DATABASE MANAGEMENT SYSTEM

`

P. SREELATHA
MCA., (Ph.D)

Dr. M. SREEDEVI
Ph.D

Reg.No.U74120 MH2013 PTC 251205
Harshwardhan Publication Pvt.Ltd.
At.Post.Limbaganesh,Tq.Dist.Beed
Pin-431126 (Maharashtra) Cell:07588057695,09850203295
harshwardhanpubli@gmail.com, vidyawarta@gmail.com
All Types Educational & Reference Book Publisher & Distributors www.vidyawarta.com

DATABASE MANAGEMENT SYSTEM

© **P. SREELATHA**
Dr. M. SREEDEVI

❖ **Publisher :**
Harshwardhan Publication Pvt.Ltd.
Limbaganesh, Dist. Beed (Maharashtra)
Pin-431126, vidyawarta@gmail.com

❖ **Printed by :**
Harshwardhan Publication Pvt.Ltd.
Limbaganesh, Dist. Beed, Pin-431126
www.vidyawarta.com

❖ **Page design & Cover :**
H. P. Office (Source by Google)

❖ **Edition:** Feb. 2023

ISBN 978-93-90284-70-2

❖ **Price : 200/ –**

Content

UNIT – I

Overview of Database Management System

Chapter – 1
Introduction of Database System

(Q). Define what is Data, Information and Information Processing?

- **Basic Concepts and Definition: -**
- DATA
- INFORMATION

Data:-

Def.1: - Data referred to "Collection of the Facts & Figures".

Def. 2: - Data referred to "known facts" that could be recorded and stored on computer media.

Ex: In a student table, the data would include facts such as student name, age, address, and marks.

Def. 3: - Data are "Raw Facts". The word raw indicates that the facts have not yet been processed to reveal their meaning.

Def. 4: - "Data consist of facts, text, graphics, images, sound and video segments that have meaning in the users environment".

Information: -

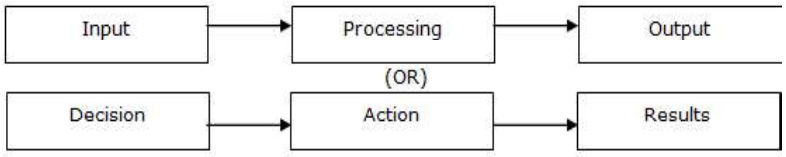
The terms data and information are closely related, and in fact are often used Interchangeably.

Def 1: - "Information is a process of meaningful of Data is known as Information".

Example: - India won the match against Australia by 100 runs.

Def 2: - "Data that have been processed in such a way as to increase the knowledge of the person who uses the data". The Information also consists as collection of images, graphics, etc.

Information Life Cycle: -



To convert data into information is to summarize them or otherwise process and present them for human interpretation.

(Q). Explain the Quality of Information? (or) Discuss Information Age?

Quality of Information is one of the most important assets. By Quality of Information, we mean information that is Accurate, Timely and Relevant are the three key attributes of information.

Today databases may contain either data or information (or both). In this "Information Age" the information must maintain the quality. By Quality Information we mean information that is

1. Accuracy
2. Timeliness
3. Relevancy

1. Accuracy: It means that the information is free from mistakes and errors. Or providing correct information. It means that information is clear and accurately reflects the meaning of data on which it is based.

2. Timeliness: It means that the recipients can get the information when they need it. The information should

maintain time frame.

3. Relevancy: Relevancy means the use of a piece of information for particular objects. It specifically for the recipient what, why, where, who and how.

(Q) Define Database and Database Management Systems?

Database: -

Def: "Database is defined as an organized collection of related data".

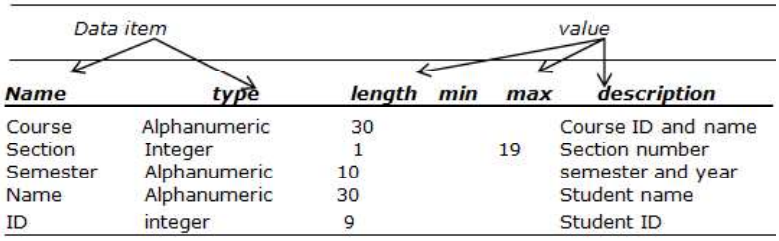
By organized we mean that the data are structured so as to be easily stored, manipulated, and retrieved by users. In the database data contain in the form of "Tables" or "Tuples" (Rows & Columns).

Metadata: -

Def. 1: "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. Metadata show the data item name, the data type, length, minimum and maximum allowable values and brief description of each data item.

Ex: some sample metadata for the Class Roster are listed in the following table.



Name	type	length	min	max	description
Course	Alphanumeric	30			Course ID and name
Section	Integer	1		19	Section number
Semester	Alphanumeric	10			semester and year
Name	Alphanumeric	30			Student name
ID	integer	9			Student ID

Metadata allow database designers and users to understand what data exist, what the data mean and what the fine distinctions are between seemingly similar data items.

Def. 2: "The database contains the data you have collected and "data about data" known as Metadata".

Database Management System: -

Def. 1: "A Database Management System (DBMS) is a collection of programs (software) that allow users to store and access the data and work a database for easy manipulation of data".

EX: dbase, oracle

Def. 2: A database management system (DBMS) is a collection of programs that manages the database structure and controls access to the data stored in the database.

(Q) Define Data Hierarchy?

Data Hierarchy refers to the systematic organization of data, often in a hierarchical form. Data Organization involves Bit, Byte, Filed, Record, File and Database.

· **BIT: -** All data is stored in the computer s memory in the form of Binary digits or Bits. A bit can be either „ON or „OFF represented „1 or „0 .

· **Byte: -** Bytes is a collection of 8 bits. One byte can represent on character.

· **Field: -** Field is a group of characters. A single filed or data filed can hold a single fact or attribute of an entity.

· **Record: -** Record is a group of related fields holding all the information about one entity.

· **File: -** File is a collection of related records.

· **Database: -** "Database is defined as an organized collection of related data".

(Q) What is Database? Define various Database Operations on Database?

Database: - "Database is defined as an organized collection of related data".

By organized we mean that the data are structured so as to be easily stored, manipulated, and retrieved by users. In the database data contain in the form of "Tables" or "Tuples" (Rows & Columns).

Database Operations: - The most commonly used operations performed on the databases are:

· **Insertion: -** To add new data into the database.

· **Selection: -** To view or retrieves the stored data.

· **Updation: -** To modify or edit the existing data.

· **Deletion: -** To remove or delete the existing data from the database.

· **Sorting: -** To arrange the data in a desired order either ascending or descending.

· **Searching: -** To find out the piece of data from the database.

(Q) List of few Database Applications?

Today, database systems are widely used to manage large bodies of information. Some of the important database applications are:

· **Banking: -** For customer information, accounts and loans and banking transactions.

· **Universities: -** For student information, course registrations, results generated.

· **Airlines: -** For reservation and schedule information. Airlines were among the first to use database in a geographically distributed manner and terminals situated around the world accessed the central database system through phone lines and other data networks.

· **Credit Card Transactions: -** For purchases on Credit cards and generation of monthly statement.

· **Sales: -** For customer, product and purchase information.

· **Telecommunication: -** For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards and storing information about the communication networks.

· **Manufacturing: -** For tracking production of items in factories, inventories of items in warehouses, stores and orders for items.

· **Human Resource: -** For information about employees, salaries and payroll taxes etc.

(Q) Discuss Manual Database Vs Computerized Database?

Manual Database	Computerized Database
Manual Database is the record keeping systems in which human being manage the whole database without the support of computers.	Computerized Database is a database that can be managed or accessed by computers.
A Manual Database is like a filling cabinet and is very slow and clumsy at organizing information.	A Computerized Database is stored in a computer and is very fast at organizing information and grouping related data.
In Manual Database system, information must be found by hand rater than electronically. Someone looking for information in a manual database may have to spend hours searching for a particular piece of data, which is a very time consuming job.	A Computerized Database will typically allow users to search the enTier database for specific information in seconds.
Less Reliable.	More Reliable.
Manual Database system has slow processing speed. It takes a lot of time to perform various database operations like: searching, sorting etc.	Computerized Database system has very fast processing speed. It can handle various database operations like: searching, sorting etc on time.
Manual Database systems can be very complexand inaccurate.	Computerized Database system are more accurate and east to process.
In Manual Database, it?s very difficult to handle a large volume of data.	Computers have huge storage capacity. So a computerized database system can easily handle a large amount of data.
Difficult to modify or update manual database.	Easily to modify and update in database.

(Q) What is Database Management System (DBMS)? Define Objectives of DBMS?

Database Management System: -

Def. 1: "A Database Management System (DBMS) is a collection of programs (software) that allow users to store and access the data and work a database for easy manipulation of data".

EX: dbase, oracle

Def. 2: A Database Management System (DBMS) is a collection of programs that manages the database structure and controls access to the data stored in the database.

Objectives of DBMS: - The major objectives of Database Management System (DBMS) are:

· **Data Availability: -** Data availability refers to the fact that the data are made available to large variety of users in a meaningful format at reasonable cost so that the users can easily access the data.

· **Data Integrity: -** Data Integrity refers to the completeness, correctness and consistency of data. It ensures that data entered into the database must be complete,

accurate, valid and consistent. Integrity is related to the quality of data, which is maintained with the help of integrity constraints. These constraints are the rules that are designed to keep data consistent and correct.

· **Data Security: -** Data Security refers to the fact that only authorized users can access the data. Data Security can be enforced by Username and Passwords. If two separate users are accessing a particular data at the same time, the DBMS must not allow them to make conflicting changes.

· **Data Independence: -** Database Management System (DBMS) supports the concepts of data independence since it represents a system for managing data separately from the programs that use the data. DBMS allows the user to store, update and retrieve data in an efficient manner.

(Q) Define various Evaluation of Database Management System (DBMS)?

File-Based Systems was the predecessor to the Database Management System (DBMS). The chronological order of development of DBMS is as follows:

1. Flat-File System (1960 -1980)
2. Hierarchical Data Model (1970 – 1990)
3. Network Data Model (1970- 1990)
4. Relational Data Model (1980 – Present)
5. Object-Oriented DBMS (OODBMS) (1990 - Present)
6. Object-Relational DBMS (ORDBMS) (1990 – Present)
7. Data Warehouses (1980 – Present)
8. Web – Enabled (1990 – Present)

· **Early 1960** Charles Bachman at GE created the first generation purpose DBMS Integrated Data Store (IDS). It created the basis for the Network Model which was standardized by CODASYL (Conference on Data System Language).

· **Late 1960** IBM developed the IMS (information Management System). IMS used an alternate model, called the Hierarchical Data Model.

· **1970 E.F.Codd.** from IBM created the Relational Data Model.

· **1976 Peter Chen** presented Entity-Relational Data Model.

· **1980** SQL developed by IBM, became the Structured / Standard Query Language for Database.

· **1980 and 1990** IBM, Oracle, Informix and other developed powerful DBMS.

(Q) Explain Classification of Database Management System (DBMS)?

The DBMS can be classified into different categories on the basics of several criteria such as the "**Data Models**, they are using **Number of Users**, they are **Number of Sites** and they are **Purpose to serve**".

1. Based on Data Models: - Depending on the Data Model they use, the DBMS can be classified as Hierarchical, Network and Relational Model.

· **Hierarchical Model: -** The Hierarchical Model DBMS organizes "The data records in a Tree Structure". Therefore hierarchical of "Parent – Child Relationships". In this model database, a parent record may have more than one child, but a child always has only one parent. This is called a "One-to-Many Relationship".

· **Network Model: -** The Network Model DBMS organizes "The data records linked to one another through pointers". Which is an association between two records? A Network Database is similar to a Hierarchical Database except that each child can have more than one parent record. This is called a "Many-to-Many Relationship".

· **Relational Model: -** The Relation Model DBMS organizes "The data records in the form of the Table and

Relationships" among the tables are set using common fields. It is simple in nature because data is simply represented in Tabular format.

2. Based on Number of Users: - Depending on the Number of users the DBMS supports it is divided into two categories namely, Single-User System and Multi-user System.

· **Single-User System: -** In Single-User System, the Database resided on one computer and is only accessed by one user at a time. The user may design, maintain and write programs for accessing and manipulation the database according to the requirements.

· **Multi-User System: -** In Multi-User System, multiple users can access the database simultaneously. In multi-user DBMS, the data is both integrated and shared.

3. Based on Number of Sites: - Depending on the number of sites over which the database is distributed, it is divided into two types namely Centralized and Distributed Database Systems.

· **Centralized Database Systems: -** The Centralized Database Systems run on a single computer system. Both the Database and DBMS software s reside at a single computer site. The user interacts with the centralized systems through a dummy terminal connected to it for information retrieval.

· **Distributed Database Systems: -** In Distributed Database Systems, the database and DBMS software s are distributed over several computers located at different sites. The computers communicate with each other through various communication media such as high-speed networks.

4. Based of Purpose: - Depending on the purpose the DBMS servers, it can he classified as General and Specific purpose. It can however be designed for as airlines or railway reservation.

❑❑❑

UNIT – II

File Base Vs Database Systems

Basic File Terminology: -

Data:

Data are "Raw facts". The word raw indicates that the facts have not yet been processed to reveal their meaning.

Databases today are used to store objects such as documents, photographic images, sound, and even video segments. So the following is broadened definition

Ex: - Such as a telephone number, a birth date, a customer name, and a year-to-date (YTD) sales value.

Field:

A character or group of characters (alphabetic or numeric) that has specific meaning. A field is used to define and store data.

Record:

A logically connected set of one or more fields that describes a person, place, or thing.

File:

A collection of related records. For example, a file might contain data about vendors of ROBCOR

Company, or a file might contain the records for the students currently enrolled at S.V. University.

Each file in the system used its own application programs to store, retrieve, and modify data. And each file was owned by the individual or the department that commissioned its creation.

(Q) Defied Drawbacks of File-Based System? (or) Explain Demerits Traditional Database System? (or) Explain the File Processing System (FPS) (or)

Discuss Problems / Disadvantages / Demerits of File Processing System (FPS)?

Early day s commercial data processing system involves considerable amount of work compression of File Processing Systems (FPS). In the conventional file handling system the data is stored in the form of records and files. Each file is a collection of records and records is a group of data fields which are logically related.

Demerits / Disadvantages of File Processing Systems (FPS):

- There are listed below of demerits of File Processing Systems (FPS):

1. Data Redundancy.
2. Data Inconsistency.
3. Difficult in Data Accessing.
4. Limited Data Sharing.
5. Lengthy Development Times.
6. Excessive Program Maintain.
7. No Data Security.

1. Data Redundancy (Duplicate of Data): - "Data Redundancy means Duplicated of Data" (or) "Repeated the same information in several places called Data Redundancy". The Duplicate of data is wasteful additional storage space. Unfortunately duplicate of the data and files often results is loss of data integrity (unique). The Data Redundancy leads to higher stooge and access cost. Moreover, the same information may be duplicated in several files, this duplication of data over several files is known as data redundancy.

Example: - The same data item may have different names in different files (or) the same name may be used for different data items in different files. The Address and Telephone number of a particular customer

may appear in a file that consists of Saving account records and in a file that consists of Checking account records. This redundancy leads to higher storage & excess cost also leads to inconsistency.

2. Data Inconsistency: Data Inconsistency means different copies of the same data are not matching. That means different versions of same basic data are existing. This occurs as the result of update operations that are not updating the same data stored at different places.

3. Difficult in Data Accessing: - It is not easy to retrieve information using a conventional file processing system. Convenient and efficient information retrieval is almost impossible using conventional file processing system**.**

4. Limited Data Sharing: - Each application has its own private files with little opportunity to share data outside their own applications. A requested report may require data from several incompatible files in separate systems.

5. Lengthy Development Times: - With the File Processing System (FPS), there is little opportunity to leverage previous development efforts. Each new application requires the developer to start from design by designing new file formats and descriptions. The length development times required are often inconsistency with today s fast business environment.

6. Excessive Program Maintain: - The preceding factors create a heavy program maintenance load. Organizations that rely on traditional file processing system. In fact as much as 80% of the total information system development budget may be devoted to program maintenances in such organization.

7. No Data Security: - It is one of the major disadvantages of File Processing Systems (FPS). In FPS we can t give the security to the data, that means the data can be accessed several users.

(Q) Define DBMS Approach?

In order to remove all the limitation of a File-Based Systems, a new approach was required that must be more effective. So the concept of database was introduced. This approach is know as Database Approach / DBMS Approach.

Characteristics of DBMS Approach: -

· It is central repository of shared data. It allows several users to access the database concurrently.

· A fundamental feature of the database approach is that the database system does not only contain the data but also the complete definition and description of these data. These descriptions are basically details about the extent, the structure, the type and the format of all data and additionally, the relationship between the data. This kind of stored data is called **"Metadata" (Data about Data).**

· Data should be correct with respect to the real world entity that they represent.

· Data should be protected from unauthorized access.

· Data in a database exist permanently until it is not explicitly deleted.

Drawbacks of DBMS Approach: -

· Database Management Systems can be difficult to set-up and operate.

· Database Management Systems can be more expensive to purchase and operate.

· Recovery is more complex.

· Initial training required for all programmers and users.

(Q). Explain the Database approaches?

(or)

Discuss Role and Advantages / Merits of Database Approaches System?

The Database Approach number of advantages compare to File Processing System. The primary advantage is listed below:

1. Program Data Independent.

2. Minimal Data Redundancy.

3. Improved Data Consistency.

4. Improved Data Accessibility and Responsibility.

5. Improved Data Sharing.

6. Increased Productivity of Application Development / Reduced Program Maintains.

7. Improved Quality of Data.

8. Enforcement Standards.

9. Improved High Data Security.

1. Program-Data Independence: - The separation of data descriptions from the application programs that use the data is called data independence. With the database approach, data descriptions are stored in a central location called the repository. This property of database systems allows an organization s data to change and evolve (within limits) without changing the application programs that process the data.

2. Minimal Data Redundancy:- The design goal with the database approach is that previously separate data files are integrated into a single, logical structure. Each primary fact is recorded in only one place in the database.

3. Improved Data Consistency: - By eliminating (or controlling) data redundancy, we greatly reduce the opportunities for inconsistency.

Ex: If a customer address is stored only once, we cannot disagree on the stored values. Also, updating data values is greatly simplified when each value is stored in one place only. Finally, we avoid the wasted storage space.

4. Improved Data Accessibility and Responsiveness: - With a relational database, end users without programming experience can often retrieve and display data.

Example: - select * from student where student_id=999;

5. Improved Data Sharing: - A database is designed as a shared corporate resource. Authorized internal and external users are granted permission to use the database, and each user is provided one or more user views to facilitate this use.

6. Increased Productivity of Application Development : - A major advantage of the database approach is that it greatly reduces the cost and time for developing new business applications. From the above all factors, as a result program maintenance can be significantly reduced in a modern database environment.

7. Improved Data Quality: - The database approach provides a number of tools and processes to improve data quality.

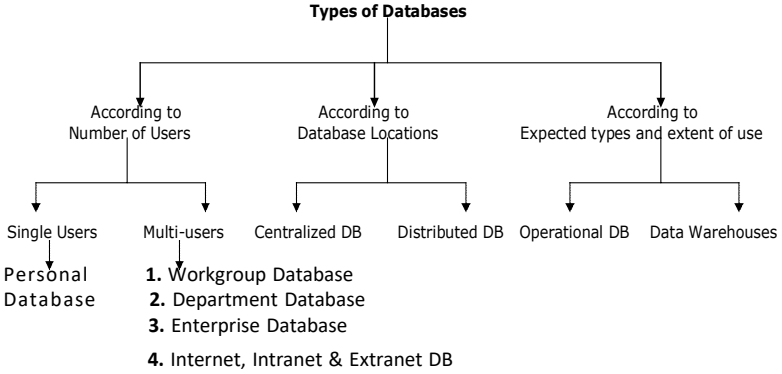
8. Enforcement Standards: - When the database approach is implemented with full management support, the database administration function should be granted single-point authority and responsibility for establishing and enforcing data standards. These standards will include naming conventions, data quality standards, and uniform procedures for accessing, updating, and protecting data.

9. Improve High Data Security: - By Using Database to provide the high data security in terms of passwords.

(Q) Explain the Range of Database Approaches?(or) Explain Types of Databases?

A DBMS can support many different types of databases.

Database can be classified according to number of users, the database locations and the expected type and extent of use.



The Range of Database Application can be divided into 5 Categories of each type there are

1. Personnel Database.
2. Work-Group Database.
3. Departmental Database.
4. Enterprise Database.
5. Internet, Intranet, Extranet Database.

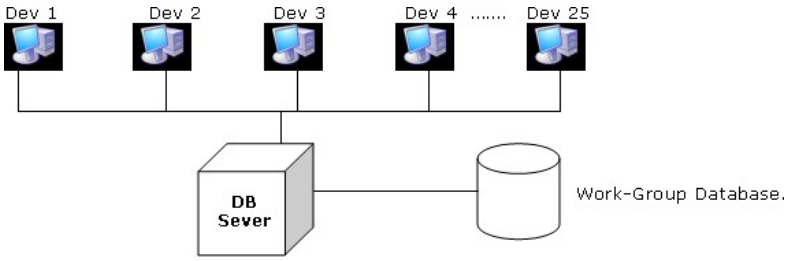
(I). ACCORDING TO NUMBER OF USERS: -

1. Personnel Database: - "The Personnel Database is designed to support one user". For example consider a personnel computer (PC) the data in PC can accessed by only one user at a time, this type of database are called Personnel Database. The Personnel Database is designed to support one user simple database application that store customer information and details of contacts. Only one user can be worked at a time with Personnel Database.

Example: - Personnel Computers Desktop or Laptops with the database.

2. Work-Group Database: - "A Work Group is a relatively Small Team of people who collaborate on the same project or application". A work group typically comprises less then 25 persons. Each member in the work group having desktop computers, each computer connected to the database servers through LAN (Local Area Network). So each member of a work

group can share the database from server.



In this above Diagram each member of the work group has a computer and the computer is linked by LAN (Local Area Network). The Database is stored and a central device called the Database Server, which is also connected to the network. Thus each members of the work group has access to the shared data different developers.

3. Departmental Database: - "An organization divided into number of functional units, each functional unit is called as a Department". For Example consider an Organization, it divided into Personnel Department, Marketing Department, Manufacturing Department and Accounting Department etc. Generally Department is the larger than the Work-Groups. The Departments consists more than 25 and less than 100 persons.

The Departmental Database is designed to support the various functions and activities of a department. The Departmental Database is accessed by only the members of the department.

4. Enterprise Database: - The term „Enterprise means collection of all departments in that organization. So the range or scope of the Enterprise Database is the enTier organization. An Enterprise Database does, how ever, support information needs from many departments. The evolution of Enterprise Database has resulted in two major developments.

I. ERP Systems (Enterprise Resource Planning).

II. Data warehouse implementation.

· **ERP System (Enterprise Resource Planning): -** It is a Business Management Database System; it integrated all functional of the enterprise, such as Manufacturing, Sales, Finance, Marketing, Inventory, and Accounting etc. ERP System are software application that provide the data necessary for the enterprise to examine and manage its activates.

· **Data warehouse:** - Data warehouse collect their content from the various operational databases, including Personnel, Work-Group & Departmental Databases.

5. Internet, Intranet, and Extranet Database: -

Internet: - The Internet is a worldwide area network (WAN) that connects users of multiple platforms through an interface known as Web Browser or Global Network any body can access the data.

Intranet: - Use of Internet protocols to "establish access to company data and information that is limited to the organization".

Extranet: - Uses the Internet protocols to "establish limited access to company data information by the company s customers & employees".

Summary of Database Applications: -

Type of Database	No. of Users	Typical architecture	Typical size of database
Personal DB	1	Desktop/Laptop computer	Megabytes
Workgroup DB	5-25	Client/server (two-tier)	Megabytes-Gigabytes
Department DB	25-100	Client/server (three-tier)	Gigabytes
Enterprise DB	>100	Client/server(Distributed or parallel server	Gigabytes- Terabytes
Internet	No limited	Client/server(Distributed or parallel server	Gigabytes- Terabytes

(II). According to Database Locations: -

1. Centralized Database: -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.

2. Distributed Database: -A database that supports data distributed across several different sites is called a distributed database.

(III). According to Expected type and Extent of use: -

1. Operational Database: - A database that s 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).

2. Data Warehouse: - A data warehouse focuses primarily on storing data used to generate information required to make tactical or strategic decisions. Such decisions typically require extensive "data massaging" (data manipulation) to extract information to formulate pricing decisions, sales forecasts, market positioning, etc. most decision support data are based on historical data obtained from operational databases.

Additionally, 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 a transaction oriented database.

(Q) Explain the File-Based Systems Vs Database System?

File-Based Systems	Database Systems
In File-Based System, each application has its own private data files.	A Database System has a collection of interrelated files and set of application programs to access and modify these files.
It is used in small systems which require new filesto represent it.	It is use in large systems which interrelated many files.
It is cheaper to design.	It is relatively expensive to design.
It has simple structure.	Its structure is relatively complex.
Data Isolated.	Data can be shared.
Recovery / Backup is simple.	Recovery / Backup is more complex.
Multiple user access to the information is difficult to provide.	Allow multiple users to access the database at same time.

(Q) Define Various Data Models in DBMS?

"A Model is a representation of reality, „Real-World Entity objects and events and their associates".

The Data model is a set of concepts that can be used to describe the data such as data type and length, relationships and consistency constraints. Data models provide necessary

data abstraction hiding the details of the data storage. That is, the main purpose of the data model is to provide a level of abstraction.

A data model is a relatively simple representation, usually graphical, of more complex real- world data structures. In general terms, a model is an abstraction of a complex real-world data environment. Within the database environment, a data model represents data structure and their characteristics, relations, constraints, and transformations.

"A data model is a blueprint of the data and its relationships used by the system. The plan created by an architect for a building is an example of a blue print".

The reasons for developing a data model are as follows:

· Models are abstractions that portray the essentials of a complex problems or structure by filtering out non-essential details, thus making the problem easier to understand.

· Models help us visualize, understand, organize, complex things thereby promoting cleaner designs and making creation easy.

The following table is traces the Evolution of the major Data Models: -

Generation	Periodof Year	Models	Examples	Comments
First	1960?s – 1980?s	File Systems	VMS / VSAM	Used mainly on IBM Mainframes Systems. Managed records, not relationships.
Second	1970?s – 1990?s	Hierarchical & Network.	IMS, ADABAS, IDS-II.	Early database systems Navigations palaces. IBM developed IMS (Information Management System).
Third	Mid-1980 –Present	Relational Data Model	DB2, Oracle, MS SQL-Server, MySQL.	E.F. Codd, From IBM created the Relational, Conceptual simplicity Entity Relationship (ER) modeling and support for relational data modeling.
Fourth	Mid 1980 –present	Object Oriented (OO) and Extended Relational	Net Objectivity / DB / DB2 / Oracle 10g.	SQL developed by IBM, became the Structures / Standard Query Language.
Fifth	1980?s – present	Data Warehousing	Data Warehousing	Support complex data extended relational products support objects and data warehousing.
Present	1990?s – present	Web-Enabled	Cartelized database	IBM, Oracle, Informix and other developed powerful DBMS.

There are basically three types of data models:-

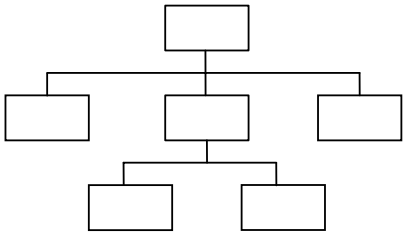
1. Record-based logical Models. (a). Hierarchical Model. (b). Network Data Model. (c). Relational Data Model.
2. Object-bases logical Models. (a). Entity-Relationship Model. (b). Object-Oriented Model.
3. Physical Models.

1. Record-based logical Model: - Record-based logical model is based on the notation of data stored as a set of records. Each record consists of a set of fields and each field is of fixed length. There are mainly three types of models namely Hierarchical Model, Network Model and Relational Model.

(a) Hierarchical Model:

The hierarchical database management system is the oldest among the database architectures. A hierarchical DBMS assumed hierarchical relationships between data, that is, parent – child relationship. In this model, data is arranged in a top-down structure that resembles an inverted tree or genealogy chart.

Data is mapped to different nodes of this tree, and they are related in nested, one-to-many relationships. Data at the top node is called the root, data in the nodes at the bottom most layer are called leaves, and data in the nodes at the intermediate levels have one parent node, and one or several child nodes. The hierarchical database management system is best applied when the conceptual data model also resembles a tree, and when most data access begins at the root. An example of Hierarchical database management system is IMS (Information Management System) DBMS.



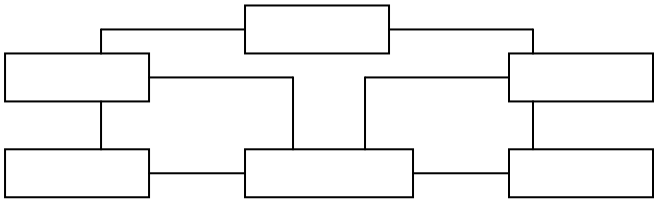
(b) Network Data Model:

The Network Model is formally defined in 1971 in the Conference on Data System Languages (CODASYL).

The network data model was created to represent complex data relations more effectively than the hierarchical model, to improve database performance, and to impose a database standard. To help establish database standards, the Conference on Data System Languages (CODASYL) created the Database Task Group (DBTG). The DBTG was defining standard specifications for an environment that would facilitate database creation and data manipulation. Finally DBTG report contains three database components.

- A schema data definition language (DDL), which enable the database administrator to define schema components.
- A DDL allows the application programs to define database components that will be used by the application.
- A DML manages the data in the database.

In the network model, a collection of records maintained network database system in M:M relationships. Network model allows a record to have more than one parent.



(c) Relational Data Model:

The Relational Data model was introduced in 1970 by E.F Codd at IBM. The Relational Database Management Systems are relatively new form historical perspective and built from theoretical consideration of the data structures by E.F.Codd s.

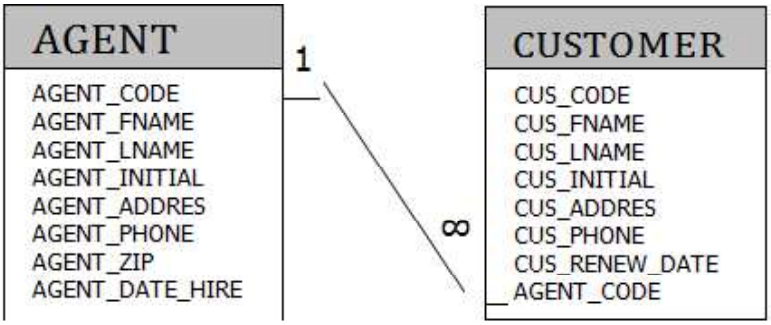
The relational model produced an automatic transmission database and it is implemented through a very

sophisticated Relational Database Management System (RDBMS), which performs the same basic functions provided by the hierarchical and network DBMS systems, in addition to a host of other functions that make the relational database model easier to understand and implement.

Examples of RDBMS are Oracle, Sybase, Informix, Microsoft SQL Server, MySql, DB2 and PostgreSQL.

The RDBMS represents data and relationships among data by a collection of table, each of which had a number of columns with unique names. The concept of tables and rows and columns is extremely simple and easy to understand.

Tables are related to each other through the sharing of a common attribute. For example, the CUSTOMER table contains a sales agent s number that is also contained in the AGENT Table.



The relational data model s rise to control is its powerful and efficient language i.e SQL. The RDBMS uses SQL to translate user queries into instructions for retrieving the requested data. Any SQL- based relational database application involves three components.

· **The end-user interface:** the interface allows the end user to interact with data.

· **A collection of tables stored in the database**: In a relational database, all data are perceived to be stored in tables.

· **SQL Engine:** SQL engine executes all queries, or data requests. The end user uses SQL to create table structures and to perform data access and table maintenance.

The most important advantage of the RDBMS is its ability to hide the complexities of the relational model from the end user and it manages all of the physical details.

A relational database provides a much higher degree of data independence than hierarchical and network databases. In relational databases, tables are not hard-linked to one another. Columns can be added to tables, tables can be added to the database and new data relationship can be added with little or no restructuring of the tables.

Main differences between Relational Model and Non-Relational Model (Network & Hierarchical Model):-

Relational Model	Non-Relational Model
Relationship defined by data content.	Relationships maintained by pointers.
Any data item is always directly addressable.	Access to data through predefined paths.
Very flexible at logical level.	A structural change very difficult, through it is not impossible.
Conceptually simple data structures.	Inherently complicated data structures.
No significances in order of rows.	Ordering of records significant.
No such restriction supports and maintained by standard set of operations.	Accessing programs require procedural and navigation capability.

2. Object-based logical Models: - In Object-based logical model database is structured in the form of objects of several types. It allows specifying data constraints separately.

These models are used in describing data at the conceptual and view levels of abstractions. Some of the most widely known data models that fall under this category are: Entity-Relationship Model and Object-Oriented Model data model.

(a) Entity-Relationship Model:

Peter Chen first introduced the ER model in 1976; it was the graphical representation of entities and their relationships in a database structure. The main constructs of the ER Model are Entities and their attributes, and the Relationships between the entities.

ER models are normally represented in an entity relationship diagram (ERD), which uses graphical representations to model database components. The ER model is based on the following components.

· **Entity:** An entity is represented in the ERD by a rectangle, also known as an rectangle box. The name of the entity written in the center of the rectangle with capital letters.

· **Attributes:** Each entity is described by a set of attributes that describes particular characteristics of the entity. Attribute in the ERD by oval.

· **Relationships:** Relationships describe associations among the data. Most relationships describe association between two entities. The ER model uses the term connectivity to label the relationship types.

(b) Object –Oriented Data Model (OODM):

Object –oriented Data model consists of objects. Each object contains attributes and methods, which operate on the attributes. Each attributes values stores in the form of variables and methods are written as a block of code. The two or more objects communicate each other using object methods. Each object is identified with a unique identifier to distinguish with other objects. The OODM is the basis for the object- oriented database management system (OODBMS).

The OO data model is based on the following components.

· An object is an abstraction of a real-world entity.

· Attributes describe the properties of an object.

· Objects that share similar characters are grouped in classes. A class is a collection of similar objects with shared structure (attributes) and behavior (methods). The class methods like finding a selected PERSON s name, changing a PERSON s name or printing a PERSON s address

· Classes are organized in a class hierarchy. The class hierarchy resembles an upside-down tree in which each class has only one parent. For example, the CUSTOMER class and

the EMPLOYEE class share a parent PERSON Class.

· Inheritance is the ability of an object within the class hierarchy to inherit the attributes and methods of the classes.

Object-oriented data models are typically using Unified Modeling Language (UML) class diagrams. This language is based on object oriented concepts that describes a set of diagrams and symbols that can be used to graphically model a system.

Let us use a simple invoice problem, in this case, invoices are generated by customers, each invoice references one or more lines, and each line represents an item purchased by customers.

3. Physical Data Model: - The Physical Data Model captures the data at the lowest level that are used for database-system implementation. These concepts are mainly meant for computer specialists, but not for the end users. Here data is represented as record formats, records orderings and access paths.

Example: - Unifying Model and Frame-Memory Model.

(Q) Explain Enterprise Database System? (or) Components / Elements of Database System?

Def: - "An Enterprise is a group of people with a common goal to achieve the signal task and enterprise is act as a signal entity". (Or) "The Enterprise is considered as systems and all the departments are its subsystems".

The term database system refers to an organization of components that define and regulate the collection, storage, management, and use of data within a database environment. From a general management point of view, the database system is composed of the five major parts shown in figure below.

1. Hardware
2. Softwares
3. People
4. Procedures
5. Data

1. Hardware:

Hardware refers to all of the system's physical devices; for example, computers (microcomputers, mainframes, workstations, and servers), storage devices (hard disk, floppy disk), network devices (hubs, switches, routers, fiber optics), and other devices it also includes the Input/Output devices (keyboard, mouse, scanner, printer), processors, main memory etc. which are used for storing and retrieving the data in a faster and efficient manner.

2. Software:

The software part consists of DBMS which acts as a bridge between the user and the database. 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 managers all hardware components and makes it possible for all 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 Access and SQL Server, Oracle Corporation s Oracle, 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.

3. People:

The peoples/ users are who manage the database and perform different operations. On the basis of primary job functions, five types of users can be identified in a database system:

1. Systems administrators,
2. Database administrators
3. Database designers,
4. Systems analysts and programmers,
5. End users.

- Systems 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. If the database design is poor, even the best application programmers and the most dedicated DBA s cannot produce a useful database environment. Because organizations strive to optimize their data resources. The database designer s job description has expended to cover new dimensions and growing responsibilities.

- 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. For example, salesclerks, supervisors, managers, and directors are all classified as end users.

4. Procedures:

Procedures are the instructions and rules that govern the design and use of the database system. Procedures play an important role in a company because they enforce the standards by which business is conducted within the organization and with customers.

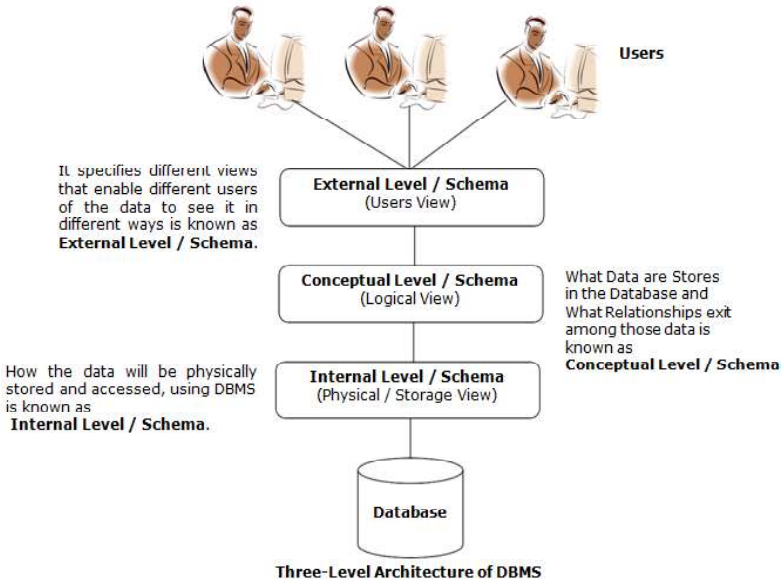
5. Data:

The word data covers the collection of facts stored in the database.

(Q) Explain DBMS Architecture? (or) Three-Level Architecture of DBMS? (or) Types of Schema?

The major purpose of database system is to provide users with an abstract view of the system. The system hides certain complex details from the users like "how data is stored and shows only those database records to the user which he/she demands". This is called as **"Data Abstraction"**.

The term Abstraction means the amount of detail you want to hide. In the same manner, the database can also be viewed from different levels of abstraction to reveal different levels of details. From bottom-up manner, we may find that there are Three-Levels of abstraction in the Database.



Three-Level Architecture of DBMS

1. External Level: - (Users View only)

· In this database view, maximum detail about the database will be hidden from the user.

· At External-Level, only the restricted portion of the database is available to the end users, because an end user does not need to know everything about the structure of the

enTier database, rather than the amount of details he/she needs to works with.

· It implements "Highest Level of Data Abstraction".

2. Conceptual Level: - (Logical View)

· This view will provide some more details about the database to the users like Structure or Schema detail of the database.

· Conceptual Level describes "What Data are Stores in the Database and What Relationships exit among those data".

· It implements "Middle Level of Data Abstraction".

3. Internal Level: - (Physical / Storage View)

· It can provide the Internal Level view of "The Actual Physical Storage of the Data".

· It can also describe the data type of these data items, the memory size of the items in the storage media.

· It defines the "Location of the Data" (physical address) of the items in the storage device.

· It implements "Lowest Level of Data Abstraction".

Logical Data Independence Vs Physical Data Independence in DBMS Architecture: -

Logical Data Independence	Physical Data Independence
It is concerned with the Structure of the Data.	It is concerned with Storage of the Data.
It is concerned with the Conceptual Schema.	It is concerned with the Internal Schema.
It is very difficult as the retrieving of data is very much dependent on the Logical Structure.	It is easy to retrieve.

(Q) Explain Types / Classified of Database Architecture?

Database Architecture essentially the location of all the pieces of information that make up the database application. The Database Architecture can be broadly classified into following types:

1. Single-Tier Architecture.

2. Two-Tier Architecture.

3. Three-Tier Architecture.

4. N-Tier Architecture.

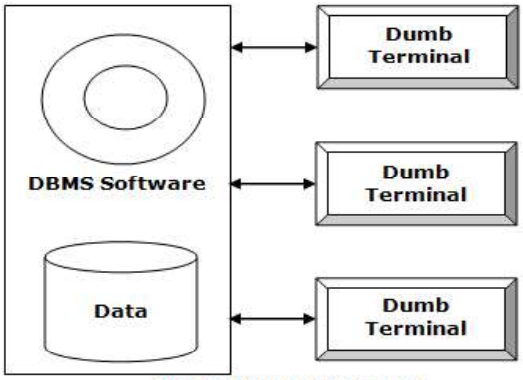
(I). Single-Tier Architecture: -

· Single-Tier Architecture consists of a Single Computer that contains a Database and a Front-End access the Database.

· Single-Tier Architecture is equivalent of running an application on a Personal Computer (PC).

· All required components to run the application are located within it.

· User Interface, business logic and data storage are all located on the same machine.



Single-Tier Architecture.

Advantages: -

· A Single-Tier System requires only one stand-alone computer.

· It also requires only one installation of proprietary software.

Disadvantages: -

· May be used by only one user at a time.

· A Single-Tier system impractical for an organization which requires two or more users to not interact with the organizational data store at the same time.

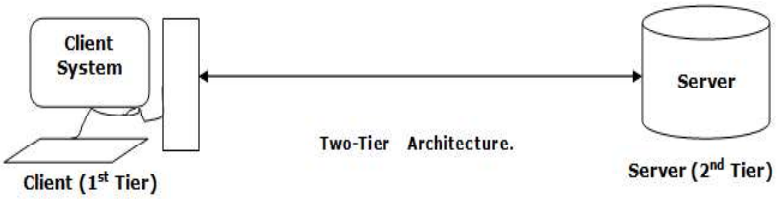
(II). Two-Tier Architecture: -

· Two-Tier Architecture consists of two systems, a Client and Server.

· The database is stored on the Server, and the interface used to access the database is installed on the Client.

· The Client being the First Tier and the Server being the Second Tier.

· The Client requests services directly from Server. Therefore Client communicates directly with the Server without help of other tools.



Advantages: -

· Easy to implements on both sites.

· Tools are relatively inexpensive.

· The Two-Tier Client and Sever provides much more attractive Graphical User Interface (GUI) applications that were possible with earlier technology.

· Two-Tier Architecture maintains a persistent connection between the Client and Server/ Database.

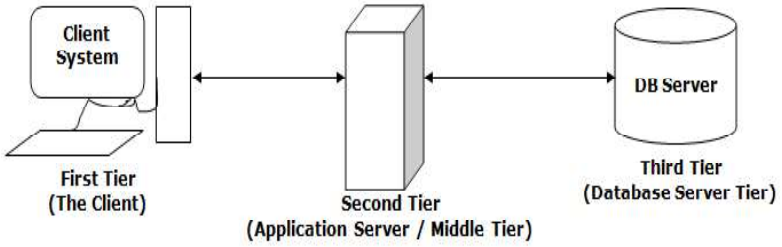
· Offers a great deal of flexibility and simplicity in management.

Disadvantages: -

· Since actual data processing of data takes places on the remote / server side. The data has to be transported over the networks. This leads to the increased network stress.

(III). Three-Tier Architecture: -

In Three-Tier Architecture, the Client requests are handled by intermediate servers which coordinate the execution of the Client request with subordinate Servers. It adds Middleware (Middle Tier), which provides a way for clients of DBMS to access data from another DBMS.



Three-Tier Architecture.

1. First-Tier (The Client Tier): - In this tier Client send the request for the information. The main responsibility of this tier is to receive user events and to control the user interface and presentation of data. As most of the software is removed from the Client, the Client is called "Thin Client".

2. Second-Tier (Application Server / Middle Tier): - The complex application logic is loaded her and available to the Client Tier on request from Client. This tier can protect direct access of data.

3. Third-Tier (Database Server Tier): - The Third tier contains the data that is needed for the application therefore this tier is responsible for the data storage. This server mostly operates on a relational database.

Advantages: -

· It improved Scalability and Integrity.

· The Middle Tier can ensure that only valid data is allowed to be updated in the database.

· It provides high security. The Client does not have direct access to the database.

· Load balancing is easier with the separation of the core business logic from the database server.

· The need for less expensive hardware because the Client is „Thin†.

· The added modularity makes it easier to modify or replace one tier without affecting the other tier.

Disadvantage: -

· More complex than Two-Tier Client-Server computing model.

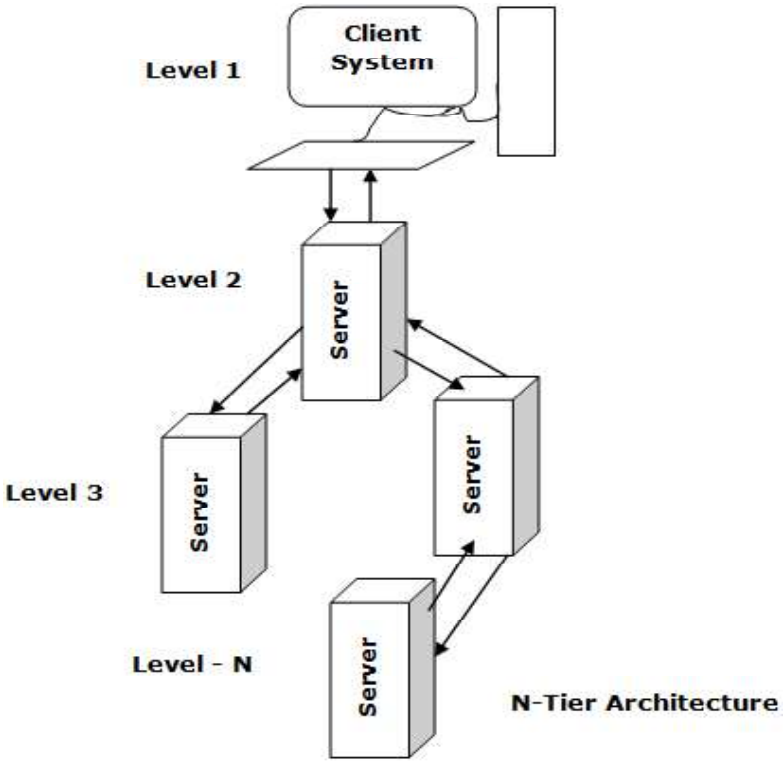
· The Client does not maintain a persistent database connection.

· A separate proxy server may be required.

· Network traffic will be increase, if a separate proxy server is used.

(IV). N-Tier Architecture: -

In Three-Tier Architecture, each Server (Application Server and Database Server) perform a specialized task or service. A server can therefore use services from other servers in order to provide its own service. The Three-Tier Architecture is potentially an N-Tier Architecture.



Advantages: -

· Overall performance has been improved.
· The business logic is centralized.
· Enhanced security level is obtained.
· Provides many services to many Clients.
· Enhanced scalability and availability.

Disadvantage: -

· Much more complicated to design than Two-Tier and Three-Tier models.
· Load balancing is harder.
· More difficult to maintain software.
· The overall cost of the software and hardware is usually high.

(Q) Define DBMS Vendors and their Products?

Some of the popular DBMS Vendors and their corresponding products are listed following:

Vendor	Product(s)
Microsoft	Access Excel SQL Server
Oracle	Oracle DBMS MySQL
IBM	DB2 Informix Dynamic Server (IDS)
Sybase	Adaptive Server Enterprise (ASE) Adaptive Server Anywhere (ASA) Watcom

❑❑❑

UNIT – III

Entity-Relationship (ER) Model

(Q) Explain Data Model Basic Building Blocks?

The basic building blocks of all data models are Entities, Attributes, Relationships and Constraints.

Entity: - An entity is a person, place, object events, in the user environment about which the organization to maintain of data.

Example: -

Place : Rompicherla, Piler, Tirupati, Hyderabad etc.

People : Students, Employees, Workers etc.

Objects : Automobiles, Machine, Building etc.

Events : Sales, Registration, Renewal etc.

Attribute: - An attribute is a property or characteristics of an Entity Type.

Example: -

STUDENT : Student_Id, Student_Name, Course.

EMPLOYEE : Emp_Id, Emp_Name, Payroll, Skills.

Relationship: - "A relationship type is a meaningful association between (or among) entity types".

Constraints: - constraints are a condition or restriction placed on the data.

(Q) Introduction of Entity Relationship Models (E-R Models)?

An Entity Relationship Model is a detailed, logical representation of the data for an organization or for a business area.

The Entity Relationships Models is expressed in teams of Entities, the relationships among those entities and the attributes of both the entities and their relationships.

An Entity Relationship Model is normally expressed as an entity relations diagrams, which is a graphical representations of Entity Relationships Models.

Definitions: -

1. Entity: - An entity is a person, place, object events, in the user environment about which the organization to maintain of data.

Example: -

Place : Rompicherla, Piler, Chittoor, Hyderabad etc.
People : Students, Employees, Workers etc.
Objects : Automobiles, Machine, Building etc.
Events : Sales, Registration, Renewal etc.

2. Entity Type: - A Collection of entities that shared a common property or characteristics.

3. Entity Instance: - A single occurrence of the entity type is called as Entity Instances.

Example: -

Entity Type: EMPLOYEE Attributes:
Emp_Name Varchar (20)
Emp_Number Varchar (10)
DOB Date

Two Instances of EMPLOYEE

(1)	Emp_Name: ABCD	**(2)** Emp_Name: WXYZ
	Emp_Number : 101010	Emp_Number : 202020
	DOB : 19-08-1992	DOB : 20-09-1990

Rules: -

1. If a naming attribute we can use initial capital letter followed by lower case letters.
2. If an attribute name consists two words we can use underscore (_) character to connected two words.
3. In naming entity we can use only user case.

4. Attribute: - An attribute is a property or characteristics of an Entity Type.

Example: -

STUDENT : Student_Id, Student_Name, Course.

EMPLOYEE : Emp_Id, Emp_Name, Payroll, Skills.

(Q). What are the Components of an E-R Model?

The overall logical structure of a database can be expressed graphically by an E-R Diagrams. Such a diagram consists of major components listed below.

· Ellipses : It represents Attributes.

· Rectangles : It represents Entity Sets.

· Lines : It represents links attributes to entity sets of relations.

· Diamonds : It represents Relationships.

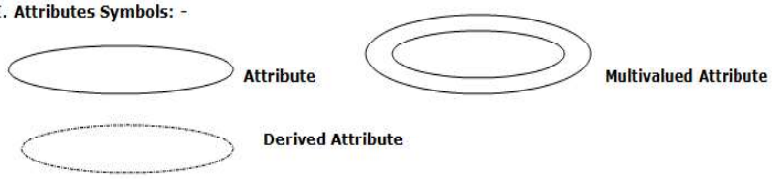
· Double Ellipse : It represents Multivalued attributed.

· Dashed Ellipse : It represents Derived attributes.

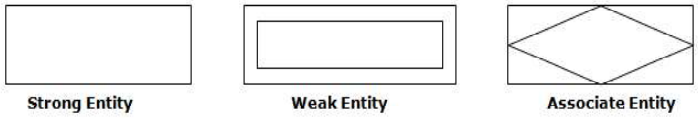
· Double Lines : It indicates total participation of an entity in a relationship set.

(Q). What are the Entity Relationship Modeling Symbols?

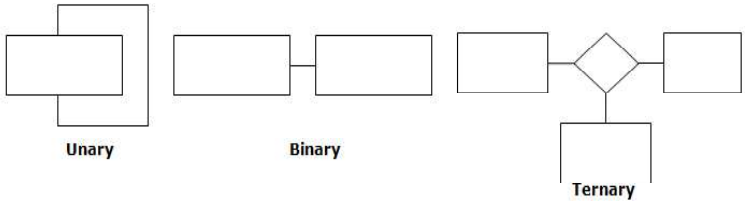
I. Attributes Symbols: -



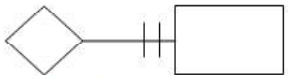
II. Entity Symbols: -



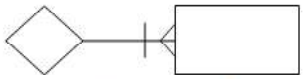
III. Relationship Degree: -



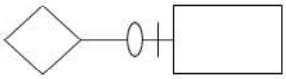
IV. Relational Cardinality: -



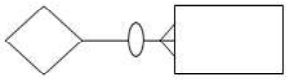
Mandatory One



Mandatory Many



Optional One



Optional Many

(Q) What is Attribute? Explain the types of Attributes in E – R Model?

Attribute: - An Attribute is a property or characteristics of an Entity Type.

Example: -

STUDENT : Student_Id, Student_Name, Course.

EMPLOYEE : Emp_Id, Emp_Name, Payroll, Skills.

Ternary

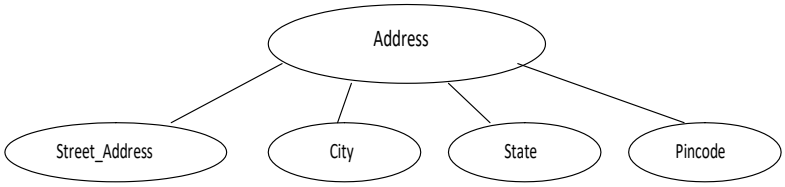
Types of Attributes: - The following types of attributes are used in an E – R Model.

1. Simple Attribute.
2. Composite Attribute.
3. Multivalued Attribute.
4. Derived Attribute.

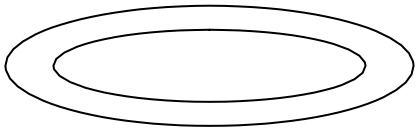
1. Simple Attribute: - A Simple or Atomic attribute is an attribute "That can t be broken down into smaller components". The Simple Attribute is represented as an Ellipse symbol.

2. Composite Attribute: - An Attribute "That can be broken down into some small parts"

Example: - The most common example is „Address , which can be usually broken down into following components like Street_Address, City, Sate and Pincode.



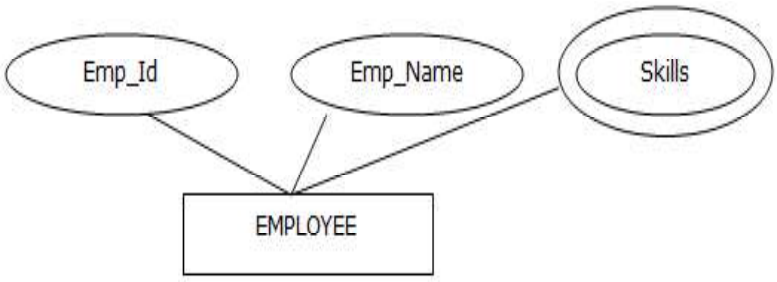
3. Multivalued Attribute: - An attribute "that may take on more than one values for a given entity instance". We indicate a multivalued attribute with an ellipse with double lines. The following symbol is represented Multivalued attribute.



Example: - The Employee entity type having the following attributes

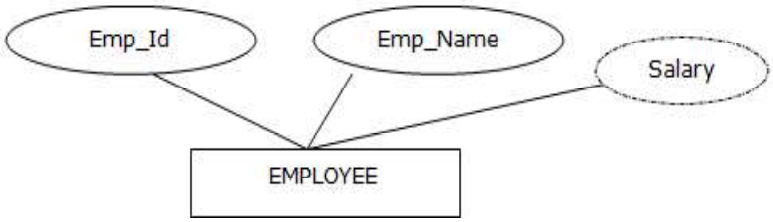
EMPLOYEE: Emp_Id, Emp_Name, Skill.

In above three attributes the last attribute named as „Skill . That may take on more than one value. Of course some employee may have more than one skills like (c, Java, VB, BDBMS, etc).



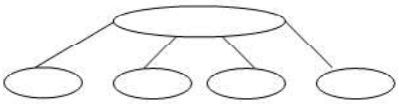
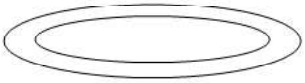
4. Derived Attribute: - A Derived Attributer is an attribute "Whose values be calculated" from related attribute values. We indicate a derived attribute in an E – R Model by using an "Ellipse with a dashed line".

Example: - To calculate the Salary an Employee.
EMPLOYEE: Emp_Id, Emp_Name, Salary.



In this above E – R Model „Salary is a derived attribute it can calculated like Hourly based, Daily based or Monthly base.

Attributes and types symbols used in E – R Model.

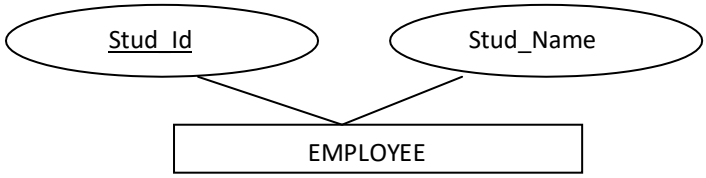
S. No	Attribute Symbols in E-R Model	Description
1		Simple Attribute
2		Composite Attribute
3		Multivalued Attribute.
4		Derived Attribute

Null Attribute: - A Null Value is used when an entity does not have a value for an attribute. Null can also designate that an attribute value is unknown. An unknown value may be either missing or not known. Such types of attribute are called as Null Attribute.

Identifier Attribute: - "An Identifier is an attribute (or combination of attribute) that uniquely identifies individual instances of an entity type".

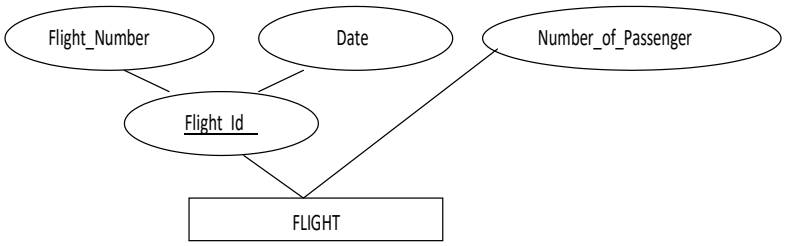
Example: - STUDENT entity type introduced as Stud_Id, Stud_Name. Here „Stud_Id is Identifier attribute, because Stud_Name is not a candidate identifier since many students

have the same name. We underline identifier names on the E – R Model shown in figure.

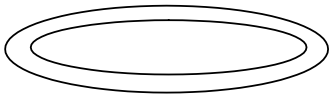


Composite Identifier: - "An Identifier that is consists of a composite attribute".

Example:- Consider as a FLIGHT entity that attributes are Flight_Id, Flight_Number, Date, Number_of_Passengers. Here candidate identifier is „Flight_Id and also uniquely identify the Flight_Number that second identifiers is called as a Composite Identifiers. Here the primary candidate identifier (Flight_Id) is must be underline and second identifier (Flight_Number) no need to indicate an underlines.



Multivalued Attribute: - An attribute "that may take on more than one values for a given entity instance". We indicate a Multivalued attribute with an ellipse with double lines. The following symbol is represented Multivalued attribute.



Special Guidelines of Naming & Defining Attribute: -

In addition to the general guidelines for naming data object. There are few special guidelines for naming attributes which are

· An attribute name is a noun (such as Customer_Id, Age, Production, Price).

· An attribute name should be Unique. No two attributes of the same entity type may have the same name.

· Each attribute name should be follow a standard formats.

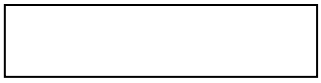
(Q) Explain the Entity, Entity Type, Entity Instances and Types of Entity s in E – R Model?

1. Entity: - An entity is a person, place, object events, in the user environment about which the organization to maintain of data.

Example: -

Place : Rompicherla, Piler, Chittoor, Hyderabad etc.
People : Students, Employees, workers etc.
Objects : Automobiles, Machine, Building etc.
Events : Sales, Registration, Renewal etc.

2. Entity Type: - "A Collection of entities that shared a common property or characteristics". An Entity type is always singular we can use capital letters for the entity name. In E – R Model the Entity is representing as a following symbol.



3. Entity Instance: - "A single occurrence of the entity type is called as Entity Instances."

Example: -

Entity Type: EMPLOYEE Attributes:
Emp_Name Varchar(20)
Emp_Number Varchar(10)
DOB Date

Two Instances of EMPLOYEE

(1) Emp_Name : ABCD
Emp_Number : 101010
DOB : 19-08-1992

(2) Emp_Name : WXYZ
Emp_Number : 202020
DOB : 20-09-1990

Types of an Entity Types in E – R Model: -

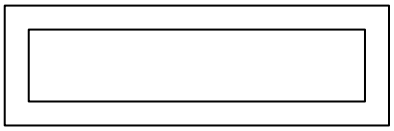
The following various types of entities can used in E-R diagram.

1. Weak Entity.

2. Strong Entity.

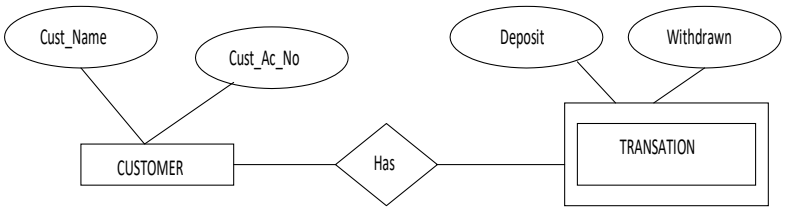
3. Associate Entity.

1. Weak Entity: - "An entity type whose existence depends on some other entity" is known as Weak Entity. A Weak Entity type has no business meaning in the E-R Diagram. The Weak Entity is representing as a double rectangular box.



Example: -

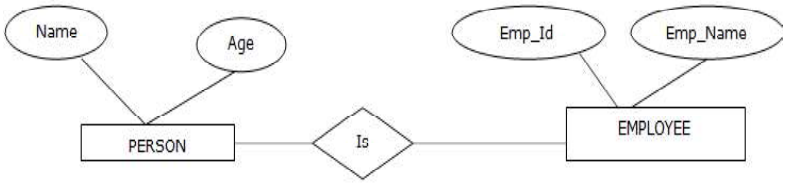
Consider the following example. Here „COUSTOMER is a entity it having two attributes like Cust_Name, Cust_Ac_No and „TRANSATION is a another entity it having a two attributes like Deposit, Withdrawn.



2. Strong Entity: - "An Entity that exists independently of other entity type" is known as a Strong Entity. It means it does not dependent another entity. The Strong Entity is representing as a solid rectangular box.

Example: -

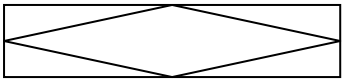
A PERSON is an Entity it s having two attribute like Name, Age. Another entity is EMPLOYEE it s having two attributes like Emp_Id, Emp_Name.



In this above example the entity EMPLOYEE is a Strong Entity. It does not dependent on any other entity.

3. Associative Entity: - "An Associative Entity is an entity type that associates the instances of one or more entity types and contains attributes that are peculiar to the relationship between those entity instances".

The associative entity is represented with the diamond relationship symbol enclosed with in the entity box.



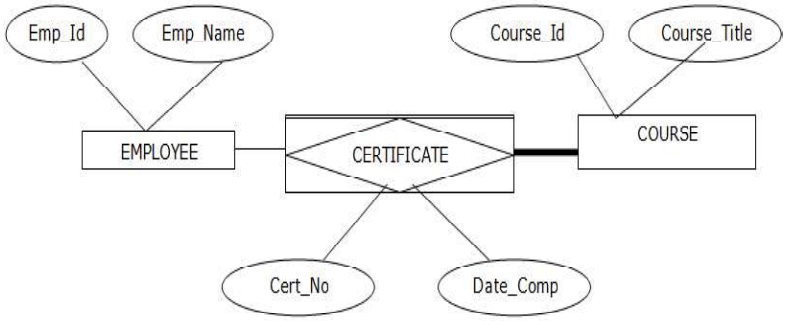
Example: -

An EMPLOYEE entity is having attributes Emp_Id, Emp_Name. A COURSE having the attributes is Course_Id, Course_Title. Another entity as CERTIFICATE having the attributes like Cert_No, Date_Comp. Now the relations between 3 entities by using associative entity are as following.

EMPLOYEE : Emp_Id, Emp_Name.

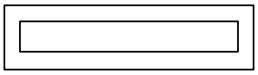
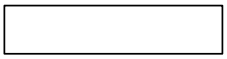
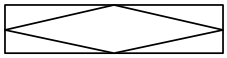
CERTIFICATE : Cert_No, Date_Comp

COURSE : Course_Id, Course_Title.



In this above example the entity CERTIFICATE is an associative entity. Because we can use this entity as entity type of relationship.

Entity and types of symbols used in E – R Model: -

S. No	Entity Symbols in E-R Model	Description
1		Weak Entity
2		Strong Entity
3		Associative Entity

Special Guidelines to Naming & Defining Entity: -

In addition to the general guidelines for naming data object. There are few special guidelines for naming entity which are

· An Entity name is a singular noun (Such as EMPLOYEE, CUSTOMER, FLIGHT)

· An Entity type name should be specific to the organization.

· An Entity type name should be concise (concise means giving a lot of information with a few words). So entity name is a minimum as possible as meanings.

· An abbreviation or short name should be specified for each entity type name.

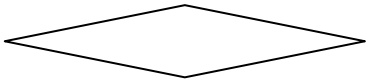
Example: - Emp_Dept instead of Employee_Department.

· The entity name should be small be same on all E-R Diagram which the entity type appears.

(Q) Explain the Relationship s and explain the Degree of Relationships?

Relationship: - "A relationship type is a meaningful association between (or among) entity types". A relationship

type is denoted by a "diamond symbol" contain the name of the relationship.



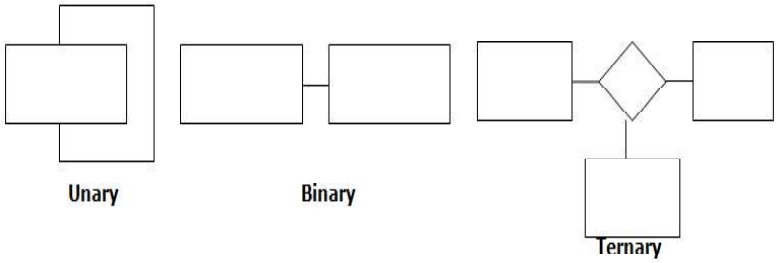
Relationship Instances: - "An association between (or among) entity instances where each relationship instances includes exactly one entity from each participating entity types".

Associative Entity: - "An entity type that associates the instances of the one or more entity types and contains attributes that are peculiar to the relationship between those entity instances".

Degrees of Relationships: - "The number of entity types that participate in a relationship". The three mostly common relationships of degrees in E-R Model there are

1. Unary Relationship (Degree 1)
2. Binary Relationship (Degree 2)
3. Ternary Relationship (Degree 3)

Symbols of Relationship Degrees: -

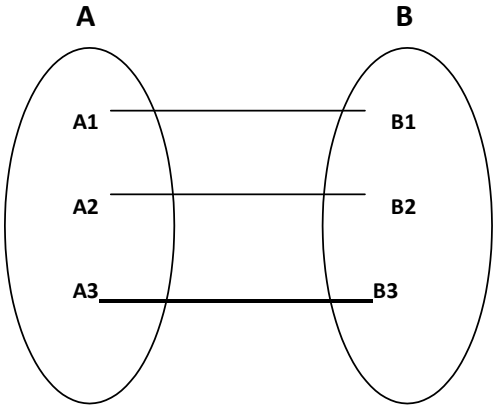


1. Unary Relationship: - "A Relationship between the instances of a single entity type" is known as a Unary Relationship. Unary relationship also called as a „Recursive relationships.

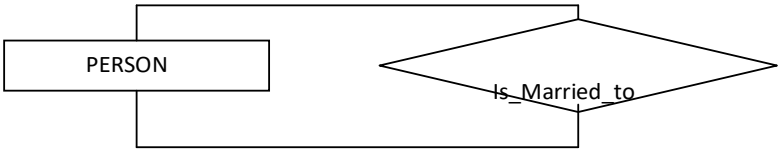
Under this Unary Relationship we have

1. One – to – One.
2. One – to – Many.

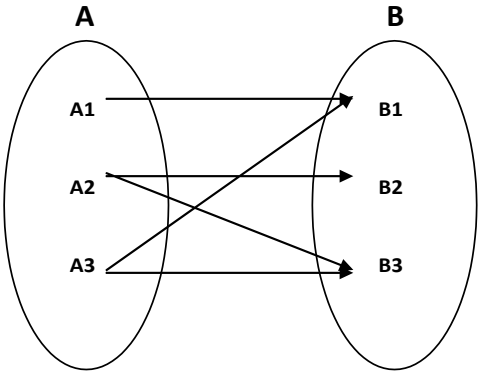
· **One – to – One: -** An entity is „A is associated with at most one entity in „B , and an entity is associated with at most one entity in „A .



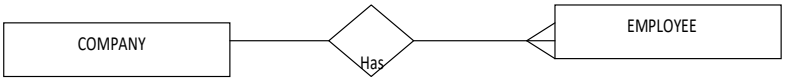
Example (1):- Consider as a E-R Diagram it having a one entity and one relationship it s their one-to- one.



· **One – to – Many: -** An entity is „A is associated with any number of entity in „B , and entity in „B how ever can be associated with at most one entity in „A .



Example (1):- Consider as a E-R Diagram it having a one entity and many relationship it s their One-to-many.



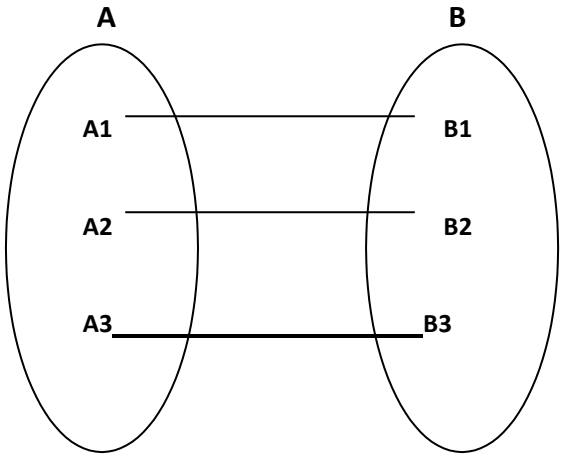
In this above example each Company has one or many employees. But each employee is employed by only one Company. This type of relationship is called as One-to-Many relationships.

2. Binary Relationship: - "A relationship between the instances of two entity types". Is known as a Binary Relationship.

Under this Binary Relationship we have

1. One – to – One.
2. One – to – Many.
3. Many – to – One.
4. Many – to – Many.

· **One – to – One: -** An entity is „A is associated with at most one entity in „B , and an entity is associated with at most one entity in „A .



Example (1):- Consider as E-R Diagram it having a one entity and one relationship it s their one-to-one.

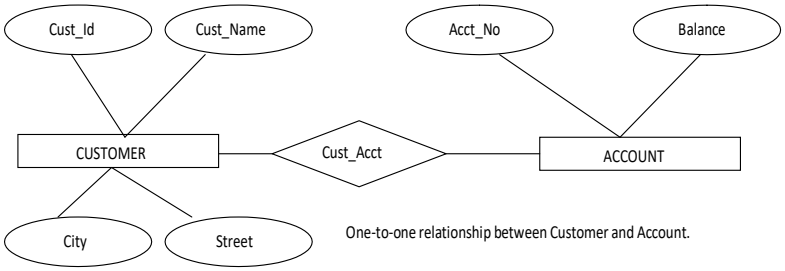
In this above example every Passenger most process a Ticket, and each Ticket must bee belong to only one Passenger.

Example (2): - Consider as a following entity having the attributes

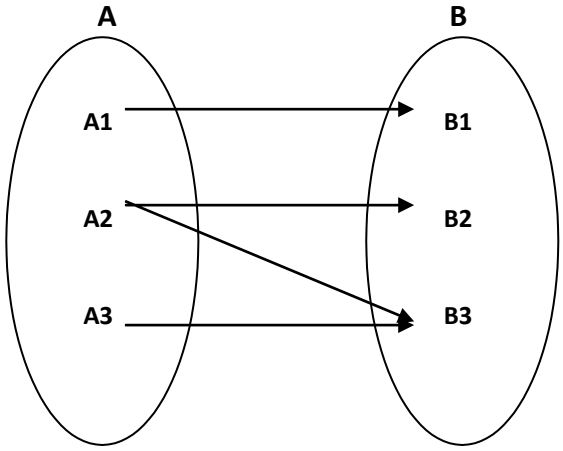
CUSTOMER : Cust_Id, Cust_Name, City, Street.

ACCOUNT : Acct_No, Balance.

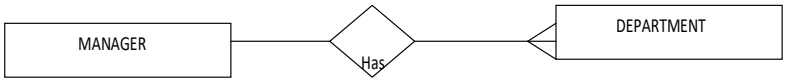
A relationship set customer account (Cust_Acct) may be one-to-one relationship. A customer having only one account Number in a bank.



· **One – to – Many: -** An entity is „A is associated with any number of entity in „B , and entity in „B how ever can be associated with at most one entity in „A .



Example (1):- Consider as a E-R Diagram it having a one entity and many relationship it s their One-to-many.



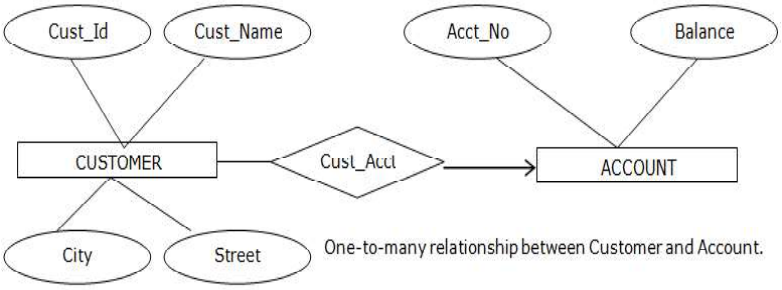
In above example one Manager has responsible for many Department responsibility of their company.

Example (2): - Consider as a following entity having the attributes

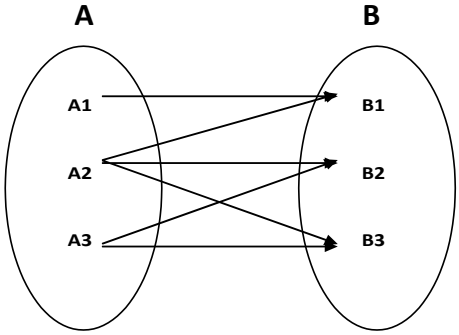
CUSTOMER : Cust_Id, Cust_Name, City, Street.

ACCOUNT : Acct_No, Balance.

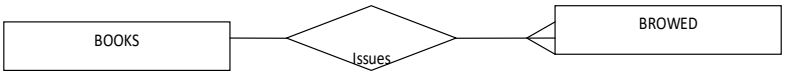
A relationship set customer account (Cust_Acct) may be one-to-many relationship. A customer having several accounts types in one bank.



· **Many – to – One: -** An entity is „A is associated with at most one entity in „B and entity in „B how ever can be associated with any number of entities in „A .



Example (1):- Consider as a E-R Diagram it having a many entity and one relationship it s their Many-to-one.



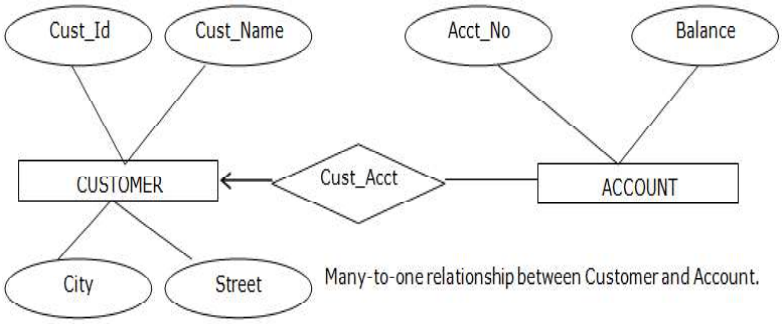
In above example Books is a Entity and Browed is a another entity. One student can issued number of books in library.

Example (2): - Consider as a following entity having the attributes

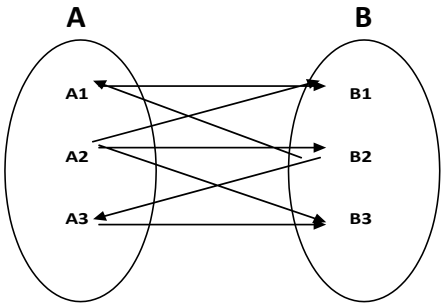
CUSTOMER : Cust_Id, Cust_Name, City, Street.

ACCOUNT : Acct_No, Balance.

A relationship set customer account (Cust_Acct) may be many-to-one relationship. Many customers having only one accounts types in one bank. (Joint Account)



· **Many – to – Many: -** An entity is „A is associated with any number of entity in „B , and entity in „B how ever can be associated with any number of entities in „A



Example (1):- Consider as a E-R Diagram it having a many entity and many relationship it s their Many-to-many

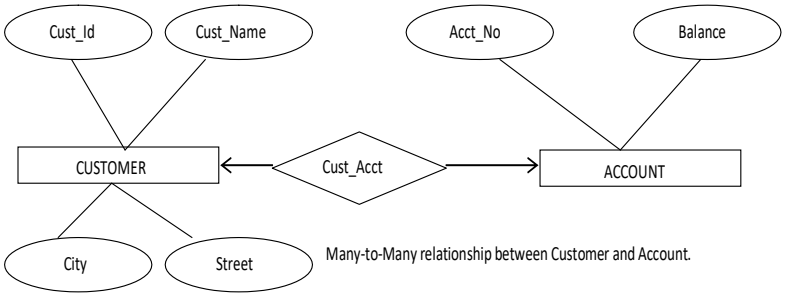
In this above example each Lecture to teach one or more subjects. One subject can be dealing or lecturing one or more lectures. In this method we called as a Many-to-Many relationship.

Example (2): - Consider as a following entity having the attributes

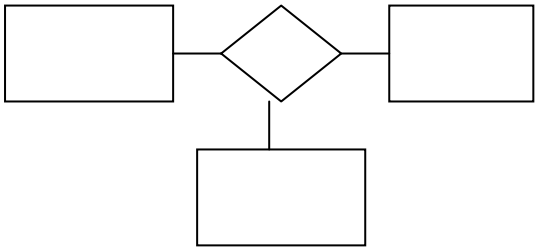
CUSTOMER : Cust_Id, Cust_Name, City, Street.

ACCOUNT : Acct_No, Balance.

A relationship set customer account (Cust_Acct) may be many-to-many relationship. In a bank many customers having many accounts.



3. Ternary Relationship: - "A Simultaneous relationship among the instances of three entity types" is known as Ternary Relationship. A business situation that leads to a ternary relationship is shown below.

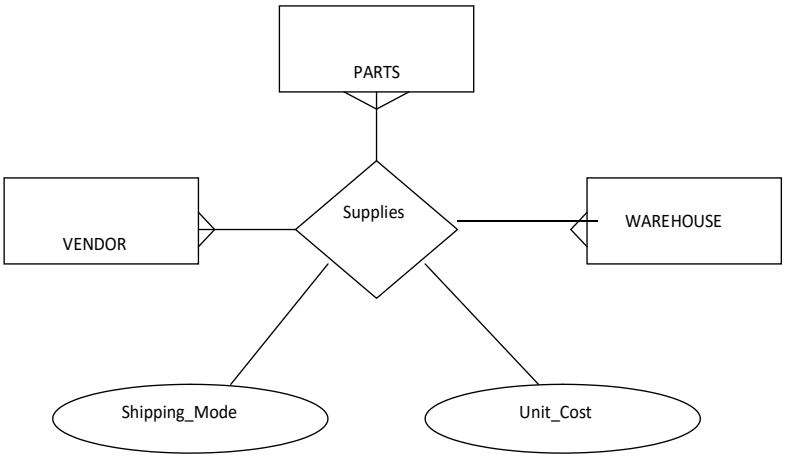


Example: - Consider a following entity set having the attributes.

Entities : VENDOR, PARTS, WAREHOUSE.

Attributes : Shipping_Mode, Unit_Cost.

Relationship : Supplies.



In this example VENDOR can Supplies various PARTS to WAREHOUSE. The relationship Supplies is used to record the specific Parts that are supplied by a given Vendor to a particular Warehouse.

❑❑❑

CHAPTER-II

Extended Entity Relationship Modeling (EE-R Model)

Introduction: -

In order to design the complete structure of any organization, ER-Modeling symbols are not sufficient. So new features are added to ER-Model to provide more accurate structure to the ER-Model. And E-R Model renamed as an "Enhanced / Extended Entity Relationship Modeling (EE-R Model)".

There are new features is like an

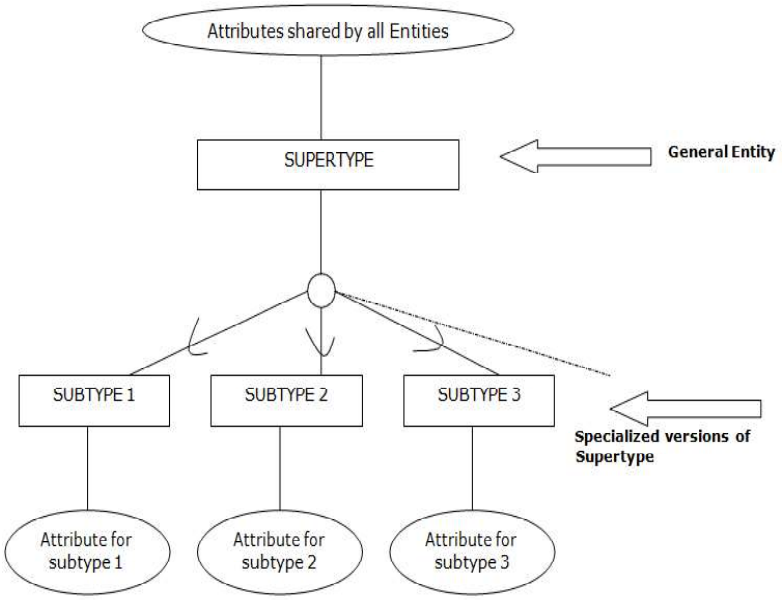
1. Supertype.
2. Subtype.
3. Generalization.
4. Specialization.
5. Attribute Inheritances.
6. Aggregation.

(Q) How to Representing Supertype / Subtype? Give one example?

Definition of Supertype: - "A generic entity type that has a relationship with one or more subtype".

Definition of Subtype: - "A Sub-grouping of the entities in an entity type that is meaningful to the organization and that share common attributes or relationships distinct from other sub-groupings".

Basic Concepts and Notation of Supertype / Subtype: -



Note: -

1. The Supertype is connected with a line to a ◯ Circle in which in term is connected by a line to each Subtypes that have been define.
2. The ∪ (U – Shape) is indicated connecting a Subtype.
3. Here all attributes are shared by all entity are associated with Supertype.
4. Attributes are unique to a particular subtype, associated with subtype.

Example: - The following example shows a representation of the EMPLOYEE Supertype with its three subtypes using EER-Notations. Consider the following entities set having the attributes.

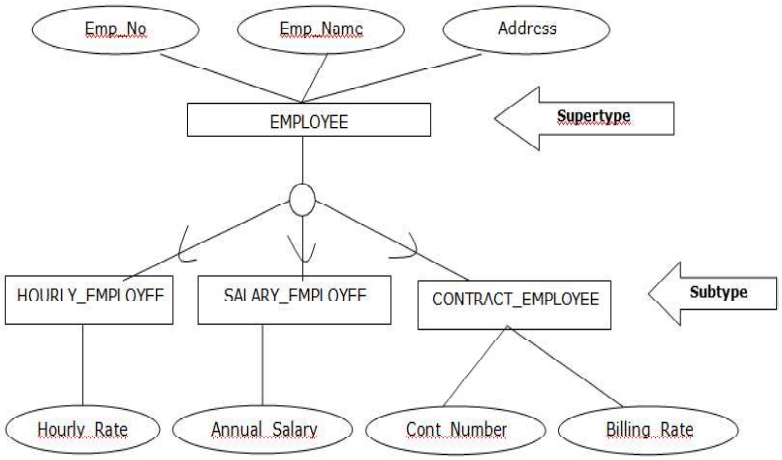
Supertype: EMPLOYEE : Emp_No, Emp_Name, Address.

Subtype:HOURLY_EMPLOYEE : Emp_No, Emp_Name, Address, Hourly_Rate.

SALARY_EMPLOYEE : Emp_No, Emp_Name, Address, Annual_Salary.

CONTRACT_EMPLOYEE : Emp_No, Emp_Name, Address, Cont_Number, Billing_Rate.

In that above all of the EMPLOYEE types several attributes in common namely Emp_No, Emp_Name, Address. In addition each type has one are more attribute distinct from the attributes of the types.



In the above figure shows a representation of the EMPLOYEE Supertype with the 3 subtypes using EER- Notation. Attributes shared by all EMPLOYEE are associated with the EMPLOYEE entity type. Attributes that are peculiar to each subtype are included with that subtype only.

Specialization Hierarchy: - Entity Supertype and Subtype are organized in a specialization hierarchy, which depicts the argument of higher-level entity Supertype (parent entities) and lower-level entity subtype (child entity).

Specialization Hierarchy formed by an EMPLOYEE Supertype and there three entity subtypes PILOT, MECHANIC and ACCOUNTANT. The specialization hierarchy reflects the 1:1 relationship between EMPLOYEE and its subtype.

The relationships depicted within the specialization hierarchy are sometimes described in terms of "is-a" relationships. For example a Pilot is an Employee, a Mechanic is an Employee and an Accountant is an Employee. However a specialization hierarchy can have many levels of supertype / subtypes relationship, that is, you can have a specialization hierarchy in which a Supertype has many subtypes, in turn one of the subtype is the Supertype to other lower-level subtypes.

A Specialization Hierarchy provides the means to:

· Support Attributes Inheritances.

· Define a special Supertype attribute know as the subtype discrimination.

· Define disjoint / overlapping constraints and complete / partial constraints.

Supertype: EMPLOYEE : Emp_Id, Emp_Name, Address.

Subtype: PILOT : Emp_Id, Emp_Name, Address, License, Ratings.

MECHANIC : Emp_Id, Emp_Name, Address, Type.

ACCOUNTANT : Emp_Id, Emp_Name, Address, Category.

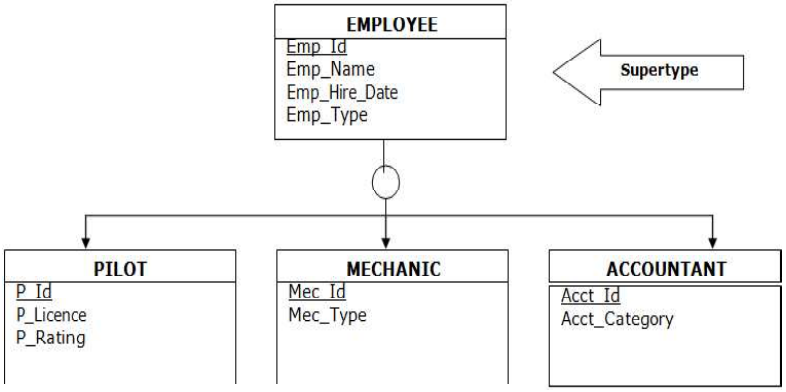
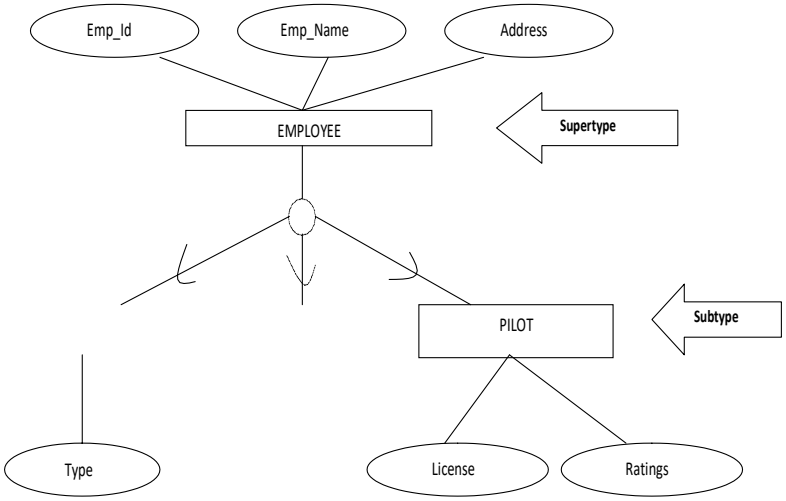


Fig: - Specialization Hierarchy.

In that above all the EMPLOYEE types several attributes in common namely Emp_Id, Emp_Name, Address.



In the above figure shows a representation of the EMPLOYEE Supertype with the 3 subtypes using EER- Notation. Attributes shared by all EMPLOYEE are associated with the EMPLOYEE entity type. Attributes that are peculiar to each subtype are included with that subtype only.

(Q) Representing Generalization and Specialization with examples?

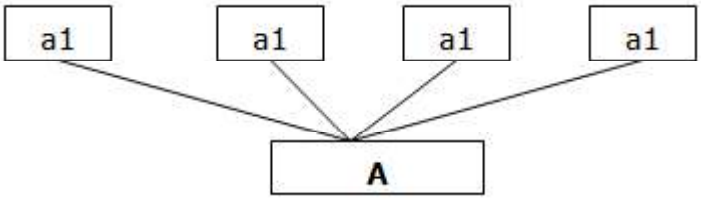
Introduction: - There are two processes namely Generalization and Specialization. There are serve as mental model in developing Supertype and subtype relationships.

Generalization: - "The process of defining amore general entity type from a set of more specialized entity type".

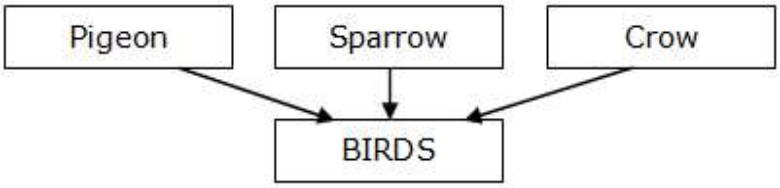
(OR)

"Generalization is the process in which subtype are grouped into Supertype".

This generalization is a follows "Bottom-Up" (Deriving supertype from subtype) approaches. Generalization and Specialization are interdependent on each other and available in on E-R Model. But it is not possible to distinguish Generalization and Specialization separately.



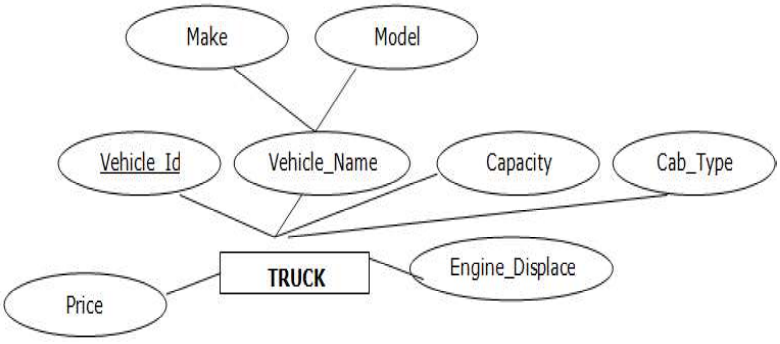
Example: - Pigeon, Sparrow and Crow can all be generalized as „Birds .



Example: - Consider the three entity types having the attributes is as follows.

· **TRUCK** : Vehicle_Id, Vehicle_Name, Capacity, Price, Cab_Type, Make, Model, Engine_Displace.
· **CAR** : Vehicle_Id, Vehicle_Name, No_of_Passengers, Price, Engine_Displace, Make, Model.
· **MOTOR CYCLE** : Vehicle _Id, Vehicle_Name, Price, Make, Model, Engine_Displace.

In this stage, the data modeler intends to represent these separately on E-R diagrams.



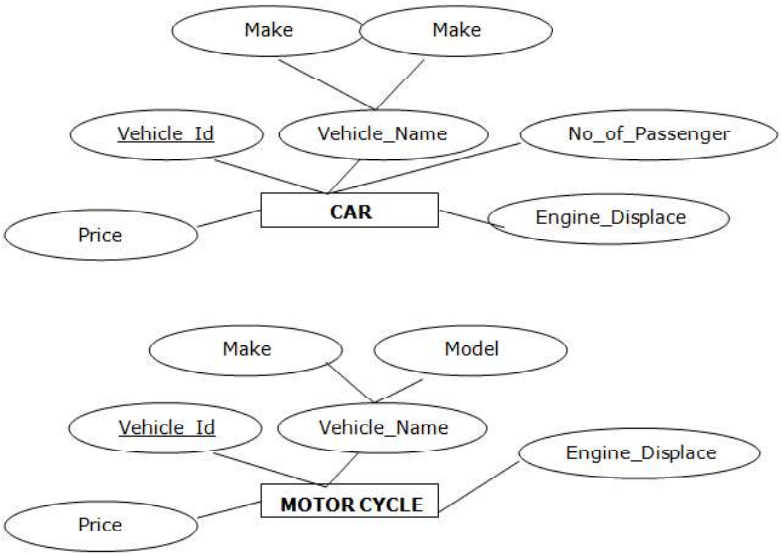


Fig: - Three entity types TRUCK, CAR, MOTOR CYCLE

Here Vehicle_Id is the Identifier of attribute and Vehicle_Name is the Composite attribute like Make & Model.

This more general entity type (named as VEHICLE) together with the resulting Supertype and subtype relationship. Generalization to „VEHICAL . Supertype.

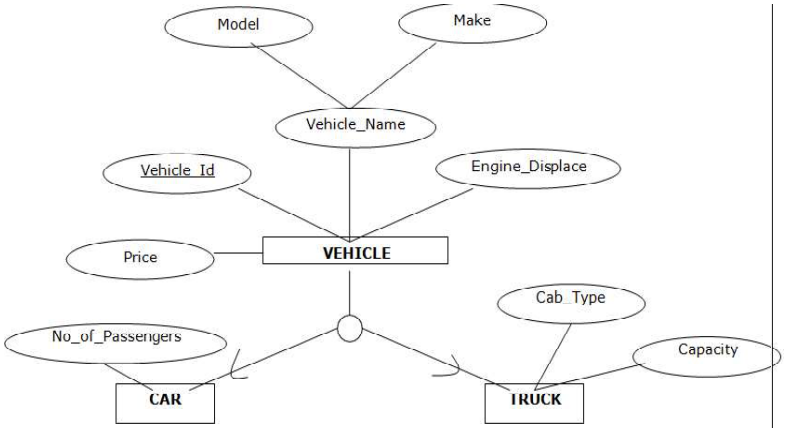


Fig: - Generalization to VEHICAL Supertype

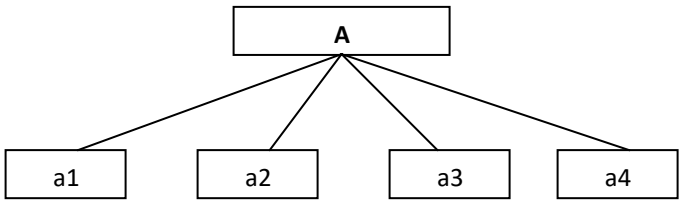
The entity CAR has the specify attribute No_of_Passengers, while TRCUK has two specify attributes like Capacity, Cab_Type. Thus generalization allowed us to group entity types along with their common attributes and at the same time preserves specific attributes that are peculiar to each subtype.

"Notices that the entity type MOTOR CYCLE is not includes in the relationship. Because it doesn t satisfied the conditions for a subtype". We will notice that only attributes of MOTOR CYCLE are those that are common to all „VEHICAL . There are no specifying attributes are MOTER CYCLE.

Specialization: - "The process of defining one or more subtypes of the Supertype / subtype and forming Supertype / subtype relationships".

(OR)

"Specialization is a process in which Supertype can be de-composed into smaller subtypes**". Supertype is located at the top position and it is divided into further subtypes. These subtypes are located under the Supertype. So Specialization follows "Top-Down" (Deriving subtype from Supertype) approach.**

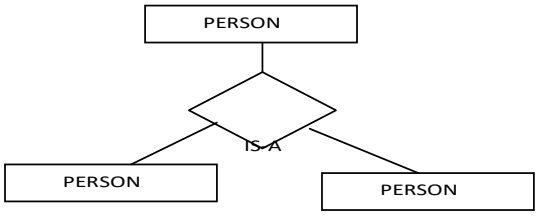


Specialization is the top-down process of identifying lower-level, more specific entity subtypes from higher-level entity Supertype. Specialization is based on grouping unique characteristics and relationships of the subtypes.

Example: - Take a group „Person . A person has name, date of birth, gender, etc. these properties are common in all persons, human beings. But in a company, persons can be

identified as Employee, Employer, Customer or Vender based on what role they play in the company. Similarly, in a college database, persons can be specialized as Lecturer, Student or a Staff based on what role they play in college as entities.

It is to be observed that the „Student and „Lecturer inherit some of the attributes of the„Person .



Example: - Suppose an entity type named as „PART together with several of its attributes. The identifier is Part_No and other attributes included Description, Unite_Price, Location, Qty_on_Hands, Routing_Number and Supplier (the last attributes is multivalued since there may be more than one supplier with associated Unit_Price for a part).

PART : Part_No, Description, Qty_on_Hands, Location, Routing_Number(product Mgf Number), Supplier, Unit_Price, Supplier_Id.

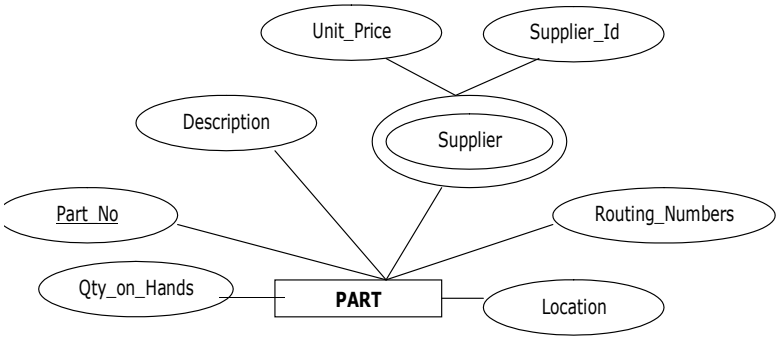


Fig: - Specialization of Entity type PART

There are two possible sources for PART: Some are „Manufactured internally, while others are „Purchased from outside suppliers. Some are attributes are apply to the all

parts, regardless of source. Thus Routing_Number applies only to „Manufacture part, while Supplier_Id and Unit_Price apply only to „Purchased part. These factors suggest that PART should be specialized by defining the subtypes MANUFACTURE PART and PURCHASED PART.

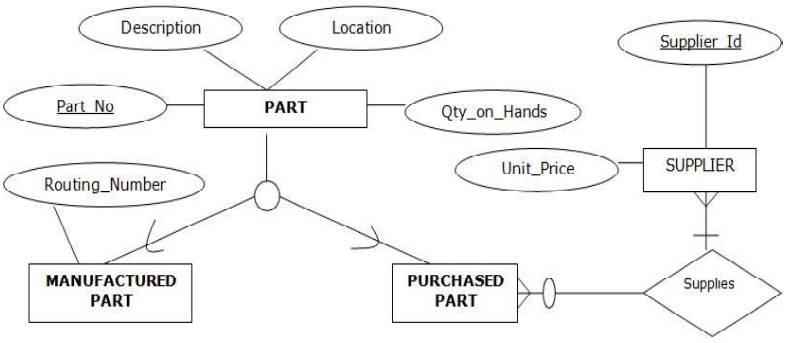
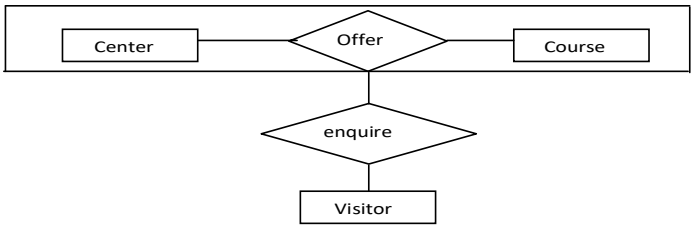


Fig: - Specialization to MANUFACTURED PART and PURCHASED PART

(Q) Explain Aggregation and Composition? With an example?

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

Example: The relation between Center and Course entities is acting as a single entity in relation with entity Visitor.



Composition: - It implies a relationship between the entities where the Child entity cannot exit independent of the Parent entity.

Example: - House (parent entity) and Room (child entity). Room does not exit separate to a House.

CHAPTER-III

Relational Data Modeling

RELATIONAL DATA MODEL: -

Dr. E. F. Codd First introduces the relational data model in 1970.

"The relational data model represents data in the form of tables."

The relational model is based on mathematical theory and therefore has a solid theoretical foundation.

The relational data model consists of the following three components

1. Data Structure: Data are organized in the form of tables with rows and columns.

2. Data manipulation: Powerful operations (using the SQL language) are used to manipulate data stores in the relations.

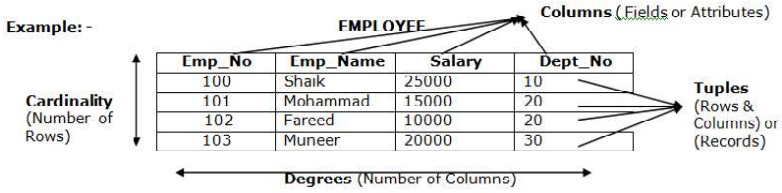
3. Data Integrity: Facilities are included to specify business rules that maintain the integrity of data when they are manipulated.

RDBMS Terminology: -

Formal Relational term	Informal equivalent
Relation	Table
Tuple	Row, Record
Attribute	Column, Field
Cardinality	No. of Rows
Degree	No. of columns
Domain	Set of legal values
Primary key	Unique identification

(Q) What is a Relation? Explain the different representation methods of relations with example?

Relation: - "A relation is a table in DMBS a relation consider as set of rows and columns". Every relation must consist named columns and unnamed rows.



Properties of Relationship: -

1. Every ration contains named columns and unnamed rows.

2. Table attributes must be unique names.

3. Table names must be unique.

4. The values in the cells should be „atomic .

Representation of a relation: -

Normally a relation may be represented in two possible methods.

1. Textual Representation.

2. Graphical Representation.

1. Textual Representation: - In this representation total table structure like name of the table and its attributes etc., are expressed in text format. In textual representation the name of the table followed by attributes with in parenthesis.

Syntax: - Table name (Attribute name 1, Attribute name 2, Attribute name n)

Example: - EMPLOYEE (Emp_Id, Emp_name, Salary, Dept_No).

2. Graphical Representation: - In this representation name of the table takes place on the top position and attributes placed with in connected rectangles.

Syntax: -

Table Name

Attr 1	Attr 2	Attr 3	.....	Attr n.

Example: -

EMPLOYEE

Emp_Id	Emp_Name	Salary	Dept_No

(Q). Explain the Characteristics / Properties of Relational Data Model? Properties of Relations or Characteristics of Relational Table: -

We have defined relations as two-dimensional tables of data. However, not all tables are relations. Relations have several properties that distinguish them from non-relational tables. We summarize these properties below.

1. A table is perceived as a two-dimensional structure composed of rows and columns.

2. Each relation (or table) in a database has a unique name.

3. An entry at the intersection of each row and column is automatic (single valued). There can be no multi-valued attributes in a relation.

4. Each row is unique; no two rows in a relation are identical.

5. Each attribute (or column) within a table has a unique name.

6. The sequence of columns (left or right) is insignificant. The columns of a relation can be interchanged without changing the meaning or use of the relation.

7. The sequence of rows (top to bottom) is insignificant. As with column, the rows of a relation may be interchanged or stored in any sequence.

8. All values in a column must conform to the same data format. For example, if the attribute is assigned an integer data format, all values in the column representing that attribute must be integers.

(Q) Explain the Relational Keys?

We must able to store and retrieve of a rows in a relation database on the values that in a row, to achieve this goal every relation must have some types of keys. Those keys are called as relational keys. The relational keys are following.

- ✓ Super Key.
- ✓ Candidate Key.
- ✓ Primary Key.
- ✓ Composite Key.
- ✓ Foreign Key.

1. Super Key: - An attribute (or Combination of attributes) that uniquely identifies each row in a table.

2. Candidate Key: - A minimal super key that is one that does not contain a subset of attributes that is itself a super key. (OR) **A candidate key can be described as a super key without redundancies, that is, a minimal super key**.

Example: Using this distinction, note that the composite key.

STU_NUM, STU_LNAME

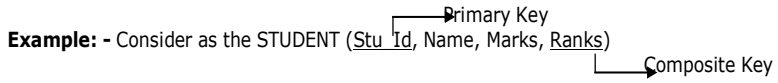
is a super key, but it is not candidate key because STU_NUM by itself is a candidate key! The combination STU_LNAME, STU_FNAME, STU_INT, STU_PHONE might also be a candidate key, as long as you discount the possibility that two students share the same last name, first name, initial, and phone number.

3. Primary Key: - "A Primary key is an attribute (or Combination of attributes) that Uniquely Identifies each row in a relation". We designate a primary key by using Underlining the attribute name,

Example: - The primary key for the relation EMPLOYEE is Emp_Id. In shorthand notation express this relation as following.

EMPLOYEE (Emp_Id, Name, Dept_Name, Salary).

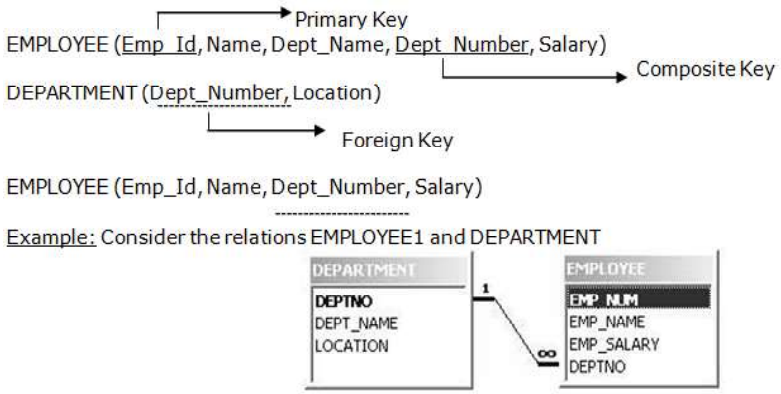
4. Composite Key: - "A Composite key is a primary key that consists of more than one Primary key or attribute" is called as a Composite key.



5. Foreign Key: - The Foreign key must represented the relationship between two tables. This is accomplished or provide through using Foreign key.

"A Foreign Key is an attribute in a relational database that several primary keys of another relation in the same database. The attribute name is „Dept_Number is a Foreign Key. It can represent in a notations by using "Dashed Underline" of the attribute.

Example: - It consider as a two different tables in a database like EMPLOYEE & DEPARTMENT



EMPLOYEE1 (**Emp_ID**, Emp_Name, Emp_salary, Deptno)
DEPARTMENT (**Deptno**,Dep_Name, Location)

This attribute Deptno is a foreign key in EMPLOYEE1. It allows a user to associate any employee with the department to which he or she is assigned. The foreign key represented in a notation by using a "Dashed Underline".

Properties of Relations: -

We have defining relations as two-dimensional tables of data. However not all tables are relations. Relations have several properties that distinguish them from non-relation tables. We summarized these properties below.

✓ Each relation (or table) in a database has a unique name.

✓ An entity intersection of each row and column is an automatic value (atomic).

✓ There can be no multivalued attributes in a relation.

✓ Each row is unique; no two rows in a relation are identical.

✓ Each attribute within a table has a unique name.

(Q) Explain the Integrity Constraints (or) Rules?

The Integrity Constraints the relational data model includes several types of constraints (or) business rules, the purpose is to facilitate maintaining the accuracy and integrity of data in the database.

The major types of integrity constraints are

1. Domain Constraints.

2. Entity Integrity.

3. Referential Integrity.

4. Action Assertion.

1. Domain Constraints: - "All of the values that appear in a column of a relation must be taken from the same domain".

A domain usually consists of the following components like Domain Name, Meaning, Data type, size (or length), and allowable values or allowable ranges. The following table shows the domain definition

Attribute	Domain Name	Descriptions	Domain Data type & Size
Cust_Id	Customer_Ids	Set of all possible Customers Id?s	Characters : size 5
Cust_Name	Customer Name	Set of all possible Customers Names	Characters : size 20
Date	Data	Set of all possible dates	Characters : mm-dd-yy
Product_Price	Product Prices	Set of all possible prices of product	Characters : Size 6 digits
Product Finish	Product Finishes	Set of all possible finishes of product	Characters : Size 12

2. Entity Integrity: - The Entity Integrity rule is designed to assure that "Every relation has a primary key" and that the data values for that primary key are all valid. In particulars, it guarantees that every "primary key attributes is not a null". The entity integrity rule states the following "No Primary key attribute may be Null".

3. Referential Integrity: - "In the relational data model associations between tables are defined through the use of Foreign Key".

For Example: the association between the CUSTOMER and ORDER tables is defined by including the Cust_Id attribute as a foreign key in ORDER.

CUSTOMER (Cust_Id, Cust_Name, Address, City, State)
ORDER (Order_Id, Order_Date, Cust_Id)

A referential integrity constraint is a rule that maintains consistency among the rows of two relations. The rule states that if there is a foreign key in one relation, either each foreign key value must match a primary key values in another relation or the foreign key value must be null.

4. Action Assertions: - Business rules and introduced a new category of business rules we called action assertions. Example a typical action assertion might state the following "A student may purchase a bus pass for the one year only if that person is a Cat card holder".

(Q) Define E. F. Codd s Rules on Data Models?

Dr. E .F. Codd s Rules: - Dr. E .F. Codd s the famous mathematician has for introduce 12 rules for the relational model. For database commonly known as Codd s rules. The rules mainly define what is required for a DBMS for it to be considered relational, i.e an RDBMS there is also one more rule. Which specifies the relational model should use the relational way to manage the database. The E.F.Codd s defines 12 rules for relational database, which any database should satisfy in order to be regarded as relational database.

1. The Information Rule.
2. Guaranteed Access Rule.
3. Systematic Treatment of Null-Values.
4. The data description rule.
5. The Comprehensive data sub-language rule.
6. The view updating rule.
7. High Level Insert Update and Delete.
8. Physical Data Independence.
9. Logical Data Independence.
10. Integrity Independence.
11. Distribution Independence.
12. Non-Sub Version Rule.

1. The Information Rule: - All information in the database should be represented in one and only one way that is the form of "Rows & Columns" or tuples.

2. Guaranteed Access Rule: - Any data must be accessible with no ambiguity. This is achieved in DBMS using the „Primary Key value and column name.

3. Systematic Treatment of Null Values: - Any missing information in a table is represented as a Null value. Support for Null values in a RDBMS must be consists on Independent of Database.

4. The data description rule: - The rule requires that a description of the database or metadata should be store in data dictionary that is part of RDBMS.

5. The Comprehensive data sub-language rule: - This rule status that an RDBMS must be completely manages the rule through SQL include DDL, DML and DCL.

6. The view updating rule: - These rule states that all views that are theoretically updatable and are also updatable by the system. This rule is very restricted senses.

7. High Level Insert, Update and Delete: - The system is able to „Insert, Update and Deleted operations fully . It can also perform the operations on multiple rows simultaneously.

8. Physical Data Independence: - These rule states that application programmers remain logically unimpaired when any changes are made in either the storage representation of data.

9. Logical Data Independence: - These rules states that application programs remain logically unimpaired when information preserving changes of any kind that theoretically permit are made in base table.

10. Integrity Independence: - All the integrity constraints like Primary key, Composite Key etc., must be separately from application programs and stored in the catalogs.

11. Distribution Independence: - These rule states that a distributed database must look to theuser or the application as a centralized database. Applications programs and interactive user should not be required to knows where data whenever data are physically centralized or distributed.

12. Non-Subversions rules: - These rule states that s if an RDBMS supports a low level language (record-at-a-time) that permits.

CHAPTER-IV

Normalization of Database Tables

(Q) what is Normalization? Explain in details with an suitable examples?

Definition: - "Normalization is a process for evaluating and correcting table structures, to minimize the data redundancies by reducing the data anomalies".

(OR)

"Normalization is the process of decomposing relations with anomalies to produce smaller, well-structured relations".

Normalization is primary a tool to validate and improve a logical design, so that is satisfies certain constraints that "avoid unnecessary duplication of data". Normalization works through a series of steps called Normal Forms. We have different types of Normal Forms there are.

Types of Normalizations: - In relational model all Normal Forms are broadly classified into two types.

1. Basic Normal Forms → 1 NF, 2 NF & 3 NF.
2. Advanced Normal Form → BCNF, 4 NF & 5NF.

Steps in Normalizations: -

· **First Normal Form (1 NF)** → "Any Multi-valued attributes have been removed", so there is a single at the intersection of each row and column of the table.

· **Second Normal Form (2 NF)** → "Any Partial Functional Dependencies have been removed".

· **Third Normal Form (3 NF)** → "Any Transitive Dependencies have been removed".

· **Boyce Codd Normal Form (BCNF)**→"Any remaining anomalies that result from functional dependencies have been removed"

· **Fourth Normal Form (4 NF)** → "Any Multi-valued Dependencies have been removed".

· **Fifth Normal Form (5 NF)** → "Any remaining anomalies have been removed".

Needs of Normalization: -

The tables are the basic building block in the database design process. Every table has its own structure. In some cases even a good database design contains a poor table structure to identify these tables and convert them as good tables. We need normalization.

The Poor table structure contains anomalies. The main need of Normalization is to reduce the anomalies. We have 3 types of data anomalies, there are

1. Insert Anomalies.
2. Update Anomalies.
3. Delete Anomalies.

1. Insert Anomalies: - If you insert any row in a relation, the remaining rows are affected then we can say that the relation has Insert Anomalies.

2. Update Anomalies: - If you update any value in a relation, then remaining values in the relation are unnecessarily affected. Then we can say that there is an Update Anomalies.

3. Delete Anomalies: - If you delete any row in a relation the remaining rows in a relation are unnecessarily deleted then we can say that there is a Delete Anomalies.

Normalization Process: - The main objective of Normalization process is to break the tables and create new tables. This process contains the following characteristics.

✓ Each table represents the single subject.

✓ No data item will be unnecessarily stored in more than one table.

✓ All Non-Keys attributes are dependents of the Primary Key.

✓ Each table doesn t contain Insertion, Update & Deletion Anomalies.

Multi-Value Attributes: - A Single Attribute contain the more than one values in the relational table. is known as Multi-Value Attribute or Repeating Groups.

Functional Dependencies and Keys: -

Definition: - "A Functional Dependency is a constraints between two attributes or two sets of attributes" The functional Dependency of "B on A" is represented by an arrow, as follows A à B. Here „A is determination and „B is the dependency.

Example: - 1. Consider the relation COURSE (Emp_Id, Course_Title, Date_Completed). We represent the functional dependency in this relation as follows.

Emp_Id, Course_Title à Date_Completed.

Determination: - "The attribute on the Left-hand side of the arrow in a functional dependency is called as a Determination".

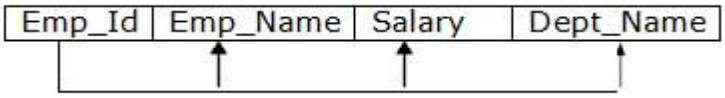
Candidate Key: - "A Candidate key is an attributes or combination of attributes that uniquely identify a row in a relation" The Candidate key must satisfy the following properties.

· **Unique Identification: -** For each row the value of the key must uniquely identify that row.

· **Non-Redundancy: -** No attributes in the key can be deleted without destroying the properties of unique identification.

Representation of Functional Dependencies: - We represent the functional dependencies for a relation using the notations shown in the following.

Example 1: - Show the representation for EMPLOYEE 1

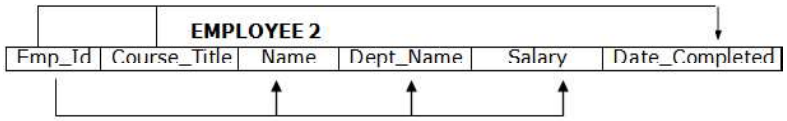


The Horizontal line in the example portrays the functional dependencies. And vertical line drops from the primary key (Emp_Id) and connects to this line. Vertical arrows then point to each of the non-key attributes that are functionally dependent on the primary key.

Example 2: - Consider the example EMPLOYEE 2. There are two functional dependencies in this relation.

1. Emp_Id : Name, Dept_Name, Salary.
2. Emp_Id, Course_Title : Date_Completed.

The primary key of EMPLOYEE 2 is a composite key neither Emp_Id nor Course_Title uniquely identifies a row in this relation.



Transitive Dependency: - "A Transitive Dependency in a relation is a Functional Dependency between two (or more) non-key attributes".

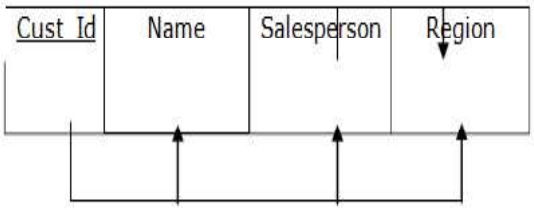
If any three attributes contains following Functional Dependencies is as A → B B→C and C→A.it automatically exits. This type of Functional Dependency is known as „Transitive Dependencies .

Example: - Consider the relation SALES (Cust_Id, Name, Salesperson, Region)

In that above relation the following Functional Dependencies can exits

1. Cust_Id → Name.
2. Cust_Id → Salesperson
3. Salesperson → Region

From 2 & 3 Functional Dependencies can exit automatically Cust_Id → Region.



The Functional Dependencies in the SALES relation are shown graphically Boyce Codd Normal Form (BCNF): -

Definition: - The Boyce Codd Normal Form (BCNF) is based on the Functional Dependency in a relation. A relation in BCNF. "iff (if and only if) every determine is a Candidate Key".

1. First Normal Form: - "A Relation First Normal Form (1 NF) if it contains „No-Multivalued attribute has remove".

This First property of a relation is that the value at the intersection of each row and column must be „**atomic .** Thus a table contains Multivalued attributes or repeating groups is not a relation.

Multivalued attribute from a table: - A given EMPLOYEE may have taken more than one Course the attributes Course_Title and Date_Competed are mutivalue, table with repeating groups.

EMPLOYEE 1

Emp_Id	Name	Dept_Name	Salary	Course_Title	Date_of_Complete
100	ReddyPrasad	Info Systems	55000	Java Dot Net	19 / 06 / 2009 10 / 07 /2010
201	Perumal	Finance	45000	Tax Acc	12 / 08 / 2009
104	Abdulla	Marketing	30000	SCAN VC++	12 / 08 / 2009 04 / 02 / 2010
106	Krishna	Accounting	25000	*****	*******
166	Rakesh	Finance	15000	VC++ SCAN	16 / 06 / 2009 12 / 08 / 2009

Here the Employee with Emp_Id s „100 , „104 & „166 has taken two courses. But Emp_Id„106 has no any course_Title & Date_Completed values are Null.

Eliminating Multivalued attributes: - EMPLOYEE1

Emp_Id	Name	Dept_Name	Salary	Course_Title	Date_of_Complete
100	ReddyPrasad	Info Systems	55000	Java	19 / 06 / 2009
100	ReddyPrasad	Info Systems	55000	Dot Net	10 / 07 / 2010
201	Perumal	Finance	45000	Tax Acc	12 / 08 / 2009
104	Adbulla	Marketing	30000	SCAN	12 / 08 / 2009
104	Abdulla	Marketing	30000	VC++	04 / 02 / 2010
106	Krishna	Accounting	25000	*****	*******
166	Rakesh	Finance	15000	VC++	16 / 06 / 2009
166	Rakesh	Finance	15000	SCAN	12 / 08 /2009

In this above table we remove the Multivalued attributes by using „atomic property (replacing or copying the same values in above row).

The above table is satisfying the atomic property. But we did not identify the any primary key in the table.

(II). Conversion To Second Normal Form: -

Second Normal Form: - "Any Partial Functional Dependencies have been removed".

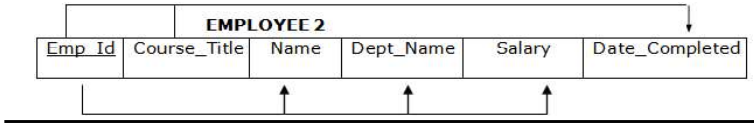
A relation is in Second Normal Form (2 NF) "If it is first normal form and every non-key attribute is fully functionally dependent on the primary key".

Example: - EMPLOYEE 2 is an example of a relation that "is not in second normal form". The primary key for this relation is the composite key Emp_Id, Course_Title. Therefore the non-key attributes Name, Dept_Name and salary is functionally dependent on part of the primary key (Emp_Id) but not on Course_Title.

EMPLOYEE 2

Emp_Id	Name	Dept_Name	Salary	Course_Title	Date_of_Complete
100	ReddyPrasad	Info Systems	55000	Java Dot Net	19 / 06 / 2009 10 / 07 /2010
201	Rakesh	Finance	45000	Tax Acc	12 / 08 / 2009
104	Abdulla	Marketing	30000	SCAN VC++	12 / 08 / 2009 04 / 02 / 2010
106	Krishna	Accounting	25000	*****	*******
166	Rakesh	Finance	15000	VC++ SCAN	16 / 06 / 2009 12 / 08 / 2009

These dependencies are show graphically in figure.



EMPLOYEE 2 is decomposed into the following two relations.

1. EMPLOYEE 1 (Emp_Id, Name, Dept_Name, Salary)
2. E_COURSE (Emp_ID, Course_Title, Date_Completed)

EMPLOYEE 1 relation satisfies condition 1 and is in second normal form. E_COURSE relation satisfies condition 3 above and is also in second normal form.

Sample data for EMPLOYEE 2

EMPLOYEE 2

Emp_Id	Name	Dept_Name	Salary
100	ReddyPrasad	Info Systems	55000
201	Rakesh	Finance	45000
104	Abdulla	Marketing	30000
106	Krishna	Accounting	25000
166	Rakesh	Finance	15000

Sample data for E_COURSE

E_COURSE

Emp_Id	Course_Title	Date_of_Complete
100	Java	19 / 06 / 2009
100	Dot Net	10 / 07 / 2010
201	Tax Acc	12 / 08 / 2009
104	SCAN	12 / 08 / 2009
104	VC++	04 / 02 / 2010
166	VC++	16 / 06 / 2009
166	SCAN	12 / 08 /2009

In the above table Emp_Id „106 has deleted. Because the Employee has not taken any Course.

(III). Conversion to Third Normal Form: -

Transitive Dependency: - "A Transitive Dependency in a relation is a Functional Dependency between two (or more) non-key attributes".

If any three attributes contains following Functional Dependencies is as A→ B B → C and C → A. it automatically exits. This type of Functional Dependency is known as

„Transitive Dependencies .

Third Normal Form: - "Any Transitive Dependencies have been removed".

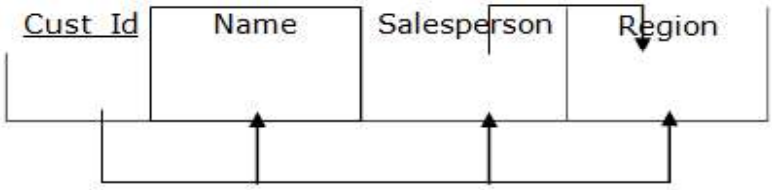
"A relation is in Third normal form (3 NF), if it is in second normal form and no „Transitive Dependencies exists .

Example: - Consider the relation SALES (Cust_Id, Name, Salesperson, Region) SALES relation with sample data

SALES

Cust_Id	Name	Salespersons	Region
8023	A	Smith	South
9167	B	Hicks	West
7924	H	Smith	South
6837	T	Herna	East
8596	E	Hicks	West
7018	AR	Prasad	North

Transitive Dependency in SALES relation



The Functional Dependencies in the SALES relation are shown graphically

In that above relation the following Functional Dependencies can exits

1. Cust_Id → Name.
2. Cust_Id → Salesperson
3. Salesperson → Region

From 2 & 3 Functional Dependencies can exit automatically Cust_Id à Region.

The Cust_Id is the Primary key, so that all of the remaining attributes are functionally dependent on this

attribute. How ever, there is a Transitive Dependency. Region is functionally dependent on Salesperson is functionally dependent on Cust_Id.

How to remove Transitive Dependency: -

The Transitive Dependency can be removed "by decomposing SALES into two relation" as shown in the following.

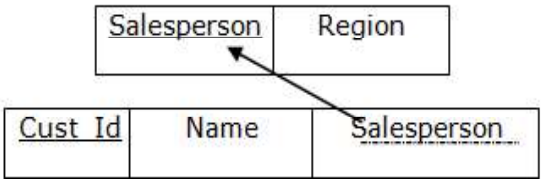
SALES 1

Cust_Id	Name	Salesperson
8023	A	Smith
9167	B	Hicks
7924	H	Smith
6837	T	Hern
8596	E	Hicks
7018	R	Prasad

SPERSON

Salesperson	Region
Smith	South
Hicks	West
Hern	East
Prasad	North

Removing a transitive dependency



The Relation in 3 NF is shown below them new relation are now in Third Normal Form, since no Transitive Dependencies exits.

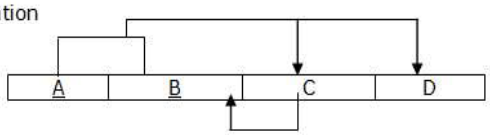
(II). Advanced / High Level Normal Forms:

(Q) Explain the Boyce Codd Normal Form (BCNF)? With an example?

Definition: - The Boyce Codd Normal Form (BCNF) is based on the Functional Dependency in a relation. A relation in BCNF. "iff (if and only if) every determine is a Candidate Key".

Difference between the BCNF & 3NF: - The difference between 3NF and BCNF is „That a Functional Dependencies A → B. the 3rd Normal Form from and BCNF is that a relation. iff „B is a primary key. „A is a not a candidate key. Where as if „B is a primary key of all BCNF. These dependencies to remaining in relations „A must Candidate key.

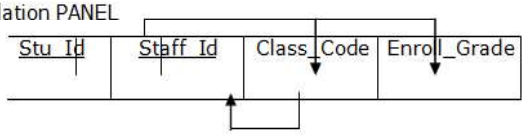
Example: - Consider a relation



PANEL is a 3NF but not BCNF

- A + B → C, D.
- C → B.

Example: - Consider a relation PANEL



PANEL is a 3NF but not BCNF

PANEL is a 3NF but not BCNF

The structure shown in relation is reflected in PANEL.

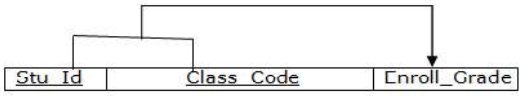
· Stu_Id + Staff_Id à Class_Code, Enroll_Grade.

· Class_Code à Staff_Id.

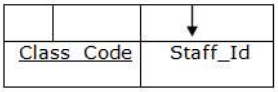
Consider PANEL that contains a Primary Kay „Stu_Id + Staff_Id . When we observe the above relation, we found that there is partial dependency and transitive dependency. So that the relation is in 3NF, but not in BCNF. Because the part of the Primary key depends on the non-key attributes. That means the relations are two candidate keys.

To convert the above relation in to BCNF to change the Primary Key from „Stu_Id+Staff_Id to „Stu_Id + Class_Code . Then split the relation into two parts like STUDENT & CLASS.

Example: - Consider a relation STUDENT



STUDENT is a 3NF and BCNF



CLASS is a 3NF and BCNF

CLASS is a 3NF and BCNF

A relational PANEL structure that is clearly in 3NF, but the table represented by this structure has a major problem, because it is trying to describe two things.

- Staff assignment to Classes.
- Student enrollment information.

Example For BCNF: -

Therefore BCNF is stronger form 3NF, such that every relation in BCNF is also in the 3NF. However a relation in the 3NF is not necessary in BCNF.

Consider the following relation„ INTERVIEW which considers the details of arrangement for all interviews of candidates. The interviews are allocated a special room on the day. The interview a room can be allocated to several interviews as required through a day. Consider the following example “INTERVIEW” table.

INTERVIEW

Candidate_Id	Interview_Date	Interview_Time	Staff	Room_No
C001	24-Aug-2010	10:30	E001	R001
C001	24-Aug-2010	11:30	E001	R001
C002	24-Aug-2010	11:30	E002	R002
C003	24-Aug-2010	10:30	E001	R002

The above table has 3 composite keys which overlapping by sharing the common attribute as shown below.

· Candidate_Id and Interview_Date.

· Interview_Id, Interview_Date and Interview_Time.

· Room_No, Interview_Date and Interview_Time.

We make the composite candidate key (Candidate_Id, Intervirew_Date) as the primary key.

Now the relation has following functional dependencies.

· Candidate_Id, Interview_Date → Interview_Time, Interview_Id, Room_No (Primary Key).

· Candidate_Id, Interview_Date, Interview_Time → Candidate_Id (Candidate Key).

· Room_No, Interview_Date, Interview_Time → Staff, Candidate_Id (Candidate key).

· Staff, Interiew_Date → Room_No (Not a Candidate Key).

INTERVIEW 1

Candidate_Id	Interview_Date	Interview_Time	Staff
C001	24-Aug-2010	10:30	E001
C001	24-Aug-2010	11:30	E001
C002	24-Aug-2010	11:30	E002
C003	24-Aug-2010	10:30	E001

ROOM

Staff	Interview_Date	Room_No
E001	24-Aug-2010	R001 (10:30)
E002	24-Aug-2010	R002 (11:30)

(Q) Explain the 4th Normal Form? With an example?

4th Normal Form: - "A relation is said to be in the 4th Normal Form if it does not contain multivalued dependencies" (MVD s).

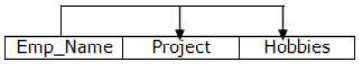
If more than one attribute are depend on the primary key, it is said to be multivalued dependencies. In multivalued dependencies more number of attributes are depend on the primary key. The NF relation must follow the below rules.

· All attributes must be dependent on the Primary Key, but they must be independent of each other.

· No row may contain two or more multivalued facts about an entity.

Multivalued dependencies are the results of 1NF due to data redundancy. If the relation contains multivalues attributes then we have to repeat the every value of that attributes to keep the relation consistent which produces multivalued dependencies.

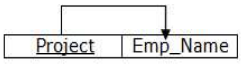
Table Name EMPLOYEE



EMPLOYEE (Emp_Name, Project, Hobby).

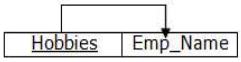
The above relation is converted into 4NF by decomposing it into two tables as follows.

Table Name PROJECT



PROJECT (Project, Emp_Name)

Table Name HOBBY



HOBBY (Hobbies, Emp_Name)

Example: -

Consider the following example table relation of EMPLOYEE that has attributes like Name, Project and Hobbies. A row in the EMPLOYEE table that an employee works for a Projects and has a Hobbies. But an employee can work in more than one project and more than one Hobby. The employee projects and hobbies are independent of one another to keep the relation consist we must have separated tables of represented every combination of employee project and employee hobbies. This contains specific multivalue dependences of on EMPLOYEE table.

EMPLOYEE

Name	Project	Hobbies
Alexis	Microsoft	Reading
Alexis	Oracle	Music
Alexis	Microsoft	Music
Alexis	Oracle	Reading
Mathews	Intel	Movies
Mathews	Sybase	Raiding
Mathews	Intel	Raiding
Mathews	Sybase	Movies

In that above table the values has been multiple-functional dependencies. Now to remove multi-value dependencies in EMPLOYEE table. By using de-composing of a table. The 4th Normal form as following.

PROJECT

Project	Emp_Name
Microsoft	Alexis
Oracle	Alexis
Intel	Mathews
Sybase	Mathews

HOBBY

Hobbies	Emp_Name
Reading	Alexis
Music	Alexis
Movies	Mathews
Raiding	Mathews

(Q) Explain the 5th Normal Form? With an example?

5th Normal Form: - A relation is said to be in 5th Normal Form if it doesn t contains multiple relationships, such type of relation are rarely available in DBMS. If any relation contains multiple relations, they should be resolved into simple relations by using normalization.

Consider the following relations LAB_PRODUCT_ COMPANY.

LAB_PRODUCT_COMPANY

Lpc_Id	Lab_Id	Prod_Id	Company_Id
Lpc0001	L001	P001	C001
Lpc0002	L002	P002	C002
Lpc0003	L003	P003	C003
Lpc0004	L004	P004	C004

In the above relation more than one Foreign keys exits, so this is called multiple relationships. Due this multiple relationship this relation is not in 5th Normal Form.

One Lab in a company can test one production or many products. Similarly one company may produce one product or more products. Therefore the above relation should be normalized to resolve multiple relationships. It is represented in the following.

LAB_PRODUCT

Lab_Id	Prod_Id
L001	P001
L002	P002
L003	P003
L004	P004

LAB_COMPANY

Lab_Id	Company_Id
L001	C001
L002	C002
L003	C003
L004	C004

In "LAB_PRODUCT" relation multiple relationships are not exits. So this relation is 5th. In the same way "LAB_COMPANY" relation doesn t contains multiples relationships. So it is also in 5th Normal Form.

UNIT – IV

Structured Query Language

Chapter – I
Introduction to Structure Query Language (SQL)

Structured Query Language (SQL) is the standard command sets, use to communicate with the relational database management systems (RDBMS).

All tasks related to RDBMS creating tables, queries, modification of tables, deleting of tables from the database. SQL is to granting the access to user and database.

History of SQL: - SQL was first introducing by "Dr. E. F. Codd". He works on Relational Model for large sharing databank. Since it s introducing many researches at the "SanJose" Research Laborites.

The mid of 1970 s development many computer programming languages based on this relation model one of this called as a "Structured English Query Language" (SEQUEL). These languages are used for an IBM prototype machines called as a "Structured English Query Languages Extensional Relational Model" (SEQUEL-XRM).

In 1976 s IBM s re-launched for this second prototype and called as a Relational-Systems (R-Systems). One research organization called relationship software from California commercially available relational database was ORACLE. Today s ORACLE manufactures a wide range of SQL products along with ORACLE relational database management systems software.

Types of SQL: - SQL is broadly classified into two types.

1. Interactive SQL.
2. Embedded SQL.

1. Interactive SQL: - "Interactive SQL is used for directly access the data from the database". Where the output of operation is used for human consumptions. Once a command is specified is "executed and output is immediately view by the users".

2. Embedded SQL: - "In the case of Embedded SQL commands are SQL Commands that are written in some another languages". Such as COBAL, ALP (Assembly Level Programming Language). This makes the programmer very fast and powerful manner.

Advantages of SQL: -

· SQL is a high level language that provides a greater degree of „abstraction .

· SQL deals with number of database management system at a time.

· SQL is an English like computer language which makes interaction between user and database very simply.

· SQL language which being simple and easy to learn can handle complex situation.

· SQL operations are performed at a set level. One select statement can retrieve multiple rows.

(Q) What SQL can Do?

· SQL can Retrieve data fro a Database.
· SQL can Execute Queries against a Database.
· SQL can Insert Records in a Database.
· SQL can Update Records in a Database.
· SQL can Delete Records in a Database.
· SQL can Creates New Tables in a Database.
· SQL can Creates Views in a Database.

(Q) Define SQL * PLUS?

SQL * PLUS is a software product from Oracle

Corporation that allows users to interactively use the SQL commands, produce formatted reports and support written command-procedures to access Oracle Database. Though SQL * PLUS, a user can Do:

· Enter, Edit, Store, Retrieve and Run SQL Commands.

· Format, Perform calculations, Print Query results in the form of Reports.

· Access and Copy data between SQL Database.

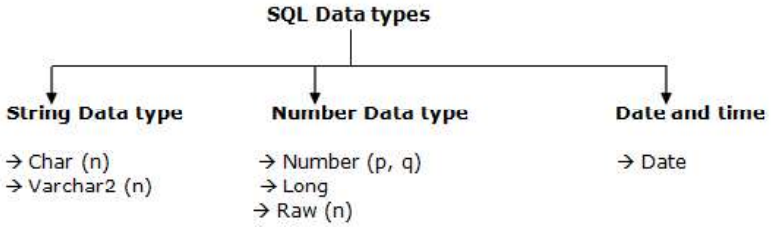
· Send messages to and accepts responses fro an End-Users.

(Q) Explain the difference between SQL and SQL * Plus?

SQL	SQL * PLUS
SQL is a special-purpose language used to define,access and manipulate data in RDBMS like Oracle	SQL * PLUS is a command line tool that allows user to type SQL commands to be executed directly against an Oracle Database.
SQL provides set of commands used to communicate with the Database to perform specific tasks.	SQL* PLUS is the environment where you canactually type your commands to perform specific tasks.
SQL commands can be used to create tablesobjects like Create, Alter, drop etc.	SQL * PLUS commands can be used to describe the structure of Database objects. For example: DESCRIBE/DESCRIPTION is DESC command.
SQL commands cannot be abbreviated.	In SQL * PLUS commands can be abbreviated.Like the commands EDIT can be ED.
Every SQL commands must be ended withsemicolon (;)	In SQL * PLUS semicolon (;) is optional.

(Q) Explain the different types of Data types in SQL?

SQL Data types: - Most programming language requires programmed declaration by using data types. The most database system required the user to specific the type of each data fields. The available data type varying from one programming language to other languages. The SQL supports the following scalar data types.



I. String Data type: -

1. Char (n): - This type of data type represented as a "Fixed length of String" of exactly n. Here „n is a allowed grater than zero, positive integers only. The maximum size for „n is 255 bytes.

Syntax: - Char (n) → **Example: -** Char (4).

2. Varchar2 (n): - This type of data type represented a variable-length character length. The maximum size for „n is 4000 bytes. Here „n is a allowed grater than zero, positive integers only.

II. Number Data type: -

1. Numeric (p, q): - This data type represented a decimal number of „p digits and signed with assume decimal points of „q digits. From the right side „p and q are integer.

Syntax: - Numeric (p, q) **Example: -** Numeric (3, 4)

2. Long: - Variable length character data up to 2^{31}-1 bytes or 2GB per row

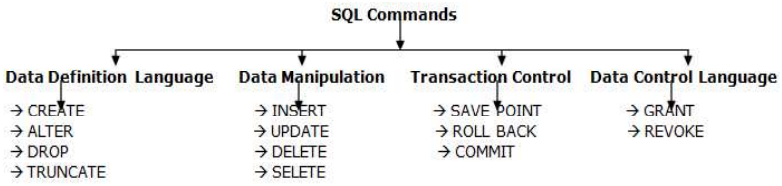
3. Raw (n): - Variable length raw binary data. A maximum„n must be specified. The size is 2000 bytes per row.

III. Date and Time: -

1. Date: - The date type is represented date values. It is fixed length date and time. Default format is "DD-MON-YYYY" example: 13-Jul-2017. The size of data is 7 bytes foe each row.

(Q) Explain the Types of SQL Commands?

SQL statements are divided into the following categories



Data Definition Language (DDL): - DDL Commands is used to "Define the Database Structure" by "Creating, Altering, Dropping and Truncating Tables". The logical data definition statements are Create Table, Creating View, Creating Index, Alter Table, Drop Table and Drop Index.

Data Manipulation Language (DML): - DML Commands allowed the user to "Manipulate data".

Manipulate data refers to "Inserting, Updating and Deleting of data".

Data Query Language (DQL): - DQL Command is one of the most commonly used SQL statements. Using DQL we can "Access data from the Table". The SQL has only one DQL statement is "SELECT".

Data Control Language (DCL): - The Data Control Language consists of commands that control the "User Access to the database objects". Thus DCL is mainly related to the "Security issues" that is determining who has access to the database objects and what operations they can perform on them. The task of the DCL is to prevent unauthorized access to data. The Database Administrator (DBA) has the power to give and take the privileges to a specific user, thus giving or revoking access to the data. The DCL commands are GRANT and REVOKE.

Data Administration Statement (DAS): - Data Administration command allows the user to perform "Audits and Analysis" on operations within the database. Two data administration commands are START AUDIT and STOP AUDIT. Database Administration is the overall administration of the database and data administration is only a subset of that.

Transaction Control Statement (TCS): - TCS are statement which manages all the changes made by the DML statements. Some of the Transaction are "COMMIT, ROLLBACK, SAVEPOINT and SET Transaction".

(I) Data Definition Language Commands (DDL): -

1. Create for Tables: - This Command is used to "Create Object / Table in the Database".

Syntax: - create table <table name> (column name 1 data type(size), column name2 data type (size) Column n name data type(size));

Ex: - create table branch(branch_code number (10),branch_name varchar2(10), city varchar2(5));

Restriction of Table Name:-

- ✓ First Letter should be Alphabet.
- ✓ There should not be any space in the Table Name.
- ✓ Any special symbols like (-) Hyphen can t be used.
- ✓ Table Name should not be keywords in Oracle

2. Alter Table: - Alter Table statements neither support any kind of changes to the width or data type of an existing column nor the deletion of an existing column.

· **Adding a New Column: -** To add a New Column to the Table.

Syntax: - alter table <table name> add (new column name 1 data type(size), new column name 2 data type(size), , new column name n data type (size));

Ex: - alter table branch add (address varchar2(10), telephone number(8));

· **Modifying the Existing Column: -** To Modifying the Existing table values.

Syntax: - alter table <table name> modify(existing column name 1 data type(size) , existing column name 2 data type(size));

Ex: - alter table branch modify (address varchar2(20), telephone number(10));

3. Drop Table: - This command is use "Drop the Structure / Table (or) Delete the Objects / Tables from the Database permanently .

Syntax: - drop table <table name>;

Ex: - drop table emp;

4. Truncate Table: - This command is used to "Delete (or) Remove all records / rows from a table", and to release the storage space used by that table. When using the Truncate Table statement, you cannot„rollback (Undo).

Syntax: - truncate table <table name>;

Ex: - truncate table emp;

(II) Data Manipulation Language (DML): -

1. Insert Table Values: - This command is used to "Insert data into a table" (or) Insert new Records in to the Table.

Syntax: - insert into <table name> values(„& column name1 , „ & column name 2 , „ & column name 3 ,„ & column name n);

Ex: - insert into branch values(„& branch_code , „ & branch_name , „ & city);(OR)

Syntax: - insert into <table name>(„column name1 , „column name 2 , „ column name 3 , …… „column name n) values(„Column Value 1 , „Column Value 2 , „Column Value 3 ,);

Ex: - insert into branch („branch_code , „ branch_name , „ city) values (1045, „SBI , „Piler);

2. Update Table Values: - This command is used to "Change the value of column (or) Updates existing data within a table".

Syntax: - update <table name> set <column name 1> = <value 1>,

<column name 2> = <value 2>, ……

<column name n> = <value n> [where <condition>];

Ex: - update branch set telephone = 9885712847 where branch_code = „007 ;

3. Delete the Table: - This command is used to "Deleting particular the records in a Table".

Syntax: - delete from <table name> [where <condition>]; Ex: - delete from branch where branch_code = „007 ;

4. Select: - This command is used to "Selected / retrieve data from the Database tables".

· **Display the Table Records: - Syntax: -** select * from <table name>;

· **View the Selective columns: -**

Syntax: - select column names from <table name>;

Ex: - select branch_code ,city from branch ;

· **View the only particular Record :-**

Syntax: - select * from <table name> where column name = " " ;

Ex: - select *from branch where city= "Tirupati" ;

(III) Transaction Control Statement (TCL): -

1. Save Point: - This is used to "Identify a point in a transaction to which you can later rollback". One transaction can have any number of Save Points.

Syntax: - savepoint <save point name>;

2. Roll back: - This is used "Restore database to original since the last COMMIT". Move to certain save points created for a transaction. Before completion of the transaction.

Syntax: - rollback <save point name>;

3. Commit: - Commit it used to end any transaction (or) "Save Work Done".

Syntax: - commit; OR commit work;

(IV) Data Control Language (DCL): -

1. Grant: - This command is used to "Grant the permissions on one table to other user (or) Gives user s access privileges to Database". To Grant all permission, we can t drop the table.

Syntax: - Grant [type of privilege] ON [user name]. [table name] to [user name] [with Grant option];

Example: - SQL> Grant select ON scott . fareedemp to shaik with grant option read;

· **Grant all:** - This command is used to Grant the permissions on all tables to other user. To Grant all permission, we can t drop the table.

Syntax: - Grant All [type of privilege] ON [user name]. [table name] to [user name] [with Grant option];

Example: - SQL> Grant All select ON scott . fareedemp to shaik with grant option write;

2. Revoke: - This command is used to "Take-out some or all permissions on a table from a user". (or) "Withdraw access privileges given with the GRANT command.

Syntax: - Revoke [type of privilege] ON [user name]. [table name] from [user name];

Example: - SQL> Revoke select ON scott . fareedemp from shaik ;

(Q) What is the Difference between DELETE, TRUNCATE and DROP commands in SQL?

(1) DELETE: - The DELETE command is used to remove rows from a table. A WHERE clause can be used to only remove some rows. If no WHERE condition is specified, all rows will be removed. After performing a DELETE operation you need to COMMIT or ROLLBACK the transaction to make the change permanent or to undo it.

```
SQL> SELECT COUNT (*) FROM

 emp;COUNT (*)
 ------------
      14

SQL> DELETE FROM emp WHERE job = 'CLERK';

4 rows deleted.

SQL> COMMIT;

Commit complete.

SQL> SELECT COUNT (*) FROM

 emp;COUNT (*)
 ------------
      10
```

(2) TRUNCATE: - TRUNCATE removes all rows from a table. The operation cannot be rolled back and no triggers will be fired. As such, TRUCATE is faster and doesn't use as much undo space as a DELETE.

SQL> TRUNCATE TABLE emp;

Table truncated.

SQL> SELECT COUNT(*) FROM emp;

COUNT(*)
0

(3) DROP: - The DROP command removes a table from the database. All the tables' rows, indexes and privileges will also be removed. No DML triggers will be fired. The operation cannot be rolled back.

SQL> DROP TABLE emp;

Table dropped.

SQL> SELECT * FROM emp; SELECT * FROM emp

* ERROR at line 1:

ORA-00942: table or view does not exist.

DROP and TRUNCATE are DDL commands, whereas DELETE is a DML command. Therefore DELETE operations can be rolled back (undone), while DROP and TRUNCATE operations cannot be rolled back.

(Q) Explain Query In SQL?

A "Query is a Question" in the Structured Query Language. As its name denote SQL is best at writing Queries. Anything that is apart of RDBMS table can be retrieved at the users request using SQL. A SQL Query has five basic parts that are

1. Select
2. From
3. Where
4. Group By
5. Order By

1. Select: - This command comprise of the list of columns that have to be displayed in the Query result. If you want all the columns of a table to be displayed, instead of writing down all the column names you could simply use an astric (*****).

2. From: - This part identified the source tables of all the columns. These could be a single table or more than one table.

3. Where: - This is an "Optional Part of a Query" this part specified the limits that the result. If a Query does not have a Where Clause all the rows are selected comparisons can also is specified in this clause.

Ex:-If the librarian wants to see the list of all members who have borrowed books on the date 15-08-2009 Select book code from member where issues date = „15-08-2009 ;

4. Group By: - This is another optional part of a Query. If is used only when the results of the Query have to be Grouped based on criteria. Expressions can be specified as criteria for this clause.

Ex: - If the average salary of employees in a department is to be from the following Query can be used.
Select dept_no, avg(pay) from employee group by dept_no;

5. Order By:- This is an optional clause that controls the order in which the rows to be display by the Queries are ordered.

Ex:- If the Librarian wants the results of the Query to be ordered by book code, this is the following query Select book_code from member where issues date = „15-08-2009 order by book_code;

(Q) Difference between Where and Having Clause in SQL?

WHERE CLAUSE	HAVING CLAUSE
Where Clause of the SELECT statement ismainly used for conditional retrieved of records from a database table.	Having Clause for SELECT statement ismainly used for conditional retrieval of records from grouped results.
Where Clauses can ne used without the Group By Clause.	Having Clause cannot be used without the Group By Clause.
Where Clause cannot contain Group Function.	Having Clause can contain Group Functions.
Where Clause selects records before grouping.	Having Clause selects records after grouping.

(Q) What are the Integrity constraints?

Integrity constraints are used to ensure accuracy and consistency of data in a relational database. Data integrity is handled in a relational database through the concept of referential integrity. There are many types of integrity constraints that play a role in referential integrity.

Types of Integrity Constraints: -

- Entity Integrity.
- Referential Integrity.
- Domain Integrity.
- User Define Integrity.

Entity integrity: - The entity integrity constraint states that no primary key value can be null. This is because the primary key value is used to identify individual Tuples in a relation. Having null value for the primary key implies that we cannot identify some tuples. This also specifies that there may not be any duplicate entries in primary key column key row.

Referential Integrity: - The referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples in the two relations. Informally, the referential integrity constraint states that a tuples in one relation that refers to another relation must refer to an existing tuple in that relation. It is a rule that maintains consistency among the rows of the two relations

Domain Integrity: - The domain integrity states that every element from a relation should respect the type and restrictions of its corresponding attribute. A type can have a variable length which needs to be respected. Restrictions could be the range of values that the element can have, the default value if none is provided, and if the element can be NULL.

User Defined Integrity: - A business rule is a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behavior of the business. E.g.: Age>=18 && Age<=60.

(Q) Grouping Data from Tables in SQL?

1. Group by Clause
2. Having Clause

1. Group by Clause: - The Group by Clause is another section of the select statement. This option clause tells Oracle to group rows based on distinct values that exist for specified columns. Therefore it crated a database containing several sets of records group together based on a condition. In Group by option we can use aggregate functions also. The syntax of Group by Clause is following.

Syntax: - select <column list> from <table name> group by <column name>;

Example: - Retrieve the product number and the total quantity order for each product from the sales_order details tables.

Table Name: - Sales_Order

Order_no	product_no	Quality_order	Quality_Dis
30402	p001	10	10
30402	p004	03	03
30402	p006	07	07
40300	p002	04	04
40300	p005	10	10
30240	p003	02	02
30430	p001	06	06
30670	p006	04	04
30670	p004	01	01
30700	p006	08	08

SQL> select product_no, sum(Quality_order) from sales_order group by product_no;

Output: -

product_no	Quality_order
P001	16
P002	04
P003	02
P004	04
P005	10
P006	19

2. Having Clause: - The Having Clause can be used in

conjunction with the group by clause. "Having imposes a condition on the group by clause". Which further filters the groups created by the group by clause?

Syntax: - select <column list> from <table name> group by <column> having <condition>;

Ex: - Retrieve the product_no and the total quality ordered for products p001 & p004 from the sales_order table.

Order_no	product_no	Quality_order	Quality_Dis
30402	p001	10	10
30402	p004	03	03
30402	p006	07	07
40300	p002	04	04
40300	p005	10	10
30240	p003	02	02
30430	p001	06	06
30670	p006	04	04
30670	p004	01	01
30700	p006	08	08

SQL> select product_no, sum (Quality_order) from sales_order group by product_no having sum(Quality_Order)>10;

Output: -

product_no	**Quality_order**
P001	16
P004	04
P006	19.

(Q) Explain the SQL Operators?

Operators and conditions are used to perform operations such as additions, subtractions, comparisons on the data items is an SQL Statements. "Operators is a symbol, which represent some particular actions. Operator operates on operands".

There are two types of operators like Unary and Binary operators. Unary operators contain only one operand that is +ve or –ve values. Binary operators contain more than two operands.

Example: - 1. Unary Operators: - +ve values or –ve values.

2. Binary Operators: - A + B ; Here "+" is a operator between two operands.

There are following SQL Operators:

1. Arithmetic Operators

2. Comparison Operators

3. Logical Operators

4. Set Operators.

1. Arithmetic Operators: - Arithmetic Operators are used in SQL expressions to Add, Subtract, multiply, divide and negate data values. The result of this expression is a numeric value.

Arithmetic Operators	
Unary Operators	
+, -	Denotes as +ve or –ve expressions
Binary Operators	
+	Addition
-	Subtraction
*	Multiplication
/	Division

3. **Comparison Operators: -** These operators are used to compare one expression with another. The result of a comparison is True, False or Unknown.

Operator	Description
=	Equality
!=	Not Equal to
>	Grater than
<	Less than
>=	Grater than Equal
<=	Less than Equal
IN	Equal to any member of list
NOT IN	Not Equal to any member of list
IS NULL	Testes for Null Values
LIKE	Return True when the First Expression matches the pattern of Second Expression. Wild Card like (%) allowed.
ALL	Compared a value to every value in the list
ANY	Compared a value to each value in the list
BETWEEN	Compared two lists

3. Logical Operators: - A logical operator is used to produce a single result from combining the two separate conditions. There are AND, OR , NOT Operators.

Operator	Definition
AND	Return True if Both component conditions are True; otherwise return False
OR	Return True if either component conditions is True; otherwise return False
NOT	Return True if the condition is False; Otherwise return False.

Truth Table for Logical Conditions: -

List(a)	Operator	List(b)		Result
True	AND	True	=	True
True	AND	False	=	False
False	AND	True	=	False
False	AND	False	=	False

List(a)	Operator	List(b)		Result
True	OR	True	=	False
True	OR	False	=	False
False	OR	True	=	False
False	OR	False	=	True

NOT Operator	Result
True	False
False	True

Consider the following example a=10; b=15; c=20;

(i) if a>b AND a>c = False AND False à False.

(ii) if a<b AND b<c = True AND True à True.

(iii) if a>b OR b>c = False OR False à True.

(iv) if a<b OR b<c = True OR True à False.

(v) if NOT a>b = NOT False à True.

(vi) if NOT a<b = NOT True à False.

4. Set Operator: - Set Operators combine the results of two separate Queries into a single result.

Operator	Description
UNION	Returns all Distinct row a from both Queries
UNION ALL	Returns all row a from both Queries
INTERSECT	Returns all row selected by both Queries
MINUS	Returns all distinct rows that are in the first query but notin the second one.

(Q) Describe the NULLS in Action?

According to the Dr. E.F.Codd s rules systematic treatment Null Values the missing information to a table represented by Null Values. If a table contains Null Values or Not Null Values the commands can be used.

SQL provides a special construct represented by the keyword NULL. According to standard, "There is no Literal for Null values, although the keyword NULL is used in some places to indicate that a Null value is desired". The Literal Null can appear only in the following contexts.

- As a default specification within a column or domain definition.
- As an insert/update item specifying a value to be placed in a column position on INSERT/ UPDATE.
- As a CASE result
- As a part of referential specification.

1. IS NULL: It displays records which having Null values to the specified attributes.

2. IS NOT NULL: It display s records which is Not having Null Values to the specified attributes.

Effect of NULLS: As a general rule it is always better to avoid Nulls. But there are certain cases when you cannot prevent having Nulls in your table. In order to use properly. The following Nulls will behave in different situation.

- ✓ Null values in aggregate functions are ignored.
- ✓ Conditional statement is extended from the Boolean two values, therefore True/False. If three valued therefore True/False/Unknown logic.

✓ All operations except || will return null if any of the operands are Null.

✓ To test for Null, the comparison operators IS NULL and IS NOT NULL must be used.

✓ A conversion function with Null as argument returns Null.

(Q) Explain Aggregate Function or Grouped Functions?

The Aggregate Functions greatly enhanced the power of the SQL statement. They let you summarized the data from the tables. An Aggregate Functions takes an entire column of the data as its argument and produces a single data item that summarized the columns. SQL provides six aggregate functions. These are powerful tools and can improve the data retrieval power considerably. These are some rules, which must be following while using these functions. They are:

1. SUM
2. AVG
3. MAX
4. MIN
5. COUNT
6. COUNT(*)

Rules of Aggregate Functions:

✓ For SUM and AVG the argument must be of type Numeric

✓ Except for the special case COUNT (*), the argument may be preceded by the key word DISTINCT to eliminate the duplicate rows before the functions is applied to a column. The DISTINCT is legal for MAX and MIN but meaningless.

✓ The special function COUNT (*) which is used to all rows without any duplicate elimination and so the keyword DISTINCT is not allowed for this functions.

✓ The argument can t involve any Aggregate functions references or table expressions at any level of nesting. For Example the SQL “select avg(min(qtly) as “ is illegal.

✓ Any null in the column is eliminated before the function is applied.

✓ When using MAX & MIN with string data. The comparison of the strings is dependent on the character set.

Aggregate Functions:

1. SUM () Function: The SUM ( ) Function returns the "Sum of Values in the Expression"

Syntax: - select sum(column name) column name from <table name >;

Example: - To view the reservation table

SQL> select * from reservation;

Output: -

Pass name	**Class**	**Total_fare**
Shaik	B	3000
Mohammad	F	3500
Fareed	E	2500

SQL> select sum (total_fare) total_fare from reservation;

Output: - Sum is 9000

2. AVG () Function: The AVG ( ) Function return the "Avg of Values in the Expression".

Syntax: - select avg(column name) column name from <table name >;

Example: - To view the reservation table

SQL> select * from reservation;

Output: -

Pass_name	**Class**	**Total_fare**
Shaik	B	3000
Mohammad	F	3500
Fareed	E	2500

SQL> select avg(total_fare) total_fare from reservation ;

Output: - Avg is 3000

3. MAX () Function: - The MAX ( ) Function return the "Maximum of Values in the Expression".

Syntax: - select max(column value) column name from <table name>;

SQL> select max(total_fare) total_fare from reservation ;

Output: - Max is 3500

4. MIN () Function:- The MIN ( ) Function return the "Minimum of Values in the Expression".

Syntax: - select min(column value) column name from <table name>;

SQL> select min(total_fare) total_fare from reservation ;

Output: - Min is 2500

5. COUNT ( ) Function:- COUNT () Function return the "Number of values in a columns"

Syntax: - select count(column value) column name from <table name>;

SQL> select count(pass_name) pass_name from reservation

Output: - Count is 3

6. COUNT (*) Function:- COUNT (*) Function return the "Number of rows in a table"

Syntax: - count(*)from <table name>;

For suppose we want to count number of columns in table

SQL> select count (*)from reservation ;

Output: -

Pass_name	Class	Total_fare
Shaik	B	3000
Mohammad	F	3500
Fareed	E	2500

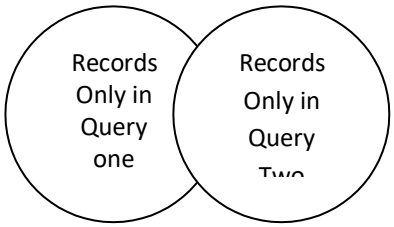
(Q) Explain the Set Operators in SQL?

In SQL supports four Set Operations namely

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

1. UNION Operator: - Union is a combination of Two or more rows from similar tables to produce results. This combination is done using the Union Operation. The Union

Operator combination two or more rows from similar tables and returns only „Distinct values from done. "It doesn t display duplicates rows".



Eliminate Duplicate: - The most RDBMS provides the provision of retaining or eliminating the duplicates. The UNION verb is eliminates the duplicates. But UNION ALL is allowed the any duplicates of data. But in some cases it improves performances to use UNION ALL instead of UNION.

Restriction on using a Union Clause:

- Number of columns in that all the queries should be the same.
- The data type of the column in each query must be the same.
- Aggregate functions can t be use with Union Clause.
- Unions can t be use in Sub-Queries.

Syntax: -
```
select <column names> from <table name 1> where [ condition]
                    UNION
Select<column names> from <table name 2> where [condition] ;
```

Query: - List all Distinct Employee id, names in the Table1 and Table2

SQL> select * from far1;

ID	NAME	SALARY
100	Naresh	50000
101	Kumar	40000
102	Swamy	35000
103	Fareed	20000
104	Lakshmi	15000

```
SQL> select * from far2;
        ID NAME                 SALARY
---------- -------------------- ----------
       106 Kalavathi            10000
       105 James                15000
       103 Fareed               20000
       107 Gayathri             15000
       101 Kumar                40000
```

SQL> select id,name from far1 union select id,name from far2;

Output: -

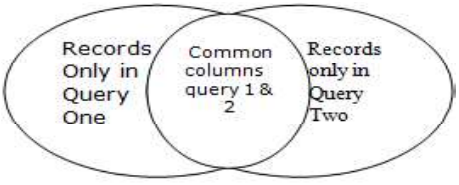
ID	NAME
100	Naresh
101	Kumar
102	Swamy
103	Fareed
104	Lakshmi
105	James
106	Kalavathi
107	Gayathri

8 rows selected.

2. Syntax: - select <column names> from <table name> where [condition]

UNION ALL Operators: - "This Operator is used to display all records from both Tables including Duplicating records**".** So it is useful to find the total number of records in two tables. It is represented by a keyword UNION ALL.

Syntax: - select <column names> from <table name> where [condition]
UNION ALL
Select<column names> from <table name> where [condition] ;



Query: - List all Employee id, names in the Table1 and Table2

SQL> select id,name from far1 union all select id, name from far2;

Output: -

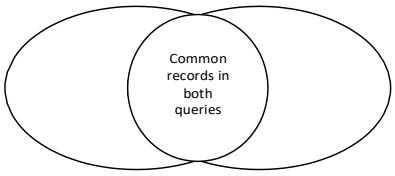
ID	NAME
100	Naresh
101	Kumar
102	Swamy
103	Fareed
104	Lakshmi
106	Kalavathi
105	James
103	Fareed
107	Gayathri
101	Kumar

10 rows selected.

Difference between Union and Union All: -

UNION	UNION ALL
It displays distinct rows from both tables.It cannot display duplicate records.	It can display distinct rows and duplicate records also.
Using the keyword UNION	Using the keyword UNION ALL
We cannot find out the total number of records in a table	We can find out the table numbers of records in a table.

3. INTERSECT Operator: - "The Intersect Operator returns the rows that are common between the Tables".



Query: - List the common Employee id, names from Table1 & Table2

Syntax: - select <column names> from <table name> where [condition]
INTERSECT
Select<column names> from <table name> where [condition] ;

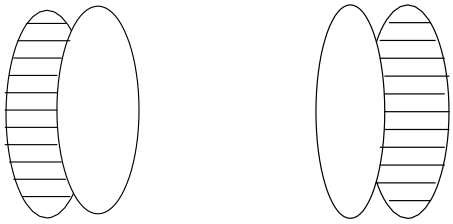
SQL> select id, name from far1 intersect select id, name from far2;

Output: -

ID	NAME
101	Kumar
103	Fareed

2 rows selected.

4. MINUS Operator: - The Minus Operator combine the result of the two query and returns only those values selected by the "First Query and not the Second Query".



A – B **B – A**

Query: - List the Employee id, names present in Table1 but not in Table2.

Syntax: -

select <column names> from <table name> where [condition]
MINUS
Select<column names> from <table name> where [condition]

SQL> select id,name from far1 minus select id, name from far2;

Output: -

ID	NAME
100	Naresh
102	Swaamy
104	Lakshmi

Query: - List the Employee id, names present in Table2 but not in Table1.

SQL> select id, name from far2 minus select id, name from far1;

UNIT – V

Procedural Language / SQL (PL/SQL)

Chapter – 1
Introduction of PL/SQL

Introduction: -

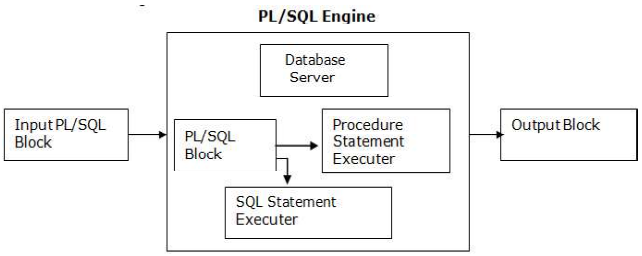
· SQL does not support programming. It is used to write single line statement called Quarries.

· To support programming constructs, additional features are added to SQL and name it as PL/SQL.

· PL/SQL is very useful to develop programming.

· PL/SQL supports almost all programming constructs like variables, conditional statements, looping etc.,

Architecture of PL/SQL: -



PL/SQL Engine

PL/SQL Block Structure: -

The Programming area in PL/SQL is called block. This block mainly contains Three parts. They are:

1. Declarative Part.
2. Executable Part.
3. Exception Handling part.

Structure:-

```
SQL>  Declare
      ……
      ……          Variable Declaration
      ……
      Begin
            Statement 1
Statement 2       Executable Part
      ……
      End;
Exception         Error Handling Part
```

The above syntax shows that every PL/SQL block declarative part should be indicated with a key word called "Declare". Executable part must be enclosed with in begin and end key words. Exception Handling part can be indicated with a key word called it "Exception".

Input & Output Statement in PL/SQL: -

1. Input Statement: - The Input statement in PL/SQL is **" & "** Operator. The **„&** Operator it is to give the Input values to the Variables.

Ex:- a : = & a;

x : = & x; ……

2. Output Statement: -The Output Statement in PL/SQL is "dbms_output.put_line(„Message /Variables);"

Ex: - SQL> dbms_output.put_line(„ Welcome to PL/SQL);

(Q) Explain various Advantages and Disadvantages of PL/SQL? Advantages of PL/SQL: -

Block Structures: PL SQL consists of blocks of code, which can be nested within each other. Each block forms a unit of a task or a logical module. PL/SQL Blocks can be stored in the database and reused.

Procedural Language Capability: PL/SQL consists of procedural language constructs such as conditional

statements (if else statements) and loops like (FOR loops).

Better Performance: PL SQL engine processes multiple SQL statements simultaneously as a single block, thereby reducing network traffic.

Error Handling: PL/SQL handles errors or exceptions effectively during the execution of a PL/SQL program. Once an exception is caught, specific actions can be taken depending upon the type of the exception or it can be displayed to the user with a message.

Modularity: - Modularity means divide an application or a process into manageable, well defined modules. PL/SQL allows process to be divided into different modules such as procedures and functions called „subprograms .

Portability: - Applications written in PL/SQL are portable to any platform on which Oracle runs. One you write a program in PL/SQL, it can be used in any environment without any change at all. (WORM).

Disadvantages of PL/SQL: -

Proprietary of Oracle: - PL/SQL is proprietary to Oracle which means if you were to change database vendors then you would have to re-write all your Oracle PL/SQL programs. Also if you use a mixture of databases or develop applications to run on different databases, you will either have to support lots of source code or write your applications in a database-neutral language like: Java or Visual Basic.

Poor I/O Feature: - The other limitation of Oracle PL/SQL is that there is very little support for I/O either to Read/Write files to user interface.

Insufficient Memory: - More Memory may be required on the Oracle database server when using Oracle PL/SQL packages as whole packages is loaded into memory as soon as any object in package is accessed.

(Q) Explain Fundamental of PL/SQL (or) PL/SQL Language Elements?

Like every other programming language, PL/SQL has a set of elements like Character Set, Reserved Words, Identifiers, and Literals etc. these elements of PL/SQL are used to represented real world entity or objects and operations.

- Character Set.
- Reserved Words.
- Delimiters.
- Literals.
- Lexical Units.

Character Set: - PL/SQL programs are written as lines of text using a specific set of characters. The PL/SQL characters set includes: Letters (a..z and A..Z), Numbers (0..9) Tabs, Space, Special Symbols (+,-, <>, !, #, [], {}). PL/SQL is not a case sensitive.

Reserved Words: - Reserved words have special syntactic meaning of PL/SQL and so it can not be redefined. Reserved words are generally written in upper case to promote readability, but it can be written in lower or mixed case.

Delimiters: - Delimiters are simple or compounds symbols that have special meaning to PL/SQL. Delimiters are like: +, -, <>, !=, ||, /*, :=, etc.

Literals: - A literal is an explicit numeric, character, string or Boolean values that are not represented by identifiers.

Lexical Units: - A line of PL/SQL text contains groups of characters known as lexical unit. To improve readability, you can separate lexical units by spaces. In fact, you must separate adjacent identifiers by a space of punctuations.

(Q) Explain how to declare variables and constants in PL/SQL?

PL/SQL Variables: - These are placeholders that store the values that can change through the PL/SQL Block.

The General Syntax to declare a variable is: Variable_Name datatype [NOT NULL := value];

· Variable_Name is the name of the variable.
· datatype is a valid PL/SQL datatype.
· NOT NULL is an optional specification on the variable.
· value or DEFAULT values also an optional specification, where you can initialize a variable.
· Each variable declaration is a separate statement and must be terminated by a semicolon.

For example: - If you want to store the current salary of an employee, you can use a variable.

```
DECLARE
salary number (6);
```

· "salary" is a variable of datatype number and of length 6.

When a variable is specified as NOT NULL, you must initialize the variable when it is declared.

For example: - The below example declares two variables, one of which is a not null.

```
DECLARE
salary number(4);
dept varchar2(10) NOT NULL := "HR Dept";
```

The value of a variable can change in the execution or exception section of the PL/SQL Block. We can assign values to variables in the two ways given below.

1) We can directly assign values to variables.

The General Syntax is: - variable_name:= value;

2) We can assign values to variables directly from the database columns by using a SELECT.. INTO statement.

The General Syntax is: SELECT column_name INTO variable_name FROM table_name [WHERE condition];

Example: - The below program will get the salary of an employee with id '1116' and display it on the screen.

```
DECLARE
var_salary number(6); var_emp_id number(6) = 1116;
```

```
BEGIN
        SELECT salary INTO var_salary FROM employee
        WHERE emp_id = var_emp_id; dbms_output.
put_line(var_salary);
        dbms_output.put_line('The employee '|| var_emp_id
|| ' has salary ' || var_salary); END;/
```

NOTE: - The backward slash '/' in the above program indicates to execute the above PL/SQL Block.

In PL/SQL block, variables are declared in the DECLARE section.

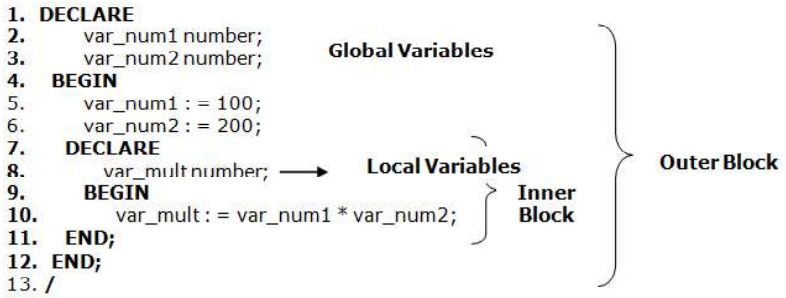
Scope of Variables: -

PL/SQL allows the nesting of Blocks within Blocks i.e., the Execution section of an outer block can contain inner blocks. Therefore, a variable which is accessible to an outer Block is also accessible to all nested inner Blocks. The variables declared in the inner blocks are not accessible to outer blocks. Based on their declaration we can classify variables into two types.

Local Variables - These are declared in an inner block and cannot be referenced by outside Blocks.

Global Variables - These are declared in an outer block and can be referenced by its itself and by its inner blocks.

For Example: - In the below example we are creating two variables in the outer block and assigning their product to the third variable created in the inner block. The variable 'var_mult' is declared in the inner block, so cannot be accessed in the outer block i.e. it cannot be accessed after line 11. The variables 'var_num1' and 'var_num2' can be accessed anywhere in the block.



PL/SQL Constants: - As the name implies a constant is a value used in a PL/SQL Block that remains unchanged throughout the program. A constant is a user-defined literal value. You can declare a constant and use it instead of actual value.

For example: - If you want to write a program which will increase the salary of the employees by 25%, you can declare a constant and use it throughout the program. Next time when you want to increase the salary again you can change the value of the constant which will be easier than changing the actual value throughout the program.

The General Syntax to declare a constant is: constant_name CONSTANT datatype : = VALUE;

· constant_name is the name of the constant i.e. similar to a variable name.

· The word CONSTANT is a reserved word and ensures that the value does not change.

· VALUE - It is a value which must be assigned to a constant when it is declared. You cannot assign a value later.

For example: - To declare salary_increase, you can write code as follows:

```
DECLARE
        salary_increase CONSTANT number (3) := 10;
```

You must assign a value to a constant at the time you declare it. If you do not assign a value to a constant while declaring it and try to assign a value in the execution section, you will get a error. If you execute the below Pl/SQL block you will get error.

```
DECLARE
        salary_increase CONSTANT number (3); BEGIN
        salary_increase : = 100; dbms_output.put_line
(salary_increase); END;
```

(Q) Define Data types / Records in PL/SQL?

PL/SQL Records: -

s Records are another type of datatypes which oracle allows to be defined as a placeholder. Records are composite datatypes, which means it is a combination of different scalar datatypes like char, Varchar, number etc. Each scalar data types in the record holds a value. A record can be visualized as a row of data. It can contain all the contents of a row.

Declaring a record: - To declare a record, you must first define a composite datatype; then declare a record for that type.

The General Syntax to define a composite datatype is: -

TYPE record_type_name IS RECORD (first_column_ name column_datatype, second_column_name column_ datatype, ……);

· record_type_name – it is the name of the composite type you want to define.

· first_col_name, second_col_name, etc.,- it is the names the fields/columns within the record.

· column_datatype defines the scalar datatype of the fields.

There are different ways you can declare the data type of the fields: -

1) You can declare the field in the same way as you declares the fields while creating the table.

2) If a field is based on a column from database table, you can define the field type as follows:

column_name table_name.column_name%type;

By declaring the field datatype in the above method, the datatype of the column is dynamically applied to the field. This method is useful when you are altering the column specification of the table, because you do not need to change the code again.

NOTE: You can use also %type to declare variables and constants. The General Syntax to declare a record of a user-defined datatype is:

record_name record_type_name;

The following code shows how to declare a record called employee_rec based on a user-defined type.

DECLARE TYPE employee_type IS RECORD (employee_id number (5), employee_first_name varchar2 (25), employee_last_name employee.last_ name%type, employee_dept employee.dept%type);

If all the fields of a record are based on the columns of a table, we can declare the record as follows:

record_name table_name%ROWTYPE;

For example, the above declaration of employee_rec can as follows:

DECLARE employee_rec employee%ROWTYPE; The advantages of declaring the record as a ROWTYPE are: -

1. You do not need to explicitly declare variables for all the columns in a table.

2. If you alter the column specification in the database table, you do not need to update the code.

The disadvantage of declaring the record as a ROWTYPE is: -

When u creates a record as a ROWTYPE, fields will be created for all the columns in the table and memory will be used to create the data type for all the fields. So use ROWTYPE

only when you are using all the columns of the table in the program.

NOTE: When you are creating a record, you are just creating a data type, similar to creating a variable. You need to assign values to the record to use them.

The following table consolidates the different ways in which you can define and declare a PL/SQL record.

Syntax	Usage
TYPE record_type_name IS RECORD (column_name1 datatype, column_name2 datatype, ...);	Define a composite datatype, where each field is scalar.
col_name table_name.column_name%type;	Dynamically define the datatype of a column based on a database column.
record_name record_type_name;	Declare a record based on a user-defined type.
record_name table_name%ROWTYPE;	Dynamically declare a record based on an entire row of a table. Each column in the table corresponds to a field in the record.

(Q) Explain the Control Statements in PL/SQL?

Branching: - Branching is a process in which the control will be jumped from one statement to another statement. In a program, they jumping may be based on condition or without condition.

Branching with condition is called conditional branching and the branching without condition called unconditional branching.

For conditional branching SQL provides the following statement.

1. If then
2. If then else
3. If then else if.

1. If Statement: - It is used test only one condition. It returns when if the condition is True. Otherwise the condition will exit. Every If statement should be ended with keyword "End if".

Syntax: - If < Condition > then

Statement 1;

Statement 2;

…….

End if;

Example: - If (num > 0) then
dbms_output.put_line („Positive Number || Num);

2. If then else: - It is used to test a condition if Condition is true then return True statement, otherwise its return s false statements.

Syntax: - If< condition > then
Statement 1;
Else
Statement 2;
Else if;

Example: - If (a> b) then
dbms_output.put_line(„a is big || a); else
dbms_output.put_line(„b is big || b); else if;

3. If then Else if: - This statement is used to test multiple conditions one by one sequentially.

Syntax: - if <condition 1> then
Statement 1;
Else if < condition 2 > then Statement 2;
Else if < condition 3 > then Statement 3;
…….
Else
Statement n;
End if;

In the above syntax if the condition 1 is True, statement 1 will executed. If condition 1 is false, it will go for checking condition 2 and so on.

Looping: - Looping is a process in which a block of statements will be executed repeatedly until given condition is True. PL/SQL provides two types of looping statements they are.

1. For Looping
2. While Looping.

1. For Looping: - This looping statement will execute loop body repeatedly initial values to final value for every

execution. The loop counter will be incremented by one. So by using this statement we can also find the number of iterations. This statement also called as "Recursive Looping" statement.

Syntax: - For < Initial value > in < range from> .. <final value>

```
Loop
…….
……          Loop Body.
…….
End Loop;
```

Reverse Values: For counter value in <reverse> <range from> .. <final value>

```
Loop
…….
……          Loop Body.
…….
End Loop;
```

Example: - (1) for I in 1..100

```
Loop
        dbms_output.put_line(i); End loop
```

Output: - Its prints 1 to 100 values

(2) for I in reverse 1..100

```
Loop
        dbms_output.put_line(i); End loop
```

Output: - its prints 100 to 1 values.

2. While Loop: - This looping statement can executed loop body based on a condition. This statement with check the condition first if the condition is true loop body will be executed. Otherwise loop will be terminated.

Syntax: - while <condition>

```
Loop
    Statement 1;
    Statement 2;    Loop Body
    …….
```

End loop;

Example: - while (i < 10)

Loop

dbms_output.put_line(i); i:= i+1;

End loop;

(Q) Discusses about the Cursors?

Cursors are a private working area or cursors are a temporary working area. "It is mainly used for retrieving multiple rows based on multiple attributes at a time. Cursors are very useful to update big structure of a table at a time". Cursors can calculate calculation according to given conditions and fetching rows into base tables.

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.

Cursors are mainly divided into two types in SQL. There are

1. Implicit Cursors.(user define)
2. Explicit Cursors.(system define)

1. Implicit Cursors: - These are automatically executed by SQL Engine Internally. (or) 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.

Ex: - Create Table, Alter Table, Create View, Update etc..,

2. Explicit Cursors: - The Cursors which are coded by database programmers are called Explicit Cursors. The Execution of explicit cursors is depending on the database programmer s code all the user crated cursors are 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.

Cursors Operations: - Every Explicit Cursors can be operated by using the following cursors operations.

1. Declaring cursors.
2. Opening a cursor
3. Fetching row in to cursor
4. Closing a cursor.

1. Declaring a Cursor: - Before using any cursor, that should be declared first. In one PL/SQL Program we may use many cursors; all cursors must be defined under declarative part of the PL/SQL block. A cursor is a database object, it is declared by using the following syntax.

Syntax: cursor <cursor name> is <select statement>;

Ex: cursor c1 is select * from Emp;

2. Opining a cursor: - In order to work with cursor, we must open it after creation. To open a cursor open key word is used.

Syntax: - Open cursor name;

Ex: - Open c1;

3. Syntax: - Fetch <cursor name> into host variables;

Fetching Row into cursor: - To calculate values of variables are useful after calculations. The values must be fetched to host variables. To do so fetching cursors is used.

Ex: - Fetch c1 into TA, DA and HRA;

Fetching must be performing to all rows successfully. So it must be included with in looping statement.

4. Closing Cursors: - After working successfully with a cursor, it must be closed when we close a cursor. It will release

all recourse. To close a cursors "close" key word is used.

Syntax:- Close <cursor name>;

Ex: - close c1;

Cursors Attributes: - Cursors Attributes are used to know the status of the cursor at a particular time. There are

1. % is open
2. % row count
3. % row type
4. % found
5. % not found

1. Syntax: - < cursors name> % is open ;

% is Open: - This attribute is used to know whether a cursor is opened or not. It returns true if the cursor is opened, otherwise it return false.

Ex: - c1 % is open ;

2. Syntax: - <cursor name> % row count ;

% Row count: - It provides the information about number of rows successfully fetched into table currently.

Ex: - c1 % row count;

3. % row type: - It can equate the data type of table attributes and host variable, this attribute always be used while defining a cursor.

Syntax: - <cursor name> % row type ;

Ex: - cursor c2 is select * from Emp c2 % row type;

4. Syntax: - <cursor name> % found ;

% Found: - It provides the Information about recent fetch operation. If the fetch statement is successfully completed it returns true otherwise its returns false.

Ex: - if c2 is % found then

Update emp set net sal:= TA+HRA+DA

5. Syntax:- <cursor name> % not found;

% not Found: - This attribute is used to find whether records are available or not in the cursor. Cursor operations can be performing until % not found is false.

Ex: - loop

Fetch c2 into TA,DA,BASIC and HRA Exit when c1% not found;

Net sal:= basic+TA+DA End loop;

Various Guidelines should follow while coding Cursors:-

It will improve performances and maintainability of the programs.

✓ Declare as many cursors as needed. There is no limit on the number of cursors that can used in programs.

✓ A void using certain cursors for modification. A cursor cannot be used for Updates or Deletes

✓ Cursors cannot be used for Updates or Deleted if the "Declare Cursor" statement includes a Union, Distinct, Group By and Having Clauses.

✓ Include only the columns that are being update.

✓ Always use "FOR UPDATE OF" when updating with a cursors.

✓ Use WHERE CURRENT OF to delete single rows using cursors.

✓ Open cursor before Fetching.

✓ Initialized host variables before opening the Cursor.

Close the Cursors explicitly. Even thought the RDBMS close all opens cursors at the end of program

Attributes	Return Value	Example
%FOUND	The return value is TRUE, if the DML statements like INSERT, DELETE and UPDATE affect at least one row and if SELECT ….INTO statement return at least one row.	SQL%FOUND
	The return value is FALSE, if DML statements like INSERT, DELETE and UPDATE do not affect row and if SELECT….INTO statement donot return a row.	
%NOTFOUND	The return value is FALSE, if DML statements like INSERT, DELETE and UPDATE at least one row and if SELECT ….INTO statement return at least one row.	SQL%NOTFOUND
	The return value is TRUE, if a DML statement likes INSERT, DELETE and UPDATE do not affect even one row and if SELECT ….INTO statement does not return a row.	
%ROWCOUNT	Return the number of rows affected by the DML operations INSERT, DELETE, UPDATE, SELECT	SQL%ROWCOUNT

Procedures & Functions.

(Q) Explain the SUB PROGRAM in PL/SQL?

A Sub Program is a named PLSQL blocks. Sub Programs are created by users. Every Sub Program can be stored in the data base. A „Stored Procedure or in simple a „Proc is a named PL/SQL block which performs one or more specific task.

This is similar to a procedure in other programming languages. A procedure has a header and a body. The header consists of the name of the procedure and the parameters or variables passed to the procedure. The body consists or declaration section, execution section and exception section similar to a general PL/SQL Block. A procedure is similar to an anonymous PL/SQL Block but it is named for repeated usage.

Advantages of sub program: -

· It is very easy to coding sub program and sub program are stored in the data base.

· So a sub program may be used by any number of times the same sub program may be shared by a number of users at a time.

· In the distributed environment when we invoke a sub program.

· Its code will be executed and a specific task will be performed.

· Sub program contains arguments are parameters.

Types of sub programs: - PLSQL sub programs are only divided into two types they are.

1. Procedures
2. Functions

1. Procedures: A procedure is a sub program which is stored on the data base. Generally a procedure contains one input parameter and display more output values. We can also input parameter and display more output values. We can also execute a procedure for different output variables.

2. Functions: A function is another type of sub program

is PLSQL. Functions always should return in a single value. Generally a function contains more input parameters and return only one output value.

Modes of Parameters: Parameters are the values passed to a function are procedure. Parameters also called arguments. The modes of arguments are of three types, they are...

1. In Mode / Parameter: It is used for only input purpose.
2. Out Mode / Parameter: It is used for only output purpose.
3. In Out Mode / Parameter: It is used for input and output purpose.

(Q). Explain the Procedures in PL/SQL?

Creation of Procedures: A procedure is a sub program which performs a specific action with in the data base. Procedures are used to manipulate the data with in data base. A procedure contains mainly two parts they are:

1. Declarative part / Header Section.
2. Body of the procedure / Body Section.

1. Declarative part: The declarative part is used for declare a procedure in the procedure declaration. The parameters also declared with a specific mode. The declarative part begins with a key word called "create", and ends with the last parameter.

2. Body of the procedure: The procedure body contains procedure code. This code is used to manipulate the data base. The body of the procedure begins with a key word "called" is and ends with a key word called "end". After creation of host variables the procedure should be execute. Execute a procedure the following syntax is used.

SQL> Exec procedure name (in parameter; out parameters); <enter>

PLSQL procedure successfully completed.

General Syntax to create a procedure is: -

CREATE [OR REPLACE] PROCEDURE proc_name [list of parameters]

IS
Declaration section BEGIN
Execution section EXCEPTION
Exception section END;

· By using CREATE OR REPLACE together the procedure is created if no other procedure with the same name exists or the existing procedure is replaced with the current code.

· The syntax within the brackets [ ] indicate they are optional.

· **IS -** marks the beginning of the body of the procedure and is similar to DECLARE in anonymous PL/SQL Blocks.

· The code between IS and BEGIN forms the Declaration section.

NOTE: - A procedure may or may not return any value.

The below example creates a procedure „employer_details which gives the details of the employee:

```
1> CREATE OR REPLACE PROCEDURE employer_
details
2> IS
3> CURSOR emp_cur IS
4> SELECT first_name, last_name, salary FROM
emp_tbl;
5> emp_rec emp_cur%rowtype;
6> BEGIN
7> FOR emp_rec in sales_cur 8> LOOP
9> dbms_output.put_line(emp_cur.first_name || ‘ ‘
||emp_cur.last_name
10>     || ‘ ‘ ||emp_cur.salary);
11> END LOOP;
12>END;
13> /
```

How to execute a Stored Procedure?

There are two ways to execute a procedure.

1) From the SQL prompt.

EXECUTE [or EXEC] procedure_name;

2) Within another procedure – simply use the procedure name. procedure_name;

NOTE: In the examples given above, we are using backward slash „/ at the end of the program. This indicates the oracle engine that the PL/SQL program has ended and it can begin processing the statements.

(Q). Explain the Functions in PL/SQL? Purpose: -

Use the CREATE FUNCTION statement to create a standalone stored function or a call specification.

A stored function (also called a user function or user defined function) is a set of PL/SQL statements you can call by name. Stored functions are very similar to procedures, except that a function returns a value to the environment in which it is called. User functions can be used as part of a SQL expression.

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.

A Function is a sub program; it is used to estimate a value. A Function can take multiple in parameters and returns only a single value. In parameters are enclosed with in parentheses followed by function name. The returning value is indicated with a key word called “return”. A function contains two parts, they are.

1. Declarative part / Header Section.
2. Body part

1. Declarative part: - In the Declarative part, a function is declared. Declarative part begins with a key word called create. Function declaration contains name of the function followed by input parameters. It also includes return data type of the function. A function declarative part ends with a key word called “returns”.

2. Body part: - Under the body part, the function coding takes place. A function body starts with a key word called "is" and ends with a key word called "end".

Syntax: SQL>create or replace function

Function name (in parameters list) Declarative part / Header Section. Return data type

Is

Output variable; Begin Statement1;

Statement2; Body part.

……..

Return (variable); End;

1. Return Type: The Header section defines the return type of the function. The return datatype can be any of the oracle datatype like Varchar, number etc.

2. The execution and exception section both should return a value which is of the datatype defined in the header section.

For example: - Let s create a function called "employer_details_func" similar to the one created in stored proc

```
1>  CREATE OR REPLACE FUNCTION employer_details_func
2>  RETURN VARCHAR(20);
3> IS
5> emp_name VARCHAR(20);
6> BEGIN
7> SELECT first_name INTO emp_name
8> FROM emp_tbl WHERE empID = '100';
9> RETURN emp_name;
10> END;
11> /
```

In the example we are retrieving the „first_name of employee with empID 100 to variable„emp_name . The return type of the function is VARCHAR which is declared in line no

2. The function returns the 'emp_name' which is of type VARCHAR as the return value in line no 9.

How to execute a PL/SQL Function: -

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;

If „employee_name is of datatype Varchar we can store the name of the employee by assigning the return type of the function to it.

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); This line displays the value returned by the function.

DIFFERENCES BETWEEN FUNCTIONS AND PROCEDURES:

PROCEDURES	FUNCTIONS
1. Procedures are used to perform some action of the data base.	1. Functions are used to estimate or calculate a value
2. It takes only one in parameter.	2. It takes one are more on parameters.
3. It contains more than one out parameters.	3. It contains only one out parameters.
4. It does not contains "return" key word.	4. Every function must contain return key word.
5. A procedures can store in data dictionary which is called users data sources.	5. A function also stored in data dictionary, Which is called users data sources.
6. To procedures name should not be same.	6. To functions name should not be same.

Q Create function in PLSQL to add two numbers.

```
SQL> Create or replace function
        Add_two (a number, b number) Return number;
        Is=
        C number; Begin
        C:= a + b;
        Return©; End;
SQL> / <enter>
```

Function created

SQL> var sum number; <enter> SQL> Erec

is:=add=two(2,5); <enter>

PLSQL procedures successfully completed. SQL> Select add_two (5A) from dval;

Add_two (5A);

(Q) How to pass parameters to Procedures and Functions in PL/SQL?

In PL/SQL, we can pass parameters to procedures and functions in three ways.

1) IN type parameter: These types of parameters are used to send values to stored procedures.

2) OUT type parameter: These types of parameters are used to get values from stored procedures. This is similar to a return type in functions.

3) IN OUT parameter: These types of parameters are used to send values and get values from stored Procedures.

NOTE: If a parameter is not explicitly defined a parameter type, then by default it is an IN type parameter.

1. IN parameter: - This is similar to passing parameters in programming languages. We can pass values to the stored procedure through these parameters or variables. This type of parameter is a read only parameter. We can assign the value of IN type parameter to a variable or use it in a query, but we cannot change its value inside the procedure.

The General syntax to pass an IN parameter is: -

CREATE [OR REPLACE] PROCEDURE procedure_name (param_name1 IN datatype, param_name12 IN datatype ...)

· param_name1, param_name2... are unique parameter names.

· Datatype - defines the datatype of the variable.

· IN - is optional, by default it is an IN type parameter.

2. OUT Parameter: - The OUT parameters are used to send the OUTPUT from a procedure or a function. This is a write-only parameter i.e., we cannot pass values to OUT parameters while executing the stored procedure, but we can

assign values to OUT parameter inside the stored procedure and the calling program can receive this output value.

The General syntax to create an OUT parameter is: -

CREATE [OR REPLACE] PROCEDURE proc2 (param_name OUT datatype)

The parameter should be explicitly declared as OUT parameter.

3. IN OUT Parameter: - The IN OUT parameter allows us to pass values into a procedure and get output values from the procedure. This parameter is used if the value of the IN parameter can be changed in the calling program.

By using IN OUT parameter we can pass values into a parameter and return a value to the calling program using the same parameter. But this is possible only if the value passed to the procedure and output value have a same datatype. This parameter is used if the value of the parameter will be changed in the procedure.

The General syntax to create an IN OUT parameter is: -

CREATE [OR REPLACE] PROCEDURE proc3 (param_name IN OUT datatype)

The below examples show how to create stored procedures using the above three types of parameters.

Example1: -

Using IN and OUT parameter: - Let s create a procedure which gets the name of the employee when the employee id is passed.

```
1> CREATE OR REPLACE PROCEDURE emp_name (id
IN NUMBER, emp_name OUT NUMBER)
2> IS
3> BEGIN
4> SELECT first_name INTO emp_name
5> FROM emp_tbl WHERE empID = id;
6> END;
7> /
```

We can call the procedure „emp_name in this way from a PL/SQL Block.

```
1> DECLARE
2> empName varchar(20);
3> CURSOR id_cur SELECT id FROM emp_ids;
4> BEGIN
5> FOR emp_rec in id_cur 6> LOOP
7> emp_name(emp_rec.id, empName);
8> dbms_output.putline('The employee ' || empName
|| ' has id ' || emp-rec.id); 9> END LOOP;
10> END;
11> /
```

In the above PL/SQL Block In line no 3; we are creating a Cursor „id_cur which contains the employee id. In line no 7; we are creating the procedure „emp_name , we are passing the „id as IN parameter and „empName . In line no 8; we are displaying the id and the employee name which we got from the procedure „emp_name .

Example 2: -

Using IN OUT parameter in procedures: -

```
1> CREATE OR REPLACE PROCEDURE emp_salary_
increase
2> (emp_id IN emptbl.empID%type, salary_inc IN OUT
emptbl.salary%type)
3> IS
4> tmp_sal number;
5> BEGIN
6> SELECT salary
7> INTO tmp_sal
8> FROM emp_tbl
9> WHERE empID = emp_id;
10> IF tmp_sal between 10000 and 20000 THEN
11> salary_inout := tmp_sal * 1.2;
12> ELSIF tmp_sal between 20000 and 30000 THEN
```

```
13> salary_inout := tmp_sal * 1.3;
14> ELSIF tmp_sal > 30000 THEN
15> salary_inout := tmp_sal * 1.4;
16> END IF;
17> END;
18> /
```

The below PL/SQL block shows how to execute the above 'emp_salary_increase' procedure.

```
1> DECLARE
2> CURSOR updated_sal is
3> SELECT empID,salary
4> FROM emp_tbl;
5> pre_sal number;
6> BEGIN
7> FOR emp_rec IN updated_sal LOOP
8>pre_sal := emp_rec.salary;
9> emp_salary_increase(emp_rec.empID, emp_rec.
salary);
10> dbms_output.put_line('The salary of ' ||
emp_rec.empID ||
11>'increased from'||pre_sal||'to '||emp_rec.salary);
12> END LOOP;
13> END;
14> /
```

(Q) Discuss the Exception Handling in PL/SQL?

Exception Handling: -

PL/SQL provides a feature to handle the Exceptions which occur in a PL/SQL Block known as exception Handling. Using Exception Handling we can test the code and avoid it from exiting abruptly. When an exception occurs messages which explains its cause is received.

Parts of Exceptions: - PL/SQL Exception message consists of three parts.

1. Type of Exception.
2. An Error Code.
3. A message.

By Handling the exceptions we can ensure a PL/SQL block does not exit abruptly.

Structure of Exception Handling: - The General Syntax for coding the exception section.

```
DECLARE
        Declaration section BEGIN
        Exception section EXCEPTION
        WHEN ex_name1 THEN
        -Error handling statements WHEN ex_name2 THEN
        -Error handling statements WHEN Others THEN
        -Error handling statements END;
```

General PL/SQL statements can be used in the Exception Block. When an exception is raised, Oracle searches for an appropriate exception handler in the exception section. For example in the above example, if the error raised is 'ex_name1 ', then the error is handled according to the statements under it. Since, it is not possible to determine all the possible runtime errors during testing for the code, the 'WHEN Others' exception is used to manage the exceptions that are not explicitly handled. Only one exception can be raised in a Block and the control does not return to the Execution Section after the error is handled.

If there are nested PL/SQL blocks like this.

```
DELCARE
        Declaration section BEGIN
DECLARE
        Declaration section BEGIN
        Execution section EXCEPTION
        Exception section END;
EXCEPTION
        Exception section END;
```

In the above case, if the exception is raised in the inner block it should be handled in the exception block of the inner PL/SQL block else the control moves to the Exception block of the next upper PL/SQL Block. If none of the blocks handle the exception the program ends abruptly with an error.

Types of Exception: - There are 3 types of Exceptions

a) Named System Exceptions.

b) Unnamed System Exceptions.

c) User-defined Exceptions.

a) Named System Exceptions: - System exceptions are automatically raised by Oracle, when a program violates a RDBMS rule. There are some system exceptions which are raised frequently, so they are pre- defined and given a name in Oracle which is known as Named System Exceptions.

For example: NO_DATA_FOUND and ZERO_DIVIDE are called Named System exceptions. Named system exceptions are:

1) Not Declared explicitly.

2) Raised implicitly when a predefined Oracle error occurs.

3) caught by referencing the standard name within an exception-handling routine.

Exception Name	Reason	Error Number
CURSOR_ALREADY_OPEN	When you open a cursor that is already open.	ORA-06511
INVALID_CURSOR	When you perform an invalid operation on a cursor like closing a cursor, fetch data from a cursor that is not opened.	ORA-01001
NO_DATA_FOUND	When a SELECT...INTO clause does not return any row from a table.	ORA-01403
TOO_MANY_ROWS	When you SELECT or fetch more than one row into a recordor variable.	ORA-01422
ZERO_DIVIDE	When you attempt to divide a number by zero.	ORA-01476

For Example: Suppose a NO_DATA_FOUND exception is raised in a proc, we can write a code to handle the exception as given below.

```
BEGIN
        Execution section EXCEPTION
        WHEN NO_DATA_FOUND THEN
        dbms_output.put_line ('A SELECT...INTO did not return
```

any row.'); END;

b) Unnamed System Exceptions: - Those system exception for which oracle does not provide a name is known as unnamed system exception. These exceptions do not occur frequently. These Exceptions have a code and an associated message.

There are two ways to handle unnamed system exceptions:

1. By using the WHEN OTHERS exception handler, or
2. By associating the exception code to a name and using it as a named exception.

We can assign a name to unnamed system exceptions using a **Pragma** called **EXCEPTION_INIT. EXCEPTION_INIT** will associate a predefined Oracle error number to a programmer defined exception name.

Steps to be followed to use unnamed system exceptions are

- They are raised implicitly.
- If they are not handled in WHEN Others they must be handled explicitly.
- To handle the exception explicitly, they must be declared using Pragma EXCEPTION_INIT as given above and handled referencing the user-defined exception name in the exception section.

The general syntax to declare unnamed system exception using EXCEPTION_INIT is: -

DECLARE

exception_name EXCEPTION; PRAGMA
EXCEPTION_INIT (exception_name, Err_code); BEGIN
Execution section EXCEPTION
WHEN exception_name THEN handle the exception
END;

For Example: Lets consider the product table and order_items table from SQL joins.

Here product_id is a primary key in product table and a foreign key in order_items table. If we try to delete a product_id from the product table when it has child records in order_id table an exception will be thrown with oracle code number -2292. We can provide a name to this exception and handle it in the exception section as given below:

```
DECLARE
        Child_rec_exception EXCEPTION; PRAGMA
        EXCEPTION_INIT (Child_rec_exception, -2292); BEGIN
        Delete FROM product where product_id= 104;
EXCEPTION
        WHEN Child_rec_exception
        THEN Dbms_output.put_line('Child records are present for this product_id.'); END;/
```

c) User-defined Exceptions: - Apart from system exceptions we can explicitly define exceptions based on business rules. These are known as user-defined exceptions. Steps to be followed to use user-defined exceptions:

- They should be explicitly declared in the declaration section.
- They should be explicitly raised in the Execution Section.
- They should be handled by referencing the user-defined exception name in the exception section.

Advantages of using Procedures or Functions: -

1. Security: - Stored Procedure and Functions can help enforce data security.

2. Performance: - It improves the database performances in the following way:

- Amount of Information sent over a Network is less.
- No Compilation step is required to execute the code.

3. Memory Allocation: - The amount of memory used reduces as stored procedures or functions have shared memory capability. Only one copy of procedures needs to be

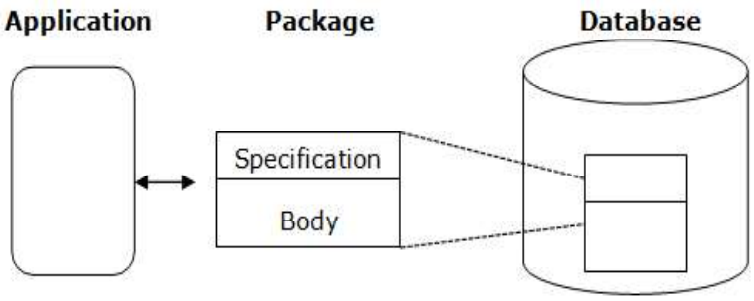
loaded for execution by multiple parameters.

4. Productivity: - By written procedures and functions redundant coding can be avoided increasing.

(Q). Explain the concepts of Packages in PL/SQL? Package:

A package is a schema object that groups logically related PL/SQL types, items, and subprograms. Packages usually have two parts, a specification and a body, although sometimes the body is unnecessary. The specification (spec for short) is the interface to your applications; it declares the types, variables, constants, exceptions, cursors, and subprograms available for use. The body fully defines cursors and subprograms, and so implements the spec.

Package Interface: -



To create packages, use the CREATE PACKAGE statement, which you can execute interactively from

SQL*Plus. Here is the syntax: -

CREATE [OR REPLACE] PACKAGE package_name [AUTHID {CURRENT_USER | DEFINER}]

{IS | AS}

[PRAGMA SERIALLY_REUSABLE;]

[collection_type_definition ...] [record_type_definition ...] [subtype_definition ...] [collection_declaration ...] [constant_declaration ...] [exception_declaration ...] [object_declaration ...] [record_declaration ...]

[variable_declaration ...] [cursor_spec ...] [function_spec ...]
[procedure_spec ...] [call_spec ...]
[PRAGMA RESTRICT_REFERENCES (assertions) ...]
END [package_name];
[CREATE [OR REPLACE] PACKAGE BODY package_name {IS | AS} [PRAGMA SERIALLY_REUSABLE;]
[collection_type_definition ...] [record_type_definition ...]
[subtype_definition ...] [collection_declaration ...]
[constant_declaration ...] [exception_declaration ...]
[object_declaration ...] [record_declaration ...]
[variable_declaration ...] [cursor_body ...] [function_spec ...]
[procedure_spec ...] [call_spec ...]
[BEGIN
sequence_of_statements] END [package_name];]

Public Declaration: -The spec holds public declarations, which are visible to your application. You must declare subprograms at the end of the spec after all other items (except Pragma that name a specific function; such Pragma must follow the function spec).

Private Declarations: - The body holds implementation details and private declarations, which are hidden from your application. Following the declarative part of the package body is the optional initialization part, which typically holds statements that initialize package variables.

The AUTHID clause determines whether all the packaged subprograms execute with the privileges of their definer (the default) or invoker, and whether their unqualified references to schema objects are resolved in the schema of the definer or invoker.

Example of a PL/SQL Package: -

In the example below, you package a record type, a cursor, and two employment procedures. Notice that the procedure hire_employee uses the database sequence empno_seq and the function SYSDATE to insert a new

employee number and hire date, respectively.

CREATE OR REPLACE PACKAGE emp_actions AS — spec TYPE EmpRecTyp IS RECORD (emp_id INT, salary REAL); CURSOR desc_salary RETURN EmpRecTyp;

PROCEDURE hire_employee (ename VARCHAR2,
job VARCHAR2,
mgr NUMBER,
sal NUMBER, comm NUMBER, deptno NUMBER);

PROCEDURE fire_employee (emp_id NUMBER); END emp_actions;

CREATE OR REPLACE PACKAGE BODY emp_actions AS — body CURSOR desc_salary RETURN EmpRecTyp IS

SELECT empno, sal FROM emp ORDER BY sal DESC; PROCEDURE hire_employee (
ename VARCHAR2, job VARCHAR2,
mgr NUMBER,
sal NUMBER, comm NUMBER, deptno NUMBER) IS
BEGIN

INSERT INTO emp VALUES (empno_seq.NEXTVAL, ename, job, mgr, SYSDATE, sal, comm, deptno);

END hire_employee;

PROCEDURE fire_employee (emp_id NUMBER) IS BEGIN

DELETE FROM emp WHERE empno = emp_id; END fire_employee;

END emp_actions;

Only the declarations in the package spec are visible and accessible to applications. Implementation details in the package body are hidden and inaccessible. So, you can change the body (implementation) without having to recompile calling programs.

Advantages of PL/SQL Packages: -

Packages offer several advantages: Modularity, Easier

Application Design, Information Hiding, Added Functionality, and Better Performance.

1. Modularity: - Packages let you encapsulate logically related types, items, and subprograms in a named PL/SQL module. Each package is easy to understand, and the interfaces between packages are simple, clear, and well defined. This aids application development.

2. Easier Application Design: - When designing an application, all you need initially is the interface information in the package specs. You can code and compile a spec without its body. Then, stored subprograms that reference the package can be compiled as well. You need not define the package bodies fully until you are ready to complete the application.

3. Information Hiding: - With packages, you can specify which types, items, and subprograms are public (visible and accessible) or private (hidden and inaccessible). For example, if a package contains four subprograms, three might be public and one private. The package hides the implementation of the private subprogram so that only the package (not your application) is affected if the implementation changes. This simplifies maintenance and enhancement. Also, by hiding implementation details from users, you protect the integrity of the package.

4. Added Functionality: - Packaged public variables and cursors persist for the duration of a session. So, they can be shared by all subprograms that execute in the environment. Also, they allow you to maintain data across transactions without having to store it in the database.

5. Better Performance: - When you call a packaged subprogram for the first time, the whole package is loaded into memory. So, later calls to related subprograms in the package require no disk I/O. Also, packages stop cascading dependencies and thereby avoid unnecessary recompiling. For example, if you change the implementation of a packaged

function, Oracle need not recompile the calling subprograms because they do not depend on the package body.

Triggers

(Q) What is Triggers? Explain how to create Triggers? How many types is their and also explain The advantages and disadvantages of Triggers?

Trigger and Routines is very useful database object. Because they are stored in the database and controlled by the RDBMS. Thus the code required. Both Trigger & Routines consisted of procedure code.

Triggers: - A Trigger is a stored procedure that implicitly executed, when an Insert, Update or Delete is issued against the associated table. We can make a Trigger to Fire (Executed) only for DML statements (Insert, Update, Deleted).

"If a condition started with in the trigger is met, then a prescribed action is taken". The Trigger code is stored in the database and runs automatically when ever the Triggering event such as an update occurs. Since Triggers are stored and executed in the database. They execute against all applications that access the database. We can delete the Trigger by using Drop command.

Uses of Trigger: - Database Trigger can be used for following purpose.

1. To generate data automatically
2. To audit data modification
3. To enforce Integrity Constrains. (Maintain accuracy of DB)

Parts of Triggers:- There are mainly 3 parts.

1. Trigger Statement (Event)
2. Trigger Restriction (Conditions)
3. Trigger Body (Action)

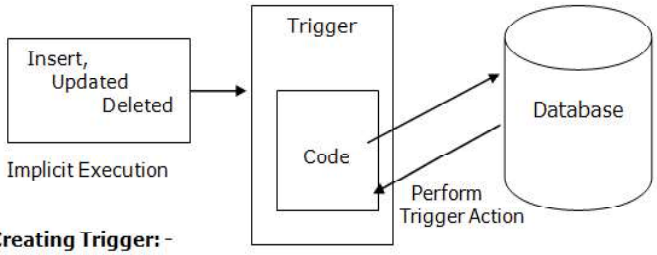
1. Trigger Statement: - The Trigger statement specified DML statement like Insert, Update, Delete and it Fires the Trigger body(action).

2. Trigger Restriction: - The Restriction is specified using WHEN Clause.

3. Trigger Body: - The Trigger body is a PL/SQL block that is executed when a Trigger statement is issued.

Trigger
Database
Code
Perform Trigger Action
Insert,
Updated
Deleted

These parts are reelected in the coding structure of triggers. Triggers are associated with tables and run transparently when pre-define events happen to the data in the table.



Syntax for Creating Trigger: -

SQL> CREATE [OR REPLACE] TRIGGER trigger_name {BEFORE/ AFTER}

{INSERT / DELETE/ UPDATE} ON table_name

[FOR EACH ROW/ STATEMENT [WHEN (trigger_ name)]] <condition> DECLARE

Declaration of Variable/ constants; BEGIN

Body of the Statement; EXCEPTION

Exception of PL/SQL Block; END;

Types of Trigger: - Triggers are mainly divided into two parts based on usage. There are

1. Before / After

2. For each Row / Statement.

1. Before / After: - The Before / After options can be used to specify when the trigger body should be fire with respect to trigger statement.

- If Before option is used then Oracle Engine Fires the Trigger before executing the statement.
- If After option is used then Oracle Engine Fires the Trigger After executing the Trigger statement.

2. for each Row/Statement: -

· **Statement Level Trigger:-** A Trigger which is created only for one record are called "Statement Level Trigger". In practice these are not used frequently by default. Every Trigger is statement level Trigger.

· **Row Level Trigger:-** The Trigger which are created on the total table is know as "Row Level Trigger" by using Row Level Trigger. We can make a Trigger to fire on selected columns for more than once. When working with Row Level Triggers. We can use two qualifiers called "Old" & "New". The most commonly used Triggers are Row Level Triggers.

Example: - You may use a Database Trigger to log the fact that an account receivable clerk has lowered the amount owing on an outstanding account.

Ex: - Create or replace trigger T1 before insert On bank for each row

```
Begin
dbms_output.put_line("Trigger Fired");
End;
```

Ex: - create or replace trigger T2 before update On bank for each row

```
Begin
Update bank set balance= 1000;
dbms_output.put_line("Update");
End;
```

Enabling & Disabling Triggers: - Triggers don t affect all rows in the table. They affect only Transactions of the specified type. When the Triggers Enable. The Triggers will not effect any transactions created before to a Trigger by default. Triggers are enabled when it is created, but some times we may need to disable Triggers disabling Triggers during data loads improve the performances of the data load. In such a way the data load partially succeeded and the Triggers was executed for a portion of the data load records. It is also possible to since a trigger twice for the same transaction. To enable a trigger "Alter Trigger" command is used with the "enable" key word.

Syntax for Enable / Disable Trigger: -

SQL> Alter table <table name> enable / disable all triggers;

Ex: - SQL> Alter table bank disable all trigger;

Dropping Trigger: - If there are any unwanted Triggers in the data base. They must be removed from that the database, it is also called Dropping a Trigger. To drop any Trigger the following syntax can be used.

Syntax: - drop trigger <trigger name>;

Ex:- SQL> drop trigger bank_person;

Trigger dropped.

Advantages of Triggers: -

Triggers can be used to makes the RDBMS to take some action. When a database related event has occurred the following advantages of Triggers.

- The Business rules can be stored in the database consistently with each and every updated operation.
- The reduce complexity of the application program that the database.

Disadvantages of Trigger: -

- When the business rules are moved in to the database, setting up, the database becomes a more complex task.

➢ With the triggers, the business rules are hidden in the database and the application program can cause an enormous (plenty) amount of database activity.

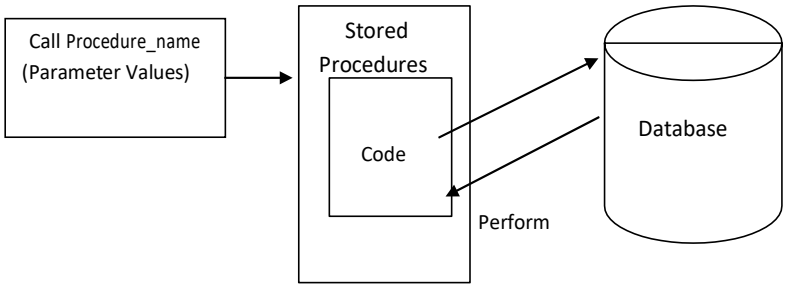
(Q) What is Routines? Explain how to create Routines it s advantages of Routines?

Routines: - SQL invoked Routines can be either Procedure or Functions. Routines are Stored Blocks of code that have to be called in order to operate. Routines don t run automatically.

➢ **Functions: -** A Functions returns one value and has only Input Parameters.

➢ **Procedure: -** A collection of Procedures and SQL statements that are assigned a Unique name with in the schema and stored in the database. When it is desired to run the procedures it is called by its name. When called, all the statements in the procedures will be executed.

A Procedure may have Input Parameters, Output Parameters and Parameters that are both Input & Output parameters.



Create Routine Syntax: -

SQL> { CREATE PROCEDURE / CREATE FUNCTION } Routine_Name ([Parameter [{, Parameter …..]])

[RETURNS data type result_Case] / * for Function only * /

[LANGUAGE { ADA / C / COBOL / FORTRAN / PASCAL/ PLI/ SQL }]

[PARAMETER STYLE { SQL / GENERAL }]

[SPECIFIC Specific_Name]

[DETERMINISTIC / NOT DETERMINISTIC][NO SQL / CONTAINS SQL / READS SQL DATA / MODIFIES / SQL DATA]

[RETURN NULL ON NULL INPUT / CALL ON NULL INPUT]

[DYNAMIC RESULT SETS Unsigned_Integer] / * for Procedures only * /

[STATIC DISPATCH] / * for Function only */

Routine_Body End;

Advantages of Routines: -

There are following advantages of the Routines

- ❖ Flexibility
- ❖ Efficiency
- ❖ Sharability
- ❖ Applicability

1. Flexibility: - Routines may be used in more situations that constraints or triggers, which are limited to data modification circumstances. Just as Triggers have more code options those constraints, routines have more code options than Triggers.

2. Efficiency: - Routines can be careful crafted and optimized to run more quickly than slower, generic SQL Statements.

3. Sharability: - Routines may be cached on the server and made available to all users, so that they do not have to be rewritten.

4. Applicability: -Routines may apply to the entire database rather than being kept with one application. This advantages is a corollary to sharability.

Differences between Triggers & Routines:

Triggers	Routines
Triggers are database Objects	Routines also database Object
Triggers are Stored in database	Routines are also Stored in database
Triggers are Controlled by RDBMS	Routines are also Controlled by RDBMS
Triggers are Run Automatically	Routines are don?t Run Automatically
Triggers are Implicit execution	Routines are Explicitly execution
Triggers has 3 parts namely 1. Event. 2. Condition. 3. Action	Routines can be either Procedure or Functions.
Without name Trigger can Run Automatically	Run the Procedures it is called by its Name.