UNIT – 1

Overview of Database Management System

1Q) What is file-based system? Explain Drawbacks of file-Based System?

Ans) The systems that are used to organize and maintain data files are known as file based data systems. These file systems are used to handle a single or multiple files and are not very efficient.

Functionalities

The functionalities of a File-based Data Management System are as follows:

1. A file based system helps in basic data management for any user.
2. The data stored in the file based system should remain consistent. Any transactions done in the file based system should not alter the consistency property.
3. The file based system should not allow any illegal or potentially hazardous operations to occur on the data.
4. The file based system should allow concurrent access by different processes and this should be carefully coordinated.
5. The file based system should make sure that the data is uniformly structured and stored so it is easier to access it.

Advantages of File Based System

1. The file Based system is not complicated and is simpler to use.
2. Because of the above point, this system is quite inexpensive.
3. Because the file based system is simple and cheap, it is normally suitable for home users and owners of small businesses.
4. Since the file based system is used by smaller organizations or individual users, it stores comparatively lesser amount of data. Hence, the data can be accessed faster and more easily.

Disadvantages of File Based System

1. The File based system is limited to a smaller size and cannot store large amounts of data.
2. This system is relatively uncomplicated but this means it cannot support complicated queries, data recovery etc.
3. There may be redundant data in the file based system as it does not have a complex mechanism to get rid of it.
4. The data is not very secure in a file based system and may be corrupted or destroyed.
5. The data files in the file based system may be stored across multiple locations. Consequently, it is difficult to share the data easily with multiple users.

2Q) Explain basic concepts of Database Management System. (Or Explain about Data and information, Database, Database management System,

Ans) The following terms are basic concepts of Database Management System,

- Data
- Information
- Metadata
- Database
- Database Management System

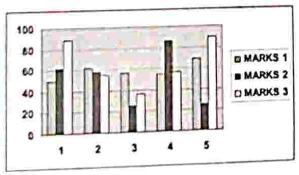
Data: Data consists of raw facts, text, graphics, image, audio and video segments that have meaning in the user's environment.

Ex: customer name, student address, photo image of an employee, etc.,

Information: Information is the result of processing raw data to reveal its meaning.

Ex:

SNO	SNAME	MARKS 1	MARKS 2	MARKS 3
1	XYZ	50	62	88
2	ABC	62	58	55
3	PQR	56	25	36
4	MNO	55	86	56
5	UVW	69	25	89



Some key points:

- Data constitute the building blocks of information.
- Information is processed by processing data.

- Information is used to reveal the meaning of data.
- Accurate, relevant, and timely information is the key to good decision making.
- Good decision making is the key to organizational survival in a global environment.

Metadata: Metadata are data that describe the properties or characteristics of other data. Some of these properties include data definitions, data structures and rules or constraints.

Ex:

Field Name	NULL	Type	Description
Empno	Not NULL	Number(4)	Employee Number
Ename		Varchar2(20)	Employee Name
ESal		Number(8,2)	Employee Salary

Database: A Database is a shared, integrated computer structure that stores a collection of End-user data and Metadata.

DBMS: A Database Management System is a collection of programs that manages the database structure and controls access to the data stored in the database.

Ex: MS-Access, MS-SQL Server, DB2, Oracle.

3Q) What are the Objectives of DBMS? (or) What are the Functions of DBMS?

Ans) A DBMS performs several important functions that guarantee the integrity and consistency of the data in the database. Most of those functions are transparent to end users, and most can be achieved only the use of a DBMS.

They include

- Data dictionary management,
- Data storage management,
- Data transformation and presentation,
- Security management,
- Multi-user access control,
- Backup and recovery management,
- Data integrity management,

- Database access languages and application programming interfaces, and
- Database communication interfaces.

Data dictionary management: The DBMS stores definitions of the data elements and their relationships (metadata) in a data dictionary. The DBMS uses the data dictionary to look up the required data component structures and relationships, thus relieving you from having to code such complex relationships in each program.

Data storage management: The DBMS creates and manages the complex structures required for data storage. A modern DBMS provides storage not only for the data, but also for related data entry forms or screen definitions, report definitions, data validation rules, procedural code structures to handle video and picture formats, and so on.

Data transformation and presentation: The DBMS transforms entered data to conform to reed data structures. The DBMS formats the physically reed data to make it conform to the user's logical expectations.

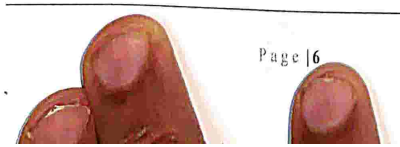
Security management: The DBMS creates a security system that enforces user security and data privacy. Security rules determine which users can access the database, which data items each user can access, and which data operations (read, add, delete, or modify) the user can perform.

Multi-user access control: To provide data integrity and data consistency, the DBMS uses sophisticated algorithms to ensure that multiple users can access the database concurrently without compromising the integrity of the database.

Backup and recovery management: The DBMS provides backup and data recovery to ensure data safety and integrity. Current DBMS systems provide special utilities that allow the DBA to perform routine and special backup and restore procedures.

Data integrity management: The DBMS promotes and enforces integrity rules, thus minimizing data redundancy and maximizing data consistency. The data relationships stored in the data dictionary are used to enforce data integrity.

Database access languages and application programming interfaces: The DBMS provides data access through a query



language. A query language is a nonprocedural language—one that lets the user specify what must be done without having to specify how it is to be done.

Database communication interfaces: Current-generation DBMSs accept end-user requests via multiple, different network environments. For example, the DBMS might provide access to the database via the Internet through the use of Web browsers such as Mozilla Firefox or Microsoft Internet Explorer.

4Q) Explain Evolution of Data Base Management System?

Ans) 1960s: The introduction of the term *database* coincided with the availability of direct access storage (disks and drums) from the mid-1960s onwards. IBM introduced their own DBMS in 1966, known as Information Management System (IMS).

1970s: IBM started working on a prototype system loosely based on Codd's concepts as System R in the early 1970s. **E.F. Codd** introduced the relational model in 1970. It provides a conceptually simple model for data as relations (typically considered "tables") with all data visible. In this era DB2 from IBM is the first DBMS product based on the relational model.

1980s: The 1980s ushered in the age of desktop computing. The new computers empowered their users with spreadsheets like Lotus 1-2-3 and database software like DBase. DBase was one of the top selling software titles in the 1980s and early 1990s.

1990s: In 1990, the DBMS took on a new object oriented approach joint with relational DBMS. In this approach, text, multimedia, internet and web use in conjunction with DBMS were available and possible. In 1997 XML applied to database processing, which solves long standing database problems.

2000s : In the 2000s, the NoSQL database started to emerge which features document orientated databases that use fast key-value stores.

5Q) What are the Classification of Database Management System (or) Explain Different type of Databases.

Ans) A DBMS can support many different types of databases. Databases can be classified according to,

- The number of Users,

- The database locations, and
- The data usage.

According the number of Users : The number of users determines whether the database is classified as Single-user or Multi-user.

A **single-user** database supports only one user at a time. A single-user database that runs on a personal computer is called a desktop database.

A **multi-user** database supports multiple users at the same time. When the multi-user database supports a relatively small number of users (usually fewer than 50) or a specific department within an organization, it is called a **workgroup** database. When the database is used by the entire organization and supports many users (more than 50, usually hundreds) across many departments, the database is known as an **enterprise** database.

According to the Data Locations : Location might also be used to classify the database. For example, a database that supports data located at a single site is called a **centralized** database. A database that supports data distributed across several different sites is called a **distributed** database.

According to the Data Usage : A database that is designed primarily to support a company's day-to-day operations is classified as an **operational database** (sometimes referred to as a transactional or production database). In contrast, a **data warehouse** focuses primarily on storing data used to generate information required to make tactical or strategic decisions. the data warehouse can store data derived from many sources. To make it easier to retrieve such data, the data warehouse structure is quite different from that of an operational or transactional database.

6Q)Explain about DBMS Approach?

Ans) The database approach emphasis the integration and sharing of data through out of the organization. This approach requires fundamental of reorientation process starting with top management.

Types of database approach:

1. Personal database: these are designed to support only one user. It includes laptops. Here only one user can accessed at a time. Here the main use is High security.

2. Work group database: A workgroup database is relatively small team of people. These people are belongs to same database. Here no of users are less than 25. It supports LAN.

3. Departmental database: A department is functional unit within organization. A departmental is generally larger than a work group and is responsible for more databases. Range is between 25&100.

4. Enterprise database: the term enterprise means collection of all department in that organization. So here range is entire organization. The best examples for it is Enterprise Resource Planning and Data warehouse.

5. Internet, Intranet and Extranet database:

Internet is a worldwide network that connects users of multiple forms.

Extranet uses the internet protocols establish limited access to company data and information by the company's customer & suppliers.

Intranet uses internet protocols to establish company data and information by the organization.

7Q) Explain Role and Advantages of DBMS.

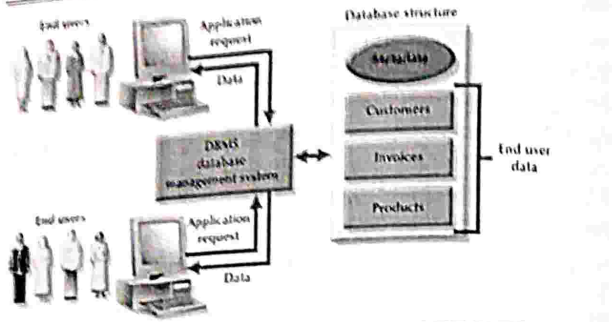
Ans) A Database Management System is a collection of programs that manages the database structure and controls access to the data stored in the database.

A DBMS serves as the intermediary between the user and the 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 DBMS. The DBMS receives all application requests and translates them into the complex operations required to fulfill those requests. The DBMS hides much of the database's internal complexity from the application programs and users.

Advantages of DBMS:

- Improved data sharing
- Improved data security
- Better data integration
- Minimized data inconsistency
- Improved data access
- Improved decision making
- Increased end-user productivity

• The DBMS manages the interaction between the end user and the database



Improved data sharing: The DBMS helps create an environment in which end users have better access to more data and better-managed data. Such access makes it possible for end users to respond quickly to changes in their environment.

Improved data security: A DBMS provides a framework for better enforcement of data privacy and security policies.

Better data integration: Wider access to well-managed data promotes an integrated view of the organization's operations and a clearer view of the big picture. It becomes much easier to see how actions in one segment of the company affect other segments.

Minimized data inconsistency: Data inconsistency exists when different versions of the same data appear in different places. The probability of data inconsistency is greatly reduced in a properly designed database.

Improved data access: The DBMS makes it possible to produce quick answers to ad hoc queries. From a database perspective, a query is a specific request issued to the DBMS for data manipulation—for example, to read or update the data.

Improved decision making: Better-managed data and improved data access make it possible to generate better quality information, on which better decisions are based.

Increased end-user productivity: The availability of data, combined with the tools that transform data into usable information, empowers

end users to make quick, informed decisions that can make the difference between success and failure in the global economy.

8Q) Explain ~~(Anis/spark)~~ ANSI/SPARC Data Model? (Or) Explain Degree of Abstraction?

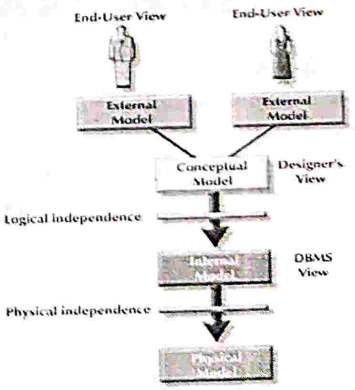
Ans) In the early 1970's the American National Standard Institute(ANSI) Standards Planning and Requirements Committee(ANSI/SPARC)defined a frame work for data modeling based on degree of data abstraction.

Data abstraction is the idea that a database design begins with a high level view and as it approaches implementation level, the level of detail increases. Using levels of abstraction can also be very helpful in integrating multiple views of data at different levels of an organization.

In 1970, the American National Standards Institute (ANSI) Standards Planning and Requirements Committee (SPARC) established a framework for database design based on the degrees of abstraction. The ANSI/SPARC architecture is composed of three levels of data abstraction: external, conceptual, and internal.

The External Model

The external environment is the end users' view of the data. The term end user refers to people who use the application programs to manipulate the data and generate information. The end users view of data usually applies to their specific business needs and those of their organizational unit.



The benefits of representing the design through the external model are:

- It is easier to identify the data needed by the end users.
- It makes the designer's job easy

- It helps to ensure security constraints in the database design.
- It makes application program development much simpler.

The Conceptual Model

The conceptual model is created by taking all views and forming a global view of the entire database. The conceptual model is generally represented by ER Diagrams. Another name for the conceptual model is the logical design of the database.

The benefits of representing the design through the conceptual model are:

- It provides a relatively easily understood macro level view of the data environment.
- It is independent of both software and hardware.

The Internal Model

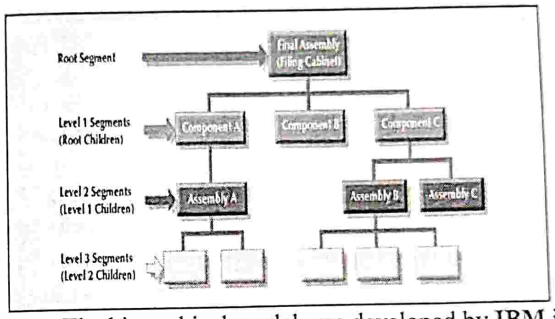
Once a specific DBMS has been selected, the internal model maps the conceptual model to the DBMS. The internal model is the database as "seen" by the specific DBMS. The goal in designing the internal model is to achieve logical independence, where the internal model can be changed without affecting conceptual model.

9Q) Explain about data models? Or Explain Different Types of Data Models?

Ans: A data model is a collection of concepts for describing data, its relationships, and its constraints Provides a clearer and more accurate description and representation of data Standard platform that enables database designers and end-users. The following are the different types data models…

- **Hierarchical Model**
- **Network Model**
- **Relational Data Model**
- **Entity Relational Model**
- **Object Oriented Data Model**

Hierarchical Model



- The hierarchical model was developed by **IBM** in 1968.
- The data is organized in a **tree structure** where the nodes represent the records and the branches of the tree represent the fields.
- Since the data is organized in a tree structure, the parent node has the **links** to its child nodes.
- If we want to search a record, we have to traverse the tree from the root through all its parent nodes to reach the specific record. Thus, **searching for a record is very time consuming.**
- The **hashing function** is used to locate the root.
- **SYSTEM2000** is an example of hierarchical database.

Advantages:

1. It promotes data sharing
2. Parent/Child relationship promotes conceptual simplicity
3. Data security is provided and enforced by DBMS
4. Parent/Child relationship promotes data integrity.
5. it is efficient with 1:M relationships.

Disadvantages:

1. *Complex implementation:* the implementation of a database design can become very complicated.

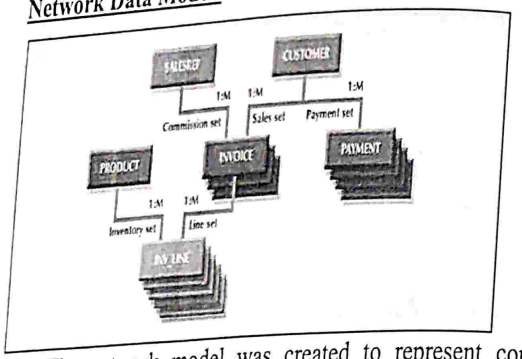
2. Difficult to manage. Any changes in the database structure require changes in all application programs that access the database.

3. Lacks structural independence. Structural independence exists when changes in the database structure.

4. Implementation limitations. A common many-to-many (M:N) relationship is difficult to implement in a hierarchical model.

5. There is no data definition or data manipulation language in the DBMS.

Network Data Model:



- The network model was created to represent complex data relationships more effectively than the hierarchical model, to improve database performance.
- In the network data model, the database consists of a collection of set-type occurrences.
- Each set-type occurrence has one occurrence of OWNER RECORD, with zero or more occurrences of MEMBER RECORDS.
- The member sets belonging to different owners are disjoint.
- In 1971, the Conference on Data Systems Languages (CODASYL) formally defined the network model.

Advantages:

1. Provide very efficient "High-speed" retrieval.
2. The network model is conceptually simple and easy to design.
3. Ability to handle more relationship types
4. The network model can Ease of data handle the one-to-many and many-to- many relationships.

Disadvantages

1. Complex implementation requires knowledge of physical data storage characteristics.
2. Navigational system yields complex implementation, application development and management.
3. Structural changes require changes in all application programs.

Relational Data Model:

- The **relational model** was first developed by E. F. Codd (of IBM) in 1970.
- The relational model produced an "automatic transmission" database to replace the "standard transmission" databases that preceded it.
- The relational data model is implemented through a very sophisticated **relational database management system** (RDBMS).
- The RDBMS manages all of the physical details, while the user sees the relational database as a collections of tables in which data are stored.
- A relational table stores a collection of related entities.

Database name: Ch02_InsureCo Table name: AGENT (first six attributes)

AGENT_CODE	AGENT_LNAME	AGENT_FNAME	AGENT_INITIAL	AGENT_AREACODE	AGENT_PHONE
501	Alby	Alex	B	713	228-1249
502	Hahn	Leah	F	615	882-1244
503	Okon	John	T	615	123-5589

Link through AGENT_CODE

Table name: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_RENEW_DATE	AGENT_CODE
10010	Ramas	Alfred	A	615	844-2573	05-Apr-2004	502
10011	Dunne	Leona	K	713	894-1238	16-Jun-2004	501
10012	Smith	Kathy	W	615	894-2285	29-Jan-2005	502
10013	Olowski	Paul	F	615	894-2180	14-Oct-2004	502
10014	Orlando	Myron		615	222-1672	28-Dec-2004	501

Advantages:

1. Structural independence is promoted by the use of independent tables
2. Tabular view substantially improve conceptual simplicity.
3. Easier database design, implementation, management, and use.
4. Ad hoc query capability is based on SQL.
5. Powerful RDBMS isolates the end user from physical-level details and improves implementation and management simplicity.

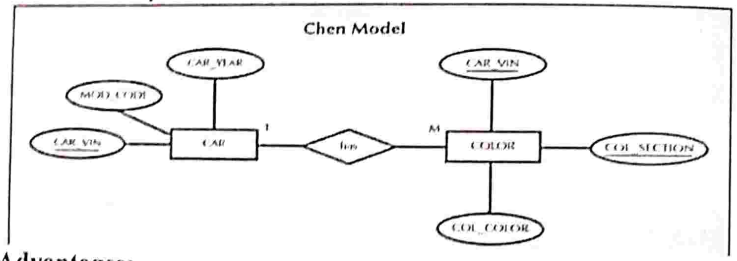
Disadvantages:

1. The RDBMS requires Substantial hardware and system software overhead.

2. Conceptual simplicity gives relatively untrained people the tools to use a good system poorly, and if unchecked, it may produce the same data anomalies found in file system.
3. It may promote islands of information problems as individuals and departments can easily develop their own applications.

Entity Relationship Data Model:

- The **Entity Relationship** (ER) **model or ERM** is a widely accepted and adapted graphical tool for data modeling.
- PeterChen first introduced the ER data model in 1976.
- ER models are normally represented in an entity relationship diagram(ERD), which uses graphical representation to model database components.
- The ER model is based on Entities, Attributes and Relationships.
- An Entity is anything about which data are to be collected and stored.
- An Attribute is a particular characteristic of an Entity.
- Relationship is an association between tow entities.



Advantages:

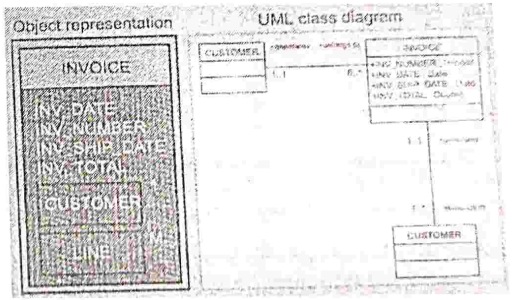
1. Visual modeling yields exceptional conceptual simplicity
2. Visual representation makes it an effective communication tool.
3. It is integrated with the relational data model. Such integration helps make relational database design a very structured process.

Disadvantages:

1. There is limited constraint representation.
2. There is limited relationship representation.
3. There is no data manipulation language.
4. Loss of information content occurs when attributes are removed from entities to avoid crowded displays.

The Object Oriented Model:

- In the Object oriented data model (OODM), both data and their relationships are contained in a single structure known as an Object.
- The OODM is the basis for the Object-Oriented DataBase Management System(OODBMS).
- The object-oriented data model is based on Objects, Attributes, and Classes.
- An object is an abstraction of a real-world entity. An object may be considered equivalent to an ER model's entity.
- Attributes describe the properties of an object.
- A **class** is a collection of similar objects with shared structure (attributes) and behavior (methods).
- Object-oriented data models are typically depicted using Unified Modeling Language(UML) class diagrams.



Advantages:

1. Semantic content is added
2. Visual presentation includes semantic content.
3. Inheritance promotes Data integrity.
4. The OODM's object autonomy ensures both structural and data independence.

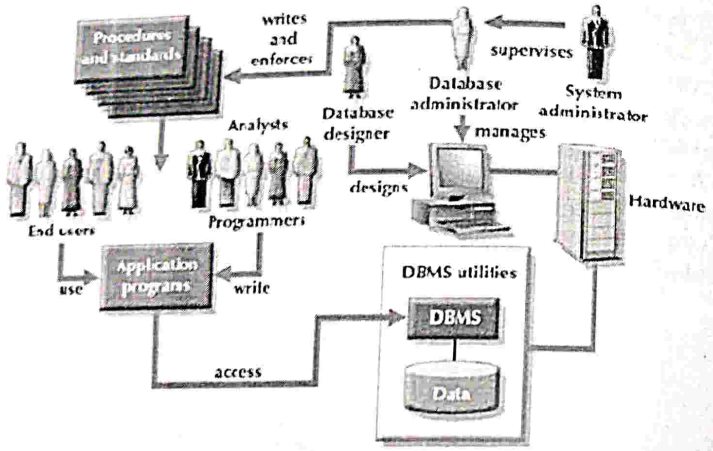
Disadvantages:

1. Object oriented standards have evolved very slowly compared to other data model standards.
2. It is a Complex navigational system.
3. There is a steep learning curve.
4. High system overhead slows transactions.

10Q) Explain Components and Interfaces of Database Management System?

Ans) The five components of the database system environment shown in the following Figure.

THE DATABASE SYSTEM ENVIRONMENT



Hardware: Hardware refers to all of the system's physical devices; for example, computers (microcomputers, workstations, servers, and supercomputers), storage devices, printers, network devices (hubs, switches, routers, fiber optics), and other devices.

Software: Although the most readily identified software is the DBMS itself, to make the database system function fully, three types of software are needed: operating system software, DBMS software, and application programs and utilities.

- Operating system software manages all hardware components and makes it possible for other software to run on the computers. Examples of operating system software include Microsoft Windows, Linux, Mac OS, UNIX, and MVS.

- DBMS software manages the database within the database system. Some examples of DBMS software include Microsoft SQL Server, Oracle Corporation's Oracle, MySQL AB's MySQL and IBM's DB2.
- Application programs and utility software are used to access and manipulate data in the DBMS and to manage the computer environment in which data access and manipulation take place.

People: This component includes all users of the database system. On the basis of primary job functions, five types of users can be identified in a database system: systems administrators, database administrators, database designers, systems analysts and programmers, and end users.

- System administrators oversee the database system's general operations.
- Database administrators, also known as DBAs, manage the DBMS and ensure that the database is functioning properly.
- Database designers design the database structure. They are, in effect, the database architects.
- Systems analysts and programmers design and implement the application programs. They design and create the data entry screens, reports, and procedures through which end users access and manipulate the database's data.
- End users are the people who use the application programs to run the organization's daily operations.

Procedures: Procedures are the instructions and rules that govern the design and use of the data-base system. Procedures are a critical, although occasionally forgotten, component of the system. Procedures also are used to insure that there is an organized way to monitor and audit both the data that enter the database mi the information that is generated through the use of that data.

Data: The word data covers the collection of facts stored in the database. Because data are the raw material from which information is generated

11Q) Explain about Database Architecture?

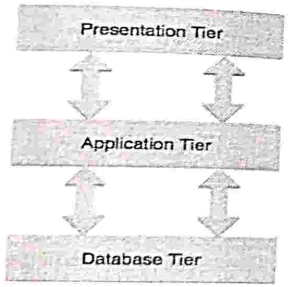
Ans) The design of a DBMS depends on its architecture. It can be centralized or decentralized or hierarchical. The architecture of a DBMS can be seen as either single tier or multi-tier. An n-tier architecture divides the whole system into related but independent n modules, which can be independently modified, altered, changed, or replaced.

In 1-tier architecture, the DBMS is the only entity where the user directly sits on the DBMS and uses it. Any changes done here will directly be done on the DBMS itself. It does not provide handy tools for end-users. Database designers and programmers normally prefer to use single-tier architecture.

If the architecture of DBMS is 2-tier, then it must have an application through which the DBMS can be accessed. Programmers use 2-tier architecture where they access the DBMS by means of an application. Here the application tier is entirely independent of the database in terms of operation, design, and programming.

3-tier Architecture

A 3-tier architecture separates its tiers from each other based on the complexity of the users and how they use the data present in the database. It is the most widely used architecture to design a DBMS.



- **Database (Data) Tier** – At this tier, the database resides along with its query processing languages. We also have the relations that define the data and their constraints at this level.
- **Application (Middle) Tier** – At this tier reside the application server and the programs that access the database. For a user, this application tier presents an abstracted view of the database. End-users are unaware of any existence of the database beyond the application. At the other end, the database tier is not aware of any other user beyond the application tier. Hence, the application layer sits in the middle and acts as a mediator between the end-user and the database.

- **User (Presentation) Tier** – End-users operate on this tier and they know nothing about any existence of the database beyond this layer. At this layer, multiple views of the database can be provided by the application. All views are generated by applications that reside in the application tier.

12Q) Explain about Situations where DBMS is not Necessary?(or) Explain Disadvantages of DBMS.

Ans) The following are some disadvantages of DBMS.

- Increased costs
- Management complexity
- Maintaining currency
- Vendor dependence
- Frequent upgrade/replacement cycles

Increased costs: Database systems require sophisticated hardware and software and highly skilled personnel. The cost of maintaining the hardware, software, and personnel required to operate and manage a database system can be substantial.

Management complexity: Database systems interface with many different technologies and have a significant impact on a company's resources and culture.

Maintaining currency: To maximize the efficiency of the database system, we must keep our system current. Therefore, we must perform frequent updates and apply the latest patches and security measures to all components.

Vendor dependence: Given the heavy investment in technology and personnel training, companies might be reluctant to change database vendors.

Frequent upgrade/replacement cycles: DBMS vendors frequently upgrade their products by adding new functionality. Such new features often come bundled in new upgrade versions of the software. Some of these versions require hardware upgrades.

13Q) Explain about DBMS Vendors and Their Products.

Ans) Most of the databases are considered design, implementation and management issues based on product database. The focus on

production databases are the database most frequently encountered in common activities such as enrolling in a class, registering a car, buying a product, or making a bank deposit or withdrawal.

Second, data warehouse database derive most of their data from production databases, and if production database are poorly designed, the data warehouse database based of them will lose their reliability and values as well.

Consider the following table for both products and vendors

PRODUCT	NUMBER OF USERS			DATA LOCATION		DATA USAGE	
		MULTIUSER					
	SINGLE USER	WORK GROUP	ENTER PRISE	CENTR ALIZED	DISTRI BUTED	OPERA TIONAL	DATA WARE HOUSE
MS ACCESS	X	X		X		X	
MYSQL server	X*	X	X	X	X	X	X
IBM DB2	X*	X	X	X	X	X	X
oracle DBMS	X*	X	X	X	X	X	X

* Vendors offers single user/personal DBMS version.

UNIT - 2

Entity-Relationship Model

1Q) Explain the building blocks of an entity relationship diagram?

Ans)

Ans: A **data model** is the relatively simple representation, usually graphical, of complex real-world data structures. The model's main function is to help us understand the complexities of the real-world environment. Within the database environment, a data model represents data structures and their characteristics, relations, constraints, and transformations.

Basic Building Blocks of Data Model:

The basic building blocks of the Entity Relationship Diagram data model are **entities, attributes, and relationships.**

An **entity** is anything, such as a person, place, thing, or event, about which data are to be collected and stored. Entities may be physical objects such as customers or products

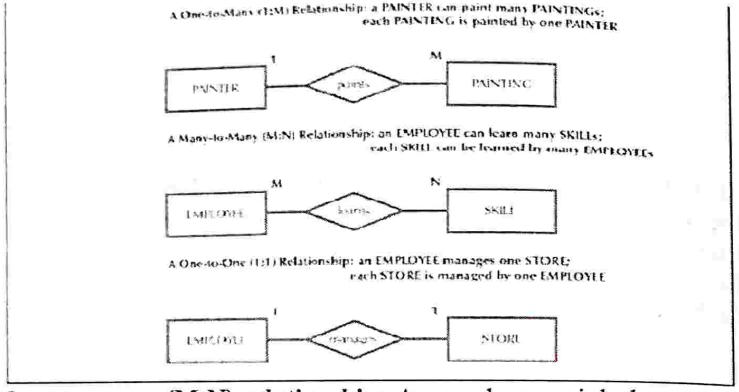
An **attribute** is a characteristic of an entity.

Example: A **CUSTOMER** entity would be described by attributes such as customer_no, customer_name, customer_phone, and customer_address. The attributes are the equivalent of fields in file systems.

A **relationship** describes an association among (two or more) entities. For example, a relationship between customers and agents might be described as "an agent can serve many customers and each customer might be served by one agent."

Data models use three types of relationships:

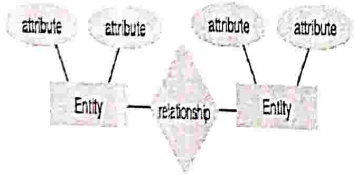
One-to-many (1:M) relationship: A painter paints many different paintings, but each one of them is painted by only one painter. Thus, the painter (the "one") is related to the paintings (the "many"). Therefore, Database designers label the relationship "PAINTER paints PAINTING" as a 1:M Relationship.



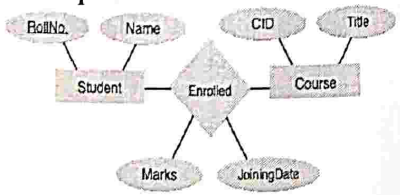
Many-to- many (M:N) relationship: An employee might learn many job skills, and each job skill might be learned by many employees. Database designers label the relationship "EMPLOYEE learns SKILL" as M:N Relationship.

One-to-one (1:1) relationship: A retail company's management structure may require that each one of its stores be managed by a single employee. In turn, each store manager – who is an employee – only manages a single store. Therefore, the relationship "EMPLOYEE manages STORE" is labeled 1:1 Relationship.

Syntax:



Example:



2Q) Explain about classification of entity sets?(or) What is an Entity? Explain types of Entities.

Ans) Entity:

An Entity is a person, place, object, event or concept in the user environment about which the organization wishes to maintain data.

Ex:- **Person :** Employee, Student, Patient
Place : Store, Warehouse, State
Object : Machine, Building, Automobile
Event : Sale, Registration, Renewal
Concept: Account, Course, Workcenter

Entity Type:

An Entity type is a collection of entities that share common properties or characteristics.

Entity instance :

An Entity instance is a single occurrence of an entity type.

Ex:- **Entity type :** Employee
Attributes: ENO, ENAME , ADDRESS
Two Entity instances are
1 Arun ATP
2 Babu ATP

Naming and defining Entity types:

The following are some guidelines for naming Entity types.

1. An entity type name is a singular noun, such as CUSTOMER, STUDENT, AUTOMOBILE.
2. The name should be descriptive for the organization and distinct from all other entity type names within that organization.
3. The abbreviation or short name should be specified for each entity type name.
4. Event entity types should be named for the result of the event, not the activities or process of the event.
5. The name used for the same entity type should be the same on all E-R diagrams.

Entity Types are classified into two types. They are 1)Strong Entity type 2) Weak Entity type.

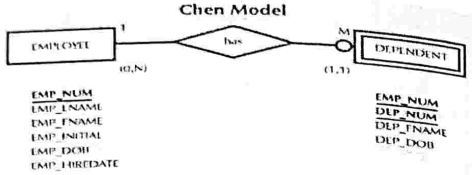
Strong Entity type :

A strong entity type is an entity type that exists independently of other entity types. Instances of a strong entity type always have a unique characteristic called an Identifier.

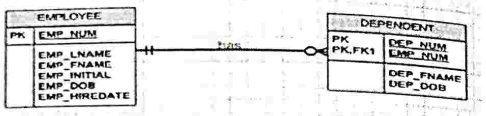
Weak entity type :

A Weak entity type is an entity type whose existence depends on some other entity type. A weak entity type has no business meaning in the E-R diagram without the entity type on which it depends.

Chen Model



Crow's Foot Model



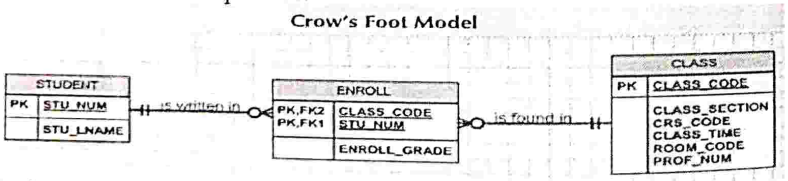
The entity type on which the weak entity type depends is called the Identifying owner. A weak entity type does not have its own Identifier. The relationship between a weak entity type and its owner is called an Identifying relationship.

In the above figure EMPLOEE is strong entity type and DEPENDENT is weak entity type.

Associative (Composite) Entities:

The associative entity is composed of the primary keys of each of the entities to be connected. The associative entities are used to implement a M:M relationship between two or more entities. The associative entity is also called as composite or bridge entity. The associative entity may also contain additional attributes that play no role in the connective process.

Crow's Foot Model



In the above figure ENROLL is Composite entity, it is based on the primary keys of the entities STUDENT and CLASS.

3Q)Explain about attribute classification?(or) What is an Attribute? Explain types of Attributes.

Ans). An Attribute is a property or characteristic of an entity type that is of interest to the organization.

Ex:- Entity type : STUDENT

Attributes: Student_ID, Student_NAME, Student_ADDRESS

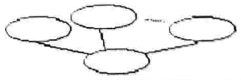
Naming and defining Attributes:

The following are some guidelines for naming Attributes.

1. An attribute name is a noun, such as Customer_ID, Age, Product_Price.
2. An attribute name should be unique.
3. To make an attribute unique and for clarity purposes, each attribute name should follow a standard format.
4. Similar attributes of different entity types should use the same qualifiers and classes.

The following are the different types of Attributes.

1. Composite Attribute
2. Simple Attribute
3. Single valued Attribute
4. Multi valued Attribute
5. Derived Attribute

E R Notation	Meaning
	Simple Attribute
	Key attribute
	Derived attribute
	Multi-valued attribute
	Composite attribute

Composite Attribute :

A composite attribute is an attribute that can be broken down into component parts. The composite attributes may appear above or below the composite attribute on an E-R diagram. Composite attributes provide considerable flexibility to users.

Simple Attribute :

A simple attribute is an attribute that cannot be broken down into smaller components. It is also called Atomic attribute.

Single Valued Attribute :

A single valued attribute is an attribute that may take only one value for a given instance.

Multi Valued Attribute :

A Multi valued attribute is an attribute that may take on more than one value for a given entity instance. It is indicate with an ellipse with double lines on an E-R diagram.

Derived Attribute :

A derived attribute is an attribute whose values can be calculated from related attribute values. It is indicate with dashed line on an E-R diagram.

4Q)Explain about relationship degree?

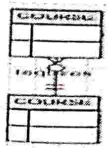
Ans) A Relationship Degree indicates the number of entities or participants associated with a relationship.

The following are different types of relationship degree.

- Unary relationship
- Binary relationship
- Ternary relationship

Unary relationship

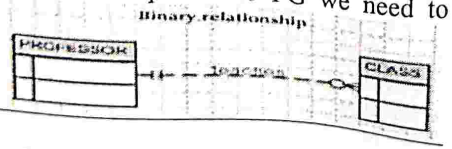
A Unary relationship exists when an association is maintained within a single entity. Unary relationship is also called as recursive relationship.



The above figure shows 'COURSE requires COURSE' relationship. Here to join in a course we require completion of another course. For example to do PG we need to complete UG.

Binary relationship

A Binary relationship

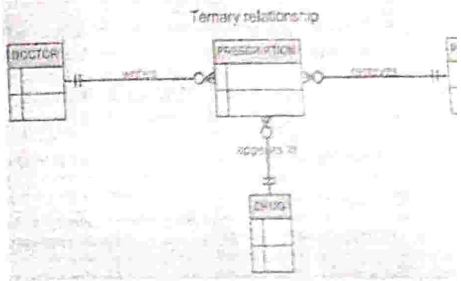


exists when two entities are associated in a relationship. Binary relationships are most common relationships.

In the above figure the relationship 'a PROFESSOR teaches one or more CLASSes' represents a binary relationship.

Ternary relationship

A Ternary relationship exists when three entities are associated in a relationship.



The above figure shows the following relationships.

1. A DOCTOR writes one or more PRESCRIPTIONs
2. A PATIENT may receive one or more PRESCRIPTIONs
3. A Drug may appear in one or more PRESCRIPTIONs.

5Q) Explain about relationship classification?

Types of Relationship Mapping

Following are the types of Relationship Mapping,

1. One - to - One Relationship
2. One - to - Many Relationship
3. Many - to - One Relationship
4. Many - to - Many Relationship

1. One - to - One Relationship

- In One - to - One Relationship, one entity is related with only one other entity.
- One row in a table is linked with only one row in another table and vice versa.

For example: A Country can have only one Capital City.

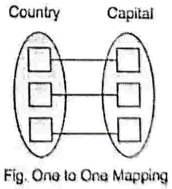


Fig. One to One Mapping

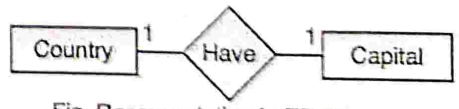


Fig. Representation in ER Diagram

2. One - to - Many Relationship

- In One - to - Many Relationship, one entity is related to many other entities.
- One row in a table A is linked to many rows in a table B, but one row in a table B is linked to only one row in table A.

For example: One Department has many Employees.

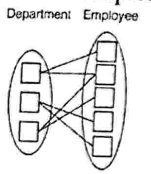


Fig. One to Many Mapping

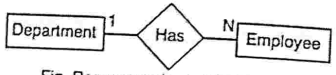


Fig. Representation in ER Diagram

3. Many - to - One Relationship

- In Many - to - One Relationship, many entities can be related with only one other entity.

For example: No. of Employee works for Department.

- Multiple rows in Employee table is related with only one row in Department table.

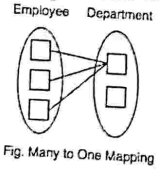


Fig. Many to One Mapping

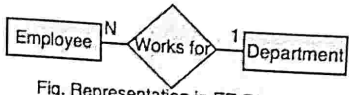


Fig. Representation in ER Diagram

4. Many - to - Many Relationship

- In Many - to - Many Relationship, many entities are related with the multiple other entities.
- This relationship is a type of cardinality which refers the relation between two entities.

For example: Various Books in a Library are issued by many Students.

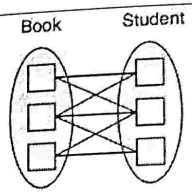


Fig. Many to Many Mapping

Fig. Representation in ER Diagram

6Q) Explain reducing ER Model to table

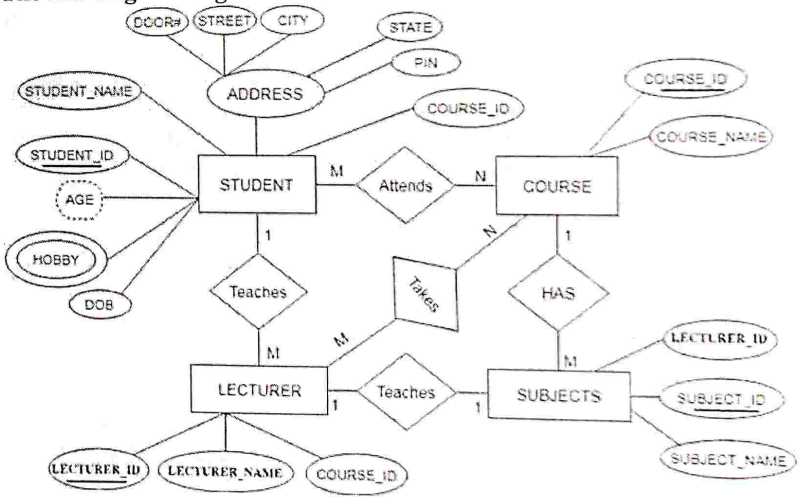
Ans)

Reduction of ER diagram into Table

Database can be represented using the notations and these notations can be reduced to collection of tables.

In the database, every entity set or relationship set can be represented in tabular form.

The ER diagram is given below:



There are some points for converting the ER diagram into table:

- **Entity type becomes a table.**

In the given ER diagram, LECTURE, STUDENT, SUBJECT and COURSE forms individual tables.

- **All single valued attribute becomes a column for the table.**

In the STUDENT entity, STUDENT_NAME and STUDENT_ID form the column of STUDENT table. Similarly COURSE_NAME and COURSE_ID form the column of COURSE table and so on.

- **Key attribute of the entity type represented by the primary key.**

In the given ER diagram, COURSE_ID, STUDENT_ID, SUBJECT_ID and LECTURE_ID are the key attribute of the entity.

- **Multi-valued attribute are represented by separate table.**

In the student table, a hobby is a multi-valued attribute. So it is not possible to represent multiple values in a single column of STUDENT table. Hence we create a table STUD_HOBBY with column name STUDENT_ID and HOBBY. Using both the column, we create a composite key.

- **Composite attribute represented by components.**

In the given ER diagram, student address is a composite attribute. It contains CITY, PIN, DOOR#, STREET and STATE. In the STUDENT table, these attributes can merge as individual column.

- **Derived attributes are not considered in the table.**

In the STUDENT table, Age is the derived attribute. It can be calculated at any point of time by calculating the difference between current date and Date of Birth.

Using these rules, you can covert ER diagram into tables and columns and assign the mapping between the tables. Table structure for the given ER diagram is as below:

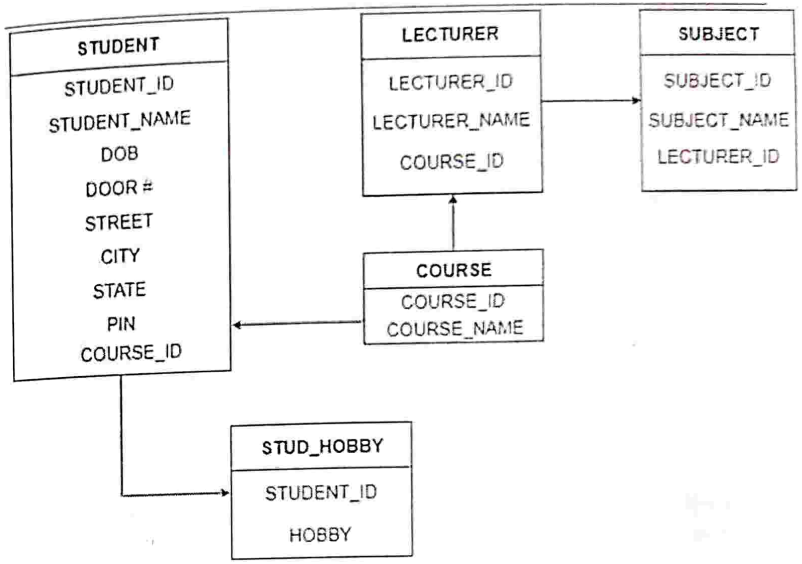


Figure: Table structure

7Q)Explain about enhanced entity-relationship model (EER model)?

Ans) EER Model

EER is a high-level data model that incorporates the extensions to the original ER model.

It is a diagrammatic technique for displaying the following concepts

- Sub Class and Super Class
- Specialization and Generalization
- Union or Category
- Aggregation

These concepts are used when the comes in EER schema and the resulting schema diagrams called as EER Diagrams.

Features of EER Model

- EER creates a design more accurate to database schemas.

- It reflects the data properties and constraints more precisely.
- It includes all modeling concepts of the ER model.
- Diagrammatic technique helps for displaying the EER schema.
- It includes the concept of specialization and generalization.
- It is used to represent a collection of objects that is union of objects of different of different entity types.

A. Sub Class and Super Class

- Sub class and Super class relationship leads the concept of Inheritance.
- The relationship between sub class and super class is denoted with (d) symbol.

1. Super Class

- Super class is an entity type that has a relationship with one or more subtypes.
- An entity cannot exist in database merely by being member of any super class.
  For example: Shape super class is having sub groups as Square, Circle, Triangle.

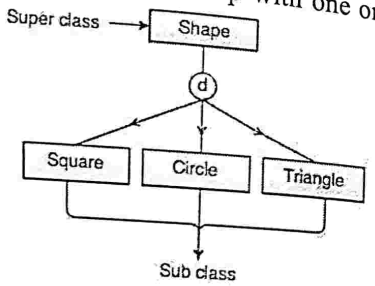


Fig. Super class/Sub class Relationship

2. Sub Class

- Sub class is a group of entities with unique attributes.
- Sub class inherits properties and attributes from its super class.
  For example: Square, Circle, Triangle are the sub class of Shape super class.

B. Specialization and Generalization

1. Generalization

- Generalization is the process of generalizing the entities which contain the properties of all the generalized entities.
- It is a bottom approach, in which two lower level entities combine to form a higher level entity.
- Generalization is the reverse process of Specialization.

- It defines a general entity type from a set of specialized entity type.
- It minimizes the difference between the entities by identifying the common features.
  For example:

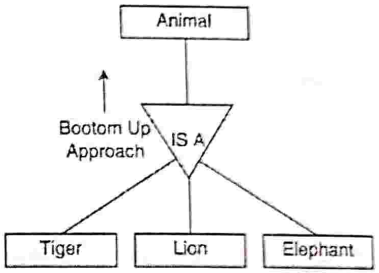


Fig. Generalization

In the above example, Tiger, Lion, Elephant can all be generalized as Animals.

2. Specialization

- Specialization is a process that defines a group entities which is divided into sub groups based on their characteristic.
- It is a top down approach, in which one higher entity can be broken down into two lower level entity.
- It maximizes the difference between the members of an entity by identifying the unique characteristic or attributes of each member.
- It defines one or more sub class for the super class and also forms the super class/subclass relationship.
  For example

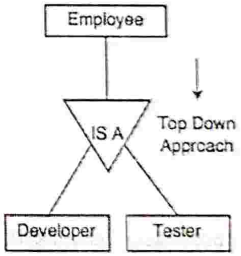


Fig. Specialization

In the above example, Employee can be specialized as Developer or Tester, based on what role they play in an Organization.

C. Category or Union

- Category represents a single super class or sub class relationship with more than one super class.
- It can be a total or partial participation. **For example** Car booking, Car owner can be a person, a bank (holds a possession on a Car) or a company.

Category (sub class) → Owner is a subset of the union of the three super classes → Company, Bank, and Person. A Category member must exist in at least one of its super classes.

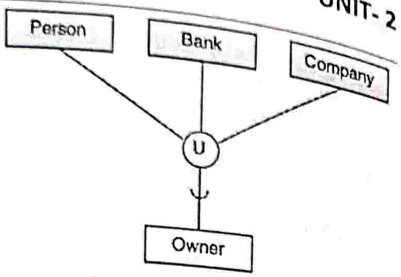


Fig. Categories (Union Type)

D. Aggregation

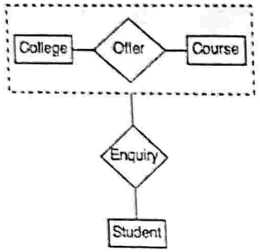


Fig. Aggregation

- Aggregation is a process that represents a relationship between a whole object and its component parts.
- It abstracts a relationship between objects and viewing the relationship as an object.
- It is a process when two entities are treated as a single entity.

In the above example, the relation between College and Course is acting as an Entity in Relation with Student.

8Q) Explain about IS A relationship and attribute inheritance?
Ans)

Generalization

Generalization is an abstracting process of viewing sets of objects as a single general class by concentrating on the general characteristics of the constituent sets while suppressing or ignoring their differences.

Notation of Generalization :

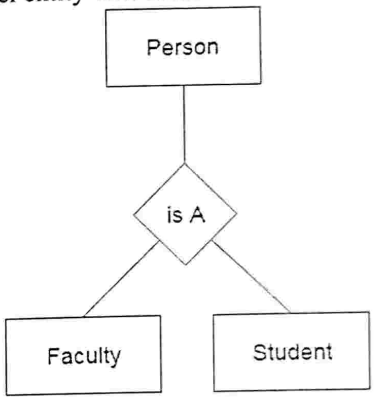
Generalization is represented by a triangle with a line.

Generalization is an IS_A relationship

- Generalization is like a bottom-up approach in which two or more entities of lower level combine to form a higher level entity if they have some attributes in common.
- In generalization, entity of higher level can also combine with the entities of lower level to form further higher level entity.

- Generalization is more like subclass and super-class system but the only difference is the approach. Generalization uses the bottom-up approach.
- In generalization, entities are combined to form a more generalized entity i.e. subclasses are combined to make a super-class.

For example: Faculty and Student entities can be generalized and create a higher level entity with name Person.



Attribute inheritance: Generally inheritance means derive a new class form the old class. If we can create new or using attribute in new entity with the help of old entity. This process is nothing but attribute inheritance.

Example: Consider the simple example for attribute inheritance

Person:

Aadhar_no	DOB	age	gender

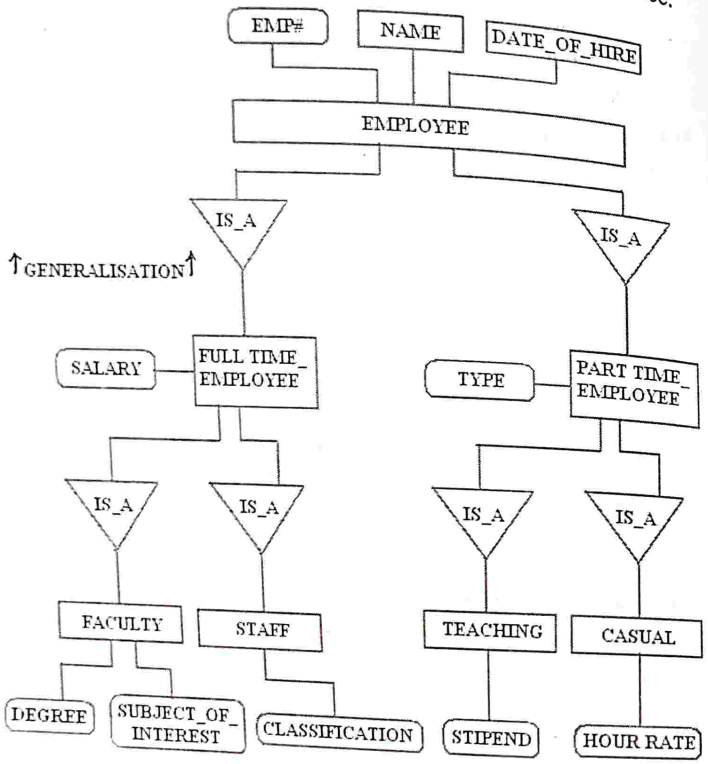
Student:

Aadhar_no	Rno	Marks	result

Employee:

Aadhar_no	Eid	basic	gross

In the above examples person, student and Employee are entities. And observe the aadhar_no is an attribute appeared in all entities. This attribute derived attribute. This process is called attribute inheritance.

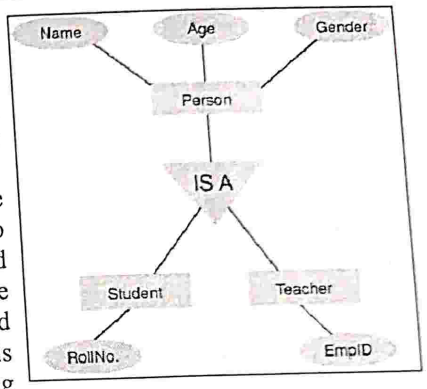


9Q) Explain about multiple inheritance?

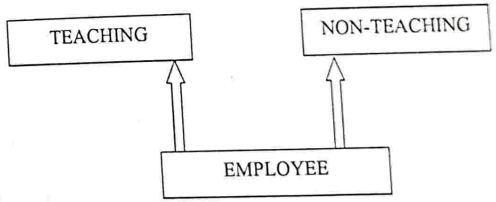
Ans)Multiple Inheritance

We use all the above features of ER-Model in order to create classes of objects in object-oriented programming. The details of entities are generally hidden from the user; this process known as **abstraction.** Inheritance is an important feature of Generalization and Specialization. It allows lower-level entities to inherit the attributes of higher-level entities.

For example, the attributes of a Person class such as name, age, and gender can be inherited by lower-level entities such as Student or Teacher.



Example: Consider the following example here two tables called Teaching and Non-teaching table are available. Another table called employee which is combination of both teaching and non-teaching table's properties and its attributes. Here we can also implement the constraints from super classes to sub classes. This is the main diagram for multiple inheritance.



10Q) Explain about constraints on specialization and generalization?

Ans) Consider the following tables for implementing the rules or constraints for specialization and generalization.

SUPER CLASS: EMPLOYEE	Name	Dob	Address	Job_Type

SUBCLASSES:

1.SECRETARY	Typing Speed
2.TECHNICIAN	Trade
3.ENGINEER	Eng_Type

Rules:

1. Specialization and generalization must and should be apply the relationship between two classes like one is parent class and another one is sub class.
2. Sub class can access the data form the super class with maintained of attribute condition.
3. We can also derive the attribute form super class to sub class
4. Write the query on subclass like secretary, technician and engineer with the help of employee entity.
5. If an attribute in super class which will affect on all subclasses then it is called attribute defined specialization. It is also called defined. Here Job_type is defined attribute

   Ex: select * from engineer where job_ype="engineer";
6. To implement any integrity constrains on both classes with the help of inheritance.
7. If we collect the data from the super class based on condition then such attribute is called predefine attribute subclasses or condition defined subclasses.

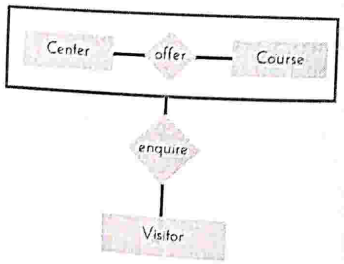
   Ex: select * from engineer where address="atp";
8. We can also use the own attribute called user defined attribute.
9. If we want maintain the specialization in different entities (sub classes). Here may be derived attribute is different. This rule is called disjoint sets.
10. Specialization may be implementing on the general entities like partial specialization or complete specialization.

11Q) Explain about aggregation?

Ans)

Aggregation

Aggregation is a process when relation between two entities is treated as a **single entity**.



In the diagram above, the relationship between **Center** and **Course** together, is acting as an Entity, which is in relationship with another entity **Visitor**. Now in

real world, if a Visitor or a Student visits a Coaching Center, he/she will never enquire about the center only or just about the course, rather he/she will ask enquire about both.

Example:

Consider the query "Find the average salary of instructors in the Computer Science department."We write this query as follows:

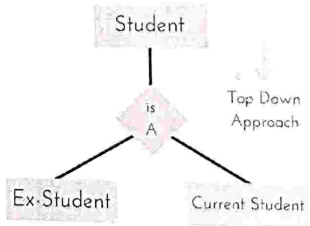
```
select avg (salary)
from instructor
where dept name= 'Comp. Sci.';
```

The result of this query is a relation with a single attribute, containing a single tuple with a numerical value corresponding to the average salary of instructors in the Computer Science department. The database system may give an arbitrary name to the result relation attribute that is generated by aggregation; however, we can give a meaningful name to the attribute by using the as clause as follows:

```
select avg (salary) as avg salary
from instructor
where dept name= 'Comp. Sci.';
```

12Q) Explain about composition or Specialization?

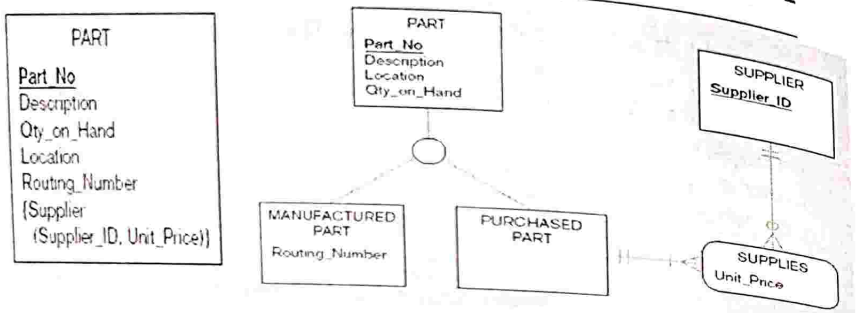
Ans) The process of defining one or more subtypes of the supertype and forming supertype/subtype relationships is called Specialization. It is also called composition. Each subtype is formed based on some distinguishing characteristic such as attributes or relationships specific to the subtype.



Specialization is opposite to Generalization. It is a top-down approach in which one higher level entity can be broken down into two lower level entity. In specialization, a higher level entity may not have any lower-level entity sets, it's possible.

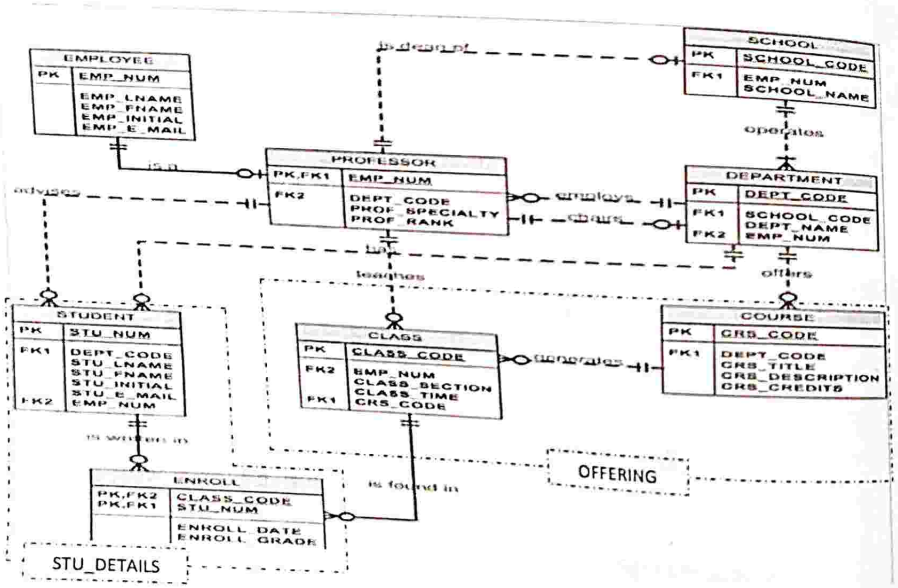
Example:

Entity type PART Specialization to MANUFACTURED PART and PURCHASED PART



13Q) Explain about entity clusters?

Ans) EER diagrams are difficult to read when there are too many entities and relationships, to reduce this problem we use Entity Clusters. An **Entity cluster** is a set of one or more entity types and associated relationships grouped into a single abstract entity type.



The above EER diagram contains two entity clusters.

- OFFERING, which group the COURSE and CLASS entities and Relationships.
- STU_DETAILS, which group the STUDENT and ENROLL entities and Relationships.

14Q) Explain about connection types?

Ans) Entity:

An Entity is a person, place, object, event or concept in the user environment about which the organization wishes to maintain data.

Ex:- Person : Employee, Student, Patient
Place : Store, Warehouse, State
Object : Machine, Building, Automobile
Event : Sale, Registration, Renewal
Concept: Account, Course, Workcenter

Entity Type:

An Entity type is a collection of entities that share common properties or characteristics.

Entity instance :

An Entity instance is a single occurrence of an entity type.

Ex:- Entity type : Employee
Attributes: ENO, ENAME , ADDRESS
Two Entity instances are
1 Arun ATP
2 Babu ATP

Naming and defining Entity types:

The following are some guidelines for naming Entity types.

1. An entity type name is a singular noun, such as CUSTOMER, STUDENT, AUTOMOBILE.
2. The name should be descriptive for the organization and distinct from all other entity type names within that organization.
3. The abbreviation or short name should be specified for each entity type name.
4. Event entity types should be named for the result of the event, not the activities or process of the event.
5. The name used for the same entity type should be the same on all E-R diagrams.

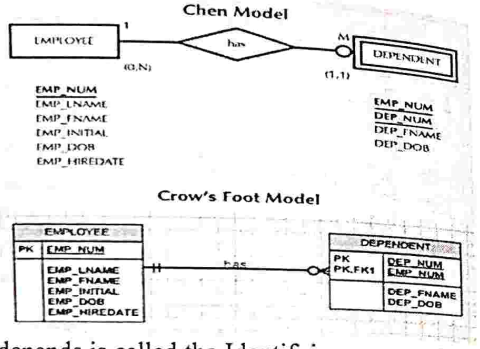
Entity Types are classified into two types. They are 1)Strong Entity type 2) Weak Entity type.

Strong Entity type :

A strong entity type is an entity type that exists independently of other entity types. Instances of a strong entity type always have a unique characteristic called an Identifier.

Weak entity type :

A Weak entity type is an entity type whose existence depends on some other entity type. A weak entity type has no business meaning in the E-R diagram without the entity type on which it depends.

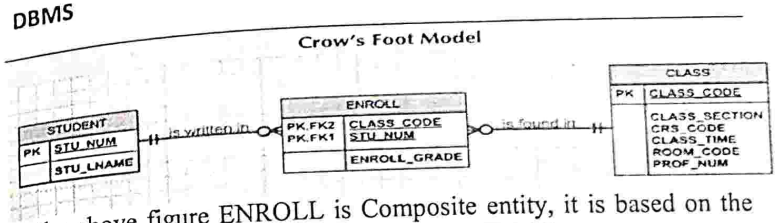


The entity type on which the weak entity type depends is called the Identifying owner. A weak entity type does not have its own Identifier. The relationship between a weak entity type and its owner is called an Identifying relationship.

In the above figure EMPLOEE is strong entity type and DEPENDENT is weak entity type.

Associative (Composite) Entities:

The associative entity is composed of the primary keys of each of the entities to be connected. The associative entities are used to implement a M:M relationship between two or more entities. The associative entity is also called as composite or bridge entity. The associative entity may also contain additional attributes that play no role in the connective process.



In the above figure ENROLL is Composite entity, it is based on the primary keys of the entities STUDENT and CLASS.

15Q) Explain Advantages of ER Model in DBMS

The Entity-Relationship (ER) model was originally proposed by Peter in 1976 as a way to unify the network and relational database views. Simply stated, the ER model is a conceptual data model that views the real world as entities and relationships. A basic component of the model is the Entity-Relationship diagram, which is used to visually represent data objects.

ER model is a logical representation of an enterprise data. ER model is a diagrammatic representation of logical structure of database.

Advantages of ER Model

1.Conceptually it is very simple: ER model is very simple because if we know relationship between entities and attributes, then we can easily draw an ER diagram.

2.Better visual representation: ER model is a diagrammatic representation of any logical structure of database. By seeing ER diagram, we can easily understand relationship among entities and relationship.

3.Effective communication tool: It is an effective communication tool for database designer.

4.Highly integrated with relational model: ER model can be easily converted into relational model by simply converting ER model into tables.

5.Easy conversion to any data model: ER model can be easily converted into another data model like hierarchical data model, network data model and so on.

UNIT – 3

Relational Model

1Q) Explain about CODD Rules?

Ans) Codd's Rule for Relational DBMS

E.F Codd was a Computer Scientist who invented the **Relational model** for Database management. Based on relational model, the **Relational database** was created. Codd proposed 13 rules popularly known as **Codd's 12 rules** to test DBMS's concept against his relational model. Codd's rule actually define what quality a DBMS requires in order to become a Relational Database Management System(RDBMS). Till now, there is hardly any commercial product that follows all the 13 Codd's rules. Even **Oracle** follows only eight and half(8.5) out of 13. The Codd's 12 rules are as follows.

Rule zero

This rule states that for a system to qualify as an **RDBMS**, it must be able to manage database entirely through the relational capabilities.

Rule 1: Information rule

All information(including metadata) is to be represented as stored data in cells of tables. The rows and columns have to be strictly unordered.

Rule 2: Guaranted Access

Each unique piece of data(atomic value) should be accesible by : **Table Name + Primary Key(Row) + Attribute(column).**

NOTE: Ability to directly access via POINTER is a violation of this rule.

Rule 3: Systematic treatment of NULL

Null has several meanings, it can mean missing data, not applicable or no value. It should be handled consistently. Also, Primary key must not be null, ever. Expression on NULL must give null.

Rule 4: Active Online Catalog

Database dictionary(catalog) is the structure description of the complete **Database** and it must be stored online. The Catalog must be governed by same rules as rest of the database. The same query language should be used on catalog as used to query database.

Rule 5: Powerful and Well-Structured Language

One well structured language must be there to provide all manners of access to the data stored in the database. Example: SQL, etc. If the database allows access to the data without the use of this language, then that is a violation.

Rule 6: View Updation Rule

All the view that are theoretically updatable should be updatable by the system as well.

Rule 7: Relational Level Operation

There must be Insert, Delete, Update operations at each level of relations. Set operation like Union, Intersection and minus should also be supported.

Rule 8: Physical Data Independence

The physical storage of data should not matter to the system. If say, some file supporting table is renamed or moved from one disk to another, it should not effect the application.

Rule 9: Logical Data Independence

If there is change in the logical structure(table structures) of the database the user view of data should not change. Say, if a table is split into two tables, a new view should give result as the join of the two tables. This rule is most difficult to satisfy.

Rule 10: Integrity Independence

The database should be able to enforce its own integrity rather than using other programs. Key and Check constraints, trigger etc, should be stored in Data Dictionary. This also make **RDBMS** independent of front-end.

Rule 11: Distribution Independence

A database should work properly regardless of its distribution across a network. Even if a database is geographically distributed, with data

stored in pieces, the end user should get an impression that it is stored at the same place. This lays the foundation of **distributed database.**

Rule 12: Nonsubversion Rule

If low level access is allowed to a system it should not be able to subvert or bypass integrity rules to change the data. This can be achieved by some sort of looking or encryption.

2Q) Explain about relational data model?

Ans)

Relational Data Model:

- The **relational model** was first developed by E. F. Codd (of IBM) in 1970.
- The relational model produced an "automatic transmission" database to replace the "standard transmission" databases that preceded it.
- The relational data model is implemented through a very sophisticated **relational database management system** (RDBMS).
- The RDBMS manages all of the physical details, while the user sees the relational database as a collections of tables in which data are stored.
- A relational table stores a collection of related entities.

Database name: Ch02_InsureCo Table name: AGENT (first six attributes)

AGENT_CODE	AGENT_LNAME	AGENT_FNAME	AGENT_INITIAL	AGENT_AREACODE	AGENT_PHONE
501	Alby	Alex	B	713	228-1249
502	Hahn	Leah	F	615	882-1244
503	Okon	John	T	615	123-5589

Link through AGENT_CODE

Table name: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_RENEW_DATE	AGENT_CODE
10010	Ramas	Alfred	A	615	844-2573	05-Apr-2004	502
10011	Dunne	Leona	K	713	894-1238	16-Jun-2004	501
10012	Smith	Kathy	W	615	894-2285	29-Jan-2005	502
10013	Olowski	Paul	F	615	894-2180	14-Oct-2004	502
10014	Orlando	Myron		615	222-1672	28-Dec-2004	501

Advantages:

1. Structural independence is promoted by the use of independent tables
2. Tabular view substantially improve conceptual simplicity.

3. Easier database design, implementation, management, and use.
4. Ad hoc query capability is based on SQL.
5. Powerful RDBMS isolates the end user from physical-level details and improves implementation and management simplicity.

Disadvantages:

1. The RDBMS requires Substantial hardware and system software overhead.
2. Conceptual simplicity gives relatively untrained people the tools to use a good system poorly, and if unchecked, it may produce the same data anomalies found in file system.
3. It may promote islands of information problems as individuals and departments can easily develop their own applications.

3Q) Explain about concept of key?(or) Define a Key and explain All Keys in DBMS?

Ans:

A key is an attribute (also known as column or field) or a combination of attribute that is used to identify records. Sometimes we might have to retrieve data from more than one table, in those cases we require to join tables with the help of keys. The purpose of the key is to bind data together across tables without repeating all of the data in every table.

The following are different types of keys:-

- Super Key
- Candidate Key
- Primary Key
- Secondary Key
- Foreign Key

Super Key – An attribute or a combination of attributes that are used to identify the records uniquely is known as Super Key. A table can have many Super Keys.

Ex:-

Stu_num
Stu_num, Stu_Name
Stu_num, Stu_Name, Stu_Address

Candidate Key – It can be defined as minimal Super Key. A super key that does not contain a subset of attributes that is itself a super key.

Ex:-

Stu_num
Stu_phone_num

Primary Key – A Candidate Key that is used by the database designer for unique identification of each row in a table is known as Primary Key. A Primary Key can consist of one or more attributes of a table.

Ex:-

Stu_num
Emp_Id, Emp_phone_num

Secondary Key – An attribute or combination of attributes used strictly for data retrieval purposes is called Secondary Key.

Ex:-

Stu_name, Stu_dob
Emp_name, Emp_city

Foreign Key – A foreign key is an attribute or combination of attribute in one base table that points to the primary key of another table. The purpose of the foreign key is to ensure referential integrity of the data.

Ex:-

Employee(Emp_id, Emp_name, Emp_basic)
Department(Dept_id, Dept_name, Emp_id)

In the above example Emp_id is the Primary key for Employee table and Foreign Key for Department table.

4Q) Explain about relational integrity?(or)Explain Integrity Constraints or Integrity Rules?

Ans) Integrity constraints are used to enforce the business rules. In Relational database integrity constraints are very important to good database design. Many RDBMSs enforce integrity constraints automatically. Integrity constraints are stored as part of the table definition.

Types of Integrity Constraints:-

- Entity Integrity Constraints

- Referential Integrity Constraints
- NOT NULL Integrity Constraints
- UNIQUE Key Integrity Constraints

Entity Integrity Constraints:

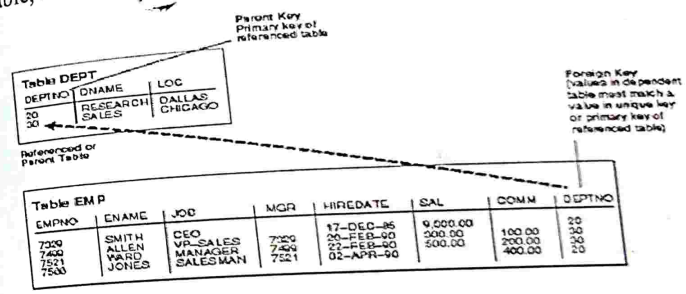
Entity integrity constraint requires that all primary key entries are unique, and no part of a primary key may be null.

Ex:-

No invoice can have a duplicate number, nor can it be null.
All students are uniquely identified by their HallTicket Number.

Referential Integrity Constraints

A referential integrity constraint requires that for each row of a table, the value in the foreign key matches a value in a parent key.



The above figure shows a foreign key defined on the DEPTNO column of the EMP table. It guarantees that every value in this column must match a value in the primary key of the DEPT table.

NOT NULL Integrity Constraints

By default, all columns in a table allow nulls. **Null** means th absence of a value. A NOT NULL constraint requires a column of table must contain a value.

UNIQUE Key Integrity Constraints

A UNIQUE key integrity constraint requires that every value a column or set of columns be unique--that is, no two rows of a ta have duplicate values in a specified column or set of columns.

5Q) Explain about Relation algebra and relation algebra operations?

Ans) Relational database systems are expected to be equipped with a query language that can assist its users to query the database instances. There are two kinds of query languages – relational algebra and relational calculus.

Relational Algebra: Relational algebra is a procedural query language, which takes instances of relations as input and yields instances of relations as output. It uses operators to perform queries. An operator can be either unary or binary. They accept relations as their input and yield relations as their output. Relational algebra is performed recursively on a relation and intermediate results are also considered relations.

The fundamental operations of relational algebra are as follows –

1. Select
2. Project
3. Union
4. Set different
5. Cartesian product
6. Rename

We will discuss all these operations in the following sections.

1. Select Operation (σ): It selects tuples that satisfy the given predicate from a relation.

Notation – σp(r). Where σ stands for selection predicate and r stands for relation. p is prepositional logic formula which may use connectors like and, or, and not. These terms may use relational operators like – =, ≠, ≥, < , >, ≤.

For example :

1.σsubject = "database"(Books)

Output :Selects tuples from books where subject is 'database'.

2.σsubject = "database" and price = "450"(Books)

Output: Selects tuples from books where subject is 'database' and 'price' is 450.

3.σsubject = "database" and price = "450" or year > "2010"(Books)

Output: Selects tuples from books where subject is 'database' and 'price' is 450 or those books published after 2010.

2. Project Operation (∏): It projects column(s) that satisfy a given predicate.

Notation: ∏A1, A2, An (r) Where A1, A2 , An are attribute names of relation r. Duplicate rows are automatically eliminated, as relation is a set.

For ex: ∏subject, author (Books)

Selects and projects columns named as subject and author from the relation Books.

3. Union Operation (∪): It performs binary union between two given relations and is defined as r ∪ s = { t | t ∈ r or t ∈ s}

Notation : r U s Where r and s are either database relations or relation result set .

For a union operation to be valid, the following conditions must hold –

- r, and s must have the same number of attributes.
- Attribute domains must be compatible.
- Duplicate tuples are automatically eliminated.

1.∏ author (Books) ∪ ∏ author (Articles)

Output – Projects the names of the authors who have either written a book or an article or both.

4.Set Difference (−): The result of set difference operation is tuples, which are present in one relation but are not in the second relation.

Notation: r − s

Q)Finds all the tuples that are present in r but not in s.

∏ author (Books) − ∏ author (Articles)

Output – Provides the name of authors who have written books but not articles.

5. Cartesian Product (X): Combines information of two different relations into one.

Notation:

r X s Where r and s are relations and their output will be defined as

r X s = { q t | q ∈ r and t ∈ s}

σauthor = 'tutorialspoint'(Books X Articles)

Output – Yields a relation, which shows all the books and articles written by tutorialspoint.

6.Rename Operation (ρ): The results of relational algebra are also relations but without any name. The rename operation allows us to

rename the output relation. 'rename' operation is denoted with small Greek letter rho ρ.

Notation: ρ x (E) Where the result of expression E is saved with name of x.

Additional operations are:

- Set intersection
- Assignment
- Natural join

6Q) Advantages of relational databases

Ans) Advantages:

1. Ease of use: The revision of any information as tables consisting of rows and columns is much easier to understand .

2. Flexibility: Different tables from which information has to be linked and extracted can be easily manipulated by operators such as project and join to give information in the form in which it is desired.

3. Precision: The usage of relational algebra and relational calculus in the manipulation of he relations between the tables ensures that there is no ambiguity, which may otherwise arise in establishing the linkages in a complicated network type database.

4. Security: Security control and authorization can also be implemented more easily by moving sensitive attributes in a given table into a separate relation with its own authorization controls. If authorization requirement permits, a particular attribute could be joined back with others to enable full information retrieval.

5. Data Independence: Data independence is achieved more easily with normalization structure used in a relational database than in the more complicated tree or network structure.

6. Data Manipulation Language: The possibility of responding to query by means of a language based on relational algebra and relational calculus

EX: SQL is easy in the relational database approach. For data organized in other structure the query language either becomes complex or extremely limited in its capabilities.

7Q) Explain limitations or Disadvantages of relational algebra?

Ans)Disadvantages/ limitations:

1. Performance: A major constraint and therefore disadvantage in the use of relational database system is machine performance. If the number of tables between which relationships to be established are large and the tables themselves effect the performance in responding to the sql queries.

2. Physical Storage Consumption: With an interactive system, for example an operation like join would depend upon the physical storage also. It is, therefore common in relational databases to tune the databases and in such a case the physical data layout would be chosen so as to give good performance in the most frequently run operations. It therefore would naturally result in the fact that the lays frequently run operations would tend to become even more shared.

3. Slow extraction of meaning from data: if the data is naturally organized in a hierarchical manner and stored as such, the hierarchical approach may give quick meaning for that data.

8Q) Explain about relational calculus? Explain types of relational calculus?

Or

Explain tuple relational calculus(TRC) and domain relational Calculus (DRC).

Ans) In contrast to Relational Algebra, Relational Calculus is a non-procedural query language, that is, it tells what to do but never explains how to do it.

Relational calculus exists in two forms –

1. Tuple Relational Calculus (TRC)
2. Domain Relational Calculus (DRC)

1.Tuple Relational Calculus (TRC): Filtering variable ranges over tuples

Notation – {T | Condition}

Returns all tuples T that satisfies a condition.

For example –

{ T.name | Author(T) AND T.article = 'database' }

Output – Returns tuples with 'name' from Author who has written article on 'database'.

TRC can be quantified. We can use Existential (∃) and Universal Quantifiers (∀).

For example –

{ R| ∃T ∈ Authors(T.article='database' AND R.name=T.name)}

Output – The above query will yield the same result as the previous one.

2.Domain Relational Calculus (DRC): In DRC, the filtering variable uses the domain of attributes instead of entire tuple values (as done in TRC, mentioned above).

Notation –

{ a1, a2, a3, ..., an | P (a1, a2, a3, ... ,an)}

Where a1, a2 are attributes and P stands for formulae built by inner attributes.

For example –

{< article, page, subject > | ∈ TutorialsPoint ∧ subject = 'database'}

Output – Yields Article, Page, and Subject from the relation TutorialsPoint, where subject is database.

Just like TRC, DRC can also be written using existential and universal quantifiers. DRC also involves relational operators. The expression power of Tuple Relation Calculus and Domain Relation Calculus is equivalent to Relational Algebra.

9Q) Explain about QBE?

Ans) Query by Example (QBE) is a database query language for relational databases. It was devised by Moshé M. Zloof at IBM Research during the mid-1970s, It is the first graphical query language, using visual tables where the user would enter commands, example elements and conditions. Many graphical front-ends for databases use the ideas from QBE today. Originally limited only for

the purpose of retrieving data, QBE was later extended to allow other operations, such as inserts, deletes and updates, as well as creation of temporary tables.

The motivation behind QBE is that a parser can convert the user's actions into statements expressed in a database manipulation language, such as SQL. Behind the scenes, it is this statement that is actually executed. A suitably comprehensive front-end can minimize the burden on the user to remember the finer details of SQL, and it is easier and more productive for end-users to select tables and columns by selecting them rather than typing in their names,

QBE is a seminal work in end-user development, frequently cited in research papers as an early example of this topic.QBE is supported in several relational database front ends, notably Microsoft Access, which implements "Visual Query by Example", as well as Microsoft SQL Server Enterprise Manager. It is also implemented in several object-oriented databases.

QBE is based on the logical formalism called tableau query, although QBE adds some extensions to that, much like SQL is based on the relational algebra.

UNIT – 4

Structured Query Language

1Q)Explain about History of SQL Standard?

Ans) History of SQL

SQL was invented and developed by IBM in early 1970's. It stands for Structured Query Language. The SQL implemented by ORACLE CORPORATION. It is useful to store, to modify and to retrieve the data. It is also useful to perform calculations.

Benefits of SQL Commands:

It is a Non-Procedure Language, because multiple records can access rather than accessing a single record.

It is a portable language, because any RDBMS Commands can ran on same SQL.

It is a simple Language, Which contains simple commands like create, insert, delete update etc.,

2Q) Explain about Commands in SQL?

Ans)

SQL Commands are classified into two types. They are:

DDL : DATA DEFINITION LANGUAGE.

CREATE TABLE
ALTER TABLE
TRUNCATE TABLE
DROP TABLE

DML : DATA MANIPULATION LANGUAGE

SELECT
INSERT
DELETE
UPDATE.
COMMIT
ROLLBACK

3Q)Explain about Data Types in SQL?

Ans)

In order to create a table we need to specify a data type for individual columns in the create table command. The following are some data types supported by SQL.

CHARACTER DATATYPES:

The following are the character data types supported by Oracle.

CHAR:

This data type is used when a fixed length character string is required. It can store alphanumeric values. The column length of such a data type can vary between 1- 255 bytes. By default is one byte.

- If the user enters a value shorter than the specified length then the database would blank -pad to the fixed length.
- In case, If the user enters a value larger than the specified length then the database would return an error.

VARCHAR2:

It supports a variable length character string. It also stores alphanumeric values. Maximum size for this data type would be from 1 - 2000 bytes. while defining this data type we should specify the size. Using varchar2 we can save disk space when compared to char. This statement can be justified with the help of an example. Considering a column assigned with varchar2 data type of size 30 bytes, If the user enters 10 bytes of character. Then the column length in that row would only be 10 bytes and not 30 bytes. In the case of char. '" would still occupy 30 bytes because the remaining would be blank padded by Oracle.

NOTE: Currently varchar datatype is equivalent to varchar2 datatype.

DATE:

Date data type is used to store date and time in a table. Oracle database makes use of its own format to store date in a fixed length of 7 bytes each for century, month ,date ,year, hour ,minute and second. Default date data type is "dd-mon-yy". To view system's date and time we can use the SQL function called sysdate( ) .Valid date is from Jan 1,4712 BC to Dec 31,4712 AD.

NUMBER:

This data type can store positive numbers, negative numbers, zeroes, fixed point numbers, floating point numbers of magnitude ranging between 1.0* 10 -130 to 9,9..9*10 125 with a position of 38.

- column name number
- column _name number (p)
- column _name number (p,s)

{p=38,s=0} {fixed point} {floating point}

Where p is the precision which refers to the total number of digits, It varies between 1 to 38, S is the scale which refers to number of digits to the right of the decimal point, which varies between -84 to 127 .

4Q)Explain about Data Definition Language?

Ans) DDL COMMANDS

The Data Definition Language is used to create an object(e.g. table), alter the structure of an object and also drop the object.

CREATE TABLE:

It is useful to create a new table.

Syntax:

create table <table name>(column definition, column definition,……..);

Column is the name of the Column in a table and definition includes the data type, width, and constraints.

Rules to write a table name:

- In a table name the first character must be an alphabet, remaining characters may be either alphabet or numeric.
- Blank spaces and special symbols are not allowed, except under score.
- The maximum length of a table name is '80' char.
- Two different table should not have the same name,.
- A table name should not be a reserve word.

Ex: -

Create table emp (eno number(4), ename varchar2(25), esal number(7,2));

ALTER TABLE:

It is used to modify the structure of an existed table.

1. to add new columns

Syntax:

alter table < table name> add (columns datatype);

Example:

alter table emp add (caddress ~~ename~~ varchar2(20));

2. to modify existed columns: Syntax:

. alter table <table name> modify (columns

Example:

alter table emp modify (ename varchar2(10));

TRUNCATE TABLE:

It is useful to delete rows or records from selected tables.

Syntax:

truncate table <table name>;

Example: truncate table emp;

DROP TABLE:

It deletes a table including rows and structure of a table.

Syntax:

drop table <table name >;

Example:

drop table emp;

DESC : [DISCRIPTION COMMAND)

It displays structure of a table.

Syntax:

desc <table name>;

Example:

desc emp;

5Q)Explain about Selection & projection Operation?

Ans) Select Operation: This operation is used to select rows from a table (relation) that specifies a given logic, which is called as a predicate. The predicate is a user defined condition to select rows of user's choice.

Syntax:

select * from table1 where field1 = 'Value';

Ex: display student details whose result is pass

select * from student where result="pass";

Here It will display all fields but display students whose result is pass

Project Operation : If the user is interested in selecting the values of a few attributes, rather than selection all attributes of the Table (Relation), then one should go for PROJECT Operation.

Projection means choosing **which columns** (or expressions) the query shall return.

Syntax:

select field1,field2 from table1 ;

Ex: display student details like sno, name only.

select sno, sname from student;

Here It will display all students details but only two fields like sno and name not all fields.

Selection and Projection:

SELECT empno,ename,dno,job from Emp WHERE job='CLERK';

In the above query the columns "empno, ename, dno,job" those comes under projection, "where job='clerk'" comes under selection

6Q)Explain about Aggregate functions?

Ans)

SQL can perform various mathematical summaries such as counting the number of rows that contain a specified condition, finding the minimum or maximum values for some specified attribute, and averaging the values in a specified column. These functions returns one value based on the group of rows.

MAX( ) :

This function is useful to returns the highest value from the entire column.

Syntax:

Max (column name)

Example:

select max (esal) from emp;

MIN( ) :

This function is useful to returns the smallest value from the entire table.

Syntax:

Min (column name)

Example: Select min(esal) from emp;

SUM( ):

This function is useful to ad all values in a column and returns the total value.

Syntax:

sum(column name)

Example: Select sum (esal) from emp;

A VG ( ):

This function is useful to calculate the average value of a column.

Syntax:

AVG(column name)

Example:

Select avg(esal) from emp;

COUNT:

This function is useful to count the number of rows or number of values in a table. three types of formats.

COUNT (*):

This function counts all rows including duplicate rows and null rows.

Example:

select count(*) from emp;

COUNT (COLUMN NAME):

This function counts all values in a column without including null values.

Example:

select count (ename) from emp;

COUNT (DISTINCT COLUMN NAME)

This function counts all values in a table without including duplicate values and null values.

Example:

select count (distinct ename) from emp;

7Q)Explain about Data Manipulation Language?

Ans) DML COMMANDS

INSERT COMMAND:

It is useful to add one or more rows to a table.

a) to add values to all columns:

Syntax:

insert into <table name> values (field1_value,field2_value,filed3_value,……);

Rules to write list of values:

The values must be suppurated by commas.

Character and Date type values must be enclosed within single quotations. The values must match the type, number and order of columns names.

Create a table details.

```
create table details (name varchar2 (20), age number(3));
insert into details values('swamy',40);
insert into details values('raju' ,20);
```

b) to insert values using column address:

Syntax:

Insert into <table name> values (&column1_name, &column2_name,……);

Example:

Insert into details values ('&name',&age);

SELECT COMMAND:

Select command is a Query on database. A Query is a request for information. It is a most commonly used database operation to display records of the table.

To display table:

Example:

Select * from emp;

To select a particular table:

Syntax:

select <table name> from tab;

Example: .

select telephone from tab;

To display records from a table:

Syntax:

Select <columns (or) * > from <table name> [where <condition] [order by column name]

Example:

select name from telephone;

select * from telephone;

It displays all rows from a telephone table.

To display specified columns information:

Example:

Select name, calls from telephone;

Where clause:

It is useful to add conditions in select commands.

Syntax:

select <columns (or) * > from <table name> where <condition>; .

Example:

select * from telephone where calls >20;

UPDATE COMMAND:

It is useful to modify the contents of a selected table.

Syntax.

update<table name>set column=value, column=value,.....
where <condition>;

Example:

update telephone set price=l where calls =20;

DELETE COMMAND:

It is useful to delete records from a selected table.

Syntax:

delete from <table name > where <condition>;

Example:

delete from telephone where calls=0;

8Q) Explain about Table Modification Commands?

Ans)

Times change, needs change, and so do the requirements of your database. It may be the case that you need to alter a column's name, add a new column, change the data type of a column, or remove the table altogether. DDL provides a way for you to make such changes.

The following modification on the table structure:

- Renaming a table
- Renaming a column
- Changing a column's data type
- Adding a constraint
- Removing a constraint
- Adding a column
- Removing a column
- Dropping a table

Existing tables can be altered with an ALTER TABLE statement. An ALTER TABLE statement is part of DDL, and is for altering a table **schema** only; we'll look at updating *data* in a table later in this book.

The basic format of an ALTER TABLE statement is:

ALTER TABLE table_to_change HOW TO CHANGE THE TABLE additional arguments

Although the SQL statements for all of these actions use the same initial ALTER TABLE clause, the specific syntax for each varies according to the action.

Action	Command	Notes
Add a column to a table	ALTER TABLE table_name ADD COLUMN column_name data_type CONSTRAINTS;	Alters a table by adding a column with a specified data type and optional constraints.
Alter a column's data type	ALTER TABLE table_name ALTER COLUMN column_name TYPE data_type;	Alters the table by changing the datatype of column.
Rename a table	ALTER TABLE table_name RENAME TO new_table_name;	Changes the name of a table in the currently connected to database.
Rename a column within a table	ALTER TABLE table_name RENAME COLUMN column_name TO new_column_name;	Renames a column of the specified table.
Add column constraint	ALTER TABLE table_name ALTER COLUMN column_name SET constraint clause;	Adds a specified constraint to the specified table column.
Add table constraint	ALTER TABLE table_name ADD CONSTRAINT constraint_name constraint clause;	Adds a specified constraint to the specified table.
Remove a column constraint	ALTER TABLE table_name ALTER COLUMN column_name	Removes a constraint from the specified table.

	DROP CONSTRAINT;	
Remove a table constraint	ALTER TABLE table_name DROP CONSTRAINT constraint_name;	Removes a constraint from the specified table.
Remove a column from a table	ALTER TABLE table_name DROP COLUMN column_name	Removes a column from the specified table.
Delete a table from the database	DROP TABLE table_name;	Permanently deletes the specified table from its database.

9Q)Explain about Table Truncation?

Ans) The SQL TRUNCATE TABLE command is used to delete complete data from an existing table.

You can also use DROP TABLE command to delete complete table but it would remove complete table structure form the database and you would need to re-create this table once again if you wish you store some data.

Syntax: The basic syntax of a TRUNCATE TABLE command is as follows.

TRUNCATE TABLE table_name;

Example: Following is the example of a Truncate command.

SQL > TRUNCATE TABLE CUSTOMERS;

Now, the CUSTOMERS table is truncated and the output from SELECT statement will be as shown in the code block below –

SQL> SELECT * FROM CUSTOMERS;
Empty set (0.00 sec)

10Q) Explain about Imposition of Constraints in SQL?

Ans)

Constraints are the rules enforced on the data columns of a table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database.

Constraints could be either on a column level or a table level. The column level constraints are applied only to one column, whereas the table level constraints are applied to the whole table.

- NOT NULL Constraint – Ensures that a column cannot have NULL value.
- DEFAULT Constraint – Provides a default value for a column when none is specified.
- UNIQUE Constraint – Ensures that all values in a column are different.
- PRIMARY Key – Uniquely identifies each row/record in a database table.
- FOREIGN Key – Uniquely identifies a row/record in any of the given database table.
- CHECK Constraint – The CHECK constraint ensures that all the values in a column satisfies certain conditions.
- INDEX – Used to create and retrieve data from the database very quickly.

Constraints can be specified when a table is created with the CREATE TABLE statement or you can use the ALTER TABLE statement to create constraints even after the table is created.

Dropping Constraints: Any constraint that you have defined can be dropped using the ALTER TABLE command with the DROP CONSTRAINT option.

For example, to drop the primary key constraint in the EMPLOYEES table, you can use the following command.

ALTER TABLE EMPLOYEES DROP CONSTRAINT EMPLOYEES_PK;

Some implementations may provide shortcuts for dropping certain constraints. For example, to drop the primary key constraint for a table in Oracle, you can use the following command.

ALTER TABLE EMPLOYEES DROP PRIMARY KEY;

Integrity Constraints: Integrity constraints are used to ensure accuracy and consistency of the data in a relational database.

There are many types of integrity constraints that play a role in Referential Integrity (RI). These constraints include Primary Key,

mentioned above. Unique Constraints and other constraints which are

11Q)Explain about Join Operation?

We can join the rows of two tables by using where clause to combines the specific rows. The two tables must be related tables.

Syntax:

Select <columns> from table 1, table2, where <condition>;

Joins are four types:

1. Simple Join or Cross Join
2. Natural Join or Equi Join
3. Outer Join
4. Self Join

SIMPLE JOIN:

This is the most common join, we can join the data from two tables, which contains common Column.

Ex:- Select * from T1, T2;

A Cross Join performs a relational product of two tables.

Ex: - Select * from T1 CROSS JOIN T2;

NATURAL JOIN: Returns only the rows with matching values in the matching columns. The matching columns must have the same names and similar data types.

Ex:- Select * from T1 NATURAL JOIN T2;

EQUI JOIN:

A join which is done based on the equalities is called equijoin. We care using "=" operator in where clause.

Example:

Select * from em,dept where em.deptno=dept.deptno;

select * from emp.names, dept.deptnames from emp,dept where emp.deptno=dept.deptno;

NON- EQUI JOIN:

In this join we are using only relational operators [<, >,!=) other than"=".

Example:

select * from emp,dept where emp.deptno>dept.deptno;

OUTER JOIN:

A Outer Join produces rows that do not have matching values in common are also included in the result table.

There are three types of outer join.

1. Left outer join
2. Right outer join
3. Full outer join

Left outer join : - Returns rows with matching values and includes all rows from the left table with unmatched values.

Ex:- Select * from T1 left outer join T2 on T1.c1=T2.c1;

Right outer join:- Returns rows with matching values and includes all rows from the right table with unmatched values.

Ex:- Select * from T1 right outer join T2 on T1.c1=T2.c1;

Full outer join:- Returns rows with matching values and includes all rows from the both tables with unmatched values.

Ex:- Select * from T1 full outer join T2 on T1.c1=T2.c1;

ALIAS NAMES:

We can take the other names to actual table. The temporary names assign to a table is called Alias name. We can take the alias names insted of actual table name.

Example:-

Select a.ename,b.deptname from emp a, emp b where a.deptno=b.deptno;

We can also take two alias names for a single table to assume a single table as two tables.

SELF JOIN:

A join of a table joining the rows of a table to the rows of the same table is called self join/ We can join the one row with the other rows in a table.

Example:

select a.ename, b.ename from emp a, emp b where a.salary >b.salary order by a.ename;

12Q) Explain about Set Operation?
Ans)
The set operators are useful to combines the result of the queries into one. The two tables must contain the same number of columns without long data type to use the set operator. These are four type:

1. Unions
2. Union all
3. Intersect
4. Minus.

UNION:

This operator combines all rows selected by either query, without including duplicate rows.
Example:

select * from emp1 union select * from emp2;

UNION ALL:

This operator combines all rows selected by either query including duplicate rows.
Example:

select * from emp1 union all select * from emp2;

INTERSECT:

This operator displays all common rows selected by both queries.
Example:

select * from emp1 intersect select * from emp2;

MINUS:

This operator cancels the common rows selected by both queries and displays the other rows selected only by the first query.
Example:

select * from emp1 minus select * from emp2;

13Q)Explain about View? (or) Virtual Tables.
Ans) A VIEW is a virtual table, through which a selective portion of the data from one or more tables can be seen. Views do not contain data of their own. They are used to restrict access to the database or to hide data complexity.

A view is stored as a SELECT statement in the database. DML operations on a view like INSERT, UPDATE, DELETE affects the

data in the original table upon which the view is based.
Syntax:
CREATE VIEW view_name AS SELECT column_list FROM table_name [WHERE condition];
Here view_name is the name of the VIEW.
The SELECT statement is used to define the columns and rows that you want to display in the view.
Ex:-
CREATE VIEW view_p AS SELECT product_id, product_name FROM product;

14Q)Explain about Sub Query?
Ans) Ans) A sub query is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved. Sub queries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN etc.
There are a few rules that sub queries must follow:

- Sub queries must be enclosed within parentheses.
- A sub query can have only one column in the SELECT clause, unless multiple columns are in the main query for the sub query to compare its selected columns.
- An ORDER BY cannot be used in a sub query, although the main query can use an ORDER BY. The GROUP BY can be used to perform the same function as the ORDER BY in a sub query.
- Sub queries that return more than one row can only be used with multiple value operators, such as the IN operator.
- A sub query cannot be immediately enclosed in a set function.
- The BETWEEN operator cannot be used with a sub query; however, the BETWEEN can be used within the sub query.

Sub queries with the SELECT Statement:
Ex:
SQL> SELECT * FROM CUSTOMERS WHERE ID IN (SELECT ID FROM CUSTOMERS WHERE SALARY > 4500) ;
Sub queries with the INSERT Statement:

Sub queries also can be used with INSERT statements. The INSERT statement uses the data returned from the sub query to insert into another table.

Example:

Consider a table CUSTOMERS_BKP with similar structure as CUSTOMERS table. Now to copy complete CUSTOMERS table into CUSTOMERS_BKP, following is the syntax:

```
SQL> INSERT INTO CUSTOMERS_BKP SELECT * FROM
     CUSTOMERS WHERE ID IN (SELECT ID FROM
     CUSTOMERS) ;
```

Sub queries with the UPDATE Statement:

The sub query can be used in conjunction with the UPDATE statement. Either single or multiple columns in a table can be updated when using a sub query with the UPDATE statement.

Example:

Following example updates SALARY by 0.25 times in CUSTOMERS table for all the customers whose AGE is greater than or equal to 27:

```
SQL> UPDATE CUSTOMERS SET SALARY = SALARY * 0.25
     WHERE AGE IN (SELECT AGE FROM CUSTOMERS_BKP
     WHERE AGE >= 27 );
```

Sub queries with the DELETE Statement:

The sub query can be used in conjunction with the DELETE statement like with any other statements mentioned above.

Example:

Following example deletes records from CUSTOMERS table for all the customers whose AGE is greater than or equal to 27:

```
SQL> DELETE FROM CUSTOMERS WHERE AGE IN (SELECT
AGE FROM CUSTOMERS_BKP WHERE AGE > 27 );
```

Correlated subquery:

A 'correlated sub-query' is a term used for specific types of queries in SQL in computer databases. It is a sub-query (a query nested inside another query) that uses values from the outer query in its WHERE clause. The sub-query is evaluated once for each row processed by the outer query.

Ex: -

```
SELECT employee_number, name FROM employee AS e1 WHERE
```



```
salary > (SELECT avg(salary) FROM employee WHERE
department = e1.department);
```

In the above example we are finding the list of employees (employee number and names) having more salary than the average salary of all employees in that employee's department.

15Q)Explain about Embedded SQL?

Ans)

1. SQL provides a powerful declarative query language. However, access to a database from a general-purpose programming language is required because,
   - SQL is not as powerful as a general-purpose programming language. There are queries that cannot be expressed in SQL, but can be programmed in C, Fortran, Pascal, Cobol, etc.
   - Non declarative actions -- such as printing a report, interacting with a user, or sending the result to a GUI -- cannot be done from within SQL.
2. The SQL standard defines embedding of SQL as embedded SQL and the language in which SQL queries are embedded is referred as host language.
3. The result of the query is made available to the program one tuple (record) at a time.
4. To identify embedded SQL requests to the preprocessor, we use **EXEC SQL** statement:

   EXEC SQL embedded SQL statement END-EXEC

   Note: A semi-colon is used instead of END-EXEC when SQL is embedded in C or Pascal.
5. Embedded SQL statements: declare cursor, open, and fetch statements.

```
EXEC SQL
        declare c cursor for
        select cname, ccity
        from deposit, customer
        where deposit.cname = customer.cname and
deposit.balance > :amount
        END-EXEC
```

Where amount is a host-language variable.

EXEC SQL open c END-EXEC

A series of fetch statement are executed to make tuples of the results available to the program.

EXEC SQL fetch c into :cn, :cc END-EXEC

EXEC SQL close c END-EXEC

6. Embedded SQL can execute any valid update, insert, or delete statements.
7. Dynamic SQL component allows programs to construct &submit SQL queries ar run time

SQL-92 also contains a module language, which allows procedures to be defined in SQL

UNIT – 5

PL/SQL

1Q) Explain about Structure of PL/SQL with example?

Ans) The Basic Syntax of PL/SQL which is a block-structured language; this means that the PL/SQL programs are divided and written in logical blocks of code. Each block consists of three sub-parts

1 Declarations: This section starts with the keyword DECLARE. It is an optional section and defines all variables, cursors, subprograms, and other elements to be used in the program.

2 Executable Commands: This section is enclosed between the keywords BEGIN and END and it is a mandatory section. It consists of the executable PL/SQL statements of the program. It should have at least one executable line of code, which may be just a NULL command to indicate that nothing should be executed.

3 Exception Handling: This section starts with the keyword EXCEPTION. This optional section contains exception(s) that handle errors in the program.

Every PL/SQL statement ends with a semicolon (;). PL/SQL blocks can be nested within other PL/SQL blocks using BEGIN and END.

Following is the basic structure of a PL/SQL block –

```
DECLARE
  <declarations section>
BEGIN
  <executable command(s)>
EXCEPTION
```

```
                    <exception handling>
                END;
```

The 'Hello World' Example

```
DECLARE
  message  varchar2(20):= 'Hello, World!';
BEGIN
  dbms_output.put_line(message);
END;
/
```

The end; line signals the end of the PL/SQL block. To run the code from the SQL command line, you may need to type / at the beginning of the first blank line after the last line of the code. When the above code is executed at the SQL prompt, it produces the following result –

Output: Hello World
PL/SQL procedure successfully completed.

2Q) Explain about PL/SQL Language Elements

Ans) A variable is nothing but a name given to a storage area that our programs can manipulate. Each variable in PL/SQL has a specific data type, which determines the size and the layout of the variable's memory; the range of values that can be stored within that memory and the set of operations that can be applied to the variable.

Variable Declaration in PL/SQL:PL/SQL variables must be declared in the declaration section or in a package as a global variable.

The syntax for declaring a variable :

variable_name [CONSTANT] datatype [NOT NULL] [:= | DEFAULT initial_value]

Where, variable_name is a valid identifier in PL/SQL, datatype must be a valid PL/SQL data type or any user defined data type Some valid variable declarations along with their definition are shown below –

```
sales number(10, 2);
pi CONSTANT double precision := 3.1415;
name varchar2(25);
address varchar2(100);
```


When you provide a size, scale or precision limit with the data type, it is called a constrained declaration. Constrained declarations require less memory than unconstrained declarations. For example –

```
sales number(10, 2);
name varchar2(25);
address varchar2(100);
```

Initializing Variables in PL/SQL:

Whenever you declare a variable, PL/SQL assigns it a default value of NULL. If you want to initialize a variable with a value other than the NULL value, you can do so during the declaration, using either of the following –

- The DEFAULT keyword
- The assignment operator

```
counter binary_integer := 0;
greetings varchar2(20) DEFAULT 'Have a Good Day';
```

You can also specify that a variable should not have a NULL value using the NOT NULL constraint. If you use the NOT NULL constraint, you must explicitly assign an initial value for that variable.

Variable Scope in PL/SQL:

PL/SQL allows the nesting of blocks, i.e., each program block may contain another inner block. If a variable is declared within an inner block, it is not accessible to the outer block. There are two types of variable scope –

a) Local variables – Variables declared in an inner block and not accessible to outer blocks.

b) Global variables – Variables declared in the outermost block or a package.

3Q)Explain about Data Types

Ans) PL/SQL - Data Types:

Data Types in PL/SQL are variables, constants and parameters must have a valid data type, which specifies a storage format, constraints, and a valid range of values. We will focus on the SCALAR and the LOB data types in this chapter. The other two data types will be covered in other chapters.

PL/SQL Scalar Data Types and Subtypes

S.No	Date Type & Description
1	**Numeric:** Numeric values on which arithmetic operations are performed.
2	**Character:** Alphanumeric values that represent single characters or strings of characters.
3	**Boolean:** Logical values on which logical operations are performed.
4	**Datetime:** Dates and times.

PL/SQL Character Data Types and Subtypes

S.No	Data Type & Description
1	**CHAR:** Fixed-length character string with maximum size of 32,767 bytes
2	**VARCHAR2:**Variable-length character string with maximum size of 32,767 bytes
3	**RAW:** Variable-length binary or byte string with maximum size of 32,767 bytes, not interpreted by PL/SQL
4	**NCHAR:** Fixed-length national character string with maximum size of 32,767 bytes
5	**NVARCHAR2:**Variable-length national character string with maximum size of 32,767 bytes
6	**LONG:** Variable-length character string with maximum size of 32,760 bytes
7	**LONG RAW:** Variable-length binary or byte string with maximum size of 32,760 bytes, not interpreted by PL/SQL
8	**ROWID:**Physical row identifier, the address of a row in an ordinary table

PL/SQL Boolean Data Types:

BOOLEAN *data type stores logical values that are used in logical operations. The logical values are the Boolean values* **TRUE** *and* ***FALSE*** *and the value* **NULL**.

PL/SQL Datetime and Interval Types

The DATE datatype is used to store fixed-length datetimes,

Field Name	Valid Datetime Values
YEAR	-4712 to 9999 (excluding year 0)
MONTH	01 to 12
DAY	01 to 31
HOUR	00 to 23
MINUTE	00 to 59
SECOND	00 to 59.9(n), where 9(n) is the precision of time fractional seconds

PL/SQL Large Object (LOB) Data Types:

Data Type	Description	Size
BFILE	Used to store large binary objects in operating system files outside the database.	System-dependent. Cannot exceed 4 gigabytes (GB).
BLOB	Used to store large binary objects in the database.	8 to 128 terabytes (TB)
CLOB	Used to store large blocks of character data in the database.	8 to 128 TB

NULLs in PL/SQL:PL/SQL NULL values represent **missing** or **unknown data** and they are not an integer, a character, or any other specific data type. Note that **NULL** is not the same as an empty data string or the null character value **'\0'**.

4Q)Explain about Operators Precedence

Ans)

Operators are used to specify conditions in an SQL statement and to serve as conjunctions for multiple conditions in a statement.

- Arithmetic operators
- Comparison operators
- Logical operators
- Special operators

1.ARTHMATIC OPERATORS:

Operator	Description
+	Addition - Adds values on either side of the operator
-	Subtraction - Subtracts right hand operand from left hand operand
*	Multiplication - Multiplies values on either side of the operator
/	Division - Divides left hand operand by right hand operand
^	power of – to calculate power of a given number

2.REATIONAL (OR) COMPARISION OPERATORS:

Operator	Description
=	Checks if the value of two operands are equal or not, if yes then condition becomes true.
!=	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.
<>	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.

3.LOGICAL OPERATORS

Operator	Description
AND	The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause.
NOT	The NOT operator reverses the meaning of the logical operator with which it is used. Ex. NOT EXISTS, NOT BETWEEN, NOT IN etc. This is negate operator.
OR	The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause.

4.SPECIAL OPERATORS

BETWEEN IS NULL IN LIKE

BETWEEN: Used to check whether attribute value is within a range of values.

Example:

select * from telephone where calls between 0 AND 20;

IS NULL : Used to check whether attribute value is null.

Example:

select p_code, p_indate from product where p_indate is null;

IN: Used to check whether attribute value matches any value within a value list

Example:

select * from telephone where calls in (10,20,30);

LIKE: Used to check whether attribute value matches given string pattern

Example:

select * from telephone where consumer like '%S';

EXISTS : Used to check if subquery returns any rows.

Example :

Select * from vendor where exists(select * from product where p_qoh<=p_min);

5Q)Explain about Control Structure

ANS)

1.IF- THEN STATEMENT: It is the simplest form of the IF control statement, frequently used in decision-making and changing the control flow of the program execution. The IF statement associates a condition with a sequence of statements enclosed by the keywords THEN and END IF. If the condition is TRUE, the statements get

executed, and if the condition is FALSE or NULL, then the IF statement does nothing.

Syntax: Syntax for IF-THEN statement is –

```
IF condition THEN
  S;
END IF;
```

Where condition is a Boolean or relational condition and S is a simple or compound statement.

2. IF THEN ELSE STATEMENT: A sequence of IF-THEN statements can be followed by an optional sequence of ELSE statements, which execute when the condition is FALSE.

Syntax: Syntax for the IF-THEN-ELSE statement is –

```
IF condition THEN
  S1;
ELSE
  S2;
END IF;
```

Where, S1 and S2 are different sequence of statements. In the IF-THEN-ELSE statements, when the test condition is TRUE, the statement S1 is executed and S2 is skipped; when the test condition is FALSE, then S1 is bypassed and statement S2 is executed.

3.CASE STATEMENT: Like the IF statement, the CASE statement selects one sequence of statements to execute. However, to select the sequence, the CASE statement uses a selector rather than multiple Boolean expressions. A selector is an expression, the value of which is used to select one of several alternatives.

Syntax:The syntax for the case statement in PL/SQL is –

```
CASE selector
  WHEN 'value1' THEN S1;
  WHEN 'value2' THEN S2;
  WHEN 'value3' THEN S3;
  ...
  ELSE Sn; -- default case
END CASE;
```


4.NESTED IF-ELSE STATEMENT: It is always legal in PL/SQL programming to nest the IF-ELSE statements, which means you can use one IF or ELSE IF statement inside another IF or ELSE IF statement(s).

Syntax:

```
IF( boolean_expression 1)THEN
  -- executes when the boolean expression 1 is true
  IF(boolean_expression 2) THEN
    -- executes when the boolean expression 2 is true
    sequence-of-statements;
  END IF;
ELSE
  -- executes when the boolean expression 1 is not true
  else-statements;
END IF;
```

6Q)Explain about Iterative Control

Ans)PL/SQL – Loops: There may be a situation when you need to execute a block of code several number of times. In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on.

1. LOOP: Basic loop structure encloses sequence of statements in between the LOOP and END LOOP statements. With each iteration, the sequence of statements is executed and then control resumes at the top of the loop.

Syntax: The syntax of a basic loop in PL/SQL programming language is

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

Here, the sequence of statement(s) may be a single statement or a block of statements. An EXIT statement or an EXIT WHEN statement is required to break the loop.

2. WHILE LOOP: A WHILE LOOP statement in PL/SQL programming language repeatedly executes a target statement as long as a given condition is true.

Syntax

```
WHILE condition LOOP
  sequence_of_statements
END LOOP;
```

3.FOR: A FOR LOOP is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

Syntax:

```
FOR counter IN initial_value .. final_value LOOP
  sequence_of_statements;
END LOOP;
```

4.Nested Loops: PL/SQL allows using one loop inside another loop. Following section shows a few examples to illustrate the concept.

The syntax for a nested basic LOOP statement in PL/SQL is as follows –

```
LOOP
  Sequence of statements1
  LOOP
    Sequence of statements2
  END LOOP;
END LOOP;
```

The syntax for a nested FOR LOOP statement in PL/SQL is as follows –

```
FOR counter1 IN initial_value1 .. final_value1 LOOP
  sequence_of_statements1
  FOR counter2 IN initial_value2 .. final_value2 LOOP
    sequence_of_statements2
  END LOOP;
END LOOP;
```

The syntax for a nested WHILE LOOP statement in Pascal is as follows –

```
WHILE condition1 LOOP
  sequence_of_statements1
  WHILE condition2 LOOP
    sequence_of_statements2
  END LOOP;
END LOOP;
```


5. Labeling a PL/SQL Loop: PL/SQL loops can be labelled. The label should be enclosed by double angle brackets (<< and >>) and appear at the beginning of the LOOP statement. The label name can also appear at the end of the LOOP statement. You may use the label in the EXIT statement to exit from the loop.

7Q) Explain about Cursors& Steps to create a Cursors

Ans) A cursor is a temporary work area created in the system memory when a SQL statement is executed. A cursor contains information on a SELECT statement and the rows of data accessed by it. This temporary work area is used to store the data retrieved from the database, and manipulate this data. A cursor can hold more than one row, but can process only one row at a time. The set of rows the cursor holds is called the active set.

There are two types of cursors in PL/SQL: Implicit and Explicit.

Implicit cursors:

These are created by default when DML statements like, INSERT, UPDATE, and DELETE statements are executed. They are also created when a SELECT statement that returns just one row is executed.

Explicit cursors:

They must be created when you are executing a SELECT statement that returns more than one row. Even though the cursor stores multiple records, only one record can be processed at a time, which is called as current row. When you fetch a row the current row position moves to next row.

Both implicit and explicit cursors have the same functionality, but they differ in the way they are accessed.

To create an explicit cursor we use the following syntax inside a PL/SQL Declare section.

Syntax: CURSOR cursor_name IS select_query;

Ex: CURSOR PROD_CURSOR IS SELECT P_CODE, P_NAME FROM PRODUCT WHERE P_QOH>(SELECT AVG(P_QOH) FROM PRODUCT);

Once we declared a cursor, we can use specific PL/SQL cursor processing commands OPEN, FETCH and CLOSE anywhere

...words of the PL/SQL block.
...pening the cursor for processing.
OPEN cursor_name;
FETCH : It is used to retrieve data from the cursor and copy it to the PL/SQL variables for processing.
Syntax: FETCH cursor_name INTO variable1, variable2,….;
CLOSE: It is used to close the cursor for processing.
Syntax: CLOSE cursor_name;

Oracle provides few attributes called as cursor attributes to check the status of FETCH operation. The cursor attributes available are %FOUND, %NOTFOUND, %ROWCOUNT, and %ISOPEN.
%FOUND : It returns TRUE, if the last FETCH returned a row and FALSE if not. Syntax : Cursor_name%FOUND
%NOTFOUND : It returns TRUE, if the last FETCH did not return any row and FALSE if it did.
Syntax : Cursor_name %NOTFOUND
%ROWCOUNT : It returns the number of rows fetched.
Syntax : Cursor_name %ROWCOUNT
%ISOPEN : It returns TRUE if the cursor is open or FALSE if the cursor is closed.
Syntax : Cursor_name%ISOPEN

%FOUND, %NOTFOUND and %ROWCOUNT attributes returns an error if the cursor is not opened.

8Q)Explain about Procedure

Ans) A stored procedure is a named collection of procedural and SQL statements. Stored procedures are also stored in the database. One of the major advantages of stored procedures is that they can be used to encapsulate and represent business transactions.

There are two clear advantages to the use of stored procedures:

- Stored procedures substantiallyreduce network traffic and increase performance.
- Stored procedures help reduce code duplication by means of code isolation and code sharing.

Syntax:-

CREATE [OR REPLACE] PROCEDURE procedure_name [argument1 datatype, argument2 datatype,…..] [IS/AS]

Declaration section
BEGIN
PL/SQL statements
END;

Here argument specifies the parameters that are passed to the stored procedure.
Data type is one of the procedural SQL data types used in the RDBMS.
Declaration section is used to declare variables.
Ex:-

```
CREATE OR REPLACE PROCEDURE PROC_PRODUCT
AS BEGIN
        UPDATE PRODUCT SET
        P_DISCOUNT=P_DISCOUNT+0.50 WHERE
        P_QOH>P_MIN*2;
        DBMS_OUTPUT.PUT_LINE('Update finished');
        END;
```

To execute the stored procedure we use EXEC command.
Syntax: EXEC procedure_name[parameters_list);
Ex:- EXEC PROC_PRODUCT;

9Q) Explain about Function

Ans) A function is a named PL/SQL Block which is similar to a procedure. The major difference between a procedure and a function is, a function must always return a value, but a procedure may or may not return a value.
Syntax:

```
CREATE [OR REPLACE] FUNCTION function_name
[parameters]
RETURN return_datatype IS
BEGIN
  PL/SQL statements;
  Return return_variable;
END;
```

Here the return datatype can be any of the oracle datatype like varchar, number etc.
Ex:-

```
CREATE OR REPLACE FUNCTION employer_details_func
  RETURN VARCHAR(20) IS  emp_name VARCHAR(20);
BEGIN
     SELECT first_name INTO emp_name
     FROM emp_tbl WHERE empID = 100;
     RETURN emp_name;
END;
```

In the example we are retrieving the 'first_name' of employee with empID 100 to variable 'emp_name'.
A function can be executed in the following ways.

1) Since a function returns a value we can assign it to a variable.
employee_name := employer_details_func;

2) As a part of a SELECT statement
SELECT employer_details_func FROM dual;

3) In a PL/SQL Statements like,
dbms_output.put_line(employer_details_func);

10Q) Explain about Packages

Ans) Packages are schema objects that groups logically related PL/SQL types, variables, and subprograms. A package will have two mandatory parts –

1.Package specification
2.Package body or definition

1.Package Specification: The specification is the interface to the package. It just DECLARES the types, variables, constants, exceptions, cursors, and subprograms that can be referenced from outside the package. In other words, it contains all information about the content of the package, but excludes the code for the subprograms.

All objects placed in the specification are called public objects. Any subprogram not in the package specification but coded in the package body is called a private object.

The following code snippet shows a package specification having a single procedure. You can have many global variables defined and multiple procedures or functions inside a package.

```
CREATE PACKAGE cust_sal AS
  PROCEDURE find_sal(c_id customers.id%type);
END cust_sal;
/
```

When the above code is executed at the SQL prompt, it produces the following result –

Package created.

2.Package Body: The package body has the codes for various methods declared in the package specification and other private declarations, which are hidden from the code outside the package.

The CREATE PACKAGE BODY Statement is used for creating the package body. The following code snippet shows the package body declaration for the cust_sal package created above. I assumed that we already have CUSTOMERS table created in our database as mentioned in the PL/SQL - Variables chapter.

```
CREATE OR REPLACE PACKAGE BODY
cust_sal AS
 PROCEDURE                    find_sal(c_id
customers.id%TYPE) IS
  c_sal customers.salary%TYPE;
  BEGIN
    SELECT salary INTO c_sal
    FROM customers
    WHERE id = c_id;
    dbms_output.put_line('Salary: '|| c_sal);
  END find_sal;
END cust_sal;
/
```

When the above code is executed at the SQL prompt, it produces the following result –

Package body created.

Using the Package Elements: The package elements (variables, procedures or functions) are accessed with the following syntax –

package_name.element_name;

Q11) Explain about Exceptions Handling

Ans) In this chapter, we will discuss Exceptions in PL/SQL. An exception is an error condition during a program execution. PL/SQL supports programmers to catch such conditions using EXCEPTION block in the program and an appropriate action is taken against the error condition.

There are two types of exceptions

1.System-defined exceptions

2.User-defined exceptions

1.Syntax for Exception Handling: The general syntax for exception handling is as follows. Here you can list down as many exceptions as you can handle. The default exception will be handled using WHEN others THEN –

```
DECLARE
  <declarations section>
BEGIN
  <executable command(s)>
EXCEPTION
  <exception handling goes here >
  WHEN exception1 THEN
    exception1-handling-statements
  WHEN exception2  THEN
    exception2-handling-statements
  WHEN exception3 THEN
    exception3-handling-statements
  ........
  WHEN others THEN
    exception3-handling-statements
END;
```

Exceptions are raised by the database server automatically whenever there is any internal database error, but exceptions can be raised explicitly by the programmer by using the command RAISE. Following is the simple syntax for raising an exception –

```
DECLARE
  exception_name EXCEPTION;
BEGIN
  IF condition THEN
    RAISE exception_name;
  END IF;
EXCEPTION
  WHEN exception_name THEN
  statement;
END;
```


2.User-defined Exceptions: PL/SQL allows you to define your own exceptions according to the need of your program.These must be declared and then raised explicitly, using either a RAISE statement or the procedure DBMS_STANDARD.RAISE_APPLICATION_ERROR.

The syntax for declaring an exception is –

```
DECLARE
  my-exception EXCEPTION;
```

3.Pre-defined Exceptions: PL/SQL provides many pre-defined exceptions, which are executed when any database rule is violated by a program. For example, the predefined exception NO_DATA_FOUND is raised when a SELECT INTO statement returns no rows.

Q12)Explain about Triggers in detail?

Ans) A Trigger is procedural SQL code that is automatically invoked by the RDBMS upon the occurrence of a given data manipulation event.

Triggers can be useful for:

- A trigger is invoked before or after a data row is inserted, updated, or deleted.
- A trigger is associated with a database table.
- Each database table may have one or more triggers.
- A trigger is executed as part of the transaction that triggered it.

A trigger is created by using CREATE TRIGGER command.

Syntax:

```
CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER } {INSERT [OR] | UPDATE [OR] |
DELETE}
```

ON table_name [FOR EACH ROW]
[DECLARE]
[Variable_namedata type [:=initial_value]]
BEGIN
PL/SQL statements
END;

A trigger definition contains the following parts;

CREATE [OR REPLACE] TRIGGER trigger_name: - This clause creates a trigger with the given name or overwrites an existing trigger with the same name.

{BEFORE | AFTER } - This clause indicates at what time should the trigger get fired. i.e for example: before or after updating a table.

{INSERT [OR] | UPDATE [OR] | DELETE} - This clause determines the triggering event. More than one triggering events can be used together separated by OR keyword.

[ON table_name] - This clause identifies the name of the table or view to which the trigger is associated.

[FOR EACH ROW] - This clause is used to determine whether a trigger must fire when each row gets affected.

Ex:-

```
CREATE OR REPLACE TRIGGER TRG_PRODUCT AFTER
INSERT OR UPDATE OF P_QOH ON PRODUCT
BEGIN
    UPDATE PRODUCT SET P_ORDER=1 WHERE P_QOH <=
P_MIN;
END;
```


UNIT I: Overview of Database Management System

1. What is DBMS? What are the advantages and disadvantages of DBMS?
2. Explain draw backs of File Processing System?
3. Explain DBMS approach?
4. Explain components of Database system?
5. Explain Database Architecture?
6. What is Data Independence? Explain types of data independence?

UNIT II: Entity-Relationship Model:

7. Define Data Model? Explain various data models.
8. What is E-R Model? Explain Building Blocks of an E-R Model Diagram?
9. What is Entity? Explain its types?
10. What is attribute? Explain its Types?
11. What is Relationship? Explain the Degree of relationship in ER-Model?

UNIT III: Relational Model :

12. What is a Key ? Explain Different types of keys used in Relational Model ?
13. Explain Codd's relational database Rules?
14. Explain about Relational Algebra Operations?
15. Explain Tuple and Domain Relational calculus?

UNIT IV: Structured Query Language :

16. Write short note on SQL commands?
17. Explain SQL Data types ?
18. Explain in details about DDL commands?
19. Explain in details about DML commtands ?
20. Explain WHERE clause ,HAVING clause, ORDER BY clause and GROUP clause ?
21. Explain different types of operators used in SQL ?

Question Bank

22. Explain Aggregate functions or Group functions in SQL?
23. Discuss the various types of Joins in SQL with suitable examples?
24. What are Nested Queries or Sub queries? Discuss it with relevant examples?
25. Discuss the role of SET Operators on various tables in a database?
26. What is a View? Describe with example?

UNIT V: PL/SQL:

27. Explain PL/SQL conditional control Structures.
28. Explain PL/SQL Loop control Structures with example?
29. What is a Cursor? Explain.
30. What is a package? Explain?
31. Give the difference between procedure and function?
32. What is Exception? Explain Exception handling?
33. What is Database Trigger? Discuss various types of triggers used in PL/SQL?