II BSC – SEM 3 - DATABASE MANAGEMENT SYSTEM

Unit-I

Overview of Database Management System: Introduction to data, information, database, database management systems, file-based system, Drawbacks of file-Based System, database approach, Classification of Database Management Systems, advantages of database approach, Various Data Models, Components of Database Management System, three schema architectureof data base, costs and risks of database approach

Unit-II

Entity-Relationship Model: Introduction, the building blocks of an entity relationship diagram, classification of entity sets, attribute classification, relationship degree, relationship classification, reducing ER diagram to tables, enhanced entity-relationship model (EER model), generalization and specialization.

Unit-III

Relational Model: Introduction, CODD Rules, relational data model, concept of key, relational integrity, relational algebra, relational algebra operations, advantages of relational algebra, limitations of relational algebra, Functional dependencies and normal forms up to 3rd normal form.

Unit-IV

Structured Query Language: Introduction, History of SQL Standard, Commands in SQL, Data Types in SQL, Data Definition Language, Selection Operation, Projection Operation, Aggregate functions, Data Manipulation Language, Table Modification Commands, Join Operation, Set Operations, View, Sub Query.

Unit –V

PL/SQL: Introduction, Shortcomings of SQL, Structure of PL/SQL, PL/SQL Language Elements, Data Types, Operators Precedence, Control Structure, Steps to Create a PL/SQL, Program, Iterative Control, Procedure, Function, Database Triggers, Types of Triggers.

UNIT – I (INTRODUCATION TO DBMS)

[Q]. Discuss about Data and Information?

DATA AND INFORMATION:

Data:

- Data may be defined as collection of any raw facts, Figures, Observations, assumptions oroccurrences in the real world.
- Data may be represented using numbers and letters or combination of both.
- Data will always be in raw shape.
- Data is unorganized facts and figures.
- It can be generated or prepared from various departments and other sources.
- Data individually carries less meaning.
- Data individually may not be useful for decision making.
- It sometimes as information.
- **Example**: Collection of the applicant, address, age, etc. are facts related to people, these factsare represented by using Alphabets, Numbers or combination both.

Information:

- The processed data is called as Information.
- Information may be defined as processed data.
- Information is more refined in its shape.
- Information is more meaningful when compared with data.
- Information can be used for decision making.
- Information may also be defined as organized or classified data.
- Information can be distributed to various departments and people in the organization.
- It sometimes appears as data.
- **Example**: If we collect Marks obtained by 75 students in Computers subject --- This is **data**. Whereas if we prepare a list of highest or lowest marks of the same ---This is **Information**

[Q]. Write down the difference between Data & Information?

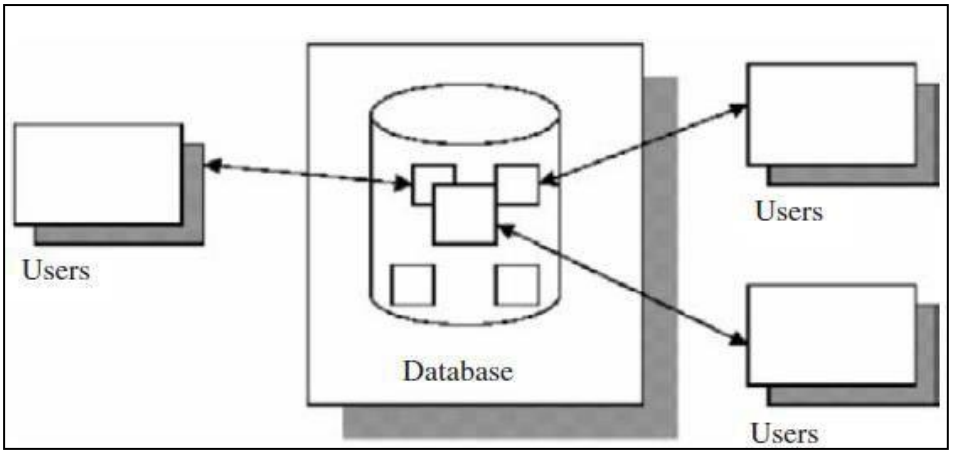
DATA	INFORMATION
• Collection of raw facts is called as data.(i.e., anything can be data) • It is not significant to business • It is in unorganized form • It does not help in decision making • **Example:** any number, text, speech, voice, etc.	• The processed data is called as Information • It is significant to business • It is in organized form • It help in decision making • **Example:** student report card sheet

[Q]. Define Database. Give some examples?

DATABASE :

- The **database** is organized collection of related information. (or) The collection of related data is called as Database. It is used for Retrieving, processing and reorganizing the desired information.

- The simplified view of database system is shown below…



- **Database examples in our daily life:**
  - The dictionary,
  - The telephone directory,
  - The electricity board,
  - The Class room register,
  - Colleges and Universities
  - Files on your computer hard drive
  - Employee details in a company
  - TV guide
  - Airline reservation system
  - Motor vehicle registration system

[Q]. Define Database Management System. And give some examples?

DATABASE MANAGEMENT SYSTEM (DBMS) :

- It is a collection of programs working together to control Data Manipulations (add,change, remove),Retrievals (Read) and sharing on Database.

(OR)

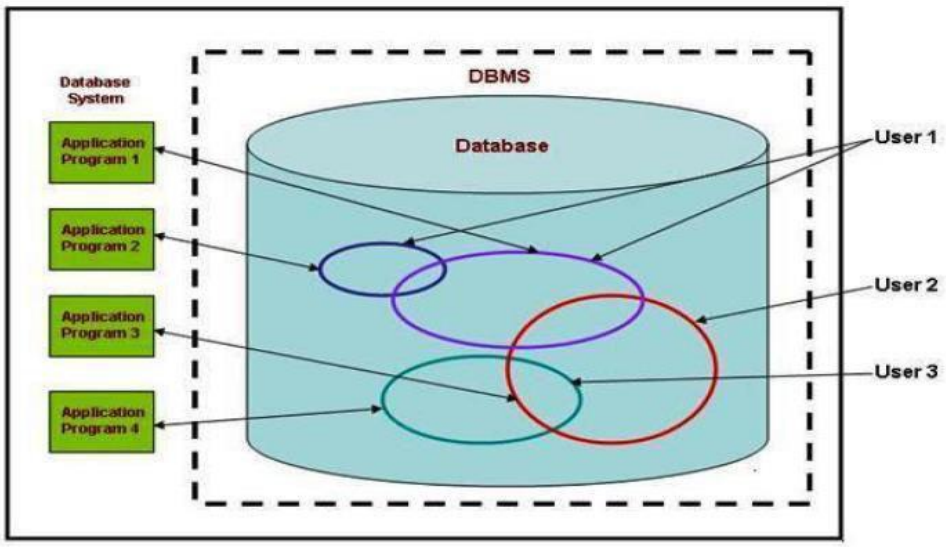
- It is program or software it manages the data in the database based on the request from user.

(OR)

- A Database management system (DBMS) is a collection of programs that allows users to createand maintain a database.

(OR)

- A Database Management System (DBMS) is a collection of programs that manages the databasestructure and controls access to the data stored in the database.
- The DBMS serves as the intermediary between the user and database. The database structure itself is stored as a collection of files and the only way to access the data in those files is through the DBMS. It presents the end user or application program with single, integrated view of data in the database.
- **Examples:** Oracle RDBMS, IBM DB2, Microsoft SQL Server



[Q]. Explain the Functions of Database Management System?

FUNCTIONS (OR) OBJECTIVES (OR) SERVICES OF DBMS:

- **Database Management System:** A Database management system (DBMS) is a collection of programs that allows users to create and maintain a database. It acts as an intermediator between user and database.
- A DBMS performs several functions that provide the integrity and consistency of data in the database. DBMS performs following important functions within the database.

- **Data Dictionary Management :**
  - The DBMS maintains the definition of data elements in a data dictionary. If any changes in database is automatically recorded in the data dictionary.
- **Data storage Management :**
  - DBMS creates the structure for database in the physical storage devices. It provides amechanism for permanent storage of data.
- **Data Definition Management :**
  - The DBMS creates the structure of data to store in which the data is stored.
- **Data Manipulation Management :**
  - The DBMS provides ability to add new data into the database (or) retrieve, update anddelete existing data in the database.
- **Authorization :**
  - The DBMS protects the database against unauthorized access either intentional (or) accidental.
- **Backup and recovery :**
  - The DBMS provides a mechanism for Backup data and recovery from different types of failures.
- **Concurrency control :**
  - The DBMS supports sharing of data among multiple users. The DBMS provides a mechanismfor concurrent access to the database.
- **Transaction Management :**
  - The transaction management provides access or change the content of database.

- **Integrity Service :**
  - The DBMS provides integrity rules to minimize data redundancy and maximize the dataconsistency.

[Q]. Define DBMS. Explain the Classifications (Types) of DBMS?

Database Management System:

- A Database management system (DBMS) is a collection of programs that allows users to create and maintain a database. It acts as an intermediator between user and database.
- The DBMS classified into different categories based on **data models**, **no. of users** and **the purpose**.

Based on data models:

- Depending on the data models, the DBMS classified into three categories - **hierarchical, network and relational DBMS.**
  - **Hierarchical DBMS :**
    - Organizes the data records in a tree structure i.e Hierarchy of parent and child relationship.
    - In a hierarchical data – base, a parent record may have more than one child, but a child always has only one parent. This is called „one – to- many relationship".
  - **Network DBMS :**
    - Organizes the data records liked to one another through pointers, which is an association between two records.
    - A network database is similar to a hierarchical database except that each child can have more than one parent record. This is called "many- to – many relationship".
  - **A relational DBMS :**
    - Organizes the data records in the form of table and relationship among the tables are set using fields.
    - It is simple in nature because data is simply represented in tabular format.

Based on no. of Users:

- Depending on no. of users the DBMS is divided into two categories – **single-user system** and **multi-user system.**
  - **Single - User System :**
    - In **single user system**, database resides on one computer and is only accessed by one user at a time. The user may design, maintain, and write programs for accessing and manipulating the database according to the requirements.
  - **Multi – User System :**
    - In **multi – user system**, multiple users can access the database simultaneously. In multiuser DBMS, the data is both integrated and shared.
    - For example, the online book database is a multi-user database system in which the data of books, authors, and publishers are stored centrally and can be accessed by many users.

Based on the purpose :

- Depending on the purpose, the DBMS classified into two categories – **general purpose DBMS** and **special purpose DBMS.**
- DBMS is a general purpose software system. It can however, be designed for specific purposes such as airline or railway reservation. Such systems cannot be used for other applications.
- These database systems fall under the category of online transaction processing (OLTP) systems. Online transaction processing systems is specially used for data entry and retrieval. It supports large number of concurrent transactions without excessive delays.

- An automatic Teller Machine for a bank is an example of online commercial transaction processing application. The OLTP technology is used in various industries, such as banking, airlines, supermarkets, manufacturing etc.

[Q]. Write a brief note on Evolution of Database Management System?

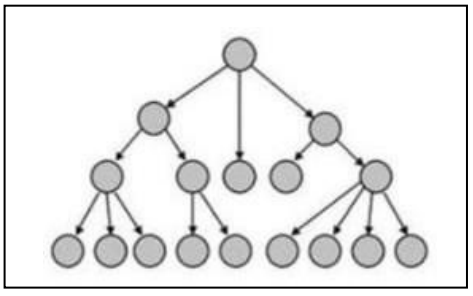
EVOLUTION OF DBMS :

- At the time of computers there was no data processing on those days. Computers are only used for engineering and scientific calculations. Gradually computers were introduced into the business world. For business applications computers must able to store and manipulate large amount of data.
- For this purpose introduce a new concept called file processing system. But file processing system has number of limitations is there. To overcome the limitations one more new system introduced called **Database management System**.
- DBMS were first introduced during the 1960s and have continued to evolve during subsequent decades. The database management system supports different types of database technologies or architectures or models.
- The Evolution (order of development of DBMS) of Database systems is as follows…
  - Flat files (1960 -1980)
  - Hierarchical (1970 -1990)
  - Network (1970 -1990)
  - Relational (1980 - present)
  - Object oriented (1990 - present)
  - Object relational (1990 - present)
  - Web enabled (1990 - present)

➢ **Flat files (1960-1980) :**

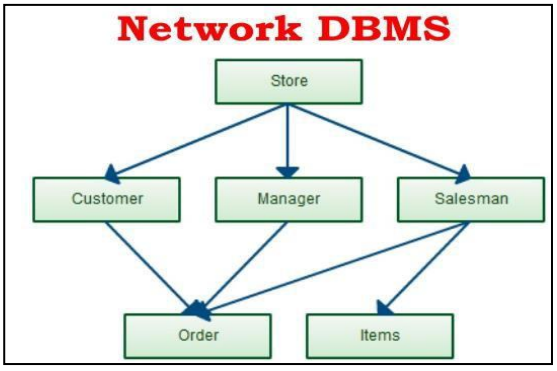
- In Flat file database system, data is stored in a single file or table. A flat file can be a plaintext file or a binary file. In a flat file database, there is no relationship between the records.
- **Example:**

	SNO	SNAME	GROUP
Record 1	1	A	BCOM
Record 2	2	B	BSC



➢ **Hierarchical (1970-1990) DBMS :**

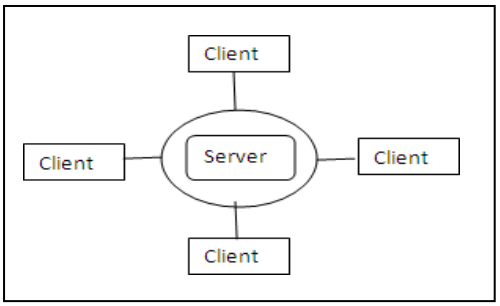
- In a Hierarchical DBMS, data is stored in a parent-children relationship mode i.e., data is organized in tree structure.
- In this, each child has only one parent and a parent contains multiple children. It maintains one-to-many relationship.



➢ **Network (1970-1990) DBMS :**

- In Network DBMS, data is stored in a parent-children relationship mode i.e., data is organized in graph structure.
- In this, each record contains multiple parent and child relationship. It maintains many-to-many relationship.

- **Relational (1980-present) DBMS :**
  - In a Relational DBMS, data is stored in the form of tables. Table contains primary keys and alternative keys. In a table, each row represents a **Record** and each column represents an **attribute**. A Relational database model is proposed by E.F.Codd in 1970.
  - **Examples**: Oracle, SQL Server, MySQL etc.



- **Object oriented Database (1990-present) :**
  - In Object oriented database system, data or information is stored in the form of objects as used in OOP language like C++ or JAVA.
- **Object relational database (1990-present) :**
  - Object relational DBMS is a combination of both Relational Database (RDBMS) and Object- oriented database (OODBMS). It supports basic components of any OODBMS in its schema and the query language used like objects, classes and inheritance.

[Q]. What is File-Based System? Explain the Drawbacks of File-Based System?

File-based System:

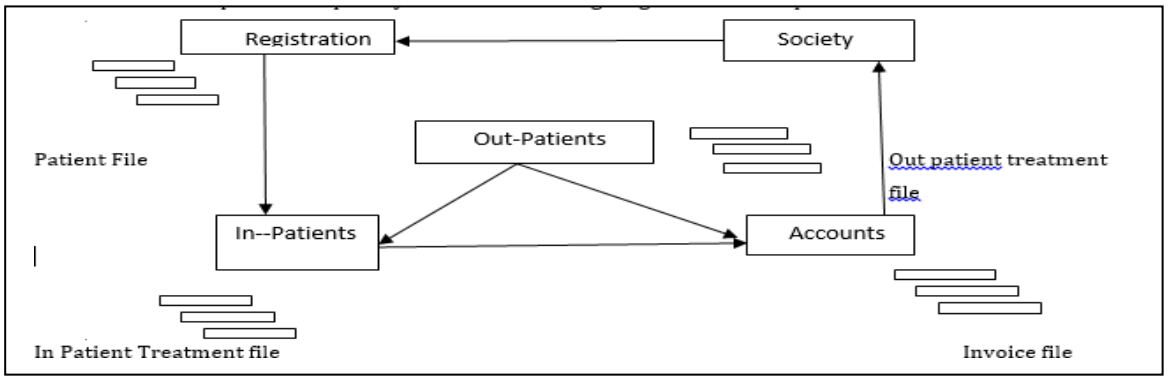
- A file based system is a method of storing and organizing the computer files and data that make easy to find and access it.
- Characteristics:
  - It is a group of files for storing data of an organization.
  - Each file is independent from one another.
  - Each file is called a flat file.
  - Files are designed by using programming languages like C,C++ etc.

Drawbacks of File based System:

The following are the drawbacks of file based system like….

- **Data Redundancy (Duplication of Data) :**
  - Data Redundancy means duplication of data values, i.e. same information is duplicated in several files. The file system data management forces the storage of same basic data in different locations. Duplication is wasteful because it costs time and money and also takes additional storage space.
- **Data Inconsistency :**
  - Data Inconsistency exists when different copies of same data appear in different places. The data in a file can become inconsistent when more than one person modifies the data. Entering wrong data is also another reason for inconsistency
- **Data Security :**
  - In file based system, the security of data is very low because the data is maintained in a flat file and it is easy to access.
- **Data Isolation :**
  - When the data is stored in separate files it becomes difficult to access. It becomes more complicated when the data has to be retrieved from more than one file
- **Difficulty of getting quick answers (Queries) :**
  - In file based system obtaining answers (results) for new requirements (queries) takes more time to obtain results.

- **Lack of Backup and recovery :**
  - In file based system there is no facility for backup and recovery from system failure.
- **Limited data sharing :**
  - In file based system, the data is stored in a decentralized manner, hence sharing of the data is complex.
- **Unable to represent relationships among data :**
  - In file based system there is no facility to represent relationship among data in different file for a single system.
- **Transactional problems :**
  - The file based system does not satisfy transactional problems (which are called as ACID properties (A- Atomic, C - Consistency, I - Integrity, D- Durability).
- **Concurrency problems :**
  - When multiple users update same data at a time then it may results in a problem. In filebased system, it is very difficult to handle this problem.
  - **Example of File-based System:** Consider the example of a Hospital System.



  - In this system we are using four files.
    - **Patient File**: At the Registration.
    - **In-patient Treatment File**: A t the in-patient section.
    - **Out-patient Treatment File**: At the out-patient section.
    - **In Voice File**: At the accounts section.

These files are maintained in different sections of the hospital in a decentralized manner, certain data items (patient no, patient name, patient address) will be duplicated. This will have some undesirable results.

[Q]. Explain the Advantages and Dis – advantages of Data Base Management System?

Database Management System:

- A Database management system (DBMS) is a collection of programs that allows users to create and maintain a database. It acts as an intermediator between user and database.

ADVANTAGES OF DBMS :

- In present days by using Database Management System we have a lot of advantages within that some of the most important like…

- **Controlled Redundancy of data :**
  - The database approach is a centralized place to save data. Hence, the amount of data redundancy is minimized. Data redundancy (or duplication) is minimized by applying normalization process in database design.
- **Getting quick answers (Queries) for complex queries :**
  - DBMS makes possible to produce quick answers to the queries by changing SQL queries in programs.
- **Improved Data Sharing :**
  - In database approach, data is shared by different applications or users simultaneously. In DBMS, multiple users will access same data and can do changes.
- **Relationships among data :**
  - We can apply relationships among data to improve performance of applications and consistency (or correctness) of data.
- **Efficient data access :**
  - DBMS uses techniques to store and retrieve the data efficiently.
- **Improved security :**
  - Data is important to any organization and also confidential. When multiple users access to the data, securing data is more critical. Hence database is protected from un-authorized users.
  - DBMS provides facilities for data security and privacy policies. This can be done by database administrator (DBA) by providing usernames and passwords only to the authorized users.
- **Improved integrity :**
  - The DBMS promotes and enforces integrity rules, thus minimizing data redundancy and maximizing data consistence. Data integrity refers to validity and consistency of data. This is done by applying some conditions.
- **Improved backup and recovery :**
  - In DBMS, if a transaction fails in middle of its execution due to system failure then DBMSwill recovers the data into its original position.

DIS-ADVANGES OF DBMS :

- **Increased complexity :**
  - A multi user DBMS becomes more complex due to functionalities. It is necessary to understand functionalities for database designers, developers, database administrator and end users. Failure to understand can lead to bad designed decisions
- **Large size of DBMS :**
  - The DBMS occupies large amount of storage space and requires more amount of memory to run efficiently.

- **Increased installation and maintenance cost :**
  - The DBMS software has a high initial cost. It requires trained person to install, operateand maintenance and also has more annual maintenance.
- **Conversion cost :**
  - The conversion cost from old database technology to modern database environment is high.

[Q]. Write the Applications of Data Bases?

APPLICATIONS OF DATA BASES :

- Some of The applications of database are…

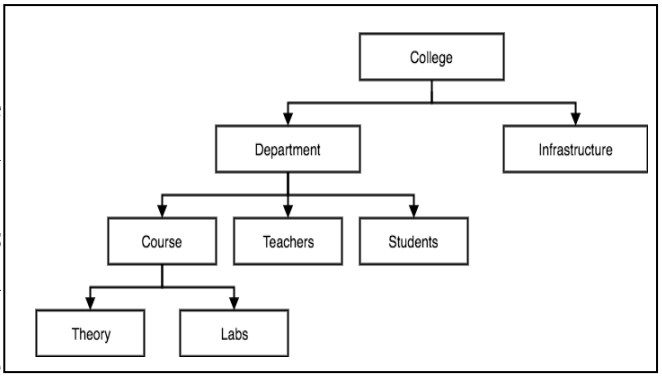
- **Banking :** For customer information, accounts, loans, Bank transactions.
- **Airlines :** For reservations and scheduled information.
- **Universities :** For student information, course registration and grading.
- **Telecommunication :** For keeping records of call mode, generating monthly bills, maintaining balances on pre-paid cards and storing information about the communication Network.

[Q]. Discuss about Data Models in detail?

Data Models:

- Data Modeling is a collection of concepts used to describe the structure of a database (i.e., data types, relationships, and constraints) and defines how data is stored, accessed and updated in a database management system. The Relational Model is the most widely used database model.

Types of Data models :



- **Hierarchical Model :**
  - The **Hierarchical Data Model** is the oldest type of data model, developed by IBM in1968.
  - In **Hierarchical database model**, data is organized in **tree-like-structure** with one **one-to-many** relationship.
  - A tree structure contains root node (or) parent node and the child nodes linked to theparent nodes.
  - In this model, each child can have only one parent node and a parent node have anynumber of child nodes.
  - For example, one department can have many courses, many professors and many students.

- **Network Model :**
  - In Network model, data is organized by collection of records, and a relationship among data is represented by links like a **graph structure** with **many-to-many** relationship.
  - In this, record types are represents are represented by Boxes, links are represented bylines.
  - For Example:

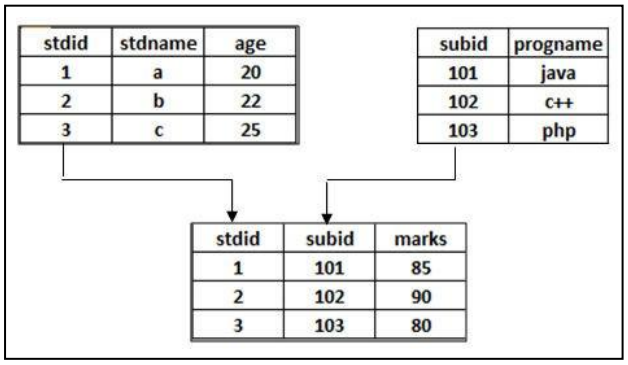
Data structure diagram

customer_Name	City	pincode	Account_no	Balance

Network model diagram

Rama	KDP	516289	33485875726	8500

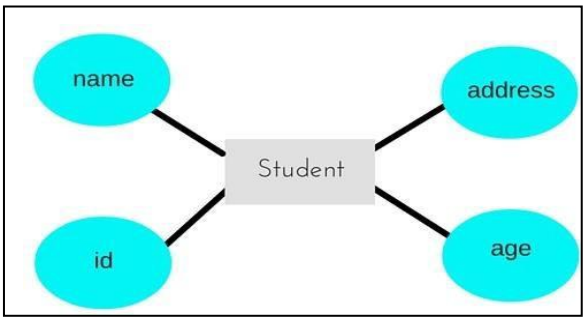
- **Relational Model :**
  - In Relational model, the data is organized in the form of **tables**.
  - In relational model, tables are also known as **relations**. Each row represents a **tuple,**
  - each column represents an **attribute**.
  - The Relational data model is implemented through a Relational Database Management System (RDBMS).Relational model was introduced by E.F Codd in 1970.
  - For Example:



- **Entity-relationship Model (E.R Model)**
  - Entity Relationship Model is the graphical representation of entities and their relationship in a database.
  - E.R. models are represented in an Entity relationship diagram (ER Diagram).The EntityRelation data model is introduced by Peter Chen in 1976.

- **The ER - Model contains following components:**

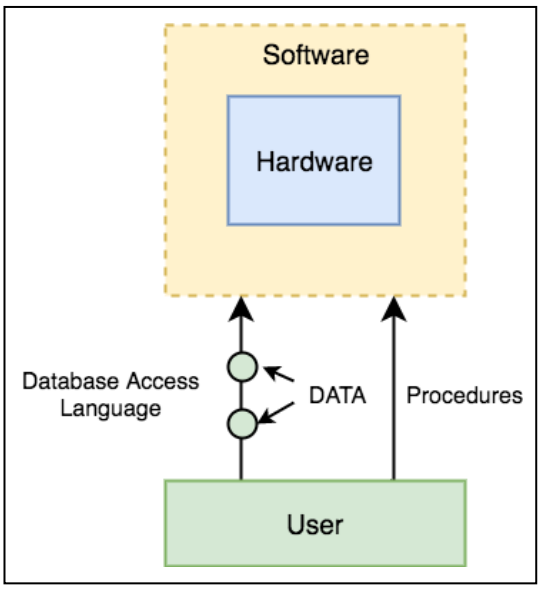
- **Entity:** entities are the real time objects. Entities represented by a rectangle.
  Ex: Student, Employee
- **Attribute:** Attributes are the characteristics of entities.
  Ex: sno, sname, group, Empno, Empname, Empaddress
- **Relationships:** A relationship describes association between the entities.
  For Example:



[Q]. Explain the Components of Database System?

COMPONENTS OF DATABASE SYSTEM:

- The database management system can be divided into five major components, they are:
  - Hardware
  - Software
  - Data
  - Procedures
  - Database Access Language
  - Users



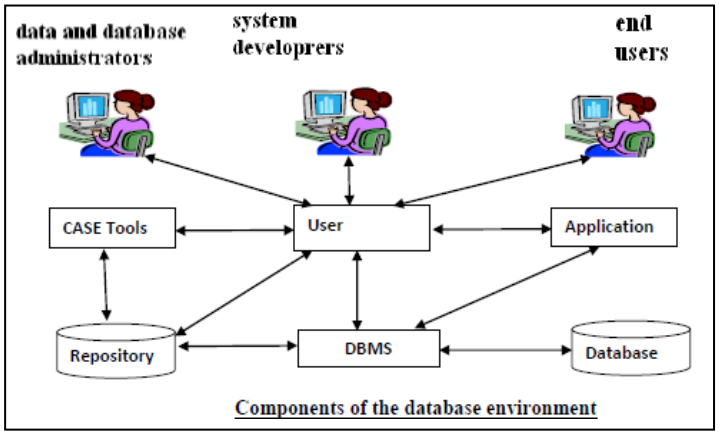
- **Software:**
  - Software is the set of programs used to control and manage the overall database.
  - It contains the DBMS software itself, the Operating System, the network software being used to share the data among users, and the application programs used to access data in the DBMS.
- **Hardware:**
  - Hardware contains a set of physical electronic devices such as computers, I/O devices, storage devices, etc., this provides the interface between computers and the real world systems.
- **Data:**
  - DBMS exists to collect, store, process and access data, the most important component. The database contains both the actual or operational data and the metadata.
- **Procedures:**
  - Procedures refer to general instructions to use a database management system. This contains procedures to setup and install a DBMS, To login and logout of DBMS software, to manage databases, to take backups, generating reports etc.
- **Database Access Language:**
  - Database Access Language is a simple language designed to write commands to access,insert, update and delete data stored in any database.
  - A user can write commands in the Database Access Language and submit it to the DBMS forexecution, which is then translated and executed by the DBMS.
  - User can create new databases, tables, insert data, fetch stored data, update data anddelete the data using the access language.
- **Users (People):**
  - **Database Administrators**: Database Administrator or DBA is the one who manages the complete database management system. DBA takes care of the security of the DBMS.
  - **Application Programmer or Software Developer**: This user group is involved in developing and designing the parts of DBMS.
  - **End User:** End users are the users who store, retrieve, update and delete data.

[Q]. Explain the Components of Database Environment in detail?

COMPONENTS OF DATABASE ENVIRONMENT:

- The major components of database environment like…
  - Computer aided software engineering tools(CASE tools)
  - Repository
  - Database
  - Database management system(DBMS)
  - User interface (Forms)
  - Application programs
  - Database administrator (DBA)
  - System developers
  - End users

- **CASE Tools:** - Computer Aided Software Engineering Tools. Some predefined tools (built in- tools) are used to design database and application programs. These tools are called Case tools. Some Case tools are:
  - Form generator
  - Report generators
  - Code generators
  - Diagramming tools



Components of the database environment

- **Repository (data dictionary):-** A repository is a centralized memory unit in the database generally; it is used to store table structures (Meta data), constraints information, database object information, database users‟ information etc.

- **Data base**: - It is a collection of "Related and Meaningful information "stored centrally at onelocation.

- **Database Management System (DBMS)** : It is a collection of programs working together to control Data Manipulations (add , change ,remove) , Retrievals (Read) and sharing on Database.

- **Application programs:-** Application programs are the programs that are developed by the programs generally application programs are used to manipulate the data in the data base. These programs provide information to users.

- **User interfaces: -** User interfaces are the programs that are used to manipulate data from the database easily generally user interface are created through the case tool form generator.

- **Database administrator: -** Database administrators are the persons who are responsible for the overall database in an organization database administrator uses case tools to improve databaseplanning and design

- **System developers:-**Persons such as analyst, designer's programmers and testers who developer's new application system developers uses case tools for system requirements analysis and design.

- **End users:-**Persons throughout the organizations who manipulated data in the database andwho requests and receive information from the database.

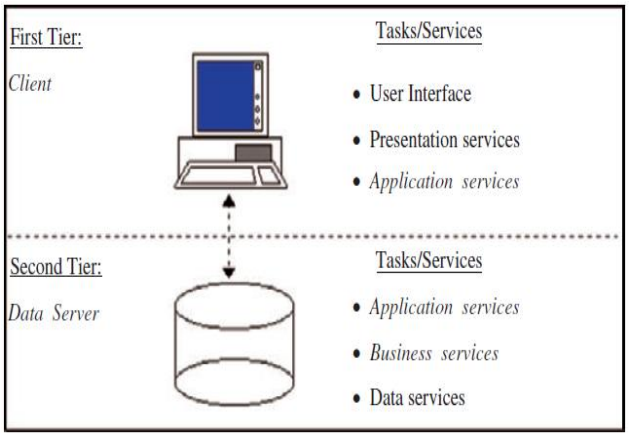
[Q]. Explain about Database Architecture? And also explain the types of Database Architecture?

DATABASE ARCHITECTURE:

- Database architecture essentially describes the location of all the pieces of information that make up the database application.
- The database architecture can be broadly classified into two-, three-, and multi-tier architecture.

Two-Tier Architecture:

- The two-tier architecture is a client–server architecture in which the client contains the presentation code and the SQL statements for data access. The database server processes the SQL statements and sends query results back to the client.
- Two-tier client/server provides a basic separation of tasks. The client, or first tier, is primarily responsible for the *presentation* of data to the user and the "server," or second tier, is primarily responsible for supplying *data services* to the client.



Two-tier client–server architecture

- **Presentation Services:**
  - "Presentation services" presents data to the user. In addition, it also provides for the mechanisms in which the user will interact with the data. More simply, presentation logic defines and interacts with the user interface.
- **Business Services/objects:**
  - "Business services" are a category of application services. Business services encapsulate an organizations business processes and requirements. These rules are derived from the steps necessary to carry out day-today business in an organization.
  - These rules can be validation rules, used to be sure that the incoming information is of a valid type and format, or they can be process rules, which ensure that the proper business process is followed in order to complete an operation.
- **Application Services:**
  - "Application services" provide other functions necessary for the application.
- **Data Services:**
  - "Data services" provide access to data independent of their location. The data can come from legacy mainframe, SQL RDBMS, or proprietary data access systems. Once again, the data services provide a standard interface for accessing data.

Advantages of Two-tier Architecture :

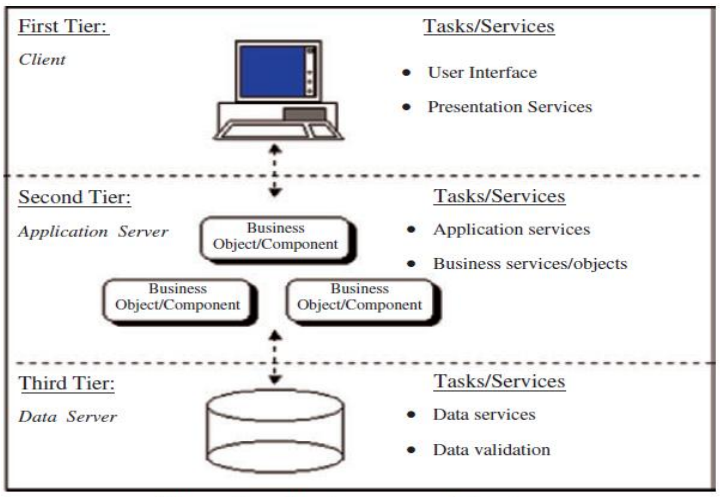
- The two-tier architecture is a good approach for systems with stable requirements and a moderate number of clients.
- The two-tier architecture is the simplest to implement, due to the number of good commercial development environments.

Drawbacks of Two-tier Architecture :

- Software maintenance can be difficult because PC clients contain a mixture of presentation, validation, and business logic code.
- To make a significant change in the business logic, code must be modified on many PC clients.
- Performance of two-tier architecture can be poor when a large number of clients submit requests because the database server may be overwhelmed with managing messages.

Three-Schema (tier) Architecture :

- The three schema architecture is also called ANSI/SPARC (American National Standard Institute/Standards planning and requirements committee) architecture or three-level architecture.
- This framework is used to describe the structure of a specific database system.
- The three schema architecture is also used to separate the user applications and physical database.
- A "Multitier," often referred to as "three-tier" or "*N*-tier," architecture provides greater application scalability, lower maintenance, and increased reuse of components.
- Three-tier architecture offers a technology neutral method of building client/server applications with vendors who employ standard interfaces which provide services for each logical "tier."
- From this figure, it is clear that in order to improve the performance; a second-tier is included between the client and the server. Through standard tiered interfaces, services are made available to the application.
- A single application can employ many different services which may reside on dissimilar platforms or are developed and maintained with different tools.

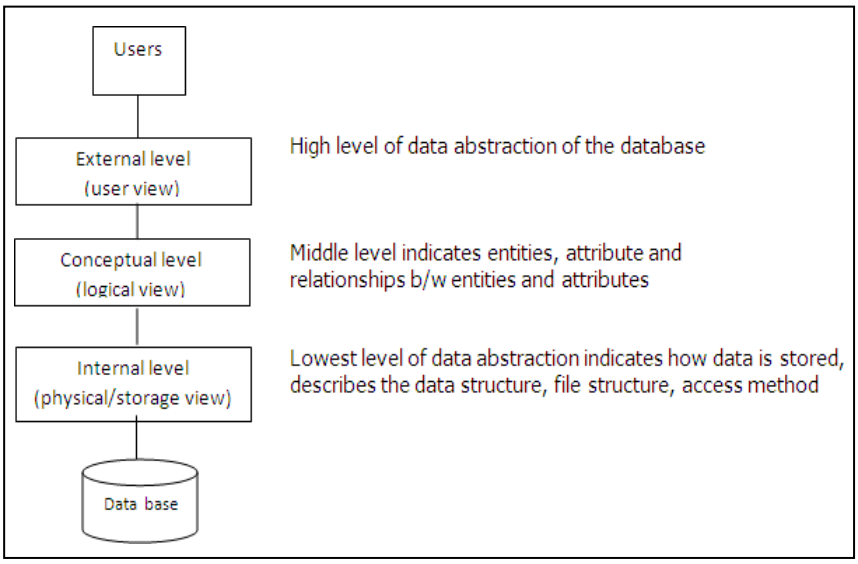


Three-tier client–server architecture

- This approach allows a developer to leverage investments in existing systems while creating new application which can utilize existing resources.
- Although the three-tier architecture addresses performance degradations of the two-tier architecture, it does not address division-of-processing concerns. The PC clients and the database server still contain the same division of code although the tasks of the database server are reduced.
- Multiple-tier architectures provide more flexibility on division of processing.
- The three schema architecture contains three-levels. It breaks the database down into three different categories.
- The 3-level of architecture are…
  - **Internal level :**
    - The internal schema defines the internal level. The internal level is the lowest level of data abstraction. This level indicates how the data will be stored into

the database and describes the file structures, data structures and methods used by the data base.

- **Conceptual level**
  - The conceptual schema defines the conceptual level. The conceptual level is the middle level abstraction. This level indicates entities, attributes, relationship between entities and attributes.
- **External level**
  - External schema defines the external level. The external level is the highest level of data abstraction. This level describes part of database, i.e. relevant to the user.



[Q]. Write about Database Approach?

Database Approach:

- A database is a collection of related information stored in a manner that many users share it for different purposes. The content of a database is obtained by integrating data from all the different sources at a centralized location in an organization.
- Such data is made available to all users as per the requirements and redundant data can be eliminated or at least minimized.
- The Data Base Management System (DBMS) governs to create an environment in which end users have better access to more and better managed data than they did before the DBMS become the data management standard.
- Some of the common database applications are student database system, business inventory, accounting information, organization data etc.
- Some commercially available DBMS are INGRES, ORACLE, DB2, Sybase, etc.
- Most database management systems support the following facilities/capabilities:
  - Creation, modification and deletion of data files.
  - Addition, modification, deletion of data
  - Retrieving of data collectively or selectively
  - Sorting or indexing of data
  - Creation of input forms and output reports.
  - To maintain data integrity and security
  - To create an environment for Data warehousing and Data mining

[Q]. Explain the Database vendors and their Products?

Database Vendors:

- A **database vendor** is an entity that offers one or more **databases** to customers for license or sale. Since there are so many database management systems available, it is important for there to be a way for them to communicate with each other.
- Some **DBMS** examples include MySQL, PostgreSQL, Microsoft Access, SQL Server, FileMaker, Oracle, **RDBMS**, dBASE, Clipper, and FoxPro.
- The systems are listed by type: relational(R), extended-relational(X), object relational (OR),object-oriented (OO), network (N) and hierarchical (H).

DBMS	Vendor	Type	Primary Market
MySQL	Freeware	R	Open Source
SQL Server	Microsoft	R	Enterprise
SQLBase	Centura Software	R	Mobile/Embedded
DB2	IBM	OR	Enterprise/VLDB
Oracle Lite	Oracle	OR	Mobile
Oracle 8I	Oracle	OR	Enterprise
PostgreSQL	Freeware	OR	Open Source

[Q]. Explain the various cost and risk factors involved in implementing a database system?

Cost and Risk factors involved in implementing a Database System:

- The database approach causes some additional costs and risks that must be recognized and managed when implementing this approach.
- In database approach in order to maintain or develop database we should take a risk and we should invest money, time and environment.
- Database approach when we develop a new database or when we maintain an existing database we should consider the following points.
- The various cost and risk factors involved in implementing a database system are:
  - **High cost:**
    - Installing a new database system may require investment in hardware and software. The DBMS requires more main memory and disk storage.
    - Moreover, DBMS is quite expensive. Therefore, a company needs to consider the overhead cost of implementing a new database system.
  - **Training new personnel:**
    - When an organization plans to adopt a database system, it may need to recruit or hire a specialized data administration group, which can coordinate with different user-groups for designing views, establishing recovery procedures and fine tuning the data structures to meet the requirements of the organization. Hiring such professionals is expensive.

- **Explicit backup and recovery:**
  - A shared corporate database must be accurate and available at all times. Therefore, a system using on-line updating requires explicit backupand recovery procedures.
- **System failure:**
  - When a computer system containing the database fails, all users have towait until the system is functional again.
  - Moreover, if DBMS or the application program fails, a permanent damage may occur to the database.
- **New, Specialized Personnel:**
  - Frequently, organizations that adopt the database approach need to hire or train individuals to design and implement databases.
  - This personnel increase seems to be expensive, but an organization should not minimize the need for these specialized skills.
- **Installation and Management Cost and Complexity:**
  - A multi-user database management system is large and complex software that has a high initial cost. It requires trained personnel to install and operate, and also has annual maintenance costs.
  - Installing such a system may also require upgrades to the hardware and data communications systems in the organization.
- **Conversion Costs:**
  - The term "*legacy systems"* is used to refer to older applications in an organization that are based on file processing. The cost of converting these older systems to modern database technology may seem prohibitive to an organization.
- **Need for Explicit Backup and Recovery:**
  - A shared database must be accurate and available at all times. This raises the need to have backup copies of data for restoring a database when damage occurs. A modern database management system normally automates recovery tasks.
- **Organizational Conflict:**
  - A database requires an agreement on data definitions and ownership as well as responsibilities for accurate data maintenance.
  - The conflicts on data definitions, data formats and coding causes updating of shared data. Handling these issues requires organizational commitment to the database approach.

UNIT II (ENTITY-RELATIONSHIP MODEL)

[Q]. Write about Entity Relationship Model? And its Advantages and Dis-advantages?

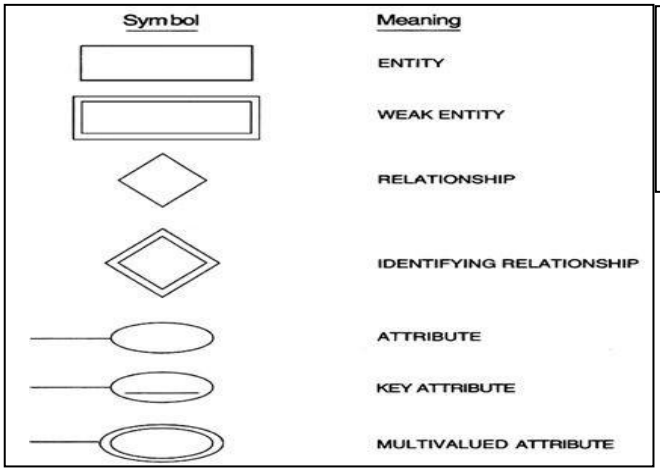
ENTITY RELATIONSHIP MODEL (ER- MODEL):

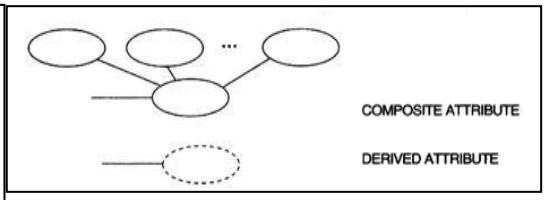
- Entity Relationship Model (ER Model) is a high-level **conceptual data model** developed by Peter Chen in 1976 that helps in designing database.
- It describes the structure of a database with the help of diagrams called as Entity Relationship Diagram.ER diagram contains set of objects called "entities" and relationships between entities.
- A **conceptual data model** is a set of concepts that describes the structure of a database and related retrieval and updating operations on database.

ER DIAGRAM:

- The ER diagram is used to represent database schema. It contains entities, attributes and relationships between entities.
- In ER diagram, a **rectangle** represents an entity, an **ellipse** represents an attribute and **diamond** represents a relationship. Lines represent linking of attributes to entity sets and entitysets to relationship sets.

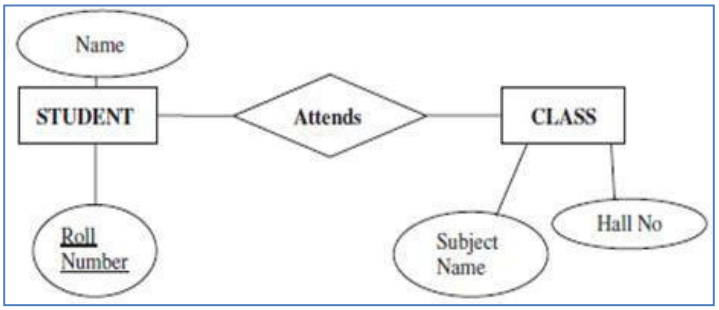
ER Diagram Notations:





Example:

- In the ER diagram, STUDENT and CLASS are two entities.
  - The STUDENT having two attributes like Roll number and the name.
  - The CLASS having two attributes like Subject Name and Hall Number. And the relationship between the two entities is **Attends**.



Features of ER Model:

- ER diagram is used to represent ER model and easily converted into relations (tables)
- ER model is used for the purpose of good database design
- It is very simple and easy to understand
- It is a top-down approach to database design

Advantages of ER Model:

- **Conceptual simplicity:** ER model represents the concepts of database along with entities and relationships. So it is easy to create and manage complex database designs.
- **Visual Representation:** ER model provides a visual representation of data and relationships among data. It is easy to understand the structure of data.
- **Effective communication tool:** The database designer uses ER model to get different views of data.
- The ER modeling provides an easily understood pictorial map for the database design.
- It is possible to represent the real world problems in a better manner in ER modeling.
- The conversion of ER model to relational model is straightforward.
- The enhanced ER model provides more flexibility in modeling real world problems.
- The symbols used to represent entity and relationships between entities are simple andeasy to follow.

Disadvantages of ER Model:

- ER model represents limited no. of relationships among entities.
- There is no industry standard notation for developing an ER diagram.
- It is popular for high level database design.

[Q]. Explain the basic building blocks of ER- Diagram with an example?

BASIC BUILDING BLOCKS OF AN ER – DIAGRAM:

- Entity Relationship Model (ER Model) is a high-level **conceptual data model** developed by Peter Chen in 1976 that helps in designing database.
- It describes the structure of a database with the help of diagrams called as Entity Relationship Diagram. ER diagram contains set of objects called "entities" and relationships between entities.
- The basic building blocks (or components) of Entity-Relationship diagram are…
  - **Entity**
  - **Attribute**
  - **Relationship**.

➢ **ENTITY:**

- An Entity is a real world object like person, place or thing. Any living and non-living objects are also called as Entity.
- An entity is represented by rectangle along with entity name. The entity names are generally written in capital letters.
- **Example:** STUDENT, CUSTOMER, EMPLOYEE and PRODUCT etc.

➢ **ATTRIBUTES:**

- An Attribute is used represents characteristics or properties of entities or entity types. An attribute is represented by "Ovals" and are connected to the entity with a line. Each oval contains name of the Attribute.
- **Example:** STU_NAME, STU_ADDRESS, ROLL_NO are the attributes of STUDENT.

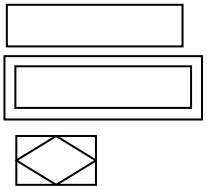
➢ **RELATIONSHIP:**

- An association between one or more entities is called "relationship". There are three different types of relationships.
  - One-One relationship
  - One-Many relationship
  - Many-Many relationship
- **Example:**
  - **Teaches** is the relationship type between LECTURER and STUDENT.
  - **Treatment** is the relationship between DOCTOR and PATIENT.

[Q]. What is Entity? Explain different types of entities (or) write aboutclassification of Entity Sets?

ENTITY SET:

- An Entity set is a collection of entities that share same attributes. (or) An entity set is a collection of similar type of entities.
- Entity sets are classified into following types:
  - **Strong entity**
  - **Weak entity**
  - **Associative entity**

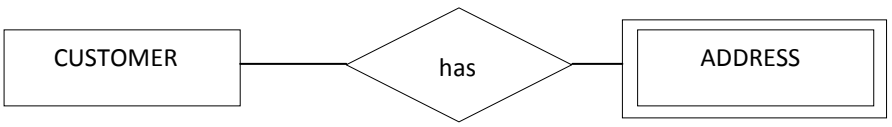


➢ **Strong Entity:**

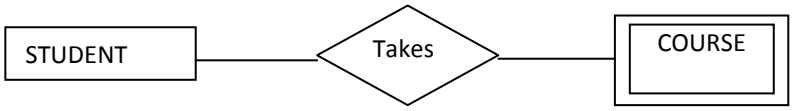
- Strong entity is an entity type whose existence does not depend on other entity. An entity that has an attribute that acts as a primary key is called **Strong Entity**. In ER diagram, strong entity is represented by "single outlined box".
- **Example:** Student takes course

➢ **Weak Entity:**

- Weak entity is an entity whose existence depends on other entity. An entity that does not has an attribute that acts as a primary key is called **Weak Entity**. In ER diagram, weak entity is represented by "double outlined box".
- **Example:** each Customer has Address



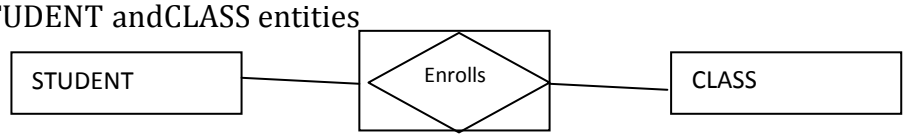
Example:



- In this example, STUDENT is a strong entity. COURSE is a weak entity. Because, if there are no students to take a particular course, then that course cannot be offered. The COURSE entity depends on the STUDENT entity.

➢ **Associative Entity:**

- Basically, it is a relation and acts as entity type. It is used implement a many-to-manyrelationship between entities.
- **Example:** In this, the relationship "enrolls" acts as an associative entity for both STUDENT andCLASS entities

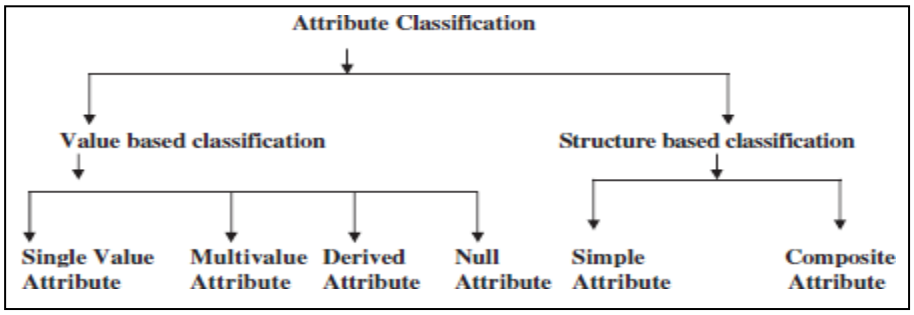


[Q]. What is Attribute? Explain different types of Attributes (or) write about classification of Attributes ?

ATTRIBUTE:

- An Attribute is used represents characteristics or properties of entities or entity types. Attributeis classified based on **value** and **structure**.

➢ **Based on value :** single value, multivalve, derived, null attribute.
➢ **Based on structure :** simple and composite attribute.

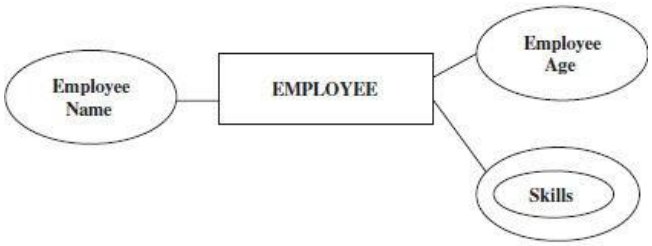


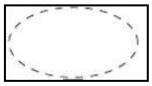
☞ **Single Value Attribute :** 

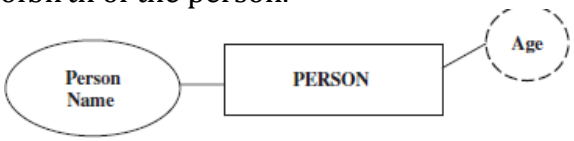
- A **Single valued Attribute** is an attribute that contains only one value. In ER diagram, thesingle value attribute is represented by "ellipse".
- **Example:** in EMPLOYEE entity**,** Employee Name, Employee Age

☞ **Multivalued Attribute:** 

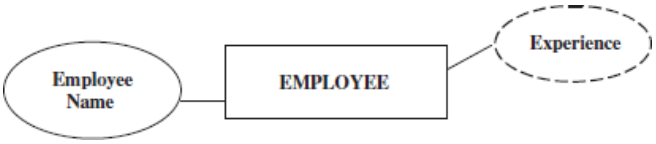
- A **Multi Value Attribute** is an attribute that contains many values. In ER diagram, themultivalued attribute is represented by "doubled ellipse".
- **Example:** In EMPLOYEE entity, "skills" is a multi-valued attribute.



- **Derived Attribute :** 
  - It is an attribute that's value is derived from the other attributes or entities then it is called as derived attribute. In ER diagram, the derived attribute is represented by "dotted ellipse".
  - **Examples:** In this example, age is the derived attribute. Because, age is derived from the date ofbirth of the person.



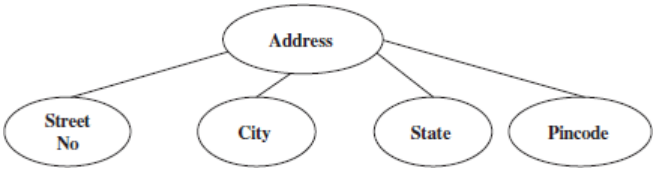
  - In this example, Experience is derived from date of joining of the employee.



- **Null value attribute:**
  - The attributes which take null values are called as null value attributes.
  - **Example:** sometimes, Phone number taken as null value if it is not available

- **Simple Attribute:**
  - A **Single Attribute** is an attribute that cannot be sub divided.
  - **Example:** in this example, streetNo, city, state, pincode

- **Composite Attribute:**
  - A **Composite attribute** is an attribute which is further subdivided into simple attributes.
  - **Example:** the "address" attribute is subdivided into Streetname, City, and State etc.



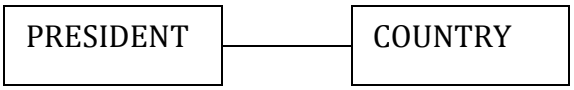
[Q]. Explain about different types of Relationship (or) Explain about classification of relationship ?

RELATIONSHIP:

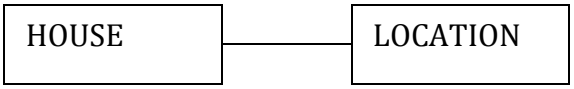
- An association between one or more entities is called as "Relationship". The relationship is classified into one-to-one relation, one-to-many relation, many-to-many relation and many-to-one relation.
- Based on the cardinality of relationships they are classified into the following categories…

- One – to – many relationship (1:M)
- Many- to –many relationship (M:M)
- One – to – one relationship (1:1)

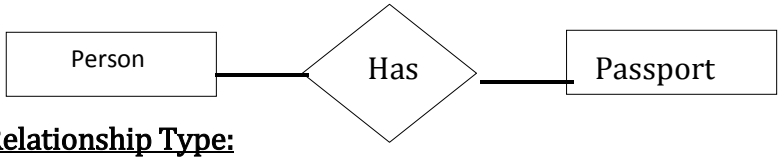
- **One-to-One Relationship Type:**
  - The relationship that associates exactly one entity to one entity is called one-to-onerelationship.
  - **Example 1:** President and the country For a particular **country** there is only one **President**. And a country willnot have more than one President.



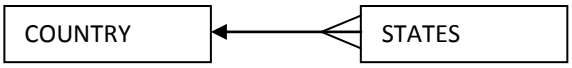
  - **Example 2:** House and Location. A house is exactly in only one location.



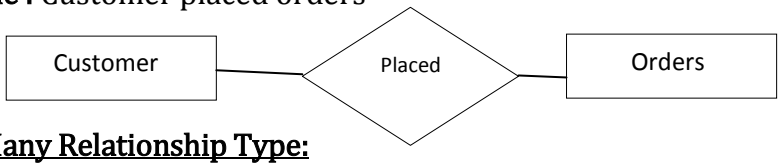
  - **Example 3: PERSON and PASSPORT . A person has exactly only one passport**



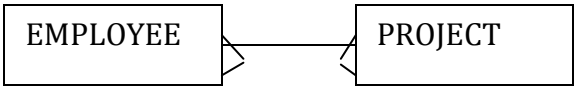
- **One-to-Many Relationship Type:**
  - The relationship that associates one entity to more than one entity is called one-to-many relationship.
  - **Example :** Country having states. For one country there can be more than one state.



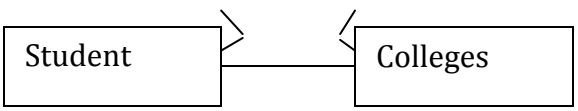
  - **Example :** Customer placed orders



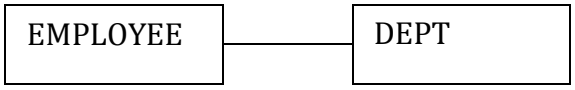
- **Many-to-Many Relationship Type:**
  - The relationship that associates more than one entity to more than one entity is calledmany-to-many relationship.
  - **Example:** EMPLOYEE and PROJECT. Many employees will be working in many projects.



  - **Example:** Student and Colleges



- **Many-to-One Relationship Type:**
  - The relationship that associates more than one entity to one entity is called many-to-one relationship.
  - **Example:** Consider EMPLOYEE and DEPARTMENT. There are many EMPLOYEESworking in one DEPARTMENT.



[Q]. What is Relationship Degree? Explain about different types of relationship degree?

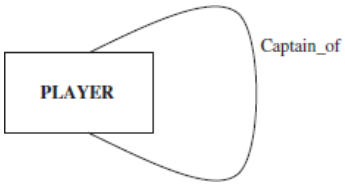
Relationship Degree :

- The association between one or more entities is called as "Relationship". The no. of entities associated with the relationship is called as "**degree of relationship**".
- The relationship degree is classified into 3 types…
  - ☞ Unary relationship
  - ☞ Binary relationship
  - ☞ Ternary relationship
  - ☞ Quaternary Relationships

☞ **Unary Relationship :**

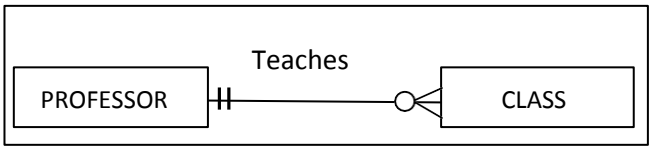
- When there is only one entity set is participating in a relation, the relationship is called as Unaryrelationship.
- **Example:** one person is married to only one person

Example:



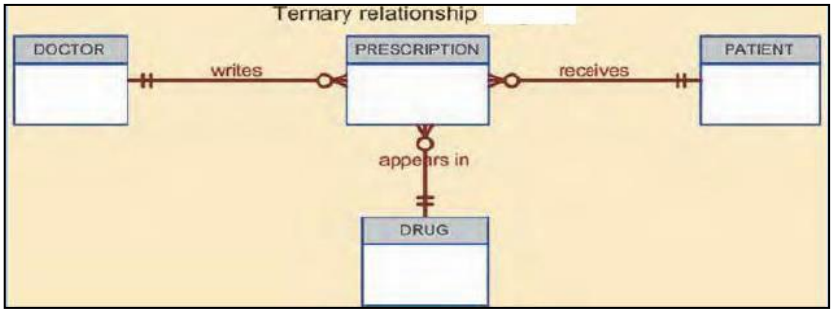
☞ **Binary Relationship :**

- When there are two entity sets participating in a relation, the relationship is called as Binaryrelationship.
- **Example:** Professor and Class. The relationship a PROFESSOR TEACHES ONE or MORE CLASS.



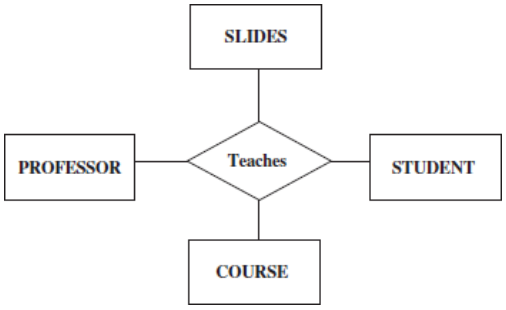
☞ **Ternary Relationship :**

- When there three entity sets participating in a relation, the relationship is called as Ternaryrelationship.
- **Example:** A doctor writes one or more prescriptions.
  - A patient may receive one or more prescriptions. A drug may appear one or more prescriptions.



☞ **Quaternary Relationships:**

- When there four entity sets participating in a relation, the relationship is called as Quaternaryrelationship.
- **Example:** "A professor teaches a course to students using slides." PROFESSOR, SLIDES, COURSE, andSTUDENT are four entities.



[Q]. Explain about Reducing ER diagram to Tables with an examples?

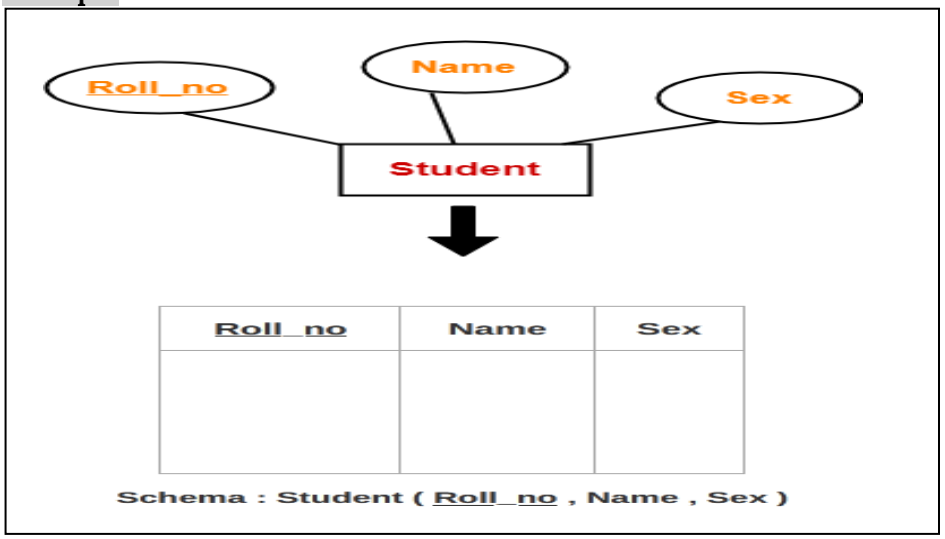
REDUCING ER-DIAGRAM TO TABLES:

- ER diagram is converted into the tables in relational model. This is because relational modelscan be easily implemented by RDBMS like MySQL, Oracle etc.

➢ **Rule-01: For Strong Entity Set With Only Simple Attributes :**

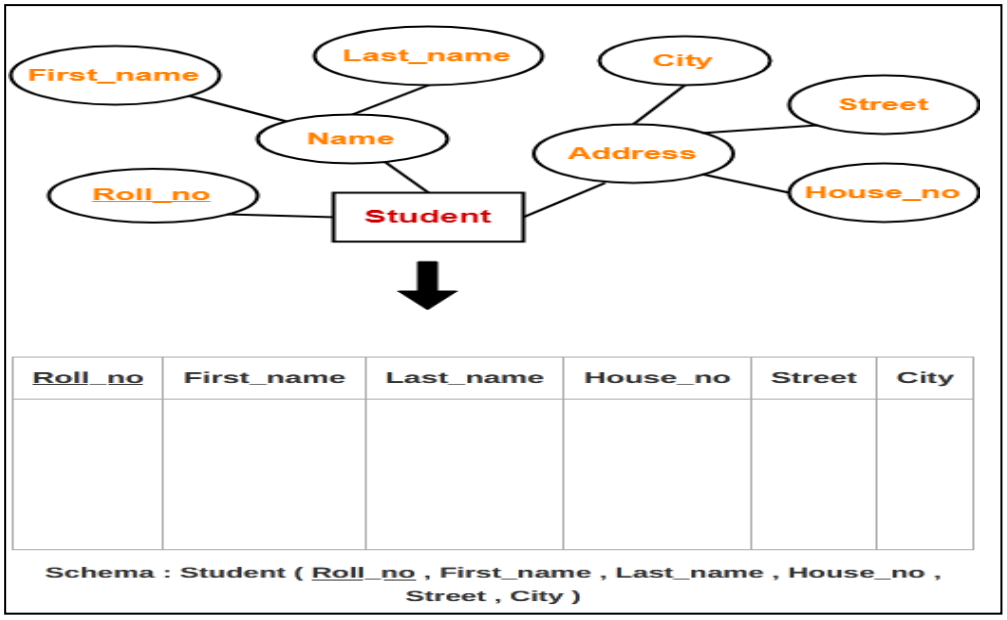
- A strong entity set with only simple attributes will require only one table in relational model. Attributes of the table will be the attributes of the entity set.
- The primary key of the table will be the key attribute of the entity set.

Example



➢ **Rule-02: For Strong Entity Set With Composite Attributes:**

- A strong entity set with any number of composite attributes will require only one tablein relational model.
- While conversion, simple attributes of the composite attributes are taken into accountand not the composite attribute itself.

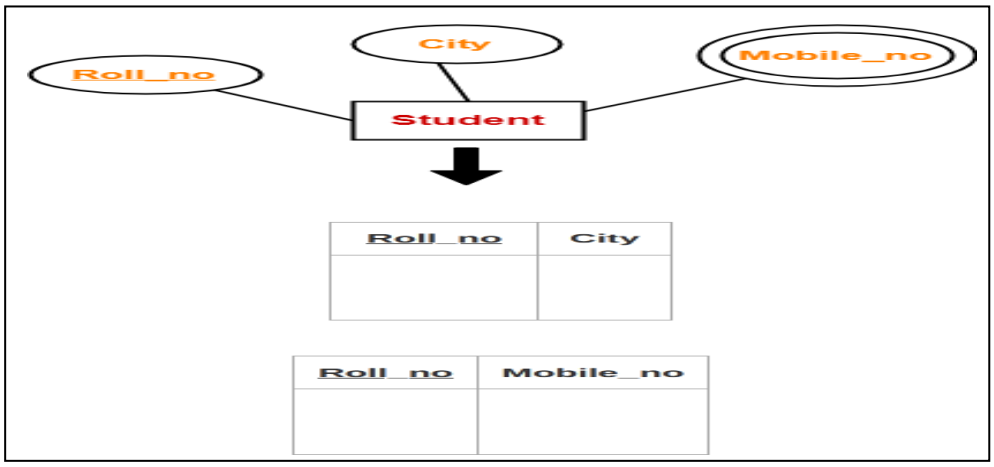


Roll_no	First_name	Last_name	House_no	Street	City

Schema : Student (Roll_no , First_name , Last_name , House_no , Street , City)

➢ **Rule-03: For Strong Entity Set With Multi Valued Attributes:**

- A strong entity set with any number of multi valued attributes will require two tables inrelational model.
- One table will contain all the simple attributes with the primary key.
- Other table will contain the primary key and all the multi valued attributes.

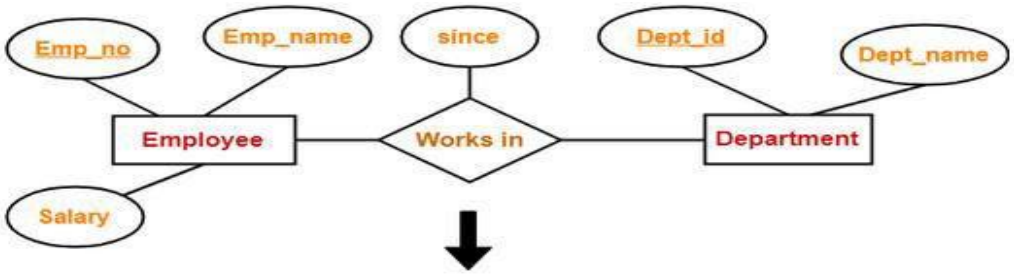


Roll_no	City

Roll_no	Mobile_no

➢ **Rule-04: Translating Relationship Set into a Table :**

- A relationship set will require one table in the relational model.Attributes of the table are…
- Primary key attributes of the participating entity sets
- Its own descriptive attributes if any Set of non-descriptive attributes will be the primary key.



Emp_no	Dept_id	since

Schema : Works in (Emp_no , Dept_id , since)

NOTE-

If we consider the overall ER diagram, three tables will be required in relational model-

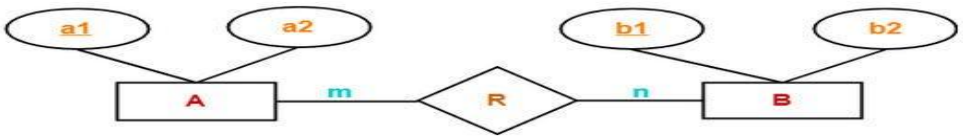
- One table for the entity set "Employee"
- One table for the entity set "Department"
- One table for the relationship set "Works in"

- **Rule-05: For Binary Relationships With Cardinality Ratios :**

The following four cases are possible like…

Case-01: Binary relationship with cardinality ratio m:n.
Case-02: Binary relationship with cardinality ratio 1:n.
Case-03: Binary relationship with cardinality ratio m:1
Case-04: Binary relationship with cardinality ratio 1:1

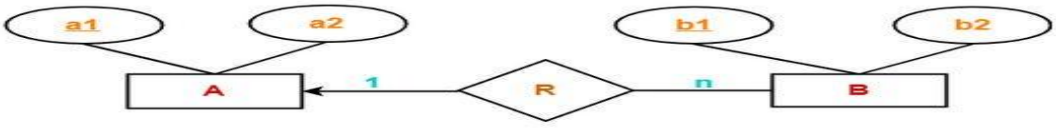
Case-01: For Binary Relationship With Cardinality Ratio m:n



Here, three tables will be required-

1. A (a1 , a2)
2. R (a1 , b1)
3. B (b1 , b2)

Case-02: For Binary Relationship With Cardinality Ratio 1:n

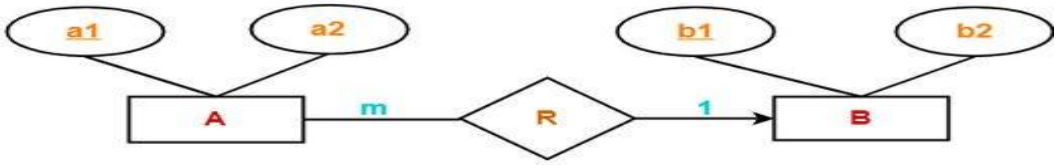


Here, two tables will be required-

1. A (a1 , a2)
2. BR (a1 , b1 , b2)

NOTE- Here, combined table will be drawn for the entity set B and relationship set R.

Case-03: For Binary Relationship With Cardinality Ratio m:1

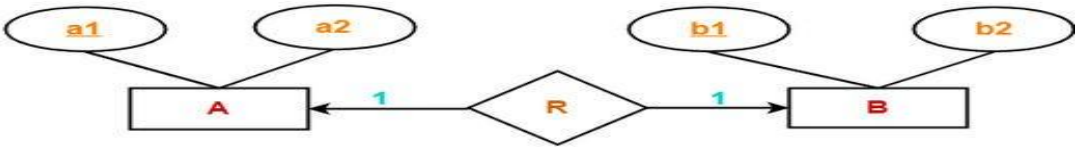


Here, two tables will be required-

1. AR (a1 , a2 , b1)
2. B (b1 , b2)

NOTE- Here, combined table will be drawn for the entity set A and relationship set R.

Case-04: For Binary Relationship With Cardinality Ratio 1:1



Here, two tables will be required. Either combine 'R' with 'A' or 'B'

Way-01:

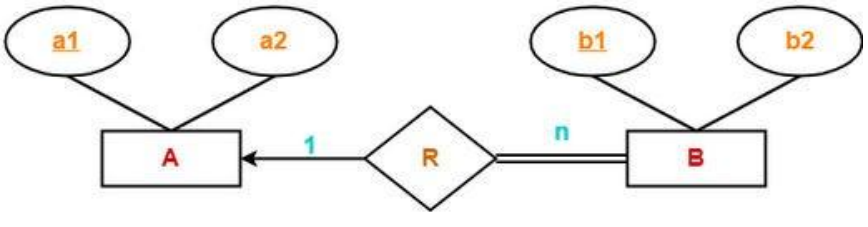
1. AR (a1 , a2 , b1)
2. B (b1 , b2)

Way-02:

1. A (a1 , a2)
2. BR (a1 , b1 , b2)

- **Rule-06: For Binary Relationship With Both Cardinality Constraints and Participation Constraints:**
  - Cardinality constraints will be implemented as discussed in Rule-05.
  - Because of the total participation constraint, foreign key acquires **NOT NULL constraint** i.e. now foreign key cannot be null.

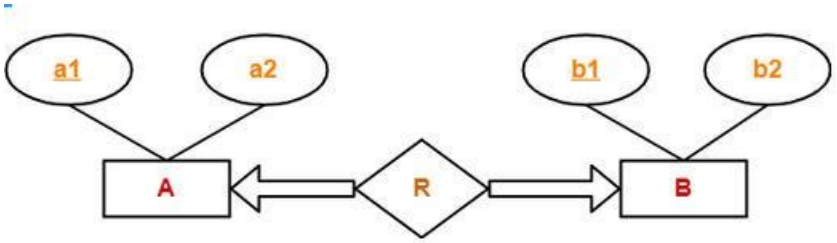
Case-01: For Binary Relationship With Cardinality Constraint and Total Participation Constraint From One Side:



- Because cardinality ratio = 1 : n , so we will combine the entity set B and relationship set R.Then, two tables will be required-
  1. A (a1 , a2)
  2. BR (a1 , b1 , b2)
- Because of total participation, foreign key a1 has acquired NOT NULL constraint, so it can't benull now.

Case-02: For Binary Relationship With Cardinality Constraint and Total Participation Constraint From Both Sides:

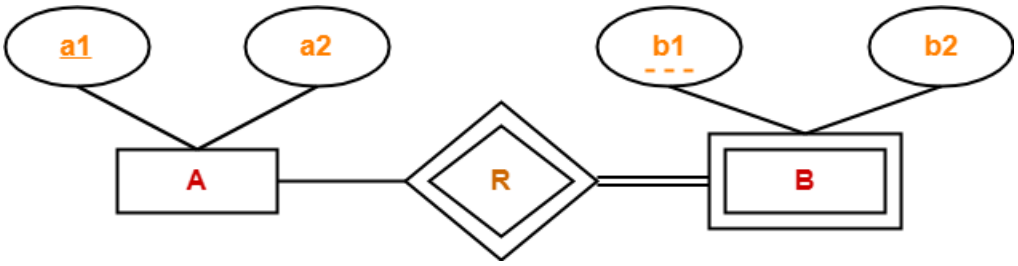
- If there is a key constraint from both the sides of an entity set with total participation, then that binary relationship is represented using only single table.



Here, only one table is required.

- ARB (a1 , a2 , b1 , b2)

➢ **Rule-07: For Binary Relationship With Weak Entity Set :**



- Weak entity set always appears in association with identifying relationship with totalparticipation constraint.
- Here, two tables will be required-
  1. A (a1 , a2)
  2. BR (a1 , b1 , b2)

[Q]. What is Enhanced Entity–Relationship Model (EER MODEL) (or) Explain about Basic concepts of EER – Model?

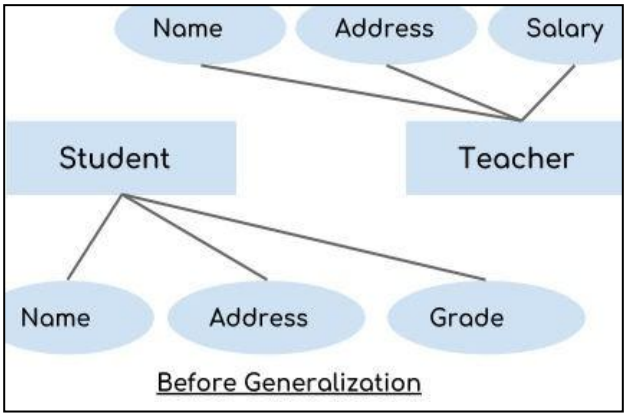
Enhanced Entity–Relationship Model:

- The basic concepts of ER modelling are not powerful for some complex applications. Hence some additional modeling concepts are required. These are provided by Enhanced ER model.
- The Enhanced ER model is the extension of the ER model with new modeling constructs like super type (super class)/subtype (subclass) relationships.

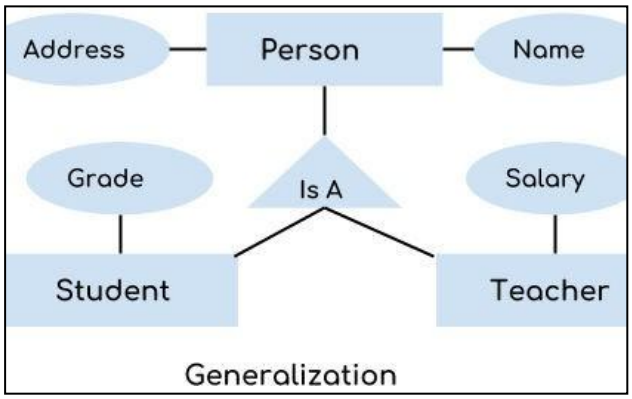
Enhanced ER model = ER model + hierarchical relationships.

- **Super type (or) Super class :**
  - Super type (or) super class is an entity type that has a relationship with one or more subtypes.
  - **For example:** PLAYER is an entity type which has a relationship with one or more subtypes like CRICKET PLAYER, FOOTBALLPLAYER, HOCKEY PLAYER, TENNIS PLAYER, etc.
- **Subtype (or) Sub class :**
  - A subtype or subclass is a sub grouping of the entities in an entity type. A subclass entity type represents a subset or sub grouping of super class entity type's instances.
  - Subtypes inherit the attributes and relationships associated with their super type.
  - **For example:** ENGINE entity type has two subtypes like PETROLENGINE and DIESEL ENGINE. A STUDENT entity type has two subtypes UNDERGRADUATE and POSTGRADUATE.
- **Generalization: (Explain about Generalization and Specialization in detail?)**
  - Generalization is a process in which the common attributes of more than one entitiesform a new entity.
  - In this, two or more lower level entities combine together to form a higher level newentity. This newly formed entity is called generalized entity.
  - The new generalized entity can further combine together with lower level entity tocreate a further higher level generalized entity.
  - Generalization is a **bottom-up** process.
  - In EER diagram, generalization is represented by labeled with ISA.
  - **Generalization Example :**

    Consider Student and Teacher entities. Attributes of Entity Student are: Name, Address & GradeAttributes of Entity Teacher are: Name, Address & Salary.
  - The ER diagram before generalization looks like this:
  - These two entities have two common attributes: Name and Address, we can make ageneralized entity with these common attributes.



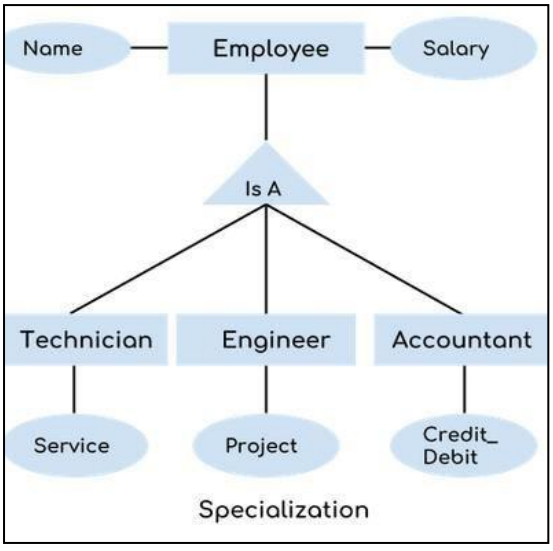
  - **The ER diagram after generalization:**
  - In the following ER Diagram after the generalization process the entities Student and Teacher only has the specialized attributes **Grade** and **Salary** respectively and their common attributes (Name & Address).



- These common attributes are now associated with a new entity Person which is in the relationship withboth the entities (Student & Teacher).

➢ **Specialization:**

- **Specialization** is a process in which an entity is divided into sub-entities.
- It is a reverse process of generalization, in generalization two entities combine together to forma new higher level entity.
- Specialization is a **top-down** process.
- In EER diagram, generalization is represented by ▽ labeled with ISA.
- **For example** – Consider an entity employee which can be further classified as sub-entities Technician, Engineer & Accountant because these sub entities have some distinguish attributes.
- **Specialization Example:**
- In this diagram, higher level entity "Employee" divided in sub entities "Technician", "Engineer" & "Accountant".
- Just for the example, that Technician handles service requests, Engineer works on a project and Accountant handles the credit & debit details.



UNIT- III (RELATIONAL MODEL)

[Q]. Explain about Relational Data Model and write its features?

RELATIONAL MODEL:

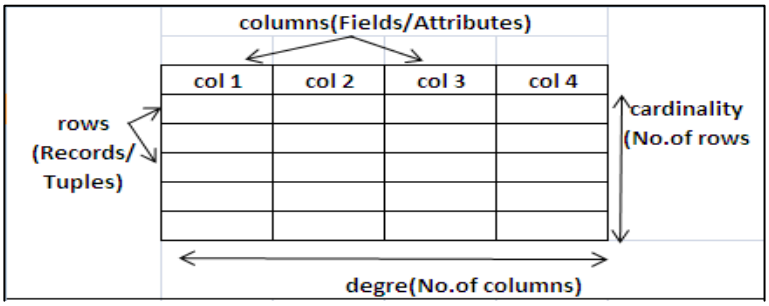
- The Relational model represents both data (entities) and relationships among data in the form of tables. Each table has multiple columns and each column has a unique name.
- In relational data model the data is stored in the form of tables. Relational data model is easier to understand than the hierarchal data models and network data models.
- Relational data model provides a logical view of the data and its relationship among other data.
- The relational model was introduced in 1970 by E.F.Codd. The relational data model isimplemented through a Relational Database Management System (RDBMS).

Feature of Relational Data model:

- The key features of relational data model are as follows:
  - Each row in the table is called tuple.
  - Each column in the table is called attribute.
  - The intersection of row with the column will have data value.
  - In relational model rows and attributes can be in any order.
  - By definition, all rows in a relation are distinct. No two rows can be exactly the same.
  - Relations must have a key. Keys can be a set of attributes.
  - For each column of a table there is a set of possible values called its **domain**.
  - Domain is the set of valid values for an attribute.
  - Degree of the relation is the number of attributes (or columns) in the relation.
  - Cardinality of the relation is the number of tuples (or rows) in the relation.

- The terms commonly used by user, model, and programmers are given later.

User	Model	Programmer
Row	Tuple	Record
Column	Attribute	Field
Table	Relation	File



Example of Relational Model: EMPLOYEE

Empno	Empname	Deptname	Salary	deptno
101	Soohan	Medical	45000	10
102	Bhavya	IT	60000	20
103	Madhu	Sales	55000	30

- **In that above relation:**
  The degree of the relation (i.e., is the number of column in the relation) = 5. The cardinality ofthe relation (i.e., the number of rows in the relation) = 3.

[Q]. Explain about CODD'S RULES ?

CODD'S RULES:

- In 1985, **Edgar Frank Codd** defined 12 important relational database rules that make a database system as relational database system.
- After publishing the original article Codd stated that there are no systems that will satisfy every rule. Any database system that follows minimum 6 Codd‟s rules is known as RDBMS.

➢ **Rule 1: Information:**

- The data stored in a database, may it be user data or metadata, must be a value of sometable cell. Everything in a database must be stored in a table format.

➢ **Rule 2: Guaranteed Access:**

- Every item of data must be logically addressable with the help of a table name, primary key value and column name.

➢ **Rule 3: Systematic Treatment of NULL values:**

- The RDBMS must be able to support null values to represent missing or accessible to users with appropriate authority and are stored in the data dictionary.

➢ **Rule 4: Active online catalog:**

- The structure description of the entire database must be stored in an online catalog, known as data dictionary, which can be accessed by authorized users. These are accessible to users with appropriate authority and are stored in the data dictionary. (or)
- The metadata must be stored as ordinary data in a table within the database. Such datamust be available to authorized users.

➢ **Rule 5: Comprehensive data sub language:**

- The relational database supports many languages. It must support data definition, view definition, data manipulation integrity constraints, authorizations and transaction management.

➢ **Rule 6: View Update:**

- All views that are theoretically updatable must also be updatable by the RDBMS.

➢ **Rule 7: High level Insert, Update and Delete:**

- A database must support high-level insertion, updating, and deletion. This must not be limited to a single row, that is, it must also support union, intersection and minus operations to yield sets of data records.

➢ **Rule 8: Physical data independency:**

- The data stored in a database must be independent of the applications that access the database.

- Any change in the physical structure of a database must not have any impact on how the data is being accessed by external applications.

➢ **Rule 9: Logical data independency:**

- The logical data in a database must be independent of its users view (application). Anychange in logical data must not affect the applications using it.
  - For example, if two tables are merged or one is split into two different tables, there should beno impact or change on the user application. This is one of the most difficult rule to apply.

➢ **Rule 10: Integrity Independency:**

- All relational integrity constraints (like primary key, foreign key, unique, check, default, not null) must be definable in the relational language and stored in the system catalogs.
  - (Note: - Integrity means completeness, correctness and consistency).

➢ **Rule 11: Distribution Independency:**

- The end-user must not be able to see that the data is distributed over various locations. (Note: - Data independence means users should not have awareness of whether a database isdistributed at different sites or not.)

➢ **Rule 12: Non Sub Version:**

- If the system supports low – level access to the data, there must not be a way to bypassthe integrity rules of the database. This is necessary for data integrity.

[Q]. Explain about Integrity Constraints (Or) Relational Integrity Constraints?

Integrity Constraints:

- The data available in the database must be complete and correct data. To maintain correctness and completeness of data oracle has given some data integrity constraints.
- Relational data base integrity rules are very important to good data base design. Many RDBMSs enforce integrity rules automatically.
- Data integrity constraints refer to the correctness, completeness and consistency of data in the database. It is another form of protecting database.
- **Integrity constraints** are set of rules used to maintain the quality of information. The different types of data integrity constraints are
  - ➢ Domain integrity constraints
    - Not null
    - Check
  - ➢ Entity integrity constraints
    - Unique
    - Primary key
  - ➢ Referential integrity constraints
    - References

- **Domain integrity constraints:**
  - It is used to check the validity of entries for a given column and it restricts duplicate values into table columns.
  - **Not null:** - It is used to restrict null values, any number of duplicate values allowed.
  - **Check:** - It is used to provide conditional restrictions on table columns.
  - **Example:**

Empno	Empname	Deptname	Salary	deptno
101	Soohan	Medical	45000	10
102	Bhavya	IT	60000	20
103	Madhu	Electrical	50000	a

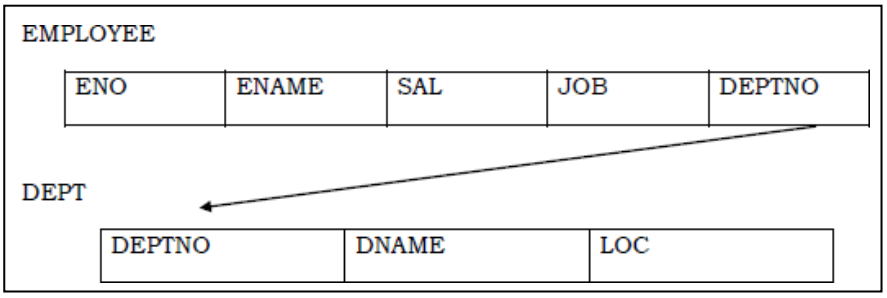
Not allowed, because deptno is an integer type

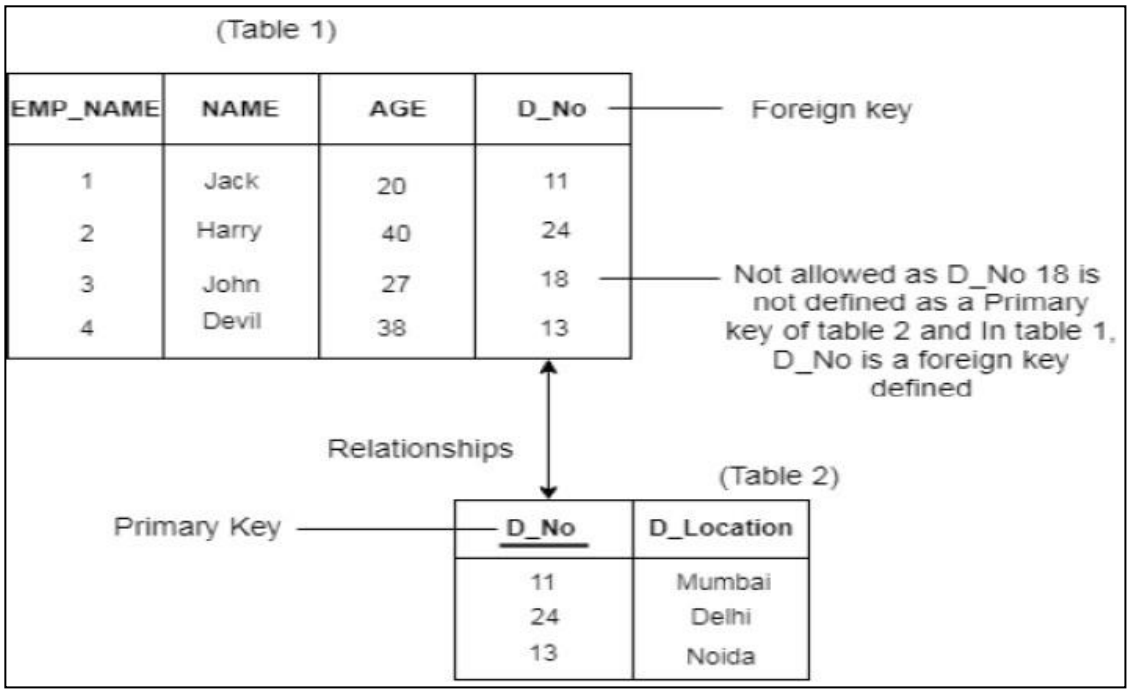
- **Entity integrity constraints :**
  - It is used to provide conditional restrictions on table columns. The Entity integrity constraint says that primary key value can't be null or duplicate. The primary key uniquely identifies rows in a relation.
  - **Unique**: - Used to restrict duplicate values but any no. of null values are allowed.
  - **Primary key**: - used to restrict both Null and duplicate values.
  - **Example:**

Empno	Empname	Deptname	Salary	deptno
101	Soohan	Medical	45000	10
102	Bhavya	IT	60000	20
	Madhu	Electrical	50000	30

Not allows, because primary key not contain null values

- **Referential Integrity Constraints:**
  - It is a rule that states that either each foreign key value must match a primary key value in another relation or the foreign key value must be null. It is used to establish relation between two tables.
  - **Foreign key**: It is used to define relationship between 2 Tables. It allows Null and duplicates values. It can be related to either Primary key or unique constraint column of other Table.
  - **Example:**





- **Key constraints:**
  - Key constraints are used to uniquely identify fields or attributes in a relation. A primary key cancontain only unique and null value in the relational table.
  - **Example:**

Empno	Empname	Deptname	Salary	deptno
101	Soohan	Medical	45000	10
102	Bhavya	IT	60000	20
	Madhu	Electrical	50000	30

Not allows, because all rows must be unique

[Q]. Explain the Concept of Keys? And also Explain various types of Keys?

Concept of Keys:

- In the relational model, keys are important because they are used to ensure that each row in a table is uniquely identified. They are also used to establish relationships among tables and to ensure the integrity of the data.
- Therefore, a proper understanding of the concepts and use of keys in the relational model is very important. A key consists of one or more attributes that determines other attributes.
- **Key:** - A key is an attribute or combination of attributes that identifies (determine) other attributes uniquely.
  - **Simple key**:
    - The key which consist only one attribute is called as "simple key".
  - **Super key:**
    - An attribute (or combination of attributes) that uniquely identifies each row in a table **(or)** Super key of an entity set is a set of one or more attributes whose values uniquely determine each entity.

- **Composite key**:
  - The key which consist multiple attributes is called as "composite" key **(or)** If a primary key contains two or more attributes then that type of primary key is called composite key.
- **Candidate Key:**
  - Candidate key is a minimal (irreducible) super key. A super key doesn't contain a subset of attributes that is itself is a super key.
- **Primary Key:**
  - The primary key is candidate key that uniquely identify all other attribute values in any given row. It cannot contain null entries.
- **Secondary Key:**
  - An attribute or combination of attributes used to strictly for data retrieval purpose.
- **Foreign Key:**
  - An attribute or combination of attributes in one table whose values must either match the primary key in another table. It is used to establish relation between two tables.
    - Used to define relationship between 2 Tables.
    - It allows Null and duplicates values.
  - It can be related to either Primary key or unique constraint column of other Table.
    PK / UNQ <-- ----> FK

[Q]. Define Relational Algebra. And write down the advantages and Limitations?

RELATIONAL ALGEBRA:

- Relational algebra is a **procedural** query language that works on relational model. The purpose of a query language is to retrieve data from database or perform various operations such as insert, update, and delete on the data.
- When I say that relational algebra is a procedural query language, it means that it tells what data to be retrieved and how to be retrieved.
- On the other hand relational calculus is a non-procedural query language, which means it tells what data to be retrieved but doesn't tell how to retrieve it.

Advantages:

- Relational algebra is based on the set theory which is a mathematical concept due to which it has a scope of development.
- Like mathematics there can be many expressions for the same operation, in a similar way if there are two relational algebraic expressions for the same operation then the query optimizer will switch to the most efficient query.
- It is a high-level query language.

Limitations:

- Relational algebra cannot perform arithmetic operations.
- It is unable to do aggregation operations even it cannot compute transitive closure.
- It cannot modify the data present in the database.

[Q]. What is Relational Algebra Operations? Explain various types of Relational Algebra Operations?

Relational Algebra Operations :

- Some operators of relational algebra are unary that is they operate only on one relation and some operators are binary that is they operate on two relations.
- SELECT, PROJECT and RENAME are the unary operators and UNION, SET DIFFERENCE, CARTESIAN PRODUCT and JOIN are the binary operators.
- The operators or the operations in relational algebra can be classified into two categories:
  - **Basic/Fundamental Operations:**
    - Select (σ)
    - Project (Π)
    - Union (U)
    - Set Difference (-)
    - Cartesian product (X)
  - **Derived Operations:**
    - Natural Join (⋈)
    - Left, Right, Full outer join (⟕, ⟖, ⟗)
    - Intersection (∩)
    - Division (÷)

Basic/Fundamental Operations:

- **Select Operator (σ) :**
  - Select Operator is denoted by sigma (σ) and it is used to find the tuples (or rows) in a relation(or table) which satisfy the given condition.
  - **Syntax of Select Operator (σ):** $\sigma_{\text{Condition/Predicate}}$ (Relation/Table_name)
  - **Example:**

    $\sigma_{\text{salary>61000}}$ (instructor)

- **Project Operator (Π) :**
  - Project operator is denoted by Π symbol and it is used to select desired columns (or attributes)from a table (or relation).
  - **Syntax of Project Operator (Π):** $\Pi_{\text{column\_list}}$ (table_name)
  - **Example:**

    $\Pi_{\text{Customer\_Name, Customer\_City}}$ (CUSTOMER)

    $\Pi_{\text{name, dept\_name}}$ (instructor);

- **Union Operator (U) :**
  - Union operator is denoted by U symbol and it is used to select all the rows (tuples) from twotables (relations) without duplicate values.
  - **Syntax of Union Operator (U):** R1 U R2
  - **Example:**

    Π Student_Name (COURSE) U Π Student_Name (STUDENT)

- **Intersection Operator (∩) :**
  - Intersection operator is denoted by ∩ symbol and it is used to select common rows (tuples)from two tables (relations).
  - **Syntax of Intersection Operator (∩):** R1 ∩ R2
  - **Example:**

$$\Pi_{Student\_Name}(COURSE) \cap \Pi_{Student\_Name}(STUDENT)$$

- **Set Difference (-) :**
  - Set Difference is denoted by – symbol. It is used to select all tuples(rows) that are present inRelation R1 but **not** present in Relation R2.
  - **Syntax of Set Difference (-):** R1 – R2
  - **Example:**

$$\Pi_{Student\_Name}(STUDENT) - \Pi_{Student\_Name}(COURSE)$$

- **Cartesian product (X) :**
  - Cartesian Product is denoted by X symbol. It is used to combine each tuple of first relation R1with the each tuple of second relation R2
  - **Syntax of Cartesian product (X):** R1 X R2
  - **Example:**

R X S

Derived Operations:

- **JOIN:**
  - The JOIN operator combines rows from two or more tables. There are several types ofjoins.
  - Consider the following tables:

PRODUCT

P_Code	P_Name	P_Price	Vendor_Code
123	Tyres	200	V123
124	Tubes	400	V124
125	Bolts	50	

VENDOR

Vendor_Code	Vendor_Name	Vendor_Address
V123	Novel Automobiles	Chennai
V124	Excel Automobiles	Mumbai
V125	New Automobiles	Hyderabad

- **Natural Join :**
  - A Natural Join joins tables by selecting the rows with common values in their commonattributes.
  - **Query:** PRODUCT NATURALJOIN VENDOR

P_Code	P_Name	P_Price	Vendor_Code	Vendor_Name	Vendor_Address
123	Tyres	200	V123	Novel Automobiles	Chennai
124	Tubes	400	V124	Excel Automobiles	Mumbai

- **Equi join :**
  - In Equijoin the tables on the basis of equality condition that compares specified columns ofeach table. In Equijoin the comparison operator '+' is used in the condition.
    (Or)
  - **Inner join** produces only the set of records that match in both Table A and Table B.
  - **Query:** PRODUCT.Vendor_code = VENDOR.Vendor_code

P_Code	P_Name	P_Price	Vendor_Code	Vendor_Name	Vendor_Address
123	Tyres	200	V123	Novel Automobiles	Chennai
124	Tubes	400	V124	Excel Automobiles	Mumbai

- **Outer Join :**
  - In Outer Join the matched pair of records would be written and any unmatched values in othertable would be NULL.
  - **Left Outer Join:** In Left Outer Join matched the records would be return and any unmatchedvalues in the other table would be NULL.
  - **Query:** PRODUCT LEFTOUTERJOIN VENDOR

P_Code	P_Name	P_Price	Vendor_Code	Vendor_Name	Vendor_Address
123	Tyres	200	V123	Novel Automobiles	Chennai
124	Tubes	400	V124	Excel Automobiles	Mumbai
124	Bolts	50			

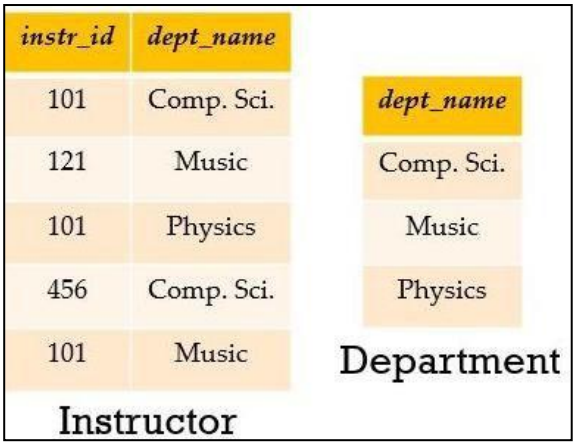
  - **Right Outer Join:** In Right Outer Join the matched records would be and any unmatchedvalues in the right table would be NULL.
  - **Query:** PRODUCT RIGHTOUTERJOIN VENDOR

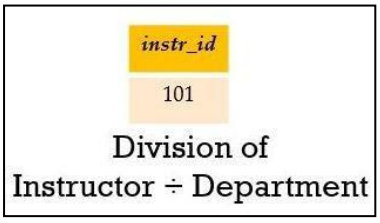
P_Code	P_Name	P_Price	Vendor_Code	Vendor_Name	Vendor_Address
123	Tyres	200	V123	Novel Automobiles	Chennai
124	Tubes	400	V124	Excel Automobiles	Mumbai
			V125	New Automobiles	Hyderabad

  - **Full Outer Join: Full outer join** produces the set of all records in Table A and Table B, with matching records from both sides where available. If there is no match, the missing side will contain null.
  - **Query:** PRODUCT FULLOUTERJOIN VENDOR

P_Code	P_Name	P_Price	Vendor_Code	Vendor_Name	Vendor_Address
123	Tyres	200	V123	Novel Automobiles	Chennai
124	Tubes	400	V124	Excel Automobiles	Mumbai
125	Bolts	50			
			V125	New Automobiles	Hyderabad

- **Divide:**
  - The DIVIDE operator uses one single column table as a deviser and two column tableas the dividend.
  - The output of DIVIDE operator is a single column with a values column-A fromthe dividend table rows where the values of the common column in both tables match.
  - **Example:**
    Now if we require the name of the instructors who teach in all the departments. Then we will apply division operation on the two relations i.e. ***instructor ÷ department***.





[Q]. What is Normalization? Explain the need for Normalization in Database Design?

Normalization:

- In the logical database design, we transform the ER diagrams into relations. Before proceeding with the physical database design we need a method to validate the logical design.
- Normalization is a primary tool to validate and improve the logical design. So Normalization is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like Insertion, Update and Deletion Anomalies.
- It is a multi-step process that puts data into tabular form by removing duplicated data from the relation tables.
- Normalization is used for mainly two purposes…
  - Eliminating redundant (useless) data.
  - Ensuring data dependencies make sense (only storing related data).

(Or)

  - Eliminating data anomalies (insertion, deletion, update).
  - Reduce data redundancy.

Need for normalization: -

- If a table have redundant data, the users have to face the errors (Or) inconsistency at the time of modifying the data. These types of errors are called **"anomalies".** Generally anomalies are three types these are
  - Insertion anomaly
  - Deletion anomaly
  - Update anomaly

Problem without Normalization :

- Without Normalization, it becomes difficult to handle and update the database, without facing data loss.
- Insertion, Updation and Deletion Anamolies are very frequent if Database is not Normalized. To understand these anomalies let us take an example of **Student** table.

S_id	S_Name	S_Address	Subject_opted
401	Aravind	Hyderabad	Comp
402	Ashok	Nellore	Maths
403	Santhi	Amaravathi	Maths
404	Aravind	Hyderabad	Physics

Types of anomalies:

- **Updation Anamoly :**
  - To update address of a student who occurs twice or more than twice ina table, we will have to update **S_Address** column in all the rows, else data will become inconsistent.
- **Insertion Anamoly:**
  - Suppose for a new admission, we have a Student id(S_id), name and address of a student but if student has not opted for any subjects yet then we have to insert **NULL** there, leading to Insertion Anamoly.
- **Deletion Anamoly:**
  - If (S_id) 401 has only one subject and temporarily he drops it, when we delete that row, entire student record will be deleted along with it.

FUNCTIONAL DEPENDENCY:

- Normalization is based on the analysis of functional dependency. A functional dependency is a constraint between two or more attributes.
- For example in a relation "R" attributes "B" is functionally dependent on the attribute "A" means the value of "A" uniquely determines the value of "B". The functional dependency is represented by a right arrow ($\rightarrow$).

  $A \rightarrow B$
- An attribute may be functionally dependence on two or more attributes. **For example** the relation STUDENT contains attribute like Sno, Sname, course_id and date completed.
- In the above example, the attribute date _complete is functionally dependent on course_Id.

  Sno, Course_Id → date completed

- **Determinant**:- The attributes on the left hand side of the arrow in a functional dependency is called determinant. In the above example determinants are Sno, Course_Id.

[Q]. Explain about Basic Normal Forms in detail?

Basic Normal Forms :

- Normalization is the process of efficiently organizing the data in database. Normalization is a primary tool to validate and improve the logical design, so that is satisfies certain conditions that avoid unnecessary duplication of data.
- It is the process decomposing relations with anomalies to produce smaller, well-structures relations.
- Normalization is used for mainly two purposes,
  - Eliminating redundant (useless) data.
  - Ensuring data dependencies make sense (only storing related data).

    (Or)
  - Eliminating data anomalies (insertion, deletion, update).
  - Reduce data redundancy.

Types of normal forms:

- First normal form (1 NF)
- Second normal form (2 NF)
- Third normal form (3NF)

First normal form (1NF):-

- A relation is said to be in First normal form if the values in the relation are atomic. Insimple words, there should be no repeating groups in particular column.
- A value can be defined as an atomic value, if it doesn't contain any multi valued attribute and no composite attribute.
- **Ex:** For example consider a table which is not in First normal form.

Student Table:

Student	Age	Subject
Aravind	15	Comp, Maths
Ashok	14	Maths
Santhi	17	Maths

- In First Normal Form, any row must not have a column in which more than one value is saved, like separated with commas. Rather than that, we must separate such data into multiple rows.

Student Table following 1NF will be:

Student	Age	Subject
Aravind	15	Comp
Aravind	15	Maths
Ashok	14	Maths
Santhi	17	Maths

- Using the First Normal Form, data redundancy increases, as there will be many columns withsame data in multiple rows but each row as a whole will be unique.
- **Rule:** Any multi- valued attributes (repeating groups) have been removed, so there is a singlevalue at the intersection of each row and column of the table.

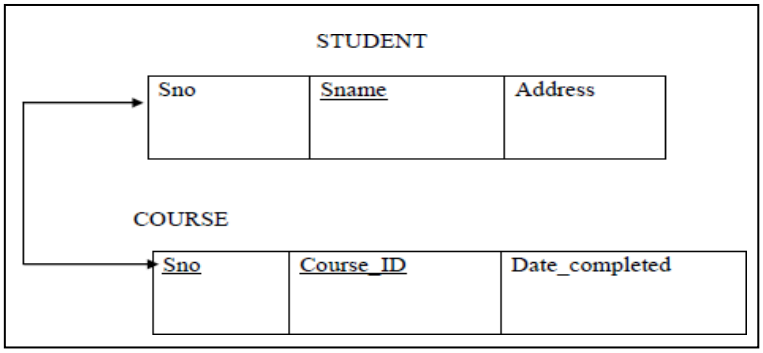
➢ Second normal form (2 NF):-

- A relation is said to be in second normal form, if it is in first normal form and contains no **partial functional dependency** (***partial functional dependency*** is a functional dependency in which one or more non-key attributes are functionally dependent on some part of the primary key).
- That means every non-key attribute is fully functionally dependent on the full set of primary key attribute. If a relation is in second normal form if any one of the following conditions applies…
  - A relation contains only one primary key.
  - No non-key attribute in the relation.
- **For example,** a relation **student** contains attribute like Sno, Sname, Course_Id and date_ completed.
- In this example, the primary key is a composite key of Sno, course_Id. Here the relation STUDENT is not in second normal form because there is a partial functional dependency.
- Here the non-key attributes Sname, Address is functionally dependent on the part of the primary key (Sno). So we decompose the above relation into new relations.

STUDENT

Sno	Sname	Address	Course_ID	Date_completed

Sno ---- > Sname (partial functional dependency)
Sno ---- > Address (Partial functional dependency)
Sno, Course_Id ---------->Date _completed (fully functional dependency)



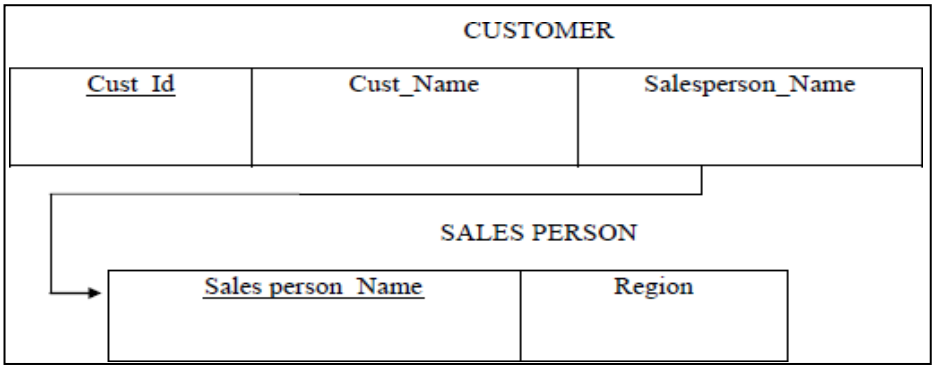
➢ Third Normal form:-

- A relation is said to be in 3rd normal form, if it is in second normal form and contains no **transitive dependency**. Here transitive dependency means a functional dependency between two or more non – key attributes.
- **For example** consider a relation **CUSTOMER** with attributes like **Cust_id, cust_name, sales person_name** and **Region**. Here the primary attribute is "**Cust_id**".

CUSTOMER

Cust_Id	Cust_name	Sales person_name	Region

- The above relation contains a transitive dependency because the non-key attribute **Region** is functionally dependent on another non-key attribute **sales person_name**.
- So the relation **CUSTOMER** is not in third normal form. So we decompose the above relation into two meaningful relations.

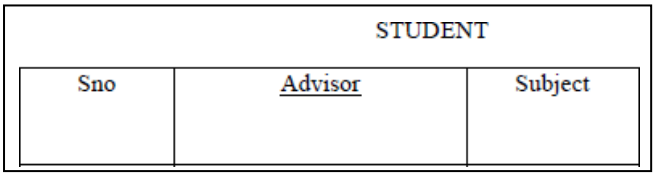


BOYCE-CODD NORMAL FORM (BCNF):-

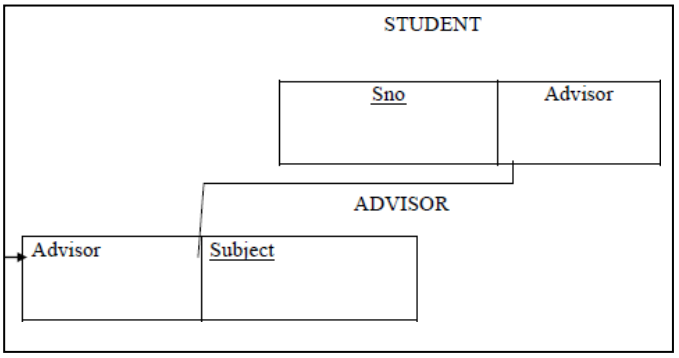
- A relation is in Boyce-codd normal form, if it is in third normal form and all determinantsare cardinalities keys (primary keys).
- **For example** consider a relation STUDENT with attributes like Sno, Subject and Advisor. Here primary key is a composite key of Sno, Subject. The above relation contains two functional dependencies.

Sno, Advisor ---------- >Subject

Subject.....> Advisor



- In the above example, in the second functional dependency determinant is subject. It is not a candidate key. So the relation STUDENT is not in Boyce Codd normal form. So we decompose the above relation into two meaningful relations.



UNIT - IV (STRUCTURED QUERY LANGUAGE)

Structured Query Language:

- SQL stands for **Structured Query Language**. It is pronounced as **SQL** or **Sequel**.
- It is a special purpose language used to define, access and manipulate data in a Relational database management system (RDBMS).
- The original version called "Sequel"(Structures English Query Language) developed by IBM in mid 1970s.
- SQL was first introduced as a commercial DBSystem in 1979 by Oracle Corporation.
- All DBMS Systems like MySQL, Oracle, MS Access, DB2, Sybase, Informix, Postgres and SQLServer uses SQL as standard database language.
- It is a powerful data manipulation language, allows as interface between user and RDBMS.

[Q]. Define Data Type. Explain the various Data Types in SQL?

Data Type:

- Data types will decide what kind of values need to be hold into the variables. Data types are used for to allocate memory for given data. The various data types available in SQL are as follows.
- In Relational Database, the data is stored in the form of tables. When we create a table, we must specify a data type for each of its column. Oracles provide built-in as well as user-defined data types.
- SQL supports following types of built-in data types
  - Number data types
  - String data types
  - Date and time data types
  - Long data type
  - Raw data type
  - Long Raw data type
  - LOB data types

➢ **Number data type:-**

Number data type stores integer and floating point values.. It has thefollowing format.

- **NUMER(P,S):**

  It is used to store numeric value. Here 'p' is precision value (from 1 - 38)and 's' is scale value (from -84 - 127). The maximum size is 38 digits. The default size is 38 digits.

- **INTEGER :**

  It is used to specify an integer value.

➢ **String data type:-**

These data types stores characters and strings up to 32k size. It can hold letters, numbers or binary data. Following are some data types of character data type.

- **CHAR(n):**

  It is used for to store alphanumeric values. it is used to store fixed length characters. Maximum size is 255 bytes and default size is 1 byte.

- **VARCHAR2(n):**
  It is used for to store alphanumeric values. It is used to store variable length strings. Maximum size is 4000 bytes
- **VARCHAR (n):**
  It is used to store variable length strings. No default size. Maximum size is 2000 bytes

➢ **Date data type:**

- **DATE :**
  Data type is used to store both date and time information ranges from 01-jan- 4712to 31-dec-9999. Default format is: DD-MON-YY. It allocates 7 fixed bytes of memory.

➢ **Long data type:**

Long data type stores variable- length character strings. It stores up to 32760 bytes of data.

➢ **Raw Data type:**

RAW data type stores fixed – length binary data with a maximum size of2000 bytes. It can hold graphic characters or digitized pictures of up to 32k.

➢ **Long Raw Data type:**

LONG RAW data type stores binary data like graphics characters ordigitized pictures. It can hold maximum of 2GB data. Hence it allows large set of binary data.

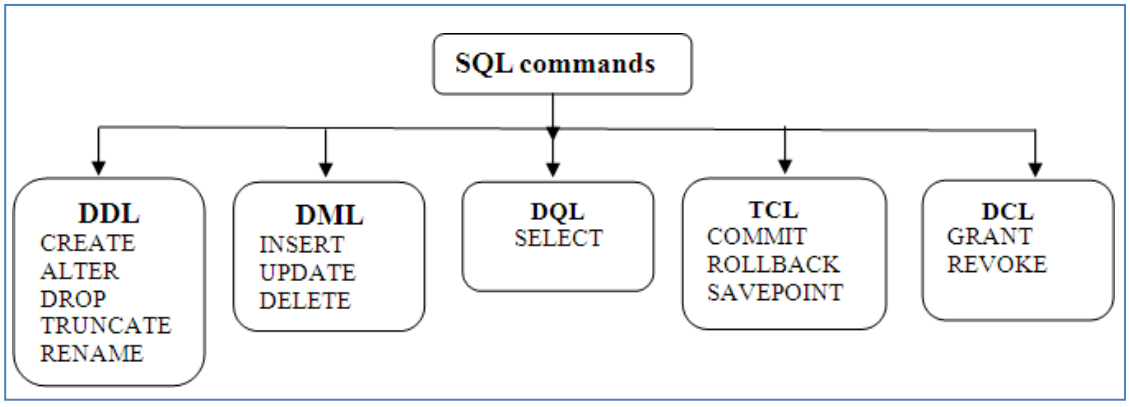
➢ **LOB:**

Large Object data type stores unstructured information like manage file, video file, and audio file etc. and its storage capacity up to 4GB.

[Q]. Explain the various types of SQL Commands?

SQL Commands:

- SQL commands are instructions used to communicate with the Database to perform specialtask on data. SQL commands are divided into following categories:



➢ **Data Definition Language (DDL):**

- These are used to define various database objects i.e. tables, views, indexes etc. We can create, modify, delete and view the database objects by using these commands.
- The DDL commands are Auto Commit commands, since they process and save objects automatically in the database.

- DDL commands like…
  - **CREATE –** is used to create the database and its objects (like table, index, views).
  - **DROP –** is used to delete objects (i.e. table) from the database.
  - **ALTER**-is used to modify the structure (i.e. columns) of the database.
  - **TRUNCATE–**is used to remove all records and releases storage space.
  - **RENAME –**is used to rename a table.

➢ **Data Manipulation Language (DML):**

- These commands are used to manipulate data within the ORACLE database. They are used to insert, delete, update, and query the data in the database objects.
- These commands do not implicitly commit the current transaction.
- DML commands like…
  - **INSERT –** is used to insert data values(i.e rows) into a table.
  - **UPDATE** – is used to update existing data within a table.
  - **DELETE –** is used to delete only records (i.e. rows) from a table.

➢ **Data Retrieval/Querying Language (DRL/DQL):**

- These SQL commands are used for retrieving or selecting data from Database.
- DRL commands like…
  - **SELECT –** is used to retrieve data from the database.

➢ **Transaction Control Language (TCL):**

- Transaction Control statements manage changes made by Data Manipulation Language commands. These commands are used to save or cancel the process of DML commands.
- TCL commands like…
  - **COMMIT**– permanently saves data changes.
  - **ROLLBACK–**restores data to their original values in case of any error occurs.
  - **SAVEPOINT**–sets a save point.

➢ **Data Control Language (DCL) :**

- These commands are used to control user access to the database objects. They used to grant and remove various privileges and roles to users.
- DCL commands like…
  - **GRANT –**gives user's access permissions to database.
  - **REVOKE –** undo user's access permissions given by GRANT command.

[Q]. Explain the DDL Commands with an example?

DATA DEFINITION LANGUAGE (DDL) COMMANDS :

- DDL stands for **Data Definition Language**. DDL commands are used to create, modify and remove the databases and database objects like tables, views etc.
- A new table is created using CREATE command followed by table name and column definitions. Once table is created, it can be modified using ALTER command. If the table is no need, the DROP command is used to delete the table.

- Some of the DDL commands are CREATE, ALTER, DROP, TRUNCATE, RENAME.
  - **CREATE Command :**

    The **CREATE TABLE** command is used to create tables to store data.

    Syntax:

    SQL>**CREATE TABLE** <*table_name*> (*column1 datatype,column2 datatype,…….column-n datatype);*

    Example:

    SQL> create table student (sno number(2), sname varchar2(10), marks number(3));

  - **ALTER Command :**

    The **ALTER TABLE** command is used to perform add, modify, delete and rename columns in anexisting table.

    - **Adding a column :**

      Syntax: **ALTER TABLE** *table_name* **ADD *COLUMN*** *datatype(size)*;

      Ex: SQL>ALTER TABLE student ADD sdob date;
    - **Modifying a column :**

      Syntax: **ALTER TABLE** *table_name* **MODIFY *COLUMN*** *datatype*;

      Ex: SQL> ALTER TABLE student MODIFY sname varchar2(20,2);
    - **Removing a column :**

      Syntax: **ALTER TABLE** *table_name* **DROP COLUMN** *column datatype*;

      Ex: SQL>ALTER TABLE student DROPCOLUMNsdob;
    - **Renaming a column :**

      Syntax: **ALTER TABLE** <*table_name*> **RENAME COLUMN** <*old_column_name*> **TO** <*new_column_name*>;

      Ex: SQL>ALTER TABLE student RENAME COLUMN marks to smarks;

  - **TRUNCATE Command:**

    The **TRUNCATE TABLE** command is used to remove all rowsand to release storage space from atable. It is similar to the DELETE command without WHERE clause.

    Syntax: **TRUNCATE TABLE** *table_name*;

    Ex: SQL>TRUNCATE TABLE student;

  - **DROP Command:**

    The **DROP TABLE** command is used to removes structure and all the data of a table.

    Syntax: **DROP TABLE** *table_name*;

    Ex: SQL>DROP TABLE student;

  - **RENAME Command:**

    The **RENAME** command is used to change name of the existing table.

    Syntax: **RENAME** old_*table_name* **TO** *new_table_name*;

    Ex: SQL>RENAME student TO stud;

[Q]. Explain about DML Commands with examples?

DATA MANIPULATION LANGUAGE (DML) COMMANDS :

- DML stands for **Data Manipulation Language**. DML commands are used to insert, delete and modify the data in database. Some of the DML commands are INSERT, UPDATE and DELETE.

➢ **INSERT Command:**

- The **INSERT INTO** command is used to insert new rows into a table. There are two methods toinsert values

Syntax: SQL>INSERT INTO *table_name* VALUES (*value1, value2, …. Valuen*);
(or)
SQL>INSERT INTO *table_name* VALUES *(&col1, &col2, ….&coln*);

Example: SQL>INSERT INTO student values (1, 'rama', 100);
SQL>INSERT INTO student VALUES (&no, '&name', '&sdob', &marks);

➢ **UPDATE Command:**

- It is used to update or modify the existing data (i.e. rows)in a table. It modifies specific rows based on the condition given in WHERE clause or modifies all rows when condition is not specified.

Syntax:

UPDATE *table_name* **SET** *col1* = val1, *col2* = val2……. *Col n* = val n **WHERE** *condition*; (Or)
UPDATE *table_name* **SET** *col1* = value1, *col2* = value2…*col n* = value n;

Example: SQL>UPDATE student SET marks = 500;
SQL>UPDATE student SET marks = 500 WHERE no = 2;
SQL>UPDATE student SET marks = 500, name = 'rama' WHERE no = 1;

➢ **DELETE Command :**

- The DELETE is used to delete only rows from a table based on the condition given in the WHEREclause or deletes all the rows when condition is not specified.

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

Example : SQL>DELETE student;
SQL>DELETE student WHERE no = 2;

[Q]. Explain about Table Modification Commands (Or) Alter command with examples?

ALTER COMMAND:

- The **ALTER TABLE** command is used to perform add, modify, delete and rename columns in anexisting table.
- **Adding a column**

It is used to add a column to a table using ALTER command along with ADD clause.

Syntax: **ALTER TABLE** *table_name* **ADD *COLUMN*** *datatype(size)*;

Ex: SQL> ALTER TABLE student ADD sdob date;

- **Modifying a column**

  It is used to modify a column datatype using ALTER command along with MODIFY clause.

  Syntax: **ALTER TABLE** *table_name* **MODIFY** ***COLUMN*** *datatype(size)*;

  Ex: SQL> ALTER TABLE student MODIFY sname varchar2(20,2);

- **Removing a column**

  It is used to remove a column using ALTER command along with DROP COLUMN.

  Syntax: **ALTER TABLE** *table_name* **DROP COLUMN** *column_name*;

  Ex: SQL> ALTER TABLE student DROP COLUMN sdob;

- **Renaming a column**

  It is used to rename a column name using ALTER command along with RENAME COLUMN.

  Syntax: ALTER TABLE *table_name* **RENAME COLUMN** *old_col_name* **TO** *new_col_name*;

  Ex: SQL> ALTER TABLE student RENAME COLUMN marks TO smarks;

[Q]. Explain about Relational Set Operators in SQL with examples (SET Operators)?

SET OPERATORS IN SQL:

- The set operations are used to combine two sets of rows (i.e. SELECT statements) into a single set. Some of the set operators are – **UNION, UNION ALL, INTERSECT, MINUS.**
- Consider the following two tables:

  SQL> create table person1(pid number(2),pname varchar2(20));

  SQL>create table person2(pid number(3),pname varchar2(20));

➢ **UNION :**

- The **UNION** operator is used to combine two SELECT statements without duplicate rows. In caseof UNION, no. of columns and datatype must be same in both tables.

  Syntax: **SQL>**SELECT columnlist FROM table1 WHERE condition **UNION**
  SELECT columnlist FROM table2 WHERE condition;

  Example: SQL>select *from person1 union select *from person2;

➢ **UNION ALL:**

- The **UNION ALL** operator is used to combine two SELECT statements with duplicate rows.

  Syntax: **SQL>**SELECT column list FROM table1 WHERE condition **UNION ALL**
  SELECT column list FROM table2 WHERE condition;

  Example: SQL> select *from person1 union all select *from person2;

➢ **INTERSECT:**

- The **INTERSECT** operator is used to returns only the common rows from two SELECT statements.

  Syntax: **SQL>**SELECT column list FROM table1 WHERE condition **INTERSECT**
  SELECT column list FROM table2 WHERE condition;

  Example: SQL> select *from person1 intersect select *from person2;

- **MINUS:**
  - The **MINUS** operator is used to returns rows only by the first query(or statement) that are notpresent in the second query(or statement)

Syntax: **SQL>**SELECT column list FROM table1 WHERE condition **MINUS**
SELECT column list FROM table2 WHERE condition;

Example: SQL> select *from person1 minus select *from person2;

[Q]. Explain about Selection and Projection Operations with examples?

PROJECTION OPERATION:

The projection operation selects specific columns from a table(or relation)and other columns are discarded. We use the Greek letter '$\prod$' to denote the projection. The column name appears as a subscript of 'π'. The argument 'relation' is given in the parenthesis following the 'π'. The projection operation can be considered as column-wise filtering.

Notation: $\prod_{A1,A2,\ldots An}(R)$ *(or)* $\pi_{(\text{attribute list})}(\text{Relation})$

Where $\prod$ stands for projection predicate and **R** stands for relation, A1,A2... are attribute names

For example –

- $\prod$ subject, author (Books)
  Selects and projects columns named as subject and author from the relation Books.
- $\prod$ dob, empno(Employee)
  Selects and projects columns named as dob and empno from the relation Employee
- To display the firstname, lastname and salary of all employees who works in department number >4.
  $\prod_{firstname,lastname,sal}(\sigma_{dno>4}(employee))$

SELECTION OPERATION:

- This operation is used to select rows from a table (or relation) that specifies a given logic, which is called as a predicate. In relational algebra, it is denoted by the symbol σ (lower case sigma).
- The **condition** (Predicate) appears as a subscript to σ. The argument "relation" is given in parenthesis following to σ. The selection operation can be considered as row-wise filtering.

Notation: $\sigma_{p}(R)$ *(or)* $\sigma_{predicate}(R)$ (or) $\sigma_{selection\ condition}(R)$

- Where, σ represents the Select operation, R is the name of relation (i.e. table name). The comparison (<, >, ≤, ≥, =, ≠) and logical (AND, OR, NOT) operators are allowed in conditions(i.e. in predicate).

- **Example :**

$\sigma_{dno=4}(employee)$

Select the EMPLOYEE tuples whose dept is 4

$\sigma_{dno=4\ AND sal>25000}($

Select the tuples for employees who work in dept 4 and salary > 25000 per year

[Q]. Explain about Join Operations (Joins) in SQL with examples?

JOINS:

- The JOIN is used to combine the rows from two or more tables. The common key field is specified and common columns must be same datatype.
- Several operators are used to join tables, like =, <, >, <>, <=, >=, !=, BETWEEN, LIKE, andNOT but the most common operator is the **equal to** (=)symbol.

TYPES OF JOINS:

- There are different types of joins available in SQL like…
  - INNER or EQUI JOIN
  - OUTER JOIN
  - CARTESIAN JOIN
  - NON EQUI JOIN
  - SELF JOIN
- Consider the following tables:

```
SQL> select * from dept4;
   DEPTNO DNAME      LOC
   ------ ---------- ----------
       10 mkt        hyd
       20 fin        bang
       30 hr         bombay

SQL> select * from emp4;
    EMPNO  ENAME      JOB       MGR    DEPTNO
   ------- ---------- --------- ------- -------
      111  saketh     analyst       444      10
      222  sudha      clerk         333      20
      333  jagan      manager       111      10
      444  madhu      engineer      222      40
```

- **INNER JOIN (or) EQUI JOIN :**
  - The INNER JOIN is also known as EQUI JOIN. It display all the matched records from tables based on join condition using '=' operator.
  - **Example:**

SQL> select empno,ename,job,dname,loc from **emp4 e,dept4 d** where e.deptno=d.deptno;

EMPNO	ENAME	JOB	DNAME	LOC
111	saketh	analyst	mkt	hyd
333	jagan	manager	mkt	hyd
222	sudha	clerk	fin	bang

- **LEFT OUTER JOIN**
  - It displays all the matching records and the records which are not matching in left hand side table.
  - **Example :**

SQL> select empno,ename,job,dname,loc from **emp4 e LEFT OUTER JOIN dept4 d**on(e.deptno=d.deptno);

EMPNO	ENAME	JOB	DNAME	LOC
111	saketh	analyst	mkt	hyd
333	jagan	manager	mkt	hyd
222	sudha	clerk	fin	bang
444	madhu	engineer	NULL	NULL

- **RIGHT OUTER JOIN :**
  - It displays all the matching records and the records which are not matching in right hand side table.
  - **Example :**

SQL> select empno,ename,job,dname,loc from **emp4 e RIGHT OUTER JOIN dept4 d** on(e.deptno=d.deptno);

EMPNO	ENAME	JOB	DNAME	LOC
111	saketh	analyst	mkt	hyd
333	jagan	manager	mkt	hyd
222	sudha	clerk	fin	bang
NULL	NULL	NULL	hr	bombay

- **FULL OUTER JOIN :**
  - This will display the all matching records and the non-matching records from both tables.
  - **Example :**

SQL> select empno,ename,job,dname,loc from **emp4 e FULL OUTER JOIN dept4 d** on(e.deptno=d.deptno);

EMPNO	ENAME	JOB	DNAME	LOC
333	jagan	manager	mkt	hyd
111	saketh	analyst	mkt	hyd
222	sudha	clerk	fin	bang
444	madhu	engineer	NULL	NULL
NULL	NULL	NULL	hr	bombay

- **CARTESIAN JOIN (or) CROSS JOIN :**
  - The CARTESIAN JOIN (or) CROSS JOIN returns the Cartesian product. In cross join, all rows in the first table are joined to all rows in the second table.
  - **Example:** SQL> select *from emp4,dept4;

(or)

SQL> select empno,ename,job,dname,loc from emp4 **CROSS JOIN** dept4 ;

EMPNO	ENAME	JOB	DNAME	LOC
111	saketh	analyst	mkt	hyd
222	sudha	clerk	mkt	hyd
333	jagan	manager	mkt	hyd
444	madhu	engineer	mkt	hyd
111	saketh	analyst	fin	bang
222	sudha	clerk	fin	bang
333	jagan	manager	fin	bang
444	madhu	engineer	fin	bang
111	saketh	analyst	hr	bombay
222	sudha	clerk	hr	bombay
333	jagan	manager	hr	bombay
444	madhu	engineer	hr	bombay

[Q]. Define View. Explain the detailed note on Views with examples?

VIEWS:

- A view is a ***virtual table*** *or* ***pseudo table***. That is, a view looks like a table and does not store data, it just displays the data. A view is a named and validated SQL query that is stored in the database. A View can be created from a single table, multiple tables, or another view.

Creating a View:

- Views are created from a single table using **CREATE VIEW** command.
- **Syntax:**

SQL>**CREATE VIEW** *viewname* **AS SELECT** col_name1,col_name2,……., col_name n **FROM** *table_name* **WHERE** condition;

- **Example:**

SQL> create table customers(id number(5),name varchar2(10),age number(5),addressvarchar2(10),salary number(5));

SQL> select *from customers;

SQL > CREATE VIEW **customers_view** AS SELECT name, age FROM customers;

SQL > SELECT * FROM **customers_view**;

Views from multiple tables:

- Views from multiple tables are called as **complex views**. We can create a view frommultiple tables by using a **JOIN** in the **SELECT** statement.

- **Example:**

```
SQL> create table STAFF (lecid varchar2(10),name varchar2(10),position
        varchar2(10));
SQL>create table COURSE (courseid varchar2(10),coursename
        varchar2(10),lecid varchar2(10));
SQL>create view COURSE_STAFF AS SELECT coursename, STAFF.name from
        COURSE, STAFF where COURSE.lecID=STAFF.lecID;
SQL> SELECT *FROM COURSE_STAFF;
```

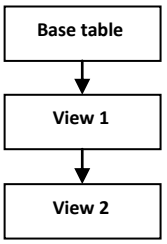
View from view:

- We can create a view from another view using the following format…

  SQL>CREATE *VIEW2* **AS** SELECT * FROM **VIEW1**

- **Example:**

```
SQL> create table STAFF(empid varchar2(10),ename varchar2(10),
     deptnamevarchar2(10),salary number(5),age number(5));
SQL> select *from STAFF;
```



Case 1- creating **ITSTAFF** view from **STAFF** table:

```
SQL> create view ITSTAFF AS SELECT * from STAFF where deptname='IT';
```

Case 2–creating **SENIORSTAFF** view from **ITSTAFF** view:

```
SQL> create view SENIORSTAFF AS SELECT * from ITSTAFF where age>30;
```

Dropping Views:

- The DROP VIEW is used to drop or delete a view from database. Dropping a view has not affecton base tables.
- **Syntax:** SQL>DROP VIEW viewname;
- **Example:** SQL>DROP VIEW CUSTOMERS_VIEW;

[Q]. Explain the different types of Views with examples?

VIEW: A view is a named and validated SQL query that is stored in the database.SQL contains following types of views…

- Simple View
- Complex View
- Inline view
- Materialized View

- **Simple View:**
  - A Simple view derives data from only one table. A simple view does not contain functions or group of data. A simple view always allows DML operation through the view.

- **Complex View:**
  - A Complex view derives data from many tables. These contain function or group of data. Complex views can be constructed on more than one base table. A complex view can contain **JOIN** conditions, **GROUP BY** clause, **ORDER BY** clause
  - **Example:**

    SQL>CREATE VIEW view3 AS SELECT e.* FROM empe,dept d WHERE e.deptno = d.deptno;
- **Inline View:**
  - An inline view is a SELECT statement in the FROM clause of another SELECT statement.
  - **Example:**

    SQL> SELECT *FROM (SELECT deptno FROM emp)emp,dept WHERE dept.deptno = emp.deptno;

[Q]. Define Sub Queries. Explain the types of Sub Queries (or) Nested Queries?

Sub Queries or Nested Queries:

- A query within another query is called a sub query. We can define any number of sub queries with in a query. But the system executes the inner most query first, based on the inner query output, outer query will be executed.
- **Syntax:**

  SQL>SELECT *column_name* FROM *table* WHERE *column_name*= (SELECT *column_name* FROM *table* WHERE *conditions*);

Rules:

- Sub queries enclosed within parenthesis.
- Sub query must be placed in the right hand of the comparison operator.
- Sub query cannot contain a ORDERBY clause. GROUP BY is used instead of ORDER BY ina sub query.
- A query can contain more than one sub-queries.
- Sub query returns a single row when a standard operator such as =, >, < is used.
- Sub query cannot enclose in a set function.
- BETWEEN operator cannot be used with a sub query.
- Sub queries can be used in Insert, Update and Delete Statements.

TYPES OF SUBQUERIES:

- SQL supports following types of sub queries like….

Single row Sub queries:

- In single row sub query, it will return only one value. The following operators are used withsingle row sub queries.< , >, <=, >=,==, !=
- **Example: Display all the employees whose job same as allen job.**

SQL> select * from emp where job=(select job from emp where ename="allen");

Display all the employees who belongs to smith employee department.

SQL>select * from emp where deptno=(select deptno from emp where ename='smith');

Multi row sub queries:

- In multi row sub query, it will return more than one value. In such cases we should includeoperators like any, all, in or not in between the comparison operators are used.

- **Example: Display details of employee whose salary greate than the range of 20000 and 30000.**

```
SQL> Select *from emp where sal > any ( select sal from emp where sal between
        20000 and30000);
```

Retrieve that department names in which department does not have any employee.

```
SQL> select *from dept where deptno not in (select distinct(deptno) from emp);
```

Multiple sub queries:

- There is no limit on the number of sub queries included in a where clause. It allows nesting of aquery within a sub query.
- **Example:**

Display details of employee who are getting maximum salry in the employee table.

```
SQL> select * from emp where sal = (select max(sal) from emp where sal <
        (selectmax(sal) fromemp));
```

UNIT –V (PL/SQL)

PL/SQL :

- PL/SQL stands for "Procedural Language extensions to SQL" developed by Oracle Corporation in the early 90's to increase the capabilities of SQL.
- It is a combination of SQL along with the procedural features of programming languages. It allows the users and designers to develop complex database applications using control structures and procedural elements like procedures, functions, and modules.

[Q]. Write about the Advantages of PL/SQL?

PL/SQL Advantages:

- ***Support for SQL:*** PL/SQL allows us to use all SQL commands, SQL functions, operators and data types.
- ***Block Structure***: PL/SQL is a block-structured language. Each program in PL/SQL is written as a block. Each block performs a task. PL/SQL Blocks are stored in the database and reused.
- ***Control structure:*** PL/SQL contains procedural language constructs like conditional statements and looping statements.
- ***Better Performance***: PL/SQL engine processes multiple SQL statements simultaneously as a single block, thereby reducing network traffic.
- ***Modularity:*** PL/SQL allows process to be divided into different modules like procedures and functions, called as *subprograms*.
- ***Portability:*** The programs written in PL/SQL are portable to any platform
- ***Error Handling***: PL/SQL handles errors (or) exceptions during the execution of a PL/SQL program. Once an exception is found, specific actions can be performed depending on the type of the exception or it can be displayed to the user with a message.

[Q]. Explain the Structure of PL/SQL Program?

Structure of PL/SQL Program:

- PL/SQL is a block-structured language. This means, programs are divided into logical units called as ***Blocks***. In PL/SQL, the named blocks are called as ***sub-programs*** and unnamed blocks are called as ***anonymous blocks***.
- The anonymous block is the simplest unit in PL/SQL. It is called anonymous block because it is not saved in the Oracle database.
- Sub-programs can be referred as either ***functions*** or ***procedures.*** Subprograms can be nested within one another and grouped in larger units called ***packages***.
- A PL/SQL program contains three sections like…
    - Declaration section
    - Execution section
    - Exception handling section

- **Syntax (or) Structure of PL/SQL :**

```
DECLARE
        Declaration statements;
BEGIN
        Execution statements;
EXCEPTION
        Exception handling statements;
END;
```

➢ **Declaration Section :**

- It is the first section of the PL/SQL block. This section is an optional part.
- The Declaration section starts with the keyword 'DECLARE'.
- In this section, we define variables, constants, cursors, exceptions, subprograms whichare used to manipulate data in the execution section.
- This section followed by execution section.

➢ **Execution Section :**

- It is the main part of PL/SQL that used to executes the code.
- This section starts with 'BEGIN' and ends 'END' keyword.
- This section contains both SQL and PL/SQL code and also controls statements.
- This section followed by Exception-Handling section.

➢ **Exception-Handling Section :**

- It is the last part and optional section of the PL/SQL blocks
- It is used for handling runtime errors and warnings. It also contains PL/SQL statements.
- This section starts with the keyword 'EXCEPTION'.

Example:

```
SQL> set serveroutput on;
SQL> BEGIN
        dbms_output.put_line ('Hello World');
        END;
         /
```

[Q]. Explain about writing and executing PL/SQL Program with an example?

Writing a PL/SQL program:

While writing a PL/SQL program

- Place a semi-colon at the end of every pl/sql statement.
- The keywords DECLARE,BEGIN and EXECPTION are not followed by semi-colon.
- The END keyword always followed by semi-colon
- When the block is executed successfully, then the message output is as follows:

PL/SQL procedure successfully completed.

Execute PL/SQL program:

To display database's output on the screen, follow the below steps:

- First, use the **SET SERVEROUTPUT ON** command to instruct SQL*Plus to echo database'soutput after executing the PL/SQL block.
- Second, use the **DBMS_OUTPUT.PUT_LINE** procedure to output a string on the screen.

Example 1:

```
SQL>Set serveroutput on
SQL>BEGIN
        dbms_output.put_line ('Hello World..');
    END;
    /
```

Example 2:

```
SQL>Set serveroutput on
SQL>DECLARE
    text VARCHAR2(25);BEGIN
     text:= 'Hello World';
    dbms_output.put_line (text);
    END;
```

[Q]. Explain about the Elements of PL/SQL?

ELEMENTS OF PL/SQL:

- PL/SQL contains set of elements like character set, reserved words, identifiers, literals etc.

➢ **Character set :**

- PL/SQL programs are written by using a specific set of characters.
- The PL/SQL character set contains upper and lower case letters, numbers, Tabs, whitespaces, carriage returns, Special symbols etc.

➢ **Reserved words :**

- A reserved word has special meaning and it cannot be changed. Reserved words are written in uppercase or lower or mixed cases.
- Example: BEGIN, DECLARE, END

➢ **Delimiters :**

- Delimiters are simple or compound symbols that have special meaning to PL/SQL.
  - Simple symbols are like +, - , * , / , = , @(remote access indicator) , ;(statement terminator)
  - Compound symbols are like <, >, !=, ||(concatenation) , :=(assignment) , --(single line comment), /* …. */ (multi line comment).

➢ **Identifiers :**

- In PL/SQL, Identifiers are used for naming constant, variable, exception, cursors,

procedures, function, package, trigger, label etc.

Properties of Identifiers:

- Identifiers must start with a letter
- It contains dollar sign ('$'), underscore ('_') and hash sign ('#')
- whitespace, hyphens, slashes are not allowed
- It is case-insensitive

➢ **Literals :**

- A literal is a numeric, character, string, or Boolean value not represented by an identifier (or A literal refers to fixed value that cannot changed during the program execution).
- PL/SQL, literals are case-sensitive.PL/SQL supports the following types of literals like…
  - Numeric Literals
  - Character Literals
  - String Literals
  - BOOLEAN Literals
  - Date and Time Literals

[Q]. Write about Variables and Constants in PL/SQL with examples?

VARIABLES IN PL/SQL :

- In PL/SQL, variable is an identifier used to store the value that can be changed during the execution of PL/SQL program.
- To declare a variable, use a variable name followed by the data type and terminated by a semicolon (;).
- PL/SQL variables must be declared in the declaration section as follows…
- **Syntax:** Variable_name datatype;

(or)

variable_name datatype := default_value;

(or)

variable_name datatype [not null := default_value] ;

Rules :

- Variable name contains alphabets, numbers, dollar signs, underscores.
- Variable names are case-insensitive.
- Keywords are not used as a variable name.
- Variables must be declared in the declaration section.

Examples:

```
Set serveroutput on
DECLARE
    a number := 10;
    b number := 20;
    c number;
BEGIN
    c := a + b;
```

```
        dbms_output.put_line('Addition is: ' || c );
    END;
```

CONSTANTS IN PL/SQL:

- A constant refers to fixed value that does not changed during the execution of PL/SQL program.
- In PL/SQL, a constant is declared using the **CONSTANT** keyword. It requires an initial value and does not allow to change that value.
- **Syntax:** constant_name CONSTANT datatype := value ;
- **Example:**

```
set serveroutput on
  DECLARE
        pi CONSTANT real := 3.14159;
        radius REAL := 3;
        area REAL := (pi * radius**2);
  BEGIN
        dbms_output.put_line('PI:' || pi || 'Radius:' || radius);
        dbms_output.put_line('Area:' || area);
  END;
  /
```

[Q]. Explain about various Data types available in PL/SQL?

DATA TYPES IN PL/SQL :

- PL/SQL supports several types of data types that are used for declaring variables. The data type specifies the size and what type of value to be stored in a variable.
- **Types of Data types:**
  - **Scalar data type:–** It is used to store fixed or floating point numbers. **Example:**number(p),number(p,s),integer,int,real,float,char,character,long,varchar ,varch ar2,date, Boolean
  - **Large Object (LOB) data type: -** It is used to store large objects like text, graphic, images,video clips and sound forms.

Scalar Data Types :

- **NUMERIC Data Type:**
  - This datatype is used to store fixed or floating point numbers up to 38 digits of precision (or length). It contains number(p),number(p,s) decimal, real, float, etc.
  - **Example:** A number(5,2); B number(5); C number;
- **Character Data Type:**
  - This datatype is used to stores alphanumeric values in string format. The values of Character datatype are placed between single quotes. It contains char (max. size 1byte), varchar (max. size 2000 bytes), varchar2 (max. size 4000 bytes) etc.
  - **Example:**

```
grade CHAR;
name  CHAR(10):= 'reddy';
name VARCHAR2(10) := ‘reddy';
```

- **BOOLEAN Data Type:**
  - The Boolean data type can store only true, false and null. The default value for this data typeis "False".
  - **Example:** var1 BOOLEAN;
- **DATE Data Type:**
  - This data type is used to store both date and time values. The default data format is 'DD-MM-YY'. Valid dates range from January 1, 4712 BC to December 31,9999 AD. It allocate 7 fixed bytes of memory
  - **Example:** newyear DATE:='01-JAN-2019';
    current_date DATE:=SYSDATE;

LOB Data Type:

- The LOB data type is used to store large objects like images, multimedia files, etc. LOB is usedinstead of the LONG data type.
  - **BFILE:** It is used to store large binary objects in to operating system file. The size cannotexceed 4GB. BFILE data type read only.
  - **BLOB:** It is used to store large unstructured binary objects in to OS file. Memory Capacity is 8 to 128 TB.
  - **CLOB:** It is used to store large blocks of character data in the database. Memory Capacity is 8 to128 TB.
  - **NCLOB:** It is Used to store large blocks of NCHAR data in the database. Memory Capacityis 8 to128 TB.

COMPOSITE DATA TYPES (PL/SQL attributes):

- It is used to define the PL/SQL variables dynamically according to table structure.

- **%Type**: -
  - It is column type declaration. It is used to define the variables according to the specific column structure.
  - **Syntax**: variable <table name>. <Column name>;
  - **Example**: veno emp.empno%type;
- **%Row type**: -
  - It is record type declaration. It is used to define the variable according to the complete table structure.
  - **Syntax**: variable <table name> %row type;
  - **Example:** I emp%rowtype;

[Q]. Explain about Control Structures in PL/SQL with examples?

CONTROL STRUCTURES in PL/SQL :

- In PL/SQL, the control structure allows us to control the behavior of the block. PL/SQL supportsfollowing types of control structures.

- Conditional control structures.
- Iterative (or) Looping control structures.
- Case expressions.

CONDITIONAL CONTROL STRUCTURES :

- In PL/SQL, conditional control structures are used to perform actions based on specificconditions. PL/SQL supports following types of conditional control structures
  - IF-THEN
  - IF-THEN-ELSE
  - IF-THEN-ELSIF

IF-THEN Statement :

- The IF-THEN statement is used to execute sequence of statements based on condition.
- If thecondition is true, then statements inside the IF block is executed, otherwise it does nothing.

Syntax:	**Example:**

IF <condition> THEN
statements;
END if;
	SQL> Set serveroutput onSQL>
	DECLARE
	a NUMBER :=10;
	BEGIN
	IF(a > 10) THEN
	dbms_output.put_line('a is greater than10');
	END IF;
	END;
	/

IF-THEN-ELSE Statement :

- It is extension of IF statement. It is similar to IF but additionally it contains ELSE block. If the condition is TRUE, then the statements inside the IF block is executed, otherwise the statements inside the ELSE block is executed.

- Syntax :

```
IF <condition> THEN
        Statements;
ELSE
        Statements;
END IF;
```

Example:	**Example:**

SQL>Set serveroutput on
SQL>DECLARE
a NUMBER:=11;
BEGIN
IF(mod(a,2)=0) THEN
	SQL>Set serveroutput on
	SQL>DECLARE
	a NUMBER :=10;
	b NUMBER:=20;
	BEGIN

dbms_output.put_line('a is even');
ELSE
dbms_output.put_line('a is odd);
END IF;
END;
/
	IF(a > b) THEN
	dbms_output.put_line('a is greater than b');
	ELSE
	dbms_output.put_line('b is greater than a');
	END IF;
	END;

☞ **IF-THEN-ELSIF Statement :**

- The IF-THEN-ELSIF statement is used to select one alternative from several alternatives, where each alternative has its own conditions to be satisfied. If the condition is FALSE or NULL, thenELSIF block is tested.
- If any condition is true, the statements in that block are executed and control passes to the next statement. Otherwise, the statements in the ELSE block is executed.
- **Syntax:** IF condition 1 THEN
  Statements;
  ELSIF condition 2 THEN
  Statements;
  ELSE
  Statements;
  END IF;

Example:
SQL>Set serveroutput on
SQL>DECLARE
a NUMBER;
b NUMBER;
c NUMBER;
BEGIN
a:=&a;
b:=&b;
c:=&c;
	IF(a > b and a>c) THEN
	dbms_output.put_line('a is big');
	ELSIF (b>a and b>c) THEN
	dbms_output.put_line('b is big');
	ELSE
	dbms_output.put_line('c is big');
	END IF;
	END;

➢ **ITERATIVE (OR) LOOPING CONTROL STRUCTURES :**

- Iterative or looping control statements are used to execute sequence of statements for specificno. of times. PL/SQL supports following types of looping statements…
  - Simple loop
  - WHILE loop
  - FOR loop

–

☞ **Simple loop :**

- It is the simplest loop structure in PL/SQL. The sequence of statements are placed inside theLOOP and END LOOP. The **EXIT** or **EXIT WHEN** is used to break the loop.
- **Syntax: LOOP**

```
        Sequence of statements;
    END LOOP;
```

- **Example: SQL>set serveroutput on**

```
SQL> declare
      a number :=0;
       begin
          loop
               a := a+1;
              dbms_output.put_line(' value of a is: '||a);
               exit when a>5 ;
          end loop;
       end;
```

☞ **WHILE LOOP Statement :**

- A **WHILE LOOP** statement is used to execute a statement repeatedly until given condition is true.
- First, the condition is tested. If the condition is TRUE, the sequence of statements is executed,otherwise loop is terminated and control passes to the next statement.

Syntax:	**Example:**

WHILE condition LOOP
Statements;
END LOOP;
	SQL> set serveroutput on
	SQL> DECLARE
	n NUMBER := 0;
	BEGIN
	WHILE n < 10 LOOP
	n := n + 1;
	END LOOP;
	DBMS_OUTPUT.PUT_LINE('Sum :' || n);
	END;
	/

☞ **FOR LOOP Statement :**

- A **FOR LOOP** is used to execute the code for a no. of times repeatedly.
- **Syntax : FOR counter IN [REVERSE] initial_value ..final_valueLOOP**

```
        statements;
    END LOOP;
```

Example	**Example**
SQL> Set serveroutput on SQL> DECLARE	SQL> SET SERVEROUTPUT ON; SQL> DECLARE

```
a number(2);
BEGIN
FOR a IN10 .. 20 LOOP
dbms_output.put_line(a);
END LOOP;
END;
/
```

```
n NUMBER := 10;
BEGIN
FOR i IN REVERSE 1..n LOOP
DBMS_OUTPUT.PUT_LINE(i);
END LOOP;
END;
/
```

- **CASE statements :**

  - **CASE statement:**
    - A CASE statement is similar to IF-THEN-ELSIF statement. The CASE statement selects one alternative among several alternatives.
    - The CASE statement uses **"selector"** to select the alternatives instead of Boolean expression. Each alternative is assigned with a predefined value.
    - If the **value** is matched with **selector value** then the statements inside WHEN clause is executed.
    - Syntax: CASE (selector)

```
WHEN 'value 1' THEN
        statements;
WHEN 'value2' THEN
        statements;
ELSE
        Default Statements;
END;
```

- **Example:**

```
SQL>Set serveroutput on
SQL>DECLARE
a NUMBER :=55;      b NUMBER :=5;
arth_operationVARCHAR2(20) :='MULTIPLY';
BEGIN
  CASE (arth_operation)
  WHEN 'ADD' THEN          dbms_output.put_line('Addition: '|| a+b );
  WHEN 'SUBTRACT' THEN     dbms_output.put_line('Subtraction: '||a-b );
  WHEN 'MULTIPLY' THEN     dbms_output.put_line('Multiplication:'|| a*b);
  WHEN 'DIVIDE' THEN       dbms_output.put_line('Division:'|| a/b);
  ELSE                     dbms_output.put_line('Invalid operation');
  END;
  END;
```

☞ **SEARCHED CASE Statement :**

- The SEARCHED CASE statement is similar to the CASE statement. Instead of selector, SEARCHED CASE uses the **Boolean expression** defined in the WHEN clause.
- If the first WHEN clause satisfies the condition then it is executed, and the controller skip the remaining alternatives.
- **Syntax: CASE**

```
WHEN expression1 THEN
        statements;
WHENexpression2 THEN
                    ELSE
        Default statements;
END;
```

- **Example:**

```
SQL>Set serveroutput on
SQL>DECLARE
    a NUMBER :=55;          b NUMBER :=5;
    arth_operation VARCHAR2(20) :=' MULTIPLY';
BEGIN
    CASE  WHEN arth_operation = 'ADD' THEN
            dbms_output.put_line('Addition:'||a+b );
    WHEN arth_operation = 'SUBTRACT' THEN
            dbms_output.put_line('Subtraction:'|| a-b);
    WHEN arth_operation = 'MULTIPLY' THEN
            dbms_output.put_line('Multiplication:'|| a*b );
    WHEN arth_operation = 'DIVIDE' THEN
            dbms_output.put_line('Division:'|| a/b );
    ELSE
           dbms_output.put_Iine('Invalid operation');
     END;
END;
```

[Q]. Explain about Procedure (Or) Stored Procedures in Pl/SQL with examples?

PROCEDURE (or) STORED PROCEDURES in PL/SQL :

- A Procedure (or sub-program) is a small part of a program that performs a particular task, and it does not need to return any value.
- A procedure contains header, declaration section, executable section and optional exception handling section.

➢ **Creating a Procedure:**

- In PL/SQL, a procedure is created by using **CREATE PROCEDURE** statement with a list of parameters. When creating a procedure, define IN/OUT/INOUT parameters:

- **Syntax: CREATE OR REPLACE PROCEDURE** Procedure_Name Parameter_Name [**IN | OUT | IN OUT**] datatype(size)**IS | AS**

```
DECLARE
        declaration_section
BEGIN
        executable_section
[EXCEPTION]
        exception_section
END procedure_name;
/
```

Where,

- **CREATE** keyword is used to create a new procedure
- **OR REPLACE** option is used to modify an existing procedure.
- **Procedure_Name** is used to specify the name of the procedure created by CREATEstatement
- **Parameter_Name** specifies name of the PL/SQL variable
- **Datatype** specifies datatype of the argument or procedure.
- The **AS** keyword is used instead of the **IS** keyword for creating a standalone procedure.

➢ **Calling/Execute a Procedure:**

- A procedure is called by using the EXECUTE keyword or called by giving its name followed byparameters.
- **Syntax:** SQL>Execute Procedure-Name;
  Example: SQL> EXECUTE prog1(10,20);

Creating procedure:

```
set serveroutput on;
CREATE OR REPLACE PROCEDURE
Sum(a IN
number, b IN number)
IS
c number;
BEGIN
c := a+b;
dbms_output.put_line('Sum of two nos= '
|| c);
END Sum;
```

Executing a Procedure:

```
set serveroutput on;
DECLARE

x number;
y number;
BEGIN
x := &x;
y := &y;
Sum(x,y);
END;
/
Enter value for x: 25
old  5: x := &x;
new  5: x := 25;
Enter value for y: 60
old  6: y := &y;
new  6: y := 60;
Sum of two nos= 85
PL/SQL  procedure  successfully completed
```

➢ **Alter a Procedure:**

- The ALTER statement is used to recompile or re-design an existing procedure.
- **Syntax:** SQL>Alter Procedure [Procedure-Name];

➢ **Delete or Drop a Procedure:**

- The DROP keyword is used to delete an existing procedure.
- **Syntax:** SQL>Drop Procedure [Procedure-Name];
- **Example:** SQL> drop procedure prog1;

[Q]. Explain about Functions in PL/SQL with examples?

FUNCTIONS in PL/SQL :

- A function is a named PL/SQL block that takes one or more parameter and returns one value. Like procedure, a function contains header, declaration section, executable section and optional exception handling section.
- A function contains RETURN clause in the header section and at least one RETURN statement in the execution section.

➢ **Creating a Function:**

- In PL/SQL, a function is created by using **CREATE FUNCTION** statement with a list ofparameters. When creating a procedure, define IN/OUT/INOUT parameters.
- **Syntax:**

```
CREATE OR REPLACE FUNCTION function_name Parameter_Name
        [IN | OUT | IN OUT ] datatype(size)RETURN datatype
        IS | AS
DECLARE
        declaration_section
BEGIN
        executable_section
[EXCEPTION]
        exception_section
END function_name;
/
```

Where,

- **CREATE FUNCTION** keyword is used to create a new function
- **OR REPLACE** option is used to modify an existing function.
- **function_Name** is used to specify the name of the function.
- **Parameter_Name** specifies name of the variable whose value is passed to the function.
- **RETURN** specifies datatype of return value
- **Datatype** specifies datatype of the argument or procedure.

➢ **Execute a Function:**

- A function accepts one or more parameters but returns only one value.
- **Syntax:**

```
Function_Name (parameterlist);
```

- **Delete or Drop a Function:**
  - The DROP keyword is used to delete a function.
  - **Syntax:**

```
SQL>DROP FUNCTION Function_Name;
```

Example:

Creating a Function:	**Calling a Function:**

SQL> create or replace function adder(n1 in number,
n2 in number)
return number
is
n3 number(8);
begin
n3 :=n1+n2;
return n3;
end;
/
	SQL> DECLARE
	n3 number(2);
	BEGIN
	n3 := adder(11,22);
	dbms_output.put_line('Addition is: ' || n3);
	END;
	/
	Addition is: 33
	PL/SQL procedure successfully completed.

[Q]. Explain about Triggers with examples and also write it's benefits?

TRIGGERS:

- Triggers in oracle are blocks of PL/SQL code which are executed automatically based on someaction or event.
- These events can be:
  - DDL statements (CREATE, ALTER, DROP, TRUNCATE).
  - DML statements (INSERT, SELECT, UPDATE, DELETE).
  - Database operation like connecting or disconnecting to oracle (LOGON, LOGOFF, SHUTDOWN).
- Triggers are automatically and repeatedly called upon by oracle engine on satisfying certaincondition. Triggers can be activated or deactivated depending on the requirements.

Benefits Triggers:

- Triggers can be written for the following purposes –
  - Generating some derived column values automatically.
  - Enforcing referential integrity.
  - Event logging and storing information on table access.
  - Auditing.
  - Synchronous replication of tables.
  - Imposing security authorizations.
  - Preventing invalid transactions.

Creating Triggers:

- We can create trigger using the CREATE TRIGGER statement. If trigger activated, implicitly fire DMLstatement and if trigger deactivated can't fire.
- **Syntax:**

```
CREATE [OR REPLACE] TRIGGER trigger_name
[BEFORE | AFTER | INSTEAD OF]
[INSERT | UPDATE | DELETE [OF col_name]]
```

```
ON table_name
[REFERENCING OLD AS O, NEW AS N]
[FOR EACH STATEMENT | FOR EACH ROW]
WHEN conditionDECLARE
      Declaration statements
BEGIN
      PL/SQL Code
END trigger_name;
```

Where

- **CREATE [OR REPLACE] TRIGGER trigger_name:** Create a trigger with the given name. If already have overwrite the existing trigger with defined same name.
- **BEFORE | AFTER :** Indicates when the trigger get fire(i.e., specify the timing of the trigger's occurrences). BEFORE trigger execute before when statement execute before. AFTER trigger execute after the statement execute. INSTEAD OF is used when a view is created.
- **[INSERT, UPDATE, DELETE [COLUMN NAME..]:** Determines the performing trigger event. We can define more than one triggering event separated by OR keyword.
- **ON table_name:** Define the table name to performing trigger event.
- **Referencing [OLD AS OLD | NEW AS NEW]:** Give referencing to an old and new values of the data. **:old** means use existing row to perform event and **:new** means use executingnew row to perform event.
- **FOR EACH ROW | FOR EACH STATEMENT:** is the clause used to specify row level trigger and fire only once when the entire sql statement is execute.
- **WHEN Condition:** it is Optional. Use only for row level trigger. Trigger fire when specified condition is satisfy.

Example:

```
CREATE OR REPLACE TRIGGER
CheckAgeBEFORE
INSERT OR UPDATE ON student
FOR EACH ROW
BEGIN
      IF :new.Age>30 THEN
      raise_application_error(-20001, 'Age should not be greater than 30');END IF;
END;
```

Output: Trigger created.

- Following is the STUDENT table,

ROLLNO	SNAME	AGE	COURSE
11	Anu	20	BSC
12	Asha	21	BCOM
13	Arpit	18	BCA
14	Chetan	20	BCA
15	Nihal	19	BBA

- After initializing the trigger CheckAge, whenever we will insert any new values or update the existing values in the above table STUDENT our trigger will check the **age** before executing INSERT or UPDATE statements and according to the result of triggering restriction or condition it will execute the statement.

Let's take a few examples and try to understand this concept like....

Example-1:

INSERT into STUDENT values(16, 'Saina', 32, 'BCOM');

Output:

Age should not be greater than 30

Example-2:

INSERT into STUDENT values(17, 'Anna', 22, 'BCOM');

Output:

1 row created

Example-3:

UPDATE STUDENT set age=31 where ROLLNO=12;

Output:

Age should not be greater than 30

Example-4:

UPDATE STUDENT set age=23 where ROLLNO=12;

Output:

1 row updated.

EXAMPLE 2:

Existing data: SQL>Select * from employees;

EMP_ID	NAME	AGE	ADDR	SALARY
1	Shweta	23	Delhi	50000
2	Bharti	22	Karnal	52000
3	Deepika	24	UP	54000
4	Richi	25	US	56000
5	Bharat	21	Paris	58000
6	Sahdev	26	Delhi	60000

Trigger:

```
CREATE OR REPLACE TRIGGER show_salary_difference
BEFORE DELETE OR INSERT OR UPDATE ON employeesFOR
EACH ROW
WHEN (NEW.EMP_ID > 0)
DECLARE
        sal_diff    number;
  BEGIN
      sal_diff := :NEW.salary  - :OLD.salary;
      dbms_output.put_line('Old salary: ' || :OLD.salary);
      dbms_output.put_line('New salary: ' || :NEW.salary);
      dbms_output.put_line('Salary difference: ' || sal_diff);
  END;
  /
```

☞ **Note: The above trigger will execute for every INSERT, UPDATE or DELETE operations performed on the EMPLOYEES table.**

Drop a trigger:

SQL>DROP TRIGGER trigger_name;

[Q]. Explain the parts of Triggers and write down the types of Triggers?

PARTS OF A TRIGGER :

Whenever a trigger is created, it contains the following three sequential parts:

- **Triggering Event or Statement:** The statements due to which a trigger occurs is called triggering event or statement. Such statements can be DDL statements, DML statements or any database operation, executing which gives rise to a trigger.
- **Trigger Restriction:** The condition or any limitation applied on the trigger is called trigger restriction. Thus, if such a condition is **TRUE** then trigger occurs otherwise it does not occur.
- **Trigger Action:** When the triggering SQL statement is execute, trigger automatically call and PL/SQL trigger block execute.

TYPES OF TRIGGERS:

- A trigger's type is defined by the type of triggering transaction and by the level at which the trigger is executed. In the following sections, you will see descriptions of these classifications, along with relevant restrictions.

  - **Row-Level Triggers:**
    Row-level triggers execute once for each row in a transaction. Row-level triggers are the most common type of trigger; they are often used in data auditing applications.

  - **Statement-Level Triggers:**
    Statement-level triggers execute once for each transaction. For example, if a single transaction inserted 500 rows into a table, then a statement-level trigger on that table would only be executed once.

[Q]. Explain various types of Triggers with an example?

TYPES OF TRIGGERS IN PL/SQL :

- PL/SQL clearly indicated that Triggers can be classified into three categories:
  - **Level Triggers**
  - **Event Triggers**
  - **Timing Triggers**

- **Level Triggers**
  There are 2 different types of level triggers, they are…
  - **ROW LEVEL TRIGGERS(default for view)-**Executed once for the Entire DML Operation.
    - Executed once for the Entire DML statements like INSERT, UPDATE, DELETE etc.
    - It always uses a FOR EACH ROW clause in a triggering statement.
  - **STATEMENT LEVEL TRIGGERS (default for tables):**
    - Executed once for each row affected by the event

- **Event Triggers :**

  There are 3 different types of event triggers, they are…

  - **DDL EVENT TRIGGER :**
    - It fires with the execution of every DDL statement (CREATE, ALTER, DROP, TRUNCATE).
  - **DML EVENT TRIGGER :**
    - It fires with the execution of every DML statement (INSERT, UPDATE, DELETE).
  - **DATABASE EVENT TRIGGER**
    - It fires with the execution of every database operation which can be LOGON,LOGOFF, SHUTDOWN, SERVERERROR etc.

- **Timing Triggers:**

  There are 2 different types of event triggers, they are…

  - **BEFORE TRIGGER:**
    - It fires before executing DML statement.
    - Trigger statement may or may not executed depending upon the before condition block.
  - **AFTER TRIGGER:**
    - It fires after executing DML statement.

[Q]. write a short note on types of Triggers?

TYPE OF TRIGGERS :

- **BEFORE Trigger:** BEFORE trigger execute before the triggering DML statement (INSERT, UPDATE, DELETE) execute. Triggering SQL statement is may or may not execute, depending on the BEFORE trigger conditions block.
- **AFTER Trigger:** AFTER trigger execute after the triggering DML statement (INSERT, UPDATE, DELETE) executed. Triggering SQL statement is execute as soon as followed by the code of trigger before performing Database operation.
- **ROW Trigger**: ROW trigger fire for each and every record which are performing INSERT, UPDATE, DELETE from the database table. If row deleting is define as trigger event, when trigger file, deletes the five rows each times from the table.
- **Statement Trigger**: Statement trigger fire only once for each statement. If row deleting is define as trigger event, when trigger file, deletes the five rows at once from the table.
- **Combination Trigger**: Combination trigger are combination of two trigger types like…
  - **Before Statement Trigger:** Trigger fire only once for each statement before thetriggering DML statement.
  - **Before Row Trigger** : Trigger fire for each and every record before the triggeringDML statement.
  - **After Statement Trigger:** Trigger fire only once for each statement after the triggering DML statement executing.
  - **After Row Trigger:** Trigger fire for each and every record after the triggering DMLstatement executing.