PRACTICAL NO. 01

AIM OF PRACTICAL :

To Study the DBMS, RDBMS and Design the Entity Relationship Diagram

LEARNING OBJECTIVES: -

1. To describe the uses of DBMS within educational settings.
2. To describe benefits and structure of a relational DBMS
3. To understand the Database Designing with ER diagrams

LEARNING OUTCOMES: -

1. Design and implement a simple DBMS to integrate into a classroom lesson.
2. Study the requirements and design the ER diagrams for the databse

After studying this practical students are able to:

1. Understand the basic concepts and terminology related to DBMS.
2. Know and use the procedures to design and implement a basic DBMS
3. Design the logical schema using E R Diagrams

SOFTWARE REQUIRED:-

SN	Name of Equipment/ Items / Software Tool	Specification	Qty Required
1.	Hardware : Computer System	Computer (i3-i5 preferable) , RAM minimum 2GB onwards	As per batch size
2.	Operating System	Windows 7 or Later Version / LINUX version 5.0 or Later Version	
3	Software	MySQL / Oracle or SQL Server	

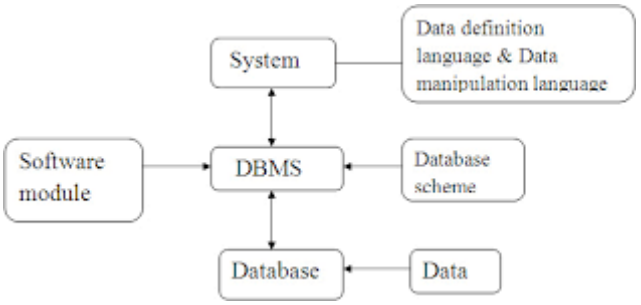
Problem Definition:

Design an E R diagram for a company having employees working in different

department on different projects

CIRCUIT DIAGRAM / BLOCK DIAGRAM:

Block Diagram of DBMS:



THEORY:

DBMS (Database Management System)

A **DBMS** is software that allows creation, definition and manipulation of database, allowing users to store, process and analyze data easily. DBMS provides us with an interface or a tool, to perform various operations like creating database, storing data in it, updating data, creating tables in the database and a lot more.

DBMS also provides protection and security to the databases. It also maintains data consistency in case of multiple users.

Here are some **examples** of popular DBMS used these days:

- MySql
- Oracle
- SQL Server
- IBM DB2
- PostgreSQL
- Amazon SimpleDB (cloud based) etc.

Characteristics of Database Management System

A database management system has following characteristics:

1. **Data stored into Tables:** Data is never directly stored into the database. Data is stored into tables, created inside the database. DBMS also allows having relationships between tables which makes the data more meaningful and connected. You can easily understand what type of data is stored where by looking at all the tables created in a database.
2. **Reduced Redundancy:** In the modern world hard drives are very cheap, but earlier when hard drives were too expensive, unnecessary repetition of

data in database was a big problem. But DBMS follows **Normalization** which divides the data in such a way that repetition is minimal.

3. **Data Consistency:** On Live data, i.e. data that is being continuously updated and added, maintaining the consistency of data can become a challenge. But DBMS handles it all by itself.
4. **Support Multiple user and Concurrent Access:** DBMS allows multiple users to work on it (update, insert, and delete data) at the same time and still manages to maintain the data consistency.
5. **Query Language:** DBMS provides users with a simple Query language, using which data can be easily fetched, inserted, deleted and updated in a database.
6. **Security:** The DBMS also takes care of the security of data, protecting the data from un-authorized access. In a typical DBMS, we can create user accounts with different access permissions, using which we can easily secure our data by restricting user access.
7. DBMS supports **transactions**, which allows us to better handle and manage data integrity in real world applications where multi-threading is extensively used.

Advantages of DBMS

- Segregation of application program.
- Minimal data duplicity or data redundancy.
- Easy retrieval of data using the Query Language.
- Reduced development time and maintenance need.
- With Cloud Datacenters, we now have Database Management Systems capable of storing almost infinite data.
- Seamless integration into the application programming languages which makes it very easier to add a database to almost any application or website.

Disadvantages of DBMS

- It's Complexity
- Except MySQL, which is open source, licensed DBMSs are generally costly.
- They are large in size.

RDBMS (Relational Database management System)

A **Relational Database management System** (RDBMS) is a database management system based on the relational model introduced by E.F Codd. In

relational model, data is stored in **relations** (tables) and is represented in form of **tuples** (rows).

RDBMS is used to manage Relational database. Relational database is a collection of organized set of tables related to each other, and from which data can be accessed easily. Relational Database is the most commonly used database these days.

In relational model in which data is stored in multiple tables where tables are related to each other using primary keys and foreign keys and indexes. RDBMS uses database normalization techniques to avoid redundancy in tables. It helps to fetch data faster using SQL query. It is widely used by enterprises and software developers to store large amount of complex data.

Examples:

- SQL server,
- Oracle
- MySQL
- MariaDB
- SQLite

Important Concept Related to RDBMS:

Table

In Relational database model, a **table** is a collection of data elements organized in terms of rows and columns. A table is also considered as a convenient representation of **relations**. But a table can have duplicate row of data while a true **relation** cannot have duplicate data. Table is the simplest form of data storage. Below is an example of an Employee table.

ID	Name	Age	Salary
1	Asha	34	13000
2	Anand	28	15000
3	Surabhi	20	18000
4	Rani	42	19020

Tuple

A single entry in a table is called a **Tuple** or **Record** or **Row**. A **tuple** in a table

represents a set of related data. For example, the above **Employee** table

has 4 tuples/records/rows.

Following is an example of single record or tuple.

1	Asha	34	13000

Attribute

A table consists of several records (row), each record can be broken down into several smaller parts of data known as **Attributes**. The above **Employee** table consists offourattributes, **ID**, **Name**, **Age** and **Salary**.

Attribute Domain

When an attribute is defined in a relation (table), it is defined to hold only a certain type of values, which is known as **Attribute Domain**. Hence, the attribute **Name** will hold the name of employee for every tuple. If we save employee's address there, it will be violation of the Relational database model.

Name
Asha
Anand
Surbhi - 9/401, OC Street, Amsterdam

Advantages of RDBMS

- It is easy to use.
- It is secured in nature.
- The data manipulation can be done.
- It limits redundancy and replication of the data.
- It offers better data integrity.
- It provides better physical data independence.
- It offers logical database independence i.e. data can be viewed in different ways by the different users.
- It provides better backup and recovery procedures.
- It provides multiple interfaces.

- Multiple users can access the database which is not possible in DBMS.

Disadvantages of RDBMS

- Software is expensive.
- Complex software refers to expensive hardware and hence increases overall cost to avail the RDBMS service.
- It requires skilled human resources to implement.
- Certain applications are slow in processing.
- It is difficult to recover the lost data.

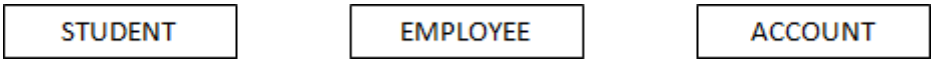
E.R Model:

The **entity-relationship (E-R)** data model is based on a perception of a real world that consists of a set of basic objects called **entities**, and of **relationships** among these objects.

The model is intended primarily for the database-design process. It was developed to facilitate database design by allowing the specification of an **enterprise schema**. Such a schema represents the overall logical structure of the database. This overall structure can be expressed graphically by an **E-R diagram**.

Entity: An **entity** is an object that exists in the real world and is distinguishable from other objects. We express the distinction by associating with each entity a set of attributes that describes the object. Ex: Student having attributes id, name, age, etc. Entity is represented by a Rectangle as follows:

.



Weak Entity: An entity set that does not have sufficient attributes to form a primary key is termed a **weak entity set**. An entity set that has a primary key is termed a **strong entity set**

A strong entity is represented by simple rectangle as shown above. A weak entity is represented by two rectangles as shown below.

A **relationship** is an association among several entities. The collection of all entities of the same type is an **entity set**, and the collection of all relationships of the same type is a **relationship set**.

A **superkey** of an entity set is a set of one or more attributes that, taken collectively, allows us to identify uniquely an entity in the entity set. We choose a minimal superkey for each entity set from among its superkeys; the minimal superkey is termed the entity set's **primary key**. Similarly, a relationship set is a set of one or more attributes that, taken collectively, allows us to identify uniquely a relationship in the relationship set. Likewise, we choose a mini-

ttribute: An oval shape is used to represent the attributes. Name of the attribute is written inside the oval shape and is connected to its entity by a line. Each Attribute belongs to a domain which is set of permissible values of the attribute. The attributes can be of may types:

Simple attribute: If an attribute cannot be divided into simpler components, it is a Simple attribute. Example for simple attribute : employee_id of an employee.

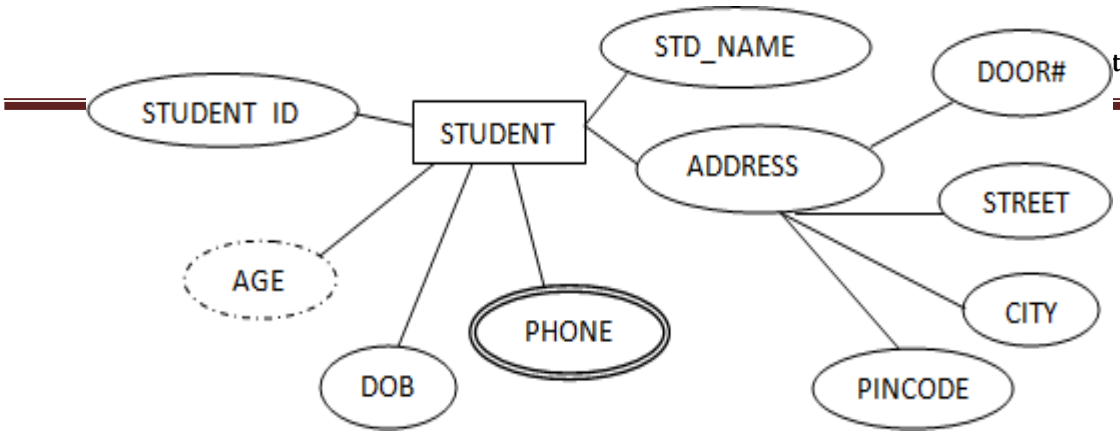
Composite attribute: If an attribute can be split into components, it is called a composite attribute.Example for composite attribute : Name of the employee which can be split into First_name, Middle_name, and Last_name.

Single valued Attributes: If an attribute can take only a single value for each

entity instance, it is a single valued attribute. example for single valued attribute : age of a student. It can take only one value for a particular student.

Multi-valued Attributes

Multivalued attributes are represented by double oval shape; whereas derived attributes are represented by oval shape with dashed lines. A composite attribute is also represented by oval shape, but these attribute will be connected to its parent attribute forming a tree structure.

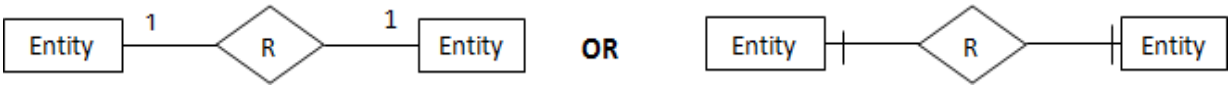


Primary Key: An underline to the attribute name is put to represent the primary key. The key attribute of the weak entity is represented by dashed underline.

Relationship: A diamond shape is used to show the relationship between the entities. A mapping with weak entity is shown using double diamond. Relationship name will be written inside them.

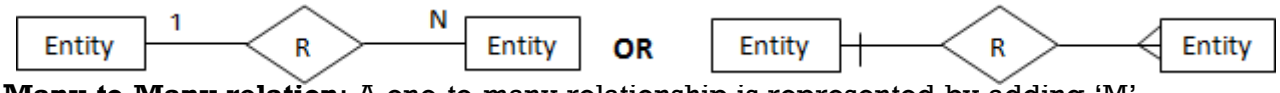
Cardinality of Relationship: Different developers use different notation to represent the cardinality of the relationship. Not only for cardinality, but for other objects in ER diagram will have slightly different notations. But main difference is noticed in the cardinality. For not to get confused with many, let us see two types of notations for each.

One-to-one relation: - A one-to-one relationship is represented by adding '1' near the entities on the line joining the relation. In another type of notation one dash is added to the relationship line at both ends.

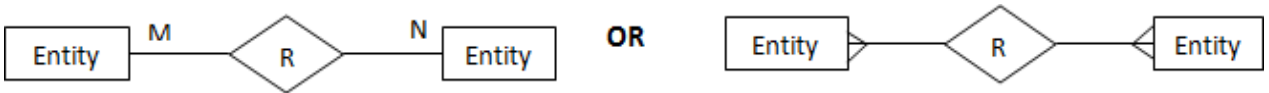


One-to-Many relation: A one-to-many relationship is represented by adding '1' near the entity at left hand side of relation and 'N' is written near the entity at right

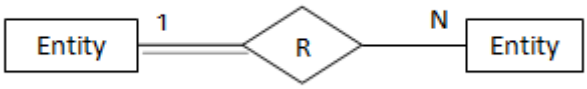
side. Other type of notation will have dash at LHS of relation and three arrow kind of lines at the RHS of relation as shown below.



Many-to-Many relation: A one-to-many relationship is represented by adding 'M' near the entity at left hand side of relation and 'N' is written near the entity at right side. Other type of notation will have three arrow kinds of lines at both sides of relation as shown below.

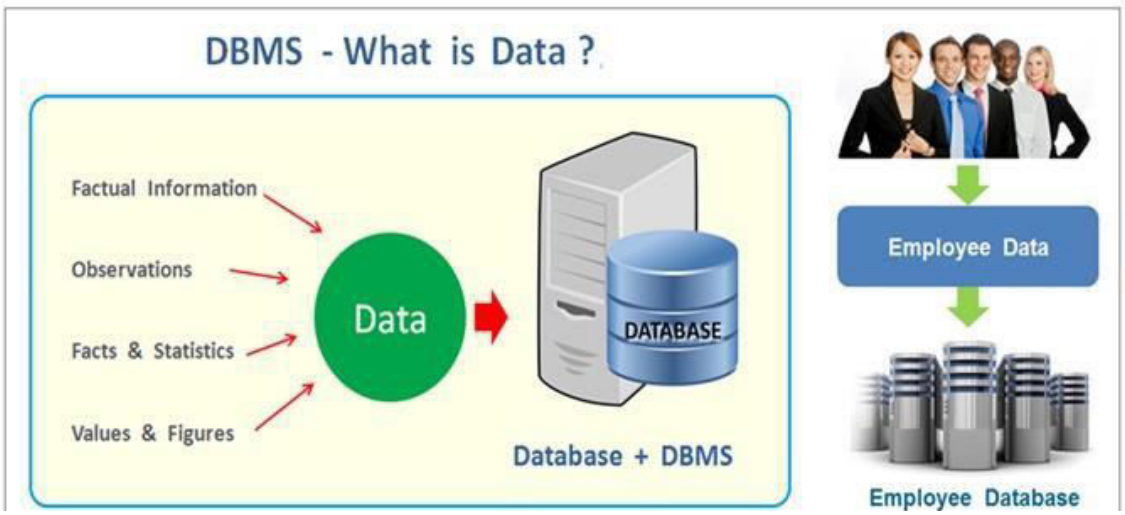


Participation Constraints: Total participation constraints are shown by double lines and partial participations are shown as single line.

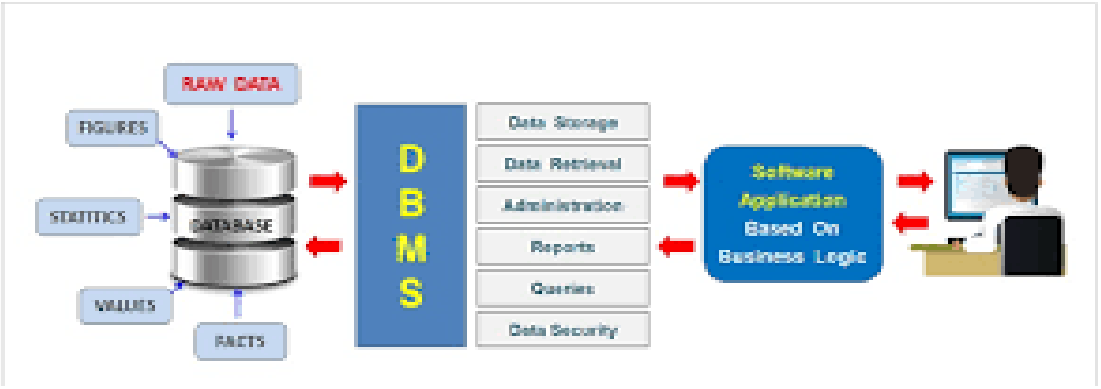


PROCEDURE/FLOW-CHARTS

DBMS - Database Management System - What Is Data ?



RDBMS - What Is RDBMS ?



E R Diagram for Company Database:

(Draw ER diagram for company database by yourself only,)

Conclusion:
Write it by your own

Signature of Faculty

PRACTICAL NO. 02

AIM OF PRACTICAL :

To Implement Data Definition Language .

LEARNING OBJECTIVES: -

1. To study SQL basics and Different SQL DDL commands like CREATE, ALTER, RENAME and DROP.
2. To study different SQL DML commands like SELECT, INSERT, UPDATE,
3. DELETE

LEARNING OUTCOMES: -

a. Use different ways of imposing the integrity constraints on the relations.
b. Design and implement a simple relation to get the results.

After studying thispractical students are able to:

1. Understand the basic concepts commands for the creation of a relation.
2. Learn the use of basic commands for data manipulation.

PROBLEM DEFINITION:

Create table to store information of students and their departmentsas follows:

Student (id, name, age, phone, address, deptid)
Department (deptid, dname, location)

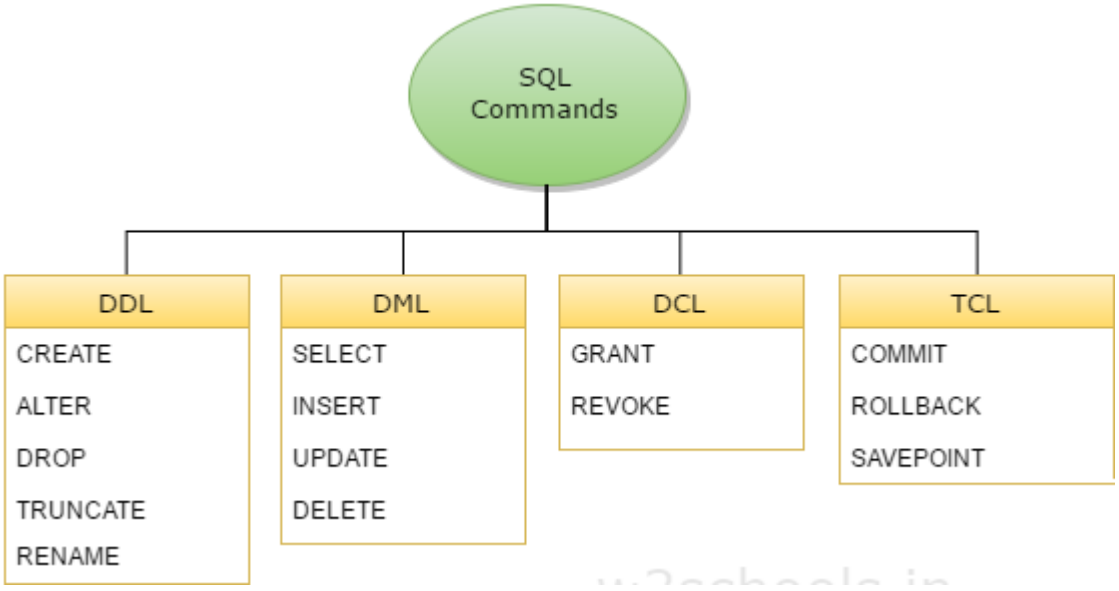
a) Create a table with several attributes and different constarints.
b) Describe the schema of the table
c) Add a column BTno in students table.
d) Change the data type of BTno.
e) Rename the column phone to phno.
f) Delete the column age.
g) Rename the table department as Branch
h) Delete the table data.
i) Delete the table schema.

SOFTWARE REQUIRED :-

SN	Name of Equipment/ Items / Software Tool	Specification	Qty Required
1.	Hardware : Computer System	Computer (i3-i5 preferable) , RAM minimum 2GB onwards	As per batch size
2.	Operating System	Windows 7 or Later Version / LINUX version 5.0 or Later Version	
3	Software	MySQL / Oracle or SQL Server	

CIRCUIT DIAGRAM / BLOCK DIAGRAM:

Block Diagram of SQL Commands:



THEORY:-

SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in relational database. SQL is the standard language for Relation Database System. All relational database management systems like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as standard database language. The database used in the practical is MySQL or Oracle .

The SQL provides the set of commands to perform different operations on the database.

These commands are classified into different types:

1. Data Definition Language

2. Data Manipulation language
3. Data Control Language
4. Transaction Control Language

TCL (Transaction Control Language) :

Transaction Control Language commands are used to manage transactions in the database. These are used to manage the changes made by DML-statements. It also allows statements to be grouped together into logical transactions.

Examples of TCL commands –

COMMIT: Commit command is used to permanently save any transaction into the database.

ROLLBACK: This command restores the database to last committed state.

It is also used with savepoint command to jump to a savepoint in a transaction.

SAVEPOINT: Savepoint command is used to temporarily save a transaction sothat you can rollback to that point whenever necessary.

DCL (Data Control Language) :

A Data Control Language is a syntax similar to a computer programming language used to control access to data stored in a database (Authorization). In particular, it is a component of Structured Query Language (SQL).

Examples of DCL commands :

GRANT: allow specified users to perform specified tasks.
REVOKE: cancel previously granted or denied permissions.

Data Definition Language

The data definition language is used to create an object, alter the structure of an object and also drop already created object. The Data Definition Languages used for table definition can be classified into following:

- Create table command
- Alter table command
- Truncate table command
- Drop table command
- Rename command
- Describe command

DDL Commands :

1) **CREATE TABLE** : It is used to create a table

Syntax:

CREATE TABLE table_name (column_name1 data_ type , column_name2 data_ type …);

Rules:

1. Oracle reserved words cannot be used.
2. Underscore, numerals, letters are allowed but not blank space.
3. Maximum length for the table name is 30 characters.
4. 2 different tables should not have same name.
5. We should specify a unique column name.
6. We should specify proper data type along with width.
7. We can include ‘‘not null’’ condition when needed. By default it is ‘null’.

2) **ALTER TABLE**: Alter command is used to:

1. Add a new column.
2. Modify the existing column definition.
3. Drop a column
4. Rename a column name
5. Add or drop integrity constraints.

Syntax:

ALTER TABLE <table_name> [add/drop/modify] attributes datatype (size);

ALTER TABLE <table_name> renames oldclname to newcolname;

ALTER TABLE <table_name> modify <colname> [add / drop] <constraint name>constraint;

ALTER TABLE MODIFY(COLUMN DEFINITION.);

3) DESCRIBING TABLE: used to describe the table schema.

Syntax: Desc <table_name>;

4) RENAME: this command is used to change the name of table view sequence or synonym.

Syntax: rename <old name> to <new name>;

5) DROP TABLE: It will delete the table structure provided the table should be empty.

Syntax: drop table <table_name>;

6) TRUNCATE TABLE: If there is no further use of records stored in a table and the structure has to be retained then the records alone can be deleted.

Syntax: TRUNCATE TABLE <TABLE NAME>;

Conclusion:

PRACTICAL NO. 03

AIM OF PRACTICAL :

To Implement Data Manipulation Language.

LEARNING OBJECTIVES: -

1. To study different SQL DML commands like SELECT, INSERT , UPDATE , DELETE

LEARNING OUTCOMES: -

a. Use different ways of imposing the integrity constraints on the relations.

b. Design and implement a simple relation to get the results.

After studying this practical students are able to:

1. Understand the basic concepts commands for the creation of a relation.
2. Learn the use of basic commands for data manipulation.

PROBLEM DEFINITION:

Create a table to store library information in college which stores the information about the books and the students and the transactions done.

Create attribute such as enrollment no, name, department, year and section, date of issue

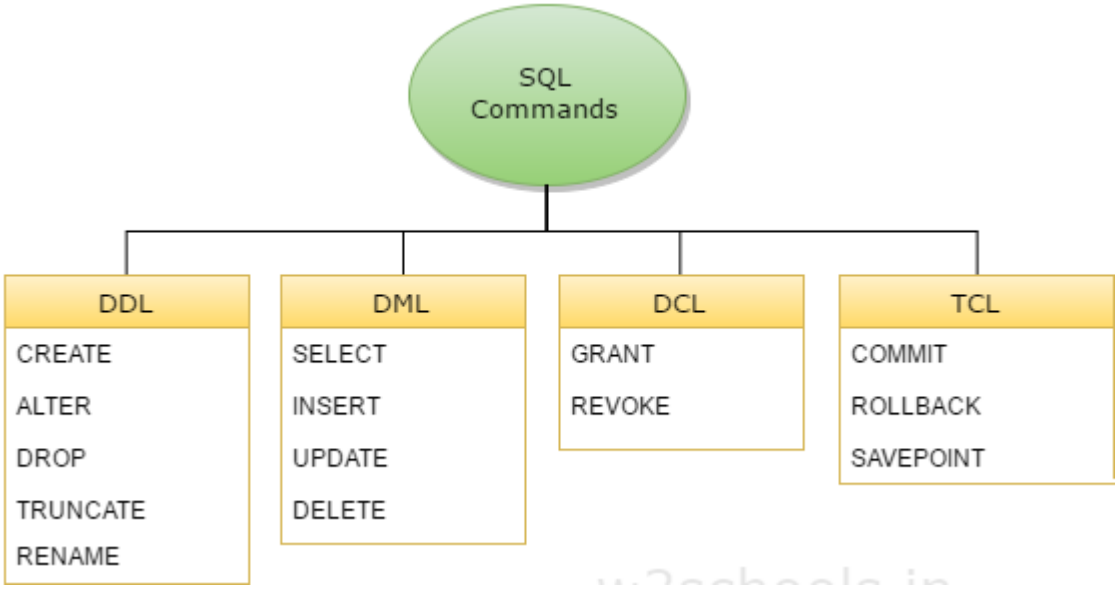
a) Create a table with several attributes and different datatypes.
b) Add a column BTno in students table.
c) Change the data type of BTno.
d) Delete the column age.
e) Update table data
f) Delete the table data.
g) Delete the table schema.

SOFTWARE REQUIRED :-

SN	Name of Equipment/ Items / Software Tool	Specification	Qty Required
1.	Hardware : Computer System	Computer (i3-i5 preferable) , RAM minimum 2GB onwards	**As per batch size**
2.	Operating System	Windows 7 or Later Version / LINUX version 5.0 or Later Version	
3	Software	MySQL / Oracle or SQL Server	

CIRCUIT DIAGRAM / BLOCK DIAGRAM:

Block Diagram of SQL Commands:



THEORY:-

SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in relational database. SQL is the standard language for Relation Database System. All relational database management systems like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as standard database language. The database used in the practical is MySQL or Oracle . The SQL provides the set of commands to perform different operations on the database.

These commands are classified into different types:

1. Data Definition Language

2. Data Manipulation language
3. Data Control Language
4. Transaction Control Language

DML COMMANDS:

DML commands are the most frequently used SQL commands and is used to query and manipulate the existing database objects. Some of the commands are INSERT, SELECT, UPDATE, DELETE.

The examples of DML in the Database Management System (DBMS) are as follows −

- SELECT − Retrieve data from the database.
- INSERT − Insert data into a table.
- UPDATE − Update existing data within a table.
- DELETE − Delete records from a database table.

Syntax for DML Commands

The syntax for the DML commands are as follows −

INSERT

Insert command is used to insert data into a table.

The syntax for insert command is as follows −

Syntax

Insert into <table_name> (column list) values (column values);

For example, if we want to insert multiple rows to the Employee table, we can use the following command −

Example

Insert into Employee(Emp_id, Emp_name) values (001, " bhanu");

Insert into Employee(Emp_id, Emp_name) values (002, " hari");

Insert into Employee(Emp_id, Emp_name) values (003, " bob");

SELECT

Select command is used to retrieve data from the database.

The syntax for the select command is as follows −

Syntax

SELECT * from <table_name>;

For example, if we want to select all rows from the Employee database, we can use the following command −

Example

SELECT * from Employee;

DELETE

Delete command is used to delete records from a database table.

The syntax for the delete command is as follows -

Syntax

Delete from <table_name>WHERE condition;

For example, if we want to delete an entire row of employee id 002, we can use the following command −

Example

DELETE from Employee WHERE Emp_id=002;

UPDATE

Update command is used to update existing data within a table.

The syntax for the update command is as follows -

Syntax

UPDATE <table_name> SET column_number =value_number WHERE condition;

For example, if we want to update the name of the employee having the employee id 001, we can use the command given below −

Example

UPDATE Employee SET Emp_name= Ram WHERE Emp_id= 001;

conclusion:

PRACTICAL NO. 04

AIM OF PRACTICAL :

To Implement various types of integrity constraints in SQL

LEARNING OBJECTIVES: -

1. To understand the use of key in formulating the relation.
2. To know the importance of having a redundancy free relation

LEARNING OUTCOMES: -

1. Implement the rules of data normalization to improve DBMS design
2. Design and implement a constraint in DBMS

After studying this practical students are able to:

1. Understand the basic constraint and terminology related to DBMS.
2. Know and use the key to design and implement a DBMS

SOFTWARE REQUIRED :-

SN	Name of Equipment/ Items / Software Tool	Specification	Qty Required
1.	Hardware : Computer System	Computer (i3-i5 preferable) , RAM minimum 2GB onwards	**As per batch size**
2.	Operating System	Windows 7 or Later Version / LINUX version 5.0 or Later Version	
3	Software	MySQL / Oracle or SQL Server	

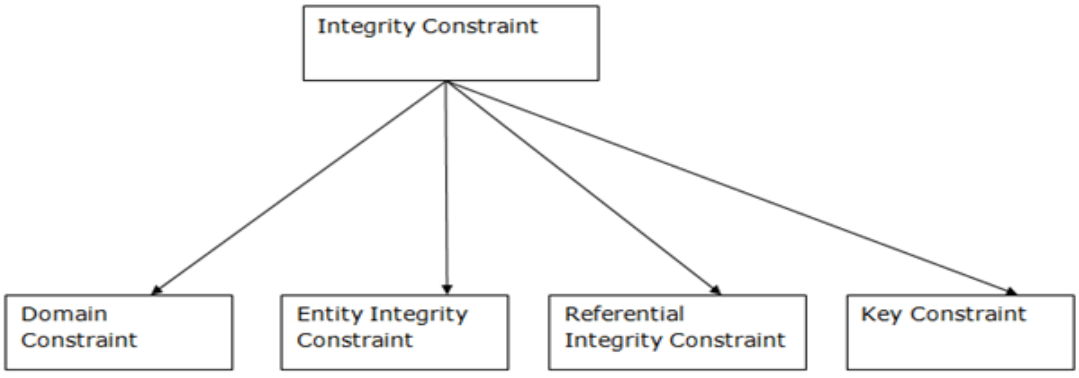
PROBLEM DEFINITION:

Add the actual data of the students and department in the tables created in practical 2 and perform the following Queries.

a) Make Id of Student a primary key.
b) Make the Name and address field as not null.
c) Impose constraint on BTno such that values are unique.
d) Insert default value of age as 20
e) Insert multiple rows in the table one by one
f) Insert multiple rows using single query.
g) Insert some rows skipping values of some attributes.
h) Display the complete data entered into the tables.
i) Update student name whose id is 3
j) Change address and phone no of student with name XYZ.
k) Delete the phone no of student 23.
l) Delete the students who belong to Amravati city.
m) Display all students who belong to city Akola.
n) Display the student names with their department names.

CIRCUIT DIAGRAM / BLOCK DIAGRAM:

Block Diagram of Constraint:



THEORY:

Database Constarints:

Integrity Constraint: An integrity constraint is a mechanism used by oracle to prevent invalid data entry into the table. It has enforcing the rules for the columns in a table. The types of the integrity constraints are:

a) Domain Integrity b) Entity Integrity c) Referential Integrity

a) Domain Integrity: it is also called as **Attribute constraints.** This constraint sets a range and any violations that take place will prevent the user from performing the manipulation that caused the breach. The scope of it is only one attribute. It includes:

i. **Not Null constraint:**

While creating tables, by default the rows can have null value .the enforcement of not null constraint in a table ensure that the table contains values.

Principle of null values:

- Setting null value is appropriate when the actual value is unknown, or when a value would not be meaningful.
- A null value is not equivalent to a value of zero.
- A null value will always evaluate to null in any expression.
- When a column name is defined as not null, that column becomes a mandatory i.e., the user has to enter data into it.
- Not null Integrity constraint cannot be defined using the alter table command when the table contain rows.
- Syntax: create table <tbnm> (attr1 datatype **not null,**……)

ii. **Check Constraint:** Check constraint can be defined to allow only a particular range of values when the manipulation violates this constraint, the record will be rejected. Check condition cannot

contain sub queries.

- Syntax: create table <tbnm> (attr1 datatype **check (condition)**,……)

III. **Default Constraint:** It is used to set the default value of the attribute to some predefine value. If the user does not provide the value, then value given as default will be assigned.

- Syntax: create table <tbnm> (attr1 datatype **Default <default value>**,……)

b) **Entity Integrity:** it is also called as **Table constraints** Maintains uniqueness in a record. An entity represents a table and each row of a table represents an instance of that entity. To identify each row in a table uniquely we need to use this constraint. There are 2 entity constraints:

- Syntax: create table <tbnm> (attr1 datatype,……attr_n datatypen , **<table constraints>**);

I. **Unique key constraint:** It is used to ensure that information in the column for each record is unique, as with telephone or drivers license numbers. It prevents the duplication of value with rows of a specified column in a set of column. A column defined with the constraint can allow null value.

If unique key constraint is defined in more than one column i.e., combination of column cannot be specified. Maximum combination of columns that a composite unique key can contain is 16.

- Syntax: create table <tbnm> (attr1 datatype,……attr_n datatypen ,**Unique** (Attrlist));

II. **Primary Key Constraint:** A primary key avoids duplication of rows and does not allow null values. It can be defined on one or

more columns in a table and is used to uniquely identify each row in a table. These values should never be changed and should never be null. A table should have only one primary key. If a primary key constraint is assigned to more than one column or combination of column is said to be composite primary key, which can contain 16 columns.

- Syntax: create table <tbnm> (attr1 datatype,……attr_n datatype_n ,**Primary Key** (Attrlist));

c) Referential Integrity: It enforces relationship between tables. To establish parent-child relationship between 2 tables having a common column definition, we make use of this constraint. To implement this, we should define the column in the parent table as primary key and same column in the child table as foreign key referring to the corresponding parent entry.

I. Foreign key A column or combination of column included in the definition of referential integrity, which would refer to a referenced key.

II. Referenced key: It is a unique or primary key upon which is defined on a column belonging to the parent table.

PROCEDURE/FLOW-CHARTS

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

Not allowed. Because AGE is an integer attribute

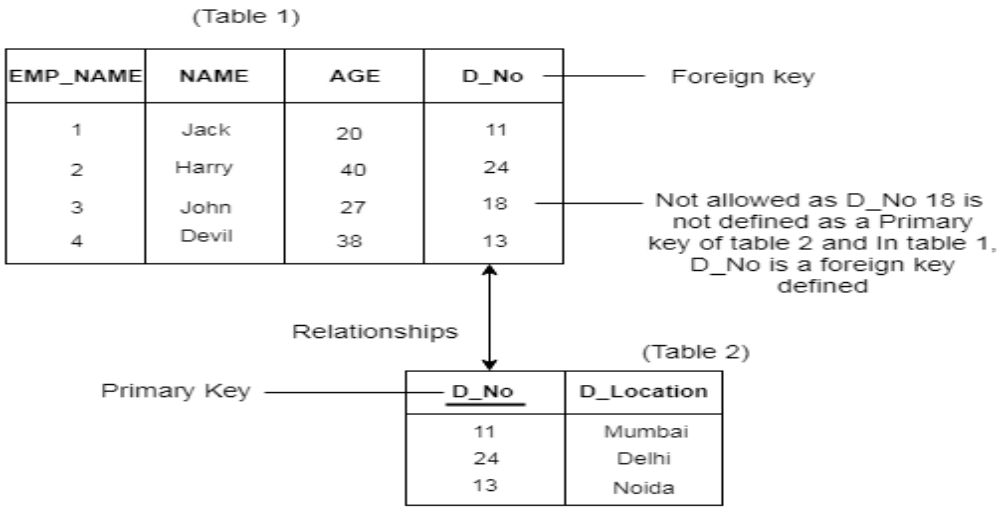
Entity Integrity constraints

EMPLOYEE

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

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

Referential Integrity constraints



Key constraints

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

Not allowed. Because all row must be unique

OUTPUT / RESULTS & ANALYSIS:

a) **Make Id of Student a primary key.**

Alter Table Student Modify Sid int prmary Key;

b) **Make the Name and address field as not null.**

Alter Table Student modify Name Varchar(25) Not Null; Alter Table Student Modify Address varchar(25) NotNull;

c) **Impose constraint on BTno such that values are unique.**

Alter Table Student Modify BTno int unique;

d) **Insert default value of age as 20**

Alter Table Student ADD age int default(20);

e) **Insert multiple rows in the table one by one**

Insert into student values (23, “name”, 23, “Amravati”, 23455, 5);

f) **Insert multiple rows using single query.**

Insert into department values (2, ”CSE”, ”building1”),
(3,”electrical”,”building2”) ;

g) **Insert some rows skipping values of some attributes.**

Insert into department (deptid, dname) values (3, “extc”);

h) **Display the complete data entered into the tables.**

Select * from Student;

i) **Update student name whose id is 3**

UPDATE Student set sname=”Amit” where deptid=3;

j) **Change address and phone no of student with name XYZ.**

UPDATE Student SET address=“Rajapeth” Phone=23456789 where sname=”XYZ”;

k) Delete the phone no of student 23.

UPDATE student SET Phone= NULL where sid=23;

l) Delete the students who belong to Amravati city.

DELETE from Student where Address=”AMRAVATI”;

m) Display all students who belong to city Akola.

SELECT * from Students where Address=”Akola”;

n) Display the student names with their department names.

SELECT sname,Dname FROM
Student, Department WHERE
Student.deptid=
Department.deptid;

Signature of Faculty

PRACTICAL NO. 05

AIM OF PRACTICAL :

To study operators, expressions, functions and sub queries in SQL

LEARNING OBJECTIVES: -

1. To implement different types of operator's expressions, built in functions and sub queries in SQL query.
2. To understand and study the use of sub queries in SQL expression.

LEARNING OUTCOMES: -

1. Learn how the particular condition should be checked
2. Study and explore the use of inbuilt functions in SQL.

After studying this practical students are able to:

1. Understand the use of operators for satisfying certain condition.
2. Learn the expressions and use of inbuilt functions & subqueries.

SOFTWARE REQUIRED :-

SN	Name of Equipment/ Items / Software Tool	Specification	Qty Required
1.	Hardware : Computer System	Computer (i3-i5 preferable), RAM minimum 2GB onwards	As per batch size
2.	Operating System	Windows 7 or Later Version / LINUX version 5.0 or Later Version	
3	Software	MySQL / Oracle or SQL Server	

PROBLEM DEFINITION:

Create a table containing info of an employee working in an organization in various dept apply some constraints, insert values and select the set of employees based on different condition using different operator and expression. Create table for employee and department as

Employee (empno, employee_name, address, manager, post, joining date, salary, age) **Department** (Dept_id, dept_name, location)

Belongs (Empno, Dept_id, hrs)

The department values must be (10,20,30,40,50)

Perform following operations by writing queries using functions-

1) Find out the difference between maximum and minimum salary of employee.
2) Display name and date of joining of employee who join in month of January.

3) Find out the most experienced employee.

4) Display employee name in small letter.

5) Calculate experience of each employee and print with employee name.

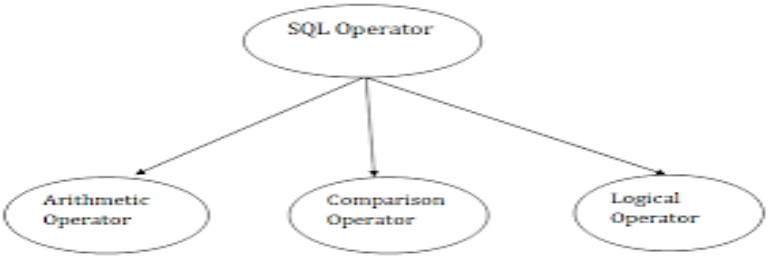
6) Display the months between 1 June 2010 & 1 August 2012.

7) List all jobs available in employee table

8) List the employee name and salary whose salary greater than salary of any name.

CIRCUIT DIAGRAM / BLOCK DIAGRAM:

Block Diagram of SQL operators:



THEORY:-

- **Operators**

There are mainly four types of operators are present in SQL memory.
1) Arithmetic Operators
2) Comparison Operators
3) Logical Operators
4) Operator for negating condition.

1) Arithmetic Operators

Arithmetic operators can perform arithmetical operations on numeric operands involved. Arithmetic operators are addition (+), subtraction (-), multiplication (*) and division (/). The + and - operators can also be used in date arithmetic.

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

%	Modulus - Divides left hand operand by right hand operand and returns remainder	b % a

2) Comparison Operators

Comparison operators are used to compare one expression to another expression. The result at comparison is true or false or unknown depending on the condition.

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

3) Logical Operators

Logical operators are used to produce a single result from the combination of two separate conditions.

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).

4) Operators for negating condition

These operators are used to reverse the meaning of expression. We can place not before any expression or keyword.

I) Single Row Functions:

Single row or Scalar functions return a value for every row that is processed in a query.

There are four types of single row functions. They are:

1) Numeric Functions:

These are functions that accept numeric input and return numeric values.

Numeric functions are used to perform operations on numbers. They accept numeric values as input and return numeric values as output. Few of the Numeric functions are:

Function Name	Return Value
ABS (x)	Absolute value of the number '*x*'
CEIL (x)	Integer value that is Greater than or equal to the number '*x*'
FLOOR (x)	Integer value that is Less than or equal to the number '*x*'
TRUNC (x, y)	Truncates value of number '*x*' up to '*y*' decimal places
ROUND (x, y)	Rounded off value of the number '*x*' up to the number '*y*' decimal places

These functions can be used on database columns.

2) Character or Text Functions: These are functions that accept character input and can return both character and number values.

Character or text functions are used to manipulate text strings. They accept strings or characters as input and can return both character and number values as output.

Few of the character or text functions are as given below:

Function Name	Return Value
LOWER (string_value)	All the letters in *'string_value'* is converted to lowercase.
UPPER (string_value)	All the letters in *'string_value'* is converted to uppercase.
INITCAP (string_value)	All the letters in *'string_value'* is converted to mixed case.
LTRIM (string_value, trim_text)	All occurrences of *'trim_text'* is removed from the left of *'string_value'*.
RTRIM (string_value, trim_text)	All occurrences of *'trim_text'* is removed from the right of *'string_value'*.

TRIM (trim_text FROM string_value)	All occurrences of *'trim_text'* from the left and right of *'string_value'* , *'trim_text'* can also be only one character long .
SUBSTR (string_value, m, n)	Returns *'n'* number of characters from *'string_value'* starting from the '*m*' position.
LENGTH (string_value)	Number of characters in *'string_value'* in returned.
LPAD (string_value, n, pad_value)	Returns *'string_value'* left-padded with *'pad_value'* . The length of the whole string will be of *'n'* characters.
RPAD (string_value, n, pad_value)	Returns *'string_value'* right-padded with *'pad_value'* . The length of the whole string will be of *'n'* characters.

3) Date Functions: These are functions that take values that are of datatype DATE as input and return values of datatype DATE, except for the MONTHS_BETWEEN function, which returns a number.

Few date functions are as given below.

Function Name	**Return Value**
ADD_MONTHS (date, n)	Returns a date value after adding *'n'* months to the date *'x'*.
MONTHS_BETWEEN (x1, x2)	Returns the number of months between dates x1 and x2.
ROUND (x, date_format)	Returns the date *'x'* rounded off to the nearest century, year, month, date, hour, minute, or second as specified by the *'date_format'*.
TRUNC (x, date_format)	Returns the date *'x'* lesser than or equal to the nearest century, year, month, date, hour, minute, or second as specified by the 'date_format'.
NEXT_DAY (x, week_day)	Returns the next date of the *'week_day'* on or after the date *'x'* occurs.
LAST_DAY (x)	It is used to determine the number of days remaining in a month from the date *'x'* specified.
SYSDATE	Returns the systems current date and time.
NEW_TIME (x, zone1, zone2)	Returns the date and time in zone2 if date 'x' represents the time in zone1.

4) Conversion Functions: These are functions that help us to convert a value in one form to another form. For Example: a null value into an actual value, or a value from one datatype to another datatype like NVL, TO_CHAR, TO_NUMBER, TO_DATE etc. You can combine more than one function together in an expression. This is known as nesting of functions.

These are functions that help us to convert a value in one form to another form. For Ex: a null value into an actual value, or a value from one datatype to another datatype like NVL, TO_CHAR, TO_NUMBER, TO_DATE.

Few of the conversion functions available in oracle are:

Function Name	**Return Value**
TO_CHAR (x [,y])	Converts Numeric and Date values to a character string value. It cannot be used for calculations since it is a string value.
TO_DATE (x [, date_format])	Converts a valid Numeric and Character values to a Date value. Date is formatted to the format specified by *'date_format'*.
NVL (x, y)	If *'x'* is NULL, replace it with *'y'*. *'x'* and *'y'* must be of the same datatype.

DECODE (a, b, c, d, e, default_value)	Checks the value of *'a'*, if *a = b*, then returns *'c'*. If *a = d*, then returns *'e'*. Else, returns *default_value*.

II) Group Functions: These functions group the rows of data based on the values returned by the query. This is discussed in SQL GROUP Functions. The group functions are used to calculate aggregate values like total or average, which return just one total or one average value after processing a group of rows.

5) Subqueries:

i. A **Subquery** or Inner query or Nested query is a query within another SQL query and embedded within the WHERE clause.
ii. A **Subquery** is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.
iii. **Subqueries** can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN etc
iv. **Nested subqueries** can also be used to create a table from an existing table.

Syntax:

Create table <tablename> (column1,column2-------column n) as (select column1,column2 column n from table_name);

OUTPUT / RESULTS & ANALYSIS:

1) **Find out the difference between maximum and minimum salary of employee.**

```
select max(salary)-min(salary) as difference from emp;
```

2) **Display name and date of joining of employee who join in month of January.**

```
select empname, to_char (dojoin, 'dd-mon-yyyy' ) from emp where
to_char (dojoin, 'mon' ) = 'jan';
```

3) **Find out the most experienced employee.**

```
SELECT empname FROM employees where DOJ in (SELECT MIN(DOJ)
from employees);
```

4) **Display employee name in small letter.**

```
select lower(empname),upper(pname) from emp;
```

5) **Calculate experience of each employee and print with employee name.**

```
select empname, (Systemdate()- doj) from employee;
```

6) **Display the months between 1 June 2010 & 1 August 2012.**

```
select distinct months_between('10/jan/12','01/aug/12') as difference from emp;
```

7) List all jobs available in employee table

```
Select Distinct post from employees;
```

8) List the employee name and salary whose salary greater than salary of any name.

```
select empname, salary from emp where salary >= (select salary from
emp where empname='Ragini');
```

CONCLUSION:-

ASSESSMENT SCHEME:-

Pre-Lab Test (2)	In Lab performance (5)	Post Lab Test (3)	Record (5)	Total (15)

Signature of Faculty

PRACTICAL NO. 07

AIM OF PRACTICAL :

To Implement various types of joins in SQL

LEARNING OBJECTIVES: -

1. To study join operations on various tables

2. To study different types of join to retrieve the data.

LEARNING OUTCOMES: -

1. Use different ways of getting the values from various tables or relations.
2. Design and implement a simple query for retrieval of data..

After studying this practical students are able to:

1. Understand the basic commands for applying the join on a relation.
2. Learn the use of basic commands for data retrival.

SOFTWARE REQUIRED :-

SN	Name of Equipment/ Items / Software Tool	Specification	Qty Required
1.	Hardware : Computer System	Computer (i3-i5 preferable) , RAM minimum 2GB onwards	As per batch size
2.	Operating System	Windows 7 or Later Version / LINUX version 5.0 or Later Version	
3	Software	MySQL / Oracle or SQL Server	

PROBLEM DEFINITION:

On the table employee and department perform different join operations as follows:

1) Display id, post, deptname and dept location of employee using equijoin,

innerjoin, natural join with USING and ON clause.

2) Display id, name, location having deptid > its dept using non equijoin.

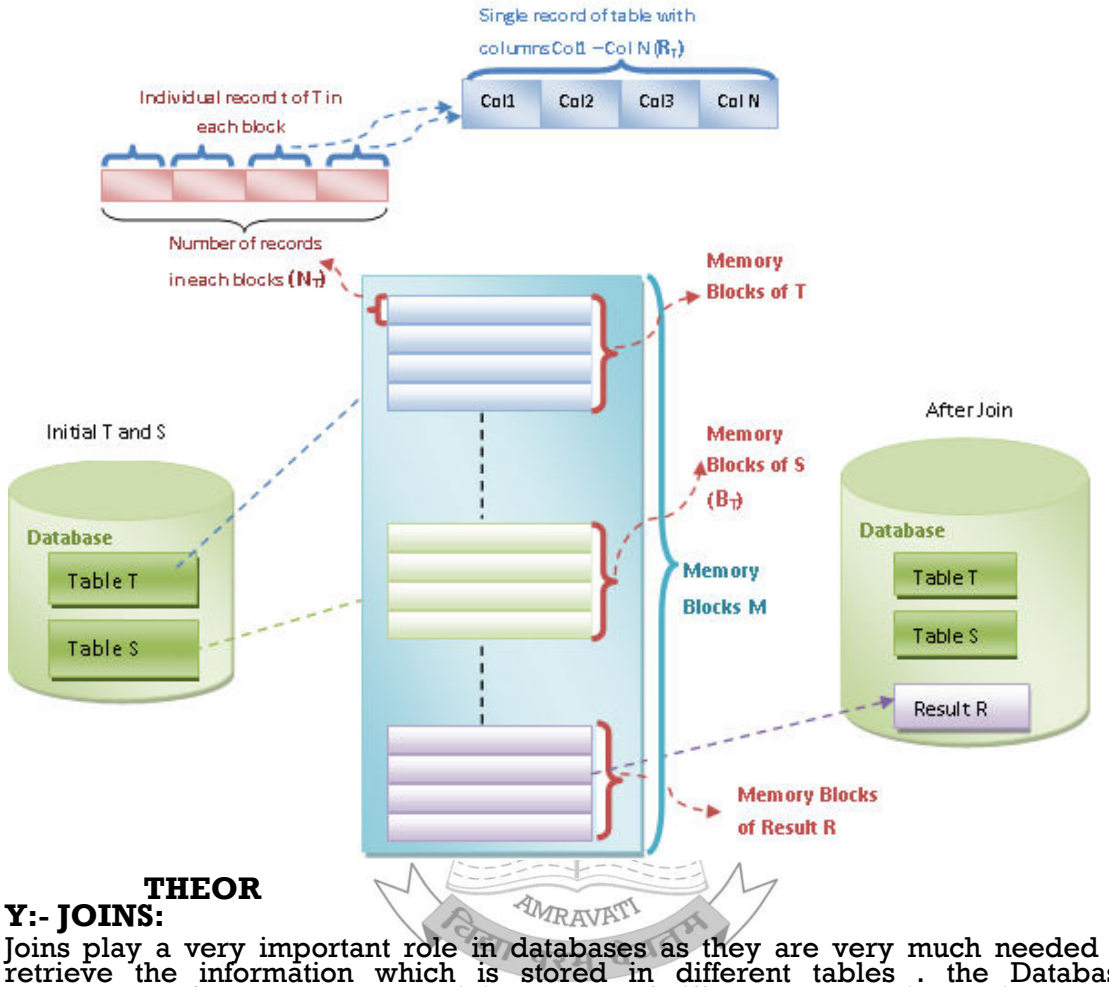
3) Display employee name manager name its department with job of manager using self join.

4) Perform Cartesian product of employee and department table using cross join.

5) Display all the information of employee and department using left right and full outer join.

CIRCUIT DIAGRAM / BLOCK DIAGRAM:

Block Diagram of Join Operations:



THEOR Y:- JOINS:

Joins play a very important role in databases as they are very much needed to retrieve the information which is stored in different tables . the Database Management System allows us to join the data of different tables using various types of joins as follows:

Inner Join-

Inner join returns only those records/rows that match/exists in both the tables. Syntax for Inner Join is as

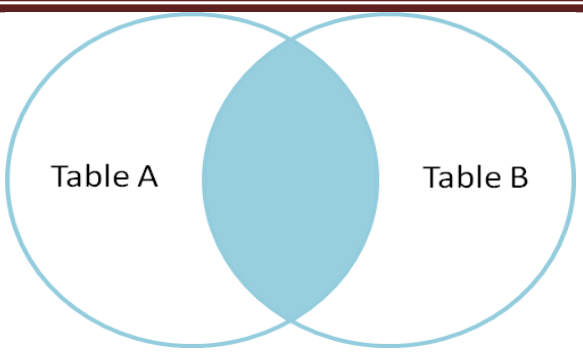
```
SELECT
table1.column1,table1.column2,table2.column1,....   FROM
table1
INNER JOIN table2
ON table1.matching_column = table2.matching_column;
```

table1: First table.

table2: Second table

matching_column: Column common to both the tables.

Note: JOIN instead of INNER JOIN. JOIN is same as INNER JOIN.



Outer Join-

We have three types of Outer Join.

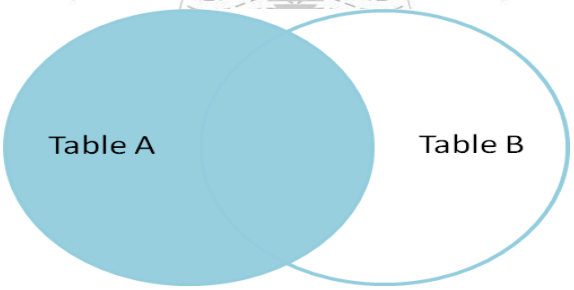
1. **Left Outer Join**

Left outer join returns all records/rows from left table and from right table returns only matched records. If there are no columns matching in the right table, it returns NULL values. Syntax for Left outer Join is as :

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
LEFT JOIN table2
ON table1.matching_column = table2.matching_column;
```

table1: First table.
table2: Second table

matching_column
: Column common to both the tables. Note: LEFT OUTER JOIN instead of LEFT JOIN, both are same.



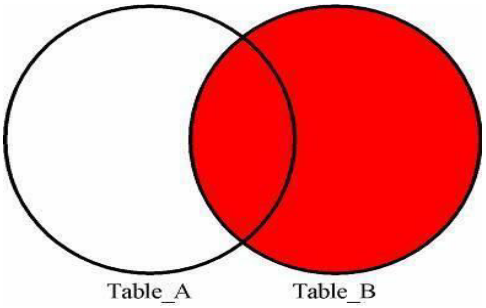
2. Right Outer Join

Right outer join returns all records/rows from right table and from left table returns only matched records. If there are no columns matching in the left table, it returns NULL values. Syntax for right outer Join is as :

```
SELECT table1.column1,table1.column2,table2.column1,.... FROM table1
RIGHT JOIN table2
ON table1.matching_column = table2.matching_column;
```

table1: First table.
table2: Second table
matching_column: Column common to both the tables.

Note: RIGHT OUTER JOIN instead of RIGHT JOIN, both are same.

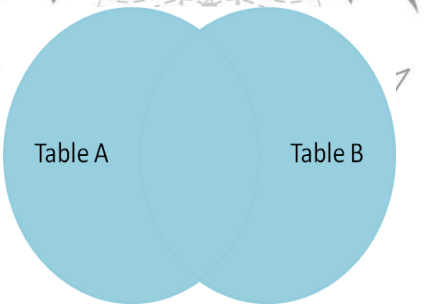


3. Full Outer Join

Full outer join combines left outer join and right outer join. This join returns all records/rows from both the tables.If there are no columns matching in the both tables, it returns NULL values. Syntax for full outer Join is as :

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
FULL JOIN table2
ON table1.matching_column = table2.matching_column;
```

table1: First table.
table2: Second table
matching_column: Column common to both the tables.



Cross Join

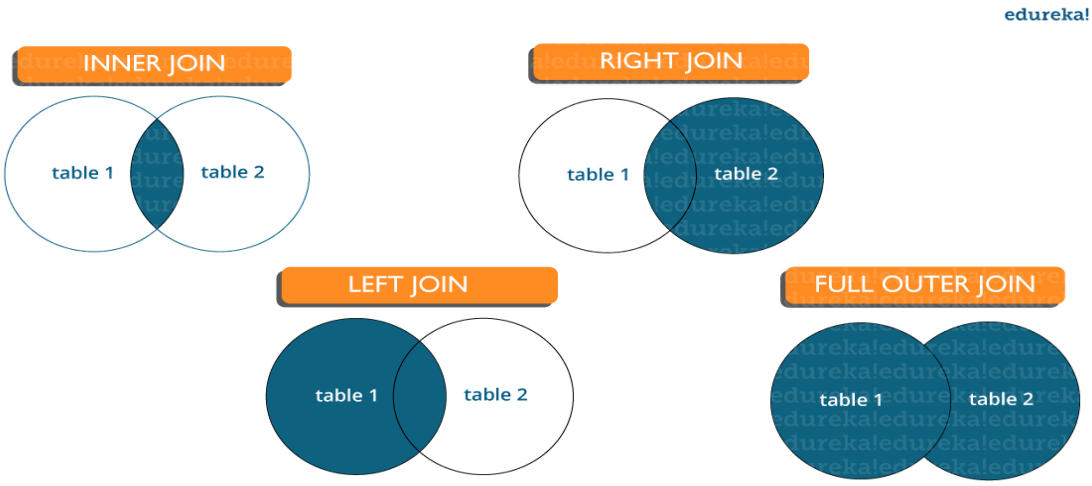
Cross join is a cartesian join means cartesian product of both the tables. This join does not need any condition to join two tables. This join returns records/rows that are multiplication of record number from both the tables means each row on left table will related to each row of right table. Syntax for right outer Join is as :

```
Select * from table_1
cross  join  table_2
```

Self Join

Self join is used to join a database table to itself, particularly when the table has a Foreign key that references its own Primary Key. Basically we have only three types of joins : Inner join, Outer join and Cross join. We use any of these three JOINS to join a table to itself. Hence Self join is not a type of Sql join.

PROCEDURE/FLOW-CHARTS



OUTPUT / RESULTS & ANALYSIS:

1) **Display id, post, deptname and dept location of employee using equijoin, innerjoin, natural join with USING and ON clause.**

```
SELECT eid, post, dname, location from Employee INNER JOIN
Department ON Employee.deptid=Department.deptid;
SELECT eid, post, dname, location from Employee INNER JOIN
Department USING (deptid);
SELECT eid, post, dname, location from Employee NATURAL JOIN
Department;
```

2) **Display id, name, location having deptid > its dept using non equijoin.**

```
SELECT eid, post, dname, location from Employee
 INNER JOIN Department on
Employee.deptid>Department.deptid;
```

3) **Display employee name manager name its department with job of manager using self join.**

```
SELECT eid, post, dname, location from Employee
as
 E INNER JOIN Employee as M on E.manager=M.eid;
```

4) **Perform Cartesian product of employee and department table using cross join.**

```
SELECT eid, post, dname, location from Employee
CROSS JOIN Department ;
```

5) **Display all the information of employee and department using left right and full outer join.**

```
SELECT eid, post, dname, location from
 Employee LEFT JOIN Department on
 Employee.deptid=Department.deptid;
SELECT eid, post, dname, location from Employee
RIGHT JOIN Department on
Employee.deptid=Department.deptid;
SELECT eid, post, dname, location from
Employee FULL JOIN Department on
Employee.deptid=Department.deptid;
```

CONCLUSION:

Here the different joins in SQL are studied, and learned about on and using clause. We have studied a equijoin, non-equijoin, inner join, outer join and its different types use have studied the cross join to take a Cartesian product of two relations and we have to come to know that using clause is not suitable for non equijoin

ASSESSMENT SCHEME:-

Pre-Lab Test (2)	In Lab performance (5)	Post Lab Test (3)	Record (5)	Total (15)

Signature of Faculty

PRACTICAL NO. 08

AIM OF PRACTICAL :

To Study and Implement VIEWS in SQL

LEARNING OBJECTIVES: -

1. To study the concept and advantages of using Views in DBMS
2. To implement views and study the updations on views

LEARNING OUTCOMES: -

1. Learn how to simplify queries using views.
2. Design create and use views in complex queries..

After studying thispractical students are able to:

1. Identify list the advantages of using views .
2. Create and use views in SQL.

SOFTWARE REQUIRED :-

SN	Name of Equipment/ Items / Software Tool	Specification	Qty Required
1.	Hardware : Computer System	Computer (i3-i5 preferable) , RAM minimum 2GB onwards	As per batch size
2.	Operating System	windows 7 or Later Version / LINUX version 5.0 or Later Version	
3	Software	MySQL / Oracle or SQL Server	

PROBLEM DEFINITION:

Using the table employee and departmentt from last practical perform different queries base on views as follows:

1) Create a view called manager which stores the information about the entire employee who is manager.

2)3) Create a view called manager which stores the information about the employee and their respective department name, deptname, job and location.

4) Display information of all the view.

5) Display the manager who works in '10' department.

6) Increment salary of a manager whose age is greater than 50 by 10%.

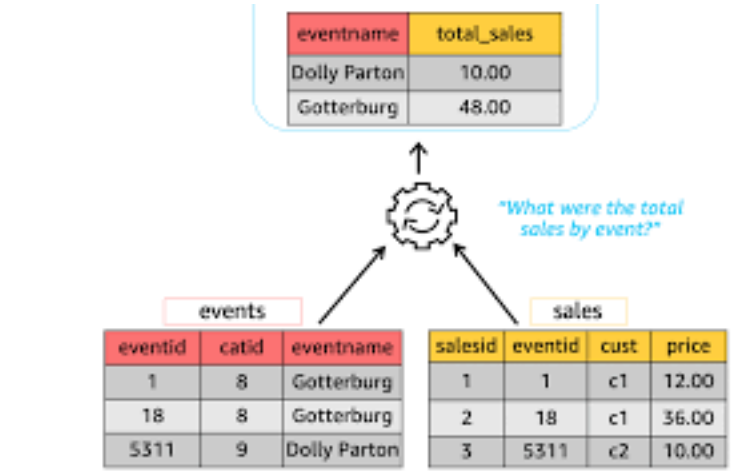
7) Show the updated information in the view and in the logical relation.

8) Update a dept name of emp who work in CSE to IT dept and show the result of a view as well as logical relation.

9) Drop the view.

BLOCK DIAGRAM:

Block Diagram of VIEWS:



THEORY:-

A view is nothing more than a SQL statement that is stored in the database with an associated name. A view is actually a composition of a table in the form of a predefined SQL query.

A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depend on the written SQL query to create a view.

Views, which are kind of virtual tables, allow users to do the following:

1. Structure data in a way that users or classes of users find natural or intuitive.

2. Restrict access to the data such that a user can see and (sometimes) modify exactly what they need and no more.

3. Summarize data from various tables which can be used to generate reports.

Creating Views:-

Database views are created using the **CREATE VIEW** statement. Views can be created from a single table, multiple tables, or another view.

To create a view, a user must have the appropriate system privilege according to the specific implementation.

The basic CREATE VIEW syntax is as follows:

```
CREATE VIEW
view_name AS SELECT
column1, column2.....
FROM table_name
WHERE [condition];
```

SQL 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
column_name(s)
FROM table_name
WHERE condition
```

SQL Dropping a View:-

You can delete a view with the DROP VIEW command.

SQL DROP VIEW Syntax:-

```
DROP VIEW view_name
```

OUTPUT / RESULTS & ANALYSIS:

1) **Create a view called manager which stores the information about the entire employee who is manager.**

   CREATE VIEW Managerview as select eid, ename , age, post, address,deptid,salary from Employee where manager=NOT NULL;

2) **Create a view called Empdata which stores the information about the employee and their respective department name, Post and location.**

   CREATE VIEW EMPDATA as

   select eid,ename, dname, location , post from Employee, department where Employee.deptid=Department.deptid;

3) **Display information of all the view.**

   Select * from Managerview;
   Select * from EMPDATA;

4) **Display the manager who works in '10' department.**

SELECT Ename from Managerview where deptid=10;

5) **Increment salary of a manager whose age is greater than 50 by 10%.**

Update Managerview set salary=salary*1.1 where age>50;

6) **Show the updated information in the view and in the logical relation.**

Select * from Manager view ;
Select * from Employees;

7) **Update a dept name of employee who work in CSE to IT dept and show the result of a view as well as logical relation.**

Update EMPDATA set dname=''IT'' where Dept=''CSE''

8) **Drop the view.**

Drop view Managers;

CONCLUSION:-

Here in this practical we studied about creating, updating, and deleting views in which we clear the idea about we cannot update a view which is created by join on two or more tables along with this we can also delete a view.

ASSESSMENT SCHEME:-

Pre-Lab Test (2)	In Lab performance (5)	Post Lab Test (3)	Record (5)	Total (15)

Signature of Faculty

PRACTICAL NO. 09

AIM OF PRACTICAL :

To Study and Implement Triggers in SQL

LEARNING OBJECTIVES: -

1. To study the concept and advantages of using Triggers in DBMS
2. To implement Triggers and understand how to use triggers

LEARNING OUTCOMES: -

1. Learn how to automate some actions in DBMS using triggers
2. Design create and Triggers in DBMS..

SOFTWARE REQUIRED :-

SN	Name of Equipment/ Items / Software Tool	Specification	Qty Required
1.	Hardware : Computer System	Computer (i3-i5 preferable) , RAM minimum 2GB onwards	As per batch size
2.	Operating System	Windows 7 or Later Version / LINUX version 5.0 or Later Version	
3	Software	MySQL / Oracle or SQL Server	

PROBLEM DEFINITION:

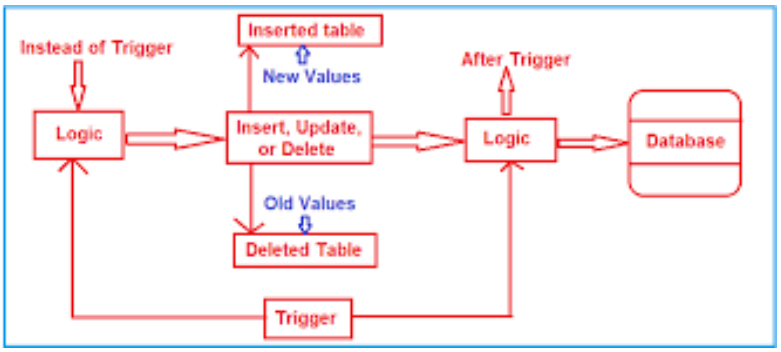
Create tables to store the information of product as productid, name, supplier name, unit price.

Create another table to store the history of the product similar to product table as productid, product name, supplier name, unit price, log date.

1) Create trigger to update to product history table when the price of product is updated in the product table.
2) Fire the trigger by executing respective queries.

BLOCK DIAGRAM:

Block diagram of Triggers:



THEORY:-

A trigger is a set of actions that are run automatically when a specified change operation (SQL INSERT, UPDATE, or DELETE statement) is performed on a specified table. Triggers are useful for tasks such as enforcing business rules, validating input data, and keeping an audit trail.

Uses for triggers:

- Enforce business rules
- Validate input data
- Generate a unique value for a newly-inserted row in a different file.
- Write to other files for audit trail purposes
- Query from other files for cross-referencing purposes
- Access system functions
- Replicate data to different files to achieve data consistency

Benefits of using triggers in business:

- Faster application development. Because the database stores triggers, you do not have to code the trigger actions into each database application.
- Global enforcement of business rules. Define a trigger once and then reuse it for any application that uses the database.
- Easier maintenance. If a business policy changes, you need to change only the corresponding trigger program instead of each application program.
- Improve performance in client/server environment. All rules run on the server before the result returns.

Implementation of SQL triggers is based on the SQL standard. It supports constructs that are common to most programming languages. It supports the declaration of local variables, statements to control the flow of the procedure, assignment of expression results to variables, and error handling.

trigger is a named database object that is associated with a table, and it activates when a particular event (e.g. an insert, update or delete) occurs for the table. The statement CREATE TRIGGER creates a new trigger in MySQL. Here is the syntax :

Syntax:

```
CREATE

[DEFINER = { user | CURRENT_USER }]

TRIGGER trigger_name
trigger_time trigger_event
ON tbl_name FOR EACH ROW
trigger_body
trigger_time: { BEFORE | AFTER }
trigger_event: { INSERT | UPDATE | DELETE }
```

trigger_name: All triggers must have unique names within a schema. Triggers in different schemas can have the same name.

trigger_time: trigger_time is the trigger action time. It can be BEFORE or AFTER to indicate that the trigger activates before or after each row to be modified.

trigger_event: trigger_event indicates the kind of operation that activates the trigger. These trigger_event values are permitted:

- The trigger activates whenever a new row is inserted into the table; for example, through INSERT, LOAD DATA, and REPLACE statements.
- The trigger activates whenever a row is modified; for example, through UPDATE statements.
- The trigger activates whenever a row is deleted from the table; for example, through DELETE and REPLACE statements. DROP TABLE and TRUNCATE TABLE statements on the table do not activate this trigger, because they do not use DELETE. Dropping a partition does not activate DELETE triggers, either.

tbl_name : The trigger becomes associated with the table named tbl_name, which must refer to a permanent table. You cannot associate a trigger with a TEMPORARY table or a view.

trigger_body: trigger_body is the statement to execute when the trigger activates. To execute multiple statements, use the BEGIN ... END compound statement construct. This also enables you to use the same statements that are permissible within stored routines.

OUTPUT / RESULTS & ANALYSIS:

1) **Create tables to store the information of product as productid, name, supplier name, unit price.**
   Create table PRODUCT (pid int primary key, pname varchar(20), Supplier varchar(20), Unit Price numeric(3,2));
2) **Create another table to store the history of the product similar to product table as productid, product name, supplier name, unit price, log date.**

Create table HISTORY (pid int primary key, pname varchar(20), Supplier varchar(20), Unit Price numeric(3,2), logdate date);

3) Create trigger to update to product history table when the price of product is updated in the product table.

```
Delimiter /

CREATE TRIGGER mytrig BEFORE
UPDATE on Product FOR EACH
ROW
BEGIN

INSERT into HISTORY values( OLD.pid,
OLD.pname, OLD.supplier, OLD.UnitPrice,
SYSDATE( ));
END

/
```

4) Fire the trigger by executing respective queries.

```
Update product set unitprice = 55.00 where
pname="halkefulke"; Select * from HISTORY;
```

CONCLUSION:-

In this practical we have studied the triggers which are executed when insert, update and delete operations are performed. A trigger is like a stored procedure which invokes automatically whenever specified event occurs. Also in this practical we come to know that triggers are a part of PI/SQL which also supports various conditional statements.

ASSESSMENT SCHEME:-

Pre-Lab Test (2)	In Lab performance (5)	Post Lab Test (3)	Record (5)	Total (15)

Signature of Faculty

Practical No 10

Aim: To study authorization in SQL

- Authorization is finding out if the person, once identified, is permitted to have the resource.
- Authorization explains that what you can do and is handled through the DBMS unless external security procedures are available.
- Database management system allows DBA to give different access rights to the users as per their requirements.
- Basic Authorization we can use any one form or combination of the following basic forms of authorizations

  i. Resource authorization:-Authorization to access any system resource. e.g. sharing of database, printer etc.

  ii. Alternation Authorization:- Authorization to add attributes or delete attributes from relations

  iii. Drop Authorization:-Authorization to drop a relation.

- **Granting of privileges:**

  i. A system privilege is the right to perform a particular action,or to perform an action on any schema objects of a particular type.

  ii. An authorized user may pass on this authorization to other users.This process is called as ganting of privileges.

  iii. Syntax:

```
GRANT <privilege list>
ON<relation name or view name>
TO<user/role list>
```

  iv. Example:

  The following grant statement grants user U1,U2 and U3 the select privilege on Emp_Salary relation:

```
GRANT  select
ON Emp_Salary
TO U1,U2 and U3.
```

- **Revoking of privileges:**

  i. We can reject the privileges given to particular user with help of revoke statement.

  ii. To revoke an authorization, we use the revoke statement.

  iii. Syntax:

```
REVOKE <privilege list>
ON<relation name or view name>
FROM <user/role list>[restrict/cascade]
```

  iv. Example:

  The revocation of privileges from user or role may cause other user or roles also have to loose that privileges.This behavior is called cascading of the revoke.

```
Revoke select
ON  Emp_Salary
FROM U1,U2,U3.
```

- Some other types of Privileges:

  i. **Reference privileges:**

  SQL permits a user to declare foreign keys while creating relations.

  Example: Allow user U1 to create relation that references key 'Eid' of Emp_Salary relation.

```
GRANT REFERENCES(Eid)
ON Emp_Salary
TO U1
```

  ii. **Execute privileges:**

  This privileges authorizes a user to execute a function or procedure.

  Thus,only user who has execute privilege on a function Create_Acc() can call function.

```
GRANT EXECUTE
ON Create_Acc
```

TO U1.