RASHTRASANT TUKADOJI MAHARAJ NAGPUR UNIVERSITY, NAGPUR

FOUR YEAR BACHELOR OF TECHNOLOGY (B.Tech.) DEGREE COURSE

SEMESTER: SIXTH (C.B.C.S)

BRANCH: INFORMATION TECHNOLOGY

Subject Name: Database Management System Lab **Subject code: BEIT601P**

Load	Practical	Credits	College Assessment Marks	University Evaluation	Total Marks
2Hrs (Practical)	2	1	25	25	50

Aim: To acquire the knowledge of database queries & apply it in real life example.

Prerequisite(s): - Discrete Mathematics, Data Structures & Program Design

Course Objectives

1	**To understand the fundamental concepts of Database Management Systems**
2	**To acquire knowledge of database query languages and transaction processing**
3	**To understand systematic database design approaches.**

Course Outcome:

At the end of this course, students are able to:

CO1	**Analyze and design Database Management System using ER model**
CO2	**Apply Indexing concepts in database languages**
CO3	**Implement database queries using database languages**
CO4	**Create normalized database design using normal forms**
CO5	**Apply Transaction Management concepts in real-time situations**

K. D. K. COLLEGE OF ENGINEERING, NAGPUR
IMPARTING TECHNICAL EDUCATION SINCE 1984
NBA Accredited - Five programs:(Civil Engg., Mechanical Engg., Electrical Engg. Comp. Sc. & Engg., Electronics & Telecomm. Engg.) Grade A, NAAC Accredited

Department of Information Technology

Semester: VI

Name of Subject: Database Management System Lab

Subject Code: BEIT601P

Course Outcome:

At the end of this course, students are able to:

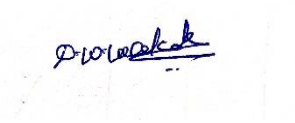
CO1	**Analyze and design Database Management System using ER model**
CO2	**Apply Indexing concepts in database languages**
CO3	**Implement database queries using database languages**
CO4	**Create normalized database design using normal forms**
CO5	**Apply Transaction Management concepts in real-time situations**

List of Practicals: -

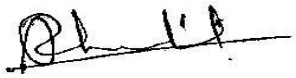
1	**ER Modeling and Normalization**
2	**Program on inheritance , using method overriding and also using "this" keyword**
3	**SQL Queries: DML**
4	**SQL Queries: TCL**
5	**SQL Queries – all types of Join, Sub-Query**
6	**PL/SQL Stored Procedure and Stored Function**
7	**Database Trigger**
8	**Database Connectivity**
9	**Mini Project**

LIST OF PROGRAMS

1. Introduction to Database Management System.
2. To study ER Model.
3. To perform DDL, DML Commands in SQL.
4. To perform aggregate functions in SQL.
5. To Perform Join and Set Operation in SQL.
6. To Perform Various Arithmetical, Relational & Logical Operators in SQL.
7. To perform various Clauses in SQL.
8. To implement integrity constraints in SQL.
9. To create View and Materialized view in SQL.
10. To Create a User and Grant Permissions in Oracle.
11. To create functions using PL/SQL.
12. To create procedures using PL/SQL.
13. To implement triggers in PL/SQL.



Prof. D.W.Wakde
Subject Teacher



DR. S. P. Khandait
Head of Department

Practical No. 01

PRACTICAL NO 1

Aim: Introduction to Database Management system

Theory:

1. **Database**

A database is a logically coherent collection of data with some inherent meaning, representing some aspect of real world and which is designed, built and populated with data for a specific purpose.

2.DBMS

It is a collection of programs that enables user to create and maintain a database. In other words it is general-purpose software that provides the users with the processes of defining, constructing and manipulating the database for various applications

3. Advantages of DBMS

The database management system has a number of advantages as compared to traditional computer file-based processing approach. The DBA must keep in mind these benefits or capabilities during databases and monitoring the DBMS.

The Main advantages of DBMS are described below.

a**. Controlling Data Redundancy**

In non-database systems each application program has its own private files. In this case, the duplicated copies of the same data is created in many places. In DBMS, all data of an organization is integrated into a single database file. The data is recorded in only one place in the database and it is not duplicated.

b. Sharing of Data

In DBMS, data can be shared by authorized users of the organization. The database administrator manages the data and gives rights to users to access the data. Many users can be authorized to access the same piece of information simultaneously. The remote users can also share same data. Similarly, the data of same database can be shared between different application programs.

c. Data Consistency

By controlling the data redundancy, the data consistency is obtained. If a data item appears only once, any update to its value has to be performed only once and the updated value is immediately available to all users. If the DBMS has controlled redundancy, the database system enforces consistency.

d. Integration of Data

In Database management system, data in database is stored in tables. A single database contains multiple tables and relationships can be created between tables (or associated data entities). This makes easy to retrieve and update data.

e. Integration Constraints

Integrity constraints or consistency rules can be applied to database so that the correct data can be entered into database. The constraints may be applied to data item within a single record or the may be applied to relationships between records.

f. Data Security

Form is very important object of DBMS. You can create forms very easily and quickly in DBMS. Once a form is created, it can be used many times and it can be modified very easily. The created forms are also saved along with database and behave like a software component. A form provides very easy way (user-friendly) to enter data into database, edit data and display data from database. The non-technical users can also perform various operations on database through forms without going into technical details of a fatabase.

g. Control Over Concurrency

In a computer file-based system, if two users are allowed to access data simultaneously, it is possible that they will interfere with each other. For example, if both users attempt to perform update operation on the same record, then one may overwrite the values recorded by the other. Most database management systems have sub-systems to control the concurrency so that transactions are always recorded with accuracy.

i. Backup and Recovery Procedures

In a computer file-based system, the user creates the backup of data regularly to protect the valuable data from damage due to failures to the computer system or application program. It is very time consuming method, if amount of data is large. Most of the DBMSs provide the 'backup and recovery' sub-systems that automatically create the backup of data and restore data if required.

j. Data Independence

The separation of data structure of database from the application program that uses the data is called data independence. In DBMS, you can easily change the structure of database without modifying the application program.

4. Disadvantages of Database Management System (DBMS):

Although there are many advantages but the DBMS may also have some minor disadvantages. These are:

1. Cost of Hardware & Software:

A processor with high speed of data processing and memory of large size is required to run the DBMS software. It means that you have to upgrade the hardware used for file-based system. Similarly, DBMS software is also Very costly.

2. Cost of Data Conversion:

When a computer file-based system is replaced with a database system, the data stored into data file must be converted to database files. It is difficult and time consuming method to convert data of data files into database. You have to hire DBA (or database designer) and system designer along with application programmers; alternatively, you have to take the services of some software houses. So a lot of money has to be paid for developing database and related software.

3. Cost of Staff Training:
Most DBMSs are often complex systems so the training for users to use the DBMS is required. Training is required at all levels, including programming, application development, and database administration. The organization has to pay a lot of amount on the training of staff to run the DBMS.

4. Appointing Technical Staff:
The trained technical persons such as database administrator and application programmers etc are required to handle the DBMS. You have to pay handsome salaries to these persons. Therefore, the system cost increases.

5. Database Failures:
In most of the organizations, all data is integrated into a single database. If database is corrupted due to power failure or it is corrupted on the storage media, then our valuable data may be lost or whole system stops.

5. Disadvantages of File Processing System

Data Redundancy
Data Redundancy means same information is duplicated in several files. This makes data redundancy.

Data Inconsistency
Data Inconsistency means different copies of the same data are not matching. That means different versions of same basic data are existing. This occurs as the result of update operations that are not updating the same data stored at different places.
Example: Address Information of a customer is recorded differently in different files.

Difficulty in Accessing Data
It is not easy to retrieve information using a conventional file processing system. Convenient and efficient information retrieval is almost impossible using conventional file processing system.

Data Isolation
Data are scattered in various files, and the files may be in different format, writing new application program to retrieve data is difficult.

Integrity Problems
The data values may need to satisfy some integrity constraints. For example the balance field Value must be grater than 5000. We have to handle this through program code in file processing systems. But in database we can declare the integrity constraints along with definition itself.

Atomicity Problem
It is difficult to ensure atomicity in file processing system.For example transferring $100 from Account A to account B.If a failure occurs during execution there could be situation like $100 is deducted from Account A and not credited in Account B.

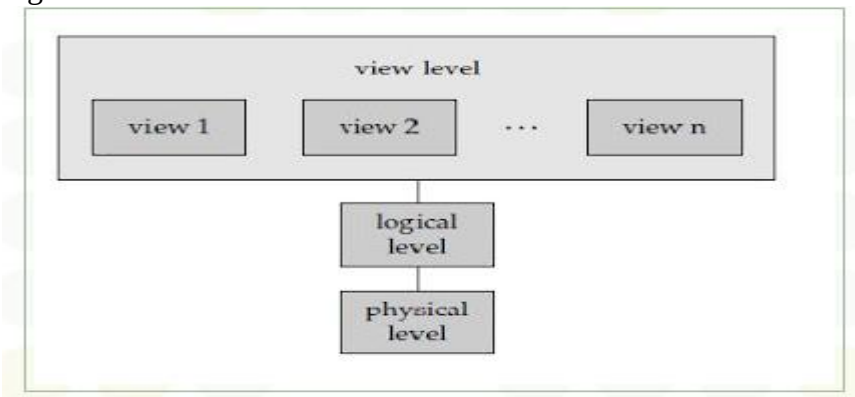
Concurrent Access anomalies
If multiple users are updating the same data simultaneously it will result in inconsistent data state. In file processing system it is very difficult to handle this using program code. This results in concurrent access anomalies.

Security Problems
Enforcing Security Constraints in file processing system is very difficult as the application programs are added to the system in an ad-hoc manner.

6. Data Abstraction
For the system to be usable, it must retrieve data efficiently. The need for efficiency has led designers to use complex data structures to represent data in the database. Since many database-systems users are not computer trained, developers hide the complexity from users through several levels of abstraction, to simplify users' interactions with the system:

Physical Level: The lowest level of abstraction describes how the data are actually stored. The physical level describes complex low-level data structures in detail.

Logical Level: The next-higher level of abstraction describes what data are stored in the database, and what relationships exist among those data. The logical level thus describes the entire database in terms of a small number of relatively simple structures. Although implementation of the simple structures at the logical level may involve complex physical-level structures, the user of the logical level does not need to be aware of this complexity. Database administrators, who must decide what information to keep in the database, use the logical level of abstraction.



The three levels of data abstraction

The three levels of data abstraction

View Level: The highest level of abstraction describes only part of the entire database. Even though the logical level uses simpler structures, complexity remains because of the variety of information stored in a large database. Many users of the database system do not need all this information; instead, they need to access only a part of the database. The view level of abstraction exists to simplify their interaction with the system. The system may provide many views for the same database.

7. Elements of a Database Management System?

The major components of a DBMS are explained below:

1. DML Precompiled

It converts DML statement embedded in an application program to normal procedure calls in the host language. The precompiled must interact with the query processor in order to generate the appropriate code.

2. DDL Compiler

The DDL compiler converts the data definition statements into a set of tables. These tables contain information concerning the database and are in a form that can be used by other components of the DBMS.

3. File Manager

File manager manages the allocation of space on disk storage and the data structure used to represent information stored on disk. The file manager can be implemented using an interface to the exiting file subsystem provided by the operating system of the host computer or it can include a file subsystem written especially for the DBMS.

4. Database Manager

Databases typically require a large amount of storage space. The important responsibilities of Database manger Are:

1 **Interaction with File Manager:** The raw data is stored on the disk using the file

system which is usually provided by a conventional operating system. The databasemanager translates the various DML statements into low-level file system commands. Thus the database manager is responsible for the actual storing, retrieving and updating of data in the database.

2 **Integrity enforcement :** The data values stored in the database must satisfy certain types of consistency constrains. For example, the balance of a bank account may never fall below a prescribed amount (for example Rs.200). similarly the number of holidays per year an employee may be having should not exceed 25 days. These constraints must specified explicitly by the DBA. If such constraints are specified, then the database manager can check whether updates to the database result in the violation of any of these constraints and if so appropriate action may be imposed.

3 **Security enforcement:** As discussed above, not every user of the database needs to have access to the entire content of the database. It is the job of the database manager to enforce these security requirements.

4 **Backup and recovery:** A computer system like any other mechanical or electrical device, is subject to failure. These are a variety of causes of such failure, including disk crash, power failure and s/w errors. In each of these cases, information concerning he database is lost. It is the responsibility of database manager to detect such failures and restore the database to a state that existed prior to the occurrence of the failure. This is usually accomplished through the backup and recovery procedures.

5 **Concurrency control:** When several users update the database concurrently, the consistency of data may no longer be preserved. It is necessary for the system to control the interaction among the concurrent users, and achieving such a control is one of the responsibilities of database manager.

5. Query Processor

The database user retrieves data by formulating a query in the data manipulating language provided with the database. The query processor is used to interpret the online user's query and convert it into an efficient series of operations in a form capable of being sent to the data manager for execution. The query processor uses the data dictionary to find the structure of the relevant portion of the database and uses this information in modifying the query and preparing an optimal plan to access the database.

6. Data Administrator

One of the main reasons for having database management system is to have control of both data an programs accessing that data. The person having such control over the system is called the database administrator

The functions of DBA are :

1 **Schema definition :** The creation of the original database schema. This is accomplished by writing a set of definition which are translated by the DDL compiler to a set of tables that are permanently stored in the data dictionary.

2 **Storage Structure and access method definition :** The creation of appropriate storage structure and access method. This is accomplished by writing a set of definitions which are translated by the data storage and definition language compiler.

3 **Schema and Physical Organization Modification :** Either the modification of the

database schema or the description of the physical storage organization. These changes, although relatively rare, are accomplished by writing a set of definition which are used by either the DDL compiler or the data storage and definition language compiler to generate modification to the appropriate internal system tables (for example the data dictionary).

4 **Granting of authorization for data access :** The granting of different types of authorization for data access to the various users of the database.

5 **Integrity Constraint Specification :** These constraints are kept in a special system structure that is consulted by the database manager whenever one of the valuable tools that the DBA uses to carry out data administration in data dictionary.

7. Data Dictionary

A comprehensive data dictionary would provide the definition of data item, how they fit into the data structure and how they relate to other entities in the database. With the comprehensive base of information the data dictionary can serve several useful purposes connecting across the whole spectrum of planning, determining information requirement, designing and implementation operation and revision. The DBA uses the data dictionary in every phase of a database life cycle, starting from the embryonic data gathering phase to the design, implementation and maintenance phase. Documentation provided by a data dictionary is as valuable to end users and managers as it provided by a data dictionary is as valuable to end users and managers as it is essential to the programmers. Thus users and managers will be able to see exactly what is available in the database. You could consider a data dictionary to be road map which guides users to access information within a large database.

Q. 8. Role & Responsibilities of Database Administrator?

The **database administrator** is a **person** having central control over data and programs accessing that data. Duties of the database administrator include:

- **Scheme definition:** the creation of the original database scheme. This involves writing a set of definitions in a DDL (data storage and definition language), compiled by the DDL compiler into a set of tables stored in the data dictionary.
- **Storage structure and access method definition:** writing a set of definitions translated by the data storage and definition language compiler
- **Scheme and physical organization modification:** writing a set of definitions used by the DDL compiler to generate modifications to appropriate internal system tables (e.g. data dictionary). This is done rarely, but sometimes the database scheme or physical organization must be modified.

Hierarchical Data Model	Network Data Model	Relational Data Model
Relationship between records is of the parent child form.	Relationship between records is expressed in the form of pointers or links.	A relation that contains a key for each record involved in the relationship represents relationship between records.
Many to many relationships cannot be expressed in this data model.	Many to many relationship can be implemented in this model	Many to many relationship can be implemented in this model

It is simple, straightforward and natural method of implementing record relationships.	Record relationship implementation is very complex due to the use of pointers.	Relationship implementation is very easy through the use of a common key or composite key field(s).
Hierarchical model is useful only when there is some hierarchical characters in the database.	Network model is useful for representing such records, which have many to many relationships.	Relational model is useful for representing most of the real world objects and relationships among them.
In order to represent links between records pointers are used. Thus relations between records are physical.	In Network model also the record relations are physical.	Relational model does not maintain physical connection between records. Data is organized logically in the form of rows and columns.
Searching for a record is very difficult since one can retrieve a child only after going through its parent.	Searching a record is easy since there are multiple access paths to a data element.	A Unique, indexed key field is used to search for a data element.
During updating or deletion process, chances of data inconsistency is involved.	No problem of inconsistency exists in Network model because a data element is physically located at just one place.	Data integrity maintaining methods like Normalization process, etc. are adopted to maintain consistency.

- **Granting of authorization for dataccess:** granting different types of authorization for data access to various users
- **Integrity constraint specification:** generating integrity constraints. These are consulted by the database manager module whenever updates occur.

9. Data Independence?

The ability to modify a scheme definition in one level without affecting a scheme definition in a higher level is called **data independence**.

There are two kinds:

- **Physical data independence**
  - The ability to modify the physical scheme without causing application programs to be rewritten
  - Modifications at this level are usually to improve performance
- **Logical data independence**
  - The ability to modify the conceptual scheme without causing application programs to be rewritten
  - 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. An analogy is made to abstract data types in programming languages.

10. Compare the three traditional data models? Which is the best model and why as per you.

11. DBMS Architecture

The DBMS architecture describes how data in the database is viewed by the users. It is not concerned with how the data is handled and processed by the DBMS. The database users are provided with an abstract view of the data by hiding certain details of how data is physically stored. This enables the users to manipulate the data without worrying about where it is located

or how it is actually stored.

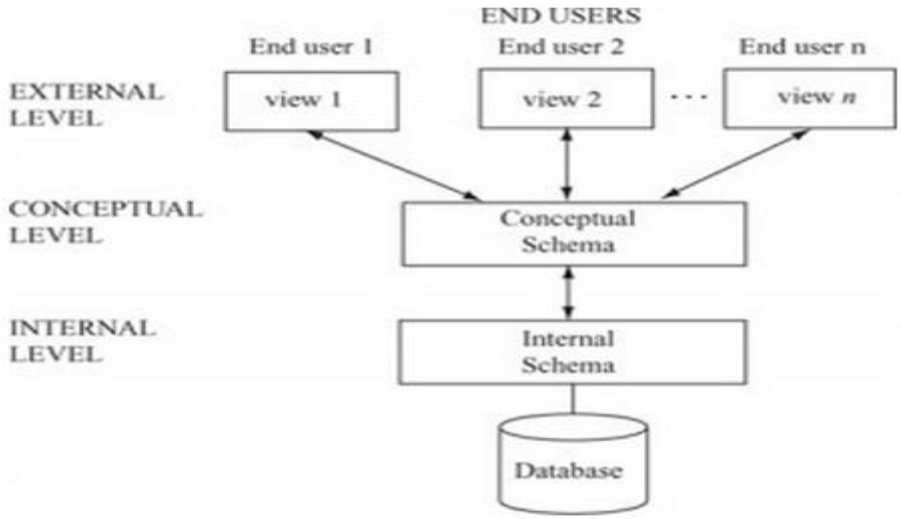
In this architecture, the overall database description can be defined at three levels, namely, internal, conceptual, and external levels and thus, named three-level DBMS architecture. This architecture is proposed by ANSI/SPARC (American National Standards Institute/Standards Planning and Requirements Committee) and hence, is also known as ANSI/SPARC Architecture. The three levels are discussed here.

Internal level: It is the lowest level of data abstraction that deals with the physical representation of the database on the computer and thus, is also known as physical level. It describes how the data is physically stored and organized on the storage medium. At this level, various aspects are considered to achieve optimal runtime performance and storage space utilization. These aspects include storage space allocation techniques for data and indexes, access paths such as indexes, data compression and encryption techniques, and record placement.

Conceptual level: This level of abstraction deals with the logical structure of the entire database and thus, is also known as logical level. It describes what data is stored in the database, the relationships among the data and complete view of the user's requirements without any concern for the physical implementation. That is, it hides the complexity of physical storage structures. The conceptual view is the overall view of the database and it includes all the information that is going to be represented in the database.

External level: It is the highest level of abstraction that deals with the user's view of the database and thus, is also known as view level. In general, most of the users and application programs do not require the entire data stored in the database. The external level describes a part of the database for a particular group of users. It permits users to access data in a way that is customized according to their needs, so that the same data can be seen by different users in different ways, at the same time. In this way, it provides a powerful and flexible security mechanism by hiding the parts of the database from certain users, as the user is not aware of existence of any attributes that are missing from the view.

These three levels are used to describe the schema of the database at various levels. Thus, the three-level architecture is also known as three-schema architecture. The internal level has an internal schema, which describes the physical storage structure of the database. The conceptual level has a conceptual schema, which describes the structure of entire database. The external level has external schemas or user views, which describe the part of the database according to a particular user's requirements, and hide the rest of the database from that user. The physical level is managed by the operating system under the direction of DBMS. The three-schema architecture is shown in Figure



Differences between a file-processing system and a DBMS

Some main differences between a database management system and a file-processing system are as per following.

File Processing System	DBMS
A database management system coordinates both the physical and the logical access to the data	File-processing system coordinates only the physical access.
A database management system reduces the amount of data duplication by ensuring that a physical piece of data is available to all programs authorized to have access to it	Data written by one programing a file-processing system may not be readable by another program.
A database management system is designed to allow flexible access to data (i.e., queries)	File-processing system is designed to allow predetermined access to data (i.e., compiled programs).
A database management system is designed to coordinate multiple users accessing the same data at the same time.	A file-processing system is usually designed to allow one or more programs to access different data files at the same time. In a file-processing system, a file can be accessed by two programs concurrently only if both programs have read-only access to the file.

12. Overall System Structure

1. Database systems are partitioned into modules for different functions. Some functions (e.g. file systems) may be provided by the operating system.

2. Components include:

File manager manages allocation of disk space and data structures used to represent information on disk.

Database manager: The interface between low-level data and application programs and queries.

Query processor translates statements in a query language into low-level instructions the database manager understands. (May also attempt to find an equivalent but more efficient form.)

DML precompiler converts DML statements embedded in an application program to normal procedure calls in a host language. The precompiler interacts with the query processor.

DDL compiler converts DDL statements to a set of tables containing metadata stored in a data dictionary.

In addition, several data structures are required for physical system implementation:

Data files: store the database itself.

Data dictionary: stores information about the structure of the database. It is used heavily. Great emphasis should be placed on developing a good design and efficient implementation of the dictionary.
Indices: provide fast access to data items holding particular values.
Figure 1.6 shows these components.

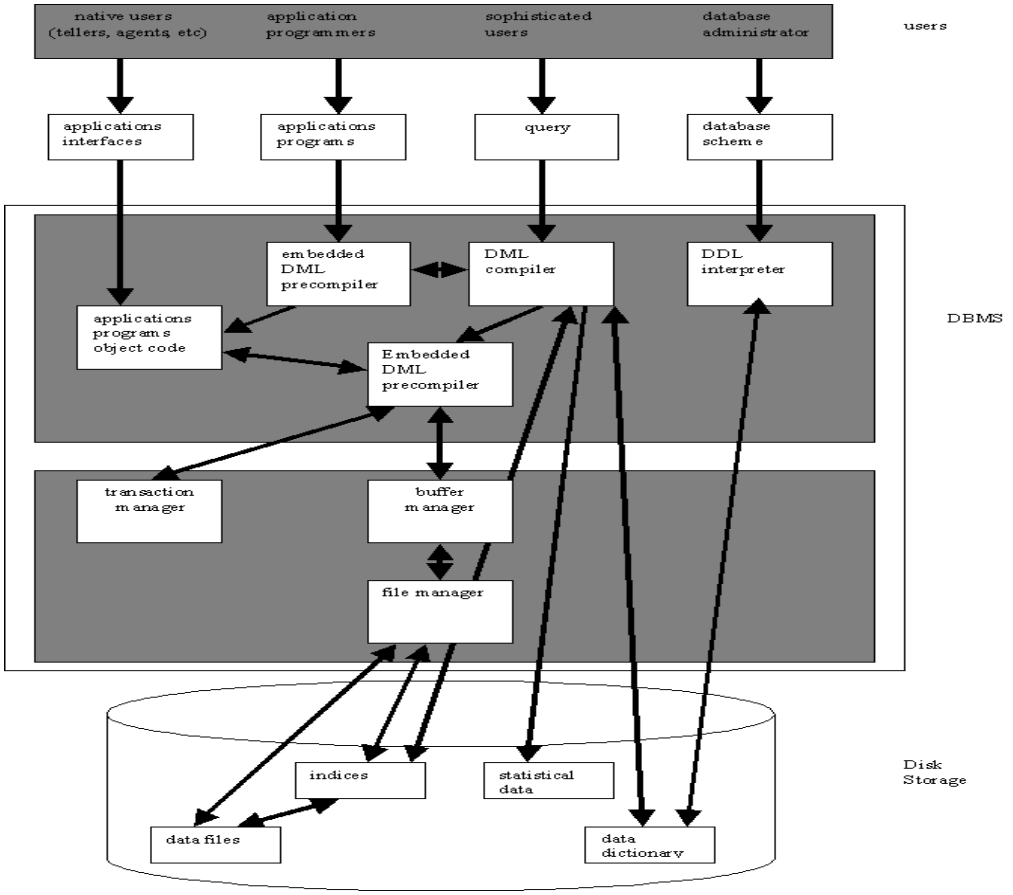


Figure 1.6: Database system structure.

Result: Thus we have studied the Introduction to Database Management system.

CONCLUSION:

COURSE OUTCOME	PROGRAM OUTCOME Attained
CO1	PO(1,2)

Practical No 02

Practical No: 2

Aim: To study ER model

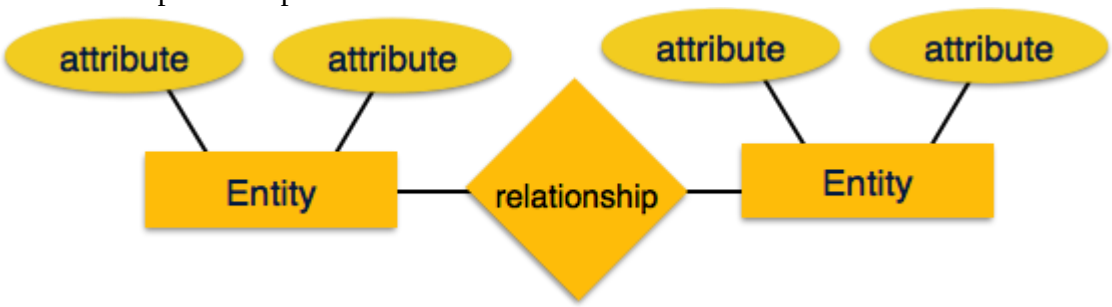
Data model tells how the logical structure of a database is modeled. Data Models are fundamental entities to introduce abstraction in DBMS. Data models define how data is connected to each other and how it will be processed and stored inside the system. The very first data model could be flat data-models where all the data used to be kept in same plane. Because earlier data models were not so scientific they were prone to introduce lots of duplication and update anomalies.

1. Entity-Relationship Model

Entity-Relationship model is based on the notion of real world entities and relationship among them. While formulating real-world scenario into database model, ER Model creates entity set, relationship set, general attributes and constraints.
ER Model is best used for the conceptual design of database.
ER Model is based on:

- **Entities** and their *attributes*
- **Relationships** among entities

These concepts are explained below.



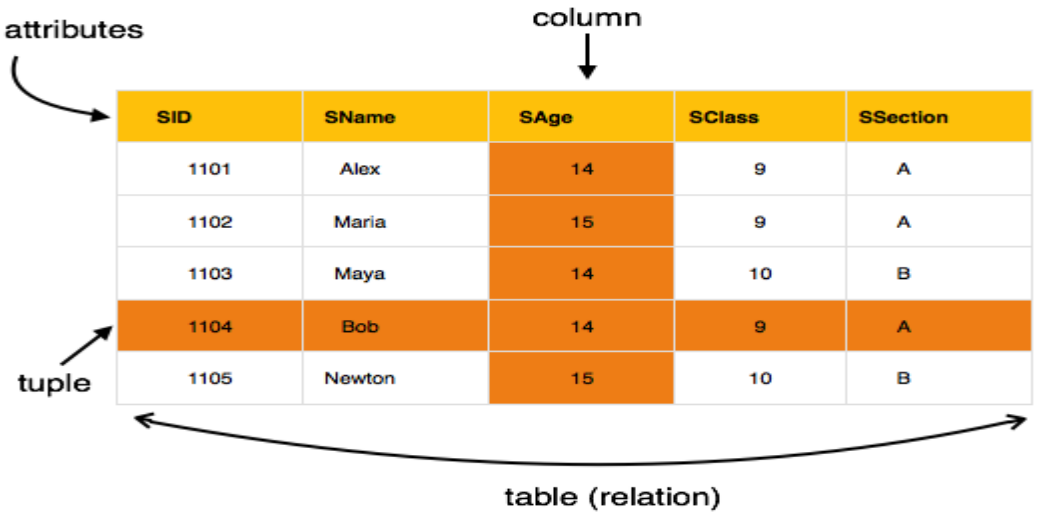
[*Image: ER Model*]

- **Entity**
  An entity in ER Model is real world entity, which has some properties called ***attributes***. Every attribute is defined by its set of values, called ***domain***.
  For example, in a school database, a student is considered as an entity. Student has various attributes like name, age and class etc.
- **Relationship**
  The logical association among entities is called ***relationship***. Relationships are mapped with entities in various ways. Mapping cardinalities define the number of association between two entities.
  Mapping cardinalities:
  - one to one
  - one to many
  - many to one
  - many to many

2. Relational Model

The most popular data model in DBMS is Relational Model. It is more scientific model then others. This model is based on first-order predicate logic and defines table as an n-ary relation. The main highlights of this model are:

- Data is stored in tables called relations.
- Relations can be normalized.
- In normalized relations, values saved are atomic values.
- Each row in relation contains unique value
- Each column in relation contains values from a same domain.



[*Image: Table in relational Model*]

3. DBMS - Data Schemas

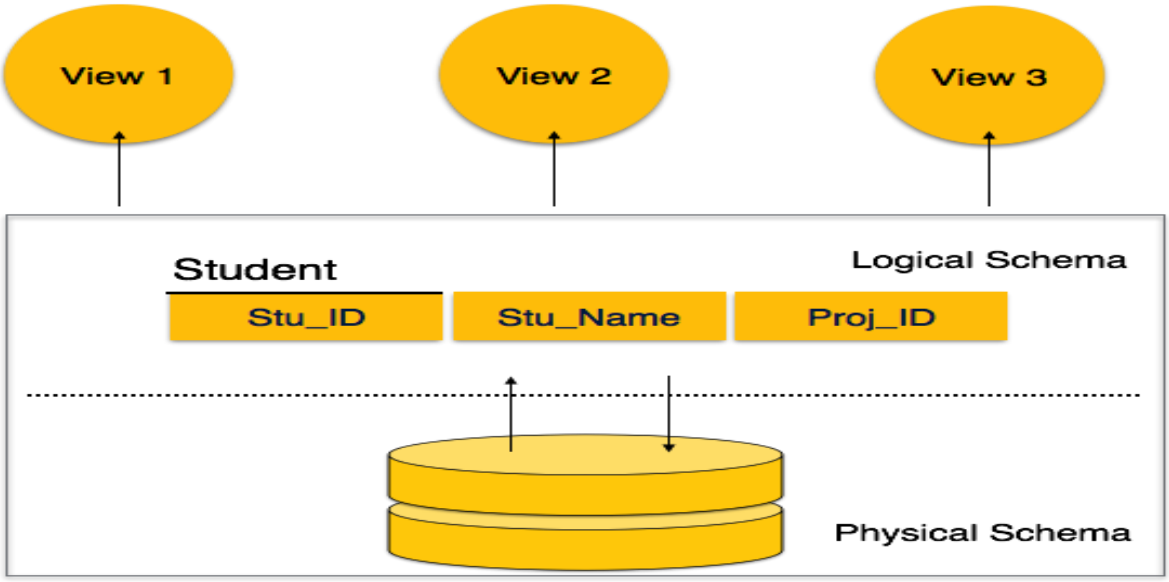
Database schema

Database schema skeleton structure of and it represents the logical view of entire database. It tells about how the data is organized and how relation among them is associated. It formulates all database constraints that would be put on data in relations, which resides in database. A database schema defines its entities and the relationship among them. Database schema is a descriptive detail of the database, which can be depicted by means of schema diagrams. All these activities are done by database designer to help programmers in order to give some ease of understanding all aspect of database.

Database schema can be divided broadly in two categories:

- **Physical Database Schema:** This schema pertains to the actual storage of data and its form of storage like files, indices etc. It defines the how data will be stored in secondary storage etc.
- **Logical Database Schema:** This defines all logical constraints that need to be applied on data stored. It defines tables, views and integrity constraints etc.

Database Instance- It is important that we distinguish these two terms individually. Database schema is the skeleton of database. It is designed when database doesn't exist at all and very hard to do any changes once the database is operational. Database schema does not contain any data or information.



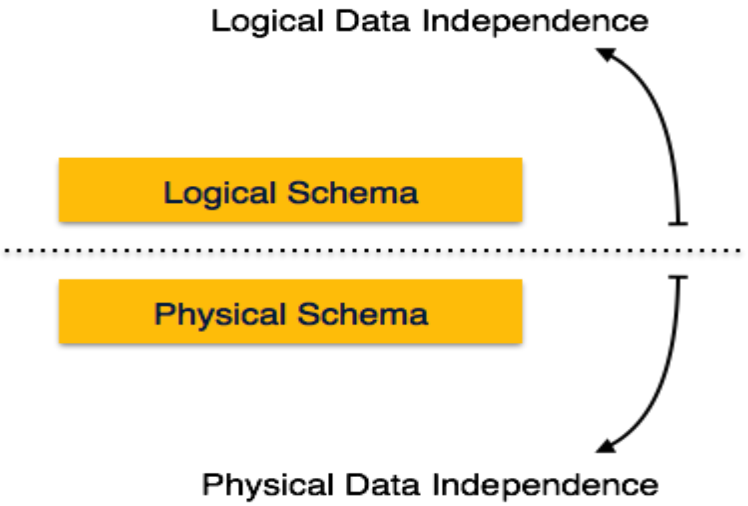
[*Image: Database Schemas*]

4. DBMS - Data Independence

If the database system is not multi-layered then it will be very hard to make any changes in the database system. Database systems are designed in multi-layers as we leant earlier.

Data Independence:

There's a lot of data in whole database management system other than user's data. DBMS comprises of three kinds of schemas, which is in turn data about data (Meta-Data). Meta-data is also stored along with database, which once stored is then hard to modify. But as DBMS expands, it needs to be changed over the time satisfy the requirements of users. But if the whole data were highly dependent it would become tedious and highly complex.



[*Image: Data independence*]

Data about data itself is divided in layered architecture so that when we change data at one layer it does not affect the data layered at different level. This data is independent but mapped on each other.

Logical Data Independence

Logical data is data about database, that is, it stores information about how data is managed inside. For example, a table (relation) stored in the database and all constraints, which are applied on that relation. Logical data independence is a kind of mechanism, which liberalizes itself from actual data stored on the disk. If we do some changes on table format it should not change the data residing on disk.

Physical Data Independence

All schemas are logical and actual data is stored in bit format on the disk. Physical data independence is the power to change the physical data without impacting the schema or logical data. For example, in case we want to change or upgrade the storage system itself, that is, using SSD instead of Hard-disks should not have any impact on logical data or schemas.

5. ER Model Basic Concepts

Entity relationship model defines the conceptual view of database. It works around real world entity and association among them. At view level, ER model is considered well for designing databases. A real-world thing either animate or inanimate that can be easily identifiable and distinguishable. For example, in a school database, student, teachers, class and course offered can be considered as entities. All entities have some attributes or properties that give them their identity.An entity set is a collection of similar types of entities. Entity set may contain entities with attribute sharing similar values. For example, Students set may contain all the student of a school; likewise Teachers set may contain all the teachers of school from all faculties. Entities sets need not to be disjoint. Attributes Entities are represented by means of their properties, called attributes. All attributes have values. For example, a student entity may have name, class, age as attributes.

There exist a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.

TYPES OF ATTRIBUTES:

- **Simple attribute:**
  Simple attributes are atomic values, which cannot be divided further. For example, student's phone-number is an atomic value of 10 digits.
- **Composite attribute:**
  Composite attributes are made of more than one simple attribute. For example, a student's complete name may have first_name and last_name.
- **Derived attribute:**
  Derived attributes are attributes, which do not exist physical in the database, but there values are derived from other attributes presented in the database. For example, average_salary in a department should be saved in database instead it can be derived. For another example, age can be derived from data_of_birth.
- **Single-valued attribute:**
  Single valued attributes contain on single value. For example: Social_Security_Number.

- **Multi-value attribute:**
  Multi-value attribute may contain more than one values. For example, a person can have more than one phone numbers, email_addresses etc.
  These attribute types can come together in a way like:
- simple single-valued attributes
- simple multi-valued attributes
- composite single-valued attributes
- composite multi-valued attributes.

ENTITY-SET AND KEYS

Key is an attribute or collection of attributes that uniquely identifies an entity among entity set.
For example, roll_number of a student makes her/him identifiable among students.

- **Super Key:** Set of attributes (one or more) that collectively identifies an entity in an entity set.
- **Candidate Key:** Minimal super key is called candidate key that is, supers keys for which no proper subset are a superkey. An entity set may have more than one candidate key.
- **Primary Key:** This is one of the candidate key chosen by the database designer to uniquely identify the entity set.

RELATIONSHIP

The association among entities is called relationship. For example, employee entity has relation works_at with department. Another example is for student who enrolls in some course. Here, Works_at and Enrolls are called relationship.

RELATIONSHIP SET:

Relationship of similar type is called relationship set. Like entities, a relationship too can have attributes. These attributes are called descriptive attributes.

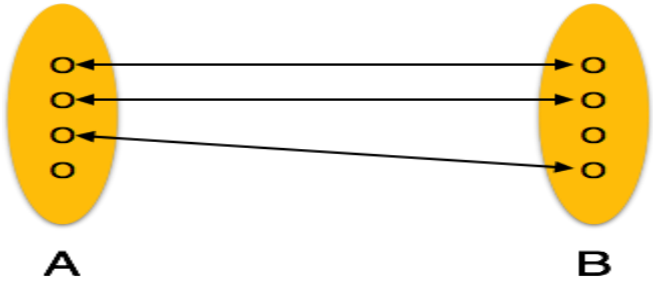
DEGREE OF RELATIONSHIP

The number of participating entities in an relationship defines the degree of the relationship.

- Binary = degree 2
- Ternary = degree 3
- n-ary = degree

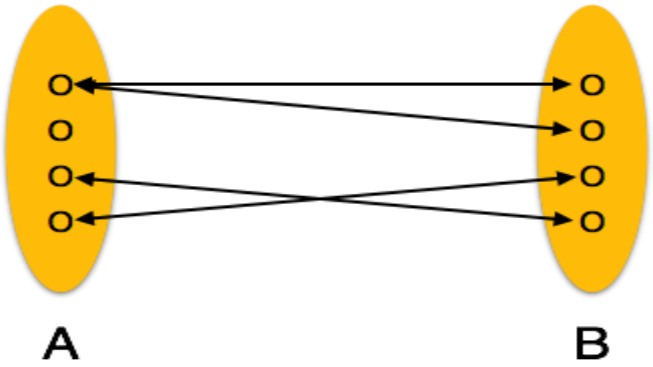
MAPPING CARDINALITIES:

Cardinality defines the number of entities in one entity set which can be associated to the number of entities of other set via relationship set.
One-to-one: one entity from entity set A can be associated with at most one entity of entity set B and vice versa.

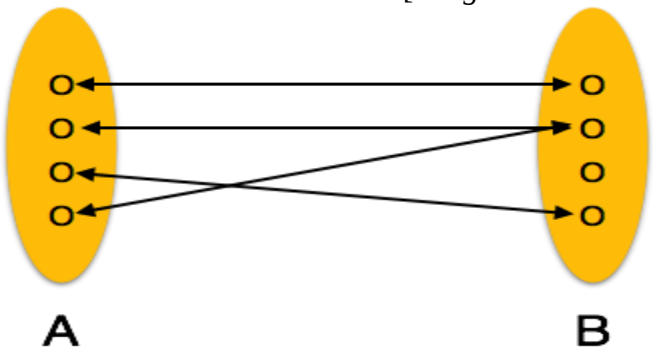


[*Image: One-to-one relation*]

- **One-to-many:** One entity from entity set A can be associated with more than one entities of entity set B but from entity set B one entity can be associated with at most one entity.



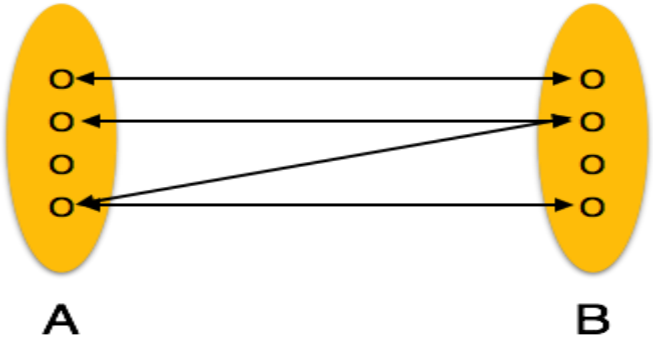
[*Image: One-to-many relation*]



- **Many-to-one:** More than one entities from entity set A can be associated with at most one entity of entity set B but one entity from entity set B can be associated with more than one entity from entity set A.

[*Image: Many-to-one relation*]

- **Many-to-many:** one entity from A can be associated with more than one entity from B and vice versa.



[*Image: Many-to-many relation*]

6. ER Diagram Representation

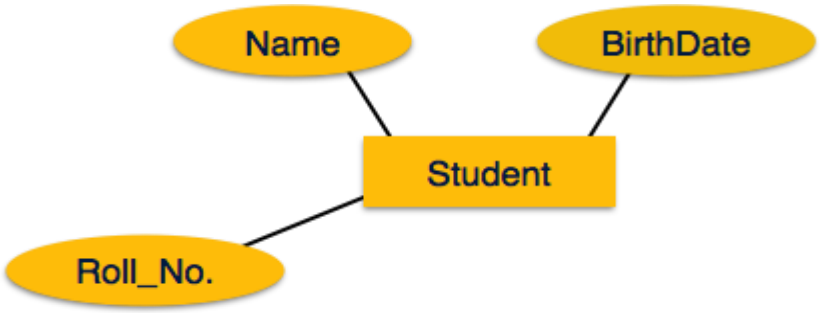
Now we shall learn how ER Model is represented by means of ER diagram. Every object like entity, attributes of an entity, relationship set, and attributes of relationship set can be represented by tools of ER diagram.Entities are represented by means of rectangles. Rectangles are named with the entity set they represent.

[*Image: Entities in a school database*]

Attributes

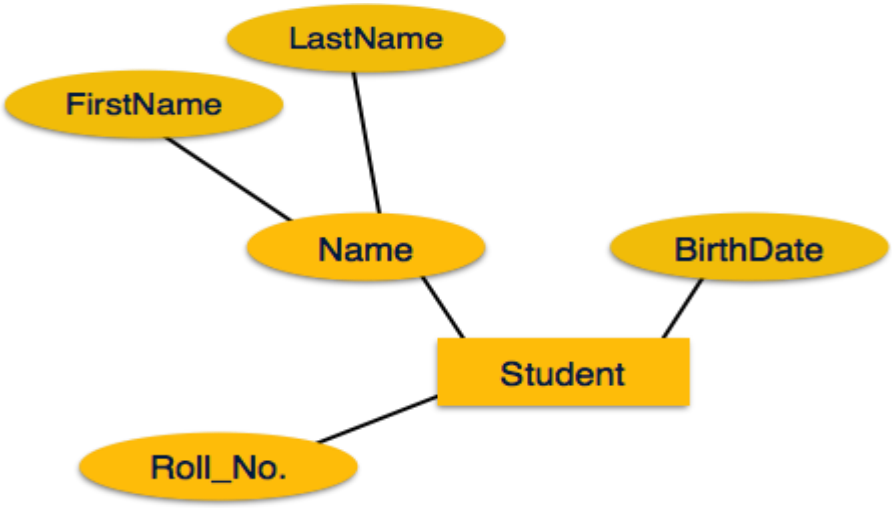
Attributes are properties of entities. Attributes are represented by means of eclipses. Every

eclipse represents one attribute and is directly connected to its entity (rectangle).



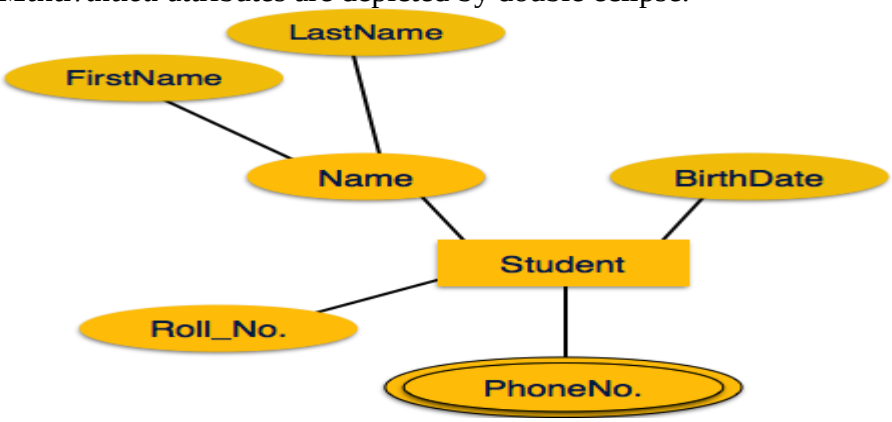
[*Image: Simple Attributes*]

If the attributes are ***composite***, they are further divided in a tree like structure. Every node is then connected to its attribute. That is composite attributes are represented by eclipses that are connected with an eclipse.



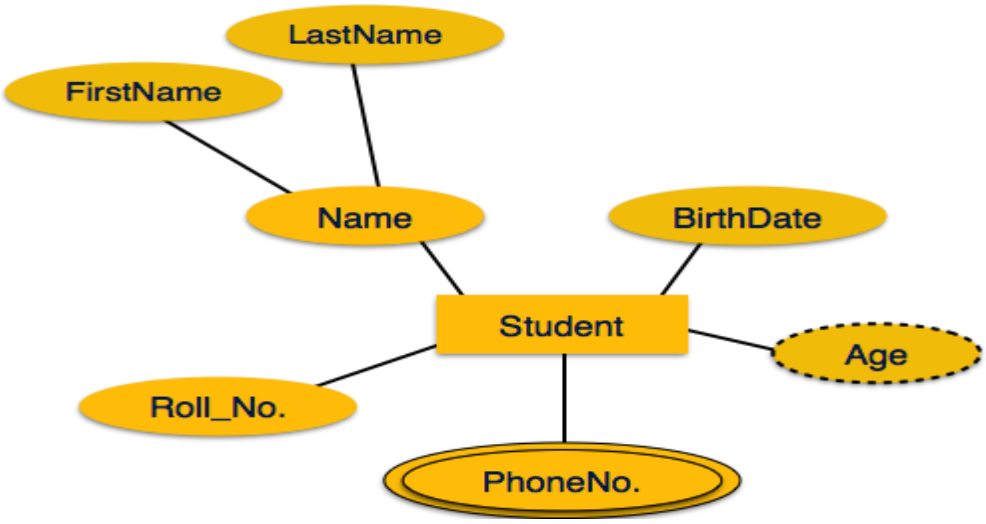
[*Image: Composite Attributes*]

Multivalued attributes are depicted by double eclipse.



[*Image: Multivalued Attributes*]

Derived attributes are depicted by dashed eclipse.



[*Image: Derived Attributes*]

7. Relationship

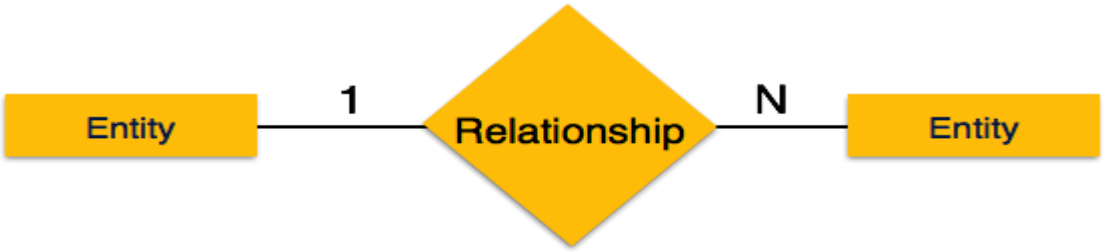
Relationships are represented by diamond shaped box. Name of the relationship is written in the diamond-box. All entities (rectangles), participating in relationship, are connected to it by a line.

- **One-to-one**
  When only one instance of entity is associated with the relationship, it is marked as '1'. This image below reflects that only 1 instance of each entity should be associated with the relationship. It depicts one-to-one relationship

[*Image: One-to-one*]

- **One-to-many**

  When more than one instance of entity is associated with the relationship, it is marked as 'N'. This image below reflects that only 1 instance of entity on the left and more than one instance of entity on the right can be associated with the relationship. It depicts one-to-many relationship



[*Image: One-to-many*]

- **Many-to-one**

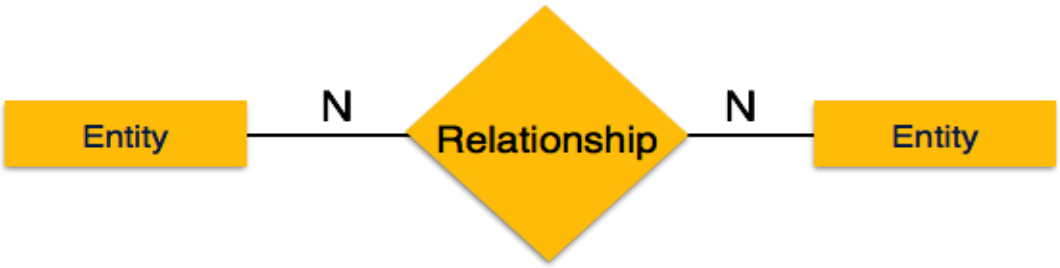
  When more than one instance of entity is associated with the relationship, it is marked as 'N'.

This image below reflects that more than one instance of entity on the left and only one instance of entity on the right can be associated with the relationship. It depicts many-to-one relationship

[*Image: Many-to-one*]

- **Many-to-many**

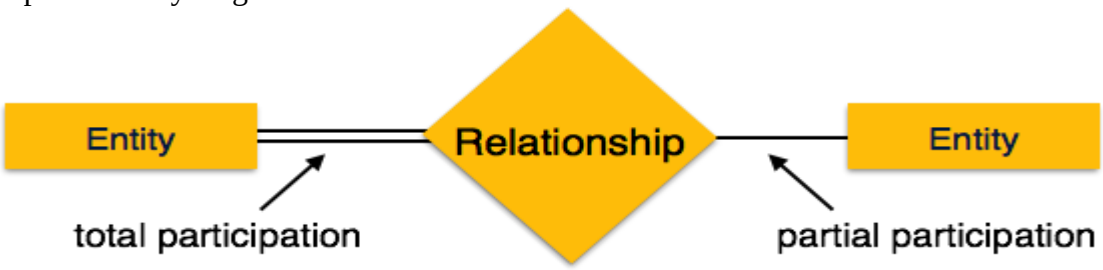
  This image below reflects that more than one instance of entity on the left and more than one instance of entity on the right can be associated with the relationship. It depicts many-to-many relationship



[*Image: Many-to-many*]

8. PARTICIPATION CONSTRAINTS

- **Total Participation:** Each entity in the entity is involved in the relationship. Total participation is represented by double lines.
- **Partial participation:** Not all entities are involved in the relation ship. Partial participation is represented by single line.



[*Image: Participation Constraints*]

Generalization, Aggregation

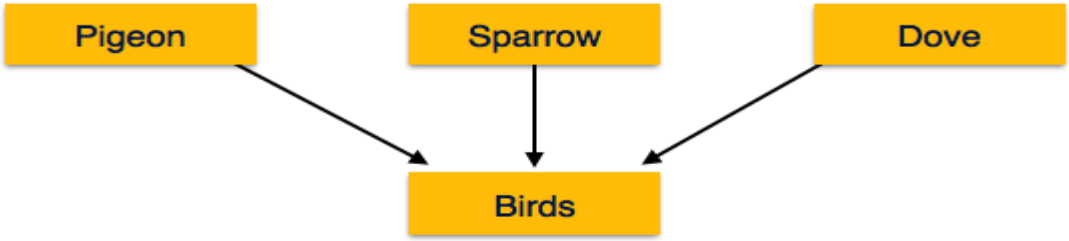
ER Model has the power of expressing database entities in conceptual hierarchical manner such that, as the hierarchical goes up it generalize the view of entities and as we go deep in the hierarchy it gives us detail of every entity included.

Going up in this structure is called generalization, where entities are clubbed together to represent a more generalized view. For example, a particular student named, Mira can be generalized along with all the students, the entity shall be student, and further a student is

person. The reverse is called specialization where a person is student, and that student is Mira.

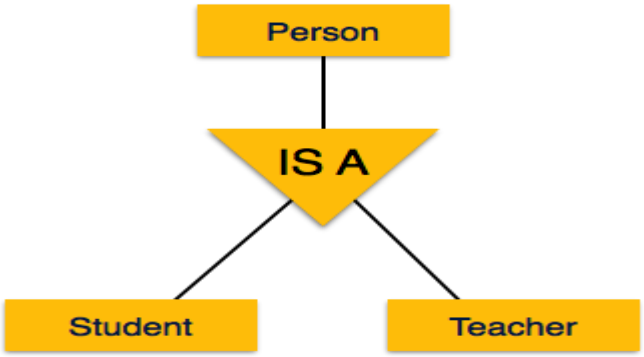
9. Generalization

As mentioned above, the process of generalizing entities, where the generalized entities contain the properties of all the generalized entities is called Generalization. In generalization, a number of entities are brought together into one generalized entity based on their similar characteristics. For an example, pigeon, house sparrow, crow and dove all can be generalized as Birds.



[*Image: Generalization*]

10. **Specialization-** Specialization is a process, which is opposite to generalization, as mentioned above. In specialization, a group of entities is divided into sub-groups based on their characteristics. Take a group Person for example. A person has name, date of birth, gender etc. These properties are common in all persons, human beings. But in a company, a person can be identified as employee, employer, customer or vendor based on what role do they play in company.



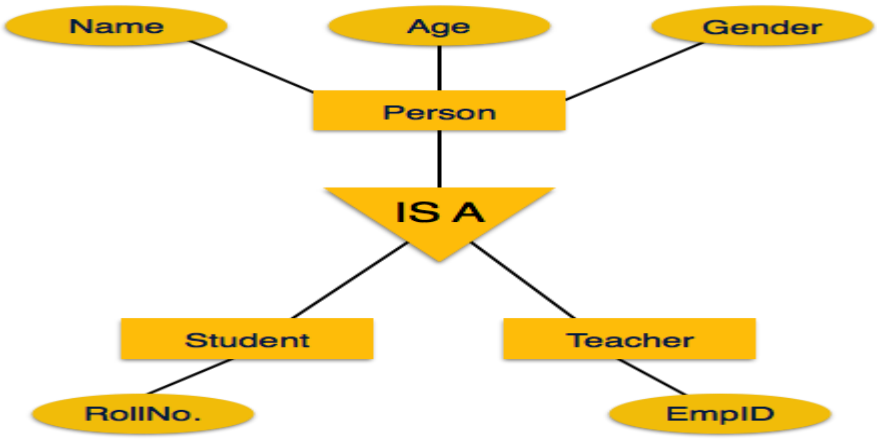
[*Image: Specialization*]

Similarly, in a school database, a person can be specialized as teacher, student or staff; based on what role do they play in school as entities.

11. Inheritance

We use all above features of ER-Model, in order to create classes of objects in object oriented programming. This makes it easier for the programmer to concentrate on what she is programming. Details of entities are generally hidden from the user, this process known as abstraction.

One of the important features of Generalization and Specialization, is inheritance, that is, the attributes of higher-level entities are inherited by the lower level entities.



[*Image: Inheritance*]

For example, attributes of a person like name, age, and gender can be inherited by lower level entities like student and teacher etc.

12. DBMS Codd's Rules

Dr Edgar F. Codd did some extensive research in Relational Model of database systems and came up with twelve rules of his own which according to him, a database must obey in order to be a true relational database.

These rules can be applied on a database system that is capable of managing is stored data using only its relational capabilities. This is a foundation rule, which provides a base to imply other rules on it.

Rule 1: Information rule-This rule states that all information (data), which is stored in the database, must be a value of some table cell. Everything in a database must be stored in table formats. This information can be user data or meta-data.

Rule 2: Guaranteed Access rule-This rule states that every single data element (value) is guaranteed to be accessible logically with 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

This rule states the NULL values in the database must be given a systematic treatment. As a NULL may have several meanings, i.e. NULL can be interpreted as one the following: data is missing, data is not known, data is not applicable etc.

Rule 4: Active online catalog

This rule states that the structure description of whole database must be stored in an online catalog, i.e. data dictionary, which can be accessed by the 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

This rule states that a database must have a support for a language which has linear syntax

which is capable of data definition, data manipulation and transaction management operations. Database can be accessed by means of this language only, either directly or by means of some application. If the database can be accessed or manipulated in some way without any help of this language, it is then a violation.

Rule 6: View updating rule

This rule states that all views of database, which can theoretically be updated, must also be updatable by the system.

Rule 7: High-level insert, update and delete rule

This rule states the database must employ 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

This rule states that the application should not have any concern about how the data is physically stored. Also, any change in its physical structure must not have any impact on application.

Rule 9: Logical data independence

This rule states that the logical data must be independent of its user's view (application). Any change in logical data must not imply any change in the application using it. For example, if two tables are merged or one is split into two different tables, there should be no impact the change on user application. This is one of the most difficult rule to apply.

Rule 10: Integrity independence

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

Rule 11: Distribution independence

This rule states that the end user must not be able to see that the data is distributed over various locations. User must also see that data is located at one site only. This rule has been proven as a foundation of distributed database systems.

Rule 12: Non-subversion rule

This rule states that if a system has an interface that provides access to low level records, this interface then must not be able to subvert the system and bypass security and integrity constraints.

13. Relational Data Model

Relational data model is the primary data model, which is used widely around the world for data storage and processing. This model is simple and have all the properties and capabilities

required to process data with storage efficiency.

Tables: In relation data model, relations are saved in the format of Tables. This format stores the relation among entities. A table has rows and columns, where rows represent records and columns represents the attributes.
Tuple: A single row of a table, which contains a single record for that relation is called a tuple.
Relation instance: A finite set of tuples in the relational database system represents relation instance. Relation instances do not have duplicate tuples.
Relation schema: This describes the relation name (table name), attributes and their names.
Relation key: Each row has one or more attributes which can identify the row in the relation (table) uniquely, is called the relation key.
Attribute domain: Every attribute has some pre-defined value scope, known as attribute domain.

Result: Thus we have studied the ER model.

CONCLUSION:

COURSE OUTCOME	**PROGRAM OUTCOME Attained**
CO1	**PO(1,2)**

Practical No 03

Practical No.3

AIM: TO Perform DDL an DML Commands in SQL

SQL Commands:

SQL commands are instructions, coded into SQL statements, which are used to communicate with the database to perform specific tasks, work, functions and queries with data. SQL commands can be used not only for searching the database but also to perform various other functions like, for example, you can create tables, add data to tables, or modify data, drop the table, set permissions for users. SQL commands are grouped into four major categories depending on their functionality:

- **Data Definition Language (DDL)** - These SQL commands are used for creating, modifying, and dropping the structure of database objects. The commands are **CREATE, ALTER, DROP, RENAME, and TRUNCATE.**
- **Data Manipulation Language (DML)** - These SQL commands are used for storing, retrieving, modifying, and deleting data.
  These Data Manipulation Language commands are:SELECT, INSERT, UPDATE, and DELETE.

DDL Command:

SQL CREATE TABLE Statement

The CREATE TABLE Statement is used to create tables to store data. Integrity Constraints like primary key, unique key, foreign key can be defined for the columns while creating the table. The integrity constraints can be defined at column level or table level. The implementation and the syntax of the CREATE Statements differs for different RDBMS.

The Syntax for the CREATE TABLE Statement is:

CREATE TABLE table_name
(column_name1 datatype, column_name2 datatype, ... column_nameN datatype)

DATA TYPES

ORACLE	SQL Server	DB2
int	int	Int
char	char	Char
varchar2	-	-
varchar	varchar	Varchar
number	numeric	Numeric
float	float	Float
decimal	decimal	Decimal
Date	smalldatetime	Date
timestamp	datetime	Timestamp

create table employee55

(

Eno number (10) primary key,

Fname varchar(10),

Lname varchar(10),

dno number(10),

jdate date,

Foreign key (dno) references department(Dno)

);

create table department

(

dno number (10) primary key,

dname varchar(10)

);

SQL ALTER TABLE Statement

The SQL ALTER TABLE command is used to modify the definition (structure) of a table by modifying the definition of its columns. The ALTER command is used to perform the following functions.

1) Add, drop, modify table columns
2) Add and drop constraints
3) Enable and Disable constraints

Syntax to add a column

```
ALTER TABLE table_name ADD column_name datatype;
```

For Example: To add a column "experience" to the employee table, the query would be like

```
ALTER TABLE employee ADD experience number(3);
```

Syntax to drop a column

```
ALTER TABLE table_name DROP column_name;
```

For Example: To drop the column "location" from the employee table, the query would be like

```
ALTER TABLE employee DROP column  location;
```

Syntax to modify a column

```
ALTER TABLE table_name MODIFY column_name datatype;
```

For Example: To modify the column salary in the employee table, the query would be like

```
ALTER TABLE employee MODIFY salary number(15,2);
```

SQL RENAME Command

The SQL RENAME command is used to change the name of the table or a database object.

If you change the object's name any reference to the old name will be affected. You have to manually change the old name to the new name in every reference.

Syntax to rename a table

```
RENAME old_table_name To new_table_name;
```

For Example: To change the name of the table employee to my_employee, the query would be like

```
RENAME employee TO my_emloyee;
```

SQL TRUNCATE Statement

The SQL TRUNCATE command is used to delete all the rows from the table and free the space containing the table.

Syntax to TRUNCATE a table:

```
TRUNCATE TABLE table_name;
```

For Example: To delete all the rows from employee table, the query would be like,

```
TRUNCATE TABLE employee;
```

Difference between DELETE and TRUNCATE Statements:
DELETE Statement: This command deletes only the rows from the table based on the condition given in the where clause or deletes all the rows from the table if no condition is specified. But it does not free the space containing the table.
TRUNCATE statement: This command is used to delete all the rows from the table and free the space containing the table.

SQL DROP Statement:

The SQL DROP command is used to remove an object from the database. If you drop a table, all the rows in the table is deleted and the table structure is removed from the database. Once a table is dropped we cannot get it back, so be careful while using DROP command. When a table is dropped all the references to the table will not be valid.

Syntax to drop a sql table structure:

```
DROP TABLE table_name;
```

For Example: To drop the table employee, the query would be like

```
DROP TABLE employee;
```

Difference between DROP and TRUNCATE Statement:

If a table is dropped, all the relationships with other tables will no longer be valid, the integrity constraints will be dropped, grant or access privileges on the table will also be dropped, if want use the table again it has to be recreated with the integrity constraints, access privileges and the relationships with other tables should be established again. But, if a table is truncated, the table structure remains the same, therefore any of the above problems will not exist.

SQL INSERT Statement

```
The INSERT Statement is used to add new rows of data to a table.
```

```
Inserting the data directly to a table.Syntax for SQL INSERT is:
```

```
INSERT INTO TABLE_NAME
```

```
[ (col1, col2, col3,...colN)]
```

```
VALUES (value1, value2, value3,...valueN);
```

- `col1, col2,...colN -- the names of the columns in the table into which you want to insert data.`

While inserting a row, if you are adding value for all the columns of the table you need not specify the column(s) name in the sql query. But you need to make sure the order of the values is in the same order as the columns in the table. The sql insert query will be as follows

```
INSERT INTO TABLE_NAME
VALUES (value1, value2, value3,...valueN);
```

For Example: If you want to insert a row to the employee table, the query would be like,

```
INSERT INTO employee (id, name, dept, age, salary location) VALUES (105, 'Srinath', 'Aeronautics', 27, 33000);
```

If you are inserting data to all the columns, the column names can be omitted. The above insert statement can also be written as,

```
INSERT INTO employee
VALUES (105, 'Srinath', 'Aeronautics', 27, 33000);
```

Inserting data to a table through a select statement.

SQL UPDATE Statement

The UPDATE Statement is used to modify the existing rows in a table.

To insert multiple values at a same time:

```
insert all
into dept values(50,'aaa','abc')
```

```
into dept values(60,'aaa','abc')
```

```
select * from dual;
```

The Syntax for SQL UPDATE Command is:

```
UPDATE table_name
SET column_name1 = value1,
column_name2 = value2, ...
[WHERE condition]
```

- table_name - the table name which has to be updated.
- column_name1, column_name2.. - the columns that gets changed.
- value1, value2... - are the new values.

NOTE: In the Update statement, WHERE clause identifies the rows that get affected. If you do not include the WHERE clause, column values for all the rows get affected.

For Example: To update the location of an employee, the sql update query would be like,

```
UPDATE employee
SET location ='Mysore'
WHERE id = 101;
```

To change the salaries of all the employees, the query would be,

```
UPDATE employee
SET salary = salary + (salary * 0.2);
```

SQL Delete Statement

The DELETE Statement is used to delete rows from a table.

The Syntax of a SQL DELETE statement is:

```
DELETE FROM table_name [WHERE condition];
```

table_name -- the table name which has to be updated.

NOTE: The WHERE clause in the sql delete command is optional and it identifies the rows in the column that gets deleted. If you do not include the WHERE clause all the rows in the table is deleted, so be careful while writing a DELETE query without WHERE clause.

For Example: To delete an employee with id 100 from the employee table, the sql delete query would be like,

```
DELETE FROM employee WHERE id = 100;
```

To delete all the rows from the employee table, the query would be like,

```
DELETE FROM employee;
```

SQL Select statement

The SQL **SELECT** statement is used to fetch the data from a database table which returns this data in the form of a result table. These result tables are called result-sets.

Syntax

The basic syntax of the SELECT statement is as follows −

SELECT column1, column2, columnN FROM table_name;

Here, column1, column2... are the fields of a table whose values you want to fetch. If you want to fetch all the fields available in the field, then you can use the following syntax.

For Example:

1.To select all rows from the table

SQL> select * from emp;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20

7499 ALLEN SALESMAN 7698 20-FEB-81 1600 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 2450 10

7788 SCOTT ANALYST 7566 19-APR-87 3000 20

7839 KING PRESIDENT 17-NOV-81 5000 10

7844 TURNER SALESMAN 7698 08-SEP-81 1500 0 30

7876 ADAMS CLERK 7788 23-MAY-87 1100 20

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

---------- ---------- --------- ---------- --------- ---------- ---------- ----------

7900 JAMES CLERK 7698 03-DEC-81 950 30

7902 FORD ANALYST 7566 03-DEC-81 3000 20

7934 MILLER CLERK 7782 23-JAN-82 1300 10

14 rows selected.

2.To select particular rows from the table

SQL> select ename,job from emp;

ENAME JOB

---------- ---------

SMITH CLERK

ALLEN SALESMAN

WARD SALESMAN

JONES MANAGER

MARTIN SALESMAN

BLAKE MANAGER

CLARK MANAGER

SCOTT ANALYST

KING PRESIDENT

TURNER SALESMAN

ADAMS CLERK

3. SQL> select ename,job from emp where ename='SMITH';

ENAME JOB

---------- ---------

SMITH CLERK

Result: We have executed the DDL & DML command.

CONCLUSION:

COURSE OUTCOME	**PROGRAM OUTCOME Attained**
CO2	PO(1,2,3,8,9,12)

Practical No.4

Practical No.4

Aim: TO Perform aggregate function in SQL.
The SQL Aggregate Functions are functions that provide mathematical operations. If you need to add, count or perform basic statistics, these functions will be of great help.
The functions include:

- **count()** - counts a number of rows
- **sum()** - compute sum
- **avg()** - compute average
- **min()** - compute minimum
- **max()** - compute maximum

Why Use SQL Aggregate Functions?
The SQL Aggregate Functions are useful when mathematical operations must be performed on all or a grouping of values.
SQL > SQL Functions > Average

SQL uses the AVG( ) function to calculate the average of a column. The syntax for using this function is,

```
SELECT AVG("column_name")
FROM "table_name";
```

For example, if we want to get the average of all sales from the following table,

Table Store_Information

Store_Name	Sales	Txn_Date
Los Angeles	1500	Jan-05-1999
San Diego	250	Jan-07-1999
Los Angeles	300	Jan-08-1999
Boston	700	Jan-08-1999

we would type in,

```
SELECT AVG(Sales) FROM Store_Information;
```

Result:

AVG(Sales)
687.5

SQL > SQL Functions > Count

Another arithmetic function is COUNT. This allows us to COUNT up the number of row in a certain table. The syntax is,

```
SELECT COUNT("column_name")
FROM "table_name";
```

For example, if we want to find the number of store entries in our table,

Table Store_Information

Store_Name	Sales	Txn_Date
Los Angeles	1500	Jan-05-1999
San Diego	250	Jan-07-1999
Los Angeles	300	Jan-08-1999
Boston	700	Jan-08-1999

we'd key in,

```
SELECT COUNT (Store_Name)
FROM Store_Information;
```

Result:

```
 COUNT (Store_Name)
              4
```

SQL > SQL Functions > MAX Function

SQL uses the MAX function to find the maximum value in a column. The syntax for using theMAX function is,

```
SELECT MAX ("column_name")
FROM "table_name";
```

For example, if we want to get the highest sales from the following table,

Table Store_Information

Store_Name	Sales	Txn_Date
Los Angeles	1500	Jan-05-1999
San Diego	250	Jan-07-1999
Los Angeles	300	Jan-08-1999
Boston	700	Jan-08-1999

we would type in

```
SELECT MAX (Sales) FROM Store_Information;
```

Result:

```
 MAX (Sales)
      1500
```

SQL > SQL Functions > MIN Function

SQL uses the MIN function to find the maximum value in a column. The syntax for using theMIN function is,

```
SELECT MIN ("column_name")
FROM "table_name";
```

For example, if we want to get the lowest sales from the following table,

Table Store_Information

Store_Name	Sales	Txn_Date
Los Angeles	1500	Jan-05-1999
San Diego	250	Jan-07-1999
Los Angeles	300	Jan-08-1999
Boston	700	Jan-08-1999

we would type in

```
SELECT MIN(Sales) FROM Store_Information;
```

Result:

```
MIN(Sales)
    250
```

SQL > SQL Functions > SUM Function

The SUM function is used to calculate the total for a column. The syntax is,

```
SELECT SUM("column_name")
FROM "table_name";
```

For example, if we want to get the sum of all sales from the following table,
Table Store_Information we would type in, SELECT SUM(Sales) FROM Store_Information

Store_Name	Sales	Txn_Date
Los Angeles	1500	Jan-05-1999
San Diego	250	Jan-07-1999
Los Angeles	300	Jan-08-1999
Boston	700	Jan-08-1999

Result-Thus we have executed aggregate functions in SQL

CONCLUSION:

COURSE OUTCOME	**PROGRAM OUTCOME Attained**
CO2	PO(1,2,3,8,9,12)

Practical No. 05

Practical No. 5

Aim: To Perform Join and set Operation in SQL

Theory:

Join in SQL

SQL Join is used to fetch data from two or more tables, which is joined to appear as single set of data. SQL Join is used for combining column from two or more tables by using values common to both tables. Join Keyword is used in SQL queries for joining two or more tables. Minimum required condition for joining table, is
(n-1) w here n, is number of tables. A table can also join to itself know n as, Se lf Join.

Type s of Join

The following are the types of JOIN that w e can use in SQL.

Inner
Outer
Left
Right

Cross JOIN or Cartesian Product

This type of JOIN returns the cartesian product of row s of from the tables in Join. It will return a table which consists of records which combines each row from the first table with each row of the second table.

Cross JOIN Syntax is,

```
SELECT column-name-list
from table-name1
CROSS JOIN
table-name2;
```

INNER Join or EQUI Join

This is a simple J OI N in w hich the result is based on matched data as per the equality condition specified in the query

Inner Join Syntax is,

```
SELECT column-name-list
from table-name1
INNER JOIN
table-name2
WHERE table-name1.column-name = table-name2.column-name;
```

Natural JOIN

Natural Join is a type of Inner join which is based on column having same name and same datatype present in both the tables to be joined.

N atural J oin Syntax is,

```
SELECT *
from table-name1
NATURAL JOIN
table-name2;
```

Outer JOIN

Outer Join is based on both matched and unmatched data. Outer J oins subdivide further into,
Left Outer Join
Right Outer Join
Full Outer Join

Le ft Oute r Join

The left outer join returns a result table w ith the matche d data of tw o tables then remaining row s of the left table and null for the r ight table' s column.

Left Outer J oin syntax is,

```
S E L E C T  c o l u m n - n a m e - l i s t
f r o m  t a b l e - n a m e 1
L E F T  O U T E R  J O I N
t a b l e - n a m e 2
o n  t a b l e - n a m e 1 . c o l u m n - n a m e  =  t a b l e - n a m e 2 . c o l u m n - n a m e ;
```

Right Oute r Join

The right outer join returns a result table w ith the matche d data of tw o tables then remaining row s of the r ight table and null for the le ft table' s columns

```
s e l e c t  c o l u m n - n a m e - l i s t
f r o m  t a b l e - n a m e 1
R I G H T  O U T E R  J O I N
t a b l e - n a m e 2
o n  t a b l e - n a m e 1 . c o l u m n - n a m e  =  t a b l e - n a m e 2 . c o l u m n - n a m e ;
```

```
select ename,dname,dept.deptno from emp,dept where emp.deptno(+)=dept.deptno;
```

```
SQL> select ename,dname from emp right outer join dept on emp.deptno=dept.deptno;

ENAME      DNAME
---------- --------------
SMITH      RESEARCH
ALLEN      SALES
WARD       SALES
JONES      RESEARCH
MARTIN     SALES
BLAKE      SALES
CLARK      ACCOUNTING
SCOTT      RESEARCH
KING       ACCOUNTING
TURNER     SALES
ADAMS      RESEARCH

ENAME      DNAME
---------- --------------
JAMES      SALES
FORD       RESEARCH
MILLER     ACCOUNTING
```

aaa
aaa
aaa
OPERATIONS
aaa
bbb

20 rows selected.

Full Oute r Join

The full outer join returns a result table w ith the matche d data of tw o table then remaining row s of both le ft
table and then the r ight table.
Full Outer J oin Syntax is

```
s e l e c t  c o l u m n - n a m e - l i s t
f r o m  t a b l e - n a m e 1
F U L L  O U T E R  J O I N
t a b l e - n a m e 2
o n  t a b l e - n a m e 1 . c o l u m n - n a m e  =  t a b l e - n a m e 2 . c o l u m n - n a m e ;
```

Perform set operations in SQL

```
SQL> create table t1
  2  (
  3  eno number(10) primary key,
  4  ename varchar2(10)
  5  );

Table created.

SQL> create table t2
  2  (
  3  eno number(10) primary key,
  4  ename varchar2(10)
  5  );

Table created.

SQL> insert into t1 values(1,'A');

1 row created.

SQL> insert into t1 values(2,'B');

1 row created.

SQL>  insert into t1 values(3,'C');

1 row created.
```

```
SQL>  insert into t1 values(4,'D');

1 row created.

SQL> insert into t2 values(1,'A');

1 row created.

SQL> insert into t2 values(2,'B');

1 row created.

SQL> insert into t2 values(5,'E');

1 row created.

SQL> insert into t2 values(6,'F');

1 row created.

SQL> select ename from t1
  2  union select ename from t2;

ENAME
----------
A
B
C
D
E
F

6 rows selected.

SQL>
SQL> select ename from t1
  2    union all
  3  select ename from t2;

ENAME
----------
A
B
C
D
A
B
E
F
```

8 rows selected.

```
SQL> select ename from t1
  2  intersect
  3  select ename from t2;

ENAME
----------
A
B

SQL> select ename from t1
  2  intersect
  3  select ename from t2;

ENAME
----------
A  B
SQL> select ename from t1
  2  minus
  3  select ename from t2;

ENAME
----------
C
D

SQL>  select ename from t1;

ENAME
----------
A
B
C
D

SQL>  select ename from t2;

ENAME
----------
A
B
E
F

SQL> select ename from t2
  2  minus
  3  select ename from t1;
```

ENAME

E
F

Result: We have executed join and set operations in SQL.

CONCLUSION:

COURSE OUTCOME	**PROGRAM OUTCOME Attained**
CO2	PO(1,2,3,8,9,12)

Practical No. 06

Practical No. 6

Aim: To Perform Various Arithmetical, Relational, & Logical Operators in SQL
Theory:
Arithmetical Operators in SQL

Operator	Description	Example
+	Addition - Adds values on either side of the operator	a + b will give 30
-	Subtraction - Subtracts right hand operand from left hand operand	a - b will give -10
*	Multiplication - Multiplies values on either side of the operator	a * b will give 200
/	Division - Divides left hand operand by right hand operand	b / a will give 2

SQL Relational, Operators:

Assume variable a holds 10 and variable b holds 20, then:Show Examples

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

SQL Logical Operators:

Here is a list of all the logical operators available in SQL.

Operator	Description
ALL	The ALL operator is used to compare a value to all values in another value set.
AND	The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause.
ANY	The ANY operator is used to compare a value to any applicable value in the list according to the condition.
BETWEEN	The BETWEEN operator is used to search for values that are within a set of values, given the minimum value and the maximum value.
EXISTS	The EXISTS operator is used to search for the presence of a row in a specified table that meets certain criteria.
IN	The IN operator is used to compare a value to a list of literal values that have been specified.
LIKE	The LIKE operator is used to compare a value to similar values using wildcard operators.
NOT	The NOT operator reverses the meaning of the logical operator with which it is used. Eg: NOT EXISTS, NOT BETWEEN, NOT IN, etc. **This is a negate operator.**
OR	The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause.
IS NULL	The NULL operator is used to compare a value with a NULL value.
UNIQUE	The UNIQUE operator searches every row of a specified table for uniqueness (no duplicates).

SQL Arithmetic Operator

SQL> create table KDK27 (name char (10), status char(15), gender char(5), salery number (12),);
Table created.
SQL>
SQL> insert into KDK26 values('abc',101,'lux',25,10);
1 row created.
SQL> insert into KDK26 values('pqr',201,'surf',85,8);
1 row created.
SQL> insert into KDK26 values('xyz',301,'tea',75,10);
1 row created.
Insert into KDK26 values('ghj',411,'bottle',85,111);
SQL> select * from KDK26;

COMPONY	ITEM_CODE	ITEM_NAME	PRIZE	QUANTITY
abc	101	lux	25	10
pqr	201	surf	85	8

xyz 301 tea 75 10

SQL> Select item_code,item_name,prize+10 from KDK26;

ITEM_CODE	ITEM_NAME	PRIZE+10
101	lux	35
201	surf	95
301	tea	85

SQL> Select item_code,item_name,prize*10 from KDK26;

ITEM_CODE	ITEM_NAME	PRIZE*10
101	lux	250
201	surf	850
301	tea	750

SQL> Select item_code,item_name,prize-10 from KDK26;

ITEM_CODE	ITEM_NAME	PRIZE-10
101	lux	15
201	surf	75
301	tea	65

SQL> Select item_code,item_name,prize/2 from KDK26;

ITEM_CODE	ITEM_NAME	PRIZE/2
101	lux	12.5
201	surf	42.5
301	tea	37.5

SQL Comparison Operator

SQL> select * from KDK26 where compony ='abc';

COMPONY	ITEM_CODE	ITEM_NAME	PRIZE	QUANTITY
abc	101	lux	25	10

SQL> select item_code,item_name from KDK26 where prize>=80;

ITEM_CODE	ITEM_NAME
201	surf

SQL> select item_code,item_name from KDK26 where prize<80;

ITEM_CODE	ITEM_NAME
101	lux
301	tea

SQL> select item_code,item_name from KDK26 where prize>=80;

ITEM_CODE	ITEM_NAME
201	surf

SQL Logical Operator

SQL> select ename,sal from emp where sal <> 1300 and ename= 'SMITH';

ENAME SAL
---------- ----------
SMITH 1400
SQL> select ename,sal from emp where sal <> 1300;

ENAME SAL
---------- ----------
SMITH 1400
ALLEN 1600
WARD 1250
JONES 2975
MARTIN 1250
BLAKE 2850
CLARK 2450
SCOTT 3000
KING 5000
TURNER 1500
ADAMS 1100

ENAME SAL
---------- ----------
JAMES 950
FORD 3000

13 rows selected.
SQL> select ename,sal from emp where sal <> 1300 or ename!= 'SMITH';

ENAME SAL
---------- ----------
SMITH 1400
ALLEN 1600
WARD 1250
JONES 2975
MARTIN 1250
BLAKE 2850
CLARK 2450
SCOTT 3000
KING 5000
TURNER 1500
ADAMS 1100

ENAME SAL
---------- ----------
JAMES 950
FORD 3000
MILLER 1300

14 rows selected.

SQL> select ename,sal from emp where not sal=1300;

ENAME SAL
---------- ----------
SMITH 1400
ALLEN 1600
WARD 1250
JONES 2975
MARTIN 1250
BLAKE 2850
CLARK 2450
SCOTT 3000
KING 5000
TURNER 1500
ADAMS 1100

ENAME SAL
---------- ----------
JAMES 950
FORD 3000

13 rows selected.
SQL> select * from KDK26 where item_code like 201;
COMPONY ITEM_CODE ITEM_NAME PRIZE QUANTITY
---------- ---------- --------------- ---------- ----------
pqr 201 surf 85 8
SQL> select ename from emp where ename like '%S%';

ENAME

SMITH
JONES
SCOTT
ADAMS
JAMES
SQL> select ename from emp where ename like 'S%';

ENAME

SMITH
SCOTT
SQL> select * from KDK26 where item_name not in ('lux');
COMPONY ITEM_CODE ITEM_NAME PRIZE QUANTITY
---------- ---------- --------------- ---------- ----------
pqr 201 surf 85 8
xyz 301 tea 75 10
SQL> select compony,item_code,item_name from KDK26 where prize>50 and quantity<12;
COMPONY ITEM_CODE ITEM_NAME
---------- ---------- ---------------
pqr 201 surf

xyz 301 tea

SQL> select item_code,item_name from KDK26 where prize>50 or quantity<80;

```
 ITEM_CODE ITEM_NAME
---------- --------------
       101 lux
       201 surf
       301 tea
```

SQL> select * from KDK26 where compony ='abc';

```
COMPONY    ITEM_CODE ITEM_NAME          PRIZE   QUANTITY
---------- ---------- --------------- ---------- ----------
abc             101 lux                   25         10
```

SQL> select item_code,item_name from KDK26 where prize>80;

```
 ITEM_CODE ITEM_NAME
---------- --------------
       201 surf
```

SQL> select item_code,item_name from KDK26 where prize<80;

```
 ITEM_CODE ITEM_NAME
---------- --------------
       101 lux
       301 tea
```

SQL> select ename,sal from emp where ename= 'SMITH' or ename='ALLEN';

```
ENAME           SAL
---------- ----------
SMITH          1400
ALLEN          1600
```

SQL> select ename,sal from emp where sal >1400 and sal < 1600;

```
ENAME           SAL
---------- ----------
TURNER         1500
```

SQL> select ename,sal,sal+100 from emp;

```
ENAME           SAL    SAL+100
---------- ---------- ----------
SMITH          1000       1100
ALLEN          1700       1800
WARD           1350       1450
JONES          3075       3175
MARTIN         1350       1450
BLAKE          2950       3050
CLARK          2550       2650
SCOTT          3100       3200
KING           5100       5200
TURNER         1600       1700
ADAMS          1200       1300

ENAME           SAL    SAL+100
---------- ---------- ----------
JAMES          1050       1150
```

12 rows selected.

SQL> select ename,sal from emp where sal <> 1300 and ename= 'SMITH';!=

ENAME SAL
---------- ----------
SMITH 1000
SQL> select ename,sal from emp where sal <> 1300 or ename!= 'SMITH';

ENAME SAL
---------- ----------
SMITH 1000
ALLEN 1700
WARD 1350
JONES 3075
MARTIN 1350
BLAKE 2950
CLARK 2550
SCOTT 3100
KING 5100
TURNER 1600
ADAMS 1200

ENAME SAL
---------- ----------
JAMES 1050

12 rows selected.

SQL> select ename,sal from emp where sal <> 1300 or ename!= 'SMITH';

ENAME SAL
---------- ----------
SMITH 1000
ALLEN 1700
WARD 1350
JONES 3075
MARTIN 1350
BLAKE 2950
CLARK 2550
SCOTT 3100
KING 5100
TURNER 1600
ADAMS 1200

ENAME SAL
---------- ----------
JAMES 1050

12 rows selected.

SQL> select ename,sal from emp where not sal=1300;

ENAME SAL
---------- ----------
SMITH 1000
ALLEN 1700
WARD 1350
JONES 3075
MARTIN 1350
BLAKE 2950
CLARK 2550
SCOTT 3100
KING 5100
TURNER 1600
ADAMS 1200
ENAME SAL
---------- ----------
JAMES 1050

12 rows selected.

SQL> select ename from emp;

ENAME

SMITH
ALLEN
WARD
JONES
MARTIN
BLAKE
CLARK
SCOTT
KING
TURNER
ADAMS

ENAME

JAMES

12 rows selected.

SQL> select ename from emp where ename like'S%';

ENAME

SMITH

SCOTT

```
SQL> select ename from emp where ename like'%S';

ENAME
----------
JONES
ADAMS
JAMES

SQL> select ename from emp where ename like '___G';
ENAME
----------
KING

SQL> select * from dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        70 aaa            zzz
        80 bbb             aab
        90 90%
        50 aaa            abc
        60 aaa            abc
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

9 rows selected.

SQL> select dname from dept where dname like '%90%%';

DNAME
--------------
90%
SQL> select dname from dept where dname like'%#%%' escape '#';

DNAME
--------------
80%
90%
```

Result: We have executed the Various Arithmetical, Relational, & Logical Operators in SQL.

CONCLUSION:

COURSE OUTCOME	**PROGRAM OUTCOME Attained**
CO2	PO(1,2,3,8,9,12)

Practical No. 07

Practical No. 7

Aim: To perform various Clauses in SQL

SQL Comparison Keywords

There are other comparison keywords available in sql which are used to enhance the search capabilities of a sql query. They are "IN", "BETWEEN...AND", "IS NULL", "LIKE".

Comparision Operators	Description
LIKE	column value is similar to specified character(s).
IN	column value is equal to any one of a specified set of values.
BETWEEN...AND	column value is between two values, including the end values specified in the range.
IS NULL	column value does not exist.

SQL BETWEEN ... AND Operator

The operator BETWEEN and AND, are used to compare data for a range of values.

For Example: to find the names of the students between age 10 to 15 years, the query would be like,

```
SELECT first_name, last_name, age
FROM student_details
WHERE age BETWEEN 10 AND 15;
```

The output would be similar to:

first_name	last_name	age
Rahul	Sharma	10
Anajali	Bhagwat	12
Shekar	Gowda	15

SQL IN Operator:

The IN operator is used when you want to compare a column with more than one value. It is similar to an OR condition.

For example: If you want to find the names of students who are studying either Maths or Science, the query would be like,

```
SELECT first_name, last_name, subject
FROM student_details
WHERE subject IN ('Maths', 'Science')
```

The output would be similar to:

first_name	last_name	subject
Anajali	Bhagwat	Maths

Shekar	Gowda	Maths
Rahul	Sharma	Science
Stephen	Fleming	Science

You can include more subjects in the list like ('maths','science','history')
NOTE: The data used to compare is case sensitive.

SQL IS NULL Operator

A column value is NULL if it does not exist. The IS NULL operator is used to display all the rows for columns that do not have a value.

For Example: If you want to find the names of students who do not participate in any games, the query would be as given below

```
SELECT first_name, last_name
FROM student_details
WHERE games IS NULL
```

There would be no output as we have every student participate in a game in the table student_details, else the names of the students who do not participate in any games would be displayed

SQL ORDER BY

The ORDER BY clause is used in a SELECT statement to sort results either in ascending or descending order. Oracle sorts query results in ascending order by default.

Syntax for using SQL ORDER BY clause to sort data is:

```
SELECT column-list
FROM table_name [WHERE condition]
[ORDER BY column1 [, column2, .. columnN] [DESC]];
SQL> select * from dept;

    DEPTNO DNAME         LOC
---------- -------------- -------------
        70 aaa           zzz
        80 bbb            aab
        90 90%
        50 aaa           abc
        60 aaa           abc
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
9 rows selected.


SQL> select dname from dept where deptno between 20 and 70;

DNAME
--------------
aaa
```

```
aaa
aaa
RESEARCH
SALES
OPERATIONS

6 rows selected.

SQL> select deptno,dname from dept where deptno between 20 and 70;

    DEPTNO DNAME
---------- --------------
        70 aaa
        50 aaa
        60 aaa
        20 RESEARCH
        30 SALES
        40 OPERATIONS

6 rows selected.

SQL> select deptno,dname from dept where deptno in (70,80);

    DEPTNO DNAME
---------- --------------
        70 aaa
        80 bbb

SQL> select * from dept order by deptno;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
        50 aaa            abc
        60 aaa            abc
        70 aaa            zzz
        80 bbb             aab
        90 90%

9 rows selected.

SQL>  select * from dept order by deptno asc;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
```

```
        30 SALES         CHICAGO
        40 OPERATIONS    BOSTON
        50 aaa           abc
        60 aaa           abc
        70 aaa           zzz
        80 bbb           aab
        90 90%

9 rows selected.

SQL>  select * from dept order by deptno desc;

   DEPTNO DNAME         LOC
---------- -------------- -------------
        90 90%
        80 bbb           aab
        70 aaa           zzz
        60 aaa           abc
        50 aaa           abc
        40 OPERATIONS    BOSTON
        30 SALES         CHICAGO
        20 RESEARCH      DALLAS
        10 ACCOUNTING    NEW YORK

9 rows selected.

SQL>  select * from dept order by 1;

   DEPTNO DNAME         LOC
---------- -------------- -------------
        10 ACCOUNTING    NEW YORK
        20 RESEARCH      DALLAS
        30 SALES         CHICAGO
        40 OPERATIONS    BOSTON
        50 aaa           abc
        60 aaa           abc
        70 aaa           zzz
        80 bbb           aab
        90 90%

9 rows selected.
SQL>  select * from dept order by 2;

   DEPTNO DNAME         LOC
---------- -------------- -------------
        90 90%
        10 ACCOUNTING    NEW YORK
        40 OPERATIONS    BOSTON
        20 RESEARCH      DALLAS
        30 SALES         CHICAGO
```

```
50 aaa          abc
70 aaa          zzz
60 aaa          abc
80 bbb          aab

9 rows selected.

SQL>  select * from dept order by 3;

   DEPTNO DNAME         LOC
---------- -------------- -------------
       40 OPERATIONS     BOSTON
       30 SALES          CHICAGO
       20 RESEARCH       DALLAS
       10 ACCOUNTING     NEW YORK
       80 bbb          aab
       50 aaa          abc
       60 aaa          abc
       70 aaa          zzz
       90 90%

9 rows selected.
```

Result: We have executed various Clause in SQL.

CONCLUSION:

COURSE OUTCOME	**PROGRAM OUTCOME Attained**
CO2	PO(1,2,3,8,9,12)

Practical No. 08

Practical No. 8

Aim: To implement integrity constraints in SQL.

Theory:

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

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

Following are some of the most commonly used constraints available in SQL.

- UNIQUE Constraint − Ensures that all values in a column are different.
- PRIMARY Key − Uniquely identifies each row/record in a database table.
- FOREIGN Key − Uniquely identifies a row/record in any of the given database table.
- CHECK Constraint − The CHECK constraint ensures that all the values in a column satisfies certain conditions.
- NOT NULL constraint

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

There are many types of integrity constraints that play a role in **Referential Integrity (RI)**. These constraints include Primary Key, Foreign Key, Unique Constraints and other constraints which are mentioned above.

Code:

Primary key Constraint:

```
CREATE TABLE emp11
( id number(5),
name char(20),
dept char(10),
age number(2),
salary number(10),
location char(10),
CONSTRAINT emp_id_pk PRIMARY KEY (id)
);
CREATE TABLE department
```

```
( Dno number(10),
dname char(20),
CONSTRAINT dno_id_pk PRIMARY KEY (Dno)
);
```

Foreign key constraints

```
create table employee66
(
Eno number (10) primary key,
Fname varchar(10),
Lname varchar(10),
dno number(10) constraint dno_fk references department(Dno),
jdate date
);
```

Unique,Not null constraint

```
SQL> CREATE TABLE emp22
  ( id number(5) CONSTRAINT id_pk PRIMARY KEY,
 name char(20) constraint nm_null not null,
 dept char(10) constraint nm_unq unique,
 age number(2),
 salary number(10),
 location char(10)
  );

Table created.

SQL> desc emp22;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER(5)
 NAME                                      NOT NULL CHAR(20)
 DEPT                                               CHAR(10)
 AGE                                                NUMBER(2)
 SALARY                                             NUMBER(10)
 LOCATION                                           CHAR(10)

SQL> insert into emp22 values(2,'','aa','','');
insert into emp22 values(2,'','aa','','')
                         *
ERROR at line 1:
ORA-01400: cannot insert NULL into ("SYSTEM"."EMP22"."NAME")


SQL> insert into emp22 values(2,'aa','aa','','');
insert into emp22 values(2,'aa','aa','','')
```

```
SQL> insert into emp22 values(3,'aa','aa',',',');

*
ERROR at line 1:
ORA-00001: unique constraint (SYSTEM.NM_UNQ) violated
```

Check Constraint-

```
SQL> CREATE TABLE emp33
   ( id number(5) CONSTRAINT eid_pk PRIMARY KEY,
   name char(20) constraint enm_null not null,
  dept char(10) constraint enm_unq unique,
   salary number(10) check (salary>100)
   );

Table created.
SQL> insert into emp33 values(1,'aa','bb',10);
insert into emp33 values(1,'aa','bb',10)
*
ERROR at line 1:
ORA-02290: check constraint (SYSTEM.SYS_C005484) violated

SQL> insert into emp33 values(1,'aa','bb',1000);

1 row created.
```

Result- Thus we have successfully created constraints in sql.

CONCLUSION:

COURSE OUTCOME	**PROGRAM OUTCOME Attained**
CO2	PO(1,2,3,8,9,12)

Practical No. 09

Practical No. 09

Aim: To create View and Materialized view in SQL.

Theory: In SQL, a view is a virtual table based on the result-set of an SQL statement.
A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.
You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.
CREATE VIEW Syntax
CREATE VIEW view_name AS
SELECT column1,column2 .
FROM table_name
WHERE condition;

```
SQL> create view department_view
    as
    select dname
   from department;
```
View created.
```
SQL> select * from department_view;
```
DNAME
a
b
QL Updating a View
You can update a view by using the following syntax:
SQL CREATE OR REPLACE VIEW Syntax
CREATE OR REPLACE VIEW view_name AS
SELECT column1,column2,
FROM table_name
WHERE condition;
```
SQL> create or replace view department_view
    as
    select dname
   from department
   where dno=1;
```
View created.
```
SQL> select * from department_view;
```

DNAME
---
a
SQL Dropping a View
You can delete a view with the DROP VIEW command.
SQL DROP VIEW Syntax
DROP VIEW view_name;
```
SQL> drop view department_view;
```
View dropped.

Materialized View-

```
CREATE MATERIALIZED VIEW dept_mv
BUILD IMMEDIATE
 REFRESH FORCE
ON COMMIT
AS
SELECT dno,dname FROM department

insert into department values (1,'A');
insert into department values (2,'B');
select * from dept_mv
```

Result: Thus we have created View and Materialized view in sql.

CONCLUSION:

COURSE OUTCOME	**PROGRAM OUTCOME Attained**
CO2	PO(1,2,3,8,9,12)

Practical No. 10

Practical No. 10

Aim: To Create a User and Grant Permissions in Oracle.

Theory:

Use the CREATE USER statement to create and configure a database user, which is an account through which you can log in to the database, and to establish the means by which Oracle Database permits access by the user.
User-Specify the name of the user to be created.

IDENTIFIED Clause-The IDENTIFIED clause lets you indicate how Oracle Database authenticates the user.
BY password-The BY password clause lets you creates a local user and indicates that the user must specify password to log on to the database. Passwords are case sensitive.

DEFAULT TABLESPACE Clause-Specify the default tablespace for objects that the user creates. If you omit this clause, then the user's objects are stored in the database default tablespace. If no default tablespace has been specified for the database, then the user's objects are stored in the SYSTEM tablespace.

TEMPORARY TABLESPACE Clause-Specify the tablespace or tablespace group for the user's temporary segments. If you omit this clause, then the user's temporary segments are stored in the database default temporary tablespace or, if none has been specified, in the SYSTEM tablespace.

Specify tablespace to indicate the user's temporary tablespace.
Specify tablespace_group_name to indicate that the user can save temporary segments in any tablespace in the tablespace group specified by tablespace_group_name.

QUOTA Clause-Use the QUOTA clause to specify the maximum amount of space the user can allocate in the tablespace.
A CREATE USER statement can have multiple QUOTA clauses for multiple tablespaces.

UNLIMITED lets the user allocate space in the tablespace without bound.

Use the GRANT statement to give privileges to a specific user or role, or to all users, to perform actions on database objects. You can also use the GRANT statement to grant a role to a user, to PUBLIC, or to another role.

The following types of privileges can be granted:
- Delete data from a specific table.
- Insert data into a specific table.
- Create a foreign key reference to the named table or to a subset of columns from a table.
- Select data from a table, view, or a subset of columns in a table.
- Create a trigger on a table.
- Update data in a table or in a subset of columns in a table.
- Run a specified function or procedure.
- Use a sequence generator or a user-defined type.

Sql> create user test2 identified by test2

```
DEFAULT TABLESPACE example
 QUOTA 10M ON example
   TEMPORARY TABLESPACE temp
   QUOTA 5M ON system;
Sql> User Created
SQL> grant create session to ABC;
```

Result: We have created user in sql.

CONCLUSION:

COURSE OUTCOME	**PROGRAM OUTCOME Attained**
CO2	PO(1,2,3,8,9,12)

Practical No. 11

Practical No. 11

Aim: To create functions using PL/SQL.

Theory:

A standalone function is created using the **CREATE FUNCTION** statement. The simplified syntax for the **CREATE OR REPLACE PROCEDURE** statement is as follows −

```
CREATE [OR REPLACE] FUNCTION function_name
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
RETURN return_datatype
{IS | AS}
BEGIN
  < function_body >
END [function_name];
```

Where,

- *function-name* specifies the name of the function.
- [OR REPLACE] option allows the modification of an existing function.
- The optional parameter list contains name, mode and types of the parameters. IN represents the value that will be passed from outside and OUT represents the parameter that will be used to return a value outside of the procedure.
- The function must contain a **return** statement.
- The *RETURN* clause specifies the data type you are going to return from the function.
- *function-body* contains the executable part.
- The AS keyword is used instead of the IS keyword for creating a standalone function.

```
CREATE OR REPLACE FUNCTION totalEmp

RETURN number IS

  total number(2) := 0;

BEGIN

  SELECT count(*) into total

  FROM emp;



  RETURN total;

END;

/

select totalEmp from dual;
```

Result: We have executed functions using Pl/Sql

CONCLUSION:

COURSE OUTCOME	**PROGRAM OUTCOME Attained**
CO3	PO(1,2,3,4,5,8,9,10,12)

Practical No. 12

Practical No. 12

Aim: To create procedures using Pl/Sql

Theory:

A PL/SQL procedure is a named block that does a specific task. PL/SQL procedure allows you to encapsulate complex business logic and reuse it in both database layer and application layer.

A procedure is created with the **CREATE OR REPLACE PROCEDURE**statement. The simplified syntax for the CREATE OR REPLACE PROCEDURE statement is as follows −

```
CREATE [OR REPLACE] PROCEDURE procedure_name
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
{IS | AS}
BEGIN
  < procedure_body >
END procedure_name;
```

Where,

- *procedure-name* specifies the name of the procedure.
- [OR REPLACE] option allows the modification of an existing procedure.
- The optional parameter list contains name, mode and types of the parameters. **IN** represents the value that will be passed from outside and OUT represents the parameter that will be used to return a value outside of the procedure.
- *procedure-body* contains the executable part.
- The AS keyword is used instead of the IS keyword for creating a standalone procedure.

Code:

```
SQL> create or replace procedure emp_proc8(empid in number)
  2  As
  3  Begin
  4   update emp set sal= sal+100
  5  where empno=empid;
  6  end;
  7  /
```

Procedure created.

```
SQL> execute emp_proc8(7369);
```

PL/SQL procedure successfully completed.

```
SQL> select * from emp where empno='7369';
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	1200		20

```
SQL> execute emp_proc8(7369);
```

PL/SQL procedure successfully completed.

```
SQL> select * from emp where empno='7369';

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80       1300                    20

                    BINARY_FLOAT | BINARY_DOUBLE ] ]
SQL> create or replace procedure emp_proc11(empid in number, S out number)
  2  As
  3  Begin
  4   update emp set sal= sal+100
  5  where empno=empid;
  6  commit;
  7  select sal into S from emp where empno=empid;
  8  end;
  9  /

Procedure created.
SQL> variable k number;
```

SQL> execute emp_proc11(7369,:k);

PL/SQL procedure successfully completed.

SQL> print :k;

K

1400

Result: We have executed procedures using Pl/Sql

CONCLUSION:

COURSE OUTCOME	**PROGRAM OUTCOME Attained**
CO3	PO(1,2,3,4,5,8,9,10,12)

Practical No. 13

Practical No. 13

Aim: To implement triggers in Pl/Sql

Theory:

Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers are, in fact, written to be executed in response to any of the following events –

- A **database manipulation (DML)** statement (DELETE, INSERT, or UPDATE)
- A **database definition (DDL)** statement (CREATE, ALTER, or DROP).
- A **database operation** (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

Triggers can be defined on the table, view, schema, or database with which the event is associated.

Benefits of Triggers

Triggers can be written for the following purposes −

- Generating some derived column values automatically
- Enforcing referential integrity
- Event logging and storing information on table access
- Auditing
- Synchronous replication of tables

Code:

```
CREATE OR REPLACE TRIGGER show_salary_changes12
   BEFORE DELETE OR INSERT OR UPDATE ON emp
   for each row
     DECLARE
     sal_difference number;
   BEGIN
     sal_difference := :NEW.sal- :OLD.sal;
     dbms_output.put_line('Old salary: ' ||:OLD.sal);
     dbms_output.put_line('New salary: ' ||:NEW.sal);
     dbms_output.put_line('Salary difference: ' || sal_difference);
  END;
  /
```

Trigger created.

```
SQL> CREATE or REPLACE TRIGGER price_history_trigger11
  2  BEFORE update OF loc
  3  ON dept
  4  FOR EACH ROW
  5  BEGIN
  6  INSERT INTO dept_history
```

```
 7  VALUES
 8  (:old.deptno,  :old.dname,  :old.loc );
 9  END;
```

Result: Thus we have successfully implemented triggers in Pl/Sql

CONCLUSION:

COURSE OUTCOME	**PROGRAM OUTCOME Attained**
CO3	PO(1,2,3,4,5,8,9,10,12)