained the values in the earlier step.

As we all know, sorting means either in ASCENDING ORDER or DESCENDING ORDER. In the same way, ORDER BY CLAUSE sorts the data in ascending or descending order as per our requirement. The data will be sorted in ascending order whenever the ASC keyword is used with ORDER by clause, and the DESC keyword will sort the records in descending order.

By default, sorting in the SQL will be done using the ORDER BY clause in ASCENDING order if we didn't mention the sorting order.

Syntax of ORDER BY clause without asc and desc keyword:

SELECT COLUMN_NAME1, COLUMN_NAME2 FROM TABLE_NAME ORDER BY COLUMNAME;

Syntax of ORDER BY clause to sort in ascending order:

SELECT COLUMN_NAME1, COLUMN_NAME2 FROM TABLE_NAME ORDER BY COLUMN_NAME ASC;

Syntax of ORDER BY clause to sort in descending order:

SELECT COLUMN_NAME1, COLUMN_NAME2 FROM TABLE_NAME ORDER BY COLUMN_NAME DESC;

Example 1:

Write a query to sort the records in the ascending order of the employee designation from the employees table.

Query:

SELECT * FROM employees ORDER BY Designation;

Here in a SELECT query, an ORDER BY clause is applied on the column 'Designation' to sort the records, but we didn't use the ASC keyword after the ORDER BY clause to sort in ascending order. So, by default, data will be sorted in ascending order if we don't specify asc keyword.

Example 2:

Write a query to display employee name and salary in the ascending order of the employee's salary from the employees table.

Query:

SELECT Name, Salary FROM employees ORDER BY Salary ASC;

Here in a SELECT query, an ORDER BY clause is applied to the 'Salary' column to sort the records. We have used the ASC keyword to sort the employee's salary in ascending order.

BASIC PRACTICE QUESTIONS –

Q1 -What is SQL, and what are its fundamental components and functionalities?

Q2 – Discuss in detail, various commands and queries used in DDL and DML operations of SQL.

Q3 – What are aggregate functions? Explain its types with examples.

Q4 – Discuss in detail with examples, the types of SQL clauses. Also, write its query.

Q5 – Write down the difference between ORDER BY and GROUP BY clause with examples.