Department of Information Technology

Academic Year: 2023-24
Semester: III
Class / Branch: S[redacted]
Subject: SQL Lab
Name of Instructor: Prof. Charul Singh

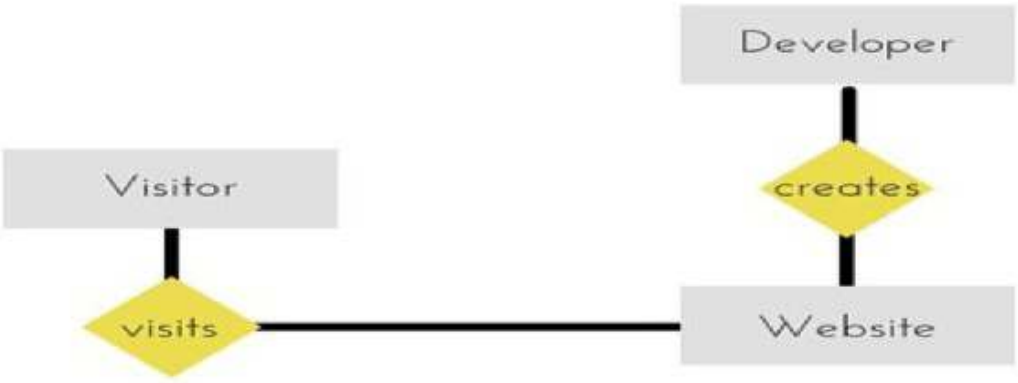
[redacted]

Experiment No. 1

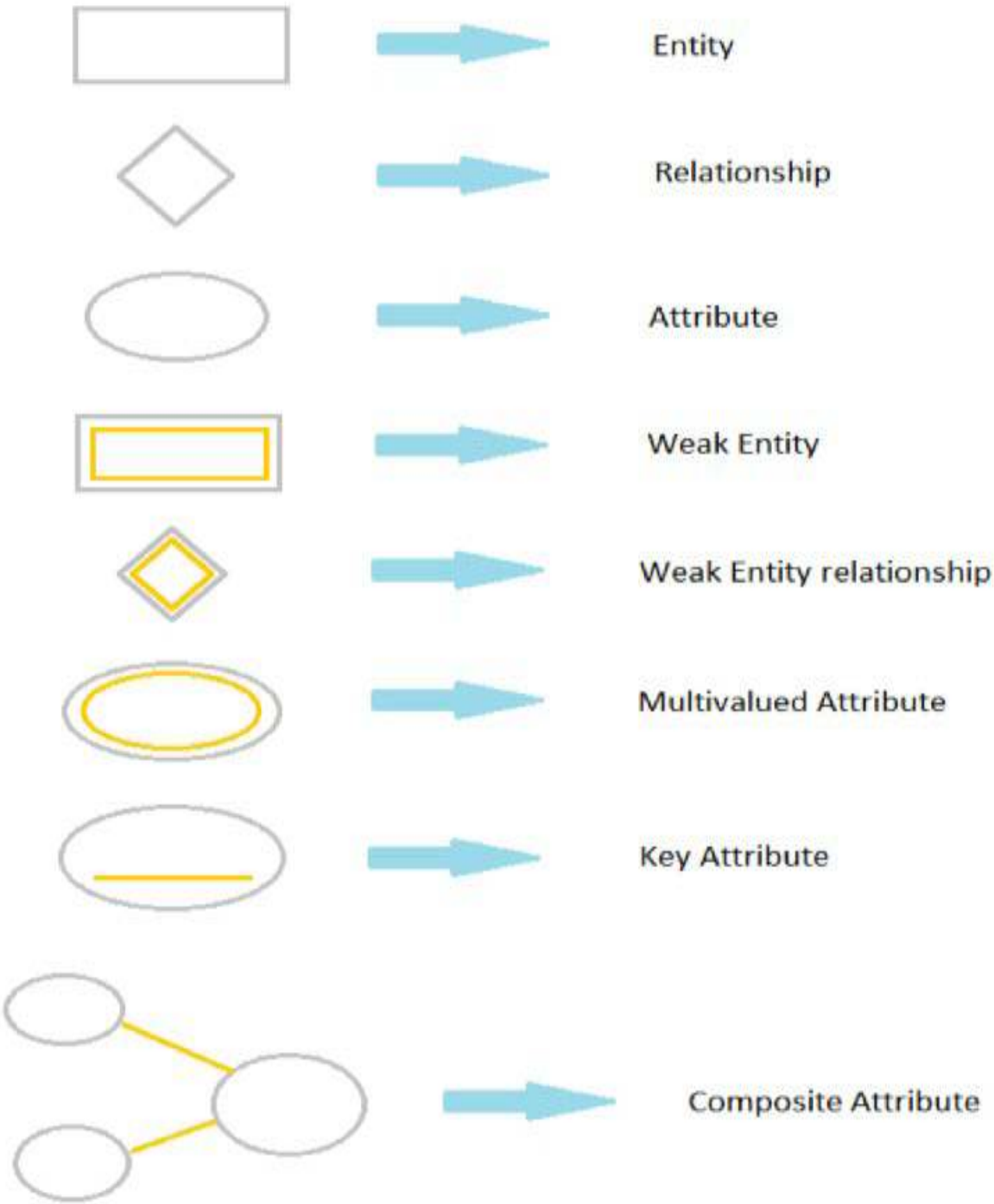
Aim:- To study and design the different Entity Relationship (ER) model components
Software used: Dia Diagram Editor 0.97.2
Theory:-

E-R Diagram

An ER model is composed of entity types and specifies relationships that can exist between instances of those entity types. An entity–relationship model is the result of using a systematic process to describe and define a subject area of business data. It does not define business process; only visualize business data. ER-Diagram is a visual representation of data that describes how data is related to each other.

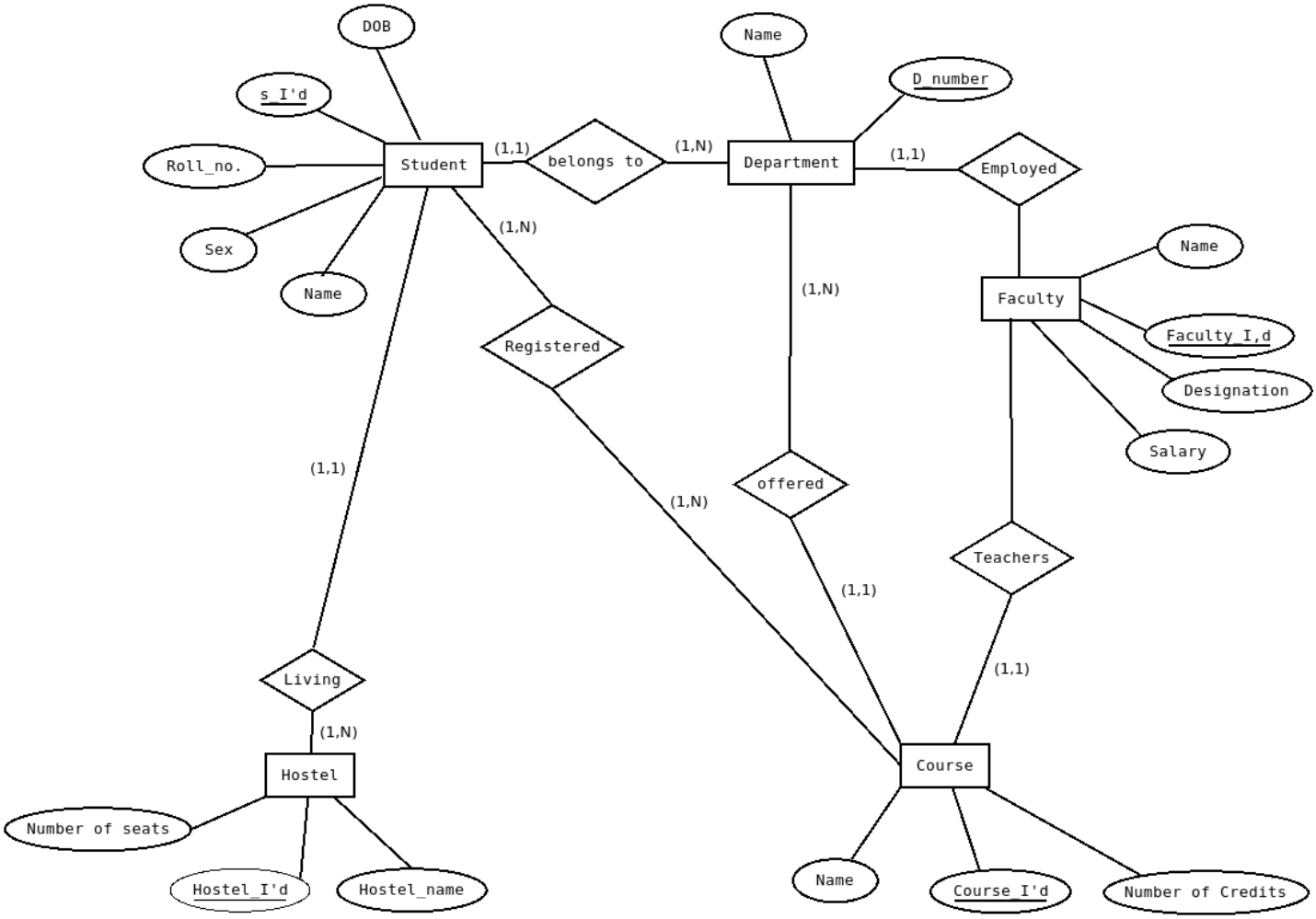


Symbols and Notations



Entity Relationship Diagram:

Draw an ER Diagram for College Administration Database (Paste your diagram)



Conclusion: Thus we have studied and designed EER diagram using Dia diagram editor. Also we studied different symbols and notations used to represent the relationship between different entities their attributes and also studied about binary & ternary relationships and their types.

Academic Year: 2023-24 **Name of Student** [redacted]
Semester: III **Student ID** [redacted]
Class / Branch [redacted] **Date of Performanc** [redacted]
Subject: SQL Lab **Date of Submissio** [redacted]
Name of Instructor: Prof. Charul Singh

Experiment No. 2

Aim:- **Aim:-** To study SQL and understand basis of MYSQL.

Software used: MySQL

Theory:-

SQL (Structured Query Language) is a standardized programming language used for managing relational databases and performing various operations on the data in them. Initially created in the 1970s, SQL is regularly used by database administrators, as well as by developers writing data integration scripts and data analysts looking to set up and run analytical queries.The uses of SQL include modifying database table and index structures; adding, updating and deleting rows of data; and retrieving subsets of information from within a database for transaction processing and analytic applications. Queries and other SQL operations take the form of commands written as statements commonly used SQL statements include select, add, insert, update, delete, create, alter and truncate.

Basic Terminologies

These are some basic terminologies of database management system

- **Data**

Data and information are created as synonymous terms. Data is the representation of the information actually stored on the disk storage in the computer. Information is the meaning of data understandable to user. e.g. “1008” is a data (numeric data) while “account no” is the information.

- **Database**

The collection of the related data about a particular enterprise is referred to as a “database”. An enterprise can be a self-contained, commercial, scientific, educational, technical or other organization. e.g. student database, bank account, items in the stores, etc.

- **Database System**

Database system is basically just a computerized record keeping system, which allow the user to access the information from the database.

- **Database Management System**

DBMS is a software system which organizes , maintains and manages the database to provide an environment i.e. convenient and efficient to use for users.
E.g. Oracle, Sybase,etc.

- **What is SQL?**

Structured Query Language or SQL is a standard computer language for accessing and manipulating database systems. SQL comprises one of the fundamental building blocks of modern database architecture. SQL defines methods using which user can create and manipulate databases on all major platforms. SQL is a set based declarative programming language and not an imperative programming language like C or BASIC.

- **Components of SQL**

SQL consists of three components:
1. Data Definition Language (DDL)
2. Data Manipulation Language (DML)
3. Data Control Language (DCL)

The Data Definition Language (DDL):
This component of the SQL language is used to create and modify tables and other objects in the database. For tables there are three main commands:

CREATE TABLE tablename to create a table in the database
DROP TABLE tablename to remove a table from the database
ALTER TABLE tablename to add or remove columns from a table in the database

The Data Manipulation Language (DML):
This component of the SQL language is used to manipulate data within a table. There are four main commands:

SELECT to select rows of data from a table
INSERT to insert rows of data into a table
UPDATE to change rows of data in a table
DELETE to remove rows of data from a table

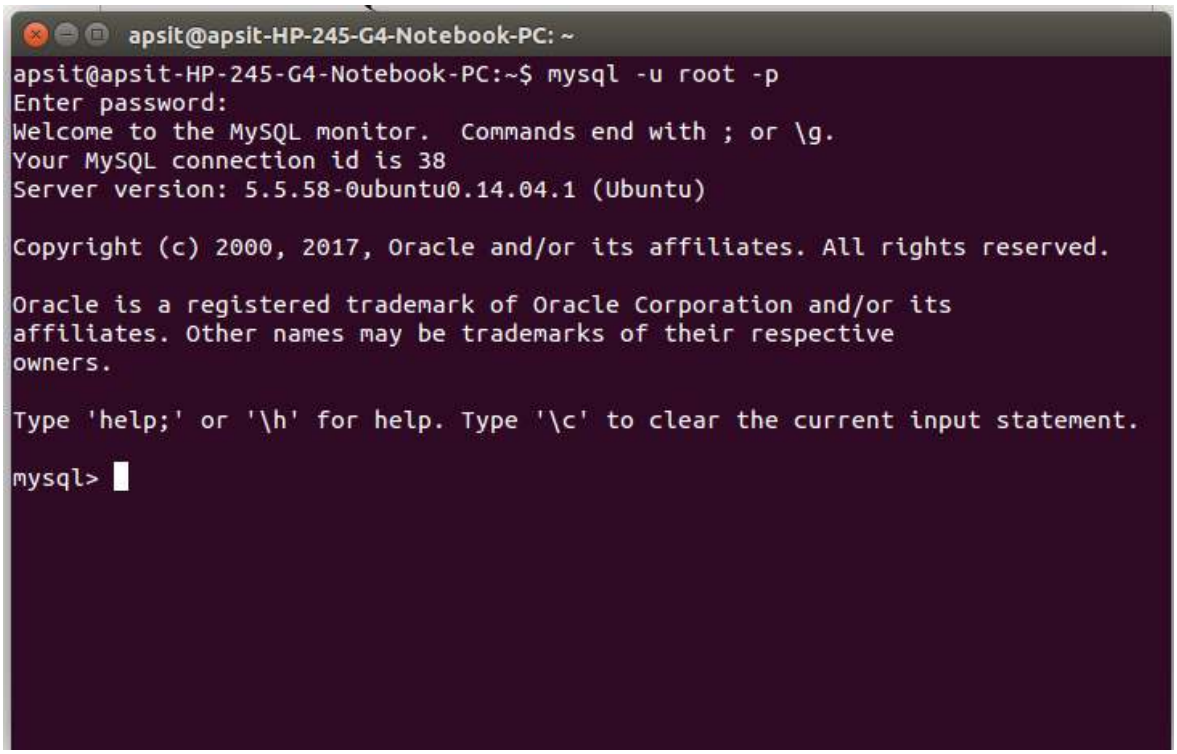
The Data Control Language (DCL):
This component of the SQL language is used to create privileges to allow users access to, and manipulation of, the database. There are two main commands:
GRANT to grant a privilege to a user
REVOKE to revoke (remove) a privilege from a user

- **What SQL do?**

SQL can execute queries against a database.
SQL can retrieve data from a database.
SQL can insert records in a database.
SQL can update records in a database.
SQL can delete records from a database.
SQL can create new databases.
SQL can create new tables in a database.
SQL can create stored procedures in a database.
SQL can set permissions on tables, procedures, and views.

- **What is MYSQL?**

MySQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by Oracle Corporation



Conclusion: Hence, we studied installation steps of Mysql server (5.5 and 5.6) and understood the commands of DDL, DML and DCL. Also we have studied how to enable the access to mysql from remote machine by alternating my.cnf configuration file.

Academic Year: 2023-24
Semester: III
Class / Branc
Subject: SQL Lab
Name of Instructor: Prof. Charul Singh

[redacted]

Experiment No. 03

Aim:- To create database using data definition language (DDL) commands.

Software used: MySQL

Theory:-

Data Definition Language (DDL) is a standard for commands that define the different structures in a database. DDL statements create, modify, and remove database objects such as tables, indexes, and users. Common DDL statements are CREATE, ALTER, and DROP.

INTEGRITY CONSTRAINT

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

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

a) Domain Integrity

This constraint sets a range and any violations that take place will prevent the user from performing the manipulation that caused the breach. It includes:

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.

b) Entity Integrity 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.

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.

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

SQL Commands:

1) CREATE TABLE

It is used to create a table.

Syntax: Create table tablename (column_name1 data_ type(size) constraints, column_name2 data_ type(size) constraints …)

Example:

Create table Emp (EmpNo number(5) primary key, EName VarChar(15), Job Char(10), DeptNo number(3));

Create table stud (sname varchar2(20) not null, rollno number(10) primary key,dob date not null);

```
mysql> create table emp(eno int, ename varchar(20), job varchar(20), sal int(6));
ERROR 1050 (42S01): Table 'emp' already exists
mysql> desc emp;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| eno   | int(11)     | YES  |     | NULL    |       |
| ename | varchar(20) | YES  |     | NULL    |       |
| job   | varchar(20) | YES  |     | NULL    |       |
| sal   | int(6)      | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

2) ALTER TABLE

Alter command is used to:

1. Add a new column.
2. Modify the existing column definition

3. To include or drop integrity constraint.

Syntax:

alter table tablename add/modify (attribute datatype(size));

Example:

- Alter table emp add (phone_no char (20));
- Alter table emp modify(phone_no number (10));
- ALTER TABLE EMP ADD CONSTRAINT Pkey1 PRIMARY KEY (EmpNo);

```
mysql> alter table emp add(phone_no char(15));
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc emp;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| eno      | int(11)     | YES  |     | NULL    |       |
| ename    | varchar(20) | YES  |     | NULL    |       |
| job      | varchar(20) | YES  |     | NULL    |       |
| sal      | int(6)      | YES  |     | NULL    |       |
| phone_no | char(15)    | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

mysql> alter table emp modify phone_no int(20);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc emp;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| eno      | int(11)     | YES  |     | NULL    |       |
| ename    | varchar(20) | YES  |     | NULL    |       |
| job      | varchar(20) | YES  |     | NULL    |       |
| sal      | int(6)      | YES  |     | NULL    |       |
| phone_no | int(20)     | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

```
mysql> alter table emp drop column job;
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc emp;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| eno      | int(11)     | YES  |     | NULL    |       |
| ename    | varchar(20) | YES  |     | NULL    |       |
| sal      | int(6)      | YES  |     | NULL    |       |
| phone_no | int(20)     | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

3) DROP TABLE

It will delete the table structure provided the table should be empty.

Example:
drop table prog20; Here prog20 is table name

```
mysql> drop table emp;
Query OK, 0 rows affected (0.01 sec)
```

4) TRUNCATE TABLE

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

Syntax: TRUNCATE TABLE <TABLE NAME>;

Example: Truncate table stud;

```
mysql> truncate table emp;
Query OK, 0 rows affected (0.01 sec)
```

5) DESC

This is used to view the structure of the table.

Example: desc emp; Name

```
mysql> desc emp;
ERROR 1146 (42S02): Table 'Employee.emp' doesn't exist
```

Conclusion :- Thus, we have understood the various commands of DDL like create,alter,drop,truncate & desc and concept of creating, altering a table using those commands.

Academic Year: 2023-24
Semester: III
Class / Bran[redacted]
Subject: SQL Lab
Name of Instructor: Prof. Charul Singh

Experiment No. 4

Aim:- To study and implement data manipulation language (DML) commands.

Queries:

Q1: Insert a single record into dept table.
Solution:
1.Decide the data to add in dept.
2.Add to dept one row at a time using the insert into syntax.

Ans:
SQL> insert into dept values (1,'IT','Tholudur');
1 row created.

Q2: Insert more than a record into emp table using a single insert command.

Ans:
SQL> insert into emp values(1,'Mathi','AP',1,10000)
1 row created.
SQL> insert into emp values(2,'Arjun','ASP',2,12000)
1 row created.
SQL> insert into emp values(3,'Gugan','ASP',1,12000)
1 row created.

```
mysql> insert into stud values(1, 'Purav_stokes', 28, 'DBMS');
Query OK, 1 row affected (0.00 sec)

mysql> insert into stud values(2, 'Rahul', 19,'DSA');
Query OK, 1 row affected (0.00 sec)

mysql> insert into stud values(3, 'Abhishek_stokes',18,'SQL');
Query OK, 1 row affected (0.01 sec)

mysql> select* from stud;
+------+-----------------+-------+-----------+
| s_id | s_name          | s_age | s_subject |
+------+-----------------+-------+-----------+
|    1 | Purav_stokes    |    28 | DBMS      |
|    2 | Rahul           |    19 | DSA       |
|    3 | Abhishek_stokes |    18 | SQL       |
+------+-----------------+-------+-----------+
3 rows in set (0.00 sec)
```

Q3: Update the emp table to set the salary of all employees to Rs15000/- who are working as ASP

Ans:

SQL> select * from emp;

EMPNO	ENAME	JOB	DEPTNO	SAL
1	Mathi		AP 1	10000
2	Arjun		ASP 2	12000
3	Gugan	ASP	1	12000

SQL> update emp set sal=15000 where job='ASP';
2 rows updated.

SQL> select * from emp;

EMPNO	ENAME	JOB	DEPTNO	SAL
1	Mathi		AP 1	10000
2	Arjun	ASP	2	15000
3	Gugan		ASP 1	15000

```
mysql> update stud set s_age=18 where s_id=1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> desc stud;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| s_id      | int(11)     | YES  |     | NULL    |       |
| s_name    | varchar(30) | YES  |     | NULL    |       |
| s_age     | int(11)     | YES  |     | NULL    |       |
| s_subject | varchar(16) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

Q4: Create a pseudo table employee with the same structure as the table emp and insert
rows into the table using select clauses.
Ans:

SQL> create table employee as select * from emp;
Table created.
SQL> desc employee;

Name	Null?	Type
EMPNO		NUMBER (6)
ENAME	NOT NULL	VARCHAR2(20)
JOB	NOT NULL	VARCHAR2 (13)
DEPTNO		NUMBER (3)
SAL		NUMBER (7)

Q5: select employee name, job from the emp table
Ans:

SQL> select ename, job from emp;

ENAME	JOB
Mathi	AP
Arjun	ASP
Gugan	ASP
Karthik	Prof
Akalya	AP
Suresh	lect

6 rows selected.
Q6: Delete only those who are working as lecturer
Ans:
SQL> select * from emp;

EMPNO	ENAME	JOB	DEPTNO	SAL

1	Mathi		AP	1	10000
2	Arjun	ASP		2	15000
3	Gugan	ASP		1	15000
4	Karthik	Prof		2	30000
5	Akalya	AP		1	10000
6	suresh		lect	1	8000

6 rows selected.

SQL> delete from emp where job='lect';
1 row deleted.

SQL> select * from emp;

EMPNO	ENAME	JOB	DEPTNO	SAL
1	Mathi		AP	1 10000
2	Arjun	ASP	2	15000
3	Gugan		ASP	1 15000
4	Karthik	Prof	2	30000
5	Akalya	AP	1	10000

```
mysql> insert into stud (s_id,s_name,s_age,s_subject) values (4,'Hamza',18,'SQL');
Query OK, 1 row affected (0.01 sec)

mysql> select* from stud;
+------+-----------------+-------+-----------+
| s_id | s_name          | s_age | s_subject |
+------+-----------------+-------+-----------+
|    1 | Purav_stokes    |    28 | DBMS      |
|    2 | Rahul           |    19 | DSA       |
|    3 | Abhishek_stokes |    18 | SQL       |
|    4 | Hamza           |    18 | SQL       |
+------+-----------------+-------+-----------+
4 rows in set (0.00 sec)
```

Q7: List the records in the emp table orderby salary in ascending order. Ans:
SQL> select * from emp order by sal;

EMPNO	ENAME	JOB	DEPTNO	SAL
1	Mathi		AP	1 10000

5	Akalya		AP	1	10000
2	Arjun	ASP	2		15000
3	Gugan		ASP	1	15000
4	Karthik	Prof	2		30000

Q8: List the records in the emp table orderby salary in descending order. Ans:

SQL> select * from emp order by sal desc;

EMPNO	ENAME	JOB	DEPTNO	SAL
4	Karthik	Prof	2	30000
2	Arjun	ASP	2	15000
3	Gugan	ASP	1	15000
1	Mathi	AP	1	10000
5	Akalya	AP	1	10000

Q9: Display deptno from the table employee avoiding the duplicated values.
Ans:
SQL> select distinct deptno from emp;

DEPTNO
1
2

```
nysql> delete from stud where s_id=4;
Query OK, 1 row affected (0.00 sec)

nysql> select* from stud;
+------+-----------------+-------+-----------+
| s_id | s_name          | s_age | s_subject |
+------+-----------------+-------+-----------+
|    1 | Purav_stokes    |    18 | DBMS      |
|    2 | Rahul           |    19 | DSA       |
|    3 | Abhishek_stokes |    18 | SQL       |
+------+-----------------+-------+-----------+
3 rows in set (0.00 sec)
```

Conclusion :- Hence, we successfully studied and implemented all the DML commands like insert,select update,delete using MySQL server.

AcademicYear:2022-23 **Semester:III**
Class/Branch:SEIT **Subject:SQLLab**

AcademicYear:2023-24
Semester:III
Class/Branch:SE(IT)
Subject:SQLLab
NameofInstructor:Prof.CharulSingh

[redacted]

ExperimentNo. 5

Aim:-Tostudyandimplementbasicandcomplex SQLqueries

QueriesforSETOperator:

Q1:Displayallthedeptnumbersavailablewiththedeptandemptablesavoidingduplicates.

Ans: SQL>selectdept
fromempunionselectdeptnofromdept;DEPTNO

1
2
30
40

```
mysql> select * from first union select * from second;
+------+--------+
| id   | name   |
+------+--------+
|    1 | Abhi   |
|    2 | Purav  |
|    3 | Stokes |
+------+--------+
```

Q2:Displayallthe deptnumbersavailablewiththe deptandemptables.Ans:

SQL>select deptfrom empunion all selectdeptnofromdept;

DEPTNO

1
2
2
1
12
1
2
30
40

9rowsselected.

```
mysql> select * from first union all select * from second;
+------+--------+
| id   | name   |
+------+--------+
|    1 | Abhi   |
|    2 | Purav  |
|    2 | Purav  |
|    3 | Stokes |
+------+--------+
```

Q3:Display**allthe deptnumbersavailableinempandnotindepttablesandviceversa.Ans:**

SQL>selectdeptfromempminusselectdeptnofromdept;DEP

TNO

12

SQL>selectdeptfromdeptminusselectdeptnofromemp;DEP

TNO

30

40

```
mysql> select * from first intersect select * from second;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax t
o use near 'select * from second' at line 1
mysql> select * from first - select * from second;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax t
o use near '- select * from second' at line 1
```

e) Queries for

JOINS: Tables used

SQL>select *fromemp;

EMPNO	ENAME	JOB	DEPTNO	SAL
1	Mathi	AP	1	10000
2	Arjun	ASP	2	12000
3	Gugan	ASP	2	20000
4	Karthik	AP	1	15000

SQL>select *fromdept;

DEPTNO	DNAME	LOC
1	ACCOUNTING	NEWYORK
2	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

```
mysql> select* from emp;
+-------+---------+------+------+-------+
| empno | ename   | job  | dept | sal   |
+-------+---------+------+------+-------+
|     1 | Mathi   | AP   |    1 | 10000 |
|     2 | Arjun   | ASP  |    2 | 12000 |
|     3 | Gugan   | ASP  |    2 | 20000 |
|     4 | Karthik | AP   |    1 | 15000 |
+-------+---------+------+------+-------+
4 rows in set (0.00 sec)

mysql> select* from dept;
+--------+------------+----------+
| deptno | dName      | Loc      |
+--------+------------+----------+
|      1 | ACCOUNTING | NEW YORK |
|      2 | RESEARCH   | DALLAS   |
|     30 | SALES      | CHICAGO  |
|     40 | OPERATIONS | BOSTON   |
+--------+------------+----------+
4 rows in set (0.00 sec)

mysql>
```

EQUI-JOIN

Q1: Display the employee details, departments that the departments are same in both theempand dept.

Ans:

SQL>select*fromemp,dept
whereemp.dept=dept.deptno;

EMPNO	ENAME	JOB	DEPTNO	SAL	DEPTNO	DNAME	LOC
1	Mathi	AP	1	10000	1	ACC	NEWYORK
2	Arjun	ASP	2	12000	2	RESEARCH	DALLAS
3	Gugan	ASP	2	20000	2	RESEARCH	DALLAS
4	Karthik	AP	1	15000	1	ACC	NEWYORK

```
mysql> select * from customer join orders ON customer.cId=orders.cId;
+------+-------+----------------------+------+------+------------+---------+------+
| cId  | cName | cEmail               | cAge | oId  | oDate      | oAmount | cId  |
+------+-------+----------------------+------+------+------------+---------+------+
|    1 | Purav | puravshah18@gmail.com |   18 |    1 | 2021-10-05 |   50000 |    1 |
|    2 | Rahul | rahul19@gmail.com     |   19 |    2 | 2021-07-20 |   10000 |    2 |
|    1 | Purav | puravshah18@gmail.com |   18 |    3 | 2022-01-11 |   40000 |    1 |
|    3 | Hamza | hamzastokes@gmail.com |   19 |    4 | 2022-05-26 |   30000 |    3 |
+------+-------+----------------------+------+------+------------+---------+------+
```

NON-EQUIJOIN

Q2: Display the employee details, departments that the departments are ame in both theempand dept.

Ans:

SQL>select*fromemp,deptwhereemp.dept!=dept.deptno;

EMPNO	ENAME	JOB	DEPTNO	SALARY	DEPTNO	DNAME	LOC
2	Arjun	ASP	2	12000	1	ACCOUNTING	NEWYORK
3	Gugan	ASP	2	20000	1	ACCOUNTING	NEWYORK
1	Mathi	AP	1	10000	2	RESEARCH	DALLAS
4	Karthik	AP	1	15000	1	ACCOUNTING	NEWYORK

```
mysql> select * from customer join orders ON customer.cId=orders.cId;
+------+-------+----------------------+------+------+------------+---------+------+
| cId  | cName | cEmail               | cAge | oId  | oDate      | oAmount | cId  |
+------+-------+----------------------+------+------+------------+---------+------+
|    1 | Purav | puravshah18@gmail.com |   18 |    1 | 2021-10-05 |   50000 |    1 |
|    2 | Rahul | rahul19@gmail.com     |   19 |    2 | 2021-07-20 |   10000 |    2 |
|    1 | Purav | puravshah18@gmail.com |   18 |    3 | 2022-01-11 |   40000 |    1 |
|    3 | Hamza | hamzastokes@gmail.com |   19 |    4 | 2022-05-26 |   30000 |    3 |
+------+-------+----------------------+------+------+------------+---------+------+
```

LEFTOUT-JOIN

Tablesused

SQL>select *fromstud1;

Regno	Name	Mark2	Mark3	Result
101	john	89	80	pass
102	Raja	70	80	pass
103	Sharin	70	90	pass
104	sam	90	95	pass

SQL>select*fromstud2;N

AME GRA

---

john s

raj s
sam a
sharin a

```
mysql> select* from customer;
+------+----------+-----------------------+------+
| cId  | cName    | cEmail                | cAge |
+------+----------+-----------------------+------+
|    1 | Purav    | puravshah18@gmail.com |   18 |
|    2 | Rahul    | rahul19@gmail.com     |   19 |
|    3 | Hamza    | hamzastokes@gmail.com |   19 |
|    4 | Abhishek | abhi123@gmail.com     |   20 |
+------+----------+-----------------------+------+
```

```
mysql> select * from orders;
+------+------------+---------+------+
| oId  | oDate      | oAmount | cId  |
+------+------------+---------+------+
|    1 | 2021-10-05 |   50000 |    1 |
|    2 | 2021-07-20 |   10000 |    2 |
|    3 | 2022-01-11 |   40000 |    1 |
|    4 | 2022-05-26 |   30000 |    3 |
+------+------------+---------+------+
```

Q3:DisplaytheStudentnameandgradebyimplementingaleftouter join.

SQL>selectstud1.name, GRAfromstud3leftjoinstud1onstud1.Name=stud3.Name;

Name Gra

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

john s
Raja s
sam a
Sharin a

```
mysql> select customer.cId,cName,oAmount from customer left join orders on customer.cId=orders.cId;
+------+----------+---------+
| cId  | cName    | oAmount |
+------+----------+---------+
|    1 | Purav    |   50000 |
|    2 | Rahul    |   10000 |
|    1 | Purav    |   40000 |
|    3 | Hamza    |   30000 |
|    4 | Abhishek |    NULL |
+------+----------+---------+
```

RIGHTOUTER-JOIN

Q4: Display the Student name, register no, and result by implementing a right outer join.Ans:

SQL>selectstud1.Name,Regno,Resultfrom stud1rightjoinstud3on stud1.name=stud3name;

Name	Regno	Result
john	101	pass
raj	102	pass
sam	103	pass
sharin	104	pass

```
mysql> select customer.cName,cEmail,oDate from customer right join orders on customer.cId=orders.cId;
+-------+----------------------+------------+
| cName | cEmail               | oDate      |
+-------+----------------------+------------+
| Purav | puravshah18@gmail.com | 2021-10-05 |
| Purav | puravshah18@gmail.com | 2022-01-11 |
| Rahul | rahul19@gmail.com     | 2021-07-20 |
| Hamza | hamzastokes@gmail.com | 2022-05-26 |
+-------+----------------------+------------+
```

FULLOUTER-JOIN

Q5: Display the Student name register no by implementing a full outer join.Ans:

SQL>selectstud1.name,regnofromstud1fullouterjoinstud2on(stud1.name= stud2.name);

Name	Regno
john	101
raj	102
sam	103
sharin	104

SELFJOIN

Q6: Write a query to display their employee names.Ans:

SQL>selectdistinctenamefromempx,deptywherexdeptno=ydeptno;

ENAME

Arjun
Gugan

```
mysql> select distinct ename from emp x, dept y where x.dept=y.deptno;
+---------+
| ename   |
+---------+
| Mathi   |
| Karthik |
| Arjun   |
| Gugan   |
+---------+
4 rows in set (0.00 sec)
```

Karthik
Mathi

Q7: Display the details of those who draw the salary greater than the average salary.Ans:

SQL> select distinct*from empxwherex.sal >=
(selectavg(sal)fromemp);EMPNO ENAME JOB
DEPTNO SAL

| --- | --- | --- | --- | --- |
| 3 | Gugan | ASP | 2 | 20000 |

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

4 Karthik AP 1 15000

```
mysql> select distinct* from emp x where x.sal>=(select avg(sal) from emp);
+-------+---------+------+------+-------+
| empno | ename   | job  | dept | sal   |
+-------+---------+------+------+-------+
|     3 | Gugan   | ASP  |    2 | 20000 |
|     4 | Karthik | AP   |    1 | 15000 |
+-------+---------+------+------+-------+
2 rows in set (0.00 sec)
```

Conclusion:Hence, we understood the concept of union,union all, intersect and joins and its typeslike left join,right join,etc. Created separate tables and performed each operation on those tablesusinf MySQLserver.

Academic Year: 2022-23
Semester: III
Class / Branch:SE(IT)
Subject: SQL Lab
Name of Instructor: Prof. Charul Singh

[redacted]
Date of Perf[redacted]
Date of Sub[redacted]

Experiment No:6

Aim: To study and implement Views and Triggers

Creating triggers in Mysql

Dropping triggers in Mysql:

```
mysql> create trigger AFTER_INSERT after insert on professor for each row insert into student set name='ABC';
Query OK, 0 rows affected (0.11 sec)

mysql> select * from student;
+---------+-------------+
| name    | department |
+---------+-------------+
| prashil | IT         |
| pravin  | IT         |
| pankaj  | IT         |
+---------+-------------+
3 rows in set (0.00 sec)

mysql> select * from professor;
+------+---------+-------------+--------+
| Id   | name    | department | salary |
+------+---------+-------------+--------+
| 1    | mahesh  | IT         | 20000  |
| 2    | suresh  | IT         | 20000  |
| 3    | jayesh  | comp       | 25000  |
| 4    | brijesh | comp       | 25000  |
| 5    | nitin   | mech       | 30000  |
| 6    | jatin   | civil      | 40000  |
+------+---------+-------------+--------+
6 rows in set (0.00 sec)

mysql> insert into professor values(7,'kripa','ETRX',50000);
Query OK, 1 row affected (0.07 sec)

mysql> select * from student;
+---------+-------------+
| name    | department |
+---------+-------------+
| prashil | IT         |
| pravin  | IT         |
| pankaj  | IT         |
| ABC     | NULL       |
+---------+-------------+
4 rows in set (0.01 sec)
```

```
drop trigger trigger_name
```

Views:

Creating view comp from original table IT

```
mysql> select * from IT;
+---------+-------+---------+
| name    | phone | address |
+---------+-------+---------+
| neha    | 11    | thane   |
| brinal  | 22    | vasai   |
| archana | 33    | thane   |
+---------+-------+---------+
3 rows in set (0.00 sec)

mysql> create view comp(name,address) as select name,address from IT where phone
=11;
Query OK, 0 rows affected (0.07 sec)

mysql> select * from comp;
+------+---------+
| name | address |
+------+---------+
| neha | thane   |
+------+---------+
1 row in set (0.00 sec)
```

adding one more column to the view

```
mysql> alter view comp(name,address,phone) as select name,address,phone from IT where address='thane';
Query OK, 0 rows affected (0.05 sec)

mysql> select * from comp;
+---------+---------+-------+
| name    | address | phone |
+---------+---------+-------+
| neha    | thane   | 11    |
| archana | thane   | 33    |
+---------+---------+-------+
2 rows in set (0.00 sec)

mysql>
```

Updating original table IT only and the result is getting reflected into original table as well as into views

```
mysql> update IT set name='mudra' where phone=11;
Query OK, 1 row affected (0.08 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from IT;
+---------+-------+---------+
| name    | phone | address |
+---------+-------+---------+
| mudra   | 11    | thane   |
| brinal  | 22    | vasai   |
| archana | 33    | thane   |
+---------+-------+---------+
3 rows in set (0.00 sec)

mysql> select * from comp;
+---------+---------+-------+
| name    | address | phone |
+---------+---------+-------+
| mudra   | thane   | 11    |
| archana | thane   | 33    |
+---------+---------+-------+
2 rows in set (0.01 sec)

mysql>
```

Applying aggregate functions to the views

```
mysql> select min(phone) as new_phone from comp;
+-----------+
| new_phone |
+-----------+
| 11        |
+-----------+
1 row in set (0.02 sec)

mysql>
```

```
mysql> drop view comp;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from comp;
ERROR 1146 (42S02): Table 'neha.comp' doesn't exist
mysql>
```

```
mysql> select * from customers;
+-------+----------+------+-----------+---------+
| cid   | cname    | age  | address   | salary  |
+-------+----------+------+-----------+---------+
|     1 | Purav    |   18 | Virar     |   50000 |
|     2 | Rahul    |   19 | Matunga   |   60000 |
|     3 | Abhishek |   20 | Diva      |   50000 |
|     4 | Hamza    |   18 | Mira Road |   40000 |
+-------+----------+------+-----------+---------+
4 rows in set (0.00 sec)
```

```
mysql> create view customers_view as select cname,age from customers;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from customers_view;
+----------+------+
| cname    | age  |
+----------+------+
| Purav    |   18 |
| Rahul    |   19 |
| Abhishek |   20 |
| Hamza    |   18 |
+----------+------+
4 rows in set (0.00 sec)
```

```
mysql> update customers_view set age=19 where cname="Purav";
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from customers_view;
+----------+------+
| cname    | age  |
+----------+------+
| Purav    |   19 |
| Rahul    |   19 |
| Abhishek |   20 |
| Hamza    |   18 |
+----------+------+
4 rows in set (0.00 sec)
```

```
mysql> delete from customers_view where cname="Hamza";
Query OK, 1 row affected (0.00 sec)

mysql> select * from customers;
+------+----------+------+---------+--------+
| cid  | cname    | age  | address | salary |
+------+----------+------+---------+--------+
|    1 | Purav    |   19 | Virar   |  50000 |
|    2 | Rahul    |   19 | Matunga |  60000 |
|    3 | Abhishek |   20 | Diva    |  50000 |
+------+----------+------+---------+--------+
3 rows in set (0.00 sec)
```

```
mysql> drop view customers_view;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from customers;
+------+----------+------+---------+--------+
| cid  | cname    | age  | address | salary |
+------+----------+------+---------+--------+
|    1 | Purav    |   19 | Virar   |  50000 |
|    2 | Rahul    |   19 | Matunga |  60000 |
|    3 | Abhishek |   20 | Diva    |  50000 |
+------+----------+------+---------+--------+
3 rows in set (0.00 sec)
```

<u>Conclusion:</u> Hence, we understood the concept of view and performed DDL,DML commands on the view using MySQL server.

Department of Information Technology

Academic Year: 2023-24
Semester: III
Class / Branch: C/IT **Subject: SQL Lab**

Experiment No: 7

Aim: To demonstrate of database connectivity using JDBC.

Software used:- Eclipse IDE, PostgreSQL, JDBC Driver

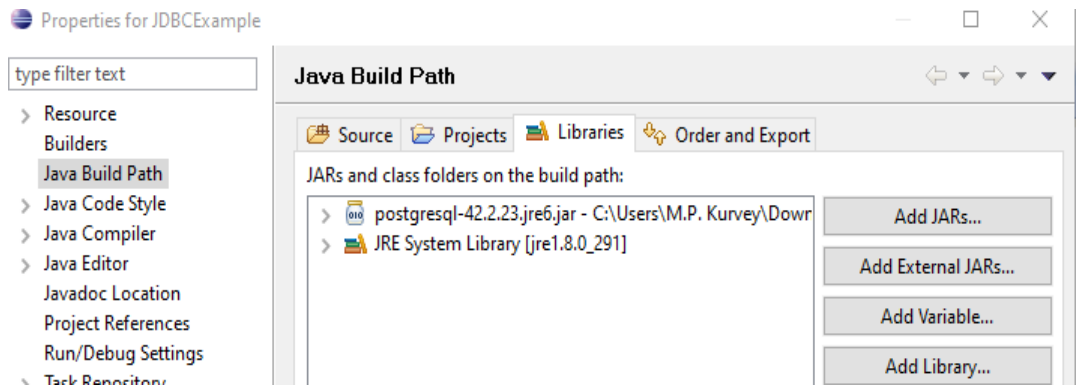
Downloading JDBC Driver for PostgreSQL

Binary Jar file downloads of the JDBC driver is available.

https://jdbc.postgresql.org/download.html

3.1 Load the driver in Eclipse IDE and Register Driver

3.2 Add the driver which is external JAR file to Libraries section of the project

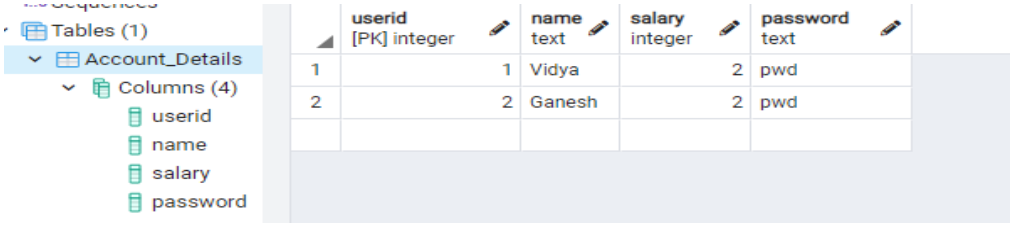


Step 2 : Create Accounts Database

In PostgreSQL design database having name Accounts.

Create table Account_Details in this database. And add columns in this table. Insert rows in this table.

Following is the Structure of Account_Details table to be created:



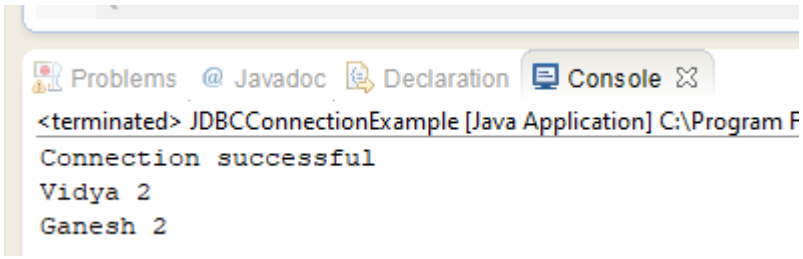
	userid [PK] integer	name text	salary integer	password text
1	1	Vidya	2	pwd
2	2	Ganesh	2	pwd

Step 3 : Write Java code to Establish the connection with Database and select multiple columns

```
import java.sql.*;
public class JDBCConnectionExample {

    private final String url = "jdbc:postgresql://localhost/Accounts";
    private final String username = "postgres";
    private final String pwd = "user";
    Connection connect;

    //Step4:Establish the connection
    //connect method to connect to database
    private void connect()
    {
            try
        {
            //getConnection: static: Return a connection object
            connect = DriverManager.getConnection(url, username, pwd);
        }
        catch(SQLException e)
        {
            System.out.println("Connection issues");
            e.printStackTrace();
        }

        if(connect!=null)
            System.out.println("Connection successful");
        else
            System.out.println("Connection issues");

    }
```

```
//execute
private void execute()
{
    try
    {
        //Create a statement
        Statement stmt = connect.createStatement(); //created a Statement object
        ResultSet result = stmt.executeQuery("Select name, salary "
                + "from public.\"Account_Details\"    " );


        while(result.next())
        {
            System.out.println(result.getString(1)+ " " + result.getInt(2));
        }
    }
    catch(SQLException e)
    {
        System.out.println("excution issues");
        e.printStackTrace();
    }


}




   public static void main(String[] args) {
       // TODO Auto-generated method stub
       JDBCConnectionExample jdbc = new JDBCConnectionExample();
       jdbc.connect();
       jdbc.execute();

   }

}
```

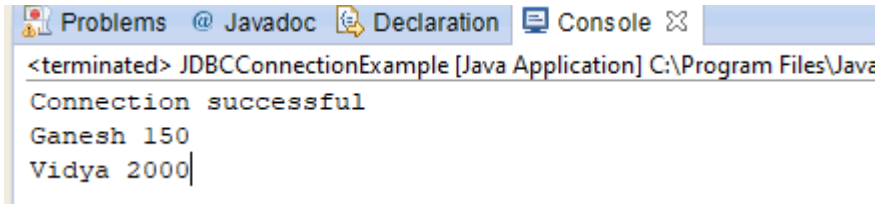
Output:



Step 4: Write Java code to update table

```
//execute
private void execute()
{
    try
    {
        //Create a statement
        PreparedStatement stmt = connect.prepareStatement("update "
                + "public.\"Account_Details\" SET salary = 2000 where userid=1 ")
        stmt.execute();
    }
    catch(SQLException e)
    {
        System.out.println("excution issues");
        e.printStackTrace();
    }


}
```

Output:



<u>Conclusion</u> : Hence, we have successfully understood how to connect the SQL database to a java program with the help of Driver Manager and using connection statement which uses the URL, username and password and also execute the sql queries using resultset, everything executed using try catch mechanism in case of any exception.

Department of Information Technology

Academic Year: 2023-24
Semester: III
Class / Branch: C/IT
Subject: SQL Lab
Name of Instructor: Prof. Charul Singh

Experiment No:8

Aim: To implement TCL commands and concurrency control techniques using locks

Software used: MySQL

Theory :

Transaction control language (TCL) commands are used to manage transactions in database. These are used to manage the changes made by DML statements. It also allows statements to be grouped together into logical transactions**.**

Commit command

Commit command is used to permanently save any transaction into database. Following is Commit command's
Syntax: *commit*;

Rollback command

This command restores the database to last commited state. It is also use with savepoint command to jump to a savepoint in a transaction.
Following is Rollback command's syntax:
rollback **to** ***savepoint-name***;

Savepoint command

Savepoint command is used to temporarily save a transaction so that you can rollback to that point whenever necessary.
Following is savepoint command's syntax:
savepoint *savepoint-name*;

Example of Savepoint and Rollback

ID	NAME
1	abhi
2	adam

4	alex

Lets use some SQL queries on the above table and see the results.

```
INSERT into class
values(5,'Rahul'); commit;

UPDATE class set name='abhijit' where
id='5'; savepoint A;

INSERT into class
values(6,'Chris'); savepoint B;

INSERT into class
values(7,'Bravo'); savepoint C;

SELECT * from class;
```

The resultant table will look like,

ID	NAME
1	abhi
2	adam
4	alex
5	abhijit
6	chris
7	bravo

Now **rollback** to **savepoint B**

```
rollback to B;
SELECT * from
class;
```

The resultant table will look like

ID	NAME
1	abhi
2	adam
4	alex
5	abhijit
6	chris

Now **rollback** to **savepoint A**

rollback to A;
SELECT * from
class;

The result table will look like

ID	NAME
1	abhi
2	adam
4	alex
5	abhijit

Transaction

A transaction can be defined as a group of tasks. A single task is the minimum processing unit which cannot be divided further.Let's take an example of a simple transaction. Suppose a bank employee transfers Rs 500 from A's account to B's account. This very simple and small transaction involves several low-level tasks.

A's Account

Open_Account(A)

Old_Balance = A.balance

New_Balance = Old_Balance - 500

A.balance = New_Balance

Close_Account(A)

B's Account

Open_Account(B)

Old_Balance = B.balance

New_Balance = Old_Balance + 500

B.balance = New_Balance

Close_Account(B)

Concurrency Control

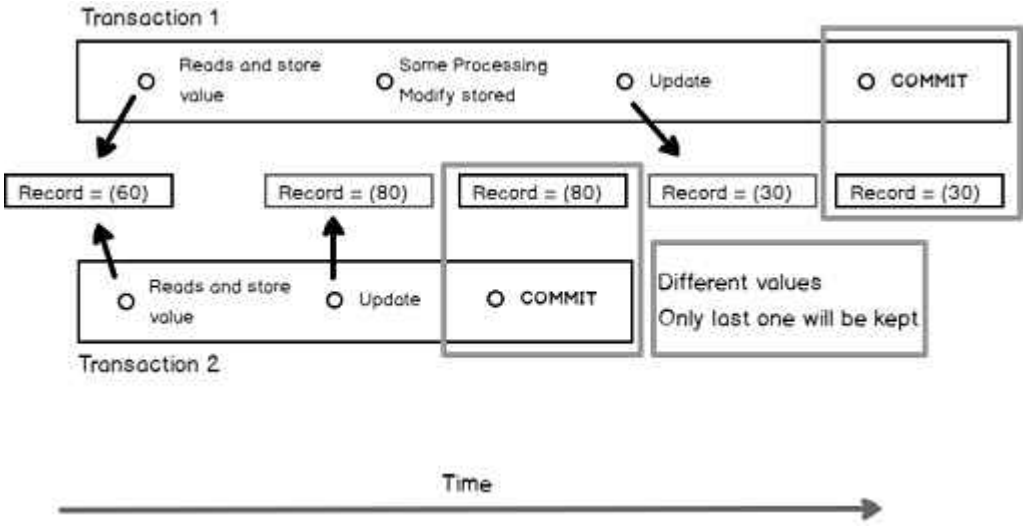
Before diving into transaction levels details, it's important to get used to typical concurrency problems and how we call them.

Lost update and dirty write

This phenomenon happens when two transactions access the same record and both updates this record. The following figure summarizes what could happen in a simple example.

In this example, we have 2 concurrent transactions that access a record with a (60) modifiable value. This record is identified either by its rowId or by a primary key column that won't be presented here for simplicity.

The first transaction reads this record, does some processing then updates this record and finally commits its work. The second transaction reads the record then updates it immediately and commits. Both transactions do not update this record to the same value. This leads to a loss for the update statement performed by second transaction.



As Transaction 1 **overwrites a value that** Transaction 2 **already modified. We could have said that** Transaction 1 **did a « dirty write » if** Transaction 2 **didn't commit its work.**

```
mysql> commit;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from s;
+------+----------+
| id   | name     |
+------+----------+
|    1 | Purav    |
|    2 | Rahul    |
|    3 | Abhishek |
|    4 | Hamza    |
|    5 | Stokes   |
+------+----------+
5 rows in set (0.00 sec)
```

```
mysql> rollback;
Query OK, 0 rows affected (0.01 sec)

mysql> select * from s;
+------+----------+
| id   | name     |
+------+----------+
|    1 | Purav    |
|    2 | Rahul    |
|    3 | Abhishek |
|    4 | Hamza    |
|    5 | Stokes   |
+------+----------+
5 rows in set (0.00 sec)
```

```
mysql> insert into  s values(6,"Rudra");
Query OK, 1 row affected (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (0.00 sec)

mysql> rollback;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from s;
+------+----------+
| id   | name     |
+------+----------+
|    1 | Purav    |
|    2 | Rahul    |
|    3 | Abhishek |
|    4 | Hamza    |
|    5 | Stokes   |
|    6 | Rudra    |
+------+----------+
6 rows in set (0.00 sec)
```

```
mysql> rollback to upd;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from s;
+------+----------+
| id   | name     |
+------+----------+
|    1 | Purav    |
|    2 | Rahul    |
|    3 | Abhishek |
|    4 | Hamza    |
|    5 | Amit     |
|    6 | Rudra    |
|    7 | Aniket   |
+------+----------+
7 rows in set (0.00 sec)
```

```
mysql> rollback to ins;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from s;
+------+----------+
| id   | name     |
+------+----------+
|    1 | Purav    |
|    2 | Rahul    |
|    3 | Abhishek |
|    4 | Hamza    |
|    5 | Stokes   |
|    6 | Rudra    |
|    7 | Aniket   |
+------+----------+
7 rows in set (0.00 sec)
```

```
mysql> savepoint del;
Query OK, 0 rows affected (0.00 sec)

mysql> select*  from s;
+------+----------+
| id   | name     |
+------+----------+
|    1 | Purav    |
|    2 | Rahul    |
|    3 | Abhishek |
|    4 | Hamza    |
|    6 | Rudra    |
|    7 | Aniket   |
+------+----------+
6 rows in set (0.00 sec)
```

<u>Conclusion:</u> We have understood the concept of TCL(Transaction Control Language) commands like commit,rollback and savepoint. Also, implemented these commands using MySQL server.

Academic Year: 2023-24 **N[redacted]**
Semester: III
Class / Branch: SEIT-C **[redacted]**

Experiment No:09

Aim: To study and implement various functions and procedures in SQL.

Software used: MySQL

Theory :-

Procedures and Functions are the subprograms which can be created and saved in the database as database objects. They can be called or referred inside the other blocks also.

Procedures & Functions

"A **procedures** or **function** is a group or set of SQL and PL/SQL statements that perform a specific task." A function and procedure is a named PL/SQL Block which is similar.The major difference between a procedure and a function is, a function must always return a value, but a procedure may or may not return a value.

Procedures:

A procedure is a named PL/SQL block which performs one or more specific task. This is similar to a procedure in other programming languages. A procedure has a header and a body. The header consists of the name of the procedure and the parameters or variables passed to the procedure.
The body consists or declaration section, execution section and exception section similar to a general PL/SQL Block. A procedure is similar to an anonymous PL/SQL Block but it is named for repeated usage.We can pass parameters to procedures in three ways :

Parameters	Description
IN type	These types of parameters are used to send values to stored procedures.
OUT type	These types of parameters are used to get values from stored procedures. This is similar to a return type in functions.
IN OUT type	These types of parameters are used to send values and get values from stored procedures

A procedure may or may not return any value.

Syntax:

```
CREATE [OR REPLACE] PROCEDURE procedure_name (<Argument> {IN, OUT, IN OUT} <Datatype>,…)
IS
 Declaration section<variable, constant> ;
BEGIN
 Execution section
EXCEPTION
 Exception section
END
```

IS - marks the beginning of the body of the procedure and is similar to DECLARE in anonymous PL/SQL Blocks. The code between IS and BEGIN forms the Declaration section. The syntax within the brackets [ ] indicate they are optional. By using CREATE OR REPLACE together the procedure is created if no other procedure with the same name exists or the existing procedure is replaced with the current code.

How to execute a Procedure?

There are two ways to execute a procedure :

- From the SQL prompt : EXECUTE [or EXEC] procedure_name;
- Within another procedure – simply use the procedure name : procedure_name;

Example:

create table named emp have two column id and salary with number datatype.

```
CREATE OR REPLACE PROCEDURE p1(id IN NUMBER, sal IN NUMBER)
AS
BEGIN
 INSERT INTO emp VALUES(id, sal);
 DBMD_OUTPUT.PUT_LINE('VALUE INSERTED.');
```

```
END;
/
```

Output:

Run SQL Command Line

```
SQL>set serveroutput on
SQL>start D://pr.sql
Procedure created.

SQL>exec p1(5,4);
VALUE INSERTED.
PL/SQL procudere successfully completed.

SQL>select * from emp;
   ID     SALARY
 -----   --------
    2       5000
```

Functions:

A function is a named PL/SQL Block which is similar to a procedure. The major difference between a procedure and a function is, a function must always return a value, but a procedure may or may not return a value.

Syntax:

```
CREATE [OR REPLACE] FUNCTION function_name [parameters]
 RETURN return_datatype;  {IS, AS}
 Declaration_section <variable,constant> ;
 BEGIN
   Execution_section
   Return return_variable;
 EXCEPTION
  exception section
   Return return_variable;
 END;
```

RETURN TYPE: The header section defines the return type of the function. The return datatype can be any of the oracle datatype like varchar, number etc.
The execution and exception section both should return a value which is of the datatype defined in the header section.

How to execute a Function?
A function can be executed in the following ways.

- As a part of a SELECT statement : SELECT emp_details_func FROM dual;
- In a PL/SQL Statements like, : dbms_output.put_line(emp_details_func);

This line displays the value returned by the function .

Example:

```
create or replace function getsal (no IN number) return number
is
 sal number(5);
begin
 select salary into sal from emp where id=no;
 return sal;
end;
/
```

Output:

Run SQL Command Line

```
SQL>select * from emp;
  ID    SALARY
-----  --------
   2      5000

SQL>start D://fun.sql
Function created.

SQL>select getsal(2) from dual;
GETSAL(2)
---------
    5000
```

In the example we are retrieving the 'salary' of employee with id 2 to variable 'sal'. The return type of the function is number.

Destroying procedure and function :

Syntax:

```
DROP PROCEDURE/FUNCTION PROCEDURE/FUNCTION_NAME;
```

Procedures VS Functions:

- A function MUST return a value
- A procedure cannot return a value
- Procedures and functions can both return data in OUT and IN OUT parameters
- The return statement in a function returns control to the calling program and returns the results of the function
- The return statement of a procedure returns control to the calling program and cannot return a value
- Functions can be called from SQL, procedure cannot
- Functions are considered expressions, procedure are not

SCREENSHOTS:

```
mysql> use warehouse;
Database changed
mysql> create table products(p_id int not null auto_increment,p_name varchar(20) not null,cost float not null default 0.0,price float not null
default 0.0, primary key(p_id));
Query OK, 0 rows affected (0.01 sec)

mysql> insert into products values("Basic widget",5,8),("Micro widget",1.50,3.5),("Mega widget",100,200);
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> insert into products (p_name,cost,price) values("Basic widget",5,8),("Micro widget",1.50,3.5),("Mega widget",100,200);
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select * from warehouse;
ERROR 1146 (42S02): Table 'warehouse.warehouse' doesn't exist
mysql> select * from products;
+------+-------------+------+-------+
| p_id | p_name      | cost | price |
+------+-------------+------+-------+
|    1 | Basic widget |    5 |     8 |
|    2 | Micro widget |  1.5 |   3.5 |
|    3 | Mega widget  |  100 |   200 |
+------+-------------+------+-------+
```

```
mysql> select *,calcProfit(cost,price) as profit from products;
+------+--------------+------+-------+--------+
| p_id | p_name       | cost | price | profit |
+------+--------------+------+-------+--------+
|    1 | Basic widget |    5 |     8 |   3.00 |
|    2 | Micro widget |  1.5 |   3.5 |   2.00 |
|    3 | Mega widget  |  100 |   200 | 100.00 |
+------+--------------+------+-------+--------+
3 rows in set (0.01 sec)
```

```
mysql> call procedureTest() \g
+--------------+
| p_name       |
+--------------+
| Basic widget |
| Micro widget |
| Mega widget  |
+--------------+
```

Conclusion :- Hence, we studied the concepts of creating a function and procedure and performing operation on the table using those functions on MySQL server.

Department of Information Technology

Academic Year: 2023-24
Semester: III
Class / Branch: SE-IT

[redacted]

Experiment No:10

Aim: To study and implement cursors in database.

Theory: A cursor allows to iterate a set of rows returned by a query and process them accordingly. Cursors can be created inside a stored procedure to handle the result set returned by a query. Mysql cursors are read-only and non-scrollable. Cursors process the results set of returned by a query row by row.

Steps for working with cursors:

- Declare <cursor_name> for select_statement
- Open <cursor_name> initializes the result set for operation
- Fetch curosr_name into variable list to retrieve the next row pointed by the cursor and move the cursor to the next in the result set.
- Close <cursor_name> to deactivate the cursor and release any memory associated with it.

Paste the Screenshots:

```
mysql> select * from emp;
+------+----------+-------------------+---------+--------+
| e_no | e_name   | job               | dept_no | sal    |
+------+----------+-------------------+---------+--------+
|    1 | Purav    | Software Engineer |       2 | 100000 |
|    2 | Rahul    | Data Scientist    |       3 |  80000 |
|    3 | Abhishek | Manager           |       4 |  50000 |
+------+----------+-------------------+---------+--------+
3 rows in set (0.00 sec)
```

```
mysql> delimiter $$
mysql> create procedure proc_emp2()
    -> begin
    -> declare v_name varchar(90);
    -> declare v_salary int;
    -> declare v_finished integer default 0;
    -> declare c1 cursor for select e_name,sal from emp;
    -> declare continue handler for NOT FOUND set v_finished=1;
    -> open c1;
    -> get_emp: LOOP
    -> fetch c1 into v_name,v_salary;
    -> if v_finished=1 then
    -> leave get_emp;
    -> end if;
    -> select concat(v_name,v_salary);
    -> end loop get_emp;
    -> close c1;
    -> end $$
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> call proc_emp2()$$
+--------------------------+
| concat(v_name,v_salary) |
+--------------------------+
| Purav100000              |
+--------------------------+
1 row in set (0.00 sec)

+--------------------------+
| concat(v_name,v_salary) |
+--------------------------+
| Rahul80000               |
+--------------------------+
1 row in set (0.00 sec)

+--------------------------+
| concat(v_name,v_salary) |
+--------------------------+
| Abhishek50000            |
+--------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

<u>Conclusion:</u> Hence, we understood the concpt of creating a cursor and calling it to manipulate the data of the table and display the selected rows using cursor when called using MySQL server.

EXPERIMENT NO. 11

Class/ Branch : SEIT-C
Semester: III

Subject: SQL Lab
Name of Instructor: Prof. Charul Singh

[redacted]

Aim: To write SQL program using FOR loop to insert rows in a data-base table.

Theory: To insert rows into a database table using Java, you need to use a combination of Java code and SQL queries. Java provides libraries to interact with databases, and you'll need to use JDBC (Java Database Connectivity) to execute SQL queries.

To insert rows into a database table using Java, you typically use JDBC (Java Database Connectivity), which provides a standard way to interact with relational databases. Here's a step-by-step guide on how to do this:

1. **Import the Required Libraries**
2. **Establish a Database Connection**
3. **Prepare the SQL Insert Statement**
4. **Set Parameter Values**
5. **Execute the Insert Statement**
6. **Handle Exceptions**
7. **Close Resources**

Here's an example Java program that uses a FOR loop to insert rows into a database table. For this example, I'll assume you are using MySQL as the database and have already set up the JDBC driver for MySQL:

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.ArrayList;
public class MultipleInsert {

    public static void main(String[] args) throws ClassNotFoundException, SQLException {

        //Create list of Employee to add in DB
        Employee emp1 = new Employee(5001, "John", "Doe",  "Software Engineer");
        Employee emp2 = new Employee(5002, "John1", "D1", "Data Analyst");

        ArrayList<Employee> employeeList = new ArrayList<Employee>();
        employeeList.add(emp1);
        employeeList.add(emp2);

        //Create Db Connection
        Class.forName("com.mysql.cj.jdbc.Driver");
        String url = "jdbc:mysql://localhost:3306/<schema name>";
        String user = "***";
        String password = "***";
        Connection connection = DriverManager.getConnection(url, user,password);

        String sql = "insert  into employees (`employeeNumber`,`lastname`,`firstname`,`jobtitle`) values (?,?,?,?)";

      //Prepare the statement
        PreparedStatement preparedStatement = connection.prepareStatement(sql);

        for(Employee employee : employeeList) {
            //bind java data to jdbc statement
            preparedStatement.setInt(1, employee.employeeId);//bind 1
            preparedStatement.setString(2, employee.lastName);// bind 2
            preparedStatement.setString(3, employee.firstName);// bind 3
            preparedStatement.setString(4, employee.jobTitle);// bind 4

            // execute Query
            int count = preparedStatement.executeUpdate();

            if (count > 0) {
                System.out.println("Employee successfully added to database :: "+employee.employeeId);
            }
        }
```

```
Problems  Javadoc  Declaration  Console
<terminated> MultipleInsert [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe  (02-Oct-2023, 1:50:41 pm – 1:50:42 pm) [pid: 9136]
Employee successfully added to database :: 5003
Employee successfully added to database :: 5004
Employee successfully added to database :: 5005
```

employeeNumber	firstName	lastName	jobTitle
5002	John1	D1	Data Analyst
5003	John	Doe	Software Engineer
5004	John1	D1	Data Analyst
5005	John2	D2	Data Senior Manager

Conclusion: This program will use a FOR loop to insert ten rows into the specified database table. It will create a connection to the database, prepare an SQL INSERT statement with placeholders for the name and age, and then execute the query inside the loop, inserting one row at a time.