



**The Motto of Our University
(SEWA)**

SKILL ENHANCEMENT

EMPLOYABILITY

WISDOM

ACCESSIBILITY

SELF-INSTRUCTIONAL STUDY MATERIAL FOR JGND PSOU

ALL COPYRIGHTS WITH JGND PSOU, PATIALA

**JAGAT GURU NANAK DEV
PUNJAB STATE OPEN UNIVERSITY, PATIALA**

(Established by Act No. 19 of 2019 of the Legislature of State of Punjab)

B.Sc.(Data Science)

Semester III

BSDB32301T

Database Management System

Head Quarter: C/28, The Lower Mall, Patiala-147001

Website: www.psou.ac.in

The Study Material has been prepared exclusively under the guidance of Jagat Guru Nanak Dev Punjab State Open University, Patiala, as per the syllabi prepared by Committee of Experts and approved by the Academic Council.

The University reserves all the copyrights of the study material. No part of this publication may be reproduced or transmitted in any form.

COURSE COORDINATOR AND EDITOR:

Dr. Amitoj Singh

Associate Professor

School of Sciences and Emerging Technologies

Jagat Guru Nanak Dev Punjab State Open University

LIST OF CONSULTANTS/ CONTRIBUTORS

Sr. No.	Name
1	Dr. Sachin Ahuja
2	Dr. Rachpal Singh



JAGAT GURU NANAK DEV PUNJAB STATE OPEN UNIVERSITY, PATIALA
(Established by Act No. 19 of 2019 of the Legislature of State of Punjab)

PREFACE

Jagat Guru Nanak Dev Punjab State Open University, Patiala was established in December 2019 by Act 19 of the Legislature of State of Punjab. It is the first and only Open University of the State, entrusted with the responsibility of making higher education accessible to all, especially to those sections of society who do not have the means, time or opportunity to pursue regular education.

In keeping with the nature of an Open University, this University provides a flexible education system to suit every need. The time given to complete a programme is double the duration of a regular mode programme. Well-designed study material has been prepared in consultation with experts in their respective fields.

The University offers programmes which have been designed to provide relevant, skill-based and employability-enhancing education. The study material provided in this booklet is self-instructional, with self-assessment exercises, and recommendations for further readings. The syllabus has been divided in sections, and provided as units for simplification.

The University has a network of 10 Learner Support Centres/Study Centres, to enable students to make use of reading facilities, and for curriculum-based counselling and practicals. We, at the University, welcome you to be a part of this institution of knowledge.

Prof. Anita Gill
Dean Academic Affairs



B.Sc. (Data Science)
Core Course (CC)
Semester III
BSDB32301T: Data Base Management System

Total Marks: 100
External Marks: 70
Internal Marks: 30
Credits: 4
Pass Percentage: 35%

Objectives

This course explains fundamental elements of relational database management systems and made student familiar with the basic concepts of relational data model, entity-relationship model, relational database design, relational algebra and SQL

Section A

Unit I: DBMS and its Architecture - Overview of DBMS, Basic DBMS terminology, Data independence. Architecture of a DBMS, Disadvantages of Traditional DBMS, Advantages and Characteristics of DBMS.

Unit II: Data Models –Relational Keys: Primary Key, Foreign Key, Candidate Key, Super Key etc., and Integrity Constraints, Relational model, Relational schema Hierarchical model, and Network model.

Unit III: Conceptual Data Modeling using E-R Data Model -Entities, attributes, relationships, generalization, specialization, specifying constraints, Conversion of ER Models to Tables, Practical problems based on E-R data model.

Unit IV: Normal Forms - Functional Dependency, Multi valued dependencies and Joined dependencies, 1NF, 2NF, 3NF, BCNF, 4NF, 5NF.

Section B

Unit V: Structured Query Language - Introduction to SQL, data types, DDL, DML, DCL, querying database tables, Data Definition Language (DDL), Creating Tables, Inserting and updating values into a Table.

Unit VI: Data Manipulation Language: Various form of SELECT- simple, using special operators, aggregate functions, group by clause, sub query, joins, co-related sub query, union clause, exist operator, Aggregate Functions.

Unit VII: Views- Introduction to views, data independence, Statements on Join Views, Dropping a VIEW. Database security, Security Techniques, Two Phase Locking Techniques.

Unit VIII: Data Control Operations - GRANT command, REVOKE command, COMMIT and ROLLBACK. Concurrency Control Techniques, Recovery Control techniques.

Suggested Readings

1. Silverschatz A., Korth F. H. and Sudarshan S., Database System Concepts, Tata McGraw Hill 6th ed., 2019
2. Elmasri R. and Navathe B. S., Fundamentals of Database Systems, Pearson 7th ed, 2016
3. Bayross I., SQL, PL/SQL the Programming Language of Oracle, BPB Publications 4th Ed., 2009
4. Vikram Vaswani., MySQL(TM): The Complete Reference, McGraw Hill Education, 2017
5. Robin Nixon, Learning Php, Mysql & Javascript Paperback (English) 4th Edition, O'Reilly Media, Inc., 2014



JAGAT GURU NANAK DEV PUNJAB STATE OPEN UNIVERSITY, PATIALA
(Established by Act No. 19 of 2019 of the Legislature of State of Punjab)

BSDB32301T: DATA BASE MANAGEMENT SYSTEM
COURSE COORDINATOR AND EDITOR: DR. AMITIJ SINGH

UNIT NO.	UNIT NAME
UNIT 1	DBMS AND ITS ARCHITECTURE
UNIT 2	DATA MODELS –RELATIONAL KEYS
UNIT 3	CONCEPTUAL DATA MODELING USING E-R DATA MODEL
UNIT 4	NORMAL FORMS
UNIT 5	STRUCTURED QUERY LANGUAGE
UNIT 6	DATA MANIPULATION LANGUAGE
UNIT 7	VIEWS
UNIT 8	DATA CONTROL OPERATIONS

B.Sc.(DATA SCIENCE)

SEMESTER-III

DATABASE MANAGEMENT SYSTEM

UNIT I: INTRODUCTION OF DBMS

STRUCTURE

1.0 Objectives

1.1 Introduction

1.2 Database Management System- Overview

1.2.1 What is a Database?

1.2.1.1 History of Database Systems

1.2.1.2 The Evolution of Database systems

1.2.1.3 Types of Databases

1.2.2 What is a Management System?

1.3 Basic terminology of DBMS

1.3.1 Goals of DBMS

1.3.2 Properties of DBMS

1.3.3 Need of DBMS

1.3.4 Features of DBMS

1.3.5 Users of DBMS

1.3.6 Levels of abstraction in a DBMS

1.4 Data Independence

1.4.1 Physical Data Independence

1.4.2 Logical Data Independence

1.5 Architecture of DBMS

1.5.1 Types of DBMS Architecture

1.5.2 Three Schema Architecture

1.6 Traditional File System VS DBMS

1.6.1 Traditional File system

1.6.2 Advantages of Traditional File System

1.6.3 Disadvantages of Traditional File System

1.6.4 Differences between Traditional File System & DBMS

1.7 Advantages of DBMS

1.8 Disadvantages of DBMS

1.9 Characteristics of DBMS

1.10 Applications of DBMS

1.11 Summary

1.12 References

1.13 Further Readings

1.14 Questions for Practice(Short & Long Answers Type)

1.0 OBJECTIVES

- To provide overview of Database Management System.
- To elaborate various types of databases.
- To explain basic DBMS terminology.
- To describe features of DBMS.
- To discuss various levels of abstraction in DBMS.
- To provide description of Data Independence.
- To explain architecture of DBMS.
- To discuss advantages and disadvantages of traditional file system.
- To differentiate traditional file system from modern DBMS.
- To describe advantages and disadvantages of DBMS.
- To describe characteristics as well as applications of DBMS.

1.1 INTRODUCTION

Database Management System is a collection of programs that allow you to create and maintain databases. It can also be defined as a software system that allows you to define, construct and manipulate databases used for various purposes such as to store customer and financial information of an organisation. In this unit, we will explain the basics of DBMS such as its Overview, Basic DBMS terminology, various terms of data independence and architecture of a DBMS. In addition, we will also discuss the disadvantages of traditional DBMS which will be followed by the advantages and characteristics of DBMS.

1.2 DATABASE MANAGEMENT SYSTEM-OVERVIEW

Database Management System or DBMS refers to the technology of storing and retrieving data of users with utmost efficiency along with appropriate security measures. As the name suggests, the database management system consists of two parts. They are:

- Database and
- Management System

1.2.1 What is a Database?

To find out what database is, we have to start from data along with a few other terms, which are the basic building block of any DBMS.

Data: It consists of facts, figures, statistics etc. having no particular meaning (e.g. 1, XYZ, 20 etc).

Record: It is a collection of related data items. For example, the three data items specified in the above data had no meaning. But if we organize them in the following way, then they collectively represent meaningful information.

Roll No.	Name	Age
1	XYZ	20

Thus, Here is the difference between data and record or information.

DATA	INFORMATION
1. It contains raw facts	1. It contains processed data
2. It is in unorganized form	2. It is in organized form
3. Data doesn't help in decision making process.	3. Information helps in decision making process.

Table/Relation: It is a collection of related records.

Roll No.	Name	Age
1	XYZ	20
2	ABC	26
3	EFG	30

The columns of this relation are called Fields, Attributes or Domains.

The rows are called Tuples or Records.

Database: It is a collection of related relations. Consider the following collection of tables:

T1

Roll No.	Name	Age
1	XYZ	20
2	ABC	26
3	EFG	30

T2

Roll No.	Address
1	IJK
2	LMN
3	QRS

T3

Roll No.	Year
1	I
2	II
3	I

T4

Year	Hostel
I	H1
II	H2

We now have a collection of 4 tables named as T1,T2,T3 and T4. They can be called a “related collection” because we can clearly find out that there are some common attributes existing in a selected pair of tables. Because of these common attributes we may combine the data of two or more tables together to find out the complete details of a student. Questions like “Which hostel does the youngest student live in?” can be answered now, although Age and Hostel attributes are in different tables.

Thus, database is a collection of inter-related data which is used to retrieve, insert and delete the data efficiently. It is also used to organize the data in the form of a table, schema, views, and reports, etc.

1.2.1.1 History of Database Systems

I. 1950’s and early 1960’s:

- Magnetic tapes were developed for data storage
- Data processing tasks such as payroll were automated, with data stored on tapes.
- Data could also be input from punched card decks, and output to printers.

II. Late 1960’s and 1970’s:

- The use of hard disks in the late 1960s changed the scenario for data processing greatly, since hard disks allowed direct access to data.
- With disks, network and hierarchical databases could be created that allowed data structures such as lists and trees to be stored on disk. Programmers could construct and manipulate these data structures.
- In the 1970’s , EF CODD defined the Relational Model.

III. In the 1980’s:

- Initial commercial relational database systems, such as IBM DB2, Oracle, Ingress, and DEC Rdb, played a major role in advancing techniques for efficient processing of declarative queries.
- Relational databases had become competitive with network and hierarchical database systems even in the area of performance.
- The 1980s also saw much research on parallel and distributed databases, as well as initial work on object-oriented databases.

IV. Early 1990’s:

- The SQL language was designed primarily in the 1990's and this is used for the transaction processing applications.
- Decision support and querying re-emerged as a major application area for databases.
- Database vendors also began to add object-relational support to their databases.

V. **Late 1990's:**

- The major event was the explosive growth of the World Wide Web.
- Databases were deployed much more extensively than ever before.
- Database systems now had to support very high transaction processing rates, as well as very high reliability and 24 * 7 availability (availability 24 hours a day, 7 days a week, meaning no downtime for scheduled maintenance activities).
- Database systems also had to support web interfaces to data.

1.2.1.2 **The Evolution of Database systems**

From pre-stage flat-file system, to relational and object-relational systems, database technology has gone through several generations and its history that is spread over more than 40 years now. The Evolution of Database systems is as follows:

- I. File Management System
- II. Hierarchical database System
- III. Network Database System
- IV. Relational Database System
- V. Object-Oriented Database System
- VI. Object-Relational Database System
- VII. Web-Enabled Database System

I. **File Management System(1960's-1980's)**

- Earlier, punched cards technology was used to store data – later, the File management system(FMS) is used to do so.
- It is also considered as a predecessor of database **in which** data is maintained in a flat file.
- FMS is a database that stores information in a single file or table.
- In a text file, every line contains one record where fields either have fixed length or they are separated by commas, whitespaces, tabs or any other character.

Advantages of File Management System

- It is best for small databases.
- It is easy to understand and implement. Fewer skills are required to handle a flat file database.
- Less hardware and software skills are required to maintain a flat file database.
- Various access methods , e.g., sequential, indexed, random.

Disadvantages of File Management System

- It may contain fields which duplicate the data as there is no automation in flat files.
- If one record is to be deleted from the system, then all the relevant information in different fields has to be deleted manually making the data manipulation inefficient.

- It wastes the computer space by requiring it to keep the information on items that are logically cannot be available.
- Information retrieving is very time consuming in a large database.
- It requires extensive programming in third-generation language such as COBOL, BASIC.

Implementation of a File Management System

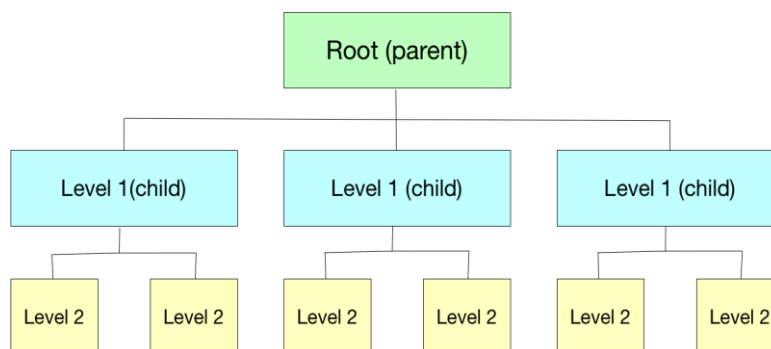
- It is implemented in Berkeley DB, SQLite, Mimesis, The Integration Engineer etc.

II. Hierarchical Database System (1968-1980)

Prominent hierarchical database model was IBM's first DBMS called **IMS (Information Management System)**

- Mid 1960s Rockwell collaborates with IBM to create the Information Management System (IMS) which lead the mainframe database market in 70's and early 80's.
- The drawbacks of previous system FMS in terms of accessing records and sorting records which took a long time was overcome in this database system via introduction of parent-child relationship between records in a database.
- The origin of the data is called the root from which several branches have data at different levels and the last level is called the leaf.

The Hierarchical Database Model



Advantages of Hierarchical Database System

- In a hierarchical database system, pace of accessing the information is speedy due to the predefined paths. This increases the performance of a database.
- The relationships among different entities are easy to understand.
- Less redundant data.
- Data independence.
- Database security and integrity.

Disadvantages of Hierarchical Database System

- Hierarchical database model lacks flexibility. If a new relationship is to be established between two entities then a new and possibly a redundant database structure has to be build.

- Maintenance of data is inefficient in a hierarchical model. Any change in the relationships may require manual reorganization of the data.
- This system is also inefficient for non-hierarchical accesses.
- Complex implementation
- Lacks structural independence.

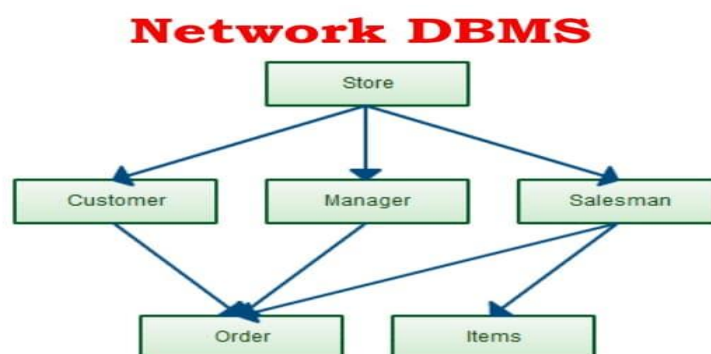
In order to avoid these drawbacks, the following system took its origin named as the Network Database System.

III. Network Database System (1960's – 1990's)

- Early 1960s, Charles Bachmann developed first DBMS at Honeywell, **Integrated Data Store (IDS)**
- It was standardized in 1971 by the **CODASYL** group (**Conference on Data Systems Languages**).
- Unlike hierarchical database model, Network database system allows multiple parent and child relationships i.e., it maintains many-to-many relationship.
- Network database system is basically a graph structure which was created to achieve three main objectives:
 - a) To represent complex data relationships more effectively.
 - b) To improve the performance of the database.
 - c) To implement a database standard.
- In a Network database system, a relationship is referred to as a set. Each set comprises of two types of records, an owner record which is same as parent type in hierarchical and a member record which is similar to the child type record in hierarchical database model.

Thus, this system identifies the following three database components:

1. Network schema—database organization[structure]
 2. Sub-schema—views of database per user
 3. Data management language — at low level , procedural
- It can also be represented as follows:



Advantages of Network Database System

- The network database system makes the data access quite easy and proficient as an application can access the owner record and all the member records within a set.
- This system is conceptually easy to design.
- This system ensures data integrity because no member can exist without an owner. So the user must make an owner entry and then the member records.
- It also ensures the data independence because the application works independently of the data.

Disadvantages of Network Database System

- This system lacks structural independence which means that to bring any change in the database structure; the application program must also be modified before accessing the data.
- A user friendly database management system cannot be established via network model.

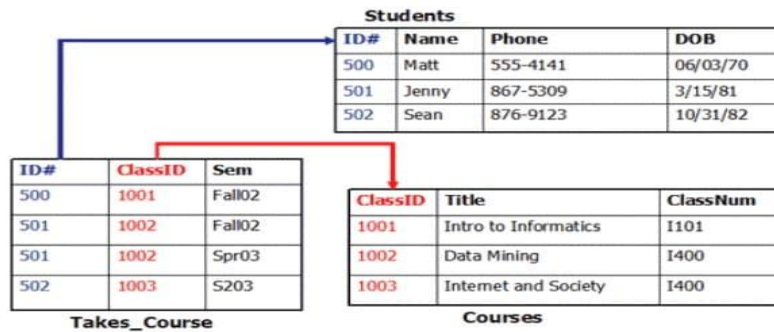
Implementation of Network Database System

- Network database system is implemented in Digital Equipment Corporation DBMS-10, Digital Equipment Corporation DBMS-20, RDM Embedded, Turbo IMAGE, Univac DMS-1100 etc.

IV. Relational Database System (1970's-present)

- The relational database model was proposed by E. F. Codd in 1970.
- It can be defined using the following two terminologies:
 1. Instance – a table with rows or columns.
 2. Schema – specifies the structure (name of relation, name and type of each column)
- This system is based on branches of mathematics called set theory and predicate logic.
- After the hierarchical and network database systems, the birth of this system was huge step ahead.
- It allows the entities to be related through a common attribute. So in order to relate two tables (entities), they simply need to have a common attribute.
- Thus using relational database ample information can be stored using small tables.
- The accessing of data is also very efficient. The user only has to enter a query, and the application provides the user with the asked information.
- Relational databases are established using a computer language, Structured Query Language (SQL). This language forms the basis of all the database applications available today, from Access to Oracle.
- Relationships between tables are also formed in this system as following.

Relational DBMS



Advantages of Relational Database System

- Relational database system supports mathematical set of operations like union, intersection, difference and Cartesian product. It also supports select, project, relational join and division operations.
- It uses normalization structure which helps to achieve data independence more easily.
- Security control can also be implemented more effectively by imposing an authorization control on the sensitive attributes present in a table.
- It uses a language which is easy and human readable.

Disadvantages of Relational Database System

- The response to a query becomes time-consuming and inefficient if the number of tables between which the relationships are established increases.

Implementation of Relational Database System

- It is implemented in Oracle, Microsoft, IBM, MySQL, PostgreSQL, SQLite etc.

V. Object – Oriented Database System (1990's – present)

- Object oriented database system is one in which the data or information is presented in the form of objects, much like in object-oriented programming language as shown in Figure 1.
- Furthermore, object oriented DBMS also facilitate the user by offering transaction support, language for various queries, and indexing options.
- Also, these database systems have the ability to handle data efficiently over multiple servers.
- Unlike relational database, object-oriented database works in the framework of real programming languages like JAVA or C++.

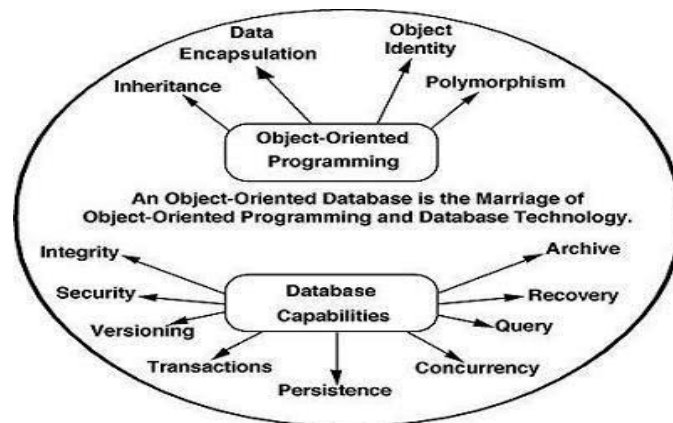


Figure 1. Makeup of an Object-Oriented Database

Advantages of Object-Oriented Database System

- If there are complex (many-to-many) relationships between the entities, the object-oriented database handles them much faster than any of the above discussed database systems.
- Navigation through the data is much easier.
- Objects do not require assembly or disassembly hence it saves the coding and execution time.

Disadvantages of Object-Oriented Database System

- Lower efficiency level when data or relationships are simple.
- Data can be accessible via specific language using a particular API which is not the case in relational databases.

VI. Object – Relational Database System (1990's – present)

- Object relational databases span the object and relational concepts.
- It displays a modified object-oriented user-display over the already implemented relational database management system. When various software interact with this modified-database management system, they will customarily operate in a manner such that the data is assumed to be saved as objects.
- The basic working of this database system is that it translates the useful data into organized tables, distributed in rows and columns, and from then onwards, it manages data the same way as done in a relational database system.
- Similarly, when the data is to be accessed by the user, it is again translated from processed to complex form.

Advantages of Object-Relational Database System

- Data remains encapsulated in object-relational database.
- Concept of inheritance and polymorphism can also be implemented in this database.

Disadvantages of Object-Relational Database System

- Object relational database is complex.

- Proponents of relational approach believe simplicity and purity of relational model are lost.
- It is costly as well.

VII. Web-Enabled Database (1990's – present)

- Web-Enabled database simply put a database with a web-based interface.
- This implies that there can be a separation of concerns; namely, the web designer does not need to know the details about the DB's underlying design.
- Similarly, the DB designer needs to concern himself with the DB's web interface.
- A web enabled database uses three layers to function: a presentation layer, a middle layer and the database layer.

Advantages of Web-Enabled Database

- A web-enabled database allows users to get the information they need from a central repository on demand.
- The database is easy and simple to use.
- The data accessibility is easy via web-enabled database.

Disadvantages of Web-Enabled Database

- Main disadvantage is that it can be hacked easily.
- Web enabled databases support the full range of DB operations, but in order to make them easy to use, they must be “dumped down”.

1.2.1.3 Types of Databases

There are various types of databases used for storing different varieties of data:

I. Centralized Database

- It is the type of database that stores data at a centralized database system.
- It comforts the users to access the stored data from different locations through several applications.
- These applications contain the authentication process to let users access data securely.
- An example of a Centralized database can be Central Library that carries a central database of each library in a college/university.

Advantages of Centralized Database

- It has decreased the risk of data management, i.e. manipulation of data will not affect the core data.
- Data consistency is maintained as it manages data in a central repository.
- It also provides better data quality, which enables organizations to establish data standards.
- It is less expensive because a few vendors are required to handle the data sets.

Disadvantages of Centralized Database

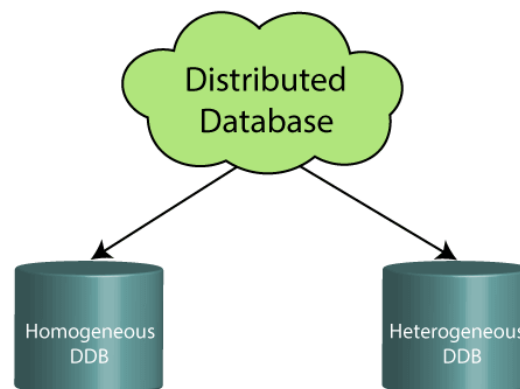
- The size of the centralized database is large, which increases the response time for fetching the data.
- It is not easy to update such an extensive database system.
- If any server failure occurs, entire data will be lost which could be a huge loss.

II. Distributed Database(DDB)

- Unlike a centralized database system, data is distributed among different database systems of an organization in distributed systems.
- These database systems are connected via communication links.
- Such links help the end-users to access the data easily.
- Examples of the Distributed database are Apache Cassandra, HBase, Ignite, etc.

Classification of Distributed Database System

- A distributed database system can further be classified into two parts as following:



Homogeneous DDB: The database systems which execute on the same operating system and use the same application process and carry the same hardware devices are known as homogeneous distributed database systems.

Heterogeneous DDB: The database systems which execute on different operating systems under different application procedures, and carry different hardware devices are known as heterogeneous distributed database systems.

Advantages of Distributed Database

- Modular development is possible in a distributed database. i.e. the system can be expanded by including new computers and connecting them to the distributed system.
- One server failure will not affect the entire data set.

Disadvantages of Distributed Database

Although, distributed DBMS is capable of effective communication and data sharing still it suffers from various disadvantages are as following below.

- **Complex nature** :The nature of Distributed DBMS is more complex than a centralized DBMS because of requirement of complex softwares. Also, It ensures no data replication, which adds even more complexity in its nature.
- **Overall Cost** :Various costs such as maintenance cost, procurement cost, hardware cost, network/communication costs, labor costs, etc, adds up to the overall cost and make it costlier than normal DBMS.
- **Security issues**:In a Distributed Database, along with maintaining no data redundancy, the security of data as well as network is a prime concern. A network can be easily attacked for data theft and misuse.
- **Integrity Control**:In a vast distributed database system, maintaining data consistency is important. All changes made to data at one site must be reflected to all the sites. The communication and processing cost is high in distributed DBMS in order to enforce the integrity of data.
- **Lacking Standards**:Although it provides effective communication and data sharing, still there are no standard rules and protocols to convert a centralized DBMS to a large distributed DBMS. Lack of standards decreases the potential of distributed DBMS.

III. Relational Database

- This database is based on the relational data model, which stores data in the form of rows(tuple) and columns(attributes), and together forms a table(relation).
- A relational database uses SQL for storing, manipulating, as well as maintaining the data.
- E.F. Codd invented the database in 1970. Each table in the database carries a key that makes the data unique from others.
- Examples of Relational databases are MySQL, Microsoft SQL Server, Oracle, etc.

Properties of Relational Database

There are following four commonly known properties of a relational model known as ACID properties, where:

- **A means Atomicity**: This ensures the data operation will complete either with success or with failure. It follows the 'all or nothing' strategy. For example, a transaction will either be committed or will abort.
- **C means Consistency**: If we perform any operation over the data, its value before and after the operation should be preserved. For example, the account balance before and after the transaction should be correct, i.e., it should remain conserved.
- **I means Isolation**: There can be concurrent users for accessing data at the same time from the database. Thus, isolation between the data should remain isolated. For example, when multiple transactions occur at the same time, one transaction effects should not be visible to the other transactions in the database.
- **D means Durability**: It ensures that once it completes the operation and commits the data, data changes should remain permanent.

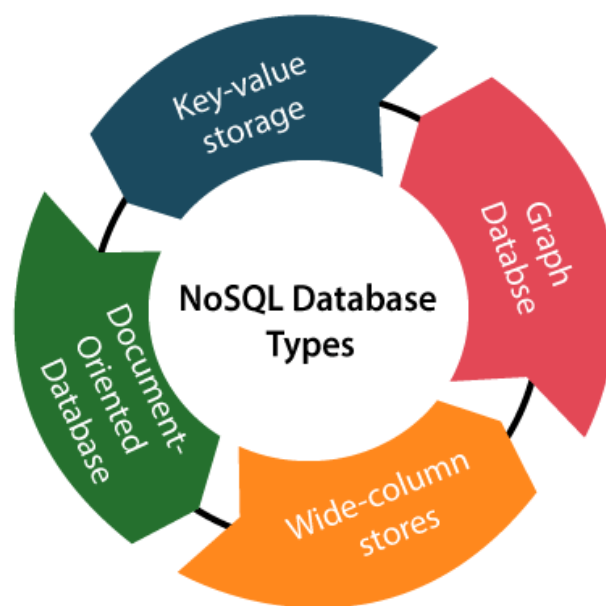
IV. NoSQL Database

- Non-SQL/Not Only SQL is a type of database that is used for storing a wide range of data sets.
- It is not a relational database as it stores data not only in tabular form but in several different ways.
- It came into existence when the demand for building modern applications increased.

Thus, NoSQL presented a wide variety of database technologies in response to the demands.

Classification of NoSQL Database

We can further divide a NoSQL database into the following four types:



Key-value storage: It is the simplest type of database storage where it stores every single item as a key (or attribute name) holding its value, together.

Document-oriented Database: A type of database used to store data as JSON-like document. It helps developers in storing data by using the same document-model format as used in the application code.

Graph Databases: It is used for storing vast amounts of data in a graph-like structure. Most commonly, social networking websites use the graph database.

Wide-column stores: It is similar to the data represented in relational databases. Here, data is stored in large columns together, instead of storing in rows.

Advantages of NoSQL Database

- It enables good productivity in the application development as it is not required to store data in a structured format.
- It is a better option for managing and handling large data sets.
- It provides high scalability.

- Users can quickly access data from the database through key-value.

Disadvantages of NoSQL Database

- NoSQL databases don't have the reliability functions which Relational Databases have (basically don't support ACID).
- This also means that NoSQL databases offer consistency in performance and scalability.
- In order to support ACID, developers will have to implement their own code, making their systems more complex.
- This may reduce the number of safe applications that commit transactions, for example bank systems.
- NoSQL is not compatible (at all) with SQL.
 - *Note:* Some NoSQL management systems do use a Structured Query Language.
 - This means that you will need a manual query language, making things slower and more complex.
- NoSQL are very new compared to Relational Databases, which means that are far less stable and may have a lot less functionalities.

V. Cloud Database

- A type of database where data is stored in a virtual environment and executes over the cloud computing platform.
- It provides users with various cloud computing services (SaaS, PaaS, IaaS, etc.) for accessing the database.
- There are numerous cloud platforms, but the best options are Amazon Web Services(AWS), Microsoft Azure, Kamatera, PhonixNAP, ScienceSoft, Google Cloud SQL, etc.

Advantages of Cloud Database

- Scalability
- Reduced Administrative Burden
- Improved Security

Disadvantages of Cloud Database

- Inflexibility
- Cost
- Downtime

VI. Personal Database

- Collecting and storing data on the user's system defines a Personal Database.
- This database is basically designed for a single user.

Advantages of Personal Database

- It is simple and easy to handle.

- It occupies less storage space as it is small in size.

Disadvantages of Personal Database

- Fewer amounts of data and information are stored in personal database management system.
- There is no connectivity with other computer to get more data.

VII. Operational Database

- The type of database which creates and updates the database in real-time is known as an operational database.
- It is basically designed for executing and handling the daily data operations in several businesses.
- For example, An organization uses operational databases for managing per day transactions.

Advantages of Operational Database

- Operational Databases systems are greatly versatile and accommodate distributed systems like NoSQL, SQL, New SQL Databases.
- These systems are highly available, fault-tolerant and highly scalable as discussed.
- They are highly secured as they offer built-in support for encryption, auditing, and protection from cyber
- They are extremely flexible and can work with multiple applications without losing the state of the database.
- Such systems are often more economical as systems in an operational database are usually based on distributed networks and systems thereby reducing costs drastically and ensuring consistency.

Disadvantages of Operational Database

- They usually have a learning curve where personnel are required to give relevant training to manage such databases and that increases the overheads expenses
- Even the installation process of such an operational database system requires time and effort where they need to be set up in an optimal way of meeting business goals and extracting most benefit from the system.
- Security could also be an issue where since the data is stored in a remote location having overall control could be difficult and we have seen recent examples of customer data being hacked from some of the most prominent tech companies.

VIII. Enterprise Database

- Large organizations or enterprises use this database for managing a massive amount of data.
- It helps organizations to increase and improve their efficiency.
- Such database also allows simultaneous access to users.

Advantages of Enterprise Database

- Multi processes are supportable over the Enterprise database.

- It allows executing parallel queries on the system.

Disadvantages of Enterprise Database

- High cost of implementation and maintenance..
- Inflexibility of the system.

Other than above mentioned database systems, following are the database systems which are already discussed in evolution part such as:

- Object-Oriented Databases
- Hierarchical Databases
- Network Databases

Database can also be classified according to the following factors. They are:

I. Number of Users

a)Single user database: It supports only one user at a time. For e.g. Desktop or personal computer database.

b)Multiuser database:It supports multiple users at the same time. For e.g. Workgroup database and enterprise database

II. Database Location

a)Centralized Database

b)Distributed Database

III. Expected type

IV. Extent of use

1.2.2 What is a Management System?

- The Management system is important because without the existence of some kind of rules and regulations, it is not possible to maintain the database.
- We have to select the particular attributes which should be included in a particular table; the common attributes to create relationship between two tables; if a new record has to be inserted or deleted then which tables should have to be handled etc.
- These issues must be resolved by having some kind of rules to follow in order to maintain the integrity of the database.
- Database systems are designed to manage large bodies of information.
- Management of data involves both defining structures for storage of information and providing mechanisms for the manipulation of information.
- In addition, the database system must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access.
- If data is to be shared among several users, the system must avoid possible anomalous results.
- Because information is so important in most organizations, computer scientists have developed a large body of concepts and techniques for managing data. These concepts and techniques form the focus of this chapter.

Self-Check Exercise

1. What is a Database? Give example.

2. Explain various types of database systems.
3. Define Relational Database System along with its advantage and disadvantage.
4. Describe the various advantages of Operational Database?
5. Explain Centralized database.
6. Write the difference between Homogeneous and Heterogeneous Distributed DataBase.

1.3 BASIC TERMINOLOGY OF DBMS

Database management system is a software which is used to manage the database. For example: MySQL, Oracle, etc are very popular commercial databases which are used in different applications.

Thus, The interactions catered for by most existing DBMS fall into four main groups:

- a) **Data Definition:** It helps in creation, modification and removal of definitions that define the organization of data in database.
- b) **Data Updation:** It helps in insertion, modification and deletion of the actual data in the database.
- c) **Data Retrieval:** It helps in retrieval of data from the database which can be used by applications for various purposes.
- d) **User Administration:** It helps in registering and monitoring users, enforcing data security, monitoring performance, maintaining data integrity, dealing with concurrency control and recovering information corrupted by unexpected failure.

1.3.1 Goals of DBMS

- The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient
- Manages large bodies of information
- Provides convenient and efficient ways to store and access information
- Secures information against system failure or tampering
- Permits data to be shared among multiple users

1.3.2 Properties of DBMS

- It represents some aspect of the real world. Changes to the real world reflected in the database.
- It is a logically coherent collection of data with some inherent meaning.
- It is designed and populated with data for a specific purpose.

1.3.3 Need of DBMS

- Before the advent of DBMS, organizations typically stored information using a “File Processing Systems”.
- Example of such systems is File Handling in High Level Languages like C, Basic and COBOL etc., these systems have major disadvantages to perform the data manipulation. So to overcome those drawbacks now we are using the DBMS.

- In addition to that, the database system must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access. If data is to be shared among several users, the system must avoid possible anomalous results.
- Therefore, DBMS was a new concept then, and all the research was done to make it overcome the deficiencies in traditional style of data management.

1.3.4 Features of DBMS

Features of database management system are:

- Prevents data redundancy.
- Prevents unauthorised access.
- Provides persistent storage for program objects and data structures.
- Provides Multiple user interfaces.
- Provides Integrity Constraints.
- Provides Backup and Recovery.

1.3.5 Users of DBMS

A typical DBMS has users with different rights and permissions who use it for different purposes. Some users retrieve data and some back it up. The users of a DBMS can be broadly categorized as follows –

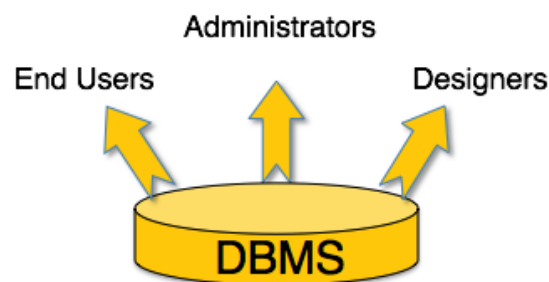


Figure: DBMS Users

I. Administrators

- Administrators maintain the DBMS and are responsible for administrating the database.
- They are responsible to look after its usage and by whom it should be used.
- They create access profiles for users and apply limitations to maintain isolation and force security.
- Administrators also look after DBMS resources like system license, required tools, and other software and hardware related maintenance.

II. Designers

- Designers are the group of people who actually work on the designing part of the database.

- They keep a close watch on what data should be kept and in what format.
- They identify and design the whole set of entities, relations, constraints, and views.

III. End Users

- End users are those who actually reap the benefits of having a DBMS.
- End users can range from simple viewers who pay attention to the logs or market rates to sophisticated users such as business analysts.

Thus, People who work with a database can be categorized as database users or database administrators.

➤ Database Users:

There are four different types of database-system users, differentiated by the way they expect to interact with the system.

a) Naive users

- Naive users are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously.
- For example, a bank teller who needs to transfer \$50 from account A to account B invokes a program called transfer.
- This program asks the teller for the amount of money to be transferred, the account from which the money is to be transferred, and the account to which the money is to be transferred.

b) Application programmers

- Application programmers are computer professionals who write application programs.
- Application programmers can choose from many tools to develop user interfaces.
- Rapid application development (RAD) tools are tools that enable an application programmer to construct forms and reports without writing a program.

c) Sophisticated users

- Sophisticated users interact with the system without writing programs.
- Instead, they form their requests in a database query language.
- They submit each such query to a query processor, whose function is to break down DML statements into instructions that the storage manager understands.
- Analysts who submit queries to explore data in the database fall in this category.

d) Specialized users

- Specialized users are sophisticated users who write specialized database applications that do not fit into the traditional data-processing framework.

1.3.6 Levels of abstraction in a DBMS

- Hiding certain details of how the data are stored and maintained. A major purpose of database system is to provide users with an “Abstract View” of the data.
- The goal of the abstraction in the DBMS is to separate the users request and the physical storage of data in the database.

- In DBMS there are 3 levels of data abstraction.

i. Physical Level

The lowest Level of Abstraction describes “How” the data is actually stored. The physical level describes complex low level data structures in detail.

ii. Logical Level

This level of data Abstraction describes “What” data is to be stored in the database and what relationships exist among those data. Database Administrators use the logical level of abstraction.

iii. View Level

It is the highest level of data Abstracts that describes only part of entire database. Different users require different types of data elements from each database. The system may provide many views for the some database.

Self-Check Exercise

1. List the properties of DBMS.
2. How can you differentiate sophisticated users from specialized users?
3. What are the goals of database management system?
4. Why do you need database management system?
5. What is the role of administrators in dbms?
6. What is the goal of abstraction in the database management system?

1.4 DATA INDEPENDENCE

- A very important advantage of using DBMS is that it offers Data Independence which is an ability to modify a scheme definition in one level without affecting a scheme definition in a higher level.
- It is of two types:
 - Physical Data Independence
 - Logical Data Independence

1.4.1 Physical Data Independence

- It is an ability to modify the physical schema without causing application programs to berewritten.
- Modifications at this level are usually to improve performance.

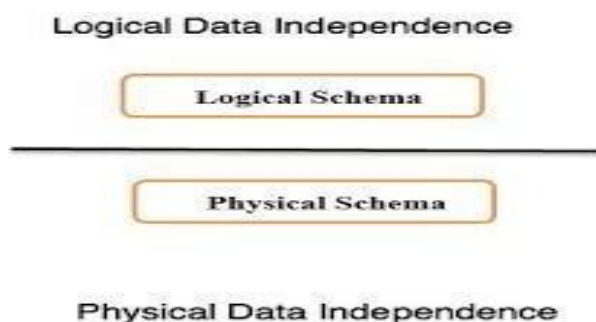


Fig: Data Independence

1.4.2 Logical Data Independence

- It is an ability to modify the conceptual schema without causing application programs to be rewritten.
- It is usually done when logical structure of database is altered.
- Logical data independence is harder to achieve as the application programs are usually heavily dependent on the logical structure of the data.

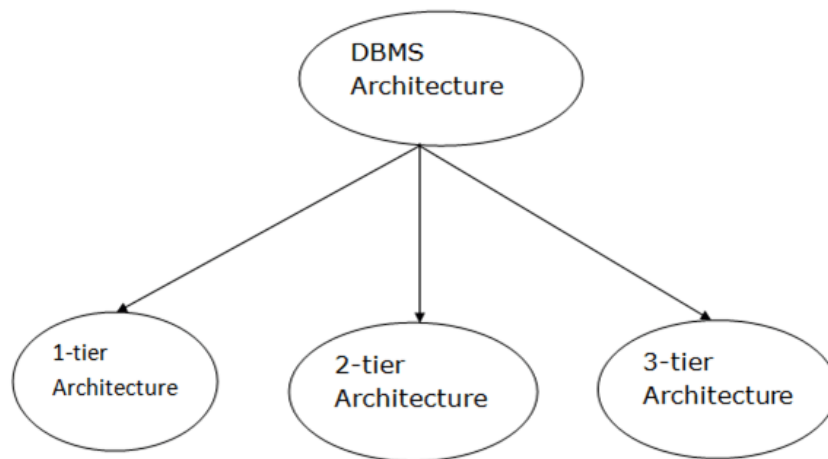
1.5 ARCHITECTURE OF DBMS

The DBMS design depends upon its architecture. The basic client/server architecture is used to deal with a large number of PCs, web servers, database servers and other components that are connected with networks.

The client/server architecture consists of many PCs and a workstation which are connected via the network.

DBMS architecture depends upon how users are connected to the database to get their request done.

1.5.1 Types of DBMS Architecture



- Database architecture can be seen as a single tier or multi-tier.
- But logically, database architecture is of two types like:

1.5.1.1 1-Tier Architecture

- In this architecture, the database is directly available to the user. It means the user can directly sit on the DBMS and uses it.
- Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.
- The 1-Tier architecture is used for development of the local application, where programmers can directly communicate with the database for the quick response.

1.5.1.2 2-Tier Architecture

- The 2-Tier architecture is same as basic client-server. In the two-tier architecture, applications on the client end can directly communicate with the database at the server side. For this interaction, API's like: **ODBC**, **JDBC** are used.
- The user interfaces and application programs are run on the client-side.
- The server side is responsible to provide the functionalities like: query processing and transaction management.
- To communicate with the DBMS, client-side application establishes a connection with the server side.

1.5.1.3 3-Tier Architecture

- The 3-Tier architecture contains another layer between the client and server. In this architecture, client can't directly communicate with the server.
- The application on the client-end interacts with an application server which further communicates with the database system.
- End user has no idea about the existence of the database beyond the application server. The database also has no idea about any other user beyond the application.
- The 3-Tier architecture is used in case of large web application.

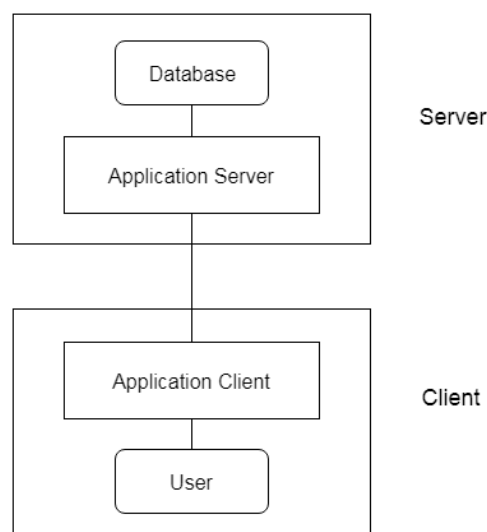
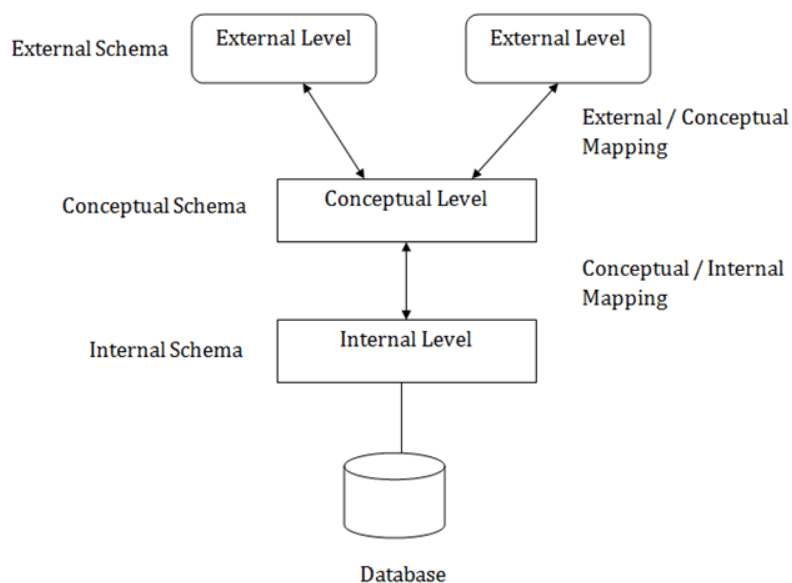


Fig: 3-tier Architecture

1.5.2 Three Schema Architecture

- The three schema architecture is also called ANSI/SPARC architecture or three-level architecture.
- This framework is used to describe the structure of a specific database system.
- The three schema architecture is also used to separate the user applications and physical database.
- The three schema architecture contains three-levels. It breaks the database down into three different categories.
- **The three-schema architecture is as follows:**



- Above shows the DBMS architecture.
- Mapping is used to transform the request and response between various database levels of architecture.
- Mapping is not good for small DBMS because it takes more time.
- In External / Conceptual mapping, it is necessary to transform the request from external level to conceptual schema.
- In Conceptual / Internal mapping, DBMS transform the request from the conceptual to internal level.

1. Internal Level

- The internal level has an internal schema which describes the physical storage structure of the database.
- The internal schema is also known as a physical schema.
- It uses the physical data model. It is used to define that how the data will be stored in a block.
- The physical level is used to describe complex low-level data structures in detail.

2. Conceptual Level

- The conceptual schema describes the design of a database at the conceptual level. Conceptual level is also known as logical level.
- The conceptual schema describes the structure of the whole database.
- The conceptual level describes what data are to be stored in the database and also describes what relationship exists among those data.
- In the conceptual level, internal details such as an implementation of the data structure are hidden.
- Programmers and database administrators work at this level.

3. External Level

- At the external level, a database contains several schemas that sometimes called as subschema. The subschema is used to describe the different view of the database.
- An external schema is also known as view schema.
- Each view schema describes the database part that a particular user group is interested and hides the remaining database from that user group.
- The view schema describes the end user interaction with database systems.

Self-check exercise:

1. What do you understand by the term data independence?
2. What is the difference between Internal level and Conceptual level in architecture of DBMS?
3. Why do you use mapping in three schema architecture of DBMS?
4. How can you differentiate 1-tier architecture from 2-tier architecture in DBMS?

1.6 TRADITIONAL FILE SYSTEM VS DBMS

1.6.1 Traditional File system

Traditional File systems were the collection of data with the help of written procedures for managing the database. It provides the details of data representation and storage of data. In traditional file system the data is stored in the group of files. The data stored in the files are dependent on each other. The traditional file system does not have a crash recovery mechanism but it has a fast file system recovery. Normally, low level languages like C, C++ and COBOL were used to design the files.

1.6.2 Advantages of Traditional File System

Following are the advantages of traditional file system:

- File processing costs less and can be more speedy than database.
- File processing design approach is well suited to mainframe hardware and batch input.
- Companies mainly use file processing to handle large volumes of structured data on a regular basis.
- It can be more efficient and costs less than DBMS in certain situations.

- Design is simple.
- Customization is easy and efficient.

1.6.3 Disadvantages of Traditional File System

Following are the disadvantages of traditional file system:

- Data redundancy and inconsistency.
- Difficulty in accessing data.
- Data isolation – multiple files and formats.
- Integrity problems
- Unauthorized access is not restricted.
- It co-ordinates only physical access.

To overcome disadvantages of File system, DBMS came in use which is a collection of inter-related data. It has a set of programs to access the data. Basically, it contains information about particular enterprise. It provides convenient and efficient environment for use.

1.6.4 Differences between Traditional File System and DBMS

File System	DBMS
A file system is software that manages and organizes the files in a storage medium. It controls how data is stored and retrieved.	DBMS or Database Management System is a software application. It is used for accessing, creating, and managing databases.
The file system provides the details of data representation and storage of data.	DBMS gives an abstract view of data that hides the details
Storing and retrieving of data can't be done efficiently in a file system.	DBMS is efficient to use as there are a wide variety of methods to store and retrieve data.
It does not offer data recovery processes.	There is a backup recovery for data in DBMS.
The file system doesn't have a crash recovery mechanism.	DBMS provides a crash recovery mechanism
Protecting a file system is very difficult.	DBMS offers good protection mechanism.
In a file management system, the redundancy of data is greater.	The redundancy of data is low in the DBMS system.
Data inconsistency is higher in the file system.	Data inconsistency is low in a database management system.
The file system offers lesser security.	Database Management System offers high

	security.
File System allows you to stores the data as isolated data files and entities.	Database Management System stores data as well as defined constraints and interrelation.
Not provide support for complicated transactions.	Easy to implement complicated transactions.
The centralization process is hard in File Management System.	Centralization is easy to achieve in the DBMS system.
It doesn't offer backup and recovery of data if it is lost.	DBMS system provides backup and recovery of data even if it is lost.
There is no efficient query processing in the file system.	You can easily query data in a database using the SQL language.
These system doesn't offer concurrency.	DBMS system provides a concurrency facility.

1.7 ADVANTAGES OF DBMS

Following are the advantages of DBMS:

- **Controls database redundancy:** It can control data redundancy because it stores all the data in one single database file and that recorded data is placed in the database.
- **Data sharing:** In DBMS, the authorized users of an organization can share the data among multiple users.
- **Easily Maintenance:** It can be easily maintainable due to the centralized nature of the database system.
- **Reduce time:** It reduces development time and maintenance need.
- **Backup:** It provides backup and recovery subsystems which create automatic backup of data from hardware and software failures and restores the data if required.
- **Multiple user interface:** It provides different types of user interfaces like graphical user interfaces, application program interfaces etc.

1.8 DISADVANTAGES OF DBMS

Database Management System is quite useful compared to the file based management system. However, it does have some disadvantages. Some of those are as follows:

- **Expensive:** Creating and managing a database is quite costly. High cost software as well as hardware are required for the database. Also highly trained staff is required to handle the database and it also needs continuous maintenance. All of these ends up making a database quite a costly venture.
- **High Complexity:** A Database Management System is quite complex as it involves creating, modifying and editing a database. Consequently, the people who handle a database or work with it need to be quite skilled or valuable data can be lost.
- **Database handling staff required:** As discussed in the previous point, database and DBMS are quite complex. Hence, skilled personnel are required to handle the database so that it works in optimum condition. This is a costly venture as these professionals need to be very well paid.
- **Database Failure:** All the relevant data for any company is stored in a database. So it is imperative that the database works in optimal condition and there are no failures. A database failure can be catastrophic and can lead to loss or corruption of very important data.
- **High Hardware Cost:** A database contains vast amount of data. So a large disk storage is required to store all this data. Sometimes extra storage may even be needed. All this increases hardware costs by a lot and makes a database quite expensive.
- **Huge Size:** A database contains a large amount of data, especially for bigger organisations. This data may even increase as more data is updated into the database. All of these leads to a large size of the database. The bigger the database is, it is more difficult to handle and maintain. It is also more complex to ensure data consistency and user authentication across big databases.
- **Upgradation Costs:** Often new functionalities are added to the database. This leads to database upgradations. All of these upgradations cost a lot of money. Moreover it is also quite expensive to train the database managers and users to handle these new upgradations.
- **Cost of Data Conversion:** If the database is changed or modified in some manner, all the data needs to be converted to the new form. This cost may even exceed the database creation and management costs sometimes. This is the reason most organisations prefer to work on their old databases rather than upgrade to new ones.

1.9 CHARACTERISTICS OF DBMS

A modern DBMS has the following characteristics –

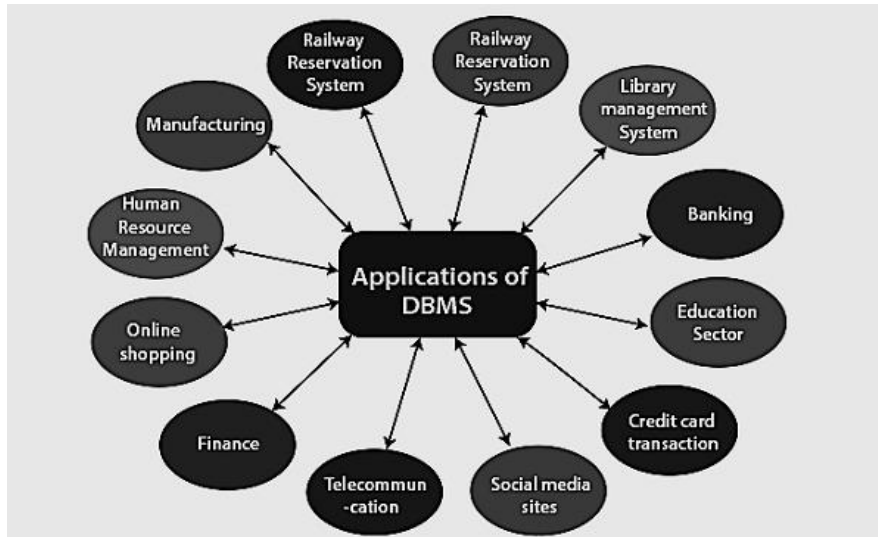
- **Real-world entity:** A modern DBMS is more realistic and uses real-world entities to design its architecture. It uses the behavior and attributes too. For example, a school database may use students as an entity and their age as an attribute.
- **Relation-based tables:** DBMS allows entities and relations among them to form tables. A user can understand the architecture of a database just by looking at the table names.
- **Isolation of data and application:** A database system is entirely different than its data. A database is an active entity, whereas data is said to be passive, on which the

database works and organizes. DBMS also stores metadata, which is data about data, to ease its own process.

- **Less redundancy:** DBMS follows the rules of normalization, which splits a relation when any of its attributes is having redundancy in values. Normalization is a mathematically rich and scientific process that reduces data redundancy.
- **Consistency:** Consistency is a state where every relation in a database remains consistent. There exist methods and techniques, which can detect attempt of leaving database in inconsistent state. A DBMS can provide greater consistency as compared to earlier forms of data storing applications like file-processing systems.
- **Query Language:** DBMS is equipped with query language, which makes it more efficient to retrieve and manipulate data. A user can apply as many and as different filtering options as required to retrieve a set of data. Traditionally it was not possible where file-processing system was used.
- **ACID Properties :** DBMS follows the concepts of **A**tomicity, **C**onsistency, **I**solation, and **D**urability (normally shortened as ACID). These concepts are applied on transactions, which manipulate data in a database. ACID properties help the database stay healthy in multi-transactional environments and in case of failure.
- **Multiuser and Concurrent Access:** DBMS supports multi-user environment and allows them to access and manipulate data in parallel. Though there are restrictions on transactions when users attempt to handle the same data item, but users are always unaware of them.
- **Multiple views:** DBMS offers multiple views for different users. A user who is in the Sales department will have a different view of database than a person working in the Production department. This feature enables the users to have a concentrate view of the database according to their requirements.
- **Security:** Features like multiple views offer security to some extent where users are unable to access data of other users and departments. DBMS offers methods to impose constraints while entering data into the database and retrieving the same at a later stage. DBMS offers many different levels of security features, which enables multiple users to have different views with different features. For example, a user in the Sales department cannot see the data that belongs to the Purchase department. Additionally, it can also be managed how much data of the Sales department should be displayed to the user. Since a DBMS is not saved on the disk as traditional file systems, it is very hard for miscreants to break the code.

1.10 APPLICATIONS OF DBMS

There are different fields where a database management system is utilized. Following are a few applications which utilize the information base administration framework as shown in below figure.



- **Railway Reservation System** – The railway reservation system database plays a very important role by keeping record of ticket booking, train’s departure time and arrival status and also gives information regarding train late to people through the database.
- **Library Management System** – Now-a-days it’s become easy in the Library to track each book and maintain it because of the database. This happens because there are thousands of books in the library. It is very difficult to keep a record of all books in a copy or register. Now DBMS is used to maintain all the information related to book issue dates, name of the book, author and availability of the book.
- **Banking** – Banking is one of the main applications of databases. We all know there will be a thousand transactions through banks daily and we are doing this without going to the bank. This is all possible just because of DBMS that manages all the bank transactions.
- **Universities and colleges** – Now-a-days examinations are done online. So, the universities and colleges are maintaining DBMS to store Student’s registrations details, results, courses and grades i.e. all the information in the database. For example, telecommunications. Without DBMS there is no telecommunication company. DBMS is most useful to these companies to store the call details and monthly postpaid bills.
- **Credit card transactions** – The purchase of items and transactions of credit cards are made possible only by DBMS. A credit card holder has to know the importance of the information which is secured through DBMS.
- **Social Media Sites** – By filling the required details we are able to access social media platforms. Many users sign up daily on social websites such as Facebook, Pinterest and Instagram. All the information related to the users are stored and maintained with the help of DBMS.

- **Finance** – Now-a-days there are lots of things to do with finance like storing sales, holding information and finance statement management etc. these all can be done with database systems.
- **Military** – In military areas the DBMS is playing a vital role. Military keeps records of soldiers and it has so many files that should be kept secure and safe. DBMS provides a high security to military information.
- **Online Shopping** – Now-a-days we all do Online shopping without wasting the time with the help of DBMS. The products are added and sold only with the help of DBMS that further includes Purchase information, invoice bills and payment.
- **Human Resource Management** – The management keeps records of each employee's salary, tax and work through DBMS.
- **Manufacturing** – Manufacturing companies make products and sell them on a daily basis. To keep records of all those details, DBMS is used.
- **Airline Reservation system** – Just like the railway reservation system, airlines also need DBMS to keep records of flights arrival, departure and delay status.

So finally, we can clearly conclude that the DBMS is playing a very important role in each and every field.

1.11 SUMMARY

- Database is a collection of inter-related data which is used to retrieve, insert and delete the data efficiently.
- Database Management System can also be defined as a software system that allows you to define, construct and manipulate databases used for various purposes such as to store customer and financial information of an organization.
- Data consists of facts, figures, statistics etc. having no particular meaning.
- Record is a collection of related data items.
- DBMS supports multi-user environment and allows them to access and manipulate data in parallel.
- A file system is a software that manages and organizes the files in a storage medium. It controls how data is stored and retrieved.
- DBMS architecture depends upon how users are connected to the database to get their request done.
- Relational database system supports mathematical set of operations like union, intersection, difference and Cartesian product.
- The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient
- Mapping is used to transform the request and response between various database levels of architecture.

1.12 REFERENCES

“Fundamentals of Database Systems”, Elmasri Navrate, 6th edition, 2013, Pearson
“Introduction to Database Systems”, C.J.Date, Pearson Education
“Data base System Concepts”, Silberschatz, Korth, McGraw Hill, V edition
Starting with Oracle, by John Day and Craig Van Slyke

1.13 FURTHER READING

Database System Concepts - 6th edition - Avi Silberschatz
Database Systems - A Practical Approach to Design, Implementation & Management
By Thomas Connolly, Carolyn Begg
Fundamentals of Database Systems - Elmasari , Navathe
Database Management Systems By Raghu Ramkrishnan, Gehrke
Database System Concepts, by Henry F. Korth

1.14 QUESTIONS FOR PRACTICE (SHORT & LONG ANSWERS TYPE)

1. What is database management system?
2. What are the various types of database system?
3. What are the applications of database management system?
4. Differentiate between traditional file system and modern DBMS.
5. List the characteristics of database management system.
6. What are the various advantages and disadvantages of hierarchical database system?
7. List the features of database management system.
8. What do you mean by physical data independence?
9. What are the types of DBMS architecture?
10. Name the users of DBMS.

B.Sc.(DATA SCIENCE)
SEMESTER-III
DATABASE MANAGEMENT SYSTEM

UNIT II: DATA MODELS

STRUCTURE

- 2.0 Objectives
- 2.2 Introduction
- 2.2 Data Models
 - 2.2.1 Data model Schema and Instance
- 2.3 Keys in DBMA
 - 2.3.1 Why do we need a Key?
- 2.4 Types of Keys
- 2.5 Difference Between Primary Key & Foreign Key
- 2.6 Integrity Constraints
 - 2.6.1 Types of Integrity Constraints
- 2.7 Relational Model
 - 2.7.1 Codd's 12 Rules of RDBMS
 - 2.7.2 Relational Model Concepts
 - 2.7.3 Properties of Relations
 - 2.7.4 Operations in Relational Model
 - 2.7.5 Equipment with a Query Language
 - 2.7.6 ER Model to Relational Model
 - 2.7.7 Best Practices for Creating a Relational Model
 - 2.7.8 Advantages of using Relational Model
 - 2.7.9 Disadvantages of using Relational Model
- 2.8 Hierarchical Data Model
- 2.9 Network Data Model
- 2.10 Difference between Hierarchical, Network and Relational Data Model
- 2.11 Summary
- 2.12 References
- 2.13 Further Readings
- 2.14 Practice Questions

2.0 OBJECTIVES

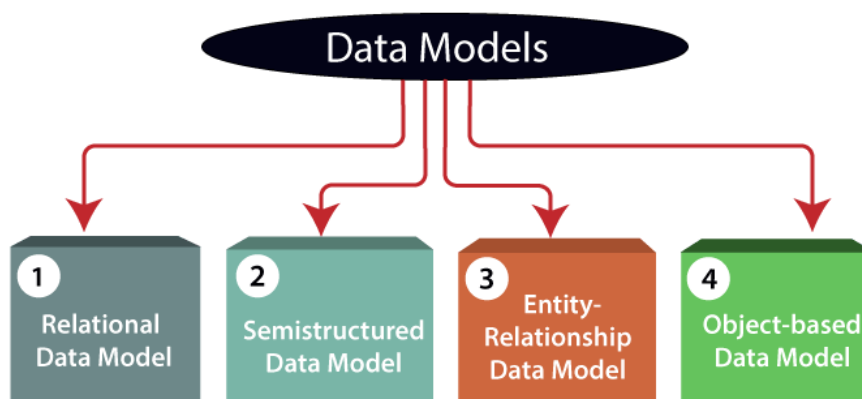
- To provide overview of data model.
- To discuss concept of various data model.
- To explain relational keys of DBMS.
- To discuss difference between primary and foreign keys.
- To explain integrity constraints.
- To discuss concept of relational model.
- To provide description of Codd's 12 Rules of RDBMS.
- To discuss Operations in Relational Model.
- To explain the terms of relational algebra and relational calculus.
- To describe ER Model to Relational Model.
- To describe advantages as well as disadvantages of Relational model.
- To explain hierarchical model and network model.

2.1 INTRODUCTION

A data model defines the logical design and structure of a database and defines how data will be stored, accessed and updated in a database management system. In this unit, we will discuss four types of data models such as Relational Data Model, Semi-Structured Data Model, Entity-Relationship Data Model, Object-based Data Model. This unit also explains the various relational keys such as primary key, foreign key, candidate key etc. We will discuss integrity constraints along with its types. Apart from above mentioned terms, concepts of various models such as relational model along with its rules, hierarchical model and network model will be discussed.

2.2 DATA MODELS

Data Model is the modeling of the data description, data semantics, and consistency constraints of the data. It provides the conceptual tools for describing the design of a database at each level of data abstraction. Therefore, there are following four data models used for understanding the structure of the database as shown in below figure:



1) Relational Data Model: This type of model designs the data in the form of rows and columns within a table. Thus, a relational model uses tables for representing data and in-between relationships. Tables are also called relations. This model was initially described by Edgar F. Codd, in 1969. The relational data model is the widely used model which is primarily used by commercial data processing applications.

2) Semi-Structured Data Model: This type of data model is different from the other three data models (explained above). The semistructured data model allows the data specifications at places where the individual data items of the same type may have different attributes sets. The Extensible Markup Language, also known as XML, is widely used for representing the semistructured data. Although XML was initially designed for including the markup information to the text document, it gains importance because of its application in the exchange of data.

3) Entity-Relationship Data Model: An ER model is the logical representation of data as objects and relationships among them. These objects are known as entities, and relationship is an association among these entities. This model was designed by Peter Chen and published in 1976 papers. It was widely used in database designing. A set of attributes describe the entities. For example, student_name, student_id describes the 'student' entity. A set of the same type of entities is known as an 'Entity set', and the set of the same type of relationships is known as 'relationship set'.

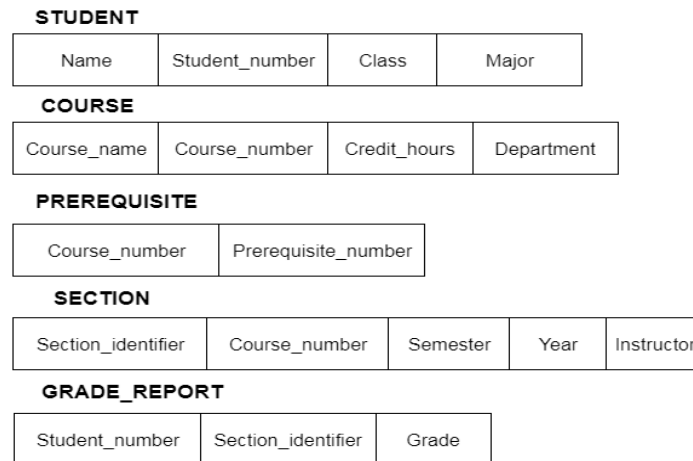
4) Object-based Data Model: An extension of the ER model with notions of functions, encapsulation, and object identity, as well. This model supports a rich type system that includes structured and collection types. Thus, in 1980s, various database systems following the object-oriented approach were developed. Here, the objects are nothing but the data carrying its properties.

Data model Schema and Instance

- The data which is stored in the database at a particular moment of time is called an instance of the database.
- The overall design of a database is called schema.
- A database schema is the skeleton structure of the database. It represents the logical view of the entire database.
- A schema contains schema objects like table, foreign key, primary key, views, columns, data types, stored procedure, etc.
- A database schema can be represented by using the visual diagram. That diagram shows the database objects and relationship with each other.
- A database schema is designed by the database designers to help programmers whose software will interact with the database. The process of database creation is called data modeling.
- A schema diagram can display only some aspects of a schema like the name of record type, data type, and constraints. Other aspects can't be specified through the schema

diagram. For example, the given figure neither show the data type of each data item nor the relationship among various files.

- In the database, actual data changes quite frequently. For example, in the given figure, the database changes whenever we add a new grade or add a student. The data at a particular moment of time is called the instance of the database.



Self-Check Exercise

1. What do you mean by Semi-structured data model ?
2. How many data models are there in DBMS?
3. Define Object-based data model.
4. Define database schema.

2.2 KEYS IN DBMS

It is an attribute or set of attributes which helps you to identify a row(tuple) in a relation(table). They allow you to find the relation between two tables. Keys help you uniquely identify a row in a table by a combination of one or more columns in that table. Key is also helpful for finding unique record or row from the table. Database key is also helpful for finding unique record or row from the table.

Example:

Employee ID	FirstName	LastName
11	Andrew	Johnson
22	Tom	Wood
33	Alex	Hale

In the above-given example, employee ID is a primary key because it uniquely identifies an employee record. In this table, no other employee can have the same employee ID.

2.3.1 Why do we need a Key?

Here are some reasons for using keys in the DBMS system.

- Keys help you to identify any row of data in a table. In a real-world application, a table could contain thousands of records. Moreover, the records could be duplicated. Keys in RDBMS ensure that you can uniquely identify a table record despite these challenges.
- It allows you to establish a relationship between and identify the relation between tables
- It helps you to enforce identity and integrity in the relationship.

2.4 TYPES OF KEYS IN DBMS

There are mainly Eight different types of Keys in DBMS and each key has it's different functionality:

1. Super Key
2. Primary Key
3. Candidate Key
4. Alternate Key
5. Foreign Key
6. Compound Key
7. Composite Key
8. Surrogate Key

Let's look at each of the keys in DBMS with example:

- **Super Key** - A super key is a group of single or multiple keys which identifies rows in a table.
- **Primary Key** - is a column or group of columns in a table that uniquely identify every row in that table.
- **Candidate Key** - is a set of attributes that uniquely identify tuples in a table. Candidate Key is a super key with no repeated attributes.
- **Alternate Key** - is a column or group of columns in a table that uniquely identify every row in that table.
- **Foreign Key** - is a column that creates a relationship between two tables. The purpose of Foreign keys is to maintain data integrity and allow navigation between two different instances of an entity.
- **Compound Key** - has two or more attributes that allow you to uniquely recognize a specific record. It is possible that each column may not be unique by itself within the database.

- **Composite Key** - is a combination of two or more columns that uniquely identify rows in a table. The combination of columns guarantees uniqueness, though individual uniqueness is not guaranteed.
- **Surrogate Key** - An artificial key which aims to uniquely identify each record is called a surrogate key. These kind of key are unique because they are created when you don't have any natural primary key.

2.4.1 Super Key

A superkey is a group of single or multiple keys which identifies rows in a table. A Super key may have additional attributes that are not needed for unique identification.

Example:

EmpSSN	EmpNum	Empname
9812345098	AB05	Shown
9876512345	AB06	Roslyn
199937890	AB07	James

In the above-given example, EmpSSN and EmpNum name are superkeys.

2.4.2 Primary Key

PRIMARY KEY in DBMS is a column or group of columns in a table that uniquely identify every row in that table. The Primary Key can't be a duplicate meaning the same value can't appear more than once in the table. A table cannot have more than one primary key.

Rules for defining Primary key:

- Two rows can't have the same primary key value
- It must for every row to have a primary key value.
- The primary key field cannot be null.
- The value in a primary key column can never be modified or updated if any foreign key refers to that primary key.

Example:

In the following example, `StudID` is a Primary Key.

StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	<u>abc@gmail.com</u>
2	12	Nick	Wright	<u>xyz@gmail.com</u>
3	13	Dana	Natan	<u>mno@yahoo.com</u>

2.4.3 Alternate Key

ALTERNATE KEY is a column or group of columns in a table that uniquely identify every row in that table. A table can have multiple choices for a primary key but only one can be set as the primary key. All the keys which are not primary key are called an Alternate Key.

Example:

In this table, StudID, Roll No, Email are qualified to become a primary key. But since StudID is the primary key, Roll No, Email becomes the alternative key.

StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	<u>abc@gmail.com</u>
2	12	Nick	Wright	<u>xyz@gmail.com</u>
3	13	Dana	Natan	<u>mno@yahoo.com</u>

2.4.4 Candidate Key

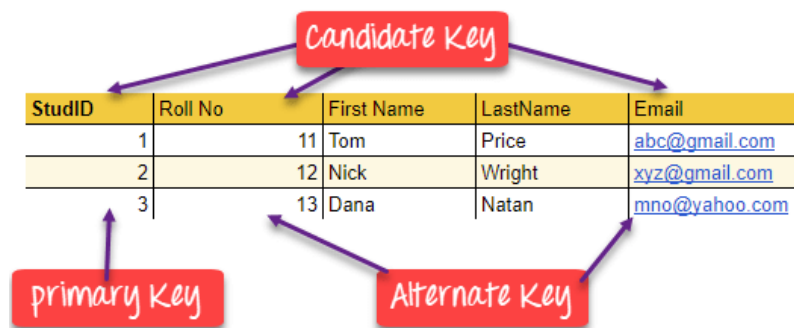
CANDIDATE KEY in SQL is a set of attributes that uniquely identify tuples in a table. Candidate Key is a super key with no repeated attributes. The Primary key should be selected from the candidate keys. Every table must have at least a single candidate key. A table can have multiple candidate keys but only a single primary key.

Properties of Candidate Key

- It must contain unique values
- Candidate key in SQL may have multiple attributes
- Must not contain null values
- It should contain minimum fields to ensure uniqueness
- Uniquely identify each record in a table

Candidate key Example: In the given table Stud ID, Roll No, and email are candidate keys which help us to uniquely identify the student record in the table.

StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	<u>abc@gmail.com</u>
2	12	Nick	Wright	<u>xyz@gmail.com</u>
3	13	Dana	Natan	<u>mno@yahoo.com</u>



Candidate Key in DBMS

2.4.5 Foreign Key

FOREIGN KEY is a column that creates a relationship between two tables. The purpose of Foreign keys is to maintain data integrity and allow navigation between two different instances of an entity. It acts as a cross-reference between two tables as it references the primary key of another table.

Example:

DeptCode	DeptName
001	Science
002	English
005	Computer

Teacher ID	Fname	Lname
B002	David	Warner
B017	Sara	Joseph
B009	Mike	Brunton

In this key in dbms example, we have two tables, teacher and department in a school. However, there is no way to see which search work in which department.

In this table, adding the foreign key in Deptcode to the Teacher name, we can create a relationship between the two tables.

Teacher ID	DeptCode	Fname	Lname
B002	002	David	Warner
B017	002	Sara	Joseph
B009	001	Mike	Brunton

This concept is also known as Referential Integrity.

2.4.6 Compound Key

COMPOUND KEY has two or more attributes that allow you to uniquely recognize a specific record. It is possible that each column may not be unique by itself within the database. However, when combined with the other column or columns the combination of composite keys become unique. The purpose of the compound key in database is to uniquely identify each record in the table.

Example:

OrderNo	ProductID	Product Name	Quantity
B005	JAP102459	Mouse	5
B005	DKT321573	USB	10
B005	OMG446789	LCD Monitor	20
B004	DKT321573	USB	15
B002	OMG446789	Laser Printer	3

In this example, OrderNo and ProductID can't be a primary key as it does not uniquely identify a record. However, a compound key of Order ID and Product ID could be used as it uniquely identified each record.

2.4.7 Composite Key

COMPOSITE KEY is a combination of two or more columns that uniquely identify rows in a table. The combination of columns guarantees uniqueness, though individually uniqueness is not guaranteed. Hence, they are combined to uniquely identify records in a table.

The difference between compound and the composite key is that any part of the compound key can be a foreign key, but the composite key may or maybe not a part of the foreign key.

2.4.8 Surrogate Key?

SURROGATE KEY is an artificial key which aims to uniquely identify each record is called a surrogate key. This kind of partial key in dbms is unique because it is created when you don't have any natural primary key. They do not lend any meaning to the data in the table. Surrogate key in DBMS is usually an integer. A surrogate key is a value generated right before the record is inserted into a table.

Fname	Lastname	Start Time	End Time
Anne	Smith	09:00	18:00

Jack	Francis	08:00	17:00
Anna	McLean	11:00	20:00
Shown	Willam	14:00	23:00

Above, given example, shown shift timings of the different employee. In this example, a surrogate key is needed to uniquely identify each employee.

Surrogate Keys in sql are Allowed When

- No property has the parameter of the primary key.
- In the table when the primary key is too big or complicated.

Self-Check Exercise

1. What do you mean by Key ?
2. How does key help you in relational database management system?
3. What is a Foreign key? Give example.
4. Define properties of Candidate key.
5. What is meant by Surrogate Key? Give example.

2.5 DIFFERENCE BETWEEN PRIMARY KEY & FOREIGN KEY

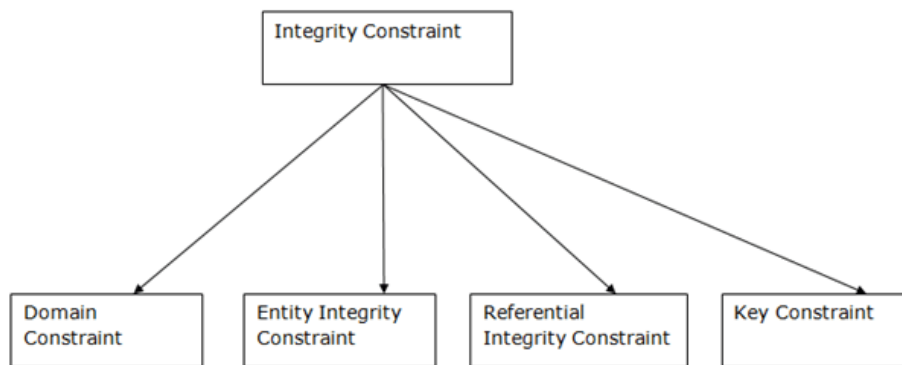
Following is the main difference between primary key and foreign key:

Primary Key	Foreign Key
Helps you to uniquely identify a record in the table.	It is a field in the table that is the primary key of another table.
Primary Key never accept null values.	A foreign key may accept multiple null values.
Primary key is a clustered index and data in the DBMS table are physically organized in the sequence of the clustered index.	A foreign key cannot automatically create an index, clustered or non-clustered. However, you can manually create an index on the foreign key.
You can have the single Primary key in a table.	You can have multiple foreign keys in a table.

2.6 INTEGRITY CONSTRAINTS

- Integrity constraints are a set of rules. It is used to maintain the quality of information.
- Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.
- Thus, integrity constraint is used to guard against accidental damage to the database.

2.6.1 Types of Integrity Constraints



Domain Constraints

- Domain constraints can be defined as the definition of a valid set of values for an attribute.
- The data type of domain includes string, character, integer, time, date, currency, etc. The value of the attribute must be available in the corresponding domain.

Example:

ID	NAME	SEMENSTER	AGE
1000	Tom	1 st	17
1001	Johnson	2 nd	24
1002	Leonardo	5 th	21
1003	Kate	3 rd	19
1004	Morgan	8 th	A

Not allowed. Because AGE is an integer attribute

2. Entity Integrity Constraints

- The entity integrity constraint states that primary key value can't be null.
- This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.
- A table can contain a null value other than the primary key field.

Example:

EMPLOYEE

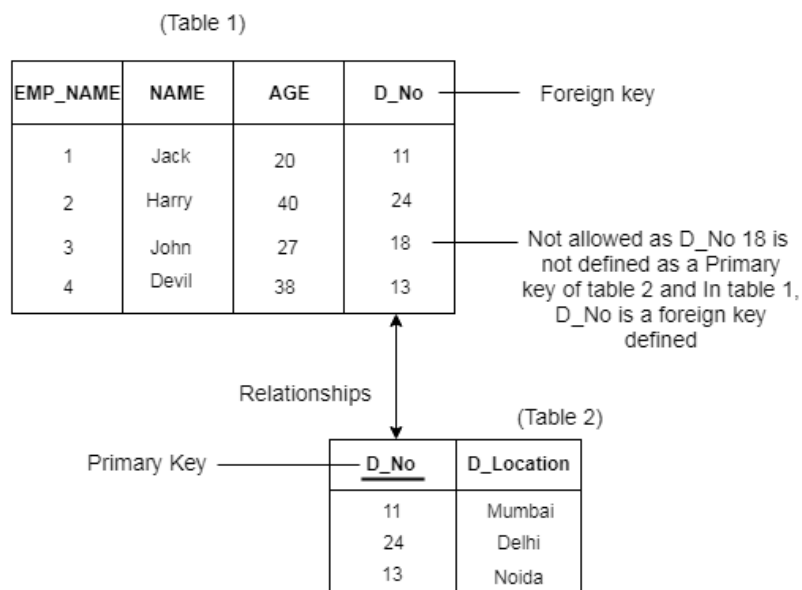
EMP_ID	EMP_NAME	SALARY
123	Jack	30000
142	Harry	60000
164	John	20000
	Jackson	27000

Not allowed as primary key can't contain a NULL value

3. Referential Integrity Constraints

- A referential integrity constraint is specified between two tables.
- In the Referential integrity constraints, if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be null or be available in Table 2.

Example:



4. Key Constraints

- Keys are the entity set that is used to identify an entity within its entity set uniquely.
- An entity set can have multiple keys, but out of which one key will be the primary key. A primary key can contain a unique and null value in the relational table.

Example

ID	NAME	SEMENSTER	AGE
1000	Tom	1 st	17
1001	Johnson	2 nd	24
1002	Leonardo	5 th	21
1003	Kate	3 rd	19
1002	Morgan	8 th	22

Not allowed. Because all row must be unique

2.7 RELATIONAL MODEL

- **Relational Model (RM)** represents the database as a collection of relations. A relation is nothing but a table of values. Every row in the table represents a collection of related data values. These rows in the table denote a real-world entity or relationship.
- The table name and column names are helpful to interpret the meaning of values in each row.
- The data are represented as a set of relations. In the relational model, data are stored as tables.
- However, the physical storage of the data is independent of the way the data are logically organized.

Some popular Relational Database management systems are:

- DB2 and Informix Dynamic Server - IBM
- Oracle and RDB – Oracle
- SQL Server and Access – Microsoft

2.7.1 Codd's 12 Rules of RDBMS

- Dr Edgar F. Codd, after his extensive research on the Relational Model of database systems, came up with twelve rules of his own, which according to him, a database must obey in order to be regarded as a true relational database.
- These rules can be applied on any database system that manages stored data using only its relational capabilities. This is a foundation rule, which acts as a base for all the other rules.

Rule 1: Information Rule

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

Rule 2: Guaranteed Access Rule

- Every single data element (value) is guaranteed to be accessible logically with a combination of table-name, primary-key (row value), and attribute-name (column value). No other means, such as pointers, can be used to access data.

Rule 3: Systematic Treatment of NULL Values

- The NULL values in a database must be given a systematic and uniform treatment. This is a very important rule because a NULL can be interpreted as one the following – data is missing, data is not known, or data is not applicable.

Rule 4: Active Online Catalog

- The structure description of the entire database must be stored in an online catalog, known as **data dictionary**, which can be accessed by authorized users. Users can use the same query language to access the catalog which they use to access the database itself.

Rule 5: Comprehensive Data Sub-Language Rule

- A database can only be accessed using a language having linear syntax that supports data definition, data manipulation, and transaction management operations. This language can be used directly or by means of some application. If the database allows access to data without any help of this language, then it is considered as a violation.

Rule 6: View Updating Rule

- All the views of a database, which can theoretically be updated, must also be updatable by the system.

Rule 7: High-Level Insert, Update, and Delete Rule

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

Rule 8: Physical Data Independence

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

Rule 9: Logical Data Independence

- The logical data in a database must be independent of its user's view (application). Any change in logical data must not affect the applications using it. For example, if two tables are merged or one is split into two different tables, there should be no

impact or change on the user application. This is one of the most difficult rule to apply.

Rule 10: Integrity Independence

- A database must be independent of the application that uses it. All its integrity constraints can be independently modified without the need of any change in the application. This rule makes a database independent of the front-end application and its interface.

Rule 11: Distribution Independence

- The end-user must not be able to see that the data is distributed over various locations. Users should always get the impression that the data is located at one site only. This rule has been regarded as the foundation of distributed database systems.

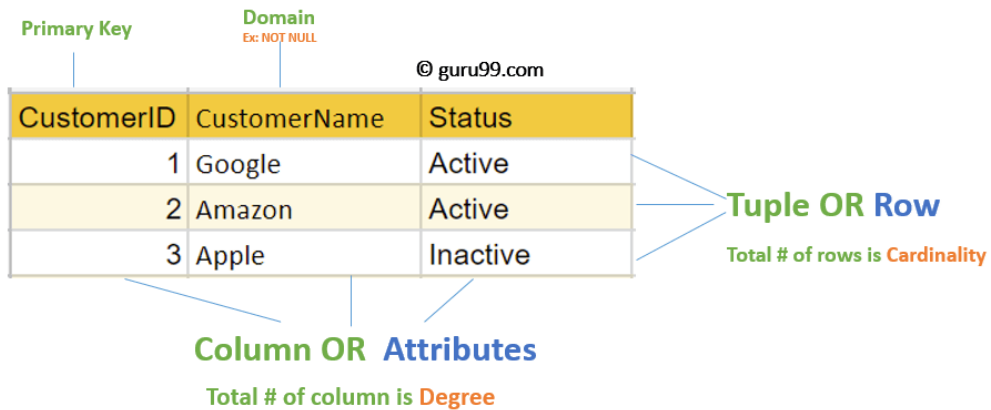
Rule 12: Non-Subversion Rule

- If a system has an interface that provides access to low-level records, then the interface must not be able to subvert the system and bypass security and integrity constraints.

2.7.2 Relational Model Concepts

1. **Attribute:** It contains the name of a column in a particular table. Attributes are the properties which define a relation. e.g., Student_Rollno, NAME,etc.
2. **Tables** – In the Relational model, the relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes.
3. **Tuple** – It is nothing but a single row of a table, which contains a single record.
4. **Relation Schema:** A relation schema represents the name of the relation and name of all columns or attributes.
5. **Degree:** The total number of attributes which in the relation is called the degree of the relation.
6. **Cardinality:** Total number of rows present in the Table.
7. **Column:** The column represents the set of values for a specific attribute.
8. **Relation instance** – Relation instance is a finite set of tuples in the RDBMS system. Relation instances never have duplicate tuples.
9. **Relation key** - In the relational key, each row has one or more attributes. It can identify the row in the relation uniquely.
10. **Attribute domain** – It contains a set of atomic values that an attribute can take.

Table also called Relation



Example: STUDENT Relation

NAME	ROLL_NO	PHONE_NO	ADDRESS	AGE
Ram	14795	7305758992	Noida	24
Shyam	12839	9026288936	Delhi	35
Laxman	33289	8583287182	Gurugram	20
Mahesh	27857	7086819134	Ghaziabad	27
Ganesh	17282	9028 9i3988	Delhi	40

- In the given table, NAME, ROLL_NO, PHONE_NO, ADDRESS, and AGE are the attributes.
- The instance of schema STUDENT has 5 tuples.
- t3 = <Laxman, 33289, 8583287182, Gurugram, 20>

2.7.3 Properties of Relations

- Name of the relation is distinct from all other relations.
- Each relation cell contains exactly one atomic (single) value
- Each attribute contains a distinct name
- Attribute domain has no significance
- tuple has no duplicate value
- Order of tuple can have a different sequence

2.7.4 Operations in Relational Model

Four basic update operations performed on relational database model are as follows:

Insert, update, delete and select.


- Insert is used to insert data into the relation
- Delete is used to delete tuples from the table.
- Modify allows you to change the values of some attributes in existing tuples.
- Select allows you to choose a specific range of data.

Whenever one of these operations are applied, integrity constraints specified on the relational database schema must never be violated.

Insert Operation

The insert operation gives values of the attribute for a new tuple which should be inserted into a relation.

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive



CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive
4	Alibaba	Active

Update Operation

You can see that in the below-given relation table CustomerName= 'Apple' is updated from Inactive to Active.

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive
4	Alibaba	Active




CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Active
4	Alibaba	Active

Delete Operation

To specify deletion, a condition on the attributes of the relation selects the tuple to be deleted.

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Active
4	Alibaba	Active



CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
4	Alibaba	Active

In the above-given example, CustomerName= "Apple" is deleted from the table.

The Delete operation could violate referential integrity if the tuple which is deleted is referenced by foreign keys from other tuples in the same database.

Select Operation

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
4	Alibaba	Active



CustomerID	CustomerName	Status
2	Amazon	Active

In the above-given example, CustomerName="Amazon" is selected

2.7.5 Equipment with a query language

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

2.7.5.1 Relational Algebra

Relational algebra is a procedural query language, which takes instances of relations as input and yields instances of relations as output.

It uses operators to perform queries. An operator can be either **unary** or **binary**. They accept relations as their input and yield relations as their output. Relational algebra is performed recursively on a relation and intermediate results are also considered relations.

The fundamental operations of relational algebra are as follows –

- Select
- Project
- Union
- Set difference
- Cartesian product
- Rename

We will discuss all these operations in the following sections.

- **Select Operation (σ)**

It selects tuples that satisfy the given predicate from a relation.

Notation – $\sigma_p(r)$

Where σ stands for selection predicate and r stands for relation. p is propositional logic formula which may use connectors like **and**, **or**, and **not**. These terms may use relational operators like $=, \neq, \geq, <, >, \leq$.

For example –

$\sigma_{\text{subject} = \text{"database"}}(\text{Books})$

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

$\sigma_{\text{subject} = \text{"database"} \text{ and } \text{price} = \text{"450"}}(\text{Books})$

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

$\sigma_{\text{subject} = \text{"database"} \text{ and } \text{price} = \text{"450"} \text{ or } \text{year} > \text{"2010"}}(\text{Books})$

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

- **Project Operation (Π)**

It projects column(s) that satisfy a given predicate.

Notation – $\Pi_{A_1, A_2, A_n}(r)$

Where A_1, A_2, A_n are attribute names of relation r .

Duplicate rows are automatically eliminated, as relation is a set.

For example –

$\Pi_{\text{subject, author}}(\text{Books})$

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

- **Union Operation (\cup)**

It performs binary union between two given relations and is defined as –

$r \cup s = \{ t \mid t \in r \text{ or } t \in s \}$

Notation – $r \cup s$

Where r and s are either database relations or relation result set (temporary relation).

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

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

$\Pi_{\text{author}}(\text{Books}) \cup \Pi_{\text{author}}(\text{Articles})$

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

- **Set Difference ($-$)**

The result of set difference operation is tuples, which are present in one relation but are not in the second relation.

Notation – $r - s$

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

$\Pi_{\text{author}}(\text{Books}) - \Pi_{\text{author}}(\text{Articles})$

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

- **Cartesian Product (X)**

Combines information of two different relations into one.

Notation – $r \times s$

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

$r \times s = \{ q \ t \mid q \in r \text{ and } t \in s \}$

$\sigma_{\text{author} = 'abc'}(\text{Books} \times \text{Articles})$

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

- **Rename Operation (ρ)**

The results of relational algebra are also relations but without any name. The rename operation allows us to rename the output relation. 'rename' operation is denoted with small Greek letter **rho** ρ .

Notation – $\rho_x(E)$

Where the result of expression **E** is saved with name of **x**.

Additional operations are –

- Set intersection
- Assignment
- Natural join

2.7.5.2 Relational Calculus

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

Relational calculus exists in two forms –

- **Tuple Relational Calculus (TRC)**

Filtering variable ranges over tuples

Notation – $\{T \mid \text{Condition}\}$

Returns all tuples T that satisfies a condition.

For example –

$\{ T.\text{name} \mid \text{Author}(T) \text{ AND } T.\text{article} = 'database' \}$

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

TRC can be quantified. We can use Existential (\exists) and Universal Quantifiers (\forall).

For example –

$\{ R \mid \exists T \in \text{Authors}(T.\text{article}='database' \text{ AND } R.\text{name}=T.\text{name}) \}$

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

- **Domain Relational Calculus (DRC)**

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

Notation –

$\{ a_1, a_2, a_3, \dots, a_n \mid P(a_1, a_2, a_3, \dots, a_n) \}$

Where a_1, a_2 are attributes and **P** stands for formulae built by inner attributes.

For example –

$\{ \langle \text{article}, \text{page}, \text{subject} \rangle \mid \in \text{abc} \wedge \text{subject} = 'database' \}$

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

Just like TRC, DRC can also be written using existential and universal quantifiers. DRC also involves relational operators.

The expression power of Tuple Relation Calculus and Domain Relation Calculus is equivalent to Relational Algebra.

2.7.6 ER Model to Relational Model

ER Model, when conceptualized into diagrams, gives a good overview of entity-relationship, which is easier to understand. ER diagrams can be mapped to relational schema, that is, it is possible to create relational schema using ER diagram. We cannot import all the ER constraints into relational model, but an approximate schema can be generated.

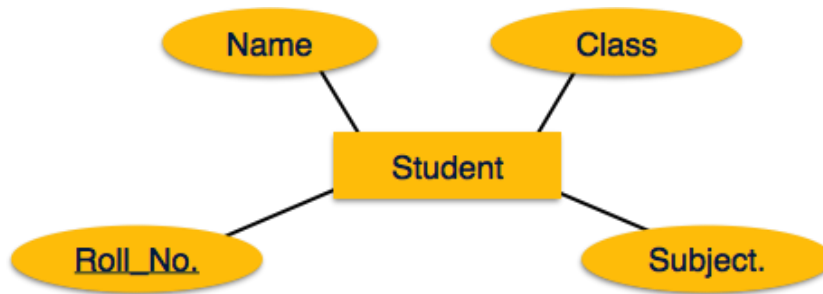
There are several processes and algorithms available to convert ER Diagrams into Relational Schema. Some of them are automated and some of them are manual. We may focus here on the mapping diagram contents to relational basics.

ER diagrams mainly comprise of –

- Entity and its attributes
- Relationship, which is association among entities.

2.7.6.1 Mapping Entity

An entity is a real-world object with some attributes.

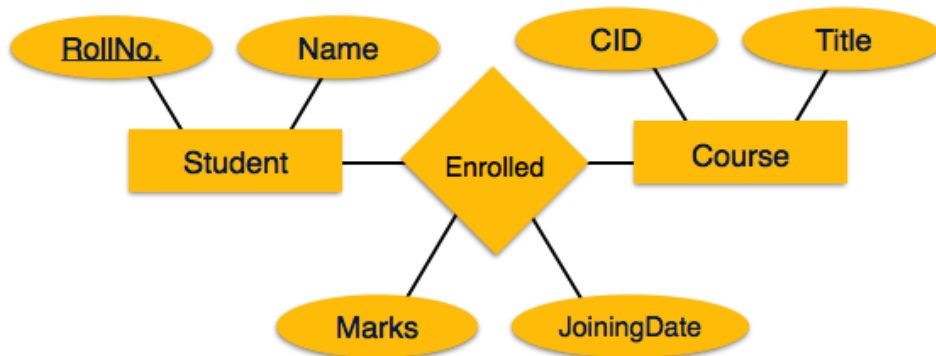


Mapping Process (Algorithm)

- Create table for each entity.
- Entity's attributes should become fields of tables with their respective data types.
- Declare primary key.

2.7.6.2 Mapping Relationship

A relationship is an association among entities.

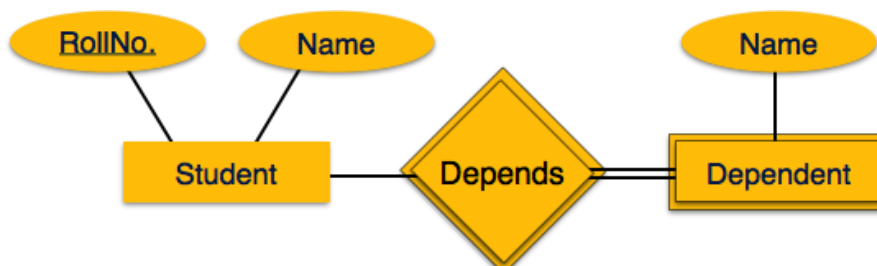


Mapping Process

- Create table for a relationship.
- Add the primary keys of all participating Entities as fields of table with their respective data types.
- If relationship has any attribute, add each attribute as field of table.
- Declare a primary key composing all the primary keys of participating entities.
- Declare all foreign key constraints.

2.7.6.3 Mapping Weak Entity Sets

A weak entity set is one which does not have any primary key associated with it.

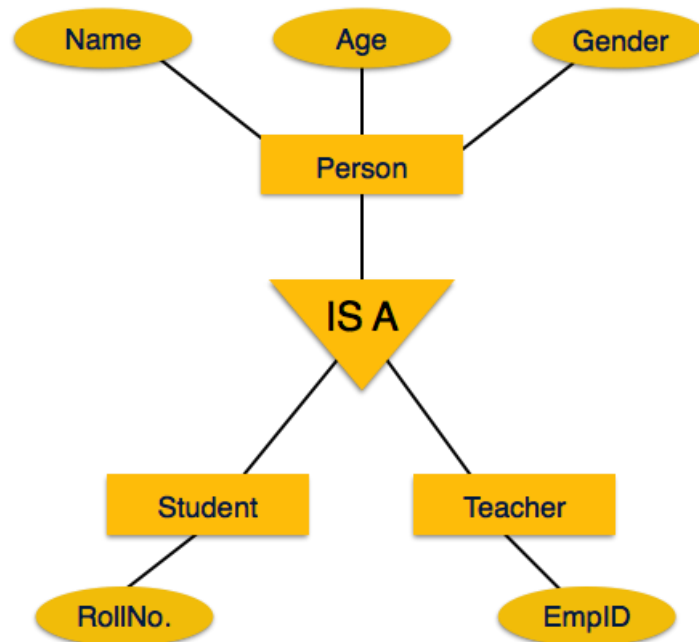


Mapping Process

- Create table for weak entity set.
- Add all its attributes to table as field.
- Add the primary key of identifying entity set.
- Declare all foreign key constraints.

2.7.6.4 Mapping Hierarchical Entities

ER specialization or generalization comes in the form of hierarchical entity sets.



Mapping Process

- Create tables for all higher-level entities.
- Create tables for lower-level entities.
- Add primary keys of higher-level entities in the table of lower-level entities.
- In lower-level tables, add all other attributes of lower-level entities.
- Declare primary key of higher-level table and the primary key for lower-level table.
- Declare foreign key constraints.

2.7.7 Best Practices for creating a Relational Model

- Data needs to be represented as a collection of relations
- Each relation should be depicted clearly in the table
- Rows should contain data about instances of an entity
- Columns must contain data about attributes of the entity
- Cells of the table should hold a single value
- Each column should be given a unique name

- No two rows can be identical
- The values of an attribute should be from the same domain

2.7.8 Advantages of using Relational Model

- **Simplicity:** A Relational data model in DBMS is simpler than the hierarchical and network model.
- **Structural Independence:** The relational database is only concerned with data and not with a structure. This can improve the performance of the model.
- **Easy to use:** The Relational model in DBMS is easy as tables consisting of rows and columns are quite natural and simple to understand
- **Query capability:** It makes possible for a high-level query language like SQL to avoid complex database navigation.
- **Data independence:** The Structure of Relational database can be changed without having to change any application.
- **Scalable:** Regarding a number of records, or rows, and the number of fields, a database should be enlarged to enhance its usability.

2.7.9 Disadvantages of using Relational Model

- Few relational databases have limits on field lengths which can't be exceeded.
- Relational databases can sometimes become complex as the amount of data grows, and the relations between pieces of data become more complicated.
- Complex relational database systems may lead to isolated databases where the information cannot be shared from one system to another.

Self-Check Exercise

1. Explain the concept of ER Model to Relational Model.
2. What is meant by the rule of distribution independence.
3. What is cardinality in relational model.
4. How can you describe relation schema?
5. Write the difference between relation instance and relation key.

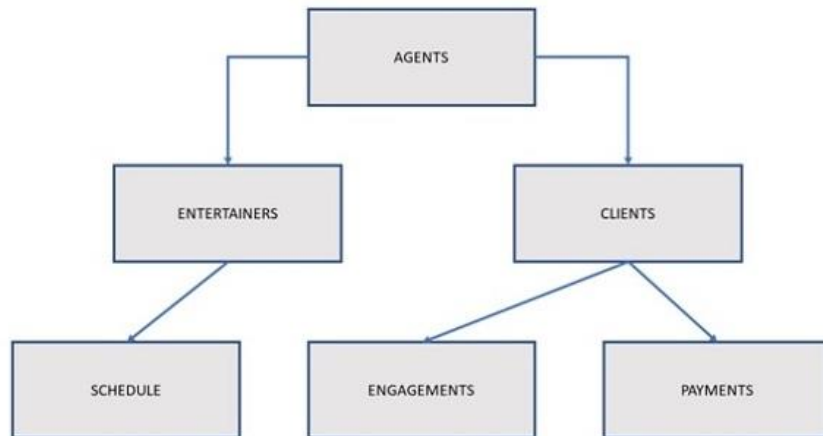
2.8 HIERARCHICAL DATA MODEL

A hierarchical model represents the data in a tree-like structure in which there is a single parent for each record. To maintain order there is a sort field which keeps sibling nodes into a recorded manner. These types of models are designed basically for the early mainframe database management systems, like the Information Management System (IMS) by IBM.

This model structure allows the one-to-one and a one-to-many relationship between two/ various types of data. This structure is very helpful in describing many relationships in the real world; table of contents, any nested and sorted information.

The hierarchical structure is used as the physical order of records in storage. One can access the records by navigating down through the data structure using pointers which are combined with sequential accessing. Therefore, the hierarchical structure is not suitable for certain database operations when a full path is not also included for each record.

Data in this type of database is structured hierarchically and is typically developed as an inverted tree. The "root" in the structure is a single table in the database and other tables act as the branches flowing from the root. The diagram below shows a typical hierarchical database structure.



Agents Database

In the above diagram, an agent books several entertainers, and each entertainer, in return has his/her own schedule. It is the duty of an agent to maintain several clients whose entertainment needs are to be met. A client books engagement through the agent and makes payments to the agent for his services.

A relationship in this database model is represented by the term parent/child. A parent table can be linked with one or more child tables in this type of relationship, but a single child table can be linked with only one parent table. The tables are explicitly linked via a pointer/index or by the physical arrangement of the records within the tables.

A user can access the data by starting at the root table and working down through the tree to the target data. The user must be familiar with the structure of the database to access the data without any complexity.

Advantages

- A user can retrieve data very quickly due to the presence of explicit links between the table structures.
- The referential integrity is built in and automatically enforced due to which a record in a child table must be linked to an existing record in a parent table, along with that if a record deleted in the parent table then that will cause all associated records in the child table to be deleted as well.

Disadvantages

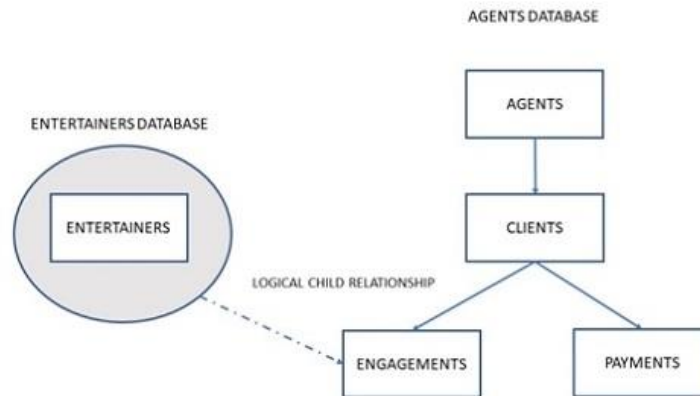
- When a user needs to store a record in a child table that is currently unrelated to any record in a parent table, it gets difficult in recording and user must record an additional entry in the parent table.
- This type of database cannot support complex relationships, and there is also a problem of redundancy, which can result in producing inaccurate information due to the inconsistent recording of data at various sites.

Consider an example using the database diagram shown in the previous diagram. A user cannot enter a new record for the entertainer in the Entertainers table until the entertainer is assigned to a specific agent in the Agents table since a record in a child table (Entertainers) must be related to a record in the parent table (Agents). Therefore, this type of database suffers from the problem of redundant data. For example, if there is a many-to-many relationship between clients and entertainers; an entertainer will perform for many clients, and a client will hire many entertainers. This type of relationship in a hierarchical database cannot easily model, so developers must introduce redundant data into both the Schedule and Engagements tables.

- The Schedule table will now have client data which contains information such as client name, address, and phone number to show for whom and where each entertainer is performing. This data is redundant because it is currently stored also in the Clients table.
- The Engagements table will now contain data on entertainers which contains information such as entertainer name, phone number, and type of entertainer to indicate which entertainers are performing for a given client. This data is also redundant because it is currently stored in the Entertainers table.

The problem with this redundancy is that it can result in producing inaccurate information because it opens the possibility of allowing a user to enter a single piece of data inconsistently.

This problem can be solved by creating one hierarchical database specifically for entertainers and another one specifically for agents. The Entertainers database will contain only the data recorded in the Entertainers table, and the revised Agents database will contain the data recorded in Agents, Clients, Payments, and Engagements tables. There is no need of as you can define a logical child relationship between the Engagements table in the Agents database and the Entertainers table in the Entertainers database. With this relationship in place, you can retrieve a variety of information, such as a list of booked entertainers for a given client or a performance schedule for a given entertainer. The below diagram describes the whole picture.



The hierarchical database suited well to the tape storage systems which is used by mainframes in the 1970s and was very popular in organizations whose database is based on those systems. But, even though the hierarchical database provided fast and direct access to data and was useful in several circumstances, it was clear that a new database model was needed to address the growing problems of data redundancy and complex relationships among data.

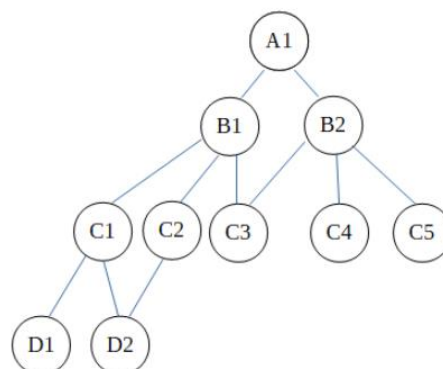
The idea behind this database model is useful for a certain type of data storage, but it is not extremely versatile and is confined to some specific uses.

For example, where each individual person in a company may report to a given department, the department can be used as a parent record and the individual employees will represent secondary records, each of which links back to that one parent record in a hierarchical structure.

2.9 NETWORK DATA MODEL

The network database model was created to solve the shortcomings of the hierarchical database model. In this type of model, a child can be linked to multiple parents, a feature that was not supported by the hierarchical data model. The parent nodes are known as owners and the child nodes are called members.

The network data model can be represented as –



Advantages of Network Model

The network model can support many to many relationships as seen in the diagram. D2 and C3 each have multiple masters. The masters for D2 are C1 and C2 while for C3 are B1 and B2. In this way, the network data model can handle many to many relationships where the hierarchical data model didn't.

Disadvantages of Network Model

There are some disadvantages in the network model even though it is an improvement over the hierarchical model. These are –

- The network model is much more complicated than the Hierarchical model. As such, it is difficult to handle and maintain.
- Although the Network model is more flexible than the Hierarchical model, it still has flexibility problems. Not all relations can be handled by assigning them in the form of owners and members.
- The structure of the Network Model is quite complicated and so the programmer has to understand it well in order to implement or modify it.

2.10 Difference between Hierarchical, Network and Relational Data Model :

Hierarchical Data Model	Network Data Model	Relational Data Model
In this model, to store data hierarchy method is used. It is the oldest method and not in use today.	It organizes records to one another through links or pointers.	It organizes records in the form of table and relationship between tables are set using common fields.
To organize records, it uses tree structure.	It organizes records in the form of directed graphs.	It organizes records in the form of tables.
It implements 1:1 and 1:n relations.	In addition to 1:1 and 1:n it also implements many to many relationships.	In addition to 1:1 and 1:n it also implements many to many relationships.
Insertion anomaly exists in this model i.e. child node cannot be inserted without the parent node.	There is no insertion anomaly.	There is no insertion anomaly.

Hierarchical Data Model	Network Data Model	Relational Data Model
Deletion anomaly exists in this model i.e. it is difficult to delete the parent node.	There is no deletion anomaly.	There is no deletion anomaly.
This model lacks data independence.	There is partial data independence in this model.	This model provides data independence.
It is used to access the data which is complex and asymmetric.	It is used to access the data which is complex and symmetric.	It is used to access the data which is complex and symmetric.
&XML and XAML use this model.	VAX-DBMS, DMS-1100 of UNIVAC and SUPRADBMS's use this model.	It is mostly used in real world applications. Oracle, SQL.

2.11 SUMMARY

- A key in DBMS is an attribute or set of attributes which helps you to identify a row(tuple) in a relation(table)
- Keys in RDBMS allow you to establish a relationship between and identify the relation between tables
- Eight types of key in DBMS are Super, Primary, Candidate, Alternate, Foreign, Compound, Composite, and Surrogate Key.
- A super key is a group of single or multiple keys which identifies rows in a table.
- A column or group of columns in a table which helps us to uniquely identifies every row in that table is called a primary key
- All the different keys in DBMS which are not primary key are called an alternate key
- A super key with no repeated attribute is called candidate key
- A compound key is a key which has many fields which allow you to uniquely recognize a specific record
- A key which has multiple attributes to uniquely identify rows in a table is called a composite key
- An artificial key which aims to uniquely identify each record is called a surrogate key
- Primary Key never accept null values while a foreign key may accept multiple null values.

- The Relational database modelling represents the database as a collection of relations (tables)
- Attribute, Tables, Tuple, Relation Schema, Degree, Cardinality, Column, Relation instance, are some important components of Relational Model
- Relational Integrity constraints are referred to conditions which must be present for a valid Relation approach in DBMS
- Domain constraints can be violated if an attribute value is not appearing in the corresponding domain or it is not of the appropriate data type
- Insert, Select, Modify and Delete are the operations performed in Relational Model constraints
- The relational database is only concerned with data and not with a structure which can improve the performance of the model
- Advantages of Relational model in DBMS are simplicity, structural independence, ease of use, query capability, data independence, scalability, etc.
- Few relational databases have limits on field lengths which can't be exceeded.
- A hierarchical model represents the data in a tree-like structure in which there is a single parent for each record.
- The network database model was created to solve the shortcomings of the hierarchical database model.

2.12 REFERENCES

“Fundamentals of Database Systems”, Elmasri Navrate, 6th edition, 2013, Pearson

“Introduction to Database Systems”, C.J.Date, Pearson Education

“Data base System Concepts”, Silberschatz, Korth, McGraw Hill, V edition

Starting with Oracle, by John Day and Craig Van Slyke

Fundamentals of Database Systems, by Navathe

2.13 FURTHER READING

Database System Concepts, by Henry F. Korth

“Database Systems, The Complete Book”, Hector Garcia-Molina, Jeffrey D. Ullman and Jennifer Widom, 6th impression, 2011, Pearson

“Data base Management Systems”, Raghu Rama Krishnan, Johannes Gehrke, 3rd Edition, 2003, McGraw Hill

2.14 QUESTIONS FOR PRACTICE (SHORT & LONG ANSWERS TYPE)

1. What do you mean by data model? Explain its types.
2. What is the difference between semi-structured data model and object-oriented data model?
3. What is a primary key? Give example.

4. What do you understand by integrity constraints?
5. How can you differentiate primary key from foreign key?
6. List the best practices for creating a Relational Model.
7. Explain the types of integrity constraints.
8. List the advantages of relational model.
9. Describe concept of hierarchical model.
10. What is the difference between relational algebra and relational calculus?
11. Explain the concept of ER Model to Relational Model.

B.Sc.(DATA SCIENCE)
SEMESTER-III
DATABASE MANAGEMENT SYSTEM

UNIT III: CONCEPTUAL DATA MODELING USING E-R DATA MODEL

STRUCTURE

- 3.0 Objectives
- 3.1 Introduction
- 3.2 ER Model
 - 3.2.1 Components of ER Diagram
 - 3.2.1.1 Entity
 - 3.2.1.2 Attribute
 - 3.2.1.3 Relationship
 - 3.2.2 Generalization
 - 3.2.3 Specialization
- 3.3 Specifying Constraints
- 3.4 Conversion of ER diagram into tables
- 3.5 Practice Problems Based on Converting ER diagram to tables
- 3.6 Summary
- 3.7 References
- 3.8 Further Readings
- 3.9 Questions for Practice (Short & Long Answers Type)

3.0 OBJECTIVES

- To provide overview of ER Model.
- To elaborate components of ER Diagram.
- To provide description of entity, attribute, relationship.
- To explain constraints.
- To discuss the concept of Generalization.
- To discuss the concept of Specialization.
- To elaborate conversion of ER diagram into tables.
- To practice problems based on converting ER diagrams into tables.

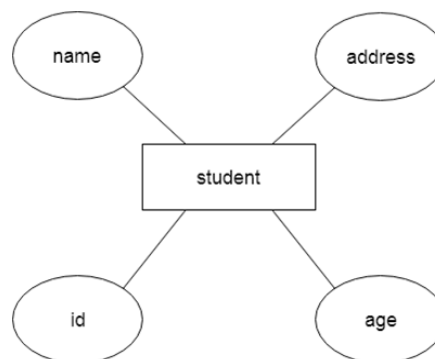
3.1 INTRODUCTION

Entity-Relationship Model describes an overall and conceptual view of the organization of data. This unit describes how the data in a database is represented using E-R model. It discusses components of ER diagram that includes entity, attribute as well as relationship. In this unit, we also discuss the concept of Generalization and Specialization. It also describes various constraints such as Not Null, Unique, domain, mapping etc. This unit also provides the Conversion of ER Models to Tables followed by practical problems based on E-R data model.

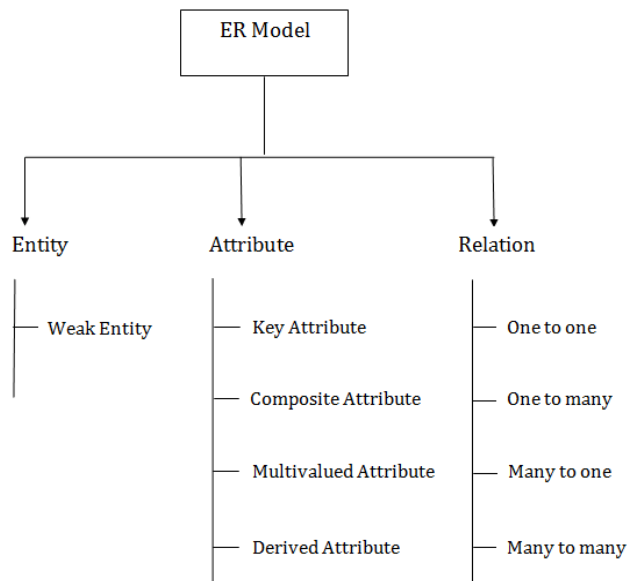
3.2 ER MODEL

- ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
- In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

For example: Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age, etc. The address can be another entity with attributes like city, street name, pin code, etc and there will be a relationship between them.



3.2.1 Component of ER Diagram



3.2.1.1 Entity

An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



3.2.1.1.1 Weak Entity

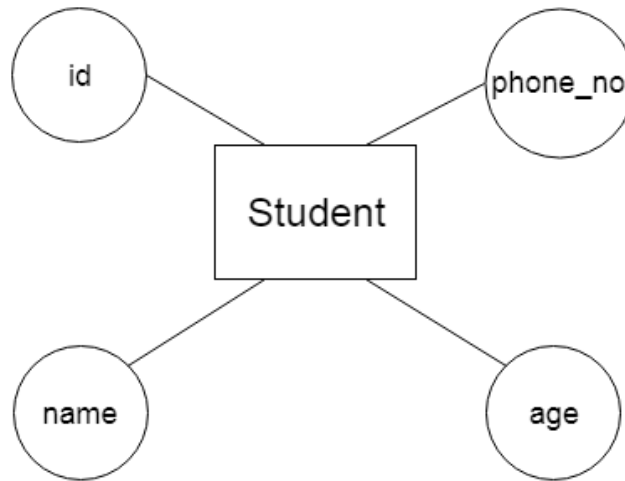
An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



3.2.1.2 Attribute

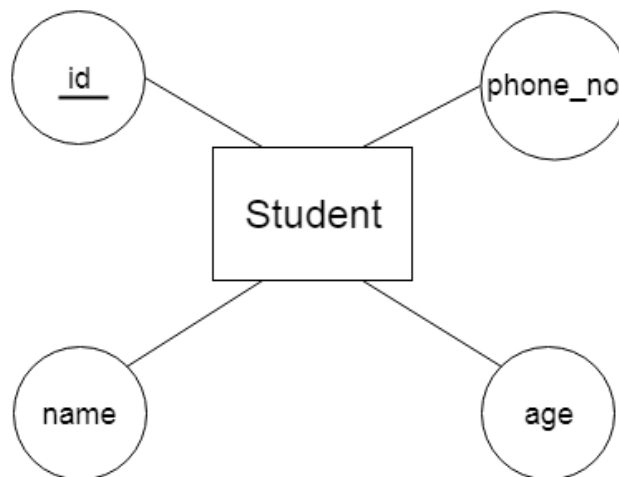
An attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

For example, id, age, contact number, name, etc. can be attributes of a student.



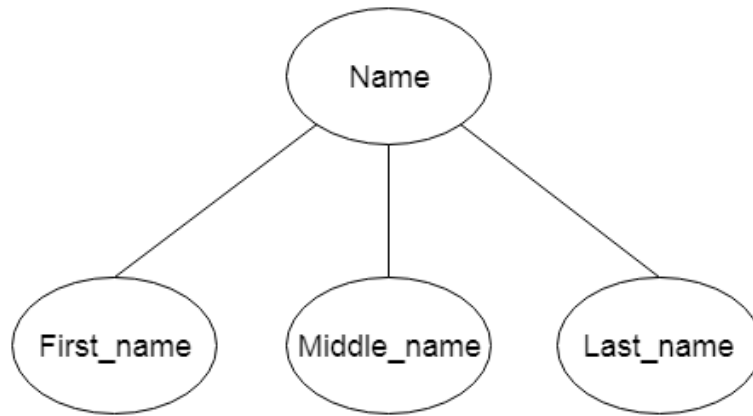
3.2.1.2.1 Key Attribute

The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.



3.2.1.2.2 Composite Attribute

An attribute composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



3.2.1.2.3 Multivalued Attribute

An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent a multivalued attribute.

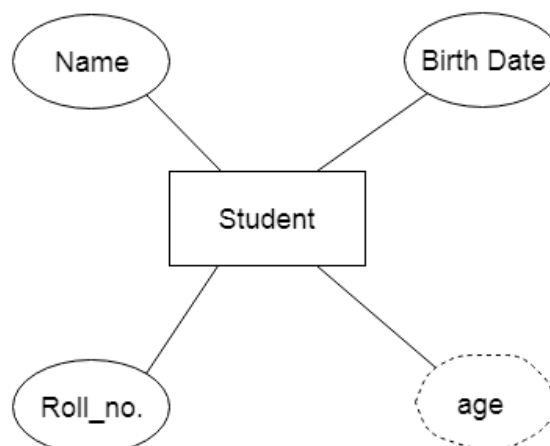
For example, a student can have more than one phone number.



3.2.1.2.4 Derived Attribute

An attribute that can be derived from another attribute is known as a derived attribute. It can be represented by a dashed ellipse.

For example, A person's age changes over time and can be derived from another attribute like Date of birth.



3.2.1.3 Relationship

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.



Types of Relationship are as follows

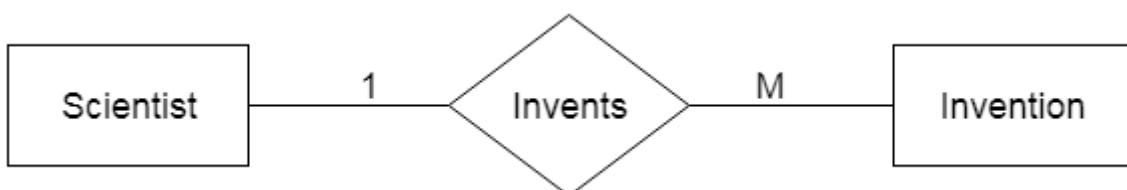
3.2.1.3.1 One-to-One Relationship: When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

For example, A female can marry to one male, and a male can marry to one female.



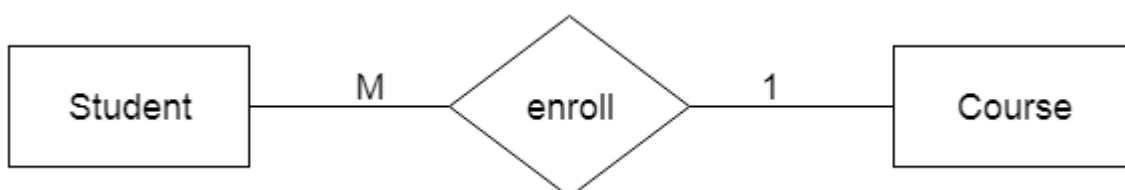
3.2.1.3.2 One-to-many relationship: When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

For example, scientists can invent many inventions, but the invention is done by the only specific scientist.



3.2.1.3.3 Many-to-one relationship: When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

For example, Student enrolls for only one course, but a course can have many students.



3.2.1.3.4 Many-to-many relationship: When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

For example, employees can assign by many projects and project can have many employees.



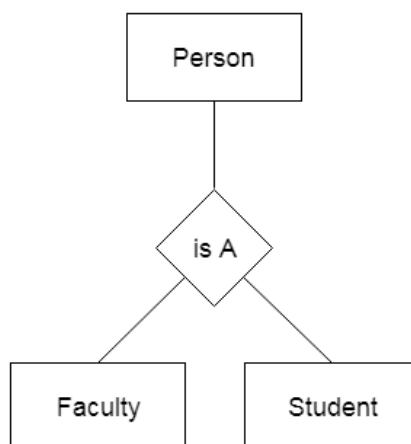
Self-Check Exercise

1. What do you mean by entity?
2. Why do you use key attribute? Give example.
3. Explain various terms of relationship.
4. Describe derived attribute. How can you represent it?

3.2.2 Generalization

- Generalization is like a bottom-up approach in which two or more entities of lower level combine to form a higher level entity if they have some attributes in common.
- In generalization, an entity of a higher level can also combine with the entities of the lower level to form a further higher level entity.
- Generalization is more like subclass and superclass system, but the only difference is the approach. Generalization uses the bottom-up approach.
- In generalization, entities are combined to form a more generalized entity, i.e., subclasses are combined to make a superclass.

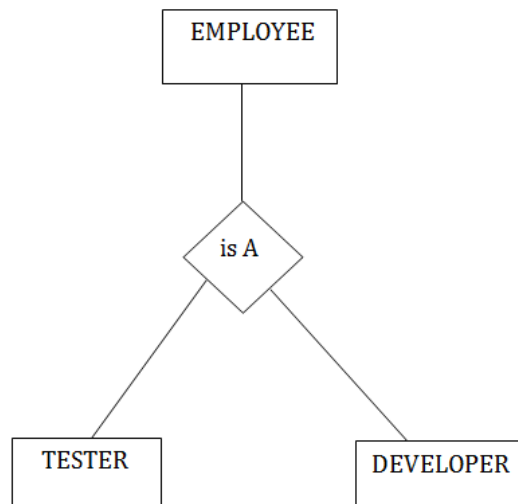
For example, Faculty and Student entities can be generalized and create a higher level entity Person.



3.2.3 Specialization

- Specialization is a top-down approach, and it is opposite to Generalization. In specialization, one higher level entity can be broken down into two lower level entities.
- Specialization is used to identify the subset of an entity set that shares some distinguishing characteristics.
- Normally, the superclass is defined first, the subclass and its related attributes are defined next, and relationship set are then added.

For example: In an Employee management system, EMPLOYEE entity can be specialized as TESTER or DEVELOPER based on what role they play in the company.



3.3 SPECIFYING CONSTRAINTS

Constraints enforce limits to the data or type of data that can be inserted/updated/deleted from a table. The whole purpose of constraints is to maintain the **data integrity** during an update/delete/insert into a table. Following are the types of various constraints:

- NOT NULL Constraints
- UNIQUE Constraints
- DEFAULT Constraints
- CHECK Constraints
- Key Constraints – PRIMARY KEY, FOREIGN KEY
- Domain Constraint
- Mapping Constraint

3.3.1 NOT NULL Constraint

NOT NULL constraint makes sure that a column does not hold NULL value. When we don't provide value for a particular column while inserting a record into a table, it takes NULL value by default. By specifying NULL constraint, we can be sure that a particular column(s) cannot have NULL values.

Example:

```
CREATE TABLE STUDENT(  
ROLL_NO INT NOT NULL,  
STU_NAME VARCHAR (35) NOT NULL,  
STU_AGE INT NOT NULL,  
STU_ADDRESS VARCHAR (235),  
PRIMARY KEY (ROLL_NO)  
);
```

3.3.2 UNIQUE Constraint

UNIQUE Constraint enforces a column or set of columns to have unique values. If a column has a unique constraint, it means that particular column cannot have duplicate values in a table.

Example:

```
CREATE TABLE STUDENT(  
ROLL_NO INT NOT NULL,  
STU_NAME VARCHAR (35) NOT NULL UNIQUE,  
STU_AGE INT NOT NULL,  
STU_ADDRESS VARCHAR (35) UNIQUE,  
PRIMARY KEY (ROLL_NO)  
);
```

3.3.3 DEFAULT Constraint

The DEFAULT constraint provides a default value to a column when there is no value provided while inserting a record into a table.

Example:

```
CREATE TABLE STUDENT(  
ROLL_NO INT NOT NULL,  
STU_NAME VARCHAR (35) NOT NULL,  
STU_AGE INT NOT NULL,  
EXAM_FEE INT DEFAULT 10000,  
STU_ADDRESS VARCHAR (35),  
PRIMARY KEY (ROLL_NO)  
);
```

3.3.4 CHECK Constraint

This constraint is used for specifying range of values for a particular column of a table. When this constraint is being set on a column, it ensures that the specified column must have the value falling in the specified range.

Example:

```
CREATE TABLE STUDENT(  
ROLL_NO INT NOT NULL CHECK(ROLL_NO >1000) ,  
STU_NAME VARCHAR (35) NOT NULL,  
STU_AGE INT NOT NULL,  
EXAM_FEE INT DEFAULT 10000,  
STU_ADDRESS VARCHAR (35) ,  
PRIMARY KEY (ROLL_NO)  
);
```

In the above example we have set the check constraint on ROLL_NO column of STUDENT table. Now, the ROLL_NO field must have the value greater than 1000.

3.3.5 Key Constraints

It contains two further Key constraints: Primary and Foreign Key.

3.3.5.1 PRIMARY KEY:

Primary key uniquely identifies each record in a table. It must have unique values and cannot contain nulls. In the below example the ROLL_NO field is marked as primary key, that means the ROLL_NO field cannot have duplicate and null values.

Example:

```
CREATE TABLE STUDENT(  
ROLL_NO INT NOT NULL,  
STU_NAME VARCHAR (35) NOT NULL UNIQUE,  
STU_AGE INT NOT NULL,  
STU_ADDRESS VARCHAR (35) UNIQUE,  
PRIMARY KEY (ROLL_NO)  
);
```

3.3.5.2 FOREIGN KEY:

Foreign keys are the columns of a table that points to the primary key of another table. They act as a cross-reference between tables.

For example:

In the below example, the Stu_Id column in Course_enrollment table is a foreign key as it points to the primary key of the Student table.

Course_enrollment table:

Course_Id	Stu_Id
C01	101
C02	102
C03	101
C05	102
C06	103
C07	102

Student table:

Stu_Id	Stu_Name	Stu_Age
101	Chaitanya	22
102	Arya	26
103	Bran	25
104	Jon	21

3.3.6 Domain Constraint

Each table has certain set of columns and each column allows a same type of data, based on its data type. The column does not accept values of any other data type.

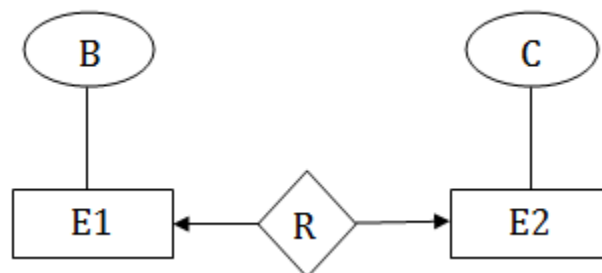
Domain Constraint = data type + Constraints (NOT NULL / UNIQUE / PRIMARY KEY / FOREIGN KEY / CHECK / DEFAULT)

3.3.7 Mapping Constraint

- A mapping constraint is a data constraint that expresses the number of entities to which another entity can be related via a relationship set.
- It is most useful in describing the relationship sets that involve more than two entity sets.
- For binary relationship set R on an entity set A and B, there are four possible mapping cardinalities. These are as follows:
 1. One to one (1:1)
 2. One to many (1:M)
 3. Many to one (M:1)
 4. Many to many (M:M)

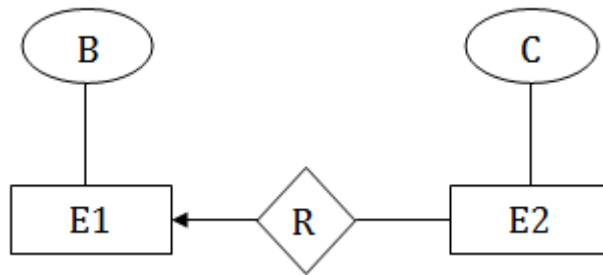
3.3.7.1 One-to-one

In one-to-one mapping, an entity in E1 is associated with at most one entity in E2, and an entity in E2 is associated with at most one entity in E1.



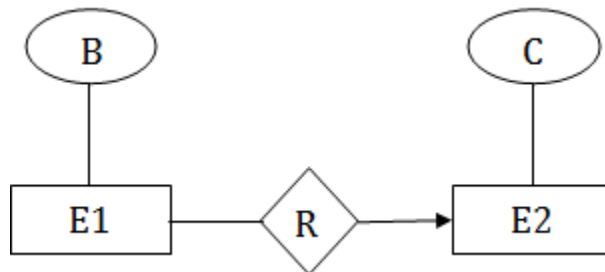
3.3.7.2 One-to-many

In one-to-many mapping, an entity in E1 is associated with any number of entities in E2, and an entity in E2 is associated with at most one entity in E1.



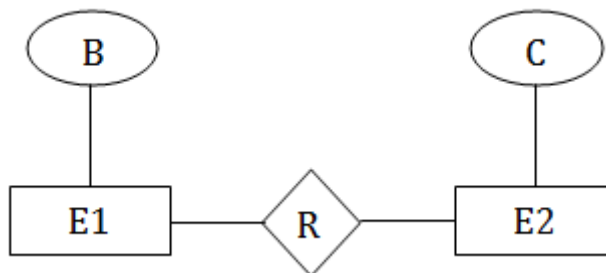
3.3.7.3 Many-to-one

In one-to-many mapping, an entity in E1 is associated with at most one entity in E2, and an entity in E2 is associated with any number of entities in E1.



3.3.7.4 Many-to-many

In many-to-many mapping, an entity in E1 is associated with any number of entities in E2, and an entity in E2 is associated with any number of entities in E1.



You can have these constraints in place while creating tables in database.

Example:

```
CREATE TABLE Customer (
customer_id int PRIMARY KEY NOT NULL,
```

```

first_name varchar(20),
last_name varchar(20)
);

CREATE TABLE Order (
order_id int PRIMARY KEY NOT NULL,
customer_id int,
order_details varchar(50),
constraint fk_Customers foreign key (customer_id)
references dbo.Customer
);

```

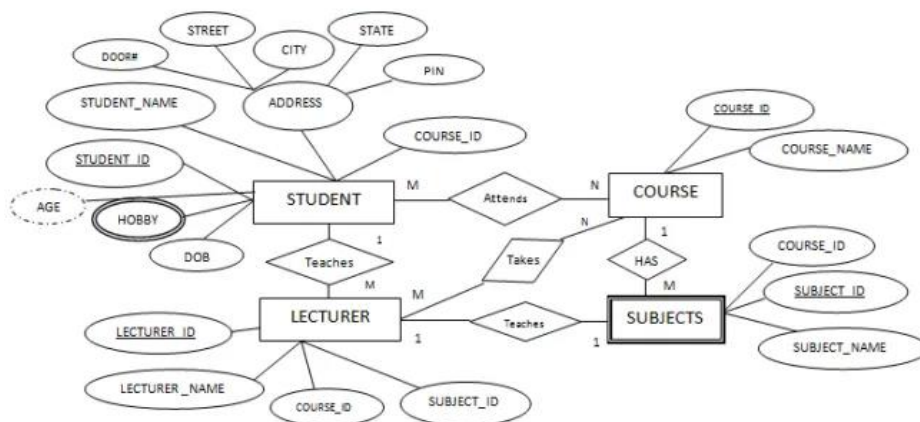
Assuming, that a customer orders more than once, the above relation represents **one to many** relation. Similarly you can achieve other mapping constraints based on the requirements.

Self-Check Exercise

1. What do you understand by generalization? Explain.
2. What is meant by specialization? Explain.
3. How can you differentiate domain constraint from Not Null constraint?
4. How can you describe unique constraint? Give example.
5. What do you mean by mapping constraint? Explain with example.

3.4 CONVERSION OF ER DIAGRAM INTO TABLES

There are various steps involved in converting it into tables and columns. Each type of entity, attribute and relationship in the diagram takes their own depiction. Consider the below ER diagram and it will be converted into tables, columns and mappings.



The basic rule for converting the ER diagrams into tables is as follows:

- Convert all the Entities in the diagram to tables.

- All the entities represented in the rectangular box in the ER diagram become independent tables in the database. In the below diagram, STUDENT, COURSE, LECTURER and SUBJECTS forms individual tables.
- All single valued attributes of an entity is converted to a column of the table

All the attributes, whose value at any instance of time is unique, are considered as columns of that table. In the STUDENT Entity, STUDENT_ID, STUDENT_NAME form the columns of STUDENT table. Similarly, LECTURER_ID, LECTURER_NAME form the columns of LECTURER table. And so on.

- Key attribute in the ER diagram becomes the Primary key of the table.

In diagram above, STUDENT_ID, LECTURER_ID, COURSE_ID and SUB_ID are the key attributes of the entities. Hence we consider them as the primary keys of respective table.

- Declare the foreign key column, if applicable.

In the diagram, attribute COURSE_ID in the STUDENT entity is from COURSE entity. Hence add COURSE_ID in the STUDENT table and assign it foreign key constraint. COURSE_ID and SUBJECT_ID in LECTURER table forms the foreign key column. Hence by declaring the foreign key constraints, mapping between the tables are established.

- Any multi-valued attributes are converted into new table.

A hobby in the Student table is a multivalued attribute. Any student can have any number of hobbies. So we cannot represent multiple values in a single column of STUDENT table. We need to store it separately, so that we can store any number of hobbies, adding/ removing / deleting hobbies should not create any redundancy or anomalies in the system. Hence we create a separate table STUD_HOBBY with STUDENT_ID and HOBBY as its columns. We create a composite key using both the columns.

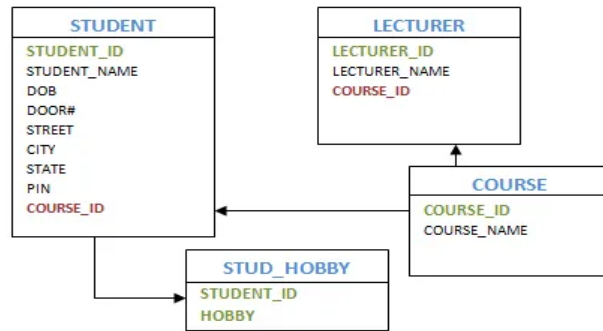
- Any composite attributes are merged into same table as different columns.

In the diagram above, Student Address is a composite attribute. It has Door#, Street, City, State and Pin. These attributes are merged into the STUDENT table as individual columns.

- One can ignore the derived attribute, since it can be calculated at any time.

In the STUDENT table, Age can be derived at any point of time by calculating the difference between DateOfBirth and current date. Hence we need not create a column for this attribute. It reduces the duplicity in the database.

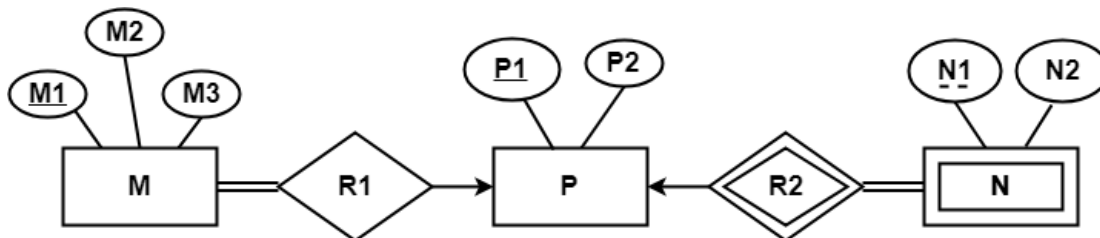
These are the very basic rules of converting ER diagram into tables and columns, and assigning the mapping between the tables. Table structure at this would be as below:



3.5 PRACTICE PROBLEMS

Problem-01:

Find the minimum number of tables required for the following ER diagram in relational model-



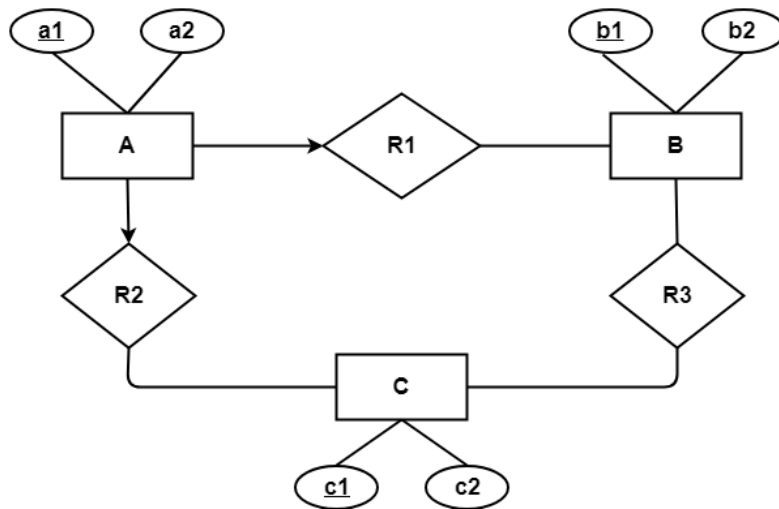
Solution-

Applying the rules, minimum 3 tables will be required-

- MR1 (M1 , M2 , M3 , P1)
- P (P1 , P2)
- NR2 (P1 , N1 , N2)

Problem-02:

Find the minimum number of tables required to represent the given ER diagram in relational model-



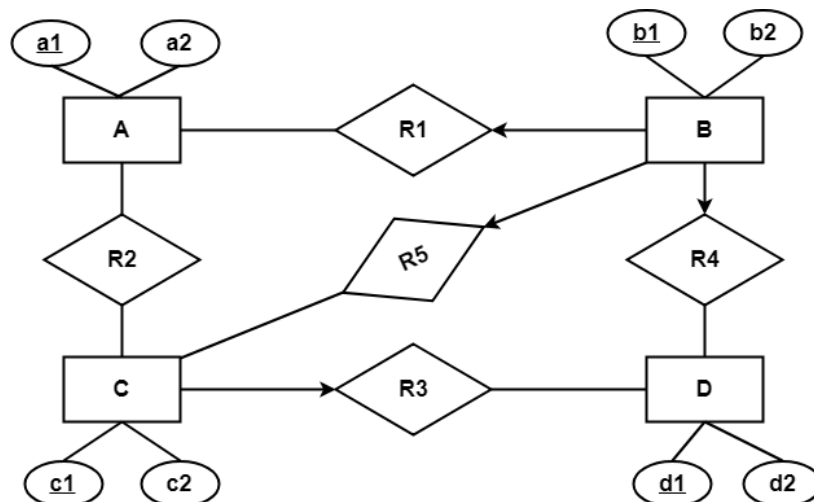
Solution-

Applying the rules, minimum 4 tables will be required-

- AR1R2 (a1 , a2 , b1 , c1)
- B (b1 , b2)
- C (c1 , c2)
- R3 (b1 , c1)

Problem-03:

Find the minimum number of tables required to represent the given ER diagram in relational model-



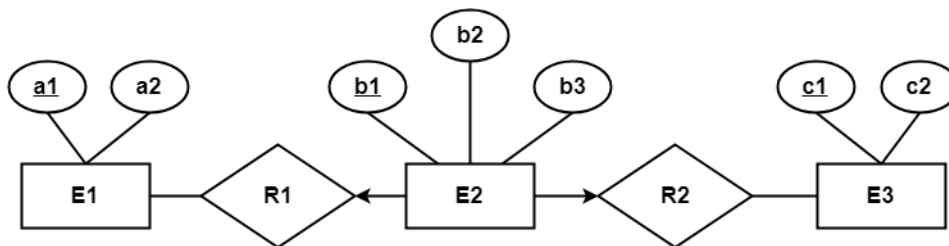
Solution-

Applying the rules, minimum 5 tables will be required-

- BR1R4R5 (b1 , b2 , a1 , c1 , d1)
- A (a1 , a2)
- R2 (a1 , c1)
- CR3 (c1 , c2 , d1)
- D (d1 , d2)

Problem-04:

Find the minimum number of tables required to represent the given ER diagram in relational model-



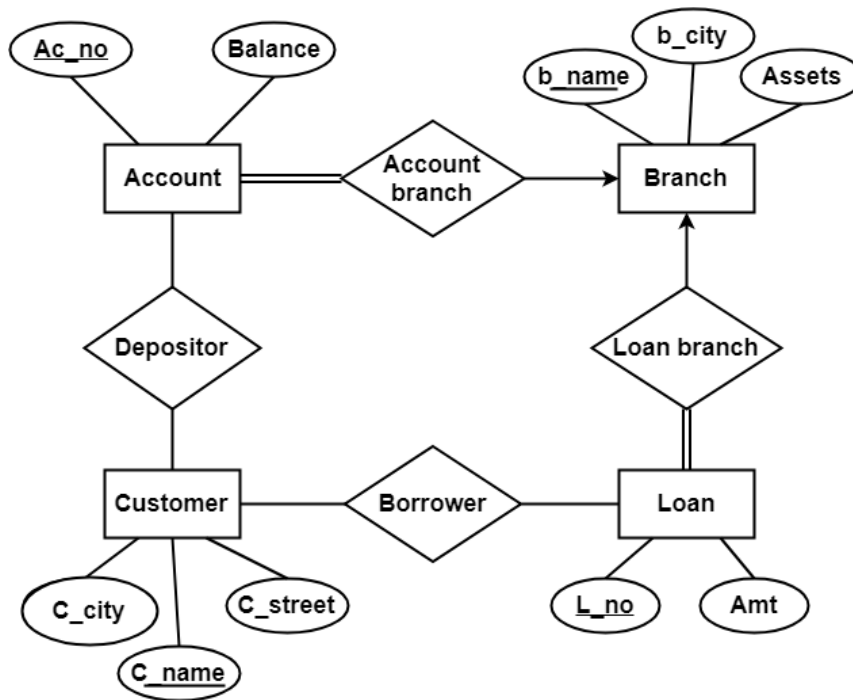
Solution-

Applying the rules, minimum 3 tables will be required-

- E1 (a1 , a2)
- E2R1R2 (b1 , b2 , a1 , c1 , b3)
- E3 (c1 , c2)

Problem-05:

Find the minimum number of tables required to represent the given ER diagram in relational model-



Solution-

Applying the rules that we have learnt, minimum 6 tables will be required-

- Account (Ac_no , Balance , b_name)
- Branch (b_name , b_city , Assets)
- Loan (L_no , Amt , b_name)
- Borrower (C_name , L_no)
- Customer (C_name , C_street , C_city)
- Depositor (C_name , Ac_no)

3.6 SUMMARY

- ER Model is used to define the data elements and relationship for a specified system.
- An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.
- An attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.
- The key attribute is used to represent the main characteristics of an entity. It represents a primary key.
- An attribute composed of many other attributes is known as a composite attribute.
- An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent a multivalued attribute.

- An attribute that can be derived from another attribute is known as a derived attribute. It can be represented by a dashed ellipse.
- Generalization is like a bottom-up approach in which two or more entities of lower level combine to form a higher level entity if they have some attributes in common.
- Specialization is a top-down approach, and it is opposite to Generalization. In specialization, one higher level entity can be broken down into two lower level entities.
- Constraints enforce limits to the data or type of data that can be inserted/updated/deleted from a table.
- A mapping constraint is a data constraint that expresses the number of entities to which another entity can be related via a relationship set.

3.7 REFERENCES

“Fundamentals of Database Systems”, Elmasri Navrate, 6th edition, 2013, Pearson
 “Introduction to Database Systems”, C.J.Date, Pearson Education
 “Data base System Concepts”, Silberschatz, Korth, McGraw Hill, V edition
 Fundamentals of Database Systems, by Navathe
<https://beginnersbook.com/2015/04/constraints-in-dbms/>
<https://www.gatevidyalay.com/er-diagrams-to-tables-practice-problems/>

3.8 FURTHER READING

Database System Concepts - 6th edition - Avi Silberschatz
 Database Systems - A Practical Approach to Design, Implementation & Management
 By Thomas Connolly, Carolyn Begg
 Database Management Systems By Raghu Ramkrishnan, Gehrke

3.9 QUESTIONS FOR PRACTICE (SHORT & LONG ANSWERS TYPE)

1. What is ER Model?
2. Name the various types of attributes?
3. What do you mean by relationship?
4. What is the difference between Generalization and Specialization?
5. What do you understand by domain constraint?
6. Explain possible mapping cardinalities along with diagram.
7. Write the basic rules for converting the ER diagrams into tables.
8. How can you differentiate Composite attribute from Multivalued attribute? Explain with example.
9. What is default constraint? Give example.
10. What do you understand by weak entity? How can you represent it?

B.Sc.(DATA SCIENCE)
SEMESTER-III
DATABASE MANAGEMENT SYSTEM

UNIT IV: NORMAL FORMS

STRUCTURE

- 4.0 Objectives
- 4.1 Introduction
- 4.2 Functional Dependency
 - 4.2.1 Types of Functional Dependency
- 4.3 Inference Rule
 - 4.3.1 Types of Inference Rule
- 4.4 Normal Forms/Normalization
 - 4.4.1 Need of Normalization
 - 4.4.2 Types of Normal Forms
- 4.5 Multivalued Dependency
- 4.6 Join Dependency
- 4.7 Summary
- 4.8 References
- 4.9 Further Readings
- 4.10 Practice Exercises

4.0 OBJECTIVES

- To describe the process of normalization.
- To provide description of terms used in normalization.
- To describe first normal form.
- To explain partial dependency.
- To describe second normal form.
- To define transitive dependency.
- To describe third normal form.
- To explain fourth & fifth normal form.
- To describe data anomalies in 1NF, 2NF & 3NF.
- To provide description of Multivalued dependency.
- To explain join dependency.

4.1 INTRODUCTION

Normalization or normal form refers to the structure of a database which was developed by IBM researcher E.F. Codd in the 1970s. Normalization in database management system is used to organize data in the form of a related tables. In this unit, we will discuss the concept of normalization and the terminology frequently used in normalization. We will discuss in detail the first, second, third, fourth and fifth normal forms used to normalise the relational tables so that redundant information is removed from a database. In addition, we will discuss the concepts such as partial dependency and transitive dependency on which second and third normal forms are based.

4.2 FUNCTIONAL DEPENDENCY

Functional dependency (FD) is a set of constraints between two attributes in a relation. It typically exists between the primary key and non-key attribute within a table.

$X \rightarrow Y$

The left side of FD is known as a determinant, the right side of the production is known as a dependent.

For example:

Assume, we have an employee table with following attributes:

Emp_Id, Emp_Name, Emp_Address.

Here,

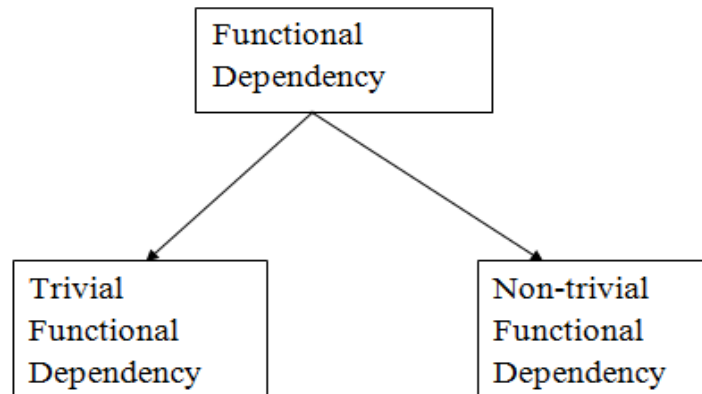
Emp_Id attribute can uniquely identify the Emp_Name attribute of employee table because if we know the Emp_Id, we can tell that employee name associated with it.

So, Functional dependency can be written as:

$Emp_Id \rightarrow Emp_Name$

We can say that Emp_Name is functionally dependent on Emp_Id. Thus, Functional dependency says that if two tuples have same values for attributes X1, X2,..., Xn, then those two tuples must have to have same values for attributes Y1, Y2, ..., Yn.

4.2.1 Types of Functional dependency



4.2.1.1 Trivial functional dependency

- If a functional dependency (FD) $X \rightarrow Y$ holds, where Y is a subset of X, then it is called a trivial FD. Thus, Trivial FDs always hold.
- The following dependencies are also trivial like:

$X \rightarrow X$ $Y \rightarrow Y$

For Example:

Consider a table with two columns Employee_Id and Employee_Name.

$\{Employee_id, Employee_Name\} \rightarrow Employee_Id$

It's a trivial functional dependency as Employee_Id is a subset of {Employee_Id, Employee_Name}.

Also, The following dependencies are trivial like

$Employee_Id \rightarrow Employee_Id$ $Employee_Name \rightarrow Employee_Name$

4.2.1.2 Non-trivial functional dependency

- $X \rightarrow Y$ has a non-trivial functional dependency if Y is not a subset of X.
- Moreover, When $X \cap Y$ is NULL, then $X \rightarrow Y$ is called as complete non-trivial functional dependency.

For Example:

ID \rightarrow Name
Name \rightarrow DOB

Self-check Exercise

1. Define Functional Dependency.
2. What is the difference between Trival and Non-Trival functional dependency?
3. Give example of functional dependency?

4.3 INFERENCE RULE (IR)

- The Armstrong's axioms are the basic inference rule to conclude functional dependencies on a relational database.
- The inference rule is a type of assertion which can be used to derive additional functional dependency from the initial set.

4.3.1 Types of Inference Rule

The Functional dependency has 6 types of inference rule:

4.3.1.1 Reflexive Rule (IR₁)

- In the reflexive rule, if Y is a subset of X, then X determines Y as shown below.

If $X \supseteq Y$ then $X \rightarrow Y$

For Example:

$X = \{a, b, c, d, e\}$
 $Y = \{a, b, c\}$

4.3.1.2 Augmentation Rule (IR₂)

- The augmentation is also called as a partial dependency. In augmentation, if X determines Y, then XZ determines YZ for any Z i.e. adding attributes in dependencies, does not change the basic dependencies.

i.e. If $X \rightarrow Y$ then $XZ \rightarrow YZ$

For Example:

For R(XYZA), if $X \rightarrow Y$ then $XZ \rightarrow YZ$

4.3.1.3 Transitive Rule (IR₃)

It is same as transitive rule in algebra. In the transitive rule, if X determines Y and Y determines Z, then X must also determine Z.

i.e. If $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$

4.3.1.4 Union Rule (IR₄)

Union rule says, if X determines Y and X determines Z, then X must also determine Y and Z.

i.e. If $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$

Proof:

$X \rightarrow Y$ (given).....eq.(1)
 $X \rightarrow Z$ (given).....eq.(2)
 $X \rightarrow XY$ (using IR₂ on eq.(1) by augmentation with X, where $XX = X$)
 $XY \rightarrow YZ$ (using IR₂ on eq.(2) by augmentation with Y)
 $X \rightarrow YZ$ (using IR₃ on eq.(3) and eq.(4))

4.3.1.5 Decomposition Rule (IR₅)

- Decomposition rule is also known as project rule. It is the reverse of union rule.
- According to this rule, if X determines Y and Z, then X determines Y and X determines Z separately.

i.e. If $X \rightarrow YZ$ then $X \rightarrow Y$ and $X \rightarrow Z$

Proof:

$X \rightarrow YZ$ (given).....eq.(1)
 $YZ \rightarrow Y$ (using IR₁ Rule)..... eq.(2)
 $X \rightarrow Y$ (using IR₃ on eq.(1) and eq.(2))

4.3.1.6 Pseudo transitive Rule (IR₆)

In Pseudo transitive Rule, if X determines Y and YZ determines W, then XZ determines W.

i.e. If $X \rightarrow Y$ and $YZ \rightarrow W$ then $XZ \rightarrow W$

Proof:

$X \rightarrow Y$ (given).....eq.(1)
 $WY \rightarrow Z$ (given).....eq.(2)
 $WX \rightarrow WY$ (using IR_2 on 1 by augmenting with W)...eq.(3)
 $WX \rightarrow Z$ (using IR_3 on eq.(3) and eq.(2))

Self-Check Exercise

1. What do you mean by inference rule? Explain its types.
2. What is reflexivity rule?
3. How can you define Union rule?
4. Define the term transitive dependency.
5. What do you mean by decomposition rule?

4.4 NORMAL FORMS (NORMALIZATION)

Database normalization is a database schema design technique, by which an existing schema is modified to minimize redundancy and dependency of data as it splits a large table into smaller tables and defines relationships between them to increase the clarity in organizing data.

4.4.1 Need of Normalization

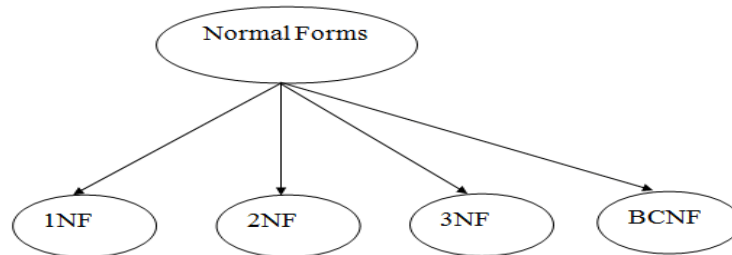
If a database design is not perfect, it may contain anomalies which are problems that can occur in poorly planned, un-normalised databases where all the data is stored in one table. It is also considered as a bad dream for any database administrator and managing a database with following anomalies is next to impossible. So, Such anomalies in database are of three types that can be defined as following:

- **Update anomalies** – If data items are scattered and are not linked to each other properly, then it could lead to strange situations. For example, when we try to update one data item having its copies scattered over several places, a few instances get updated properly while a few others are left with old values. Such instances leave the database in an inconsistent state and considered as update anomalies.
- **Deletion anomalies** – When we try to delete a record, but parts of it remains undeleted because of unawareness, the data is also saved somewhere else, so such state gives result in deletion anomalies.
- **Insert anomalies** – When we try to insert data in a record that does not exist at all then such instance gives result in insert anomalies.

Thus, Normalization minimizes the redundancy from a relation or set of relations and also eliminates the undesirable characteristics like Insertion, Update and Deletion Anomalies.

4.4.2 Types of Normal Forms

There are the four types of normal forms:



Normal Form	Description
<u>1NF</u>	A relation is in 1NF if it contains an atomic value.
<u>2NF</u>	A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
<u>3NF</u>	A relation will be in 3NF if it is in 2NF and no transition dependency exists.
<u>4NF</u>	A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
<u>5NF</u>	A relation is in 5NF if it is in 4NF and does not contain any join dependency and joining should be lossless.

4.4.2.1 First Normal Form

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

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

EMPLOYEE table:

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

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

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

Thus, It is clear that each attribute must contain only single value from its pre-defined domain.

4.4.2.2 Second Normal Form

- Before we learn about the second normal form, we need to understand the following –
 - **Prime attribute** – An attribute, which is a part of the candidate-key, is known as a prime attribute.
 - **Non-prime attribute** – An attribute, which is not a part of the prime-key, is said to be a non-prime attribute.
- If we follow second normal form, then relational must be in 1NF.
- Every non-prime attribute should be fully functionally dependent on prime key attribute i.e. if $X \rightarrow A$ holds, then there should not be any proper subset Y of X , for which $Y \rightarrow A$ also holds true.

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

TEACHER Table:

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

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

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

TEACHER_DETAIL Table:

TEACHER_ID	TEACHER_AGE
25	30
47	35
83	38

TEACHER_SUBJECT Table:

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

- Thus, there exists no partial dependency in 2NF.

4.4.2.3 Third Normal Form

- A relation will be in 3NF if it is in 2NF and does not contain any transitive partial dependency.
- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.
- A relation is in third normal form if it holds atleast one of the following conditions for every non-trivial function dependency
 $X \rightarrow Y$.

Where,

X is a super key.

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

Example:

EMPLOYEE_DETAIL table:

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

Super key in the above table:

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

Candidate key:

{EMP_ID}

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

Here,

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

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

EMPLOYEE table:

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

EMPLOYEE_ZIP table:

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

4.4.2.4 Boyce-Codd Normal Form

- BCNF is an extension of 3NF on strict terms.
- A table is in BCNF if every functional dependency $X \rightarrow Y$, X is the super key of the table.
- For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

Example: Let's assume there is a company where employees work in more than one department.

EMPLOYEE table:

EMP_ID	EMP_COUNTRY	EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
264	India	Designing	D394	283
264	India	Testing	D394	300
364	UK	Stores	D283	232
364	UK	Developing	D283	549

In the above table Functional dependencies are as follows:

EMP_ID → EMP_COUNTRY
EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

Candidate key: {EMP-ID, EMP-DEPT}

- The table is not in BCNF because neither EMP_DEPT nor EMP_ID alone are keys.
- To convert the given table into BCNF, we decompose it into three tables:

EMP_COUNTRY table:

EMP_ID	EMP_COUNTRY
264	India
264	India

EMP_DEPT table:

EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
Designing	D394	283
Testing	D394	300
Stores	D283	232
Developing	D283	549

EMP_DEPT_MAPPING table:

EMP_ID	EMP_DEPT
D394	283
D394	300
D283	232
D283	549

Functional dependencies:

EMP_ID → EMP_COUNTRY
EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

Candidate keys:

For the first table: EMP_ID

For the second table: EMP_DEPT

For the third table: {EMP_ID, EMP_DEPT}

4.4.2.5 Fourth Normal Form (4NF)

- A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
- For a dependency $X \twoheadrightarrow Y$, if for a single value of X, multiple values of Y exist, then the relation will be a multi-valued dependency.

Example:

STUDENT

STU_ID	COURSE	HOBBY
21	Computer	Dancing
21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cricket

59	Physics	Hockey
----	---------	--------

The given STUDENT table is in 3NF, but the COURSE and HOBBY are two independent entity. Hence, there is no relationship between COURSE and HOBBY.

In the STUDENT relation, a student with STU_ID, 21 contains two courses, Computer and Math and two hobbies, Dancing and Singing. So there is a Multi-valued dependency on STU_ID, which leads to unnecessary repetition of data.

So to make the above table into 4NF, we can decompose it into two tables:

STUDENT_COURSE

STU_ID	COURSE
21	Computer
21	Math
34	Chemistry
74	Biology
59	Physics

STUDENT_HOBBY

STU_ID	HOBBY
21	Dancing
21	Singing
34	Dancing
74	Cricket
59	Hockey

4.4.2.6 Fifth Normal Form (5NF)

- A relation is in 5NF if it is in 4NF and does not contain any join dependency and joining should be lossless.

- 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.
- 5NF is also known as Project-join normal form (PJ/NF).

Example:

SUBJECT	LECTURER	SEMESTER
Computer	Anshika	Semester 1
Computer	John	Semester 1
Math	John	Semester 1
Math	Akash	Semester 2
Chemistry	Praveen	Semester 1

In the above table, John takes both Computer and Math classes for Semester 1 but he doesn't take Math class for Semester 2. In this case, combination of all these fields required to identify a valid data.

Suppose we add a new Semester as Semester 3 but do not know about the subject and who will be taking that subject so we leave Lecturer and Subject as NULL. But all three columns together act as a primary key, so we can't leave other two columns blank.

So to make the above table into 5NF, we can decompose it into three relations P1, P2 & P3:

P1

SEMESTER	SUBJECT
Semester 1	Computer
Semester 1	Math
Semester 1	Chemistry
Semester 2	Math

P2

SUBJECT	LECTURER
Computer	Anshika
Computer	John
Math	John
Math	Akash
Chemistry	Praveen

P3

SEMSTER	LECTURER
Semester 1	Anshika
Semester 1	John
Semester 1	John
Semester 2	Akash
Semester 1	Praveen

Self-Check Exercise

1. What do you understand by the term 'Normalization'?
2. List the various normal forms involved in the normalization process.
3. How do you apply 2NF to a relational table?
4. How is data redundancy prevented using 1NF?
5. How do you apply 3NF to a relational table?

4.5 MULTIVALUED DEPENDENCY

- Multivalued dependency occurs when two attributes in a table are independent of each other but, both depend on a third attribute.
- A multivalued dependency consists of at least two attributes that are dependent on a third attribute that's why it always requires at least three attributes.

Example: Suppose there is a bike manufacturer company which produces two colors(white and black) of each model every year.

Bike_model	Manuf_year	Color
M2011	2008	White
M2001	2008	Black
M3001	2013	White
M3001	2013	Black
M4006	2017	White
M4006	2017	Black

Here columns COLOR and MANUF_YEAR are dependent on BIKE_MODEL and independent of each other. In this case, these two columns can be called as multivalued dependent on BIKE_MODEL. The representation of these dependencies is shown below:

BIKE_MODEL → → MANUF_YEAR

BIKE_MODEL → → COLOR

This can be read as "BIKE_MODEL multidetermined MANUF_YEAR" and "BIKE_MODEL multidetermined COLOR".

4.6 JOIN DEPENDENCY

- If a table can be recreated by joining multiple tables and each of these tables will have a subset of the attributes of the table, then the table is in Join Dependency. It is a generalization of Multivalued Dependency
- Join Dependency can be related to 5NF, wherein a relation is in 5NF, only if it is already in 4NF and it cannot be decomposed further.

Example:

<Employee>

EmpName	EmpSkills	EmpJob (Assigned Work)
Tom	Networking	EJ001
Harry	Web Development	EJ002
Katie	Programming	EJ002

The above table can be decomposed into the following three tables;

Therefore, it's not in 5NF:

<Employee Skills>

EmpName	EmpSkills
Tom	Networking
Harry	Web Development
Katie	Programming

<Employee Job>

EmpName	EmpJob
Tom	EJ001
Harry	EJ002
Katie	EJ002

<Job Skills>

EmpSkills	EmpJob
Networking	EJ001
Web Development	EJ002
Programming	EJ002

Our Join Dependency

{(EmpName,EmpSkills),(EmpName,EmpJob),(EmpSkills,EmpJob)}

The above relations have join dependency, so they are not in 5NF. That would mean that a join relation of the above three relations is equal to our original relation <Employee>.

Self-Check Exercise

1. What do you mean by join dependency?
2. What is Multivalued Dependency?
3. How do you achieve multi-valued dependency for a table?

4.7 SUMMARY

- Functional dependency (FD) is a set of constraints between two attributes in a relation.
- A database is normalized to ensure that the relational tables in the database contain only related information and store data only at single location in the database.
- The inference rule is a type of assertion which can be used to derive additional functional dependency from the initial set.
- Database normalization is a database schema design technique, by which an existing schema is modified to minimize redundancy and dependency of data.
- Multivalued dependency occurs when two attributes in a table are independent of each other but, both depend on a third attribute.
- Join Dependency can be related to 5NF, wherein a relation is in 5NF, only if it is already in 4NF and it cannot be decomposed further.
- A relation is in 5NF if it is in 4NF and does not contain any join dependency and joining should be lossless.
- 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy

4.8 REFERENCES

“Fundamentals of Database Systems”, Elmasri Navrate, 6th edition, 2013, Pearson
“Data base Systems design”, Implementation, and Management, Peter Rob & Carlos Coronel 7th Edition
“Database Systems, The Complete Book”, Hector Garcia-Molina, Jeffrey D. Ullman and Jennifer Widom, 6th impression, 2011, Pearson
“Data base Management Systems”, Raghu Rama Krishnan, Johannes Gehrke, 3rd Edition, 2003, McGraw Hill

4.9 FURTHER READINGS

Starting with Oracle, by John Day and Craig Van Slyke
Database Management Concepts, by Henry F. Korth
Fundamentals of Database Systems, by Navathe

4.10 Practice Exercises

1. What are the steps to normalize a relational table to first normal form?
2. What do you mean by functional dependency in a relational table?
3. What are the various steps to normalize a relational table to second normal form?
4. What are the various steps to normalize a relational table to third normal form?
5. How can you eliminate transitive dependency in a relational table?
6. List the various advantages of normalization.
7. What is trivial functional dependency?
8. What is fully functional dependency?
9. What is partial dependency in a relational table?
10. What is a primary key in a relational table?
11. What are the various objectives of normalization?

B.Sc.(DATA SCIENCE)

SEMESTER-III

DATABASE MANAGEMENT SYSTEM

UNIT V: STRUCTURED QUERY LANGUAGE

STRUCTURE

5.0 Objectives

5.1 Introduction to SQL

5.2 Data Types

5.3 DDL, DML and DCL (Parts of SQL)

5.4 Querying Database Tables

5.5 Data Definition Language (DDL)

5.6 Creating Tables

5.7 Inserting and Updating values into Table

5.8 Summary

5.0 OBJECTIVE

Understanding basic sql commands and architecture and structure of the language.

5.1 INTRODUCTION TO SQL

Structured Query Language (SQL, called as “Sequel”) is the first query based domain-specific programming language. It is used to manipulate and store relational databases or data in the relational database management system called as RDBMS. Simply it is very closely tied with the relational model. The major job of SQL is to create tables or relational databases, insert data, edit or update data, read and delete data in tables or relational databases. SQL handle data having relations between variables and entities (structured data). SQL solves the queries from the relational databases by using English language. It deals with all the RDBMS standard database based languages like Oracle, MS-Access, Informix, MySQL, mariaDB, postgresSQL and SQL Server. SQL is 4GL declarative language (4GL) that is based on procedural elements or procedural statements. Note that all the keywords of Structure query language are in the uppercase and is not case sensitive. SQL statements are text lines dependent and so a single statement of SQL can be used in multiple text line. Number of actions on the database can be performed by SQL statements. Relational algebra and relational calculus statements are solved by SQL. SQL commands execute any statement for any RDBMS by using SQL engine i.e. procedure to interpret the statements in query format. In SQL processing number of components like Query Engine , optimization Engine, classic query engine and Query dispatcher are included. Classic query engine solves the non-SQL queries. In SQL multiple records can be easily accessed by single SQL command. SQL scope is data definition (creation and modification), data manipulation (insert, update and delete), data query and data access control. It is the first commercial languages. In 1970, SQL first it used Edgar F. Codd’s relational model. Initially in 1970, SQL was developed by Donald D. Chamberlin and Raymond F. Boyce at IBM by studying the relational model of Edgar F. Codd. First name of SQL was SEQUEL (Structured English Query Language). It also developed a System R at IBM San Jose Research Laboratory that was the first relational model. First Relational Software Inc. a software company approved and used as first RDBMS in 1978 (Now this software company is known as Oracle Corporation). In 1979, it developed first Oracle RDBMS. In 1986, SQL became a standard approved by American National Standards Institute (ANSI) and further in 1987 standardized by International Organization for Standardization (ISO). The SQL language has been standardized by the ANSI X3H2 Database Standards Committee. Two of the latest standards are SQL-92 and SQL-99. Over the years, each vendor of relational databases has introduced new commands to extend their particular implementation of SQL. Oracle8i's implementation of the SQL language conforms to the "Entry Level" SQL-99 standards and is partially compliant with the Transitional, Intermediate, and Full levels of SQL-99. The goal of SQL is used for performing the C.R.U.D (Create, Retrieve, Update & Delete) operations on relational databases. Also administrative tasks on database performed by SQL. Administration tasks are to provide security to database and backup of data.

In the relational model, data is stored in structures called relations or *tables*. Each table has one or more attributes or *columns* that describe the table. In relational databases, the table is the fundamental building block of a database application. Tables are used to store data on Employees, Equipment, Materials, Warehouses, Purchase Orders, Customer Orders, etc. Columns in the Employee table, for example, might be Last Name, First Name, Salary, Hire Date, Social Security Number, etc.

SQL statements are issued for the purpose of:

- Data definition - Defining tables and structures in the database (DB).
- Data manipulation - Inserting new data, Updating existing data, Deleting existing data, and Querying the Database (Retrieving existing data from the database).

Another way to say this is the SQL language is actually made up of

1) the Data Definition Language (DDL) used to create, alter and drop schema objects such as tables and indexes, and

2) The Data Manipulation Language (DML) used to manipulate the data within those schema objects.

Basically SQL has five different languages in combined form. These are:

1. DQL (Data Query Language):

Full form of DQL is Data Query Language. Already stored information in the database can easily be fetched by the DQL. Data can be accessed by DQL commands. DQL commands are used to select data, view data from the table and deals with table variables and to create temp variables etc. SELECT command is the only command in DQL. The syntax is:

SELECT * FROM Table name;

Here * may be any table variable(s) or field(s) or attribute(s) or column(s).

For example, if you want to display all the columns (whole data) from student table, then command is:

SELECT * FROM student

2. DDL (Data Definition Language):

Full form of DDL is Data Definition Language. Table schemas are defined by DDL commands. DDL commands are used to create and alter the database object and database variables in the RDBMS. The various commands are CREATE TABLE , CREATE DATABASE, ALTER TABLE etc. Commonly used commands in DDL are CREATE, ALTER, TRUNCATE and DROP.

i). CREATE: CREATE command is used to create a database and database object like a table, index, view, trigger, stored procedure, etc. The general syntax is:

CREATE TABLE Table-name (attribute1 datatype, attribute2, datatype.....);

Here attribute1, 2..... are the column name or field name and data type is attribute data type.

ii). ***ALTER:*** To set the database and restruct the database object etc. by ALTER command. The general syntax is:

ALTER TABLE Table-name ADD attribute datatype;

iii). ***TRUNCATE:*** *To remove all the variables, fields or data from the table, this command is used. It empties a table. The general syntax is:*

TRUNCATE TABLE Table-name;

iv). ***DROP:*** To remove all the database fields, variables, the database and database object DROP command is used. The general syntax is:

DROP TABLE Table-name;

3. DCL (Data Control Language):

Full form of DCL is Data Control Language. Job of DCL is to control for accessing of database and allow user for specific tasks. In otherwords it is used for user and permission management. It controls the access to the database. It is used for providing and taking back the access rights on the database and database objects. Mainly GRANT and REVOKE are the DCL commands.

i). ***GRANT Command:*** To provide access right to the user, GRANT Command is used. The general syntax is :

GRANT INSERT, DELETE ON Table-name TO user-name;

ii). ***REVOKE Command:*** To take back access right from the user, REVOKE Command is used. It also cancels all the access right of the user from the database and the database object. The general syntax is:

REVOKE ALL ON Table-name FROM user-name;

4. TCL (Transaction Control Language):

Full form of TCL is Transaction Control Language. It is used for handling transactions in the database. Data integrity during transactions in the multi-user environment is checked by this command. Job of TCL commands is to rollback and commit data updation in the database. Commonly used TCL commands are COMMIT, ROLLBACK, SAVEPOINT, and SET TRANSACTION.

i). ***COMMIT Command:*** It is used to save or apply the modification in the database.

ii). ***ROLLBACK Command:*** It is used to undo the modification.

iii). ***SAVEPOINT Command:*** It is used to temporarily save a transaction. Note that the transaction can roll back to this point according to requirement.

5. DML (Data Manipulation Language):

Full form of DML is Data Manipulation Language. It is used for inserting, updating and deleting data from the database. Manipulation of data in RDBMS easily managed by the DML Commands and the various operations (adding, deleting, updating) in this are done by using INSERT INTO TableName, UPDATE tableName set data and DELETE FROM TableName etc. Commonly used DML commands are INSERT INTO, DELETE FROM, UPDATE.

i). **INSERT INTO:** It is used to add data to the database table. The general syntax is:

INSERT INTO Table-name (Attribute1, Attribute2,.....) VALUES (data1, data2,.....);

ii). **UPDATE:** It is used for updation of data in the database or table. Also a condition added by using WHERE clause in the command. The general syntax is:

UPDATE Table-name SET attribute = value WHERE condition;

iii). **DELETE:** It is used to remove data from the database table. Also condition can be added using the WHERE clause to remove a specific data according to the condition. The general syntax is:

DELETE FROM Table-name WHERE condition;

The various language elements in the SQL language are subdivided as:

- **Clauses** used for constituent components of queries and statements. It is optional.
- **Expressions** are applied for producing of data in the tables consisting of columns and rows of data
- **Predicates** is used for specification of conditions in the statements on the bases of logics or Boolean values. It control the queries flow control.
- **Queries** that is used to retrieve the data based on specific conditions and is most useful elements in the *SQL*.
- **Statements** are program flow, control transactions, sessions, connections, diagnostics etc.

Note that every SQL statements ends with semicolon (";") as statement terminator.

Applications of SQL

Followings are the various applications of SQL:

- It defines and describes the data for users.
- It allows users to access data in RDBMS.
- It is used for manipulation of data in the database.
- It also allows for embedding of one SQL modules into another and also link with pre-compilers and libraries.

- It is used for creation of databases and allow to drop databases and tables.
- It allows users to stored procedure linked with databases, to create view, to deals with functions in a database.
- It allows users to set permissions on procedures, tables and views.

5.2 DATA TYPES

Data type is used for to specify the type of the data for particular attribute or for any object in the table or database. In SQL every variable, column and expression has a related data type. Data types are used during creation of the tables or databases. According to your requirement, you can select a data type for a table column. Followings are the basic data types used in SQL:

i). **VARCHAR2:** It is of Character data type that has letters, numbers and punctuation. The syntax for this data type is:

VARCHAR2(size)

where *size* is the maximum number of alphanumeric characters the column can hold.

For example VARCHAR2(25) can hold up to 25 alphanumeric characters. In SQL or latest version of Oracle, the maximum size of a VARCHAR2 column is 8,000 bytes. The VARCHAR data type is a synonym for VARCHAR2. It is recommended to use VARCHAR2 instead of VARCHAR.

ii). **NUMBER:** It is of Numeric data type that contain integer or floating point numbers only. The syntax for this data type is:

NUMBER(precision, scale)

where *precision* is the total size of the number including decimal point and *scale* is the number of places to the right of the decimal.

For example, NUMBER(6,2) can hold a number between -999.99 and 999.99.

iii). **DATE:** It is of Date and Time data type that has a date and time portion in the format: DD-MON-YY HH:MI:SS. No additional information is needed when specifying the DATE data type. If no time component is supplied when the date is inserted, the time of 00:00:00 is used as a default. The output format of the date and time can be modified to conform to local standards. In general, to manipulate the time portion of the DATE datatype, one must make use of the TO_DATE and TO_CHAR SQL functions.

iv). **RAW:** It id free form binary data that contain binary data up to 255 characters. Data type LONG RAW can contain up to 2 gigabytes of binary data. RAW and LONG RAW data cannot be indexed and can not be displayed or queried in SQL*Plus. Only one RAW column is allowed per table.

v). **LOB:** It is of Large Object data types. These include BLOB (Binary Large Object) and CLOB (Character Large Object). More than one LOB column can appear in a table. These data

types are the preferred method for storing large objects such as text documents (CLOB), images, or video (BLOB).

Note that there are six types of data types used in the SQL with range are in the below tabular form:

i). Exact Numeric Data Types:

DATA TYPE	FROM	TO
Bigint	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
Int	-2,147,483,648	2,147,483,647
smallint	-32,768	32,767
tinyint	0	255
bit	0	1
decimal	$-10^{38} + 1$	$10^{38} - 1$
numeric	$-10^{38} + 1$	$10^{38} - 1$
money	-922,337,203,685,477.5808	+922,337,203,685,477.5807
smallmoney	-214,748.3648	+214,748.3647

ii). Approximate Numeric Data Types:

DATA TYPE	FROM	TO
Float	$-1.79E + 308$	$1.79E + 308$
Real	$-3.40E + 38$	$3.40E + 38$

iii). Date and Time Data Types:

DATA TYPE	FROM	TO
Datetime	Jan 1, 1753	Dec 31, 9999
Smalldatetime	Jan 1, 1900	Jun 6, 2079
Date	Stores a date like June 30, 1991	
Time	Stores a time of day like 12:30 P.M.	

Note – Here, datetime has 3.33 milliseconds accuracy where as smalldatetime has 1 minute accuracy.

iv). Character Strings Data Types:

Sr.No.	DATA TYPE & Description
1	Char Maximum length of 8,000 characters.(Fixed length non-Unicode characters)
2	Varchar Maximum of 8,000 characters.(Variable-length non-Unicode data).
3	varchar(max) Maximum length of 2E + 31 characters, Variable-length non-Unicode data (SQL Server 2005 only).
4	Text Variable-length non-Unicode data with a maximum length of 2,147,483,647 characters.

v). *Unicode Character Strings Data Types:*

Sr.No.	DATA TYPE & Description
1	Nchar Maximum length of 4,000 characters.(Fixed length Unicode)
2	Nvarchar Maximum length of 4,000 characters.(Variable length Unicode)
3	nvarchar(max) Maximum length of 2E + 31 characters (SQL Server 2005 only).(Variable length Unicode)
4	Ntext Maximum length of 1,073,741,823 characters. (Variable length Unicode)

vi). *Binary Data Types:*

Sr.No.	DATA TYPE & Description
1	Binary Maximum length of 8,000 bytes(Fixed-length binary data)
2	Varbinary Maximum length of 8,000 bytes.(Variable length binary data)
3	varbinary(max) Maximum length of 2E + 31 bytes (SQL Server 2005 only). (Variable length Binary data)
4	Image Maximum length of 2,147,483,647 bytes. (Variable length Binary Data)

vii). Misc Data Types:

Sr.No.	DATA TYPE & Description
1	sql_variant Stores values of various SQL Server-supported data types, except text, ntext, and timestamp.
2	Timestamp Stores a database-wide unique number that gets updated every time a row gets updated
3	Uniqueidentifier Stores a globally unique identifier (GUID)
4	Xml Stores XML data. You can store xml instances in a column or a variable (SQL Server 2005 only).
5	Cursor Reference to a cursor object
6	Table Stores a result set for later processing

A column may be specified with a NULL or NOT NULL constraint meaning the column may or may not be left blank, respectively. This check is made just before a new row is inserted into the table. By default, a column is created as NULL if no option is given. In addition to specifying NOT NULL constraints, tables can also be created with constraints that enforce referential integrity (relationships among data between tables). Constraints can be added to one or more columns, or to the entire table.

Each table may have one PRIMARY KEY that consists of a single column containing no NULL values and no repeated values. A PRIMARY KEY with multiple columns can be designated using the ALTER TABLE command. We create primary keys with NOT NULL constraints to uphold entity integrity.

Up to 255 columns may be specified per table. Column names and table names must start with a letter and may not contain spaces or other punctuation except for the underscore character. Column names and table names are case insensitive.

5.3 DDL, DML AND DCL (PARTS OF SQL)

There are four major parts of SQL, Which are given below:

- ✓ DDL
- ✓ DML
- ✓ DCL
- ✓ DQL

i). Data Definition Language or Data Description Language (DDL): DDL statements are used to define the database structure or schema. It is used for creation and modification of database objects such as tables, indices and users. DDL statements are same as to define data structure in to a computer programming. Basic statements used by DDL are `CREATE`, `ALTER`, and `DROP`. DDL was firstly used by Codasyl database model in establishing the relational model and afterward it was used by SQL. Some of the DDL commands are as:

- ✓ `CREATE` - to create objects in the database
- ✓ `ALTER` - alters the structure of the database
- ✓ `DROP` - delete objects from the database
- ✓ `TRUNCATE` - remove all records from a table, including all spaces allocated for the records are removed
- ✓ `COMMENT` - add comments to the data dictionary
- ✓ `RENAME` - rename an object

ii). Data Manipulation Language (DML): DML statements are used for managing data within schema objects. It is used as computer programming language for insertion, deletion and updation or editing or modification purposes in a database. Some operators are used by DML and it is the sublanguage of SQL. These operators are used both for reading or selecting and writing the data in and from database. Some of the DML commands are as:

- ✓ `SELECT` - retrieve data from the a database
- ✓ `INSERT` - insert data into a table
- ✓ `UPDATE` - updates existing data within a table
- ✓ `DELETE` - deletes all records from a table, the space for the records remain
- ✓ `MERGE` - UPSERT operation (insert or update)
- ✓ `CALL` - call a PL/SQL or Java subprogram
- ✓ `EXPLAIN PLAN` - explain access path to data
- ✓ `LOCK TABLE` - control concurrency

iii). Data Control Language (DCL): DCL is used for controlling the access to data stored in a database or provide an authorization. Syntax are very close to computer programming language. Some of the DCL commands are as:

- ✓ GRANT - gives user's access privileges to database
- ✓ REVOKE - withdraw access privileges given with the GRANT command
- ✓ COMMIT - save work done
- ✓ SAVEPOINT - identify a point in a transaction to which you can later roll back
- ✓ ROLLBACK - restore database to original since the last COMMIT
- ✓ SET TRANSACTION - Change transaction options like isolation level and what rollback segment to use

iv) Data Query Language (DQL): DQL statements are used for performing queries on the data within schema objects. The purpose of DQL commands is to get the schema relation based on the query passed to it. SELECT statement or SELECT Command is mostly useful command for DQL. Here data is taken from database on the bases of some condition by adding FROM or WHERE data manipulators.

5.4 QUERYING DATABASE TABLES

A Query is a set of instruction in the relational database management system (RDBMS). It is to get the data according to the statement or requirement from the database. For that purpose query tables are used. Query Table is used for **preparing the data** for easy reporting and analysis. Query can be done either from single table or multiple table by using **SQL SELECT** command. The various operations performed are used for report creation, data sharing and also to create another query from the Query Table over an existing Query Table. You can create Query Tables for filtering datasets, batching datasets together (union), transforming data, applying SQL query functions, joining datasets and more. Also some of the condition can taken for completing the exact query. The general syntax used are:

SELECT * FROM Table-name

WHERE condition

A complete SQL process is as shown in the figure 5.1 below:

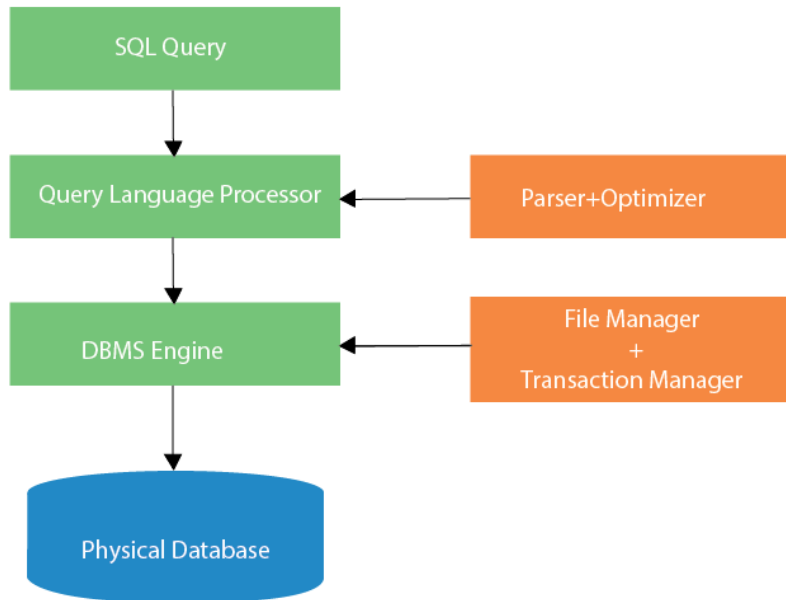


Figure 5.1

For example to fetch the student name from the STUDENT table or database, query can be written as:

```
SELECT student-name from STUDENT;
```

IF you take a condition to fetch the data for rollno 122 then the query can be written as:

```
SELECT student-name from STUDENT;
```

```
WHERE rollno=122
```

SELECT command or statement for query purpose: The basic syntax consists of four clauses as written in the below syntax of SELECT command or SELECT statement:

```
SELECT <attribute>
```

```
FROM <table-name>
```

```
WHERE <condition or boolean predicate to pick rows or columns>
```

```
[ ORDER BY <attribute> ];
```

First two clauses are compulsory from the four clauses. These are SELECT and FROM. Other two are optional. Optional clauses are WHERE and ORDER BY. The four basic clauses of a SELECT commands are:

- The SELECT clause is used for specification of attribute name separated by commas. You can select * (an asterisk character) for retrieval of all the columns.
- FROM clause is used to get the data from the table or database.

- Conditional statement takes place by using the WHERE clause. Here also boolean predicate (logical statements) can be used.
- The ORDER BY clause gives us a way to order the display of the rows in the result of the statement.

5.5 DATA DEFINITION LANGUAGE (DDL)

DDL is used for the creation of database or tables. Also view, indexes, cursor, triggers etc. can be created by using the popular DDL Create command.

Create Command: It is used to create various objects like

- ✓ Table
- ✓ View
- ✓ Index
- ✓ User
- ✓ Cursor
- ✓ Trigger
- ✓ Function
- ✓ Procedure

Create is a DDL i.e. data definition language statement and is used to create the above object. Like all other DDL statements create command cannot be roll backed

5.6 CREATING TABLES

A table or relation is defined as collection of interrelated records of tuples, where records (Tuples) are further collection of interrelated fields (attributes) and fields are defined as set of values defined over single domain. Create table command is DDL statement used to create a table. The general syntax of create table is

```
Create table table-name
(column1 datatype size(),
column2 datatype size(),
.....
.....
.....
columnN datatype size());
```

Some of the examples are:

1. Creating table from scratch:

```
create table student  
(rollno      number(4),  
name        varchar2(20),  
class       varchar2(12),  
address     varchar2(20));
```

2. Create a copy of an existing table:

Create table is also used to create a copy of an existing table rather than recreating it from scratch if required. The general syntax is:

```
Create table newtable-name  
As  
Select * from existing-table;
```

For example: create a copy of student table

```
Create table tempstu  
As  
Select * from student
```

3. Create a copy of structure of table:

Create table is also used to create the copy of the structure of an existing table in such case only structure will be copied and not contents. The general syntax is:

```
Create table newtable-name  
As  
Select * from existingtable  
Where (wrong condition)
```

For example: create a copy of structure of student table

```
create table temp-stu-stru  
as  
select * from student  
where 1=2;
```

4. Create a table with primary key:

```
Create table ptable
```

```
(rollno number (4) primary key,  
name varchar2 (20),  
class varchar2 (20)  
address varchar2 (20));
```

Here rollno is primary, hence note that

- ✓ It should not be null.
- ✓ It should be unique.

5. Create a table with other constraints:

```
create table pconstraint  
(Sid number (4) references ptable(rollno),  
regno varchar2 (4) unique key,  
name Varchar2 (20) Not Null,  
marks number (3) check (marks<=100));
```

Here

- ✓ Sid references to rollno of ptable
- ✓ regno should be unique and not duplicated.
- ✓ name should not be empty
- ✓ marks should be less than 100

6. Creating and Dropping Indexes:

An index is a performance-tuning method of allowing faster retrieval of records. An index creates an entry for each value that appears in the indexed columns. By default, Oracle creates B-tree indexes. An index is a data structure that affords rapid lookup of data in a table. An index is normally created on those columns of a table used to look up data. For example, in the employee table, the key *ssn* can be used to look up the rest of an employee's information. Creating a index on the *ssn* field would be accomplished by the following statement. The general syntax of creating index is

```
create index index_name  
on tablename(column_name);
```

For example,

```
CREATE INDEX idx_stu  
ON student(rollno) ;
```


Note, however, that by default, Oracle will automatically create an index on any column designated as a primary key. A good naming convention to use for indexes is to start with the letters `idx_` followed by the name of the table, and finally the name of the column(s) used to create the index.

➤ **Rename An Index:**

An index can be renamed by using the alter command which is a DDL statement, the syntax to rename an index is given below:

```
ALTER INDEX index_name RENAME TO new_index_name;
```

Indexes can be dropped using the DROP INDEX statement the general syntax to drop an index is

```
Drop index index_name
```

For example, to drop the `idx_stu` index, one could submit:

```
DROP INDEX idx_stu;
```

Dropping a table (using the DROP TABLE statement) automatically drops all indexes that have been built on columns in that table.

To see which indexes are defined in a schema, submit a query to the `USER_INDEXES` view of the database system catalog:

```
SELECT index_name, table_name FROM user_indexes ;
```

ID31	STUDENT
IDX_SEC	STUDENT
IDX_GAGAN	GAGAN
IDX_HONEY	HONEY
NDXCOM	TEMP

➤ **Creating and Dropping Views:**

In the SQL language, a view is a representation of one or more tables. A view can be used to hide the complexity of relationships between tables or to provide security for sensitive data in tables. In the following example, a limited view of the employee table is created. When a view is defined, a SQL statement is associated with the view name. Whenever the view is accessed, the SQL statement will be executed. In general View is a part of table it describes only certain

portion of a table. View are used to apply security to database by hiding certain portion of table. The general syntax of view is:

➤ **Creating View from Single Table:**

```
Create view viewname  
As  
Select * from tablename
```

For example a view having empno, ename and sal of EMP table can be created by following command

```
Create view vemp  
as  
select empno, ename, sal from emp
```

➤ **Creating View From Multiple Tables:**

A view can be created from more than one table. In such a case it hide the complexity associated with joins the general syntax to create view from multiple table is

```
Create view vname  
As  
select col1,col2,col3.....col N from table1, table2...tableM  
where condition;
```

For example

```
Create view vjoin  
As  
Select emp.ename,dept.dname from emp,dept  
Where emp.deptno=deptno.deptno;
```

The above view gives employee name with their department name, here employee name is taken from emp table and department name is taken from dept table

➤ **Truncate:**

Truncate is the DDL statement which is used to 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. The general syntax of truncate statement is

```
Truncate table tablename;
```

For example to truncate an emp table give the following command at SQL prompt.

Truncate table emp;

Drop:

The DROP command is a DDL statement which is used to remove following objects

- ✓ **Table**
- ✓ **View**
- ✓ **Index**

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. The general syntax of DROP statement is:

Drop Object Objectname;

For example to remove a table the general syntax will be look like as given below

Drop table tablename;

For example:

Drop table emp;

DROP is DDL commands, therefore once table is dropped it cannot be roll backed.

Note: in Oracle 10g a table can be "undropped" by using Flashback as given below:

```
SQL> FLASHBACK TABLE emp TO BEFORE DROP;
```

Alter Table:

It is another DDL statement which is commonly used in database modification. it has two objective:

- ✓ it is used to add/delete a column from existing table
- ✓ it is used to change the size of column

The general syntax of alter statement is

Alter table tablename

Add/modify columnname datatype (size);

1. Adding New Column to Student Table Having Following Columns:

- ✓ RollNo
- ✓ Name
- ✓ Class

Suppose we want to add address column to the student table, for this give the following command at SQL prompt.

```
alter table student  
add address varchar2 (30);
```

2. Deleting Column from an existing table:

Now if you want to delete class column from above student table the give the following command at SQL prompt.

```
Alter table student  
Drop column Class;
```

1. Modifying the length of Name column from 10 characters to 20 characters.

Now suppose student table's column name have 10 character length and you want to increase it from 10 to 20, and then give the following command at SQL prompt.

```
Alter table student  
Modify name varchar2 (20);
```

5.7 INSERTING AND UPDATING VALUES INTO A TABLE

Inserting which is straightforward task that is used for inserting a single row or a single record in the database. So a new row can be created by using the insert command or operation. Updating is another simple task for modification database. At a time one record can be updated. But you can also update whole sets of records at once.

Inserting a New Record: If we want to insert a new record into the database or into a table, then INSERT command is used. For example, if you want to insert a new tuple or row or record into the STUDENT table. The value for Rollno should be 122, Student-name should be “Deepak” and city should be “BATALA” and this can be added by using the INSERT Command.

```
insert into student (rollno,sname,city)  
values (122,'Deepak','BATALA')
```

Multiple values (rows or tuples or records) can inserted by using the below command as:

```
/* multi row insert */  
insert into student (rollno,sname,city)  
values (122,'Deepak','BATALA')
```

```
(123,'Balram','Amritsar')
```

Inserting Default Values: A table or database can be created or defined to take default values for specific columns. Default values without having any specific information or values can be inserted into the rows as:

```
create table Student (id integer default 0)
```

If you want default values to all the attributes or to a single attribute then DEFAULT keyword is used for this purpose. For example, DEFAULT keyword is used as:

```
insert into Student values (default)
```

You may also explicitly specify the column name, which you'll need to do anytime you are not inserting into all columns of a table:

```
insert into Student (rollno) values (default)
```

Overriding a Default Value with NULL: If you want override the default value by NULL value into the table, then lets take the first example or consider the table as:

```
create table Student (rollno integer default 0, sname VARCHAR(10))
```

If you want a row with a NULL value for rollno, then this can be written as:

You can explicitly specify NULL in your values list:

```
insert into Student (rollno, sname) values (null, 'Mona')
```

Modifying Records in a Table (Updating): You want to modify values for some or all rows in a table. For example, you might want to increase the salaries of everyone in department 20 by 10%. The following result set shows the DEPTNO, ENAME, and SAL for employees in that department:

```
Select deptno,ename,sal  
From emp  
Where deptno = 20
```

Order by 1,3

DEPTNO	ENAME	SAL
20	SMRIT	800
20	AMRIK	1100
20	JOLLY	2975
20	SURAJ	3000
20	RAJA	3000

You want to bump all the SAL values by 10%. Use the UPDATE statement to modify existing rows in a database table. For example:

```
update emp
set sal = sal*1.10
where deptno = 20
```

Use the UPDATE statement along with a WHERE clause to specify which rows to update; if you exclude a WHERE clause, then all rows are updated. The expression SAL*1.10 in this solution returns the salary increased by 10%. When preparing for a mass update, you may wish to preview the results. You can do that by issuing a SELECT statement that includes the expressions you plan to put into your SET clauses. The following SELECT shows the result of a 10% salary increase:

```
Select deptno,
      ename,
      sal
As orig_sal,
      sal*.10 as amt_to_add,
      sal*1.10 as new_sal
From emp
Where deptno=20
Order by 1,5
```

DEPTNO	ENAME	ORIG_SAL	AMT_TO_ADD	NEW_SAL
20	SMRIT	800	80	880
20	AMRIK	1100	110	1210
20	JOLLY	2975	298	3273
20	SURAJ	3000	300	3300
20	RAJA	3000	300	3300

The salary increase is broken down into two columns: one to show the increase over the old salary, and the other to show the new salary.

5.8 SUMMARY

Structured Query Language (SQL, called as “Sequel”) is the first query based domain-specific RDBMS programming language. Basically SQL has five different languages in combined form. These are DDL, DML, DCL, DQL and TCL. Data type is used for to specify the type of the data for particular attribute or for any object in the table or database. There are number of Data types used in SQL statements like VARCHAR2, Number, Date etc. There are four major parts of SQL, DDL, DML, DCL and DQL. Basic statements used by DDL are CREATE, ALTER, and DROP. Some of the DML commands are SELECT, INSERT, UPDATE, DELETE, MERGE, CALL etc. Some of the DCL commands are GRANT, REVOKE, COMMIT, SAVEPOINT and ROLLBACK etc. A Query is a set of instruction in the relational database management system (RDBMS). It is to get the data according to the statement or requirement from the database. For that purpose query tables are used. Create Command is used to create various objects like Table, View, Index, User, Cursor, Trigger, Function and Procedure etc. Inserting which is straightforward task that is used for inserting a single row or a single record in the database. So a new row can be created by using the insert command or operation. Updating is another simple task for modification database. At a time one record can be updated.

Question for practice:

- Q1. What do you know about SQL?
- Q2. How SQL was evolved?
- Q3. What are the basic SQL? Explain all.
- Q4. What do you mean by Data Types? Explain all the Data types used in SQL.
- Q5. Which data type is mostly used?

- Q6. Explain all the DDL commands with syntax and example?
- Q7. Explain all the DML commands with syntax and example?
- Q8. Explain all the DCL commands with syntax and example?
- Q9. Explain all the parts of SQL?
- Q10. What do you mean by Query?
- Q11. What are the various query statements used?
- Q12. What are the various Create commands with all the options?
- Q13. How can you insert values into the table?
- Q14. How can you update values in a table?

B.Sc.(DATA SCIENCE)
SEMESTER-III
DATABASE MANAGEMENT SYSTEM

UNIT VI: DATA MANIPULATION LANGUAGE

STRUCTURE

6.0 Objectives

6.1 Select Statement

6.2 Joins in SQL

6.3 Aggregate Functions

6.4 String Functions

6.5 SQL Having Sub Query

6.6 SQL Having Co-related Sub Query

6.7 Summary

6.0 OBJECTIVE

To understand syntax, semantic and working of data manipulation language.

6.1 SELECT STATEMENT

Select statement is a DML statement which is used to retrieve the records from a table. It **returns a result set of records**, from one table (relational database) or multiple tables. It retrieves zero row or multiple rows from one table or more database tables or database views.

Various form of SELECT statements are – simple SELECT, using special operators, aggregate functions, group by clause, sub query, joins, co-related sub query, union clause, exist operator.

Select statement is use for the following purposes:

- ✓ **To select selective data**
- ✓ **To sort the data either in ascending of descending order**
- ✓ **To select the data according to groups**
- ✓ **To filter the data according to user requirements**

The general syntax of select statement is give below

Select col1,col2... coln from tablename

Where condition

Group by column

Having condition

Order by column;

We will explain complete select statement by considering the various parts of select statement.

i). Simple Select (Complete Selection)

In this type of select statement we have to give all the column name of table to select them. For example if you have to select all the information regarding EMP table then give the following command

Select empno,ename,job,mgr,hiredate,sal,comm,deptno from emp;

7369	SMITH	CLERK	7902	17-DEC-80	5770		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	2600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20

7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	5075		10
7788	SCOTT	ANALYST	7566	19-APR-87	10000		20
7839	KING	PRESIDENT		17-NOV-81	5200		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	1131		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	3300		10

ii). Using Special Operator (*) - Short Form of Selections

In this mode you simply have to put an asterisk (*) instead of writing all the column names e.g. for the above statement we can write

*Select * From Emp*

7369	SMITH	CLERK	7902	17-DEC-80	5770		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	2600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	5075		10
7788	SCOTT	ANALYST	7566	19-APR-87	10000		20
7839	KING	PRESIDENT		17-NOV-81	5200		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	1131		30

7902	FORD	ANALYST	7566	03-DEC-81	3000	20
7934	MILLER	CLERK	7782	23-JAN-82	3300	10

iii). Partial Selection

In such type of selection we will select some column from a list according to user requirements. The data is not completely selected as in the above case, but selective data will be shown according to user requirement. This gives you a feel of database abstraction. For example

Select empno,ename,sal from emp;

7369	SMITH	5770
7499	ALLEN	2600
7521	WARD	1250
7566	JONES	2975
7654	MARTIN	1250
7698	BLAKE	2850
7782	CLARK	5075
7788	SCOTT	10000
7839	KING	5200
7844	TURNER	1500
7876	ADAMS	1100
7900	JAMES	1131
7902	FORD	3000
7934	MILLER	3300

iv). Filtered Selection

In such type of selection we will use where condition, where condition is used to restrict the number of rows i.e. it filters the rows of an EMP table

*select * from emp*

where deptno=10;

7782	CLARK	MANAGER	7839	09-JUN-81	5075	10
7839	KING	PRESIDENT		17-NOV-81	5200	10
7934	MILLER	CLERK	7782	23-JAN-82	3300	10

v). Sorting Data Using Select Statement

Select statement with order by statement is used to sort the data either in ascending or descending order. For example to sort the emp data in ascending order according to salary is given by following statement

```
Select * from emp
order by sal;
```

7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	1131		30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7499	ALLEN	SALESMAN	7698	20-FEB-81	2600	300	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	3300		10
7782	CLARK	MANAGER	7839	09-JUN-81	5075		10
7839	KING	PRESIDENT		17-NOV-81	5200		10
7369	SMITH	CLERK	7902	17-DEC-80	5770		20
7788	SCOTT	ANALYST	7566	19-APR-87	10000		20

vi). Group by Clause (Group by Expression)

Select statement can be used with group by expression to give the information according to group or in groups. It should be noted that group by clause is used with aggregate function. The following three major points should be considered while executing group by statement.

1. There should be an aggregate function in group by statement.
2. Group by statement can not be used with select * statement.
3. The column by which grouping is to be done should be present in select statement.

For example:

Select deptno, max(sal) from emp

Group by deptno;

10	5200
20	10000
30	2850

The above statement will give maximum salary of each department. We can restrict or filter the rows given by group by expression using having clause. For example

Select deptno, max(sal) from emp

Group by deptno

Having count()>5;*

30	2850
-----------	-------------

The above statement will give the maximum salaries of those departments in which there are more than 5 employees.

Examples of Select Statement:

1. Find All the Information of Employees Who Works In 10 Number Department

*Select * from emp*

Where deptno=20;

7369	SMITH	CLERK	7902	17-DEC-80	5770	20
7566	JONES	MANAGER	7839	02-APR-81	2975	20
7788	SCOTT	ANALYST	7566	19-APR-87	10000	20
7876	ADAMS	CLERK	7788	23-MAY-87	1100	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	20

2. Find all the information of employee whose commission is Null

*Select * from emp
where comm is null;*

7369	SMITH	CLERK	7902	17-DEC-80	5770		20
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	5075		10
7839	KING	PRESIDENT		17-NOV-81	5200		10
7900	JAMES	CLERK	7698	03-DEC-81	1131		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	3300		10

3. Find information of all the employees who have more than two year experience.

*Select * from emp
Where (sysdate-hiredate)>2*365;*

7369	SMITH	CLERK	7902	17-DEC-80	5770		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	2600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	5075		10
7788	SCOTT	ANALYST	7566	19-APR-87	10000		20
7839	KING	PRESIDENT		17-NOV-81	5200		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	1131		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	3300		10

6.2 JOINS IN SQL

SQL joins are used to join two or more tables that is with the help of SQL joins we can find or retrieve an information from two or more than two tables. There are four major types of joins.

- ✓ Equi joins
- ✓ Non equi joins
- ✓ Self joins
- ✓ Outer joins

a). Equi Join

The join which retrieve an information from two or more table by using the '=' operator is non as equi join. For example if you want to find the employee name and department of the employee then you will have to use equi join on EMP and DEPT table. One thing should be noted that there should be some common column in two table on which join has to be performed. as we know both emp and dept table have common column i.e. deptno.

```
SQL> Select emp.ename, dept.dname from EMP, DEPT
```

```
Where emp.deptno=dept.deptno;
```

SMITH	RESEARCH
ALLEN	SALES
WARD	SALES
JONES	RESEARCH
MARTIN	SALES
BLAKE	SALES
CLARK	ACCOUNTING
SCOTT	RESEARCH
KING	ACCOUNTING
TURNER	SALES
ADAMS	RESEARCH
JAMES	SALES
FORD	RESEARCH
MILLER	ACCOUNTING

b). Non equi join:

The joins that are not based on equality sign are known as non equi join. It can have <, > <= or >= sign it should not have “=” sign.

SQL>Select emp.empno, emp.ename,emp.sal,salgrade.grade from emp, salgrade

Where emp.sal>salgrade.losal and emp.sal<salgrade.hisal;

7876	ADAMS	1100	1
7900	JAMES	1131	1
7521	WARD	1250	2
7654	MARTIN	1250	2
7844	TURNER	1500	3
7499	ALLEN	2600	4
7566	JONES	2975	4
7698	BLAKE	2850	4
7369	SMITH	5770	5
7782	CLARK	5075	5
7839	KING	5200	5
7934	MILLER	3300	5

In the above example it shows empno, ename and sal from emp table and grade from salgrade table where sal should be greater than losal and less than hisal of salgrade table.

c). Self join

Self join is a join of a table to itself. That is a join of table with itself is known as self join. The name of the table to be joined using self join appear twice in the from clause with alias name. for example if you have to find the employee name and his manager name then following query with self join can be used.

SQL>Select a.ename,b.ename "manager" from emp a, emp b

where a.mgr=b.empno;

SMITH	FORD
ALLEN	BLAKE
WARD	BLAKE
JONES	KING
MARTIN	BLAKE
BLAKE	KING
CLARK	KING
SCOTT	JONES
TURNER	BLAKE
ADAMS	SCOTT
JAMES	BLAKE
FORD	JONES
MILLER	CLARK

Here two aliases of EMP as a and b are created to perform join operation.

d). Outer Join

The outer join return all the rows from one table and the rows from another table that satisfy the join condition. There are two subtypes of outer join

- ✓ Left outer join
- ✓ Right outer join
- **Left Outer Join:** only unmatched rows from left hand side table are retained.
- **Right Outer Join:** only unmatched rows from right hand side are retained

To understand the concept of outer join consider the following example

SQL>Select emp.ename,dept.dname from emp, dept

Where emp.deptno(+)=dept.deptno;

CLARK	ACCOUNTING
KING	ACCOUNTING
MILLER	ACCOUNTING
SMITH	RESEARCH
ADAMS	RESEARCH
FORD	RESEARCH
SCOTT	RESEARCH
JONES	RESEARCH
ALLEN	SALES
BLAKE	SALES
MARTIN	SALES
JAMES	SALES
TURNER	SALES
WARD	SALES
	OPERATIONS

SQL>Select emp.ename,dept.dname from emp,dept

Where emp.deptno=dept.deptno(+)

SMITH	RESEARCH
ALLEN	SALES
WARD	SALES
JONES	RESEARCH
MARTIN	SALES
BLAKE	SALES
CLARK	ACCOUNTING
SCOTT	RESEARCH
KING	ACCOUNTING
TURNER	SALES

ADAMS	RESEARCH
JAMES	SALES
FORD	RESEARCH
MILLER	ACCOUNTING

6.3 AGGREGATE FUNCTIONS (SQL FUNCTIONS)

SQL functions are built into Oracle and are available for use in various appropriate SQL statements. Do not confuse SQL functions with user functions written in PL/SQL. If you call a SQL function with an argument of a data type other than the data type expected by the SQL function, Oracle implicitly converts the argument to the expected data type before performing the SQL function. If you call a SQL function with a null argument, the SQL function automatically returns null. The only SQL functions that do not necessarily follow this behavior are CONCAT, NVL, and REPLACE.

In the syntax diagrams for SQL functions, arguments are indicated by their data types. When the parameter "function" appears in SQL syntax, replace it with one of the functions described in this section. Functions are grouped by the data types of their arguments and their return values. The various types of functions used in SQL are given below:

- ✓ **Aggregate functions**
- ✓ **Arithmetic functions**
- ✓ **String functions**
- ✓ **Date functions**

➤ **Aggregate Functions**

Aggregate functions return a single result row based on groups of rows, rather than on single rows. Aggregate functions can appear in select lists and in ORDER BY and HAVING clauses. They are commonly used with the GROUP BY clause in a SELECT statement, where Oracle divides the rows of a queried table or view into groups. In a query containing a GROUP BY clause, the elements of the select list can be aggregate functions, GROUP BY expressions, constants, or expressions involving one of these. Oracle applies the aggregate functions to each group of rows and returns a single result row for each group.

If you omit the GROUP BY clause, Oracle applies aggregate functions in the select list to all the rows in the queried table or view. You use aggregate functions in the HAVING clause to eliminate groups from the output based on the results of the aggregate functions, rather than on the values of the individual rows of the queried table or view. Many (but not all) aggregate functions that take a single argument accept these options:

- DISTINCT causes an aggregate function to consider only distinct values of the argument expression.
- ALL causes an aggregate function to consider all values, including all duplicates.

There are six major aggregate functions:

1. Min
2. Max
3. Sum
4. Avg
5. Count
6. variance

To explain the concept of aggregate function consider an EMP table having following records.

SQL> select * from emp;

EMPO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	5770		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	2600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	5075		10
7788	SCOTT	ANALYST	7566	19-APR-87	10000		20
7839	KING	PRESIDENT		17-NOV-81	5200		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	1131		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	3300		10
9898	HONEY						
56	BABLU						
344	GAGANDEEP						

MIN()

Min function is used to find the minimum value from a set of values e.g. if you have to find the minimum salary from EMP table the give following command at SQL prompt.

```
SQL> select min(sal) from emp;
```

1100

MAX()

Max function is used to find the maximum value from a set of values e.g. if you have to find the maximum salary from EMP table the give following command at SQL prompt.

```
SQL> select max(sal) from emp;
```

10000

SUM()

Sum function is used to find the sum of set of values e.g. if you want to find the sum of salary of all employee then use the following command

```
SQL> select sum(sal) from emp;
```

47001

AVG()

Avg function is used to find the average of set of values For example if you want to find the average of salary of all the employee then use the following command

```
SQL> select avg(sal) from emp;
```

3357.2143

COUNT

Count() function is used to count the number of records e.g. If you want to count all the employee who got the salary then use the following command

```
SQL> select count(sal) from emp;
```

14

VARIANCE ()

VARIANCE returns variance of *expr*. You can use it as an aggregate or analytic function. Oracle calculates the variance of *expr* as follows:

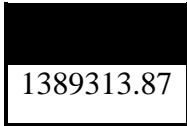
- ✓ 0 if the number of rows in *expr* = 1
- ✓ VAR_SAMP if the number of rows in *expr* > 1

If you specify DISTINCT, you can specify only the *query_partition_clause* of the *analytic_clause*. The *ORDER_BY_clause* and *windowing_clause* are not allowed.

Examples:

The following example calculates the variance of all salaries in the emp table:

```
SELECT VARIANCE(sal) "Variance" FROM emp;
```



1389313.87

Examples of Functions:

Problem: Write a query to find the second highest salary from EMP table

```
Select max(sal) from emp  
Where sal < (select max(sal) from emp);
```

Problem: Write a query to find the third highest salary from EMP table

```
Select max(sal) from emp  
Where sal <(select max(sal) from emp  
Where sal <(Select max(sal) from emp));
```

➤ Arithmetic Operator and Functions:

In SQL there are four main operator used in the queries of sub queries and are given below.

1. + (PLUS)
2. - (MINUS)
3. * (MULTIPLICATION)
4. / (DIVISION)

+ (PLUS):

This operator is used to find the sum of two or more variables. E.g. If you want to find the sum of 2 and 2 then use the following command

SQL> select (2+2) "Sum" from dual;

4

- **(MINUS):**

This operator is used to find the difference of two or more variables. For example If you want to find the subtract 3 from 10 then use the following command

SQL> select (10 – 3) "Minus" from dual;

7

* **(MULITPLY):**

This operator is used to find the product of two or more variables. E.g. If you want to find the product of 12 and 2 then use the following command

SQL> select (12*2) "Multiplication" from dual;

24

/ **(DIVISION):**

This operator is used to find the result of division of two numbers. E.g. if you want to find the result when 4 divided by 2 then use the following statement

SQL> select (4/2) from dual;

2

➤ **Arithmetic Functions or Number Functions:**

Arithmetic functions Number functions accept numeric input and return numeric values. Most of these functions return values that are accurate to 38 decimal digits. The transcendental functions COS, COSH, EXP, LN, LOG, SIN, SINH, SQRT, TAN, and TANH are accurate to 36 decimal digits. The transcendental functions ACOS, ASIN, ATAN, and ATAN2 are accurate to 30 decimal digits. The number functions are:

Mod()	Log()	Asin()	Cos()
Greatest()	Power()	Atan()	Tan()
Least()	Sin()	Atan2()	Sinh()
Sqrt()	Cos()	Sin()	Cosh()

MOD():

This function gives remainder after division of two given numbers. For example mod(7,3) gives 2 as quotient as 1 as remainder the following function gives 1 as output.

SQL> select mod(7,3) from dual;

[REDACTED]
1

GREATEST():

GREATEST returns the greatest of the list of *exprs*. All *exprs* after the first are implicitly converted to the data type of the first *expr* before the comparison. Oracle compares the *exprs* using nonpadded comparison semantics. Character comparison is based on the value of the character in the database character set. One character is greater than another if it has a higher character set value. If the value returned by this function is character data, its data type is always VARCHAR2. For example consider the following:

SQL>Select greatest ('HARRY', 'HARRIOT', 'HAROLD') "GREATEST" from dual;

[REDACTED]
HARRY

SQL>Select greatest(2,4,5,6,77,3) from dual;

[REDACTED]
77

LEAST ():

LEAST returns the least of the list of *exprs*. All *exprs* after the first are implicitly converted to the datatype of the first *expr* before the comparison. Oracle compares the *exprs* using nonpadded comparison semantics. If the value returned by this function is character data, its datatype is always VARCHAR2. For example

SQL>Select least('MANIK','ZARRIOT','ZAROLD') "LEAST" from dual;

[REDACTED]
MANIK

SQL>Select least(2,3,4,5,33,1,55) from dual;

1

SQRT():

SQRT is another important function that returns square root of n . The value n cannot be negative. SQRT returns a "real" result. For example

SQL>Select sqrt(26) "Square root" from dual;

5.09901951

SELECT SQRT(25) "Square root" FROM DUAL;

5

POWER():

POWER is a mathematical function that returns m raised to the n th power. The base m and the exponent n can be any numbers, but if m is negative, n must be an integer. For example

SQL>Select power(3,2) "Power" from dual;

9

LOG ():

LOG returns the logarithm, base m , of n . The base m can be any positive number other than 0 or 1 and n can be any positive number. For example

SQL>Select log(10,100) "Log base 10 of 100" from dual;

2

SIN():

sin() function give the value of sine. it should be noted that value given in brackets as argument are not in degree, but in radian hence sin(90) will not give 1 as output. For example

SQL> select sin(21) from dual;

.83665564

COS():

cos() function give the value of cosine. it should be noted that value given in brackets as argument are not in degree, but in radian. For example.

SQL> select cos(90) from dual;

-.4480736

TAN():

Tan() function give the value of tangent. it should be noted that value given in brackets as argument are not in degree, but in radian. For example

SQL> select tan(30) from dual;

-6.405331

CEIL():

CEIL is another important arithmetic function that returns smallest integer greater than or equal to *n*. For example

SQL>Select ceil(215.8) "Ceiling" from dual;

216

FLOOR():

FLOOR returns largest integer equal to or less than *n*. for example consider following

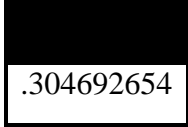
SELECT FLOOR(15.7) "Floor" FROM DUAL;

15

ASIN():

ASIN returns the arc sine of *n*. Inputs are in the range of -1 to 1, and outputs are in the range of $\pi/2$ to $\pi/2$ and are expressed in radians. Example

SELECT ASIN(.3) "Arc_Sine" FROM DUAL;



.304692654

ATAN ()

ATAN returns the arc tangent of n . Inputs are in an unbounded range, and outputs are in the range of $-\pi/2$ to $\pi/2$ and are expressed in radians. Example

```
SELECT ATAN(.3) "Arc_Tangent" FROM DUAL;
```




.291456794

ATAN2():

ATAN2 returns the arc tangent of n and m . Inputs are in an unbounded range, and outputs are in the range of $-\pi$ to π , depending on the signs of n and m , and are expressed in radians. ATAN2(n,m) is the same as ATAN2(n/m) . Example

```
SELECT ATAN2(.3, .2) "Arc_Tangent2" FROM DUAL;
```

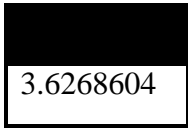


.982793723

SINH():

sinh() function computes the value of sine of hyperbolic. again the argument passed is in radian not in degree.

```
SQL> select sinh(2) from dual;
```



3.6268604

COSH():

cosh() function computes the value of cosine of hyperbolic. again the argument passed is in radian not in degree.

```
SQL> select cosh(40) from dual;
```



1.177E+17

```
SQL> select exp(2) from dual;
```

7.3890561

ABS():

ABS is a mathematical function that returns the absolute value of n that is it convert negative value to positive and does not effect on positive value

Select ABS(-15) "Absolute" from dual;

15

ACOS():

ACOS returns the arc cosine of *n*. Inputs are in the range of -1 to 1, and outputs are in the range of 0 to pi and are expressed in radians. For example

SELECT ACOS(.3)"Arc_Cosine" FROM DUAL;

1.26610367

RANK():

RANK is an analytic function. It computes the rank of each row returned from a query with respect to the other rows returned by the query, based on the values of the *value_exprs* in the *ORDER_BY_clause*. Rows with equal values for the ranking criteria receive the same rank. Oracle then adds the number of tied rows to the tied rank to calculate the next rank. Therefore, the ranks may not be consecutive numbers. Examples:

The following statement ranks the employees within each department based on their salary and commission. Identical salary values receive the same rank and cause nonconsecutive ranks.

SELECT deptno, ename, sal, comm,

RANK() OVER (PARTITION BY deptno ORDER BY sal DESC, comm) as rk

FROM emp;

DEPTNO	ENAME	SAL	COMM	RK
10	KING	5000		1
10	CLARK	2450		2
10	MILLER	1300		3
20	SCOTT	3000		1

20	FORD	3000		1
20	JONES	2975		3
20	ADAMS	1100		4
20	SMITH	800		5
30	BLAKE	2850		1
30	ALLEN	1600	300	2
30	TURNER	1500	0	3
30	WARD	1250	500	4
30	MARTIN	1250	1400	5
30	JAMES	950		6

ROUND (number function):

ROUND returns *n* rounded to *m* places right of the decimal point. If *m* is omitted, *n* is rounded to 0 places. *m* can be negative to round off digits left of the decimal point. *m* must be an integer. For example

SQL>Select round(15.193,1) "Round" from dual;

15.2

SQL>Select round(15.193,-1) "Round" from dual;

20

ASCII:

ASCII returns the decimal representation in the database character set of the first character of *char*. If your database character set is 7-bit ASCII, this function returns an ASCII value. If your database character set is EBCDIC Code, this function returns an EBCDIC value. There is no corresponding EBCDIC character function. For example

SQL>Select ASCII('Q') from dual;

81

TRUNC (number function):

TRUNC returns *n* truncated to *m* decimal places. If *m* is omitted, *n* is truncated to 0 places. *m* can be negative to truncate (make zero) *m* digits left of the decimal point. For example

SQL>Select TRUNC(15.79,1) "Truncate" from dual;

15.7

SQL>Select TRUNC(15.79,-1) "Truncate" from dual;

10

6.4 STRING FUNCTIONS

String functions are SQL built in functions that are operated upon strings. These are mainly used to process the strings that is with the help of string functions you can convert lower characters to upper character, upper characters to lower, find length of string , concatenate two or more string etc. There are various string functions available in SQL which are given below:

7369	SMITH	CLERK	7902	17-DEC-80	5770		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	2600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	5075		10
7788	SCOTT	ANALYST	7566	19-APR-87	10000		20
7839	KING	PRESIDENT		17-NOV-81	5200		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	1131		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	3300		10

9898	SIMI	0	500
56	HARMAN	0	500
344	SIMMI	0	700
111	JASKARAN	0	876

LOWER():

Lower function is used to convert all upper character into lower character e.g. if you want to convert all the ename of EMP table to lower character then use the following command.

SQL> select lower(ename) from emp;

Smith
Allen
Ward
Jones
Martin
Blake
Clark
Scott
King
Turner
Adams
James
Ford
Miller
Simi
Harman
Simmi
jaskaran

UPPER():

Upper function is used to convert all lower character into upper character e.g. if you want to convert all the ename of EMP table to upper character then use the following command.

SQL> select upper(ename) from emp;

```
SMITH
ALLEN
WARD
JONES
MARTIN
BLAKE
CLARK
SCOTT
KING
TURNER
ADAMS
JAMES
FORD
MILLER
SIMI
HARMAN
SIMMI
JASKARAN
```

INITCAP():

Initcap makes the first character or name capital all other character will be represented in small letters. For example

SQL> select initcap(ename) from emp;

```
Smith
Allen
Ward
Jones
```

Martin
Blake
Clark
Scott
King
Turner
Adams
James
Ford
Miller
Simi
Harman
Simmi
Jaskaran

LENGTH():

Length is another string function that is used to find the length of string i.e. it describes the length of string in number of character e.g. if you want to find the length of all the ename of EMP table then give the following command.

SQL> select length(ename) from emp;

5
5
4
5
6
5
5
5
4

```
6
5
5
4
6
4
6
5
8
```

SQL> select ltrim(ename,'S') from emp;

```
MITH
ALLEN
WARD
JONES
MARTIN
BLAKE
CLARK
COTT
KING
TURNER
ADAMS
JAMES
FORD
MILLER
IMI
HARMAN
IMMI
JASKARAN
```

SQL> select rtrim(ename,'S') from emp;

SMITH
ALLEN
WARD
JONE
MARTIN
BLAKE
CLARK
SCOTT
KING
TURNER
ADAM
JAME
FORD
MILLER
SIMI
HARMAN
SIMMI
JASKARAN

SQL> select chr(67) "Char" from dual;

C

SOUNDEX:

SOUNDEX returns a character string containing the phonetic representation of *char*. This function allows you to compare words that are spelled differently, but sound alike in English. For example

SQL>Select ename from emp

Where Soundex(ename) = Soundex('SMYTHE');

SMITH

VSIZE:

VSIZE returns the number of bytes in the internal representation of *expr*. If *expr* is null, this function returns null. For example

```
SQL>Select ename, VSize (ename) "BYTES" from emp
```

```
Where deptno = 10;
```

CLARK	5
KING	4
MILLER	6

SUBSTR:

SUBSTR returns a portion of *char*, beginning at character *m*, *n* characters long.

- If *m* is 0, it is treated as 1.
- If *m* is positive, Oracle counts from the beginning of *char* to find the first character.
- If *m* is negative, Oracle counts backwards from the end of *char*.
- If *n* is omitted, Oracle returns all characters to the end of *char*. If *n* is less than 1, a null is returned.

Floating-point numbers passed as arguments to SUBSTR are automatically converted to integers.

```
SQL>Select Substr('ABCDEFGF',3,4) "Substring" from dual;
```

CDEF

LPAD():

Lpad is a string function that is used to padded or fill the empty space on the left hand side of string by some special or specified character. For example

```
SQL> select lpad(ename,15,'$') from emp;
```

\$\$\$\$\$\$\$\$\$SMITH
\$\$\$\$\$\$\$\$\$ALLEN

```
$$$$$$$$$$$$WARD
$$$$$$$$$$$$JONES
$$$$$$$$$$$$MARTIN
$$$$$$$$$$$$BLAKE
$$$$$$$$$$$$CLARK
$$$$$$$$$$$$SCOTT
$$$$$$$$$$$$KING
$$$$$$$$$$$$TURNER
$$$$$$$$$$$$ADAMS
$$$$$$$$$$$$JAMES
$$$$$$$$$$$$FORD
$$$$$$$$$$$$MILLER
$$$$$$$$$$$$SIMI
$$$$$$$$$$$$HARMAN
$$$$$$$$$$$$SIMMI
$$$$$$$$JASKARAN
```

RPAD():

Rpad is a string function that is used to padded or fill the empty space on the right hand side of string by some special or specified character. For example

SQL> select rpad(ename,15,'^') from emp;

```
SMITH^^^^^^^^^^^^
ALLEN^^^^^^^^^^^^
WARD^^^^^^^^^^^^
JONES^^^^^^^^^^^^
MARTIN^^^^^^^^^^^^
BLAKE^^^^^^^^^^^^
CLARK^^^^^^^^^^^^
SCOTT^^^^^^^^^^^^
```

```

KING^^^^^^^^^^^^^
TURNER^^^^^^^^^^^^
ADAMS^^^^^^^^^^^^^
JAMES^^^^^^^^^^^^^
FORD^^^^^^^^^^^^^^
MILLER^^^^^^^^^^^^
SIMI^^^^^^^^^^^^^^
HARMAN^^^^^^^^^^^^
SIMMI^^^^^^^^^^^^^
JASKARAN^^^^^^^^^

```

Date Functions:

Date functions are inbuilt function of SQL that are associated with data manipulation. The various date function with their use are given below for consideration.

a). SYSDATE(): SYSDATE is a date function which gives the current date from the system. It should be noted that the date displayed will depend upon current system date.

SQL> select sysdate from dual;

```

07-APR-07

```

LAST_DAY(): LAST_DAY returns the date of the last day of the month that contains *d*. You might use this function to determine how many days are left in the current month.

SQL>Select SYSDATE, LAST_DAY(SYSDATE) "Last",
LAST_DAY(SYSDATE) - SYSDATE "Days Left" from dual;

```

02-OCT-05  31-OCT-05  29

```

The following example adds 5 months to the hired ate of each employee to give an evaluation date:

SQL>Select ename, hiredate, TO_CHAR(ADD_MONTHS(LAST_DAY(hiredate), 5))
"Eval Date" from emp;

SMITH	17-DEC-80	31-MAY-81
ALLEN	20-FEB-81	31-JUL-81
WARD	22-FEB-81	31-JUL-81
JONES	02-APR-81	30-SEP-81
MARTIN	28-SEP-81	28-FEB-82
BLAKE	01-MAY-81	31-OCT-81
CLARK	09-JUN-81	30-NOV-81
SCOTT	19-APR-87	30-SEP-87
KING	17-NOV-81	30-APR-82
TURNER	08-SEP-81	28-FEB-82
ADAMS	23-MAY-87	31-OCT-87
JAMES	03-DEC-81	31-MAY-82
FORD	03-DEC-81	31-MAY-82
MILLER	23-JAN-82	30-JUN-82

NEXT_DAY:

NEXT_DAY returns the date of the first weekday named by *char* that is later than the date *d*. The argument *char* must be a day of the week in the date language of your session, either the full name or the abbreviation. The minimum number of letters required is the number of letters in the abbreviated version. Any characters immediately following the valid abbreviation are ignored. The return value has the same hours, minutes, and seconds component as the argument *d*. For example Write a query to returns the date of the next Tuesday after March 15, 1998.

```
SQL>Select NEXT_DAY('15-MAR-98','TUESDAY') "NEXT DAY" from dual;
```

16-MAR-98

TO_CHAR (date conversion):

TO_CHAR converts *d* of DATE datatype to a value of VARCHAR2 datatype in the format specified by the date format *fmt*. If you omit *fmt*, *d* is converted to a VARCHAR2 value in the default date format. The '*nlsparams*' specifies the language in which month and day names and abbreviations are returned. This argument can have this form:

```
'NLS_DATE_LANGUAGE = language'
```


If you omit *nlsparams*, this function uses the default date language for your session.

```
SQL> Select TO_CHAR(HIREDATE, 'Month DD, YYYY') "New date format" from emp Where  
ename = 'BLAKE';
```

```
May 01,  
1981
```

```
SQL> select to_char(sysdate,'yyyy-mm-dd') from dual;
```

```
2007-04-07
```

```
SQL> select to_char(sysdate,'dd-mm-yyyy') from dual;
```

```
07-04-2007
```

```
SQL> select to_char(sysdate,'dd-mon-yyyy') from dual;
```

```
07-apr-2007
```

```
SQL> select to_char(sysdate,'dd-mm-yy') from dual;
```

```
07-04-07
```

```
SQL> select to_char(sysdate,'day') from dual;
```

```
saturday
```

```
SQL> select to_char(sysdate,'day-mon-yyyy') from dual;
```

```
saturday -apr-2007
```

```
SQL> select to_char(sysdate,'dd-day-mon-yyyy') from dual;
```

```
07-saturday -apr-2007
```

```
SQL> select to_char(sysdate,'dd-month-yyyy') from dual;
```

```
07-april -2007
```

```
SQL> select to_char(sysdate,'dd,day-month-yy') from dual;
```

```
07,saturday -april -07
```

6.5 SQL HAVING SUB QUERY

In SQL a Subquery is a query within another query. It is also called nested query. Subquery is actually defined as “**a query that is embedded in WHERE clause of another SQL query**”. SQL sub query actually depends upon the statement and the main query where it is applied. Sub query applied having embedded with HAVING, FROM or WHERE Clauses. The different general syntax are as:

Sub query with FROM clause:

```
SELECT column_name(s)
FROM (SELECT column_name(s) from table_name) as table_alias
WHERE condition;
```

Another syntax is (Sub query using WHERE Clause):

```
SELECT column_name(s)
FROM table_name_1
WHERE column_name expression_operator{=,NOT IN,IN, <,>, etc}(SELECT
column_name(s) from table_name_2);
```

Sub query with HAVING Clause is as:

```
SELECT column_name(s)
FROM table_name_1
WHERE condition
GROUP BY column_name(s)
HAVING Aggregate_function(column_name)expression_operator{=,
<,>}(SELECT column_name(s) from table_name_2);
```

Here **SELECT column_name(s)** is used for selection of attribute or field or column data from the table, **FROM** is used to for database name from where you want to get data, **WHERE condition** is used to for specification of condition on the basis of which it selects the data, **GROUP BY column_name(s)** is used to group rows, having similar values into summary rows and at end **HAVING condition** is used to filter groups based on the specified condition.

For example: To get the number of employees in each department, SQL statement is”

```
SELECT departmentid, count_employees  
FROM (SELECT count(DISTINCT employeeid) AS "count_employees",departmentid  
FROM employees GROUP BY departmentid) AS employee_summary  
ORDER BY count_employees;
```

Subquery in WHERE Clause filter the rows after comparing a column in the main table with the results of the subquery. For example, SQL Query is as:

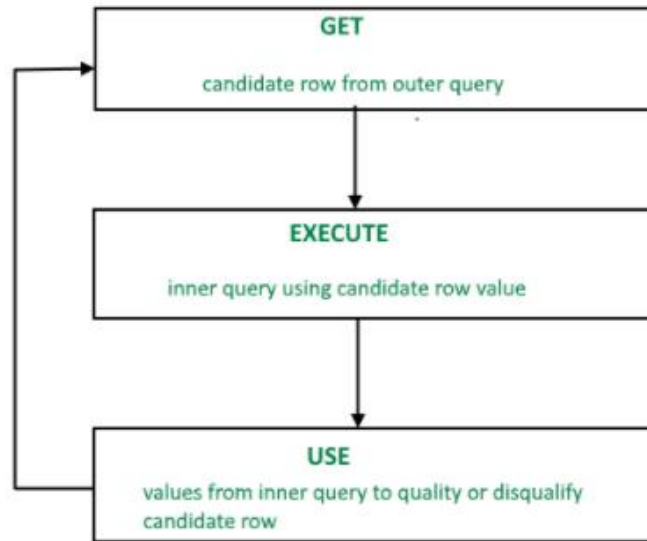
```
SELECT departmentname  
FROM department  
WHERE head IN (SELECT employeeid::varchar  
FROM employees  
WHERE city = 'Manhattan');
```

Another example of Subquery in HAVING Clause is as:

```
SELECT d.departmentname,count(e.employeeid)  
FROM department as d INNER JOIN employees as e  
ON d.departmentid::varchar = e.departmentid  
GROUP BY d.departmentname  
HAVING count(e.employeeid)>(SELECT count(employeeid) FROM employees WHERE city  
= 'New Delhi');
```

6.6 SQL HAVING CO-RELATED SUB QUERY

A correlated subquery is also called as synchronized subquery. It is a subquery **that uses the data values from the outer query i.e. inner query within outer query. It is** row-by-row processing. Every subquery is processed once for every row of the outer query.



A correlated subquery is executed once for every row processed by its parent statement like **SELECT**, **UPDATE** etc. The general syntax is as:

```

SELECT column1, column2, ....
FROM table1 outer
WHERE column1 operator
      (SELECT column1, column2
      FROM table2
      WHERE expr1 =
      outer.expr2);
  
```

A correlated subquery is one way of reading every row in a table and comparing values in each row against related data. It is used whenever a subquery must return a different result or set of results for each candidate row considered by the main query. In other words, you can use a correlated subquery to answer a multipart question whose answer depends on the value in each row processed by the parent statement.

The following query gets employees values whose salary is greater than the average salary of all employees:

```

SELECT
      Empid, name, salary
FROM employee
WHERE salary > (SELECT
      AVG(salary)
  
```

FROM employee

xi). SQL using union clause:

The UNION clause applied for combining the result-set of two or more SELECT statements. Note that each SELECT command within UNION must have the same number of columns with similar data types and are in the same order. The general syntax is:

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

UNION ALL Syntax

Another syntax by using the UNION ALL clause is as:

```
SELECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2;
```

For example, if you want to returns the cities (only distinct values) from both the "Customers" and the "Suppliers" table, then it will be:

```
SELECT City FROM Customers
UNION
SELECT City FROM Suppliers
ORDER BY City;
```

Another example is UNION With WHERE clause:

```
SELECT City, Country FROM Customers
WHERE Country='Germany'
UNION
SELECT City, Country FROM Suppliers
WHERE Country='Germany'
ORDER BY City;
```

xii) SQL with exist operator:

Use of EXISTS operator is to test for the existence of any record in the subquery and note that TRUE value returned if the subquery returns one or more records. The general syntax is:

```
SELECT column_name(s)
FROM table_name
WHERE EXISTS
(SELECT column_name FROM table_name WHERE condition);
```

For example EXISTS returns TRUE value for the suppliers having a product price less than 20:

```
SELECT SupplierName
FROM Suppliers
WHERE EXISTS (SELECT ProductName FROM Products WHERE Products.SupplierID =
Suppliers.supplierID AND Price < 20);
```

Another example of using NOT EXIST operator is to get all departments that do not have any employees, the solution is as:

```
SELECT department_id, department_name
FROM departments d
WHERE NOT EXISTS (SELECT 'X'
FROM employees
WHERE department_id
= d.department_id);
```

6.7 SUMMARY

In DML Select statement is mainly used. Select statement is a DML statement which is used to retrieve the records from a table. Various form of SELECT statements are – simple SELECT, using special operators, aggregate functions, group by clause, sub query, joins, co-related sub query, union clause, exist operator. SQL joins are used to join two or more tables that is with the help of SQL joins we can find or retrieve an information from two or more than two tables. There are four major types of joins as Equi joins, Non equi joins, Self joins, Outer joins etc. SQL functions are built into Oracle and are available for use in various appropriate SQL statements. Do not confuse SQL functions with user functions written in PL/SQL. The various types of functions used in SQL are **Aggregate functions, Arithmetic functions, String functions, Date functions etc.** There are six major aggregate functions as Min, Max, Sum, Avg, Count and variance. String functions are SQL built in functions that are operated upon strings. These are mainly used to process the strings that is with the help of string functions you can convert lower characters to upper character, upper characters to lower, find length of string , concatenate two or more string etc. In SQL a Subquery is a query within another query. It is also called nested query. Subquery is actually defined as “**a query that is embedded in WHERE clause of another SQL query**”. SQL sub query actually depends upon the statement and the main query where it is applied. Sub query applied having embedded with HAVING, FROM or WHERE Clauses.

6.8 PRACTICE EXERCISES

Q1. What are the various DML statements used?

Q2. What do you know about SQL statement?

- Q3. Explain the simple SELECT statement with syntax and examples?
- Q4. Explain the SELECT statement having special operators with syntax and examples?
- Q5. Explain the SELECT statement using the aggregate function with syntax and examples?
- Q7. Explain the SELECT statement using group clause with syntax and examples?
- Q8. What do you mean by Sub-Query?
- Q9. Explain the SELECT statement having sub-query statements with syntax and examples?
- Q10. What are the various Joins used in DML?
- Q11. Explain the SELECT statement using the Join options with syntax and examples?
- Q12. Explain the SELECT statement having co-related query statements with syntax and examples?
- Q13. Explain the SELECT statement using Union option with syntax and examples?
- Q14. Explain the SELECT statement using Exists operator with syntax and examples?

B.Sc.(DATA SCIENCE)
SEMESTER-III
DATABASE MANAGEMENT SYSTEM

UNIT VII: VIEWS

STRUCTURE

7.0 Objective

7.1 Introduction to Views

7.2 Data Independence

7.3 Statements on Join Views

7.4 Dropping a View

7.5 Database Security

7.6 Security Techniques

7.7 Two Phase Locking Techniques

7.8 Summary

7.0 OBJECTIVE

To understand different kind of views available in DBMS.

To understand statements on Join views and dropping a view

To understand database security and different security techniques

7.1 INTRODUCTION TO VIEWS

DBMS has different view from architectural point. A three-level architecture suggested by American National Standards Institute (ANSI) /Standards Planning and Requirements Committee (SPARC) in 1977. It is necessary to view data at different levels of abstraction. The three levels of the architecture are three different views of the data:

1. *External* - individual user view
2. *Conceptual* - community user view
3. *Internal* - physical or storage view

The three level database architecture allows a clear separation of the information meaning (conceptual view) from the external data representation and from the physical data structure layout. A database system that is able to separate the three different views of data is likely to be flexible and adaptable. We now briefly discuss the three different views as:

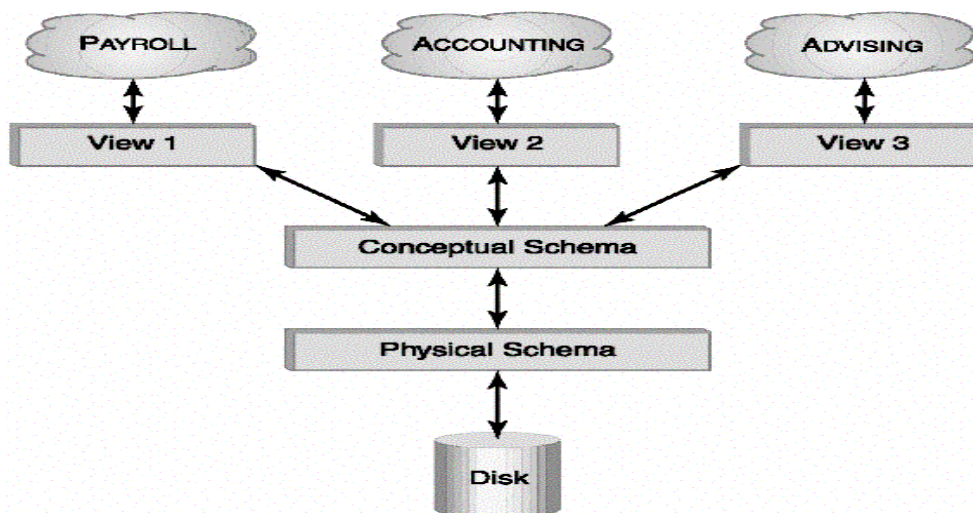


Figure 7.1

1. The external level is the view that the individual user of the database has that is why it also known as User view. This view is often a restricted view of the database and the same database may provide a number of different views for different classes of users. In general, the end users and even the applications programmers are only interested in a subset of the database. For example, a department head may only be interested in the departmental finances and student enrolments but not the library information. The

librarian would not be expected to have any interest in the information about academic staff. The payroll office would have no interest in student enrolments.

2. The conceptual view is the information model of the enterprise and contains the view of the whole enterprise without any concern for the physical implementation. This view is normally more stable than the other two views. In a database, it may be desirable to change the internal view to improve performance while there has been no change in the conceptual view of the database. The conceptual view is the overall community view of the database and it includes all the information that is going to be represented in the database. The conceptual view is defined by the conceptual schema which includes definitions of each of the various types of data.
3. The internal view is the view about the actual physical storage of data. It tells us what data is stored in the database and how. The **physical schema** describes details of how data is stored: tracks, cylinders, indices etc. on the random access disk system. It also typically describes the layout of files and type of files.

Mapping between Data Views: Assuming the three level view of the database, a number of mappings are needed to enable the users working with one of the external views. For example, the payroll office may have an external view of the database that consists of the following information only:

1. Staff number, name and address.
2. Staff tax information e.g. number of dependents.
3. Staff bank information where salary is deposited.
4. Staff employment status, salary level, leaves information etc.

The conceptual view of the database may contain academic staff, general staff, casual staff etc. A mapping will need to be created where all the staff in the different categories are combined into one category for the payroll office. The conceptual view would include information about each staff's position, the date employment started, full-time or part-time, etc.

This will need to be mapped to the salary level for the salary office. Also, if there is some change in the conceptual view, the external view can stay the same if the mapping is changed.

Data is structured in relational tables or databases by using different database objects either by using tables or using views etc. Data dependency and redundancy can be reduced in organization of tables in SQL databases. Big database can be well organized into small tables by splitting the big database by using the normalization. Normalization is used to set the data and databases in normal form for easy use in SQL programming. Note that a relationship can be created with the use of interlinked multiple tables. By using SQL commands, queries applied on tables and solve the problem easily. Complicated queries can be solved from multiple linked tables by using multiple joins.

A VIEW in SQL is similar to create virtual tables which have data from one table or one database or multiple tables or multiple databases. There is no physical existence of databases or tables and also not hold any data. Note that view name is unique in table or database as similar to a SQL table. Data can be easily fetched from the predefined SQL queries from the single database or table or from the multiple tables or databases. Below figure shows the VIEW having a query to join three relational database or tables and create a new virtual database or table from the data of multiple tables or databases.

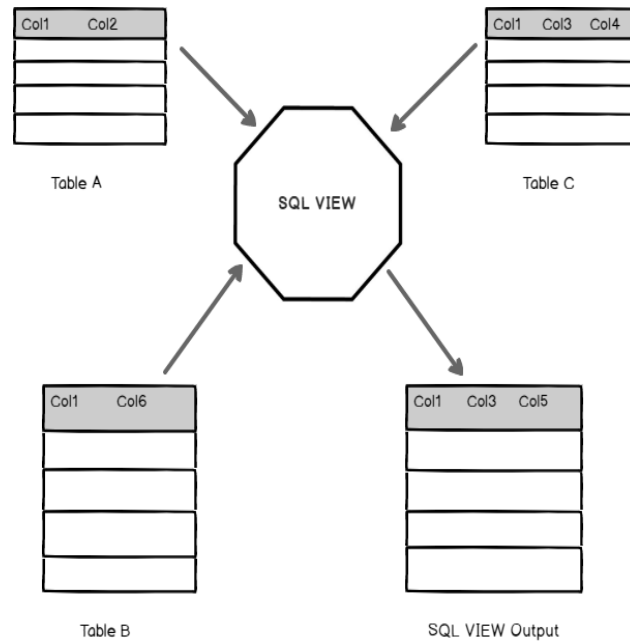


Figure 7.2

In the viewing the database security is also an important factor because there is no physical existence of a VIEW database or VIEW table and only data can be fetched by VIEW command. To create a view using the SQL command, the general syntax is as:

CREATE VIEW view-name AS

SELECT coll, col2,.....coln

FROM Table-name

WHERE condition;

For example to get all records from a table, a SQL VIEW command is:

CREATE VIEW Student-data AS

SELECT *

FROM Student

WHERE class-code=1

If we not wanted all the columns, then SQL VIEW command can be written as:

```
CREATE VIEW Student-data AS
```

```
SELECT rollno, s-name, fees
```

```
FROM Student
```

```
WHERE class-code=1
```

VIEW ENCRYPTION: VIEW can be easily encrypted by using WITH ENCRYPTION command. In past you have checked users to see the view definition. If you do not want users to view the definition, we can encrypt as:

```
CREATE VIEW Demo-View
```

```
WITH ENCRYPTION
```

```
AS
```

```
SELECT *
```

```
FROM Student
```

```
WHERE class-code=1
```

7.2 DATA INDEPENDENCE

To understand the concept of first of all understand the concept of Dependent data. It means such data which is fully dependent upon the data types, data storage location and accessing technique used. **Data independence** plays vital role in dealing with the database techniques and so is advantageous. Data Independence means data structure of the data is completely independent from the storage and accessing of the data. It is completely independent of any storage location, data types and accessing techniques. It means data at the centralized location or centralized DBMS is transparent in nature, i.e. data is completely transparent in centralized database management system. It states about the immunity of user related applications that not changes the structure and definition of the data. There are two levels of data independence:

➤ **Logical data dependency:**

It means logical structure is fully data independent and is also called as **schema definition**. Generally, in any case if any operation or change occurs on the sub set of the fields or attributes or columns in a relation or relational table or database, then in future it have no effects on the existing relation or table after inserting some of new attributes or after deletion of columns or attributes from the same relation. It means generally, logical independence means a program or statement not depends upon the structure of the data. In any case on the change of the structure of data in the program or in the relation will still have access to the database independently.

➤ **Physical data dependency:**

Meaning of physical data independence is hiding the complete storage structure from its relation or user applications. It means there is no effect on the application and storage location of the data in the relation or table or database or in multiple relations. For example, a program can access the data independently and have no effect whether data stored in hard disk or in pendrive or in floppy disc, CD ROM etc.

7.3 STATEMENTS ON JOIN VIEWS

First of let us understand the concept of View. In the SQL language, a view is a representation of one or more tables. A view can be used to hide the complexity of relationships between tables or to provide security for sensitive data in tables. In general View is a part of table it describes only certain portion of a table. View are used to apply security to database by hiding certain portion of table. In SQL Views are kind of virtual tables. A view is similar to the real table having rows and columns as in the real relation or database. View can be easily created by selecting attributes or fields or columns from one database or table or from multiple tables or databases or relations. A View may have a single row or single column or multiple rows or multiple columns depending upon the constraints given in the problem. Now further we discuss the Join View to create a view from the relation.

SQL joins are used to join two or more tables that is with the help of SQL joins we can find or retrieve an information from two or more than two tables. There are four major types of joins.

i). Equi Join:

The join which retrieve an information from two or more table by using the ‘=’ operator is non as equi join. For example if you want to find the employee name and department of the employee then you will have to use equi join on EMP and DEPT table. One thing should be noted that there should be some common column in two table on which join has to be performed. as we know both emp and dept table have common column i.e. deptno.

SQL> Select emp.ename, dept.dname from EMP, DEPT

Where emp.deptno=dept.deptno;

SMITH	RESEARCH
ALLEN	SALES
WARD	SALES
JONES	RESEARCH
MARTIN	SALES
BLAKE	SALES

CLARK	ACCOUNTING
SCOTT	RESEARCH
KING	ACCOUNTING
TURNER	SALES
ADAMS	RESEARCH
JAMES	SALES
FORD	RESEARCH
MILLER	ACCOUNTING

ii). Non equi join:

The joins that are not based on equality sign are known as non equi join. It can have <, > <= or >= sign it should not have “=” sign.

SQL>Select emp.empno, emp.ename,emp.sal,salgrade.grade from emp, salgrade

Where emp.sal>salgrade.losal and emp.sal<salgrade.hisal;

7876	ADAMS	1100	1
7900	JAMES	1131	1
7521	WARD	1250	2
7654	MARTIN	1250	2
7844	TURNER	1500	3
7499	ALLEN	2600	4
7566	JONES	2975	4
7698	BLAKE	2850	4
7369	SMITH	5770	5
7782	CLARK	5075	5
7839	KING	5200	5
7934	MILLER	3300	5

In the above example it shows empno, ename and sal from emp table and grade from salgrade table where sal should be greater than losal and less than hisal of salgrade table.

iii). Self join:

Self join is a join of a table to itself. That is a join of table with itself is known as self join. The name of the table to be joined using self join appear twice in the from clause with alias name. for example if you have to find the employee name and his manager name then following query with self join can be used.

```
SQL>Select a.ename,b.ename "manager" from emp a, emp b
      where a.mgr=b.empno;
```

SMITH	FORD
ALLEN	BLAKE
WARD	BLAKE
JONES	KING
MARTIN	BLAKE
BLAKE	KING
CLARK	KING
SCOTT	JONES
TURNER	BLAKE
ADAMS	SCOTT
JAMES	BLAKE
FORD	JONES
MILLER	CLARK

Here two aliases of EMP as a and b are created to perform join operation.

iv). Outer Join:

The outer join return all the rows from one table and the rows from another table that satisfy the join condition. There are two subtypes of outer join

Left Outer Join: only unmatched rows from left hand side table are retained.

Right Outer Join: only unmatched rows from right hand side are retained

To understand the concept of outer join consider the following example

```
SQL>Select emp.ename,dept.dname from emp, dept
```

```
      Where emp.deptno(+)=dept.deptno;
```

CLARK	ACCOUNTING
KING	ACCOUNTING
MILLER	ACCOUNTING
SMITH	RESEARCH
ADAMS	RESEARCH
FORD	RESEARCH
SCOTT	RESEARCH
JONES	RESEARCH
ALLEN	SALES
BLAKE	SALES
MARTIN	SALES
JAMES	SALES
TURNER	SALES
WARD	SALES
	OPERATIONS

SQL>Select emp.ename,dept.dname from emp,dept

Where emp.deptno=dept.deptno(+)

SMITH	RESEARCH
ALLEN	SALES
WARD	SALES
JONES	RESEARCH
MARTIN	SALES
BLAKE	SALES
CLARK	ACCOUNTING
SCOTT	RESEARCH
KING	ACCOUNTING
TURNER	SALES

ADAMS	RESEARCH
JAMES	SALES
FORD	RESEARCH
MILLER	ACCOUNTING

Join Operators:

JOIN is used to combine related tuples from two relations:

- ✓ In its simplest form the JOIN operator is just the cross product of the two relations.
- ✓ As the join becomes more complex, tuples are removed within the cross product to make the result of the join more meaningful.
- ✓ JOIN allows you to evaluate a join condition between the attributes of the relations on which the join is undertaken.

The notation used is

R JOIN join condition S

JOIN Example

R	<i>ColA</i>	<i>ColB</i>
	A	1
	B	2
	D	3
	F	4
	E	5

R JOIN	$R.ColA = S.SColA$	S		
	A	1	A	1
	D	3	D	3
	E	5	E	4

S	<i>SColA</i>	<i>SColB</i>
	A	1
	C	2
	D	3
	E	4

R JOIN	$R.ColB = S.SColB$	S		
	A	1	A	1
	B	2	C	2
	D	3	D	3
	F	4	E	4

➤ **Natural Join:**

Invariably the JOIN involves an equality test, and thus is often described as an equi-join. Such joins result in two attributes in the resulting relation having exactly the same value. A 'natural join' will remove the duplicate attribute(s).

- In most systems a natural join will require that the attributes have the same name to identify the attribute(s) to be used in the join. This may require a renaming mechanism.

- If you do use natural joins make sure that the relations do not have two attributes with the same name by accident.

➤ **Outer Join:**

Notice that much of the data is lost when applying a join to two relations. In some cases this lost data might hold useful information. An outer join retains the information that would have been lost from the tables, replacing missing data with nulls. There are three forms of the outer join, depending on which data is to be kept.

- LEFT OUTER JOIN - keep data from the left-hand table
- RIGHT OUTER JOIN - keep data from the right-hand table
- FULL OUTER JOIN - keep data from both tables

R LEFT OUTER JOIN R.ColA = S.SColA S

R	ColA	ColB
A	1	
B	2	
D	3	
F	4	
E	5	

S	SColA	SColB
A	1	
C	2	
D	3	
E	4	

Result	ColA	ColB	SColA	SColB
A	1		A	1
D	3		D	3
E	5		E	4
B	2		-	-
F	4		-	-

R RIGHT OUTER JOIN R.ColA = S.SColA S

R	ColA	ColB
A	1	
B	2	
D	3	
F	4	
E	5	

S	SColA	SColB
A	1	
C	2	
D	3	
E	4	

Result	ColA	ColB	SColA	SColB
A	1		A	1
D	3		D	3
E	5		E	4
-	-		C	2

R FULL OUTER JOIN R.ColA = S.SColA S

R	ColA	ColB
A	1	
B	2	
D	3	
F	4	
E	5	

S	SColA	SColB
A	1	
C	2	
D	3	
E	4	

Result	ColA	ColB	SColA	SColB
A	1		A	1
D	3		D	3
E	5		E	4
B	2		-	-
F	4		-	-
-	-		C	2

7.4 DROPPING A VIEW

To delete or to remove a view of a relation or an object view of a table is the dropping of the view. This can be done easily by using the DROP VIEW clause or command or statement. Note that definition of the view can be changed easily by dropping the view and further a new view or structure of the relation or table can be re-created.

DROP views from the database:

A view can always be disposed of with **DROP VIEW <view_name>** command.

DROP VIEW V1;

Note that when we execute the drop view command it removes the view definition. The underlying data stored in the base tables from which this view is derived remains unchanged. A view once dropped can be recreated with the same name. Also note that by dropping a view of a relation, there is no effect on the original data means it has unaffected database. So it is the great tool for end users for dealing the data from the relations or tables or databases, it means the table or database behind the scenes is safe. For this purpose DROP command is used. The purpose is to delete or drop an entire relation or database or table or column of data. It is not wise to remove the whole table. To make it safe we can use the View command from the SQL and further View can be removed any time when it is not required in future. That is why a view is a great option for end users, and dropping a view (instead of a table) is much less frightening.

VIEW Command in the SQL having the general syntax is as:

CREATE VIEW View-name AS

SELECT attribute1, attribute2.....

FROM Table

WHERE Condition

For example if you want to create a student view from the student table, then the SQL VIEW command is used as:

CREATE VIEW StudentView AS

SELECT rollno, sname, fees

FROM Student

WHERE Class-code=1

DROP VIEW Syntax for dropping a view is as written:

DROP VIEW View-name;

For example if you want to remove the view StudentView view table as created in the above example, the SQL command is used as:

DROP VIEW StudentView;

Note the semicolon should be at the end of the line which means database application in this command is complete. After deleting the view if you try to apply any query on the view then there will be an error occur.

7.5 DATABASE SECURITY

To preserve the database integrity, confidentiality and availability, some tools, measures and controls are designed. The finally database is secure by doing so and the above said process is called Data Security. Database security addresses and provides the protection in the following way:

- There must be data in the tables or in a database.
- There should be either a Database Management System (DBMS) or Relational Data Base Management System (RDBMS).
- Any applications or coding associated with the above defined and declared database.
- There should be physical and/or virtual database server with complete in hardware and networking.
- The computing technique/infrastructure should be for accessing the data from the database or table or from a relation

Database security is a challenging and complex endeavor for big data and the database more useable and accessible. It means more usable and accessible the database more is occurrences of threats to the data and needs more security, especially in case of big data.

Now question arises why it is important?

If there be any data breach occurs that create a failure in database maintenance and organization and also to establish the confidentiality of data in a database, then it may be a big loss to data. Following are the factors or consequences that inflict the business or your job that may be harmful after data breach:

- Your inventions, trade secrets and proprietary practices are intellectual property matter and if anybody access it then there is a big loss to you. So need Data Security.
- If your partners and customers are not feel secure during dealing with yours product or business, then this will damage your brand reputation and so need Data Security.

- Some business or jobs cannot continue to handle or operate till data breach problem is resolved.
- Any type of penalties or fine or loss dealing with the financial matters or digital money processing that create a high data threats and so need Data Security to control and avoid it.

Some of the common challenges and threats are:

- i). An insider threat is a major security threat from either a malicious insider or by negligent insider or from an outsider that is harmful to the data in a database.
- ii). Password sharing, weak passwords, accidents to data and other uninformed or unwise user existence create a problem to the important data and so is a big threat.
- iii). Hackers are also one of the major threat to the data and the database management softwares related with open source software industry.
- iv). A database-specific threat that is an insertion in non-SQL as attack or an arbitrary SQL statements that becomes harmful in HTTP headers or web applications.
- v). When attempt to write more data in a fixed length memory block after buffer overflow occurs, then there is loss of data and is a threat in data processing and accessing.
- vi). In case a denial of service (DoS) attack on the database server, server may be crashed and loss of data that creates a threat.
- vii). Failure in protection and backup data problem is another attack on important data.

These threats are impaired by using the following techniques:

- Data storage, capture and processing is a continues process and that grows the data exponentially in any institute or organizations. So after applying any data security practices or tools solve this.
- Networking infrastructure is increasing day by day and making the system more complex that creates a threat to the vital data, so by applying hybrid cloud or multicloud architecture solve this in easy way
- There is continuous grow in the worldwide regulatory compliance create a complexity in handling the data and databases related with these, so by following some rules and regulations to remove this complexity is a solution.

To evaluate database security around you, followings are the factors considered:

- Physical security should be on your database server or on cloud data center.
- Administrative and network access controls should **be** minimum to avoid any threat.
- Security should be to the end user account and devices related with server and databases to avoid any access and threat.

- Encryption should be **to** protect the data and database. All encryption keys should be handled in accordance with best-practice guidelines.
- To provide database software security, latest version of DBMS should be used.
- To provide security to the web servers and application server security that interacts with the database so that any type of threat can be avoided.
- Copies or backups or saving the images of the database is another important and best factor from any type of loss.
- Auditing should be regularly performed in a well manner by recording all the logins related with operating system and database server.

In your whole network infrastructure and environment there is need to implement layered security controls for securing yours databases, followings are the policies:

- Administrative controls to change, govern installation and to apply proper configuration management for securing the database.
- Preventative controls is another process to set encryption, apply some govern access techniques, masking and tokenization process.
- Detective controls for monitoring database activities and to prevent loss of data by using some prevention tools.

Database security policies should be integrated with and support your overall business goals, such as protection of critical intellectual property and your cyber security policies and cloud security policies. Ensure you have designated responsibility for maintaining and auditing security controls within your organization and that your policies complement those of your cloud provider in shared responsibility agreements. Security controls, security awareness training and education programs, and penetration testing and vulnerability assessment strategies should all be established in support of your formal security policies.

Data protection platforms and tools:

Followings are data protection tools and platforms:

- **Discovery:** Scanning of database tools should be at server level and local level.
- **Data activity monitoring:** Auditing and monitoring process should be to control the threats.
- **Tokenization and Encryption abilities:** Data shoul be transferred and used by using the Token technique and also coding-decoding process should be adopted.
- **Data security optimization and risk analysis:** Tools and techniques should be applied for measuring the data security optimization and to analysis the risk for data loss or threat to the database and database servers.

The Goal of the Database Administration:

It is collection of interrelated files. Data is of major concern in database and the main objective of DBA i.e. database administrator is to secure the database so that unauthorized or unauthenticated person cannot access and alter the database. The various mechanism used to secure the database are given below.

➤ *Security at file placement level:*

A database very often keeps sensible data about the users of the database. The data are placed in the files of the database and should be kept in a secure room. But there is no guarantee that the disk drives can't be stolen by an organized group of intruders. So the sensitive data like credit card numbers, addresses, health status, etc. will be misused. The databases are constructed to use many hard disk drives and table spaces can be placed (and even this is recommended) on different disks. The new modern communication environment often has a very high transmission rate and the link between the different nodes is very fast and reliable. One possibility of keeping data in different places is either to use separated SANs in different physical locations or to divide the data in smaller chunks (if logically possible) or to keep them in different instances (also on different places). The instances can be linked to work together using dblinks. Let us consider stealing of part of the disks. In this case the intruder will be able to use only a part of the data, but it will be impossible or very hard to get all data.

Another way to hide the sensitive data is to make a gateway between the root user data table and the tables with detail data. The idea is to use an intermediate table that hides the primary keys in the main user table introducing a set of different sub keys for different parts of the user data.

So normally a user will be represented with a set of random generated identification keys for different tables of the DB. The next step is to encrypt the intermediate table using cryptography tools.

1. Backup and Restore strategy:

A database backup strategy should be customized for the environment. It should take into consideration the system workload, usage schedule, importance of data, and hardware environment of the database. But there are some guidelines that apply to all databases. Incorporating these guidelines into the strategy will ensure more reliable and more cost effective backups. The main guidelines are:

- ✓ Always maintain a log of your backup schedule
- ✓ Occasionally store backups in a separate location
- ✓ Spread your database across several disks

- ✓ Always consider your existing hardware configuration and system workload
- ✓ Determine the volatility and value of your data
- ✓ Verify the integrity of your database regularly
- ✓ Verify the integrity of your backup media after each backup

a) Always maintain a backup schedule and log:

A well-designed database backup and recovery strategy is useless if the strategy is not publicized and practiced. Operational personnel need to be aware of the importance of the backup methodology, and they need to be able to quickly access backup media in the event database recovery is necessary. Accurate record-keeping and media storage are essential to a proper backup and recovery strategy. The backup schedule should include dates and times, type of backup and tape labels.

Be sure to always *review* the output log files that result from each database backup or recovery operation. Be especially watchful for errors or warnings that might indicate the correctness of the operation was compromised.

b) Occasionally store backups in an offsite location:

Backups are performed to facilitate database restoration, if your primary media fails, once the problem is corrected. There are many possible sources of failure of your primary media, and some of these sources must be considered in your backup strategy. Also, always remember that the data in your database is much more valuable than the physical media it is stored on.

If a site catastrophe occurs and your disks are destroyed, would your backup media still be safe? While it may be convenient, leaving your backup tapes on top of the tape drive is not a good idea. You should store your backup media in a separate location -- whether offsite or just in another area of the building. Although offsite storage is the safest method, access to an offsite archive will take longer than one on-site, and there may be security issues about removing confidential data from your site. A compromise might be to send backups offsite only once a month. Traditionally, you send the previous month's or week's backup offsite, keeping the most recent backup in a safe location on-site.

Magnetic tapes are an inexpensive, reliable, and reusable backup media. The environmental parameters for tapes are often broad and sometimes can lead to abuse. The audio tapes stored in your car can develop some hiss, and the video tapes in your home can drop out some colors, but if your backup tapes are damaged in this way, they are useless. You must follow the manufacturer's instructions on temperature, humidity, magnetic fields, and tape life span. A specially constructed tape library or archive is the safest storage strategy.

c) Always spread your database across several disks:

A backup strategy should provide guidelines on backups and restores of your database. In addition, your overall database strategy should include safeguards for protecting your data from

becoming corrupt and needing a restoration. While you cannot predict hardware failures, you can minimize their effects.

By spreading your database across several disks, you can eliminate having a single point of failure. Even if your database is not large enough to require multiple disks, it is strongly recommended putting the root file on a disk separate from the storage area disks. In addition, balancing busy storage areas between different disks is often a good tuning suggestion, but it also makes sense from a safety point of view.

Whether we backup our database to tapes or to disk, our backup schedule could overlap the routine **OS** backups of our system (especially if the database files share their disks with other users.) Because **OS** backups often run at a high priority, we should consider waiting until they are finished before starting a database backup. There will be less chance of mixing up tapes, less contention for the disks, and, if our system has multiple tape drives, we can allocate all of them to a multi-threaded database backup.

d) Always consider your existing hardware configuration and system workload:

The decision to backup your database either to tape or to disk will be determined primarily by your hardware environment and by personal preference. There are a few suggestions to consider:

- Rather than go to disk and then to tape, it is often faster and more reliable to go straight to tape.
- Multi-threaded backup to tape is usually faster than a single-threaded backup to disk. If you can allocate two tape drives, the backups complete in about half the normal time. Using three drives completes the task in about a third of the time.
- Historically, disk media is more reliable and faster than tapes. However, multi-threaded backup uses redundancy blocks which are more reliable than the single verify pass you get with a disk.
- Tapes are cheaper and easier to store than removable disk media. You can afford to keep more old backup versions around before you recycle the tapes. Try to maintain a minimum of 3 versions of full database backups, since a single past version is not sufficient for a reliable backup strategy.
- If you backup to non-removable disk media, you can probably only keep one past set of backups, which is also insufficient for a reliable backup strategy.

e) Determine the volatility and value of the data:

The volatility of your database refers to how often data changes. If you can determine what data is changing, and how often, you can create a strategy based on backing up the most-frequently changed storage areas more often. For example, the data in an inventory area could be updated several times an hour, an employee field could be updated at the end of every shift, and

a list of suppliers might only change every few months. Determining and mapping this workload will help determine the needs of your site.

The time required to complete incremental backups, and especially storage area incremental backups, is a good indication of changes to a database. Consider that a percentage of data in a database changes daily. Calculate this percentage and perform a full backup when 10% of a database has changed. While it is difficult to calculate this figure, it is much simpler to observe that when incremental backups take longer to complete each day, a greater percentage of a database has been changed since the last full backup. In fact, because of the extra calculations involved, an incremental backup could take longer than a full backup. A large incremental restore can take much longer than a full restore. Through observation, you should be able to determine the break-even point for performing a full backup rather than an incremental backup.

In addition to the frequency of backups, a backup strategy should also consider the differing daily usage of a database. For example, a typical backup schedule might include incremental backups every night and a full backup on Friday night. But if an examination of your usage shows that most changes are made on Wednesdays, then you should consider moving the full backup to Wednesday night. Similarly, if your database is accessed by users in multiple time zones, you might consider starting backups an hour or two later when the activity level has quieted down. Scheduling backups around user activities causes less impact on the users and results in more efficient backups.

f) Verify the integrity of your database regularly:

A backup strategy is useless if the database being backed up is corrupt to begin with. The DBMS contain online utilities to verify the correctness of a database; these tools should be used prior to starting the database backup operation. No database backup operation should be allowed to proceed if the database has not been recently verified as consistent in all respects.

It is recognized that it may not always be feasible to verify the integrity of the database before *every* database backup. However, an attempt should be made to verify the database integrity as often as operationally possible. The verification of the database (DB integrity checking) could be done in the following ways:

- using db verify utility (DBV) provided with Oracle server
- making full export to /dev/null

In situations where it is not possible to always verify the integrity of the entire database before a backup operation, for instance because there is not enough time to perform both a full verify *and* a backup, make an attempt to verify individual data or index areas that are most vital to the database.

g) Verify the integrity of your backup media after each backup:

One of the most common mistakes made is not verifying the integrity of the database backup media; at least, not until the backup is needed to be restored and then it is too late. After a database is backed up, the backup media should be verified. Verification ensures:

- that the backup strategy works as intended
- that the backup media is verifiable
- that the backup media is restorable

There are several methods available to verify the integrity of the database backup media. However, only one method is guaranteed to accurately reflect the integrity of the backup media: the actual restoration or recovery of a database. Anything else is merely verifying the media, not the contents of the media.

2. Data guard (Standby) Databases:

The Data Guard ensures high availability, data protection, and disaster recovery for enterprise data. Data Guard provides a comprehensive set of services that create, maintain, manage, and monitor one or more standby databases to enable production databases to survive disasters and data corruptions. Data Guard maintains these standby databases as transactionally consistent copies of the production database. Then, if the production database becomes unavailable because of a planned or an unplanned outage, Data Guard can switch any standby database to the production role, thus minimizing the downtime associated with the outage. Data Guard can be used with traditional backup, restoration, and cluster techniques to provide a high level of data protection and data availability.

A **Data Guard configuration** consists of one production database and up to nine standby databases. The databases in a Data Guard configuration are connected by Net services and may be dispersed geographically. There are no restrictions on where the databases are located, provided that they can communicate with each other. For example, you can have a standby database on the same system as the production database, along with two standby databases on another system.

➤ **Primary Database:**

A Data Guard configuration contains one production database, also referred to as the primary database, that functions in the primary role. This is the database that is accessed by most of your applications. The primary database can be either a single-instance database or a Real Application Clusters database.

➤ **Standby Databases:**

A standby database is a transactionally consistent copy of the primary database. A standby database is initially created from a backup copy of the primary database. Once created,

Data Guard automatically maintains the standby database by transmitting primary database redo data to the standby system and then applying the redo logs to the standby database.

Similar to a primary database, a standby database can be either a single-instance database or an Real Application Clusters database. A standby database can be either a physical standby database or a logical standby database:

➤ **Physical standby database:**

It provides a physically identical copy of the primary database, with on-disk database structures that are identical to the primary database on a block-for-block basis. The database schema, including indexes, is the same. A physical standby database is kept synchronized with the primary database by recovering the redo data received from the primary database.

➤ **Logical standby database:**

It contains the same logical information as the production database, although the physical organization and structure of the data can be different. It is kept synchronized with the primary database by transforming the data in the redo logs received from the primary database into SQL statements and then executing the SQL statements on the standby database. A logical standby database can be used for other business purposes in addition to disaster recovery requirements. This allows users to access a logical standby database for queries and reporting purposes at any time. Thus, a logical standby database can be used concurrently for data protection and reporting.

➤ **Log Apply Services:**

The redo data transmitted from the primary database is archived on the standby system in the form of archived redo logs. **Log apply services** automatically apply archived redo logs on the standby database to maintain transactional synchronization with the primary database and to allow transactionally consistent read-only access to the data. The main difference between physical and logical standby databases is the manner in which log apply services apply the archived redo logs:

- *For physical standby databases*, Data Guard uses **redo apply** technology, which applies redo data on the standby database using standard recovery techniques of the Oracle database server.

- *For logical standby databases*, Data Guard uses **SQL apply** technology, which first transforms the received redo data into SQL statements and then executes the generated SQL statements on the logical standby database. Log applies services perform the following tasks:

- Automatic application of archived redo logs on the standby database
- Automatic detection of missing redo logs on a standby system and automatic retrieval of missing redo logs from the primary database or another standby database

3. Encryption and Decryption:

Among other security technologies, DBMS protect data in E-business systems through strong, standards-based encryption. For example, Oracle has supported encryption of network data through Oracle Advanced Security since Oracle7. Oracle9i also supports protection of selected data via encryption within the database. Although encryption is not a substitute for effective access control, one can obtain an additional measure of security by electively encrypting sensitive data before it is stored in the database. Examples of such data could include:

- ✓ credit card numbers
- ✓ national identity numbers
- ✓ passwords for applications whose users are not database users *)
- ✓ trade secrets
- ✓ quarter end profits
- ✓ passwords are usually hashed with a one way scheme, but it may be necessary to encrypt plain text passwords

➤ **Key Management:**

The secrecy of encrypted data is dependent on the existence of a secret key shared between the communicating parties. Providing and maintaining such secret keys is known as "key management". Key management, including both generation and secure storage of cryptographic keys, is one of the most important aspects of encryption. If keys are poorly chosen or stored improperly, it is far easier for a malefactor to break the encryption. Rather than using an exhaustive key search attack (that is, cycling through all the possible keys in hopes of finding the correct decryption key, also known as 'brute force attack'), cryptanalysts typically seek weaknesses in the choice of keys, or the way in which keys are stored.

Remember that encryption is something that is easily done wrong and very hard to get right. Once you have chosen a secure algorithm, consider that as the first step in achieving security requirements. The next main problem is key management.

Imagine, when a corporation wants to protect credit card information from rogue DBAs, it makes no sense to trust the DBAs with the encryption keys. In the real world, key management is the hardest part of cryptography.

Key storage is one of the most important, yet difficult aspects of encryption and one of the hardest to manage properly. To recover data encrypted with a secret key, the key must be accessible to the application or user seeking to decrypt data. The key needs to be easy enough to retrieve that users can access encrypted data when they need to without significant performance degradation. The key also needs to be secure enough so that it is not easily recoverable by an unauthorized user trying to access encrypted data he is not supposed to see. There are some possibilities to store the keys:

- ✓ Store the key in the database
- ✓ Store the key in the operating system
- ✓ Have the user manage the key

7.6 SECURITY TECHNIQUES

In the information technology and internet era, number of security threats occurs. So to protect useful information and crucial data, some of the data privacy mechanism and data security techniques should be implemented. So first thing is to identify the confidential data and protect it for any type of leak or threat. You can identify carefully and audit it completely. The phishers, hackers and the pharmers are so smart today to access the data, digital data, financial data and important documents in the databases and access database servers smartly and cleaver way. To secure your vital information and to avoid any malicious or accidental threats, some data security techniques are highlighted and some of the important examples are:

1. Install the Antivirus softwares.
2. Install latest updates to purge data security techniques.
3. Do not use and even avoid to use Spyware and Adware
4. Always select and use unusual and tricky password.
5. Create a strong and powerful password.
6. Try to avoid Endanges of Emails and messages.
7. Always install Firewall and time to time verify it.
8. Always lock your important documents and files.

7.7 TWO PHASE LOCKING TECHNIQUES

Two-Phase Locking (2PL) is a concurrency control method which divides the execution phase of a transaction into three parts. It ensures conflict serializable schedules. If read and write operations introduce the first unlock operation in the transaction, then it is said to be Two-Phase Locking Protocol. This protocol can be divided into two phases:

- 1. In Growing Phase**, a transaction obtains locks, but may not release any lock.
- 2. In Shrinking Phase**, a transaction may release locks, but may not obtain any lock.

Two-Phase Locking does not ensure freedom from deadlocks.

Types of Two – Phase Locking Protocol:

Following are the types of two – phase locking protocol:

1. Strict Two – Phase Locking Protocol
2. Rigorous Two – Phase Locking Protocol
3. Conservative Two – Phase Locking Protocol

1. Strict Two-Phase Locking Protocol: Strict Two-Phase Locking Protocol avoids cascaded rollbacks. This protocol not only requires two-phase locking but also all exclusive-locks should be held until the transaction commits or aborts. It is not deadlock free. It ensures that if data is

being modified by one transaction, then other transaction cannot read it until first transaction commits. Most of the database systems implement rigorous two – phase locking protocol.

2. Rigorous Two-Phase Locking: Rigorous Two – Phase Locking Protocol avoids cascading rollbacks. This protocol requires that all the share and exclusive locks to be held until the transaction commits.

3. Conservative Two-Phase Locking Protocol: Conservative Two – Phase Locking Protocol is also called as Static Two – Phase Locking Protocol. This protocol is almost free from deadlocks as all required items are listed in advanced. It requires locking of all data items to access before the transaction starts.

7.8 SUMMARY

DBMS has different view from architectural point. A three-level architecture suggested by American National Standards Institute (ANSI) /Standards Planning and Requirements Committee (SPARC) in 1977. It is necessary to view data at different levels of abstraction. The three levels of the architecture are three different views of the data as *External, Conceptual, and Internal* view. Data Independence means data structure of the data is completely independent from the storage and accessing of the data. It is completely independent of any storage location, data types and accessing techniques. There are two levels of data independence as Physical Data Independence and Logical Data Independence. In the SQL language, a view is a representation of one or more tables. A view can be used to hide the complexity of relationships between tables or to provide security for sensitive data in tables. In general View is a part of table it describes only certain portion of a table. View are used to apply security to database by hiding certain portion of table. In SQL Views are kind of virtual tables. SQL joins are used to join two or more tables that is with the help of SQL joins we can find or retrieve an information from two or more than two tables. To delete or to remove a view of a relation or an object view of a table is the dropping of the view. This can be done easily by using the DROP VIEW clause or command or statement. To preserve the database integrity, confidentiality and availability, some tools, measures and controls are designed. The finally database is secure by doing so and the above said process is called Data Security. In the information technology and internet era, number of security threats occurs. So to protect useful information and crucial data, some of the data privacy mechanism and data security techniques should be implemented. So first thing is to identify the confidential data and protect it for any type of leak or threat. Two-Phase Locking (2PL) is a concurrency control method which divides the execution phase of a transaction into three parts. It ensures conflict serializable schedules. If read and write operations introduce the first unlock operation in the transaction, then it is said to be Two-Phase Locking Protocol.

7.9 PRACTICE EXERCISE

Q1. What do you mean by Views in RDBMS?

Q2. Explain all the three level in RDBMS?

- Q3. What do you mean by Data Independence?
- Q4. What are the various types of Data Independence?
- Q5. Differentiate between Physical Data Independence and Logical Data Independence?
- Q6. What do you mean by Join Views? What are the various Join Views statements?
- Q7. How can you drop a view?
- Q8. How can you drop a view?
- Q9. What do you know about database security?
- Q10. What are the various Database Security techniques?
- Q11. What do you mean by Two-phase Locking?
- Q12. What are the various types of two-phase locking schemes?

B.Sc.(DATA SCIENCE)

SEMESTER-III

DATABASE MANAGEMENT SYSTEM

UNIT VIII: DATA CONTROL OPERATIONS

STRUCTURE

8.0 Objectives

8.1 Data Control

8.2 Grant Command

8.3 Revoke Command

8.4 Commit and Rollback

8.5 Concurrency Control Techniques

8.6 Recovery Control Techniques

8.7 Summary

8.0 OBJECTIVES

To understand data controlling and management of data. Different commands associates with Data Control Languages.

8.1 DATA CONTROL

Management of information policies for handling the organization's information is the data control. Major Job of data control is to observe and report about working of data processing. It also handles all the managing issues and it is similar to data quality. Various types of operations or functions in data controls are inspection of data, notification, data validation, tracking, documentation and finally reporting.

Followings are the various drivers used for Data Control:

- **Compliance of data**
- **Auditability**
- **Transparency in data processing**
- **Reportability about validated data**
- **Agility**
- **Risk awareness if any occurs**

Key objectives in Data Control are as:

- To identify all the risks as soon as possible in data processing.
- To manage and implement all the policies and rules and regulations.
- It provides auditing of framework.
- To use the critical and complex data elements and also ensure the proper protection of data.
- Tracking and management of data quality for any service level agreements and services.
- All the data issues are well managed for the identification, remediation and reporting.

Various benefits using Data Control are:

- Controlling and monitoring of data between storage devices and various applications becomes easy.
- Before occurrences of any cause and complexity in data processing, data flaws are identified timely.
- It identify root causes of any issues and remediated timely.
- Identification of any loss of sensitive data (accidental or malicious) quickly.
- Backup of data before any operation can be done timely.

There are number of query languages operated on database in a RDBMS (Relational Database Management System)are SQL (Structured Query Language), Data Definition Language (DDL), Data Manipulation Language (DML), Data Query Language (DQL) and Data Control Language (DCL). Here we are discussing Data Control Language that is used for accessing the data stored in the Database. It is very useful for any updating or change in case of handling multiple databases to multiple users.

Need of Data Control Language (Why Control Language required):

Followings are some needs of data control language:

- Users in the database have the privileges to control it by using DCL.
- It control any unauthorized access to the database during updation of data.
- Only some of the DML and DDL commands for handling the database are controlled by DCL.
- It set the responsibilities to the DBA for use and controlling the data.
- DCL provides security to the database.

Working of Data Control Language (How Data Control Language Works)?

Here we see that how the statements work using the DCL:

- First of all it a big privilege to the users for accessing of various objects related with the database. Object and System are the two privileges for basic operations.
- At next any type of accessing or permission takes place during query processing to create tables, views, sessions etc.
- The system honors and give permission for performing CREATE, ALTER and DROP commands that is useful for operating the SELECT, UPDATE, INSERT, DELETE and EXECUTE commands on the data in handling various databases and database objects.
- In case of multiple users using the particular database environment, there is a difficulty in handling the data that solve by using the grant or revoke operations.
- Grant of roles to the users for using the data in the database is a privilege.
- Similarly using the revoke command data get revoked automatically for the user.

Advantages or Pros of DCL (Data Control Language):

Followings are the advantages of DCL:

- It provides the security to the data in database.
- DCL grant or remove number of permissions or the privileges to the various users for the database.

- Owner of the database or DBA grant or revoke the privileges for handling and managing any issues related with the database.

8.2 GRANT COMMAND

DCL commands are used to enforce database security in a multiple user database environment. Two types of DCL commands are GRANT and REVOKE. Only Database Administrator's or owner's of the database object can provide/remove privileges on a database object.

SQL GRANT Command: SQL GRANT is a command used to provide access or privileges on the database objects to the users. The Syntax for the GRANT command is:

GRANT privilege_name

ON object_name

TO {user_name |PUBLIC |role_name}

[WITH GRANT OPTION];

Here *privilege_name* is the privilege or access right granted to the user. This can be done

by SELECT, ALL and EXECUTE command. *object_name* is the database name to deals with the database object. The various database objects are TABLE, VIEW, STORED PROC and SEQUENCE. Here *user_name* is the name of the user to whom an access right is being granted. **PUBLIC** word is used to grant access rights to all users. **ROLES** are a set of privileges grouped together. **WITH GRANT OPTION** - allows a user to grant access rights to other users.

For Example:

```
GRANT SELECT ON employee TO user1;
```

This command grants a SELECT permission on employee table to user1. You should use the WITH GRANT option carefully because for example if you GRANT SELECT privilege on employee table to user1 using the WITH GRANT option, then user1 can GRANT SELECT privilege on employee table to another user, such as user2 etc. Later, if you REVOKE the SELECT privilege on employee from user1, still user2 will have SELECT privilege on employee table.

8.3 REVOKE COMMAND

The REVOKE command removes user access rights or privileges to the database objects. The Syntax for the REVOKE command is:

REVOKE privilege_name

ON object_name

FROM {user_name |PUBLIC |role_name }

For Example:

```
REVOKE SELECT ON employee FROM user1;
```

This command will REVOKE a SELECT privilege on employee table from user1. When you REVOKE SELECT privilege on a table from a user, the user will not be able to SELECT data from that table anymore. However, if the user has received SELECT privileges on that table from more than one users, he/she can SELECT from that table until everyone who granted the permission revokes it. You cannot REVOKE privileges if they were not initially granted by you.

Privileges and Roles:

Privileges: Privileges defines the access rights provided to a user on a database object. There are two types of privileges.

- 1) System privileges** - This allows the user to CREATE, ALTER, or DROP database objects.
- 2) Object privileges** - This allows the user to EXECUTE, SELECT, INSERT, UPDATE, or DELETE data from database objects to which the privileges apply.

Few CREATE system privileges are listed below:

System Privileges	Description
CREATE object	allows users to create the specified object in their own schema.
CREATE ANY object	allows users to create the specified object in any schema.

The above rules also apply for ALTER and DROP system privileges.

Few of the object privileges are listed below:

Object Privileges	Description
INSERT	allows users to insert rows into a table.
SELECT	allows users to select data from a database object.
UPDATE	allows user to update data in a table.
EXECUTE	allows user to execute a stored procedure or a function.

Roles: Roles are a collection of privileges or access rights. When there are many users in a database it becomes difficult to grant or revoke privileges to users. Therefore, if you define roles, you can grant or revoke privileges to users, thereby automatically granting or revoking privileges. You can either create Roles or use the system roles pre-defined by oracle. Some of the privileges granted to the system roles are as given below:

System Role	Privileges Granted to the Role
CONNECT	CREATE TABLE, CREATE VIEW, CREATE SYNONYM, CREATE SEQUENCE, CREATE SESSION etc.
RESOURCE	CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER etc. The primary usage of the RESOURCE role is to restrict access to database objects.
DBA	ALL SYSTEM PRIVILEGES

Creating Roles:

The Syntax to create a role is:

CREATE ROLE role-name
[IDENTIFIED BY password];

For Example: To create a role called "developer" with password as "pwd",the code will be as follows:

```
CREATE ROLE student-role-name
[IDENTIFIED BY pwd];
```

It's easier to GRANT or REVOKE privileges to the users through a role rather than assigning a privilege directly to every user. If a role is identified by a password, then, when

you GRANT or REVOKE privileges to the role, you definitely have to identify it with the password. We can GRANT or REVOKE privilege to a role as below.

For example: To grant CREATE TABLE privilege to a user by creating a testing role:

First, create a student Role:

```
CREATE ROLE Student
```

Second, grant a CREATE TABLE privilege to the ROLE testing. You can add more privileges to the ROLE.

```
GRANT CREATE TABLE TO testing;
```

Third, grant the role to a user.

```
GRANT testing TO user1;
```

To revoke a CREATE TABLE privilege from testing ROLE, you can write:

```
REVOKE CREATE TABLE FROM testing;
```

The Syntax to drop a role from the database is as below:

```
DROP ROLE role_name;
```

For example: To drop a role called developer, you can write:

```
DROP ROLE testing
```

Difference between GRANT and REVOKE Commands:

GRANT	REVOKE
It grants permissions to the user on the data using the database objects	It removes permissions if any granted to the users on database objects.
It assigns access rights to users.	It revokes the user access rights of users.
For each user you need to specify the permissions.	If access for one user is removed; all the particular permissions provided by that users to others will be removed.
When the access is decentralized granting permissions will be easy.	If decentralized access removing the granted permissions is difficult.

8.4 COMMIT AND ROLLBACK

The process of placing into effect in the database the updates made by a transaction is called *commit*. The process of invalidating the updates made by a transaction is called *rollback*. When commit and rollback occur is determined as follows:

- At the times defined by an SQL
- At the times that are set automatically by HiRDB

Each of these types of commit and rollback timing is explained as follows.

(1) Commit timing

(2) Rollback timing

(3) Commitment control on a HiRDB/Parallel Server

(1) Commit timing:

This section explains commit timing.

Timing defined by an SQL: The COMMIT control SQL statement can be specified to commit a transaction whenever a COMMIT statement is executed.

Commit set automatically by HiRDB: Commit is performed automatically by HiRDB when a definition SQL or PURGE TABLE statement is executed. Commit is performed automatically by HiRDB when a UAP terminates.

(2) Rollback timing:

This section explains rollback timing:

Timing defined by an SQL: The ROLLBACK control SQL statement can be specified to roll back the process to the previous commit point whenever a ROLLBACK statement is executed.

Rollback set automatically by HiRDB: If a process cannot continue during SQL execution, HiRDB will roll back the process implicitly to the previous commit point. If a UAP terminates abnormally, HiRDB will roll back the process to the previous commit point.

(3) Commitment control on a HiRDB/Parallel Server:

A HiRDB/Parallel Server provides the following two methods of commitment control:

(a) One-phase commit:

With one-phase commit, only commit processing is performed, rather than both prepare processing and commit processing (which is the case with two-phase commitment control). This means that the number of communication transactions for synchronization point processing between the front-end server and the back-end server (dictionary server) is the number of branches $\times 2$ (whereas, with two-phase commit, it is the number of branches $\times 4$), which improves transaction processing performance. To use one-phase commit, specify ONEPHASE (default) in the `pd_trn_commit_optimize` operand. One-phase commit is

used only when one branch within a single transaction is being updated. Otherwise, two-phase commit must be used.

(b) Two-phase commit:

Two-phase commit performs synchronization point processing of the transaction by separating commitment control into two phases, prepare processing and commit processing. The number of communication transactions for synchronization point processing between the front-end server and the back-end server (dictionary server) is the number of branches \times 4. Figure 6-23 shows the processing for two-phase commit.

1. COMMIT:

COMMIT in SQL is a transaction control language which is used to permanently save the changes done in the transaction in tables/databases. The database cannot regain its previous state after the execution of it. Consider the following STAFF table with records:

STAFF

Id	Name	Age	Allowance	Salary
1	Rahul	26	400	4000
2	Khaitan	46	900	9000
3	Munjhal	36	400	4500
4	Ram	28	800	8000
5	Manav	24	400	6500
6	Kaira	21	700	7800

Example:

```
sql>
```

```
SELECT *
```

```
FROM Staff
```

```
WHERE Allowance = 400;
```

```
sql> COMMIT;
```

Output:

Id	Name	Age	Allowance	Salary
1	Rahul	26	4000	400
3	Munjhal	36	4500	400
5	Manav	24	6500	400

So, the SELECT statement produced the output consisting of three rows.

2. ROLLBACK:

ROLLBACK in SQL is a transactional control language which is used to undo the transactions that have not been saved in database. The command is only be used to undo changes since the last COMMIT. Consider the following STAFF table with records:

STAFF

Id	Name	Age	Allowance	Salary
1	Rahul	26	400	4000
2	Khaitan	46	900	9000
3	Munjhal	36	400	4500
4	Ram	28	800	8000
5	Manav	24	400	6500
6	Kaira	21	700	7800

Example:

sql>

SELECT *

FROM EMPLOYEES

WHERE ALLOWANCE = 400;

sql> ROLLBACK;

Output:

Id	Name	Age	Allowance	Salary
1	Rahul	26	400	4000
2	Khaitan	46	900	9000
3	Munjhal	36	400	4500
4	Ram	28	800	8000
5	Manav	24	400	6500
6	Kaira	21	700	7800

So, the SELECT statement produced the same output with ROLLBACK command.

Difference between COMMIT and ROLLBACK:

COMMIT	ROLLBACK
COMMIT permanently saves the changes made by current transaction.	ROLLBACK undo the changes made by current transaction.
Transaction can not undo changes after COMMIT execution.	Transaction reaches its previous state after ROLLBACK.

8.5 CONCURRENCY CONTROL TECHNIQUES

Concurrency Control in Database Management System is a procedure of managing simultaneous operations without conflicting with each other. It ensures that Database transactions are performed concurrently and accurately to produce correct results without violating data integrity of the respective Database.

Concurrent access is quite easy if all users are just reading data. There is no way they can interfere with one another. Though for any practical Database, it would have a mix of READ and WRITE operations and hence the concurrency is a challenge.

DBMS Concurrency Control is used to address such conflicts, which mostly occur with a multi-user system. Therefore, Concurrency Control is the most important element for proper functioning of a Database Management System where two or more database transactions are executed simultaneously, which require access to the same data.

Potential problems of Concurrency:

Here, are some issues which you will likely to face while using the DBMS Concurrency Control method:

- **Lost Updates** occur when multiple transactions select the same row and update the row based on the value selected
- Uncommitted dependency issues occur when the second transaction selects a row which is updated by another transaction (**dirty read**)
- **Non-Repeatable Read** occurs when a second transaction is trying to access the same row several times and reads different data each time.
- **Incorrect Summary issue** occurs when one transaction takes summary over the value of all the instances of a repeated data-item, and second transaction update few instances of that specific data-item. In that situation, the resulting summary does not reflect a correct result.

Why use Concurrency method?

Reasons for using Concurrency control method is DBMS:

- To apply Isolation through mutual exclusion between conflicting transactions
- To resolve read-write and write-write conflict issues
- To preserve database consistency through constantly preserving execution obstructions
- The system needs to control the interaction among the concurrent transactions. This control is achieved using concurrent-control schemes.
- Concurrency control helps to ensure serializability

Example: Assume that two people who go to electronic kiosks at the same time to buy a movie ticket for the same movie and the same show time.

However, there is only one seat left in for the movie show in that particular theatre. Without concurrency control in DBMS, it is possible that both moviegoers will end up purchasing a ticket. However, concurrency control method does not allow this to happen. Both moviegoers

can still access information written in the movie seating database. But concurrency control only provides a ticket to the buyer who has completed the transaction process first.

Concurrency Control Protocols:

Different concurrency control protocols offer different benefits between the amount of concurrency they allow and the amount of overhead that they impose. Following are the Concurrency Control techniques in DBMS:

- Lock-Based Protocols
- Two Phase Locking Protocol
- Timestamp-Based Protocols
- Validation-Based Protocols

i). Lock-based Protocols:

Lock Based Protocols in DBMS is a mechanism in which a transaction cannot Read or Write the data until it acquires an appropriate lock. Lock based protocols help to eliminate the concurrency problem in DBMS for simultaneous transactions by locking or isolating a particular transaction to a single user.

A lock is a data variable which is associated with a data item. This lock signifies that operations that can be performed on the data item. Locks in DBMS help synchronize access to the database items by concurrent transactions. All lock requests are made to the concurrency-control manager. Transactions proceed only once the lock request is granted.

Binary Locks: A Binary lock on a data item can either be locked or unlocked.

Shared/exclusive: This type of locking mechanism separates the locks in DBMS based on their uses. If a lock is acquired on a data item to perform a write operation, it is called an exclusive lock.

1. Shared Lock (S):

A shared lock is also called a Read-only lock. With the shared lock, the data item can be shared between transactions. This is because you will never have permission to update data on the data item. For example, consider a case where two transactions are reading the account balance of a person. The database will let them read by placing a shared lock. However, if another transaction wants to update that account's balance, the shared lock prevents it until the reading process is over.

2. Exclusive Lock (X):

With the Exclusive Lock, a data item can be read as well as written. This is exclusive and can't be held concurrently on the same data item. X-lock is requested using lock-x instruction. Transactions may unlock the data item after finishing the 'write' operation. For example, when a transaction needs to update the account balance of a person. You can allow this transaction by

placing X lock on it. Therefore, when the second transaction wants to read or write, exclusive lock prevent this operation.

3. Simplistic Lock Protocol:

This type of lock-based protocols allows transactions to obtain a lock on every object before beginning operation. Transactions may unlock the data item after finishing the 'write' operation.

4. Pre-claiming Locking:

Pre-claiming lock protocol helps to evaluate operations and create a list of required data items which are needed to initiate an execution process. In the situation when all locks are granted, the transaction executes. After that, all locks release when all of its operations are over.

Starvation:

Starvation is the situation when a transaction needs to wait for an indefinite period to acquire a lock. Following are the reasons for Starvation:

- When waiting scheme for locked items is not properly managed
- In the case of resource leak
- The same transaction is selected as a victim repeatedly

Deadlock:

Deadlock refers to a specific situation where two or more processes are waiting for each other to release a resource or more than two processes are waiting for the resource in a circular chain.

ii). Two Phase Locking Protocol:

Two Phase Locking Protocol also known as 2PL protocol is a method of concurrency control in DBMS that ensures serializability by applying a lock to the transaction data which blocks other transactions to access the same data simultaneously. Two Phase Locking protocol helps to eliminate the concurrency problem in DBMS. This locking protocol divides the execution phase of a transaction into three different parts.

- In the first phase, when the transaction begins to execute, it requires permission for the locks it needs.
- The second part is where the transaction obtains all the locks. When a transaction releases its first lock, the third phase starts.
- In this third phase, the transaction cannot demand any new locks. Instead, it only releases the acquired locks.

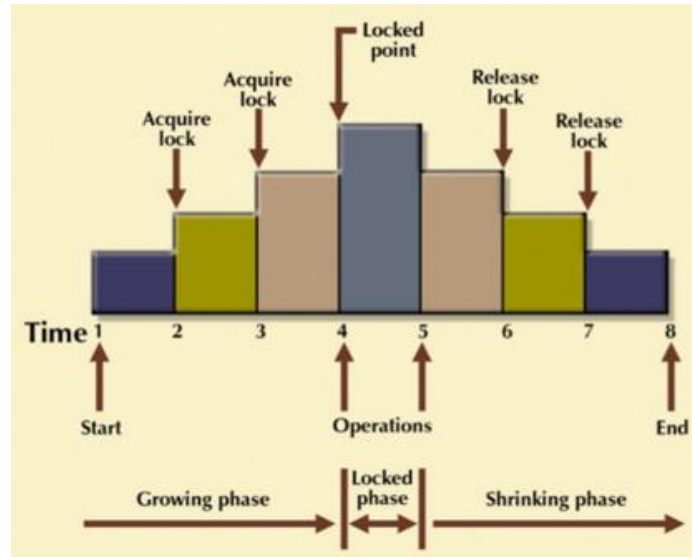


Figure 8.1

The Two-Phase Locking protocol allows each transaction to make a lock or unlock request in two steps:

- **Growing Phase:** In this phase transaction may obtain locks but may not release any locks.
- **Shrinking Phase:** In this phase, a transaction may release locks but not obtain any new lock

It is true that the 2PL protocol offers serializability. However, it does not ensure that deadlocks do not happen. In the above-given diagram, you can see that local and global deadlock detectors are searching for deadlocks and solve them with resuming transactions to their initial states.

Strict Two-Phase Locking Method:

Strict-Two phase locking system is almost similar to 2PL. The only difference is that Strict-2PL never releases a lock after using it. It holds all the locks until the commit point and releases all the locks at one go when the process is over.

Centralized 2PL:

In Centralized 2 PL, a single site is responsible for lock management process. It has only one lock manager for the entire DBMS.

Primary copy 2PL:

Primary copy 2PL mechanism, many lock managers are distributed to different sites. After that, a particular lock manager is responsible for managing the lock for a set of data items. When the primary copy has been updated, the change is propagated to the slaves.

Distributed 2PL:

In this kind of two-phase locking mechanism, Lock managers are distributed to all sites. They are responsible for managing locks for data at that site. If no data is replicated, it is equivalent to primary copy 2PL. Communication costs of Distributed 2PL are quite higher than primary copy 2PL

iii). Timestamp-based Protocols:

Timestamp based Protocol in DBMS is an algorithm which uses the System Time or Logical Counter as a timestamp to serialize the execution of concurrent transactions. The Timestamp-based protocol ensures that every conflicting read and write operations are executed in a timestamp order. The older transaction is always given priority in this method. It uses system time to determine the time stamp of the transaction. This is the most commonly used concurrency protocol.

Lock-based protocols help you to manage the order between the conflicting transactions when they will execute. Timestamp-based protocols manage conflicts as soon as an operation is created. Example: Suppose there are three transactions T1, T2, and T3.

T1 has entered the system at time 0010

T2 has entered the system at 0020

T3 has entered the system at 0030

Priority will be given to transaction T1, then transaction T2 and lastly Transaction T3.

Advantages:

- Schedules are serializable just like 2PL protocols
- No waiting for the transaction, which eliminates the possibility of deadlocks!

Disadvantages:

Starvation is possible if the same transaction is restarted and continually aborted

iv). Validation Based Protocol:

Validation based Protocol in DBMS also known as Optimistic Concurrency Control Technique is a method to avoid concurrency in transactions. In this protocol, the local copies of the transaction data are updated rather than the data itself, which results in less interference while execution of the transaction. The Validation based Protocol is performed in the following three phases:

1. Read Phase
2. Validation Phase
3. Write Phase

Read Phase:

In the Read Phase, the data values from the database can be read by a transaction but the write operation or updates are only applied to the local data copies, not the actual database.

Validation Phase:

In Validation Phase, the data is checked to ensure that there is no violation of serializability while applying the transaction updates to the database.

Write Phase:

In the Write Phase, the updates are applied to the database if the validation is successful, else; the updates are not applied, and the transaction is rolled back.

Characteristics of Good Concurrency Protocol:

An ideal concurrency control DBMS mechanism has the following objectives:

- Must be resilient to site and communication failures.
- It allows the parallel execution of transactions to achieve maximum concurrency.
- Its storage mechanisms and computational methods should be modest to minimize overhead.
- It must enforce some constraints on the structure of atomic actions of transactions.

8.6 RECOVERY CONTROL TECHNIQUES

The Databases are prone to failures due to inconsistency, network failure, errors or any kind of accidental damage. So, database recovery techniques are highly important to bring a database back into a working state after a failure. There can be any case in database system like any computer system when database failure happens. So data stored in database should be available all the time whenever it is needed. So database recovery means recovering the data when it get deleted, hacked or damaged accidentally. Atomicity is must whether is transaction is over or not it should reflect in the database permanently or it should not effect the database at all. So database recovery and database recovery techniques are must in DBMS.

Database systems, like any other computer system, are subject to failures but the data stored in it must be available as and when required. When a database fails it must possess the facilities for fast recovery. It must also have atomicity i.e. either transactions are completed successfully and committed (the effect is recorded permanently in the database) or the transaction should have no effect on the database.

There are both automatic and non-automatic ways for both, backing up of data and recovery from any failure situations. The techniques used to recover the lost data due to system crash, transaction errors, viruses, catastrophic failure, incorrect commands execution etc. are database recovery techniques. So to prevent data loss recovery techniques based on deferred update and immediate update or backing up data can be used.

There are four different **recovery techniques** are available in the Database.

1. *Mirroring*
2. *Recovery using Backups*
3. *Recovery using Transaction Logs*
4. *Shadow Paging*

1. **Mirroring:**

Two complete copies of the database maintains on-line on different stable storage devices. This method mostly uses in environments that require non-stop, fault-tolerant operations.

2. **Recovery using Backups:**

Backups are useful if there has been extensive damage to database. Backups are mainly two types :

i). Immediate Backup:

Immediate Backup are kept in a floppy disk, hard disk or magnetic tapes. These come in handy when a technical fault occurs in the primary database such as system failure, disk crashes, network failure. Damage due to virus attacks repair using the immediate backup.

ii). Archival Backup:

Archival Backups are kept in mass storage devices such as magnetic tape, CD-ROMs, Internet Servers etc. They are very useful for recovering data after a disaster such as fire, earthquake, flood etc. Archival Backup should be kept at a different site other than where the system is functioning. Archival Backup at a separate place remains safe from thefts and intentional destruction by user staff.

3. **Recovery using Transaction Logs:**

In Recovery using Transaction Logs, some following steps are :

Step1: The log searches for all the transaction that have recorded a

[**start transaction**, ‘ ‘] entry, but haven’t recorded a corresponding [**commit**, ‘ ‘] entry.

Step2: These transactions are rolling back.

Step3: Transactions which have recorded a [**commit**, ‘ ‘] entry in the log, it must have recorded the changes, they did to the database in the log. These change will follow to undo their effects on the database.

4. **Shadow Paging:**

These system can use for data recovery instead of using transaction logs. In the Shadow Paging, a database is divided into several fixed-sized disk pages, say n, thereafter a current directory creates. It having n entries with each entry pointing to a disk page in the database. the current directory transfer to the main memory. When a transaction begins executing, the current directory copies into a shadow directory. Then, the shadow directory saves on the disk. The

transaction will be using the current directory. During the transaction execution, all the modifications are made on the current directory and the shadow directory is never modified.

Checkpoint-Recovery is a common technique for imbuing a program or system with fault tolerant qualities, and grew from the ideas **used** in systems which employ transaction processing [lyu95]. It allows systems to **recover** after some fault interrupts the system, and causes the task to fail, or be aborted in some way.

Recovery and Atomicity:

When a system crashes, it may have several transactions being executed and various files opened for them to modify the data items. Transactions are made of various operations, which are atomic in nature. But according to ACID properties of DBMS, atomicity of transactions as a whole must be maintained, that is, either all the operations are executed or none. When a DBMS recovers from a crash, it should maintain the following:

- It should check the states of all the transactions, which were being executed.
- A transaction may be in the middle of some operation; the DBMS must ensure the atomicity of the transaction in this case.
- It should check whether the transaction can be completed now or it needs to be rolled back.
- No transactions would be allowed to leave the DBMS in an inconsistent state.

There are two types of techniques, which can help a DBMS in recovering as well as maintaining the atomicity of a transaction:

- Maintaining the logs of each transaction, and writing them onto some stable storage before actually modifying the database.
- Maintaining shadow paging, where the changes are done on a volatile memory, and later, the actual database is updated.

Log-based Recovery:

Log is a sequence of records, which maintains the records of actions performed by a transaction. It is important that the logs are written prior to the actual modification and stored on a stable storage media, which is failsafe. Log-based recovery works as follows –

- The log file is kept on a stable storage media.
- When a transaction enters the system and starts execution, it writes a log about it.

$\langle T_n, \text{Start} \rangle$

- When the transaction modifies an item X, it write logs as follows –

$\langle T_n, X, V_1, V_2 \rangle$

It reads T_n has changed the value of X, from V_1 to V_2 .

- When the transaction finishes, it logs –

$\langle T_n, \text{commit} \rangle$

The database can be modified using two approaches –

- **Deferred database modification** – All logs are written on to the stable storage and the database is updated when a transaction commits.
- **Immediate database modification** – Each log follows an actual database modification. That is, the database is modified immediately after every operation.

Recovery with Concurrent Transactions:

When more than one transaction are being executed in parallel, the logs are interleaved. At the time of recovery, it would become hard for the recovery system to backtrack all logs, and then start recovering. To ease this situation, most modern DBMS use the concept of 'checkpoints'.

Checkpoint:

Keeping and maintaining logs in real time and in real environment may fill out all the memory space available in the system. As time passes, the log file may grow too big to be handled at all. Checkpoint is a mechanism where all the previous logs are removed from the system and stored permanently in a storage disk. Checkpoint declares a point before which the DBMS was in consistent state, and all the transactions were committed.

Recovery:

When a system with concurrent transactions crashes and recovers, it behaves in the following manner:

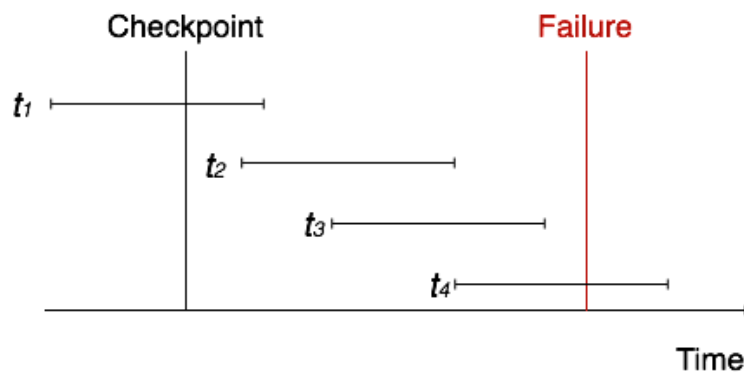


Figure 8.2

- The recovery system reads the logs backwards from the end to the last checkpoint.
- It maintains two lists, an undo-list and a redo-list.
- If the recovery system sees a log with $\langle T_n, \text{Start} \rangle$ and $\langle T_n, \text{Commit} \rangle$ or just $\langle T_n, \text{Commit} \rangle$, it puts the transaction in the redo-list.

- If the recovery system sees a log with $\langle T_n, \text{Start} \rangle$ but no commit or abort log found, it puts the transaction in undo-list.

All the transactions in the undo-list are then undone and their logs are removed. All the transactions in the redo-list and their previous logs are removed and then redone before saving their logs.

8.7 SUMMARY

Management of information policies for handling the organization's information is the data control. There are number of query languages operated on database in a RDBMS (Relational Database Management System) are SQL (Structured Query Language), Data Definition Language (DDL), Data Manipulation Language (DML), Data Query Language (DQL) and Data Control Language (DCL). Here we are discussing Data Control Language that is used for accessing the data stored in the Database. It is very useful for any updating or change in case of handling multiple databases to multiple users. DCL commands are used to enforce database security in a multiple user database environment. Two types of DCL commands are GRANT and REVOKE. SQL GRANT is a command used to provide access or privileges on the database objects to the users. The REVOKE command removes user access rights or privileges to the database objects. The process of placing into effect in the database the updates made by a transaction is called *commit*. The process of invalidating the updates made by a transaction is called *rollback*. Concurrency Control in Database Management System is a procedure of managing simultaneous operations without conflicting with each other. It ensures that Database transactions are performed concurrently and accurately to produce correct results without violating data integrity of the respective Database. Concurrency Control techniques in DBMS are Lock-Based Protocols, Two Phase Locking Protocol, Timestamp-Based Protocols and Validation-Based Protocols. The Databases are prone to failures due to inconsistency, network failure, errors or any kind of accidental damage. So, database recovery techniques are highly important to bring a database back into a working state after a failure. So database recovery and database recovery techniques are must in DBMS. There are four different recovery techniques are available in the Database are *Mirroring*, *Recovery using Backups*, *Recovery using Transaction Logs*, *Shadow Paging*.

Question for Practice

- Q1. What do you know about Data Control?
- Q2. What are the various Data Control Statements?
- Q3. What do you know about Grant command? How it works?
- Q4. What do you know about Revoke command? How it works?
- Q5. What do you know about Commit command? How it works?

Q6. What do you know about Rollback command? How it works?

Q7. Differentiate between Grant and Revoke Command?

Q8. Differentiate between Commit and Rollback command?

Q9. What do you mean by concurrency control?

Q10. What are the various Concurrency Control techniques?

Q11. What are the various recovery control techniques?