Pithapur Rajah's Government College
Department of Computer Applications
Sem: IV **II B.com (CA)** **K. Sravani Devi**
Lecturer in Computer Applications

Data Base Management System With Oracle
Unit-1

Unit 1: Overview of Database Systems: Introduction: Database system, Characteristics (Database Vs File System), Database Users, Advantages of Database systems, Database applications.

Data Models: Introduction; types of data models, Concepts of Schema, Instance and data independence; Three tier schema architecture for data independence; Database system structure, environment, Centralized and Client Server architecture for the database.

1. What Is Dbms? Explain Advantages Of Dbms?

A) A Database Management System Is a Collector Of Interrelated Data And A Set Of Programmes to Access Those Data.

The Dbms Is A General-Purpose Software System That Facilitates the Process of Defining, Constructing and Manipulating Databases For Various Applications.

Objectives Of Dbms:-

1. Data Availability - Data Are Made Available To Large Variety Of Users.

2. Data Integrity - It Refers to The Completeness, Correctness And Consistency Of Data. Related To Quality Of Data

3. Data Security - Only Authorized Users Can Access The Data

4. Data Independence - Dbms Supports The Concept Of Data Independence . Since It Represents A System For Managing Data Seperately From The Program That Use The Data

Advantages Of Dbms:-

1. Minimized Redundancy :-

The Data In A Dbms Is A More Concise Because As A General Rule The Information In The Appears Once. This Reduces Data Redundancy (Redundency Creates Several Problems Like Requiring Extra Storage, Entering Same Data More Than Once Etc Minimising Redundancy Can Therefore Significantly Reduce These Problems.

2. Elimination Of In Consistency :-

Dbms Centralize The Entire Database So That Changes Once Made Are Reflected To All The Tables. So Inconsistency Problems Is Eliminated;

3. Sharing Of Data :-

Many Users Can Share The Same Database It May Connected Through A Network.

4. **Data Security**:-

We Can Restrict Certain People From Accessing The Database Or Allow Them To See Certain Portions Of The Database While Blocking Sensitive Information.

5. **Data Integrity** :-

Accurate, Consistent, Complete And Up-To -Date. Data Is A Sign Of Data Integrity. Dbms Supports Data Integrity.

6. **Standards Can Be Enforced** :-

Standards Are Easier To Enforce Standards May Relate To The Naming Of Data, Format Of Data.

7. **Provides Backup & Recovery** :-

Centralizing A Database Provides That Schemes Such As Recovery And Backups From The Failures Including The Disk Crash, Power Failures, Software Errors Which May Help The Database To Recover From Inconsistent State To The State That Existed Prior To The Occurence Of The Failure, Through Methods Arevery Complex.

8. **Concurrent Access** :-

A Dbms Schedule Concurrent Access To The Data In Such A Manner That Users Can Think Of The Data Is Being Accessed By Only One User At A Time.

9. **Data Administration** :-

When Several Users Share The Data, Centralizing The Administration Of Data Can Offer Significant Improvement.

Disadvantages Of Dbms :-

→High-Cost.

→ Complexity Of Backup And Recovery.

→ Security Threats.

2. Explain About Characteristics And Drawbacks Of File-Based System?

A) File-Based System:-

File Based System Was An Early Attempt To Computerize The Manual Filing System. It Is Basically A Collection Of Application Programs That Performed

Services For End Users. Each Program Within A File Based System Defines Manages Its Own Data.

Characteristics Of File-Based System :-

→ It Is A Group Of Files Storing Data Of An Organization.

→Each File Is Independent From One Another.

→Each File Contained And Processed Information For One Specific Task, Such As Accounting Or Inventory.

→Each File Must Have Its Own File Management System.

Drawbacks Of File-Based System :-

1. Data Redundancy :-

Data Redundancy Means Duplication Of Data Values. Some Information Is Duplicated In Several Files. This Makes Data Redundancy. Duplication Is Wasteful. It Costs Time And Money.

2. Data Inconsistency :-

Data Inconsistency Means Different Copies Of Same Data Are Not Matching That Means Different Versions Of Same Basic Data Are Existing In Different Files. This Occurs As The Result Of Update Operations That Are Not Updating The Same Data Stored At Different Places.

3. Data Isolation:-

Data Are Scattered In Various Files And The Files May Be In Different Formats, Writing New Application Program To Retrieve The Data Is Difficult.

4. Integrity Problems:-

The Data Values Stored In The Database Must Satisfy Certain Types Of (Consistency) Constraints. For Example, The Balance Of A Bank Account May Never Fall Below A Prescribed Amount (&$25). These Constraints Are Enforced In The System By Adding Appropriate Code In The Various Application Programs.

5. Atomicity Problem

It Is Difficult To Ensure Atomicity In File- Processing System.

Eg:-Consider Transferring 100 From Account A To B .If A Failure Occurs During Execution Of The Program. It Is Possible That 100 Was Removed From Account A But Not Credited In Account B, Resulting In An Inconsistent Database State. I.E, The Funds Transfer Must Be Atomic And It Must Happen In Its Entirely Or Not At All.

6. Concurrent Access Anomalies:-

In Order To Improve The Overall Performance Of The System And Obtain A Faster Response Time, Many Systems Allow Multiple Users To Update The Data Simultaneously . In Such An Environment, Interaction Of Concurrent Updates May Result In Inconsistent Data.

7 Security Problems:-

Not Every User Of The System Should Be Able To Access All The Data. For Example In Banking Systems, Payroll Personal Need Only That Part Of The Database That Has Information About Various Bank Employees. They Do Not Need Access To Information About Customer Accounts. It Is Difficult To Enforce Such Security Constraints.

3. Explain The Components Of Database System With Neat Diagram.

A) Components And Interfaces Of Dbms :-

Dbms Acts As An Interface Between The Users And The Database.The User Requests The Dbms To Perform Various Operations On The Database, The Components Of The Dbms Perform These Requested Operations On The Database And Provide Necessary Data To Users.

A Database Management System I.E., Composed Of Following Components, Which Coordinate With Each Other To Form An Effective Database System.

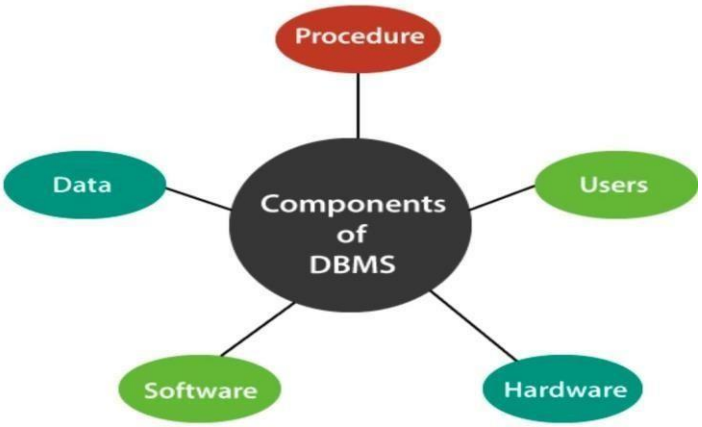
A. Data

B. Hardware

C. Software

D. Users

E.Procedure



A] Data :-

It Is A Very Important Component Of The Database System . Most Of The Organizations Generate, Store And Process Lange Amount Of Data. The Data Acts As A Bridge Between The Machine Parts. I.E., Hardware & Software And The Users) Which Directly Access It Or Access It Through Some Application Programs.Data May Be Of Different Types.

I- User Data:-

It Consists Of Table (S) Of Data Called Relations. A Relation Must Be Stuchured Properly.

Ii. Meta Data:-

Meta Data Means Data About Data" It Is A Description About The Structure Of The Data Base System) Tables Store Meta Data Which Includes.

- No Of Tables & Table Names
- No Of Fields & Field Names
- Primary Key Fields.

B] Hardware :-

The Hardware Consists Of The Secondary Storage Devices Such As Magnetic Disks, Optical Disks Magnetic Tapes Etc ... On Which Data Is Stored Together. It Also Includes The I/O Devices, Processors, Main Memory Which Are Used For Storing The Data.

C] Software:

The Software Part Consists Of Dbms Which Acts As A Bridge Between User And The Database. All Requests From Users For Access To The Database Are Handled By The Dbms. For Performing Operation Such As Insertion, Deletion And Updation. We Can Either Use An Query Language Like Sql Or Application Software Such As Visual Basic.

D] Users:-

The Users Are The People Who Manage The Database And Perform Different Operations On The Data Bases In The Database System. Users Can Be Staff, Clerks, Managers, Executives Application Programmers, Data Base Administrators And End-Users Etc On The Basis Of The Job And
Requirements Made By Them They Are Provided Access To The Database Totally Or Partially.

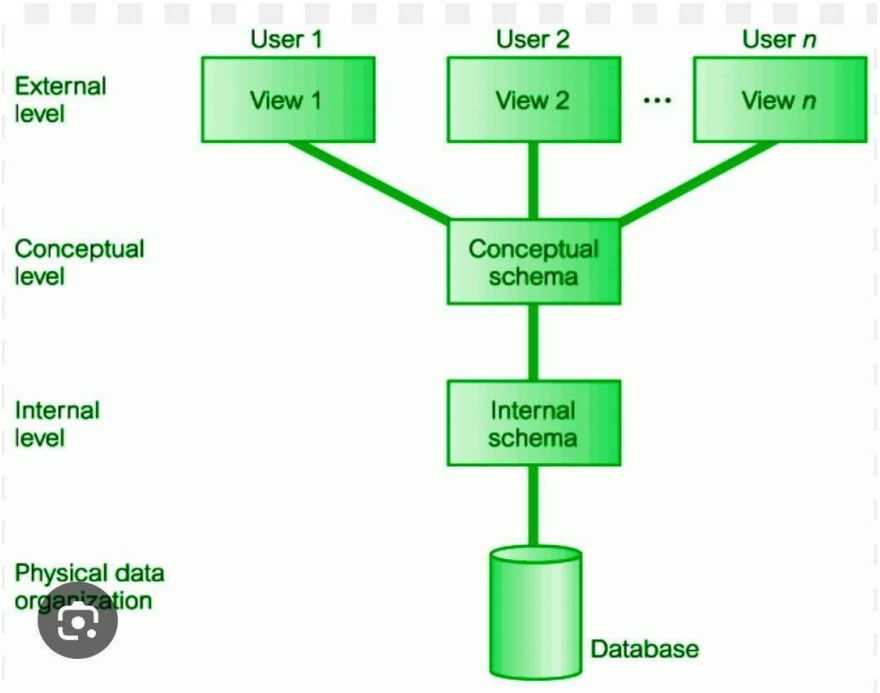
E] Procedure:-

The Last Component Of Dbms Is A Set Of Procedures Of Rules That Should Be Clearly Defined And Followed By The Users Of The Database.

4. Explain Dbms Architecture In Detail.

A) **A Dbms Architecture** : [Three Levels Of Data Abstraction]

The Three Schema Architecture :- The Goal Of Three-Schema Architecture Is To Seperate The User Applications And Physical Data Base. In This Architecture, Schemas Can Be Defined At The Following 3 Levels.



1. Internal Schema (Or) Internal Level :-

The Internal Level Has A Internal Schema; Which Describes The Physical Storage Structure Of The Database. The Internal Schema Uses A Physical Model And Described The Complete Details Of The Data Storage And Access Paths For The Database.

2. Conceptual Level :-

Conceptual Level Has A Conceptual Schema Which Describes The Structure Of The Whole Database For A Community Of Users. The Conceptual Schema Hides The Details Of Physical Storage Structures And Concentrates On Describing Entities, Data Types, Relationships, User Operations And Constraints. Usually A Representational Data Model Is Used To Describe The Conceptual Schema When A Database System Is Implemented.A High Level Data Model Or An Implementation Data Model Can Be Used At This Level.

3. External Or View Level :-

The External & View Level Includes A Number Of External Schemas Or User Views. Each External Schema Describes The Part Of The Database That A Particular User Group Is Interested In And Hides The Rest Of The Database From That User Group. A High Level Database From That User Group.A High Level Data Model Or Implementation Model Can Be Used At This Level.

The Three Schema Architecture Is A Convenient Tool With Which The User Can Visualize The Schema Levels In A Database System. Most Dbms 'S Do Not Separate The 3 Levels Completely, But Support The Three Schema Architecture. To Some Extent. Some Dbms 'S May Include Physical Level Details In The Conceptual Schema

The Dbms Must Transforms A Request Specified On An External Schema Into A Request Against The Conceptual Schema And Then Into A Requests On The Internal Schema For Processing Over The Stored Database.

Similarly The Data Extracted From The Stored Database Must Be Reformatted To Match The User's External View. The Process Of Transforming Requests And Results Between Levels Are Called Mappings.

Shorts

1. Explain About Objectives Of Dbms.

1. A Database Management System Is A Collector Of Interrelated Data And A Set Of Programmes To Access Those Data.

The Dbms Is A General-Purpose Software System That Facilitates The Process Of Defining, Constructing And Manipulating Databases For Various Applications.

Objectives Of Dbms:-

1. Data Availability - I-E Data Are Made Available To Large Variety Of Users.

2. Data Integrity - It Refers To The Completeness, Correctness And Consistency Of Data. Related To Quality Of Data

3. Data Security - Only Authorized Users Can Access The Data

4. Data Independence - Dbms Supports The Concept Of Data Independence . Since It Represents A System For Managing Data Seperately From The Program That Use The Data

2. Explain About Database Users?

1. There Are 4 Different Types Of Database System Users, Differentiated By That They The Way That Expect To Interact With The System.

I] Application Programmers :-

Application Programmers Are Computer Professionals Who Write Application Programs. Thay Con Choose From Many Tools To Develop User Interface "Rad" Tools Are Tools That Enable An Application Programmer To Construct Forms And Reports Without Writing A Programme.

Ii] Sophisticated Users:-

They Interact With The System Without Writing Programs. Programs. Instead They Form Their Requests In A Database Query Language. They Submit Each Query To A Query Processor Whose Function Is To Break Down Dml Stmt Into Instructions That The Storage Manager Understands. Analysis Who Who Submit Queries To Explore Data In The Database Fall In This Category.

Iii]Specialized Users:-

Specialized Users Are Sophisticated Users Who Write Specialized Data Base Applications That Do Not Fit Into The Traditional Dare Processing Framework.

Iv] Naive Users :-

They Are Unsophisticated Users Who Interact With The System By Invoking One Of The Application Programs That Have Been Written Previously.For Example, A Bank-Teller Who Needs To Transfer 50 From Account A To Account B Invokes A Program Called Transfer. This Program Asks The Teller For The Amount Of Money To Be Transferred, The Account From Which The Money Is To Be Transferred And The Account To Which The Money Is To Be Transferred.

3. What Are The Functions Of DBA?

1. Database Administrator :-

The Person Who Has Central Control Over The System Is Called The Database Administrator. (DBA)

Functions Of DBA:- Schema

Definition -

The DBA Creates The Original Database Schema By Executing A Set Of Data Definition Statements In The DDL.

Storage Structhore And Access-Method Definition- The DBA Creates Appropriate Storage Structures And Access Methods.

Schema Physical Organization Modification

The DBA Carries Out Changes To The Schema And Physical Organization To Reflect The Changing Needs Of The Organization, Or To Alter The Physical Organization To Improve Performance.

Granting Of Authorization For Data Access

By Granting Different Types Of Authorisation DBA Can Regulate Which Parts Of The Database, Various Users Can Access.

Integrity Constraints Specification

DBA Can Specify Integrity Constraints.

4. Distinguish Between Data And Information?

A)

Data	**Information**
1. Data Is Raw Fact & Figures. Eg: 32	1. Information Is Processed Form Of Data. Eg: Age: 32
2-It Is Not Significant To A Business.	2. It Is Significant To A Business.
3. It Does Not Help In Decision Making.	3. It Helps In Decision Making
4. Data Must Be Processed To Understand.	4. It May Be Processed Further To Make It More Understandable.
5. Difficult To Understand Properly.	5. Easy To Understand.
6. Observations And Recordings Are Done To Obtain Data.	6. Analysis Is Done To Obtain Information.

UNIT-2

Unit 2: Relational Model: Introduction to relational model, Codd's rules, concepts of domain, attribute, tuple, relation, constraints (Domain, Key constraints, integrity constraints) and their importance , concept of keys (super key, candidate key, primary key, surrogate key, foreign key) , relational Algebra & relational calculus.

Normalization: Purpose of Normalization or schema refinement, concept of functional dependency, normal forms based on functional dependency(1NF, 2NF and 3NF), Boyce-codd normal form(BCNF

LONGS

1. what is data Model? write about relational data model?

A) Data Model: A collection of concepts that can be used to describe the structure of a database.

Relational Data Model - Introduction:

The relational Model is the conceptual basic Of Relational Databases. It is proposed by Dr.E-F codd. It is considered as one of the most Popular developments in the database technology. Because it can be used for representing most of the real world objects and the relationships between them. The relational model represents the database as a collection of relations (Tables).

Advantages of Relational Model:

- Data integrity at various levels
- Data Consistency and Accuracy.
- Easy data retrieval and data sharing

Basic Concepts of Relational Model:

(or)

Relational Model Concepts

The relational Model uses formal terms like. attribute, Tuple, relation, domain, schema.

Attribute:

An attribute can be defined as a characteristic of data. A real world data feature modelled in the database is represented by an attribute .For a person the attributes can be Name, Gender, Address, Age etc... Informally, a Column & field in a table..

Tuple:-

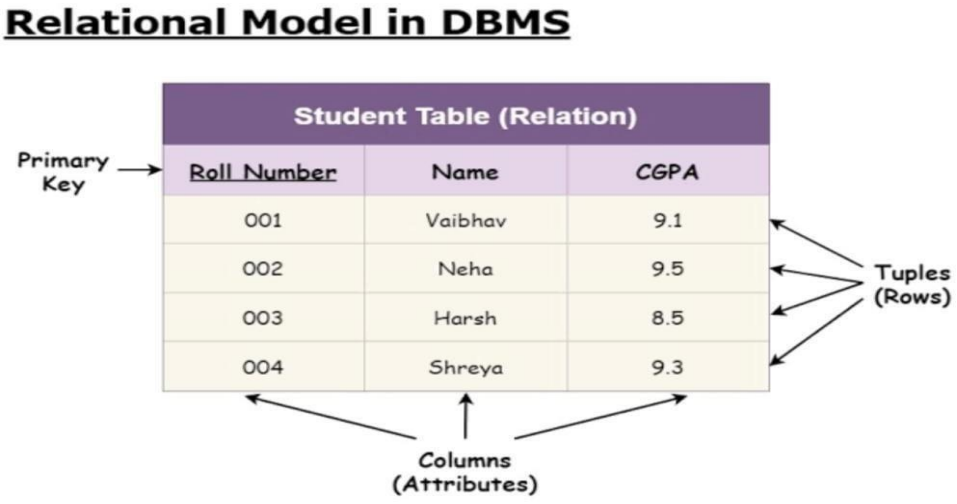
A Tuple represents a collection of information that describes a person, place, thing. Informal terms used for tuples are row in a table (or) record in in a data file.

Relation:-

A Relation is the core of the relational model. It is nothing but a table. (Date) in a relation, data is organized interms of rows and columns.

Domain:-

A Domain is the sets of possible values that are all of the same type for a given attribute.



2. Write about CoDD'S Rules?

A) Codd Rules:-

A set of 13 rules used to determine if a DBMS can be considered a RDBMS. Dr-E-F-Codd framed these rules. An RDBMS product has to satisfy at least 6 of the 13 rules to be accepted as a full-fledged RDBMS.

1) It states that all subsequent rules are based on the notion that in order for a database to be considered relational, it must use it's relational facilities exclusively to manage the database.

2) Information Rule: All information in a relational database must be logically represented in the form of tables.

3) Guaranteed Access: Every piece of the data in a table must be accessible.

4) Systematic treatment of null values: The null vales in database must be given a systematic and uniform treatment. a

5) Database description rule: The database description is represented at the logical level in the same way as ordinary data.

6) Comprehensive Data sub language: It may support many languages. But at least one of them should allow user to do the following

- Define tables & views
- Query & update the data
- set integrity constraintis
- set authorizations.

7) View updating: Any view that is the Theoritically updatable must be updatable through the RDBMS.

8) High-level insert, update and Delete: database must Support high level insertion, updation. It must not be limited to single row.

9) physical Data independence:the execution of other requests and application programs is not affected by changes in the physical data access and storage methods.

10) Logical Data independence: Logical changes in tables need not require modifications in the programs on in the format of adhoe requests.

11) Integrity Independence:- they can be changed without necessitating charges in the application program.

12) Distribution Independence:- users should not have to be aware of whether a database is distributed at different sites or not.

13) Non-Subversion Rule: If the system supports low level access for the data there must not be a way to bypass the integrity rules.

3. Explain the operations of Relational Algebra?

Relational algebra is a formal language for the manipulation of relational databases. It provides a set of operations that take one or two relations (tables) as input and produce a new relation as output.

1. Selection (σ)

Symbol: σ_{condition}(Relation)

Description: Selects rows from a relation that satisfy a given condition. It filters tuples based on specified criteria.

Example: If we have a relation Employee(Name, Age) and want to select employees older than 30, we use: σ_{Age >30}(Employee)

2. Projection (π)

Symbol: _{attributes}(Relation)

Description: Extracts specific columns from a relation, removing duplicates from the result.

Example: To retrieve only the names of employees from the Employee relation, we use:

π_{Name}(Employee)

3. Union (U)

Symbol: Relation1 U Relation2

Description: Combines tuples from two relations. The relations must have the same attributes (arity and domains must match).

Example: If R and S are two relations with the same schema, then RUS gives the set of all tuples that are in R or S, without duplicates.

4. Set Difference (-)

Symbol: Relation1 - Relation2

Description: Returns tuples that are in the first relation but not in the second relation. Both relations must have the same schema.

Example: If R contains all employees and S contains all employees who have left the company, RS will return employees who are still with the company.

5. Cartesian Product (x) Symbol:

Relation1 × Relation2

Description: Combines every tuple from the first relation with

every tuple from the second relation. The result is a relation that contains all possible pairs of tuples from the two input relations.

Example: If R(A) has 3 tuples and S(B) has 2 tuples, R × S will have 6 tuples.

6. Rename (p)

Description: Renames a relation or its attributes. This is useful when performing operations on relations with conflicting attribute names.

Symbol: p_{new_name}(Relation)

Example: To rename a relation Employee to Staff, use: p_{Staff}(Employee)

7. Intersection (∩)

Symbol: Relation1∩ Relation2

Description: Returns tuples that are present in both relations. Like union, the two relations must have the same schema.

Example: If R contains employees in department A and S contains employees in department B, ROS will return employees common to both departments.

8. Natural Join (⋈)

Symbol: Relation1 ⋈Relation2

Description: Combines two relations based on common attributes. It performs a join on attributes with the same name in both relations, removing duplicate columns from the result.

Example: If Employee(Name, DeptId) and Department(DeptId, DeptName) are two relations, a natural join on DeptId would result in a new relation with the schema Employee.Name, DeptId, DeptName.

9. Division (÷)

Symbol: Relation1÷ Relation2

Description: Used to find tuples in one relation (Relation1) that are associated with all tuples in another relation (Relation2).
Typically, Relation2 contains a single column.

Example: If R(A, B) contains courses taken by students, and S(B) contains a list of all required courses, then R + S returns the students who have taken all the required courses.

3. What is normalization? Explain the normal forms?

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

Important Points Regarding Normal Forms in DBMS

First Normal Form (1NF): This is the most basic level of normalization. In INF, each table cell should contain only a single value, and each column should have a unique name. The first normal form helps to eliminate duplicate data and simplify queries.

Second Normal Form (2NF): 2NF eliminates redundant data by requiring that each non-key attribute be dependent on the primary key. This means that each column should be directly related to the primary key, and not to other columns.

Third Normal Form (3NF): 3NF builds on 2NF by requiring that all non-key attributes are independent of each other. This means that each column should be directly related to the primary key, and not to any other columns in the same table.

Boyce-Codd Normal Form (BCNF): BCNF is a stricter form of 3NF that ensures that each determinant in a table is a

candidate key. In other words, BCNF ensures that each non-key attribute is dependent only on the candidate key.

Fourth Normal Form (4NF): 4NF is a further refinement of BCNF that ensures that a table does not contain any multi- valued dependencies.

Fifth Normal Form (5NF): 5NF is the highest level of normalization and involves decomposing a table into smaller tables to remove data redundancy and improve data integrity.

Advantages of Normal Form

- Reduced data redundancy.
- Improved data consistency.
- Simplified database design.
- Improved query performance.
- Easier database maintenance

Shorts

1. Explain about relational model?

A) Relational Data Model - Introduction:

The relational Model is the conceptual basic Of Relational Databases. It is proposed by Dr.E-F codd. It is considered as one of the most Popular developments in the database technology. Because it can be used for representing most of the real world objects and the relationships between them. The relational model represents the database as a collection of relations (Tables).

Relational Model Concepts

The relational Model uses formal terms like. attribute, Tuple, relation, domain, schema.

Attribute:

An attribute can be defined as a characteristic of data. A real world data feature modelled in the database is represented by an attribute .For a person the attributes can be Name, Gender, Address, Age etc... Informally, a Column & field in a table..

Tuple:-

A Tuple represents a collection of information that describes a person, place, thing. Informal terms used for tuples are row in a table (or) record in in a data file.

Relation:-

A Relation is the core of the relational model. It is nothing but a table. (Date) in a relation, data is organized interms of rows and columns.

Domain:-

A Domain is the sets of possible values that are all of the same type for a given attribute.

3. Explain about i) candidate key, ii) primary key ,iii) foreign key.

A) **Candidate key** It plays an essential role in Database Management Systems (DBMS) by ensuring data integrity and efficient retrieval. A candidate key refers to a set of attributes that can uniquely identify each record in a table

- A candidate key is a super key that has the extra characteristic that it would become less unique if any of its attributes were removed.
- It's a simple yet powerful key.
- A table may have more than one candidate key.

- Every candidate key is distinct and has the potential to be the main key.

Std id	Roll no	name	Mob no	Email id
A1	1	Prasanna	123456789	a@gmail.com
A2	2	ammu	987654321	b@gmail.com

In this table, each student can uniquely identify by any of the following attribute: Student_id, Roll_no, Mobile_no, Email_id. So let primary key is Student_id and Candidate keys are Student_id, Roll_no, Mobile_no, Email_id.

Primary key: A primary key (PK) is a unique identifier for each record in a database table.

- **Unique:** No duplicate values allowed.
- **Not Null:** Cannot contain null values.
- **Unchangeable**: Once assigned, cannot be modified.
- Uniquely identifies each record.
- Ensures data integrity.
- Prevents data duplication.
- Supports relationships between tables.

Roll no	name	age	marks
1	Sai	25	48
2	Ajay	24	47
3	lokesh	23	46

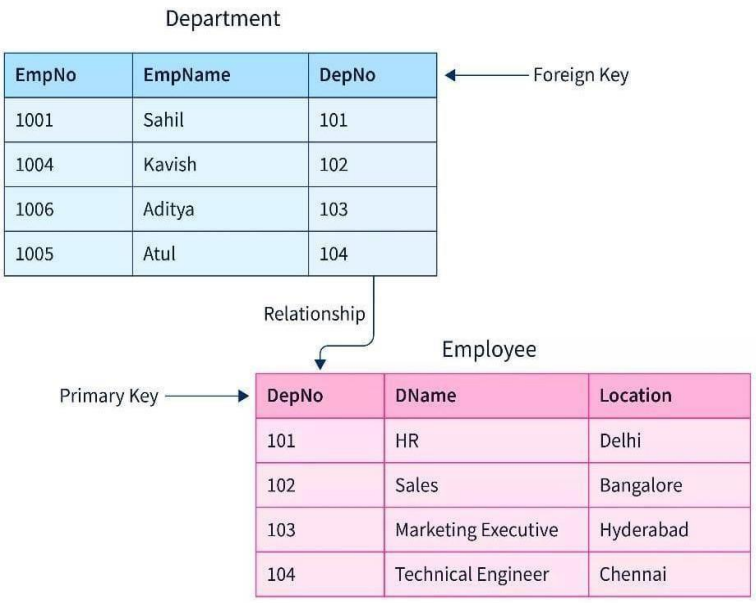
In the above eg we consider roll no as a primary key.

Foreign key

A foreign key (FK) is a field in a table that links to the primary key of another table.

- Establishes relationships between tables.
- Maintains data consistency.

- Supports data integrity.
- References the primary key of another table.
- Can be null or empty.
- Can have duplicate values.



4. Discuss about Relational calculus ?

A) Relational calculus is a formal system for querying and manipulating relational databases. It is based on mathematical logic and provides a declarative way of specifying database queries.

There are two main types of relational calculus:

1. **Tuple Relational Calculus (TRC):** This involves specifying a set of tuples (rows) that satisfy a given condition.

2. **Domain Relational Calculus (DRC):** This involves specifying a set of values for a given domain (column) that satisfy a given condition.

Relational calculus is used in various database management systems, including relational databases and SQL.

Some key concepts in relational calculus include:

1. **Relational variables:** Representing relations (tables) as variables.
2. **Range variables:** Representing individual tuples (rows) as variables.
3. **Conditions:** Specifying constraints on the data.
4. **Queries:** Specifying the desired output.

5. Explain about BCNF?

A) BCNF (Boyce-Codd Normal Form) is a database normalization technique that ensures a database table is structured in a way that minimizes data redundancy and dependency.

To be in BCNF, a table must meet the following conditions:

1. First Normal Form (1NF): Each cell must contain a single value.

2. Second Normal Form (2NF): Each non-key attribute must depend on the entire primary key.

3. Third Normal Form (3NF): If a table is in 2NF, and a non-key attribute depends on another non-key attribute, then it should be moved to a separate table.

Benefits of BCNF:

- Reduced data redundancy and inconsistency.
- Improved data integrity and accuracy.
- Simplified database maintenance and updates.
- Improved scalability and performance. Example:

Suppose we have a table "Orders" with attributes:

- OrderID (primary key)
- Product ID
- CustomerID
- Quantity
- OrderDate

UNIT-3

Unit 3: Entity Relationship Model: Introduction, Representation of entities, attributes, entity set, relationship, relationship set, constraints, sub classes, super class, inheritance, specialization, generalization using ER Diagrams,
BASIC SQL: Database schema, data types, DDL operations (create, alter, drop, rename), DML operations (insert, delete, update), basic SQL querying (select and project) using where clause, arithmetic & logical operations, aggregation, grouping, ordering.

Long answers:

1. Write about basic building blocks of entity Relation Ship diagram

Entity - Relationship Model (ER-Model) is a highlevel conceptual data model. The pictorial representation of ER-Model is called ER-diagram.

Basic Building blocks of ER-diagram (componants of ER-diagram)

They are:-

1. Enbity

2. Attribute

3. Relatio

nship

1.Entity:-

Types of

Entity

There are two types of entity

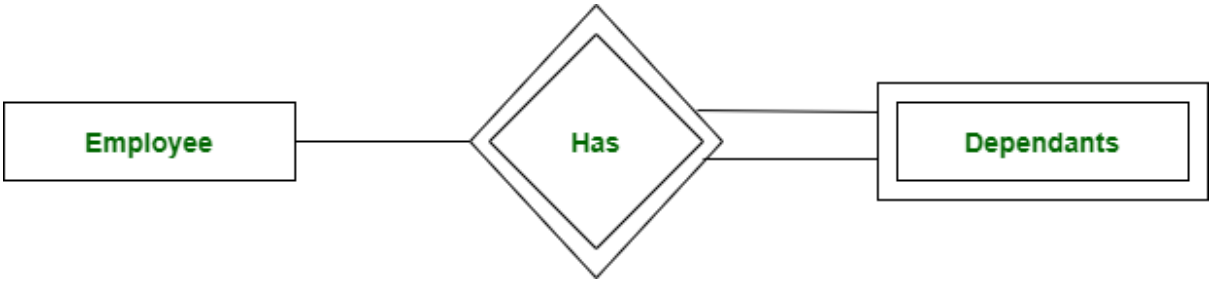
1. Strong Entity

A strong entity is a type of entity that has a key Attribute. Strong Entity does not depend on other Entity in the Schema. It has a primary key, that helps in identifying it uniquely, and it is represented by a rectangle. These are called Strong Entity Types.

2. Weak Entity

An Entity type has a key attribute that uniquely identifies each entity in the entity set. But some entity type exists for which key attributes can't be defined.

A weak entity type is represented by a Double Rectangle. The participation of weak entity types is always total. The relationship between the weak entity type and its identifying strong entity type is called identifying relationship and it is represented by a double diamond.



2. Attribute:-

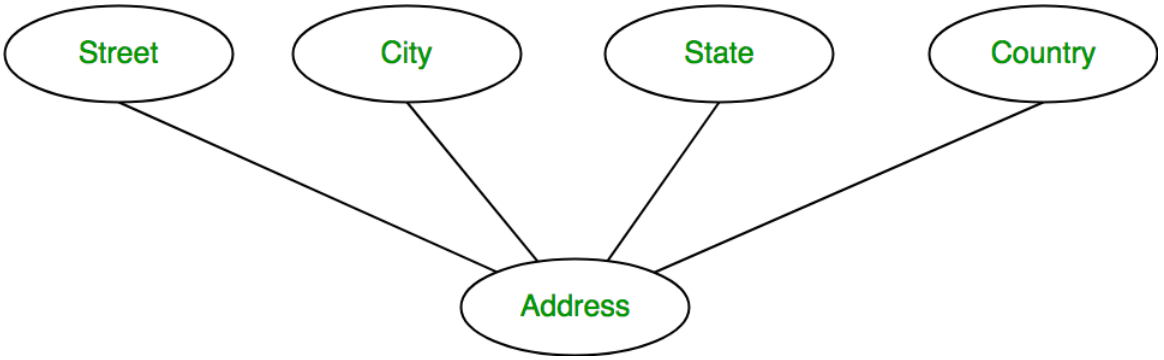
Attributes are the properties that define the entity type. For example, Roll_No, Name, DOB, Age, Address, and Mobile_No are the attributes that define entity type Student. In ER diagram, the attribute is represented by an oval.

Types of Attributes

1. **Key Attribute**
   The attribute which uniquely identifies each entity in the entity set is called the key attribute. For example, Roll_No will be unique for each student. In ER diagram, the key attribute is represented by an oval with underlying lines.

2. Composite Attribute

An attribute **composed of many other attributes** is called a composite attribute. For example, the Address attribute of the student Entity type consists of Street, City, State, and Country. In ER diagram, the composite attribute is represented by an oval comprising of oval



3. Multivalued Attribute

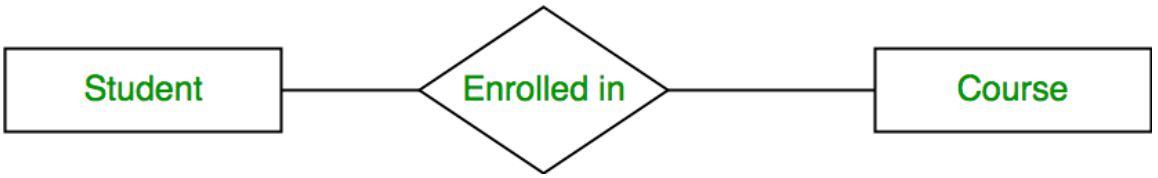
An attribute consisting of more than one value for a given entity. For example, Phone_No (can be more than one for a given student). In ER diagram, a multivalued attribute is represented by a double oval.

4. Derived Attribute

An attribute that can be derived from other attributes of the entity type is known as a derived attribute. e.g.; Age (can be derived from DOB). In ER diagram, the derived attribute is represented by a dashed oval.

3. Relation:-

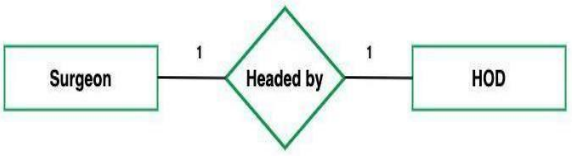
A Relationship Type represents the association between entity types. For example, 'Enrolled in' is a relationship type that exists between entity type Student and Course. In ER diagram, the relationship type is represented by a diamond and connecting the entities with lines.



Binary relationship and cardinality:-

The number of times an entity of an entity set participates in a relationship set is known as cardinality. Cardinality can be of different types:

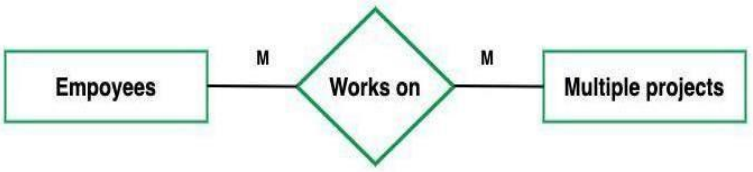
1. **One-to-One:** When each entity in each entity set can take part only once in the relationship, the cardinality is one-to-one. Let us assume that a male can marry one female and a female can marry one male. So the relationship will be one-



One-to-Many: In one-to-many mapping as well where each entity can be related to more than one entity and the total number of tables that can be used in this is 2. Let us assume that one surgeon department can accommodate many doctors. So the Cardinality will be 1 to M. It means one department has many Doctors.

Many-to-One: When entities in one entity set can take part only once in the relationship set and entities in other entity sets can take part more than once in the relationship set, cardinality is many to one. Let us assume that a student can take only one course but one course can be taken by many students. So the cardinality will be n to 1. It means that for one course there can be n students but for one student, there will be only one course.

Many-to-Many: When entities in all entity sets can take part more than once in the relationship cardinality is many to many. Let us assume that a student can take more than one course and one course can be taken by many students. So the relationship will be many to many.



2. Explain DDL and DML commands in SQL

SQL commands are extensively used to interact with databases, enabling users to perform a wide range of actions on database systems. Understanding these commands is crucial for effectively managing and manipulating data**.**

DDL (Data Definition Language)

DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database.

1. Create:-

Description: Create database or its objects (table, index, function, views, store procedure, and triggers)

Syntax: CREATE TABLE table_name (column1 data_type, column2 data_type, ...);

2. Drop:-

Description:- Delete objects from the database

Syntax: DROP TABLE table_name;

3. Alter:-

Description: Alter the structure of the database.

Syntax: **ALTER** TABLE table_name ADD COLUMN column_name data_type;

4. Rename:-

Description:- Rename an object existing in the database.

Syntax: **RENAME** TABLE old_table_name TO new_table_name;

DML (Data Manipulation Language)

The SQL commands that deal with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

1. Insert:-

Description:- Insert data into a table.

Syntax: **INSERT** INTO table_name (column1, column2, ...) VALUES (value1, value2, ...);

2. Update:-

Description:- Update existing data within a table.

Syntax: **UPDATE** table_name SET column1 = value1, column2 = value2 WHERE condition;

3. Delete:-

Description:- Delete records from a database table.

Syntax: **DELETE** FROM table_name WHERE condition;

3. Explain about set operators in SQL with examples

SQL Set Operation

The SQL Set operation is used to combine the two or more SQL SELECT statements.

Types of Set Operation

1. Union
2. Union All
3. Intersect
4. Minus

1. Union

- The SQL Union operation is used to combine the result of two or more SQL SELECT queries.
- In the union operation, all the number of datatype and columns must be same in both the tables on which UNION operation is being applied.
- The union operation eliminates the duplicate rows from its resultset.

Syntax

1. SELECT column_name FROM table1
2. UNION
3. SELECT column_name FROM table2;

Example:

The First table

ID	NAME
1	Jack
2	Harry

3	Jackson

The Second table

ID	NAME
3	Jackson
4	Stephan
5	David

Union SQL query will be:

```
1. SELECT * FROM First
2. UNION
3. SELECT * FROM Second;
```

The result set table will look like:

ID	NAME
1	Jack
2	Harry
3	Jackson
4	Stephan
5	David

2. Union All

Union All operation is equal to the Union operation. It returns the set without removing duplication and sorting the data.

Syntax:

```
1. SELECT column_name FROM table1
2. UNION ALL
3. SELECT column_name FROM table2;
```

Example: Using the above First and Second table**.**

Union All query will be like:

```
SELECT * FROM First
UNION ALL
SELECT * FROM Second;
```

The result set table will look like:

ID	NAME
1	Jack
2	Harry
3	Jackson
3	Jackson
4	Stephan
5	David

3. Intersect

- It is used to combine two SELECT statements. The Intersect operation returns the common rows from both the SELECT statements.
- In the Intersect operation, the number of datatype and columns must be the same.
- It has no duplicates and it arranges the data in ascending order by default.

Syntax:

```
SELECT column_name FROM table1
INTERSECT
SELECT column_name FROM table2;
```

Example:

Using the above First and Second table.

Intersect query will be:

```
SELECT * FROM First
INTERSECT
SELECT * FROM Second;
```

The result set table will look like:

ID	NAME
3	Jackson

4. Minus

- It combines the result of two SELECT statements. Minus operator is used to display the rows which are present in the first query but absent in the second query.
- It has no duplicates and data arranged in ascending order by default.

Syntax:

```
SELECT column_name FROM table1
MINUS
SELECT column_name FROM table2;
```

Example

Using the above First and Second table.

Minus query will be:

```
SELECT * FROM First
MINUS
SELECT * FROM Second;
```

The result set table will look like:

ID	NAME
1	Jack
2	Harry

4. Define query explain select statements with suitable examples

A query in SQL is a request for data or information from a database. The most common type of query is the SELECT statement, which retrieves data from one or more tables.

SELECT Statement

The SELECT statement allows you to specify which columns you want to retrieve from a database table, as well as filter, sort, and manipulate the data.

Basic Syntax

SELECT column1, column2, ... FROM table_name WHERE condition ORDER BY column;

Examples

1. Simple SELECT

Retrieve all columns from a table.

SELECT * FROM employees;

This query fetches all records and columns from the employees table.

2. Select Specific Columns

Retrieve specific columns from a table.

SELECT first_name, last_name FROM employees;

This query fetches only the first_name and last_name columns from the employees table.

1. Using the WHERE Clause

The WHERE clause is used to filter records that meet certain conditions.

Example:

SELECT * FROM employees

WHERE department = 'Sales';

This retrieves all records from the employees table where the department is 'Sales'.

2. Arithmetic Operations

You can perform arithmetic operations within your queries.

Example:

SELECT first_name, last_name, salary, salary * 1.1 AS updated_salary

FROM employees;

This retrieves employee names and calculates a 10% increase in their salary, displaying it as updated_salary.

Logical Operations

Logical operations (AND, OR, NOT) are used to combine multiple conditions in the WHERE clause.

Example:

SELECT * FROM employees

WHERE department = 'Sales' AND salary > 50000;

This retrieves employees from the Sales department with a salary greater than 50,000.

Using OR:

SELECT * FROM employees

WHERE department = 'Sales' OR department = 'Marketing';

This retrieves employees who work in either the Sales or Marketing departments.

Using NOT:

SELECT * FROM employees

WHERE NOT department = 'HR';

This retrieves all employees except those in the HR department.

4. Aggregation Functions

Aggregation functions are used to perform calculations on multiple rows of data, returning a single value.

Common Aggregation Functions:

- COUNT(): Counts the number of rows.
- SUM(): Sums up numeric values.
- AVG(): Calculates the average value.
- MIN(): Finds the minimum value.
- MAX(): Finds the maximum value.

Example:

SELECT COUNT(*) AS total_employees, AVG(salary) AS average_salary

FROM employees;

This retrieves the total number of employees and their average salary**.**

5. GROUP BY Clause

The GROUP BY clause groups rows sharing a property so that aggregate functions can be applied to each group.

Example:

SELECT department, COUNT(*) AS employee_count

FROM employees

GROUP BY department;

This retrieves the number of employees in each department.

6. HAVING Clause

The HAVING clause is used to filter records that work on aggregated data.

Example:

SELECT department, AVG(salary) AS average_salary

FROM employees

GROUP BY department

HAVING AVG(salary) > 60000;

This retrieves departments where the average salary is greater than 60,000.

7. ORDER BY Clause

The ORDER BY clause sorts the result set by one or more columns.

Example:

SELECT * FROM employees

ORDER BY last_name ASC;

This retrieves all employees and sorts them by last name in ascending order. Use DESC for descending order:

SELECT * FROM employees ORDER BY salary DESC;

Putting It All Together

Here's a comprehensive query that combines several concepts:

SELECT department, COUNT(*) AS employee_count, AVG(salary) AS average_salary FROM employees WHERE salary > 40000 GROUP BY department HAVING AVG(salary) > 60000 ORDER BY average_salary DESC;

SHORT ANSWERS

1. what is SQL? Explain about different data types in SQL

SQL (Structured Query Language) is a standardized programming language used to manage and manipulate relational databases. It enables users to perform various operations such as querying data, updating records, inserting new entries, and deleting existing ones.

Different Data Types in SQL

SQL data types can be broadly categorized into several groups, depending on the type of data they store. Here are some common categories and their respective data types:

1. Numeric Data Types

- **INT**: Stores integers (whole numbers). The size can vary based on the system (e.g., TINYINT, SMALLINT, MEDIUMINT, BIGINT).
- **FLOAT**: Stores floating-point numbers (decimals) with approximate precision.
- **DOUBLE**: Stores double-precision floating-point numbers for greater precision.
- **DECIMAL(p, s)**: Stores exact numeric values with a specified precision (p) and scale (s). Good for financial calculations.

2. Character and String Data Types

- **CHAR(n)**: Fixed-length string. If the input is shorter, it gets padded with spaces.
- **VARCHAR(n)**: Variable-length string. Stores only the characters inputted, up to n characters.
- **TEXT**: Stores large strings of text (variable length). Good for longer descriptions.

3. Date and Time Data Types

- **DATE**: Stores dates in the format YYYY-MM-DD.
- **TIME**: Stores time values in HH:MM:SS.
- **DATETIME**: Combines both date and time (e.g., YYYY-MM-DD HH:MM:SS).
- **TIMESTAMP**: Similar to DATETIME but also tracks time zone changes.

4. Boolean Data Type

- **BOOLEAN**: Stores TRUE or FALSE values. Often represented as 1 or 0.

5. Binary Data Types

- **BINARY(n)**: Fixed-length binary data.
- **VARBINARY(n)**: Variable-length binary data.
- **BLOB**: Used to store large binary objects.

2. Explain about aggregation and aggregate functions in sql

Aggregation in SQL refers to the process of summarizing or combining data from multiple rows into a single value. This is commonly used in data analysis to derive insights, generate reports, or perform calculations on groups of data. Aggregation functions are often used in conjunction with the GROUP BY clause to group rows that have the same values in specified columns.

Common Aggregation Functions

1. **COUNT()**: Counts the number of rows in a specified column or all rows if no column is specified.
    - Example: SELECT COUNT(*) FROM employees;

2. **SUM()**: Calculates the total sum of a numeric column.
    - Example: SELECT SUM(salary) FROM employees;
3. **AVG()**: Computes the average value of a numeric column.
    - Example: SELECT AVG(salary) FROM employees;
4. **MIN()**: Finds the minimum value in a specified column.
    - Example: SELECT MIN(salary) FROM employees;
5. **MAX()**: Finds the maximum value in a specified column.
    - Example: SELECT MAX(salary) FROM employees;

3. Write about Generalization and Specialization

Generalization and Specialization are two essential ideas used to describe the hierarchical connections between things in a database in the context of Enhanced Entity-Relationship (EER) diagrams.

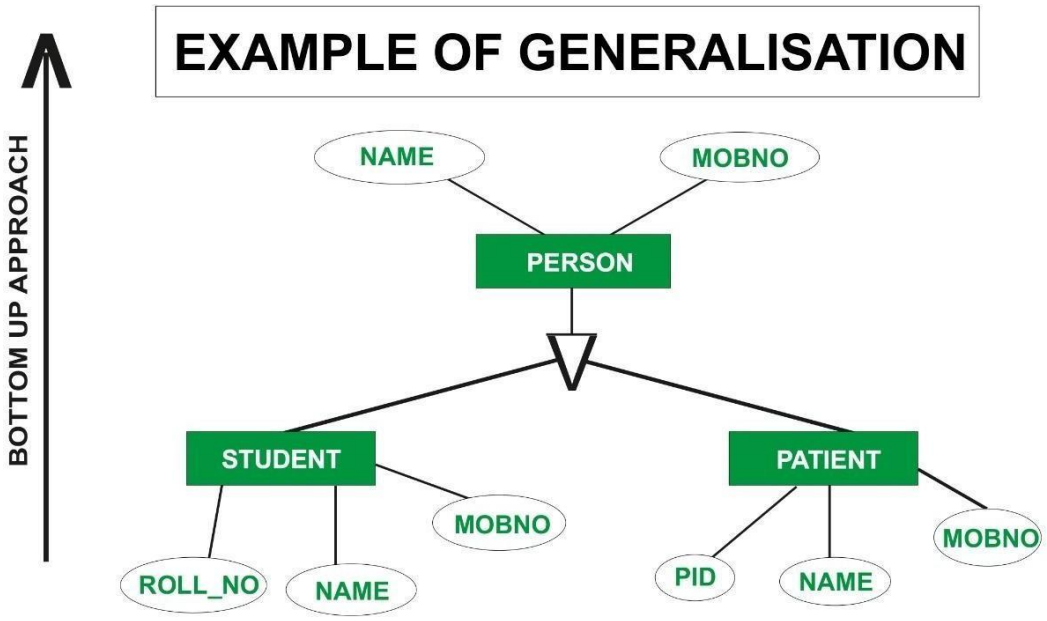
Generalization

Generalization is a bottom-up method used to combine lower-level entities into a higher-level object. This approach creates a more generic entity, known as a superclass, by combining entities with similar features. By removing duplication and arranging the data in a more organized manner, generalization streamlines the data model.

Example of Generalization

Consider two entities Student and Patient. These two entities will have some characteristics of their own. For example, the Student entity will have Roll_No, Name, and Mob_No while the patient will have PId, Name, and Mob_No characteristics.

Now in this example Name and Mob_No of both Student and Patient can be combined as a Person to form one higher-level entity and this process is called generalization process.



Specialization

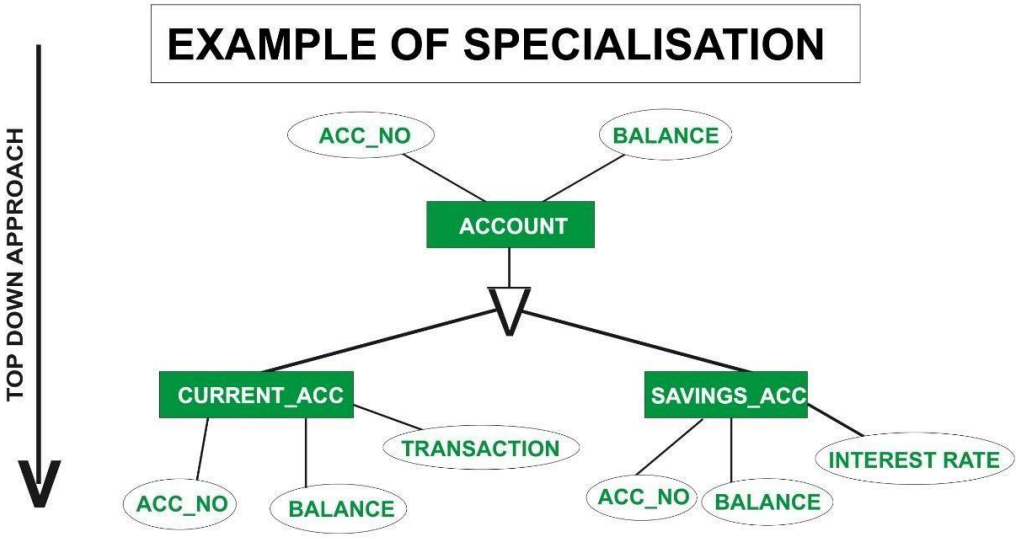
In EER diagrams, specialization is a top-down method where a higher-level entity is split into two or more lower-level entities according to their distinct qualities.

This technique, which includes splitting a single entity set into subgroups, is often connected to inheritance, in which attributes from the higher-level entity are passed down to the lower-level entities.

Example of Specialization

Consider an entity Account. This will have some attributes consider them Acc_No and Balance.

Account entity may have some other attributes like Current_Acc and Savings_Acc. Now Current_Acc may have Acc_No, Balance and Transactions while Savings_Acc may have Acc_No, Balance and Interest_Rate henceforth we can say that specialized entities inherits characteristics of higher level entity.


EXAMPLE OF SPECIALISATION
TOP DOWN APPROACH
ACC_NO
BALANCE
ACCOUNT
CURRENT_ACC
SAVINGS_ACC
TRANSACTION
INTEREST RATE
ACC_NO
BALANCE
ACC_NO
BALANCE

UNIT-IV

Unit 4: SQL: Nested queries/ sub queries, implementation of different types of joins, SQL functions(Date, Numeric, String, Conversion functions), Creating tables with relationship, implementation of key and integrity constraints, views, relational set operations , Transaction Control Language: commit, Rollback, Savepoint , DCL :Grant, Revoke

1. Explain Different Types of Joins in SQL with Real Examples

Joins in SQL allow you to retrieve data from two or more tables based on a related column between them. SQL provides several types of joins, including:

1. INNER JOIN

An **INNER JOIN** returns only the rows where there is a match in both tables. It combines records from both tables as long as a related match exists.

Example: Consider the following tables:

Customers Table:

CustID	CustomerName	City
1	Alice	New York
2	Bob	Los Angeles
3	Charlie	Chicago

Orders Table:

OrderID	CustID	OrderAmount
101	1	250
102	2	500
103	4	100

Query:

```
SELECT Customers.CustomerName, Orders.OrderAmount FROM Customers INNER JOIN
Orders ON Customers.CustID = Orders.CustID;
```

Result:

CustomerName	OrderAmount
Alice	250
Bob	500

This result includes only those customers who have placed orders (Alice and Bob).

2. LEFT JOIN

A **LEFT JOIN** returns all rows from the left table (Customers), even if there are no matches in the right table (Orders). Where no match is found, NULL values are shown.

Query:

```
SELECT Customers.CustomerName, Orders.OrderAmount FROM Customers LEFT JOIN
Orders ON Customers.CustID = Orders.CustID;
```

Result:

CustomerName	OrderAmount
Alice	250
Bob	500
Charlie	NULL

Charlie is included in the result even though they have not placed any orders.

3. RIGHT JOIN

A **RIGHT JOIN** is the opposite of a LEFT JOIN. It returns all rows from the right table (Orders), and matching rows from the left table. If no match is found, NULL is returned.
Query:

SELECT Customers.CustomerName, Orders.OrderAmount FROM Customers RIGHT JOIN Orders ON Customers.CustID = Orders.CustID;

Result:

CustomerName	OrderAmount
Alice	250
Bob	500
NULL	100

This query includes orders placed by customers who aren't listed in the **Customers** table.

4. FULL OUTER JOIN

A **FULL OUTER JOIN** returns all rows when there is a match in either the left or right table. It includes rows with no match in one table, using NULLs to fill in the gaps.

Query:

SELECT Customers.CustomerName, Orders.OrderAmount FROM Customers FULL OUTER JOIN Orders ON Customers.CustID = Orders.CustID;

Result:

CustomerName	OrderAmount
Alice	250
Bob	500
Charlie	NULL
NULL	100

This query returns all rows from both tables, even those without a match in the other table.

2. Write About String Functions in SQL with Real Examples

SQL provides several **string functions** to manipulate text data. These functions are useful for operations like concatenating, trimming, or changing case. Let's explore some common string functions:

1. CONCAT()

The `CONCAT()` function concatenates two or more strings into one.

Example:

SELECT CONCAT('First Name: ', 'Alice', ', Last Name: ', 'Johnson') AS Full Name;

Result:

Full Name: First Name: Alice, Last Name: Johnson

2. ASCII(): This function is used to find the ASCII value of a character.

Syntax: SELECT ascii('t');

Output: 116

3. CONCAT_WS(): This function is used to add two words or strings with a symbol as concatenating symbol.

Syntax: SELECT CONCAT_WS('_', 'geeks', 'for', 'geeks');

Output: geeks_for_geeks

4. **FORMAT():** This function is used to display a number in the given format.

Syntax: Format("0.981", "Percent");

Output: '98.10%'

5. **INSERT():** This function is used to insert the data into a database.

Syntax: insert into table name values(emp_id,emp_name,5000,'abc');

Output: successfully updated

6. **LENGTH():** This function is used to find the length of a word.

Syntax: length('hello');

Output: 5

7. **LOWER():** This function is used to convert the upper case string into lower case.

Syntax: select lower('HELLO WORLD');

Output: hello world

8. **UPPER():** This function is used to convert the lower case string into upper case.

Syntax: select upper('hello world');

Output: HELLO WORLD

9. **REPEAT():** This function is used to write the given string again and again till the number of times mentioned.

Syntax: SELECT REPEAT('hello', 2);

Output: hellohello

10. **REVERSE():** This function is used to reverse a string.

Syntax: select reverse('hello world');

Output:'dlrow olleh'

3. Discuss About TCL Commands in SQL

TCL (Transaction Control Language) commands in SQL manage transactions within a database. They ensure that groups of operations are executed as a single unit, providing **atomicity, consistency, isolation, and durability (ACID)** properties to transactions.

The main TCL commands are:

1. COMMIT

The `COMMIT` command is used to save all changes made during the transaction. Once committed, changes are permanent and cannot be undone.

Example:

START TRANSACTION; UPDATE Employees SET Salary = Salary + 1000 WHERE EmpID = 101; COMMIT;

This example increases the salary of an employee and commits the change.

2. ROLLBACK

The `ROLLBACK` command undoes all changes made in the current transaction if something goes wrong.

Example:

START TRANSACTION; DELETE FROM Orders WHERE OrderID = 202; ROLLBACK;

In this case, the `DELETE` operation is undone by the `ROLLBACK`, so no changes are made.

3. SAVEPOINT

The `SAVEPOINT` command creates a point within a transaction to which you can roll back without affecting the entire transaction.

Example:

START TRANSACTION; UPDATE Products SET Price = Price * 1.1; SAVEPOINT PriceUpdate; DELETE FROM Orders WHERE OrderID = 303; ROLLBACK TO PriceUpdate;

This transaction only rolls back the delete operation, leaving the price update intact.

4. RELEASE SAVEPOINT
The `RELEASE SAVEPOINT` command removes a savepoint, preventing any rollback to that point.

4. How to Create Views in SQL
Views in SQL are virtual tables based on the result of a SELECT query. They are used to simplify complex queries, improve security, and present data in a specific way without storing it physically.
Creating a View
You can create a view using the CREATE VIEW statement. The view is a named query that you can reference just like a table.
Example: Consider the following tables in a **university database**:

Students Table:		
StudentID	**Name**	**Major**
101	Alice	CS
102	Bob	Math

Marks Table:		
StudentID	**Course**	**Marks**
101	CS101	85
102	Math101	90

If you frequently need to see the names and marks of students, you can create a view:
Query:
CREATE VIEW StudentMarks AS SELECT Students.Name, Marks.Course, Marks.Marks FROM Students INNER JOIN Marks ON Students.StudentID = Marks.StudentID;
This view simplifies querying student marks without having to repeatedly write the join operation.
Using a View
Once the view is created, you can query it like a table.
Example:
SELECT * FROM StudentMarks;
Result:

Name	**Course**	**Marks**
Alice	CS101	85
Bob	Math101	90

1. Write About Nested Query in SQL
A **Nested Query** (or subquery) is a query within another SQL query. It is often used to retrieve data that will be used in the main query. Subqueries are enclosed in parentheses and can be placed in the SELECT, FROM, WHERE, or HAVING clauses of an SQL query.
Types of Nested Queries
- **Single-row subquery**: Returns a single value.

- **Multi-row subquery**: Returns multiple values.
- **Correlated subquery**: The subquery references a column from the outer query.

Example 1: Single-row Subquery

Find the details of employees who earn more than the average salary.

Query:

SELECT EmpName, Salary FROM Employees WHERE Salary > (SELECT AVG(Salary) FROM Employees);

In this example, the subquery (SELECT AVG(Salary) FROM Employees) calculates the average salary, and the main query retrieves employees earning more than that value.

Example 2: Multi-row Subquery

List all students enrolled in any of the courses offered by a specific instructor.

Query:

SELECT StudentName FROM Students WHERE CourseID IN (SELECT CourseID FROM Courses WHERE Instructor = 'Dr. Smith');

Here, the nested query finds the courses taught by Dr. Smith, and the outer query retrieves students enrolled in those courses.

2. Explain About Date Functions in SQL

Date Functions in SQL are used to manipulate and format date and time values. These functions are crucial for performing operations on date fields like calculating intervals, extracting parts of dates, and formatting output.

Common SQL Date Functions

- **NOW()**: Returns the current date and time.
- **CURDATE()**: Returns the current date.
- **DATEADD()**: Adds a specific time interval to a date.
- **DATEDIFF()**: Returns the difference between two dates.
- **EXTRACT()**: Extracts part of a date (e.g., year, month, day).

Example 1: NOW()

SELECT NOW() AS CurrentDateTime;

Result:

CurrentDateTime: 2024-10-21 10:30:45

Example 2: DATEADD()

Add 30 days to the current date.

Query:

SELECT DATEADD(CURDATE(), INTERVAL 30 DAY) AS NewDate;

Result:

NewDate: 2024-11-20

Example 3: DATEDIFF()

Find the difference in days between two dates.

Query:

SELECT DATEDIFF('2024-12-01', '2024-10-21') AS DaysDifference;

Result:

DaysDifference: 41

3. Write About DCL Commands in SQL

DCL (Data Control Language) commands in SQL are used to control access to data within the database. These commands help in managing permissions and access levels for users.

Common DCL Commands

- **GRANT**: Grants access or privileges to users.
- **REVOKE**: Removes access or privileges from users.

1. GRANT Command

The GRANT command allows a database administrator to give specific permissions to users.

Example:

GRANT SELECT, INSERT ON Employees TO 'john'@'localhost';

This command gives the user john the ability to run SELECT and INSERT queries on the **Employees** table.

2. REVOKE Command

The REVOKE command removes previously granted permissions from a user.

Example:

REVOKE INSERT ON Employees FROM 'john'@'localhost';

This command revokes the INSERT privilege from the user john on the **Employees** table.

DCL commands are used in conjunction with security policies to control how different users interact with the data in the database.

Unit-5

Unit 5: PL/SQL: Introduction, Structure , Control Structures , Cursors , Procedure , Function , Packages , Exception Handling ,Triggers.

Transaction processing Concepts : Transaction State, Implementation of Atomicity and Durability, Concurrent Executions, Serializability, Recoverability, Implementation of Isolation, Testing for Serializability, Failure Classification, Storage, Recovery and Atomicity, Recovery algorithm.

3. Discuss about cursors in PL/SQL

PL/SQL Cursors

The cursor is used to retrieve data one row at a time from the results set, unlike other SQL commands that operate on all rows at once.

Cursors update table records in a singleton or row-by-row manner.

The Data that is stored in the Cursor is called the Active Data Set.

Oracle.

cursor Actions

Key actions involved in working with cursors in PL/SQL are:

1. **Declare Cursor:** A cursor is declared by defining the SQL statement that returns a result set.
2. **Open:** A Cursor is opened and populated by executing the SQL statement defined by the cursor.
3. **Fetch:** When the cursor is opened, rows can be fetched from the cursor one by one or in a block to perform data manipulation.
4. **Close:** After data manipulation, close the cursor explicitly.
5. **Deallocate:** Finally, delete the cursor definition and release all the system resources associated with the cursor.

Types of Cursors in PL/SQL

Cursors are classified depending on the circumstances in which they are opened.

- **Implicit Cursor:** If the Oracle engine opened a cursor for its internal processing it is known as an Implicit Cursor. It is

created “automatically” for the user by Oracle when a query is executed and is simpler to code.

- **Explicit Cursor:** A Cursor can also be opened for processing data through a PL/SQL block, on demand. Such a user-defined cursor is known as an Explicit Cursor.

Implicit Cursors:

- These are automatically created by Oracle when an SQL statement (such as SELECT INTO, INSERT, UPDATE, or DELETE) is executed.
- PL/SQL handles opening, fetching, and closing of implicit cursors behind the scenes.
- Implicit cursors are used when a query returns only one row. If more than one row is returned, it raises a TOO_MANY_ROWS exception.

```
DECLARE
  v_salary NUMBER;
BEGIN
  -- Implicit cursor
  SELECT salary INTO v_salary FROM employees WHERE
employee_id = 100;
  DBMS_OUTPUT.PUT_LINE('Salary: ' || v_salary);
END;
```

Explicit Cursors:

- These are defined by the programmer to handle queries that return more than one row.
- With explicit cursors, you control the opening, fetching, and closing of the cursor explicitly.
- Explicit cursors are useful when you need to iterate through multiple rows of a result set.

Syntax for Explicit Cursor:

```
CURSOR cursor_name IS
   SELECT column1, column2, ... FROM table WHERE condition;
```

4. write about serializability in transaction processing

Serializability in transaction processing is a fundamental concept in database systems that ensures **concurrent transactions** are executed in a way that their final result is equivalent to some **serial execution** of those transactions.

Serializability is Important:

In a multi-user database environment, multiple transactions often execute concurrently to maximize system efficiency and utilization. However, without proper control, concurrent transactions can lead to issues like:

1. **Dirty Reads**: A transaction reads data that is written by another transaction but not yet committed.
2. **Non-repeatable Reads**: A transaction reads the same row twice and gets different data each time due to modifications by another transaction.
3. **Phantom Reads**: A transaction re-executes a query and sees a different set of rows because another transaction inserted or deleted rows.
4. **Lost Updates**: Multiple transactions update the same data, and the result of one update overwrites the other.

Types of Serializability:

1. **Conflict Serializability**:
    - This is the most widely used and practical form of serializability.
    - It is based on **conflicting operations** between transactions. Two operations conflict if:
        - They belong to different transactions.
        - They access the same data item.
        - At least one of them is a **write** operation.
    - If the conflicting operations between transactions can be reordered in a way that is equivalent to a serial execution, the schedule is **conflict-serializable**.

Example: Consider two transactions T1 and T2 that are

concurrently accessing the same data item X.

- T1: Read(X), Write(X)
- T2: Read(X), Write(X)

If the system can reorder the operations of T1 and T2 such that the result is the same as if T1 was executed before T2 (or vice versa), then the schedule is conflict-serializable.

2. View Serializability:

- This is a more relaxed and broader form of serializability than conflict serializability.
- A schedule is **view-serializable** if it produces the same results as some serial schedule but does not necessarily involve reordering conflicting operations.
- It is more difficult to enforce and less commonly used in practice.

5. Explain about triggers in PL/SQL

In PL/SQL, triggers are special types of stored procedures that are automatically executed (or "triggered") when specific events occur in the database. Triggers are primarily used to enforce business rules, maintain data integrity, and automate system responses to changes in the data.

Types of Triggers:

1. DML Triggers:

DML triggers are fired in response to DML operations (INSERT, UPDATE, DELETE) on a table or view.

- **Before Triggers**: Fired before the DML statement is executed.
- **After Triggers**: Fired after the DML statement is executed.
- **Row-level Triggers**: Fired for each row affected by the DML statement.
- **Statement-level Triggers**: Fired once for the entire statement, regardless of how many rows are affected.

Syntax for DML Trigger:

```
CREATE [OR REPLACE] TRIGGER trigger_name
  {BEFORE | AFTER} {INSERT | UPDATE | DELETE}
  ON table_name
  [FOR EACH ROW]
BEGIN
  -- Trigger logic
END;
```

2. INSTEAD OF Triggers:

These triggers are defined on **views** and are used to perform actions that would normally be impossible due to the complexity of the view. For example, INSTEAD OF triggers are useful for making a non-updatable view appear updatable.

Syntax for INSTEAD OF Trigger:

```
CREATE [OR REPLACE] TRIGGER trigger_name
  INSTEAD OF {INSERT | UPDATE | DELETE}
  ON view_name
BEGIN
  -- Trigger logic
END;
```

3. DDL Triggers:

DDL triggers are fired in response to DDL events like CREATE, ALTER, DROP, GRANT, or REVOKE. These triggers are typically used for auditing purposes or to enforce rules on schema changes.

Syntax for DDL Trigger:

```
CREATE [OR REPLACE] TRIGGER trigger_name
  {BEFORE | AFTER} {CREATE | ALTER | DROP |
GRANT | REVOKE}
  ON {SCHEMA | DATABASE}
BEGIN
  -- Trigger logic
END;
```

4. Compound Triggers:

A compound trigger is a type of trigger that can combine multiple timing points into a single trigger body. It is useful for managing shared data across the various timing points of a trigger (BEFORE STATEMENT, BEFORE EACH ROW, AFTER EACH ROW, AFTER STATEMENT).

6. What is transaction processing

Transaction processing refers to the execution of a series of operations on a database, which are grouped into a single, logical unit of work called a transaction. In the context of databases, a transaction represents a sequence of one or more SQL operations (like INSERT, UPDATE, DELETE, or SELECT) that must be executed as a single, indivisible unit.

Key Concepts in Transaction Processing:

1. **Atomicity**:
    - Ensures that all the operations within a transaction are executed completely or not at all. If any part of the transaction fails, the entire transaction is rolled back, leaving the database unchanged.
    - For example, if a money transfer between two bank accounts fails halfway through (perhaps the amount was debited from one account but not credited to the other), the system rolls back the transaction, ensuring neither account is affected.
2. **Consistency**:
    - Ensures that a transaction transforms the database from one **consistent state** to another. It means that the integrity constraints of the database (like foreign keys, unique constraints, and business rules) are maintained before and after the transaction.
    - For instance, after a transaction that updates an employee's salary, the employee data should still conform to all salary constraints defined in the

system.

3. **Isolation**:
    - Ensures that concurrent transactions do not interfere with each other. Even when transactions are executed concurrently, it should appear as if they were executed in serial order (one after the other), avoiding conflicts and anomalies.
    - Isolation prevents issues like dirty reads, lost updates, and uncommitted data visibility.
4. **Durability**:
    - Guarantees that once a transaction has been committed, its effects are permanently applied to the database, even in the event of a system failure (like a power outage or crash). Durability is typically ensured by writing transaction logs to stable storage.
    - After a transaction commits, any changes made to the database should persist and be recoverable even if the system goes down.

4. PL/SQL UNIT-4

Essay :-

1. What is PL/SQL ? write about structure of PL/SQL with example ?

A. PL/SQL is a block structured language that enables developers to combine the power of SQL with procedural statements.

Advantages of PL/SQL :-

The major advantages are :-

1. Support for SQL
2. Block structure
3. Better performance
4. Modularity
5. Control structures
6. Error Handling.

PL/SQL Block structure :-

The structure of PL/SQL block is as follows :

```
DECLARE
        <declarations>
BEGIN
```

< Executable Statement >

EXCEPTION

< Exception handle >

END;

A PL/SQL block consists of upto three sections.

1. DECLARE : (Declarative Section)

It starts with the "DECLARE" keyword. It is used to declare any place holders like Variable Constants, records and cursors which stores data temporarily. This section is optional.

2. EXECUTABLE Section :

This section starts with "BEGIN" keyword and ends with "END".

It is the section where program logic is written to perform any task. The programmatic constructs like loops, conditional statements and SQL statements form the part of executable section.

This section is mandatory.

3. EXCEPTION Handling :

This section starts with the keyword "EXCEPTION". Any error in the program can be handled in this section, so that PL/SQL block terminates gracefully. This section is optional.

Example :-

```
Declare
        a Number(2) := &a;
        b Number(2) := &b;
        c Number(3);
Begin
        c := a+b;
        dbms_output.put_line('sum of a&b is'||c
END;
/
```

Output :-

```
Enter value for a: 2
Enter value for b: 3
Sum of a&b is '5'
```

Essay :-

2. Write about WHILE Loop used in PL/SQL with example.

A. Loop control structures are used when we want to repeat the execution of one or more statements for specified number of times. PL/SQL provides the following types of loops :

a] Basic / Simple Loop

b] while Loop

c] For loop.

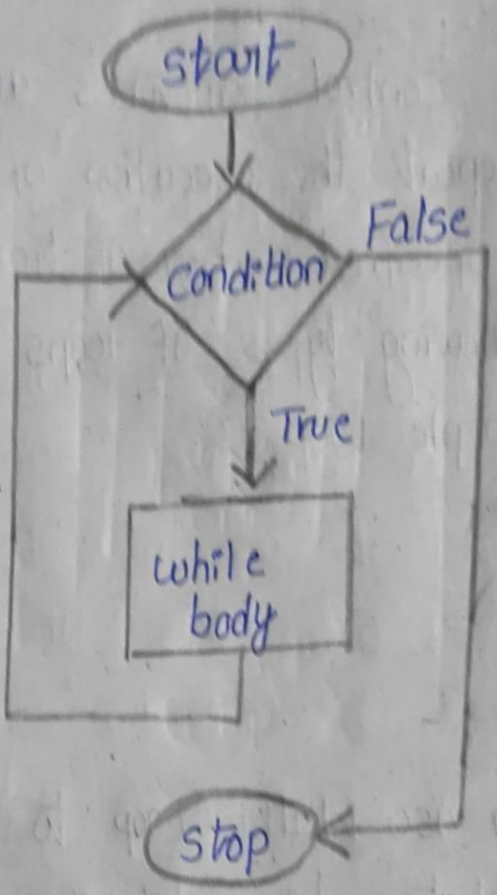
WHILE Loop :-

We can use WHILE Loop to repeat a sequence of statements until the controlling condition is TRUE. The condition is evaluated at the start of each iteration. The loop terminates when the condition is false. If the condition is FALSE at the start of the loop, then no further iterations are performed.

Syntax :-

```
WHILE condition
Loop
statements;
- - - - -
END Loop;
```

Flow chart



Example :-

```
DECLARE
    a NUMBER:=0;
BEGIN
    WHILE (a<4)
    Loop
     a:=a+1;
    DBMS_OUTPUT.PUT_LINE(a);
    END Loop;
END;
/
```

Output :- It prints numbers from 1 to 4.

3. Write about FOR loop used in PL/SQL.

Loop control structures are used when we want to repeat the execution of one or more statements for specified number of times. PL/SQL provides the following types of loops:

a] Basic / simple loop

b] while loop

c] For loop

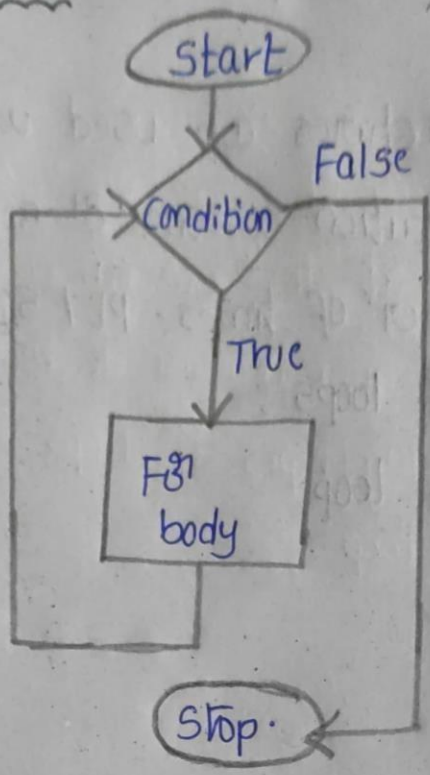
FOR LOOP :-

FOR loop is an iterative statement that allows us to execute a sequence of statements a fixed number of times. Unlike the PL/SQL WHILE loop, the no. of iterations of the PL/SQL FOR loop is known before the loop starts.

Syntax :-

```
FOR counter IN [REVERSE]
        <lowerbound>..<upperbound>
LOOP
 statements;
 ------
 END LOOP;
```

Flow chart :-



Example :-

```
DECLARE
    a number := 0;
BEGIN
    FOR I IN 1..4
    Loop
      a := a+1;
      DBMS_OUTPUT.PUT_LINE (a);
    END LOOP;
END;
/
```

Output :- It prints numbers from 1 to 4

Short :-

1. Explain about simple 'If statement' in PL/SQL?

A: Decision-making statements in programming languages decide the direction of flow of program execution. One of them available in PL/SQL is Simple If. (if then statement).

if then statement : (simple if)

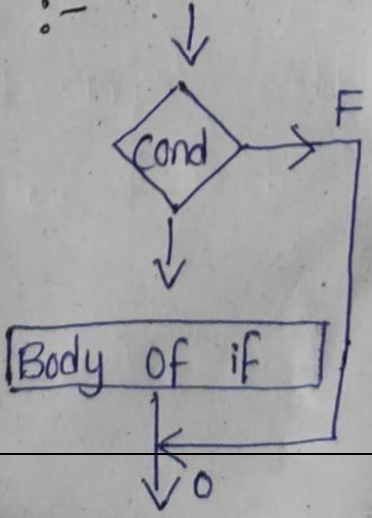
It is the most simple decision making statement.

Syntax :-

```
if condition then
    -- do something ;
end if ;
```

if the condition is true they it will execute the block of statements below it otherwise not.

Flow graph :-

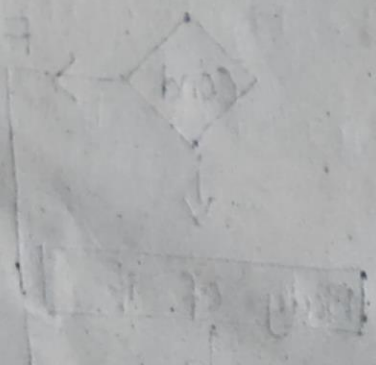


Example :-

```
Declare
        A  Number;
        B  Number;
Begin
        A:=10;
        B:=5;
        If A>B then
        dbms_output.put_Line('A is bigger than B');
        end if;
End;
/
```

Output :-

A is bigger than B.



2. Explain about Basic Loop structure in PL/SQL?

A. **Basic/Simple Loop :-**

It encloses a sequence of statements between keywords Loop and END LOOP. Each time the flow of execution reaches the END Loop statement, control is returned to the corresponding Loop above it. A basic loop allows execution of its statement at least once. without the EXIT statement, ~~at least once. Without~~ the loop would be infinite or endless. we can place one or more EXIT statements anywhere inside a loop, but not outside a loop.

Syntax :-

```
Loop
statements ;
- - - - - - -
- - - - - - -
EXIT [when condition] ;
END LOOP ;
```

Example :-

```
DECLARE
   a Number := 0 ;
BEGIN
   Loop
```

```
        a := a+1;
        dbms_output.put_line(a);
        If (a=5) Then
        EXIT;
        END If;
        END LOOP;
END;
/
```

Output :-

Prints 1, 2, 3, 4 Numbers only

[If prints the numbers from 1 to 4 only].