Vidya Vikas Education Trust's
Universal College of Engineering, Kaman Road, Vasai-401212
Accredited by B+ Grade by NAAC

LAB MANUAL OF
Database Management System
Course Code: CSL-402

Class: SE Computer Engineering Semester: IV(CBCGS)
Name: **Vivek Shivakumar Hotti**
Roll No: **31**

Mr. Ashraf Siddiqui
Subject In charge

Dr. Jitendra Saturwar
Head of Department

Department of Computer Engineering

Vision:

To be recognized globally as a department provides quality technical education that eventually caters to helping and serving the community

Mission:

To develop human resources with sound knowledge in theory and practice of computer science and engineering

To motivate the students to solve real-world problems to help the society grow

To provide a learning ambience to enhance innovations, team spirit and leadership qualities for students

Lab Code	Lab Name	Credits
CSL-402	**Database Management System Lab**	**1**

Lab Objectives:

1. To explore design and development of relational model.
2. To present SQL and procedural interface to SQL comprehensively.
3. To introduce the concept of transaction and transaction processing.

Lab Outcomes:

1. Design ER/EER diagram and convert to relational model for the real world application.
2. Apply DDL, DML, DCL and TCL commands.
3. Write simple and complex queries.
4. USePL / SQL constructor.
5. Demonstrate the concept of concurrent transactions execution and frontend-backend connectivity.

Term Work:

1. Term work should consist of at least 10 experiments.

2. Journal must include at least 2 assignments on content of theory and practical of "Database Management System"

3. The final certification and acceptance of term work ensures that satisfactory performance of laboratory work and minimum passing marks in term work.

4. Total 25 Marks (Experiments: 15-marks, Attendance Theory & Practical: 05-marks, Assignments: 05-marks)

Oral & Practical exam

1. Experiments----------(15)Marks.
2. Attendance Theory & Practical----------(05)Marks
3. Assignment----------(05)Marks

Total Marks----------(25)Marks

List of Experiments

SR. No	Title of Experiments
1	Identify the case study and detail statement of problem. Design an Entity-Relationship (ER) / Extended Entity-Relationship (EER) Model.
2	Mapping ER/EER to Relational schema model.
3	Create a database using Data Definition Language (DDL) and apply integrity constraints for the specified System.
4	Apply DML Commands for the specified system.
5	Perform DCL and TCL commands.
6	Implement various Join operations.
7	Perform Simple queries, string manipulation operations and aggregate functions.
8	Implementation of Views and Triggers.
9	Demonstrate Database connectivity.
10	Implementation and demonstration of Transaction and Concurrency control techniques using locks.

Experiment No 1: Design an Entity-Relationship (ER) / Extended Entity-Relationship (EER) Model.

AIM: - Design an Entity-Relationship (ER) / Extended Entity-Relationship (EER) Model..
Identify the case study and detailed statement of the problem. Design an Entity-Relationship (ER)/ Extended Entity-Relationship (EER) Model.

THEORY: -

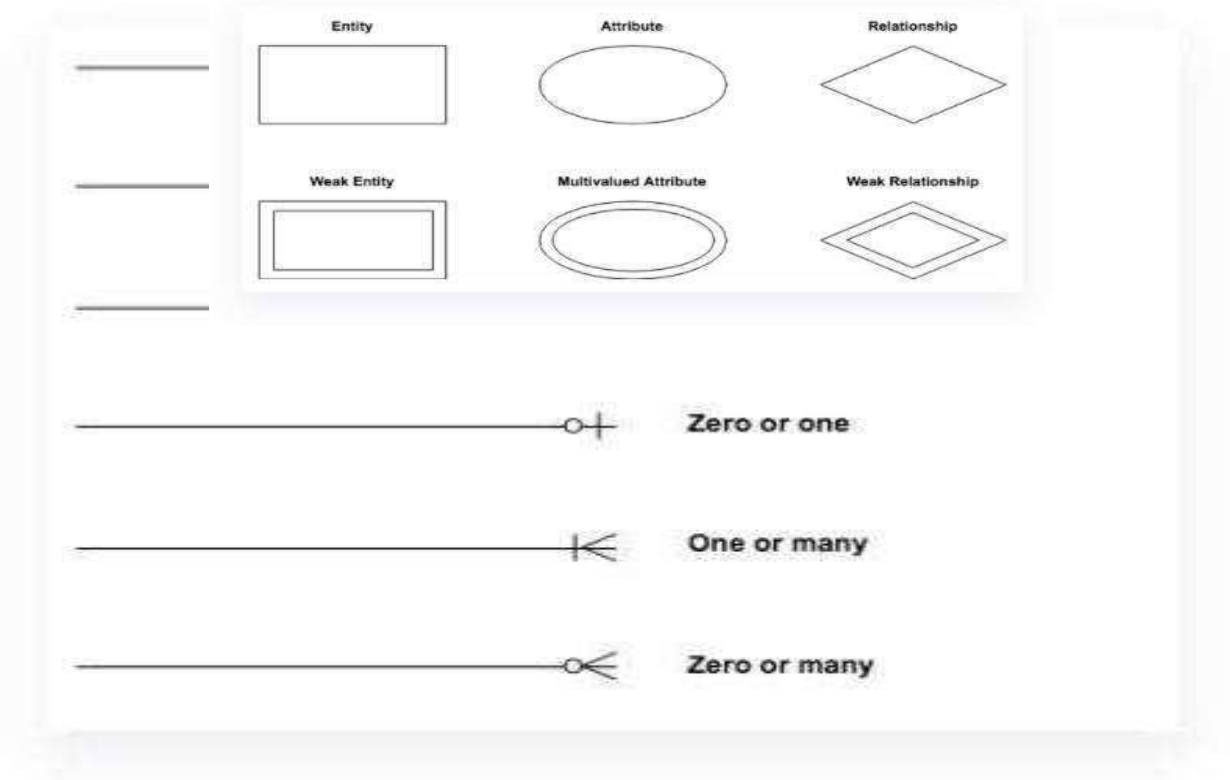
An entity-relationship (ER) diagram, also called an entity-relationship model, is aptly named: It shows the relationships between entities. It is most commonly used to organize data within databases or information systems.

There are two kinds of ER diagrams: Conceptual and Physical. Conceptual diagram models can be used as the foundation for logical data models or to form commonality relationships between ER models as a basis for data-model

A conceptual ER diagram uses six standard symbols. They are:

1) Entities are objects or concepts that represent important data. Also known as strong entities or parent entities, these entities will often have weak entities that depend on them.
2) Attributes are characteristics of an entity, i.e. many-to-many or one-to-one.
3) Relationships are associations between entities.
4) Weak entities depend on another entity.
5) Multivalued attributes are attributes that can have more than one value.
6) Weak relationships are the connections between a weak entity and its parent.

Physical diagram models are more granular, showing the processes necessary to add information to a database. Rather than using symbols, they are made up of a series of tables. Entities are connected using a system of notation called crow's foot notation. The styling of the



endpoint of each line distinguishes the relationship.

Enhanced entity-relationship (EER) diagrams are basically an expanded upon version of ER diagrams. EER models are helpful tools for designing databases with high-level models. With their enhanced features, you can plan databases more thoroughly by delving into the properties and constraints with more precision.

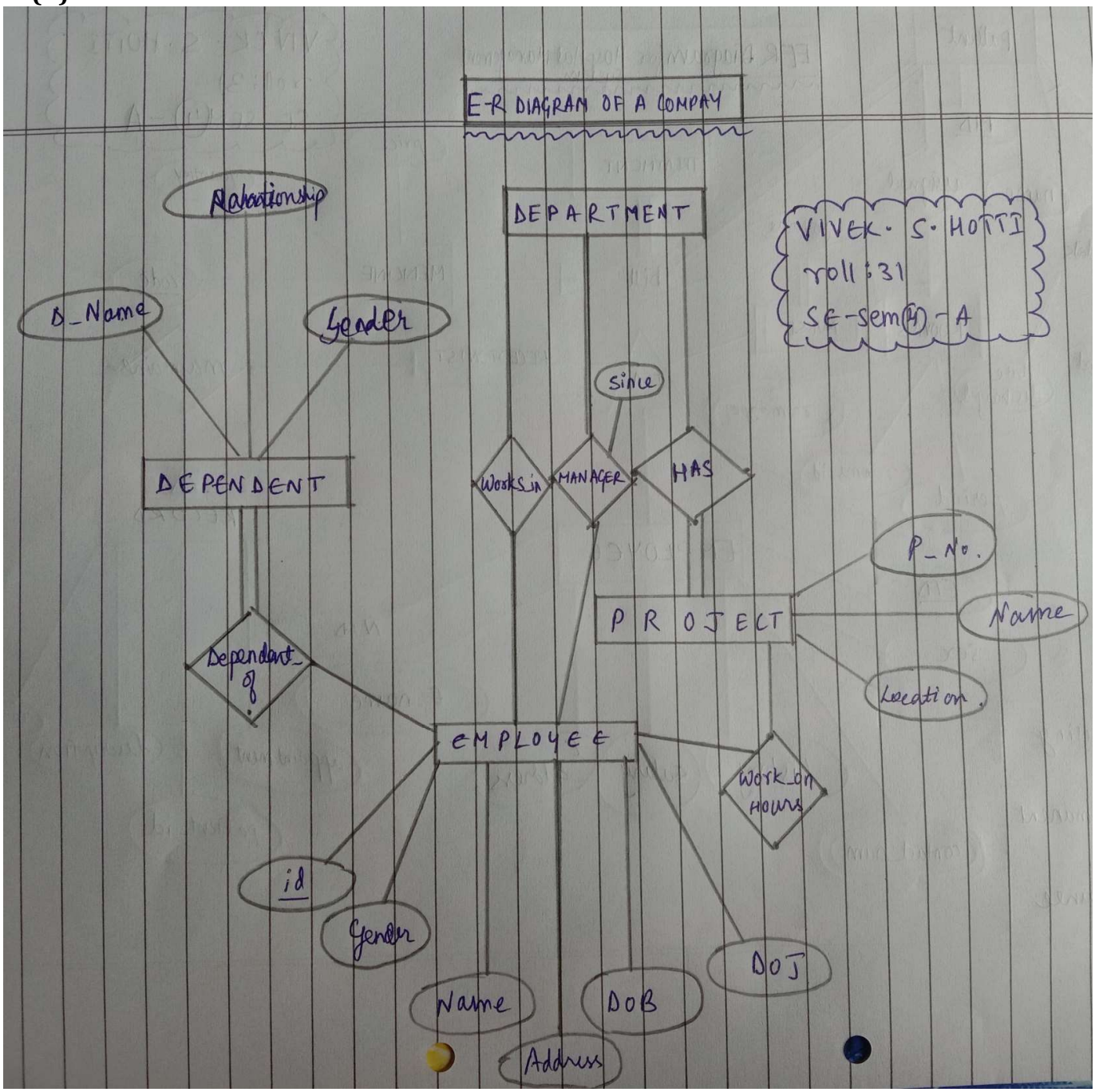
An EER diagram provides you with all the elements of an ER diagram while adding:

- Attribute or relationship inheritances
- Category or union types
- Specialization and generalization
- Subclasses and super classes

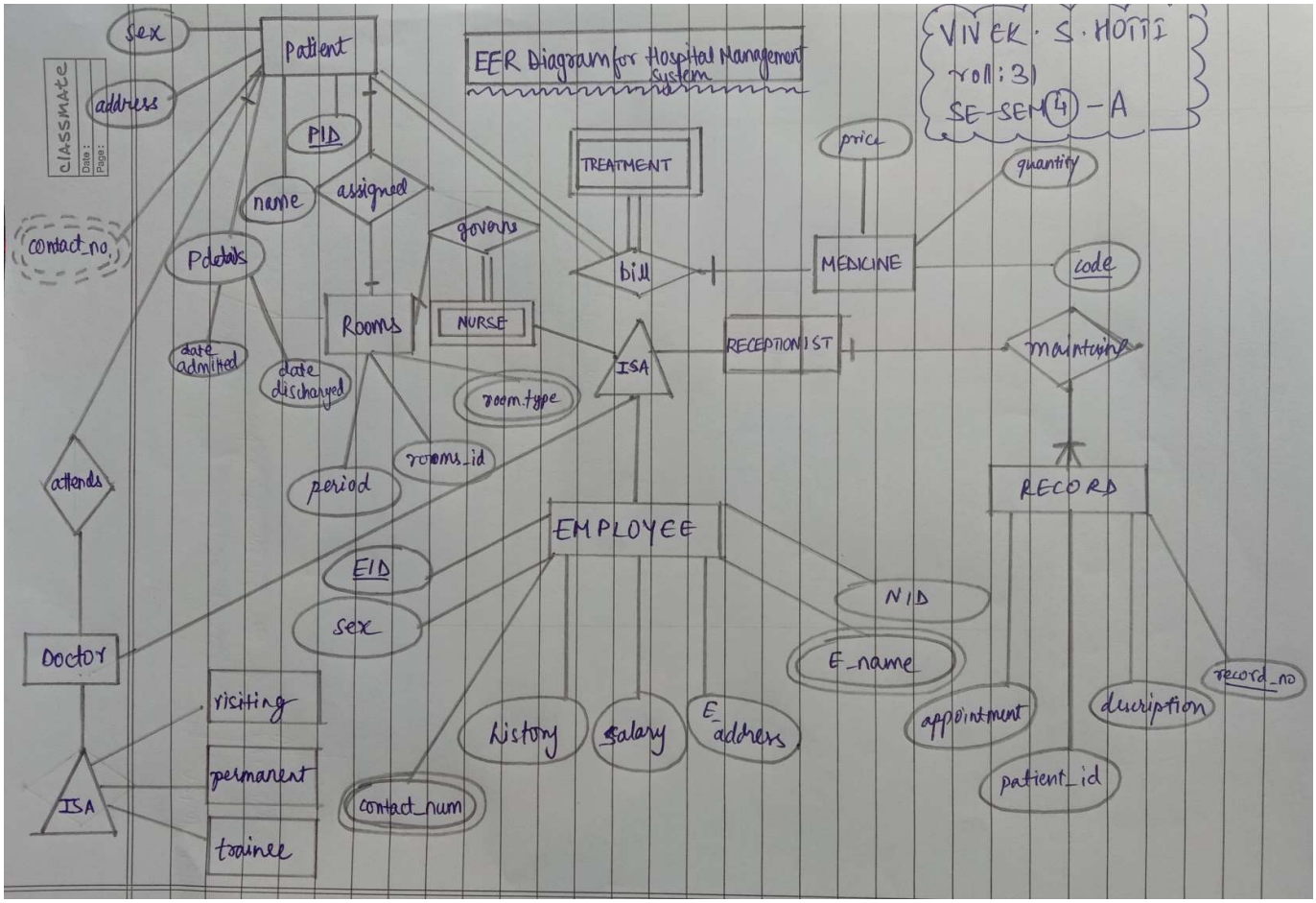
Overall, an EER diagram builds off of an ER diagram by including elements that allow for aggregating, generalizing, and specializing.

DIAGRAM / OUTPUT: -

(A)



(B)



CONCLUSION: - Thus we have Identify the Entity-Relationship (ER) / Extended Entity-Relationship (EER) Mode.

Experiment No 2: Mapping ER/EER to Relational Schema Model.

AIM: - Mapping ER/EER to Relational schema model.

THEORY: -

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

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

ER diagrams mainly comprise of –

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

Mapping Process:

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

Mapping Entity: -

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

Mapping Process:

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

(A) Mapping Relationship: -

A relationship is an association among entities.
Mapping Process:

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

(B) *Mapping Weak Entity Sets:* -

A relationship is an association among entities.
Mapping Process:

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

(C) *Mapping Hierarchical Entities:* -

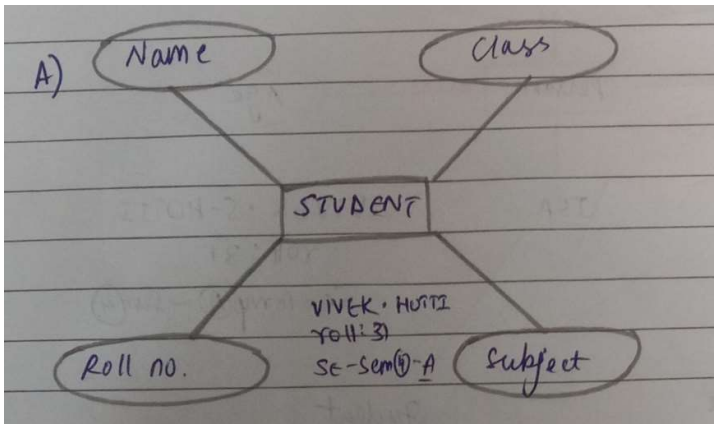
A relationship is an association among entities.

Mapping Process:

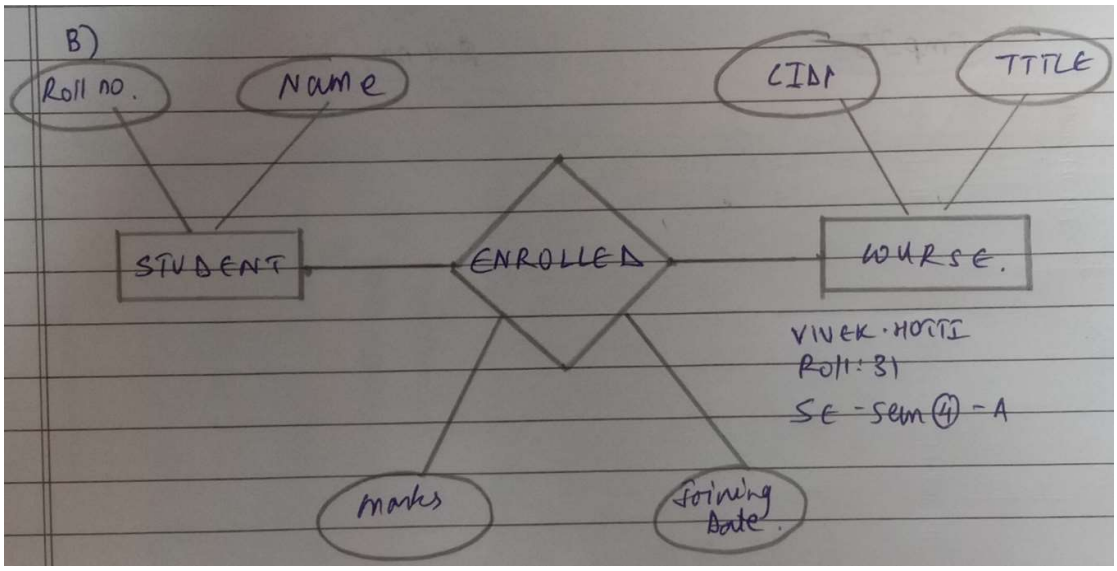
- Create tables for all higher-level entities.
- Create tables for lower-level entities.
- Add primary keys of higher-level entities in the table of lower-level entities.
- In lower-level tables, add all other attributes of lower-level entities.
- Declare primary key of higher-level table and the primary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

DIAGRAM / OUTPUT: -

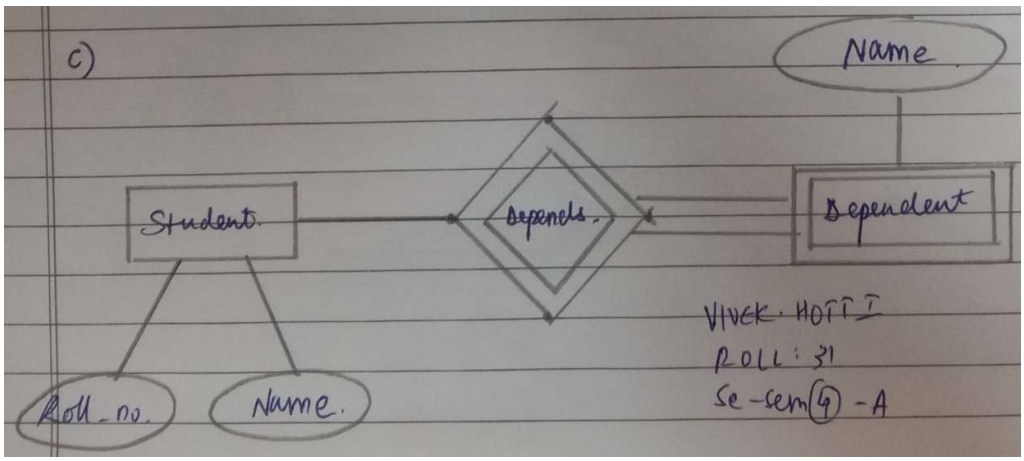
(A)



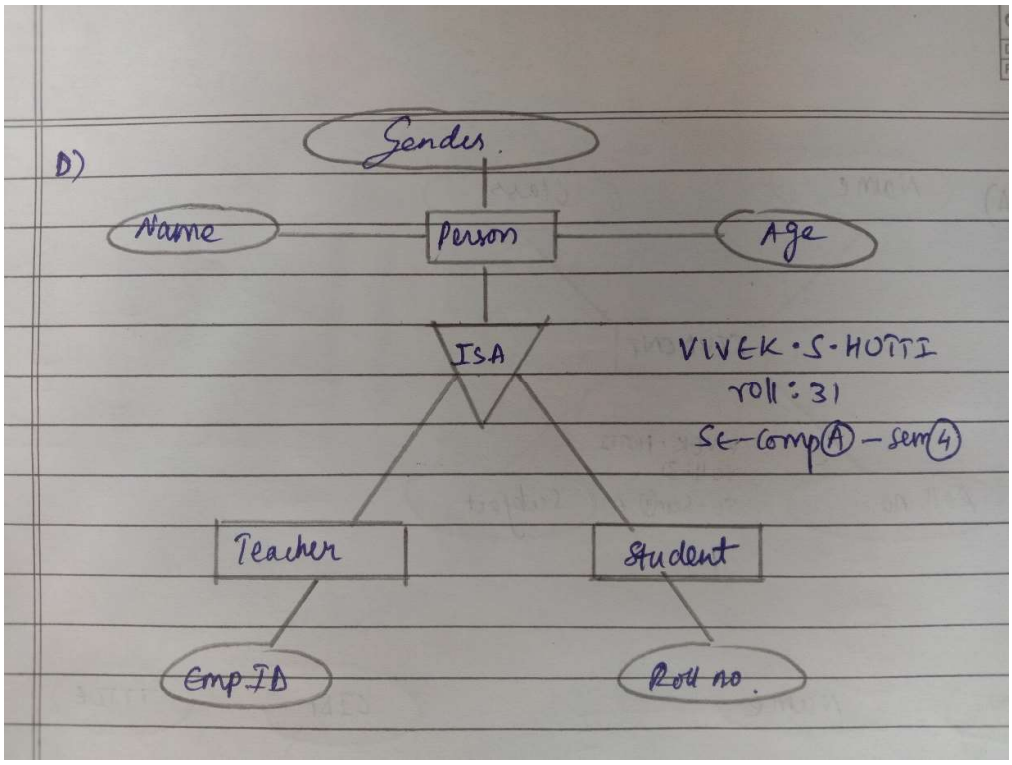
(B)



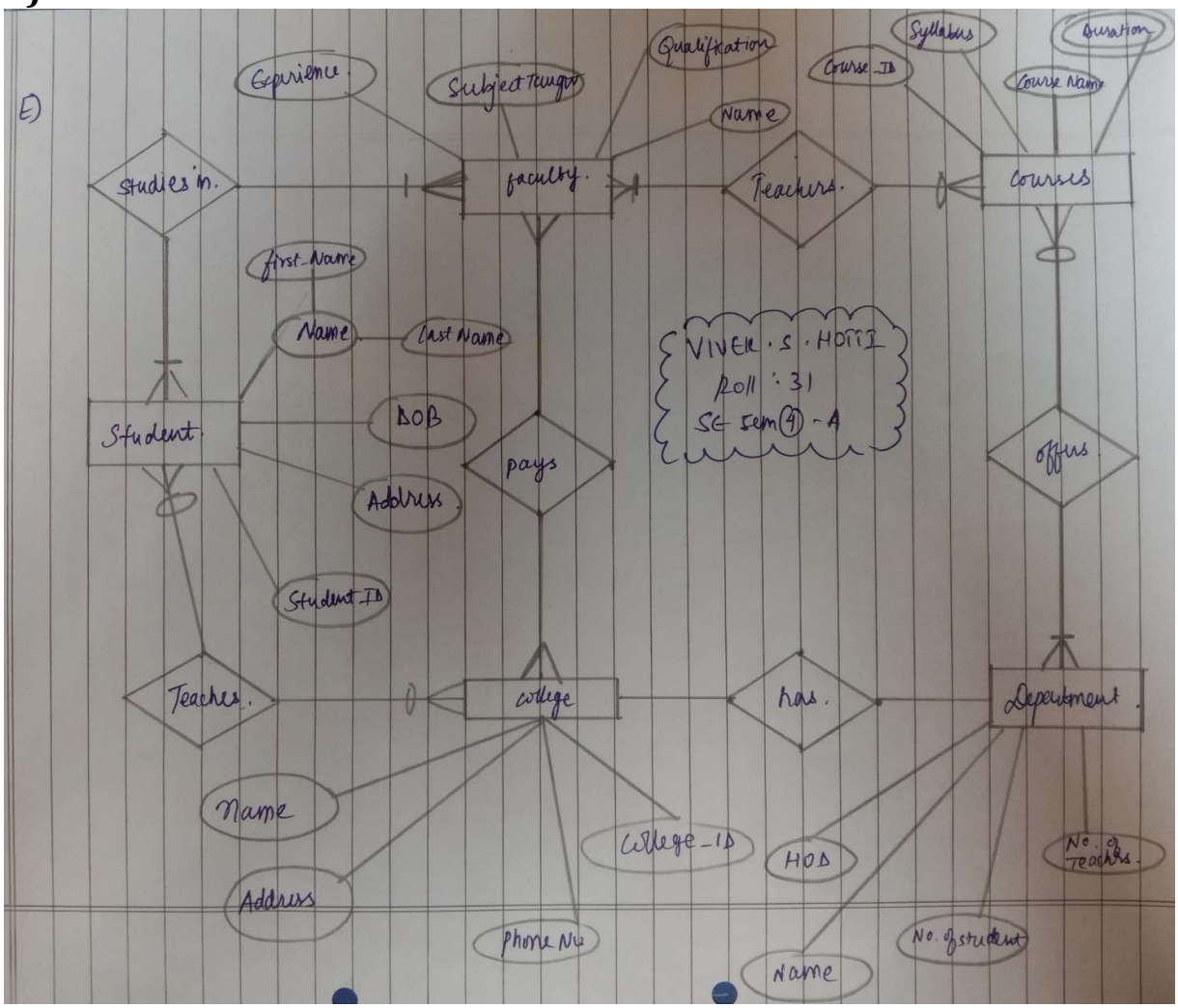
(C)


C)
Name
Student
Depends
Dependent
Roll_no.
Name
VIVEK · HOTTI
ROLL : 31
Se - sem (4) - A

(D)


D)
Gender
Name
Person
Age
ISA
VIVEK · S · HOTTI
roll : 31
Se - comp (A) - sem (4)
Teacher
Student
Emp ID
Roll no.

E)



CONCLUSION: -

Thus, we have Identified the Mapping ER/EER to Relational schema model.

Experiment No 3: Usage of Data Definition Language (DDL) to create databases and apply integrity constraints for the specified System.

AIM: - Create a database using Data Definition Language (DDL) and apply integrity constraints for the specified System.

REQUIREMENTS: - Relation Schema, MySQL Editor.

THEORY: -

DDL is a set of guidelines to which all the Structured Query Languages adhere to. As in the computer programming languages, we have the OOPS guidelines that all the programming languages adhere to, similarly, we have Data Definition Language standards that all the database languages adhere to – MySQL, Oracle, SQL Server, etc**.**

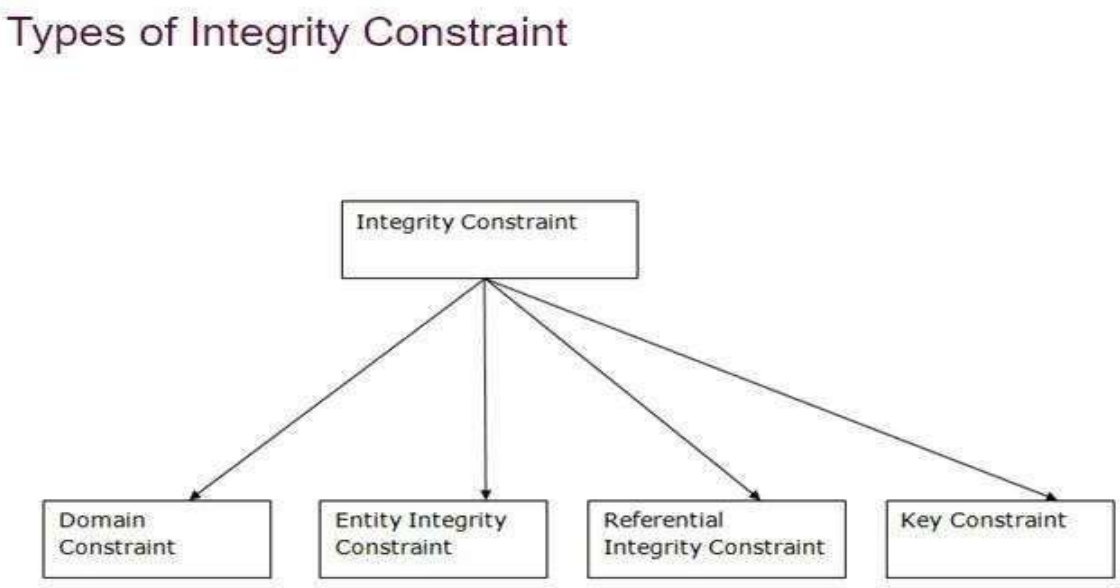
Data Definition Language deals with the structure of the database where the data is to be stored. It does not deal with the data itself. Thus, in any structured query language, a command that can modify the structure or the tables or relations of the database, is a DDL command. A command that modifies the data stored is a DML command. A command that modifies the authorization rules is a DCL command. A command that queries the database to fetch results is a DQL command.

Examples of DDL commands:

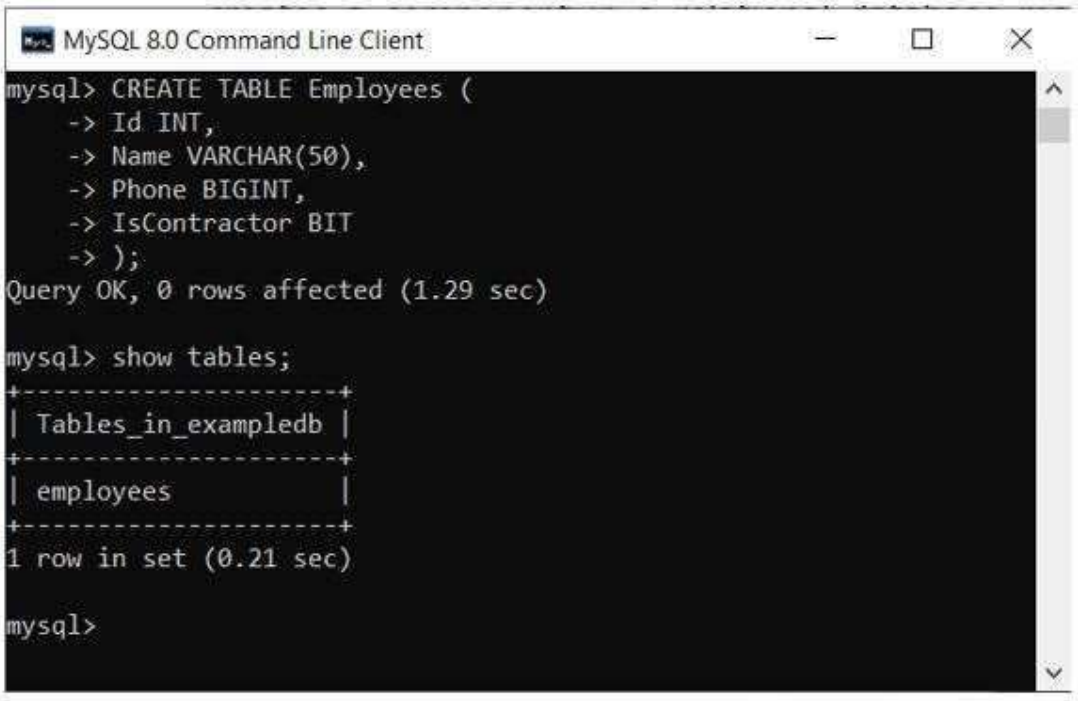
1) **CREATE** – is used to create the database or its objects (like table, index, function, views, store procedure and triggers).
2) **DROP** – is used to delete objects from the database.
3) **ALTER**-is used to alter the structure of the database.
4) **TRUNCATE**–is used to remove all records from a table, including all spaces allocated for the records are removed.
5) **COMMENT** –is used to add comments to the data dictionary.
6) **RENAME** –is used to rename an object existing in the database.

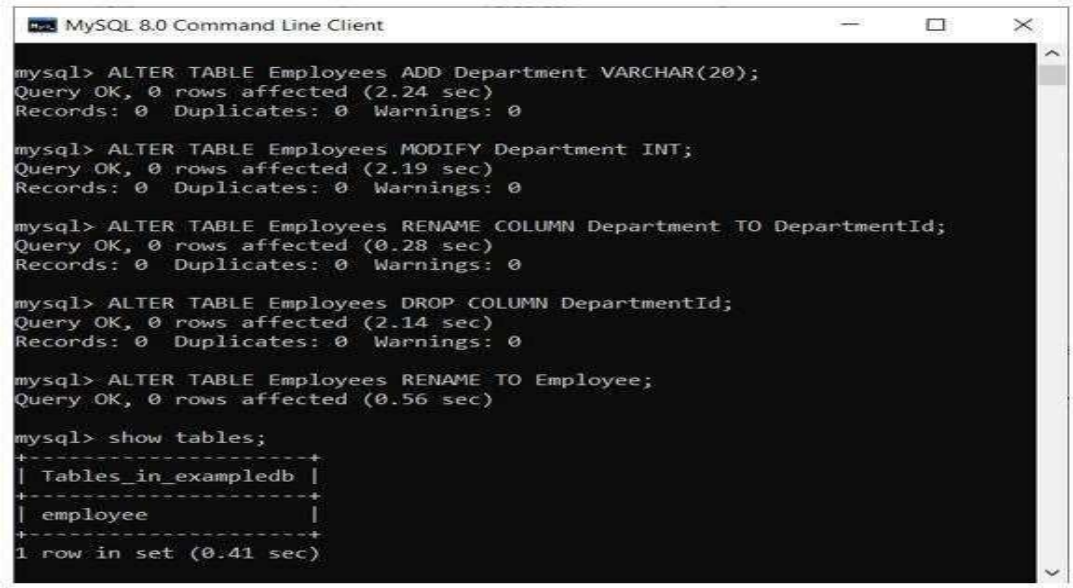
Integrity Constraints: -

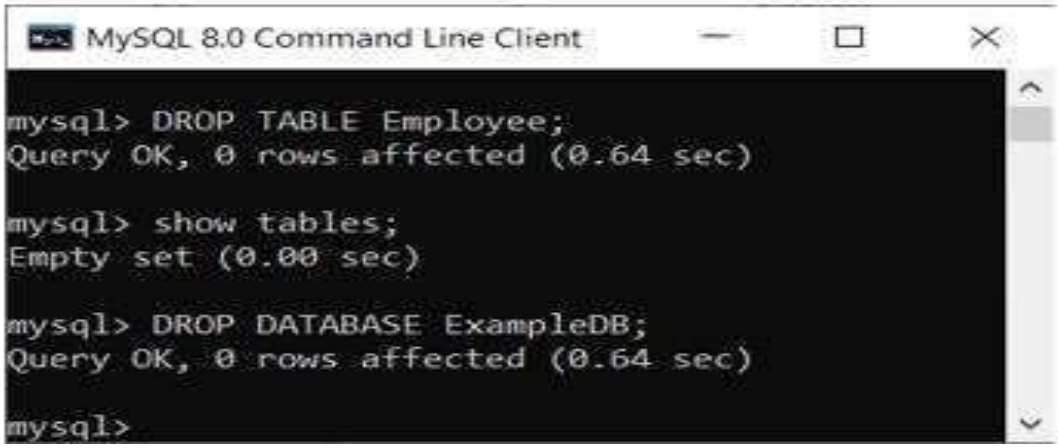
- Integrity constraints are a set of rules. It is used to maintain the quality of information.
- Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.
- Thus, integrity constraint is used to guard against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



OUTPUT: -


MySQL 8.0 Command Line Client
mysql> CREATE TABLE Employees (
-> Id INT,
-> Name VARCHAR(50),
-> Phone BIGINT,
-> IsContractor BIT
->);
Query OK, 0 rows affected (1.29 sec)
mysql> show tables;
+----------------------+
| Tables_in_exampledb |
+----------------------+
| employees |
+----------------------+
1 row in set (0.21 sec)
mysql>





CONCLUSION:. Thus We have created a database using Data Definition Language (DDL) and apply integrity constraints for the specified System.

Experiment No 4: - Apply DML Commands for the specified system.

AIM: - To Apply DML Commands for the specified system.

REQUIREMENTS: - Relational Schema, Web Browser- Google Chrome, MySQL Editor v1.6, W3Schools.

THEORY: -

Data Manipulation Language (DML) allows you to modify the

database instance by inserting, modifying, and deleting its data. It is

responsible for performing all types of data modification in a

database.

There are three basic constructs which allow database program and

user to enter data and information are:

Here are some important DML commands in SQL:

- INSERT
- UPDATE
- DELETE

(A) INSERT: -

This is a SQL query. This command is used to insert data into the row of a table.

Syntax:

INSERT INTO TABLE_NAME (col1, col2, col3,.... col N)

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

Or

INSERT INTO TABLE_NAME

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

(B) UPDATE: -

This command is used to update or modify the value of a column in the table.

Syntax:

UPDATE table_name SET [column_name1= value1,...column_nameN = valueN]
[WHERE CONDITION]

(C) DELETE: -

This command is used to remove one or more rows from a table.

Syntax:

DELETE FROM table_name [WHERE condition];

OUTPUT: -

(A)

INSERT INTO students (RollNo, FIrstName, LastName) VALUES ('60', 'Tom',
Erichsen');

(B)

```
UPDATE students
SET FirstName = 'John', LastName= 'Wick'
WHERE StudID = 3;
```

(C)

```
DELETE FROM students
WHERE FirstName = 'John';
```

CONCLUSION: - Thus we have Apply DML Commands for the specified system.

Experiment No 5: Perform DCL and TCL commands.

AIM: - To Perform DCL and TCL Commands.

REQUIREMENTS: - Relational Schema, Web Browser- Google Chrome, MySQL Editor v1.6, W3Schools

THEORY: -

->DCL a Data Control Language: -

Its commands are responsible for access restrictions inside of the database.

Let's take a look at DCL statements definitions.

GRANT: -GRANT command gives permissions to SQL user account.
For example, I want to grant all privileges to 'explain java' database for user " 'dmytro@localhost'.
Let's create a user first:
CREATE USER 'dmytro'@'localhost' IDENTIFIED BY '123';

Then we can grant all privileges using GRANT statement:
GRANT ALL PRIVILEGES ON explain java. * TO 'dmytro'@'localhost';

and we have to save changes using FLUSH command:
FLUSH PRIVILEGES;

REVOKE: - REVOKE statement is used to remove privileges from user accounts.
Example/Output:
REVOKE ALL PRIVILEGES ON explain java.* FROM 'dmytro'@'localhost';

and save changes:
FLUSH PRIVILEGES;
->TCL is a Transaction Control Language: -

Its commands are used to manage transactions in SQL databases.
This is TCL commands list:
START TRANSACTION (BEGIN, BEGIN WORK): -
START TRANSACTION is used to start a new SQL transaction.
BEGIN and BEGIN WORK are aliases for START TRANSACTION.
Example/Output:
START TRANSACTION;

after that, you're doing manipulations with a data (insert, update, delete) and at the end, you need to commit a transaction.

COMMIT: -
As a mentioned above COMMIT command finishes transaction and stores all changes made inside of a transaction.
Example/Output:
START TRANSACTION;
INSERT INTO student (name, last name) VALUES ('Dmytro', 'Shvechikov');
COMMIT;

ROLLBACK: -
ROLLBACK statement reverts all changes made in the scope of transaction.
Example/Output:
START TRANSACTION;
INSERT INTO student (name, last name) VALUES ('Dmytro', 'Shvechikov');
ROLLBACK;

CONCLUSION: - Thus we have Implement DCL and TCL commands.

Experiment No 6: Implement Various Join Operations

AIM : - To Implement various Join operations.

REQUIREMENTS : -Relations Schema , MySQL editor

THEORY: -

Join in DBMS is a binary operation which allows you to combine join product and selection in one single statement. The goal of creating a join condition is that it helps you to combine the data from two or more DBMS tables. The tables in DBMS are associated using the primary key and foreign keys.

->TYPES OF JOINS: -

- **Inner Join**
  - Theta Join
  - EQUI join:
  - Natural Join (⋈)
- **Outer Join**
  - Left Outer Join (A ⟕ B)
  - Right Outer Join (A ⟖ B)
  - Full Outer Join (A ⟗ B)

INNER JOIN: The INNER JOIN keyword selects all rows from both the tables as long as the condition satisfies. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e., the value of the common field will be the same.

LEFT JOIN: This join returns all the rows of the table on the left side of the join and matching rows for the table on the right side of join. The rows for which there is no matching row on the right side, the result-set will contain *null*. LEFT JOIN is also known as LEFT OUTER JOIN.

RIGHT JOIN: RIGHT JOIN is similar to LEFT JOIN. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join. The rows for which there is no matching row on the left side, the result-set will contain *null*. RIGHT JOIN is also known as RIGHT OUTER JOIN.

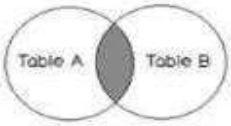
FULL JOIN: FULL JOIN creates the result-set by combining the result of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both the tables. The rows for which

there is no matching, the result-set will contain *NULL* values.

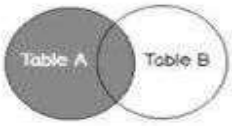
DIAGRAM / OUTPUT: -

These joins are explaining in the form of Venn diagram illustrations:

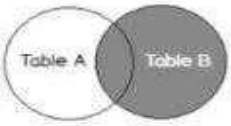
- **Inner join** returns the rows that match in both tables



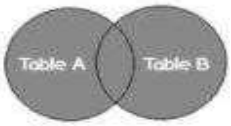
- **Left join** returns all rows from the left table



- **Right join** returns all rows from the right table



- **Full join** returns whole rows from both tables



<u>CONCLUSION</u>: - Thus we have Implement various Join operations.

Experiment No 7: Perform simple queries, string manipulation, operations and aggregate functions.

AIM: - To Perform Simple queries, string manipulation operations and aggregate functions.

REQUIREMENTS: - Relational Schema, Web Browser- Google Chrome, MySQL Editor v1.6, W3Schools

THEORY: -

SIMPLE QUERIES: -

➢ **SELECT and FROM**

The SELECT part of a query determines which columns of the data to show in the results. There are also options you can apply to show data that is not a table column.

The example below shows three columns SELECTed FROM the "student" table and one calculated column. The database stores the studentID, FirstName, and LastName of the student. We can combine the First and the Last name columns to create the FullName calculated column.

```
SELECT studentID, FirstName, LastName, FirstName + ' ' + LastName AS FullName
FROM student;
```

➢ CREATE TABLE

CREATE TABLE does just what it sounds like: it creates a table in the database. You can specify the name of the table and the columns that should be in the table.

```
CREATE TABLE table_name (
column_1 datatype,
column_2 datatype,
column_3 datatype
);
```

➢ ALTER TABLE

ALTER TABLE changes the structure of a table. Here is how you would add a column to a database:

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

➢ CHECK

The CHECK constraint is used to limit the value range that can be placed in a column.

If you define a CHECK constraint on a single column it allows only certain values for this column. If you define a CHECK constraint on a table it can limit the values in certain columns based on values in other columns in the row.

The following SQL creates a CHECK constraint on the "Age" column when the "Persons" table is created. The CHECK constraint ensures that you cannot have any person below 18 years.

```
CREATE TABLE Persons (
ID int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Age int,
CHECK (Age>=18)
);
```

SQL manipulation operations are primarily utilized for string manipulation. The built-in SQL String functions make it easier for us to find and alter string values. Cutting blanks off a string value for display. You can use LEN function to find the length of a string. It takes a single parameter containing a string expression, concatenating two strings. You can find the given word from the sentence, even you can substring the character up to the given point in the string. You can find a word from the given point and of the given length using the MID function. You can also find the nth position of the given word in a string.

OUTPUT: -

```
+-----------+------------------+------------+-----------------------+
| studentID | FirstName | LastName | FullName           |
+-----------+------------------+------------+-----------------------+
| 1 | Monique | Davis | Monique Davis                      |
| 2 | Teri | Gutierrez | Teri Gutierrez                       |
| 3 | Spencer | Pautier | Spencer Pautier                    |
| 4 | Louis | Ramsey | Louis Ramsey                        |
| 5 | Alvin | Greene | Alvin Greene                         |
|                                                          |
+-----------+------------------+------------+-----------------------+
9 rows in set (0.00 sec)

+-----------+-----------------------+-----------+---------------------+
| studentID | FullName | sat_score | rcd_updated          |
+-----------+-----------------------+-----------+---------------------+
| 1 | Monique Davis | 400 | 2017-08-16 15:34:50           |
| 2 | Teri Gutierrez | 800 | 2017-08-16 15:34:50            |
| 3 | Spencer Pautier | 1000 | 2017-08-16 15:34:50        |
| 4 | Louis Ramsey | 1200 | 2017-08-16 15:34:50          |
| 5 | Alvin Greene | 1200 | 2017-08-16 15:34:50            |
|                                                              |
+-----------+-----------------------+-----------+---------------------+
9 rows in set (0.00 sec)

+-----------+----------------------+-----------+---------------------+
| studentID | FullName | sat_score | rcd_updated        |
+-----------+----------------------+-----------+---------------------+
| 1 | Monique Davis | 400 | 2017-08-16 15:34:50          |
| 2 | Teri Gutierrez | 800 | 2017-08-16 15:34:50           |
| 4 | Louis Ramsey | 1200 | 2017-08-16 15:34:50         |
| 5 | Alvin Greene | 1200 | 2017-08-16 15:34:50          |
|                                                             |
+-----------+----------------------+-----------+---------------------+
5 rows in set (0.00 sec)
```

CONCLUSION: - Thus we have Perform Simple queries, string manipulation operations and aggregate functions.

Experiment No 8: Implementation of Views and Triggers.

AIM: - To Implementation of Views and Triggers.

REQUIREMENTS: -
Relational Schema, Web Browser- Google Chrome, MySQL Editor v1.6, W3Schools

THEORY: -

VIEWS: -

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 depends on the written SQL query to create a view. Views, which are a type 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 in such a way 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.

SQL CREATE VIEW Statement:
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;
```

Then, we can query the view as follows:
```
SELECT *FROM [Current Product List];
```

TRIGGERS: -

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.

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
- Imposing security authorizations
- Preventing invalid transactions

OUTPUT: -

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00

```
SQL > CREATE VIEW CUSTOMERS_VIEW AS
SELECT name, age
FROM  CUSTOMERS;
```

```
SQL > SELECT * FROM CUSTOMERS_VIEW;
```

name	age
Ramesh	32
Khilan	25
kaushik	23
Chaitali	25
Hardik	27
Komal	22
Muffy	24

CONCLUSION: - Thus we have Implement SQL commands to perform of Views and Triggers.

Experiment No 9 : - Demonstrate Database Connectivity.

AIM: - Demonstrate Database Connectivity.

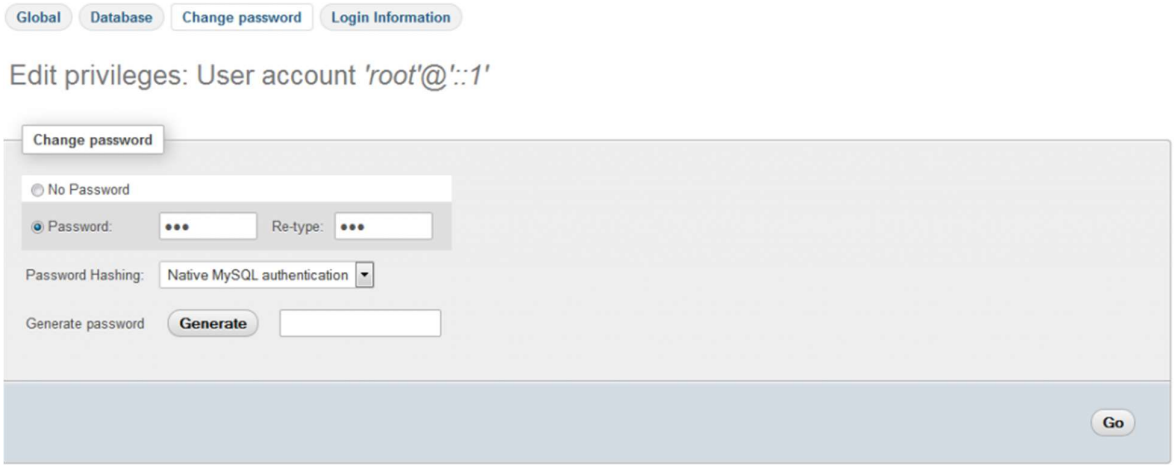
REQUIREMENTS: -
Relational Schema, Web Browser- Google Chrome, MySQL Editor v1.6, W3Schools

THEORY: -
Create MySQL Database at the Localhost :-

First, let me tell you what PHPMyAdmin is. It's a control panel from where you can manage the database that you've created. Open your browser and go to localhost/PHPMyAdmin or click "Admin" in XAMPP UI.

When you first installed XAMPP, it only created the username for it to be accessed, you now have to add a password to it by yourself. For this, you have to go to User account where the user is the same as the one shown in this picture:

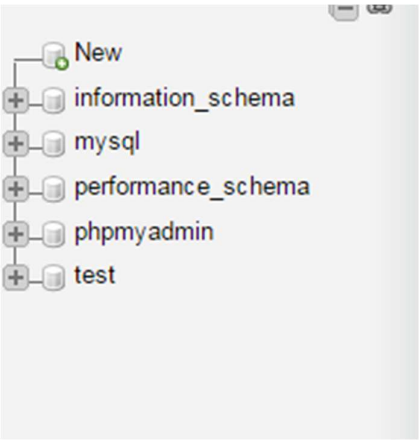
Now click *Edit privileges* and go to Change Admin password, type your password there and save it. Remember this password as it will be used to connect to your Database.



Note: It is not necessary to change the password to access databases on the localhost. It is a good practice and that is why we have used a password.

Create Database

Now return to the homepage of PHPMyAdmin. Click the *New* button to create a new database.



In the new window, name your database as per your need, I am naming it "**practice**". Now select Collation as utf8_general_ci, as we are using it for learning purposes and it will handle all of our queries and data that will be covered in this tutorial series. Now click on *Create* and your database will be created.

The newly created database will be empty now, as there are no tables in it. I will be covering that in the upcoming series where we will learn how to create tables and insert data in it. In this tutorial, we are going to connect this database to localhost using PHP.

Create a Folder in htdocs

Now, locate the folder where you installed XAMPP and open the htdocs folder (usually c:/xampp). Create a new folder inside c:/xampp/htdocs/ and name it "practice" we will place web files in this

folder. Why we have created a folder in htdocs? XAMPP uses folders in htdocs to execute and run your PHP sites.

Note: If you are using WAMP, then add your practice folder in c:/wamp/www folder.

Create Database Connection File In PHP

Create a new PHP file and name it *db_connnection.php* and save it. Why am I creating a separate database connection file? Because if you have created multiple files in which you want to insert data or select data from the databases, you don't need to write the code for database connection every time. You just have to include it by using PHP custom function ***include*** (include 'connection.php') on the top of your code and call its function and use it. It also helps when you are moving your project location from one PC to another and you have to change the values on the single file and all the changes will be applied to all the other files automatically. Write the following code in your dB connection file.

CONCLUSION: - Thus we have Demonstrate Database Connectivity.

Experiment No 10: - Implementation and demonstration of Transaction and Concurrency control techniques using locks.

AIM: - Demonstrate Database Connectivity.

REQUIREMENTS: -
Relational Schema, Web Browser- Google Chrome, MySQL Editor v1.6, W3Schools

THEORY: -

Concurrency Control in Database Management System is a procedure of managing simultaneous operations without conflicting with each other. It ensures that Database transactions are performed concurrently and accurately to produce correct results without violating data integrity of the respective Database.

Concurrent access is quite easy if all users are just reading data. There is no way they can interfere with one another. Though for any practical Database, it would have a mix of READ and WRITE operations and hence the concurrency is a challenge.

DBMS Concurrency Control is used to address such conflicts, which mostly occur with a multi-user system. Therefore, Concurrency Control is the most important element for proper functioning of a Database Management System where two or more database transactions are executed simultaneously, which require access to the same data.

Reasons for using Concurrency control method is DBMS:

- To apply Isolation through mutual exclusion between conflicting transactions
- To resolve read-write and write-write conflict issues
- To preserve database consistency through constantly preserving execution obstructions
- The system needs to control the interaction among the concurrent transactions. This control is achieved using concurrent-control schemes.
- Concurrency control helps to ensure serializability

Concurrency Control Protocols

Different concurrency control protocols offer different benefits between the amount of concurrency they allow and the amount of overhead that they impose. Following are the Concurrency Control techniques in DBMS:

- Lock-Based Protocols
- Two Phase Locking Protocol
- Timestamp-Based Protocols
- Validation-Based Protocols

Lock-based Protocols

Lock Based Protocols in DBMS is a mechanism in which a transaction cannot Read or Write the data until it acquires an appropriate lock. Lock based protocols help to eliminate the concurrency problem in DBMS for simultaneous transactions by locking or isolating a particular transaction to a single user.

A lock is a data variable which is associated with a data item. This lock signifies that operations that can be performed on the data item. Locks in DBMS help synchronize access to the database items by concurrent transactions.

All lock requests are made to the concurrency-control manager. Transactions proceed only once the lock request is granted.

Binary Locks: A Binary lock on a data item can either locked or unlocked states.

Shared/exclusive: This type of locking mechanism separates the locks in DBMS based on their uses. If a lock is acquired on a data item to perform a write operation, it is called an exclusive lock.

1. Shared Lock (S):

A shared lock is also called a Read-only lock. With the shared lock, the data item can be shared between transactions. This is because you will never have permission to update data on the data item.

For example, consider a case where two transactions are reading the account balance of a person. The database will let them read by placing a shared lock. However, if another transaction wants to update that account's balance, shared lock prevent it until the reading process is over.

2. Exclusive Lock (X):

With the Exclusive Lock, a data item can be read as well as written. This is exclusive and can't be held concurrently on the same data item. X-lock is requested using lock-x instruction. Transactions may unlock the data item after finishing the 'write' operation.

For example, when a transaction needs to update the account balance of a person. You can allows this transaction by placing X lock on it. Therefore, when the second transaction wants to read or write, exclusive lock prevent this operation.

3. Simplistic Lock Protocol

This type of lock-based protocols allows transactions to obtain a lock on every object before beginning operation. Transactions may unlock the data item after finishing the 'write' operation.

4. Pre-claiming Locking

Pre-claiming lock protocol helps to evaluate operations and create a list of required data items which are needed to initiate an execution process. In the situation when all locks are granted, the transaction executes. After that, all locks release when all of its operations are over.

Starvation

Starvation is the situation when a transaction needs to wait for an indefinite period to acquire a lock.

Following are the reasons for Starvation:

- When waiting scheme for locked items is not properly managed
- In the case of resource leak
- The same transaction is selected as a victim repeatedly

Deadlock

Deadlock refers to a specific situation where two or more processes are waiting for each other to release a resource or more than two processes are waiting for the resource in a circular chain.

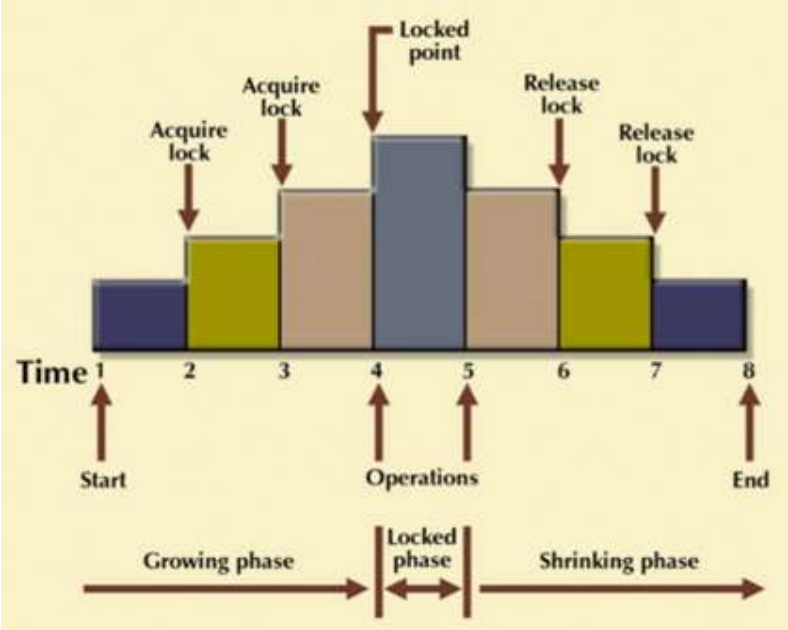
Two Phase Locking Protocol

Two Phase Locking Protocol also known as 2PL protocol is a method of concurrency control in DBMS that ensures serializability by applying a lock to the transaction data which blocks other transactions to access the same data simultaneously. Two Phase Locking protocol helps to eliminate the concurrency problem in DBMS.

This locking protocol divides the execution phase of a transaction into three different parts.

- In the first phase, when the transaction begins to execute, it requires permission for the locks it needs.
- The second part is where the transaction obtains all the locks. When a transaction releases its first lock, the third phase starts.

- In this third phase, the transaction cannot demand any new locks. Instead, it only releases the acquired locks.



The Two-Phase Locking protocol allows each transaction to make a lock or unlock request in two steps:

- **Growing Phase**: In this phase transaction may obtain locks but may not release any locks.
- **Shrinking Phase**: In this phase, a transaction may release locks but not obtain any new lock

It is true that the 2PL protocol offers serializability. However, it does not ensure that deadlocks do not happen.
In the above-given diagram, you can see that local and global deadlock detectors are searching for deadlocks and solve them with resuming transactions to their initial states.

Strict Two-Phase Locking Method

Strict-Two phase locking system is almost similar to 2PL. The only difference is that Strict-2PL never releases a lock after using it. It holds all the locks until the commit point and releases all the locks at one go when the process is over.

Centralized 2PL

In Centralized 2 PL, a single site is responsible for lock management process. It has only one lock manager for the entire DBMS.

Primary copy 2PL

Primary copy 2PL mechanism, many lock managers are distributed to different sites. After that, a particular lock manager is responsible for managing the lock for a set of data items. When the primary copy has been updated, the change is propagated to the slaves.

Distributed 2PL

In this kind of two-phase locking mechanism, Lock managers are distributed to all sites. They are responsible for managing locks for data at that site. If no data is replicated, it is equivalent to primary copy 2PL. Communication costs of Distributed 2PL are quite higher than primary copy 2PL.

CONCLUSION: - Thus we have implemented & demonstrated the Transaction and Concurrency control techniques using locks.

<---END--->