List of Practical's

Sr. No.	Name of Practical	Date	Page No.	Remarks
1.	Introduction to Database System, Structured Query Language (SQL), installation procedure of MySQL on Ubuntu and also write SQL queries to implement Data Definition Language commands.			
2.	Write SQL queries to implement Data Manipulation Language commands			
3.	Perform the operations based on select command using where, group by, order by and having clause to retrieve data from database.			
4.	Perform basic operations using arithmetic, comparison, and logical operators.			
5.	Perform the operation to combine rows from two or more tables, based on a related column between them using JOIN clause.			
6.	Create the database Trigger in SQL.			
7.	Implement Transaction Operations (Commit, Rollback, and Savepoint)			
8.	Implement lock-based protocols for concurrency control.			
9.	Develop an application to retrieve the information by connecting to the database using a host language (JAVA, C, C++) (Mini Project)			

Practical No. 1

Aim: Introduction to Database System, Structured Query Language (SQL), installation procedure of MySQL on Ubuntu and also write SQL queries to implement Data Definition Language commands.

Software Required: MySQL

Theory:

Database Systems:

A database is a collection of interrelated data and set of application programs used to access, update and manage that data. The goal of database system is to provide an environment that is both convenient and efficient to use in retrieving information from the database and storing information into the database. Databases are usually designed to manage large bodies of information. This involves:

- Definition of structures for information storage (Data modelling).
- Provision of mechanisms for the manipulation of information (File and system structure, query processing).
- Providing for the safety of information in the database (crash recovery and security).
- Concurrency control if the system is shared by the users. (Because database supports concurrent access to data).

Structured Query Language:

SQL (pronounced "lss-que-el") stands for Structured Query Language is used to communicate with a database. According to ANSI (American Nation Standard Institute), it is the standard language for relational database management system. SQL statements are used to perform tasks such as update data on a database or retrieve data from a database. Some common relational database management system that uses SQL are: Oracle, Sybase, Microsoft SQL server, Access, Ingres etc.

Although most of them also have their own additional proprietary extensions that are usually only used on their system. However, the standards SQL commands such as, "select", "Insert", "Update", "Delete", "Create", and "Drop" can be used to accomplish almost everything that one needs to do with database. SQL was standardized first by the ANSI and later by the ISO.

The first version of SQL was developed at IBM by Andrew Richardson, Donold C. Messerly and Raymond F Boyce in the early 1970's. This version initially called SEQUEL, was designed to manipulate and retrieve data stored in IBM's original relational database product, system R. IBM planted their version of SQL in 1985, while the SQL language was not formally standardized until 1986 by the American Nation Standard Institute (ANSI) as SQL-86. Subsequent versions of the SQL standard have been released by ANSI and as International organization for standardization (ISO) standards.

Domain (Data type) types in SQL char (n). Fixed length character string with user-specified.

- varchar (n)- Variable length character string, with user-specified maximum.
- int- Integer (a finite subset of the integer that is machine- dependant)
- smallint- small integer (a machine- dependant subset of the integer domain type)
- numeric (p, d)- Fixed point number, with user-specified precision of p digits, with n digits to right of decimal point.
- real, double precision- Floating point and double- precision floating point numbers, with machine dependent precision.
- float (n)- Floating point number with user-specified precision of at least n digits.

Commands in SQL can be divided into sublanguages such as-

DDL:

Data Definition Language (DDL) statements are used to define the database structure or schema.

Some examples:

- CREATE- to create objects in database.
- ALTER- to alter the structures of database.
- DROP- to delete objects from database.
- TRUNCATE- remove all records from a table including all spaces allocated for records are remove.
- RENAME- to rename an object.

DML:

Data Manipulation Language (DML) statements are used for managing data within schema objects.

Some examples:

- SELECT- retrieve data from a database.
- INSERT- inserts data into a table.
- UPDATE- updates existing data within a table.
- DELETE- deletes all records from a table, the space for records remain.
- LOOK_TABLE- control concurrency.

DCL:

Data Control Language (DCL) statements are used to control the data.

Some example:

- GRANT- gives users access privileges to database.
- REVOKE- withdraws access privileges given with the GRANT commands.

TCL:

Transaction control (TCL) statements are used to manage the changes made by DML statements. It allows statements to be grouped together into logical transactions.

Some example:

- COMMIT- saves work done.
- SAVEPOINT- identifies a point in a transaction to which you can later roll back.
- ROLLBACK- restore database to original since the last COMMIT.
- SETTRANSACTION- change transaction options like isolation level and what rollback segment to use.

Installation of MySQL on Ubuntu:

MySQL is an open-source database management system, commonly installed as part of the popular LAMP (Linux, Apache, MySQL, PHP/Python/Perl) stack. It uses a relational database and SQL (Structured Query Language) to manage its data.

The short version of the installation is simple: update your package index, install the mysql-server package, and then run the included security and database initialization scripts.

Step 1 — **Installing MySQL**

sudo apt-get update

sudo apt-get install mysql-server

Step 2 — **Configuring MySQL**

sudomysql_secure_installation

Step 3 --- **Checking version of MySQL**

mysql --version

If version of MySQL is earlier than 5.7.6 then initialize the data directory by running

sudomysql_install_db.
Step 4 — **Testing MySQL**
service mysql status

Database Languages:
Database languages are used for read, update and store data in database. There are several such languages that can be used for such purpose, one of them is SQL.
Database language can be divided into two parts:
1) The Data Definition Language (DDL)
2) The Data Manipulation Language (DML)

Data Definition Language (DDL)
The DDL part of SQL permits database tables to be created or deleted. It also defines indexes (keys), specifies links between tables, and imposes constraints between tables. The most important DDL statements in SQL are:

- **CREATE DATABASE** - creates a new database
- **CREATE TABLE** - creates a new table
- **ALTER TABLE** - modifies a table
- **DROP TABLE** - deletes a table

1) CREATE DATABASE
It is used to create a new database.
Syntax: CREATE DATABASE database-name;
For example..
Create database DCSE;
To use created database we use query as
Syntax: use database-name;
For example: use DCSE;

2) CREATE TABLE
It is used to insert a new table inside database.
Syntax: CREATE TABLE table-name
(column_name1 data-type,
column_name2 data-type,
column_name3 data-type);
For example:
CREATE TABLE student
(ID int, LastNamevarchar(255),
FirstNamevarchar(255),
Address varchar(255),
City varchar(255));

ID	LastName	FirstName	Address	City

3) ALTER TABLE
This statement is used to add, delete, or modify columns in an existing table.
A) To add a column in a table:
Syntax: **ALTER TABLE** table-name

ADD column-name **data-type;**

B) To delete a column in a table:

Syntax: **ALTER TABLE** table-name
DropColumn column-name **data-type;**

C) To change the data type of a column in a table:

Syntax: **ALTER TABLE** table-name
MODIFY column-name **data-type;**

For example.

ALTER TABLE person
ADDDateofBirth date;

Output:

ID	LastName	FirstName	Address	City	DateofBirth

4) Drop Table

The DROP TABLE statement is used to delete a table.

Syntax: **DROP TABLE** table-name;

For example: DROP TABLE person;

Task to be performed:-

- Create a table EMPLOYEE, with the column EMPNO, DEPTNO, EMPNAME, JOINING DATE, AGE, DESIGNATION, SALARY, GENDER, CITY, DISTRICT, and COUNTRY.
- Alter the table EMPLOYEE by adding another column named as DEPARTMENTNAME.
- Change the field EMPNO with EMPLOYEENUM.
- Rename the table EMPLOYEE with name EMPDATA.
- Truncate the EMPDATA table & Drop EMPDATA table.

Queries:

Viva questions:-

1. What is a Database system?
2. What are the advantages of DBMS?
3. What are the disadvantages of File Processing System?
4.What is Data Independence & its types?
5. What is Schema?

Conclusion:

PRACTICAL NO. 2

Aim: Write SQL queries to implement Data Manipulation Language commands.

Software required: MySql

Theory:-

Data Manipulation Language (DML) :

The DML statement are used to work with the table**.** The query and update commands form the DML part of SQL:

- **INSERT INTO** -To inserts new data into a database
- **UPDATE**–To updates data in a database
- **DELETE** -To deletes data from a database
- **SELECT**–To extracts data from a database

1) Insert into

The INSERT INTO statement is used to insert a new row in a table.

Syntax: **INSERT INTO** table-name
VALUES (value1, value2, value3…)

INSERT INTO person
VALUES (1,'Sumera', 'Ahmad', 'CP Road', 'Amravati')

Output:

ID	FirstName	LastName	Address	City
1	Sumera	Ahmad	CP Road	Amravati

Similarly, we can insert values every time by using **Insert Into** as shown below,

```
INSERT INTO person
VALUES (2,'Tarun', 'Varma', 'Grand Park', 'Connecticut')
INSERT INTO person
VALUES (3,’Roshan’, 'Karwa', 'Panama Street', 'New York')
INSERT INTO person
VALUES (4,'Yogita', ‘Alone', 'Antique Palace', 'Portland')
INSERT INTO person
VALUES (5,'Gaurav', 'Sawale', 'Tornado Hill', 'Rome')
```

Output:

ID	LastName	FirstName	Address	City
1	Ahmad	Sumera	CP Road	Amravati
2	Tarun	Varma	Grand Park	Connecticut
3	Roshan	Karwa	Panama Street	New York
4	Yogita	Alone	Antique Palace	Portland
5	Gaurav	Sawale	Tornado Hill	Rome

2) Update statement

The UPDATE statement is used to update existing records in a table.

Syntax: **UPDATE** table-name
SET column1=value, column2=value2...
WHERE some-column=some-value

For example:

Now we want to **update** the **address** of a person named AloneYogita in the **employee** table

We use the following SQL statement:

UPDATE person
SET Address='Alumina Street', City='Washington'
WHERELastName='Yogita' AND FirstName='Alone'

Output:

ID	FirstName	LastName	Address	City
1	Sumera	Ahmad	CP Road	Amravati
2	Tarun	Varma	Grand Park	Connecticut
3	Roshan	Karwa	Panama Street	New York
4	Yogita	Alone	**Alumina Street**	**Washington**
5	Gaurav	Sawale	Tornado Hill	Rome

Note: Notice the WHERE clause in the UPDATE syntax. The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

3) Delete Statement

The DELETE statement is used to delete rows in a table.

Syntax: **DELETE FROM** table-name
WHERE some-column=some-value

For example:

Now we want to **Delete** the person named AloneYogita in the **employee** table

We use the following SQL statement:

DELETE FROM person
WHERELastName='Yogita' AND FirstName='Alone'

Output:

ID	FirstName	LastName	Address	City
1	Sumera	Ahmad	CP Road	Amravati
2	Tarun	Varma	Grand Park	Connecticut
3	Roshan	Karwa	Panama Street	New York
5	Gaurav	Sawale	Tornado Hill	Rome

Note: Notice the WHERE clause in the DELETE syntax. The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

4) Select Statement

The SELECT statement is used to select data from a database. The result is stored in a result table, called the result-set.

Syntax: **SELECT** column-name(s)
FROM table-name
OR
SELECT * From table-name

Note: The asterisk **(*)** is a quick way of selecting all columns!

For Example:

A) Now we want to select the columns named LastName and FirstName from the table given above. We use the following SELECT statement:

SELECT LastName, FirstName**FROM** employee

Output:

FirstName	LastName
Sumera	Ahmad
Tarun	Varma
Roshan	Karwa
Gaurav	Sawale

B) Now we want to select all the columns from the table above.
We use the following SELECT statement:

SELECT * FROM person;

Output:

ID	FirstName	LastName	Address	City
1	Sumera	Ahmad	CP Road	Amravati
2	Tarun	Varma	Grand Park	Connecticut
3	Roshan	Karwa	Panama Street	New York
5	Gaurav	Sawale	Tornado Hill	Rome

SQL INTEGRITY CONSTRAINTS:

Integrity Constraints are used to apply business rules for the database tables. It is used to prevent invalid data entry into table.

Types of SQL Integrity Constraints:

1. Domain Integrity Constraint:
   - NOT NULL
   - CHECK

2. Entity Integrity Constraint:
   - PRIMARY KEY
   - UNIQUE KEY

3. SQL FOREIGN KEY OR REFERENTIAL INTEGRITY CONSTRAINT:
4. ON DELETE CASCADE

1. Domain Integrity Constraint:

It is used to maintain value according to the user specifications.

- **NOT NULL:**

Not Null constraint ensures all rows in the table contain a definite value for the column which is specified as not null. Which means a null value is not allowed.

Case A: Syntax to define a Not Null constraint at table Creation:

CREATE TABLE <table_name>(column_name1 datatype(size), column_name2 datatype(size) NOT NULL, …… column_name1 datatype(size));

Case B: Syntax to define a Not Null constraint at table Alteration:

ALTER TABLE <table_name> modify <column_name> not null

- **CHECK** :

Check constraint defines a business rule on a column. All the rows must satisfy this rule. The constraint can be applied for a single column or a group of columns.

Syntax to define a Check constraint:

CREATE TABLE <table_name>(column_name1 datatype(size), column_name2 datatype(size) CHECK(<column_name2> in ('value1','value2', ……), ….. column_namendatatype(size));

2. Entity Integrity Constraint:

- **PRIMARY KEY:**

A primary key is a field in a table which uniquely identifies each row/record in a database table. Primary keys must contain unique values. A primary key column cannot have NULL values. A table can have only one primary key, which may consist of single or multiple fields.

Case A: Syntax to define a Primary key at Table creation:

CREATE TABLE <table_name>(column_name1 datatype(size), column_name2 datatype(size) constraint <constraint_name> primary key,…..);

Case B: Syntax to define a Primary key at table Alteration:

ALTER TABLE <table_name> add constraint <constraint_name> primary key(column_name);

3. SQL FOREIGN KEY OR REFERENTIAL INTEGRITYCONSTRAINT:

The constraint identifies any column referencing the PRIMARY KEY in another table. It establishes a relationship between two columns in the same table or between different tables. For a column to be defined as a Foreign Key, it should be a defined as a Primary Key in the table which it is referring. One or more columns can be defined as Foreign key.

Case A: Syntax to define a **Foreign** key at Table creation:

CREATE TABLE <table_name>(column_name1 datatype(size), column_name2 datatype(size) references <parent_table_name> (parent_table_column_name),…….);

Case B: Syntax to define a **Foreign** key key at table Alteration:

ALTER TABLE <table_name> add constraint <constraint_name>**Foreign** key (column_name) references <parent_table_name> (parent_table_column_name);

Task to be performed:-

Insert at least 5 records in a table EMPLOYEE. Update the table EMPLOYEE using 'SET' and 'WHERE' keyword. Delete the specific row on the basis of particular condition. Explore keys on considered table. Also explore integrity constraints.

Viva questions:-

1. What is integrity Constraint?
2. What is a Null value?
3. What is Surrogate key?
4. What is referential integrity Constraint?

Conclusion:-

PRACTICAL NO. 3

Aim:-Perform the operations based on select command using where, group by, order by and having clause to retrieve data from database.

Software required: MySQL

Theory:-

1) SQL SELECT

The SQL SELECT statement returns a result set of records from one or more tables. A SELECT statement retrieves zero or more rows from one or more database tables or database views. In most applications, SELECT is the most commonly used data query language (DQL) command.

Basic SQL structure is as follows:

SELECT "column name" FROM "table name"

To illustrate the above example, Consider following table:

Table **Store_Information**

Store_name	Sales	Date
Los Angeles	$1500	Jan-05-1999
San Diego	$250	Jan-07-1999
Los Angeles	$300	Jan-08-1999
Boston	$700	Jan-08-1999

To select all the stores in this table, we key in,

SELECT Store_name FROM Store_ Information

Result:

Store_name
Los Angeles
San Diego
Los Angeles
Boston

Multiple column names can be selected, as well as multiple table names.

2) GROUP BY Clauses:

The GROUP BY is used **to** group values from a column, and to perform calculations on that column, use COUNT, SUM, AVG etc. function on the grouped column. To count number of days each employee did work. This is done by using aggregate functions in conjunction with a **GROUP BY** clause as follows:

Syntax:- SELECT column_name ,aggrfun(column)
FROM <table1>**GROUP BY** column_name;

Ex. SELECT name, COUNT(*)
FROM employee_tbl
GROUP BY name;

3) ORDER BY Clauses:

The SQL ORDER BY Keyword. The ORDER BY keyword is used to sort the result-set in ascending or descending order. The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

Syntax:- SELECT * FROM <table1>
ORDER BY(column-name) ASC|DESC;

EX. SELECT * FROM employee_tblORDER BY(pages);

4) HAVING Clauses:

The mysql HAVING clause is an optional part of and used only with the SQL SELECT statement. The mysql HAVING clause specifies a filter condition for a group of record or an aggregate. The mysql HAVING is often used with mysql GROUP BY clause If the mysql GROUP BY clause is omitted, the mysql HAVING clause will behave like a WHERE clause.

Syntax:- SELECT column_name , aggrfun(column) FROM <table1> GROUP BY column_name HAVING column_name> value;

Ex. SELECT ordernumber, sum(quantityOrdered) AS itemsCount, sum(priceeach) AS total FROM orderdetails GROUP BY ordernumber HAVING total > 1000;

Task to be performed:-

Table EMPLOYEE, with the column EMPNO, DEPTNO, EMPNAME, JOINING DATE, AGE, DESIGNATION, SALARY, GENDER, CITY, DISTRICT, and COUNTRY.

1. Retrieve data from EMPLOYEE table with salary in ascending order.
2. Display designation, minimum salary, maximum salary, average salary, salary sum of an employee of each designation.
3. Display employee name, dept no, total salary of each department with total salary is greater than 6000.
4. Display how many employees are in the table.
5. Display average salary in ascending order of each department where department salary between 5000 to 10000;
6. Find how many employees having same age.
7. Find name, designation, emp no, whose joining before year 2013 and display them in their descending order of their DOJ.
8. Find how many employees from each city.

Viva questions:-

1) What is the use of SELECT command?
2) How will you use GROUP BY clause?
3) What is the use of ORDER BY?
4) What is basic structure of SQL?
5) What is difference between WHERE & HAVING Clause?

Conclusion:-

PRACTICAL NO. 4

Aim:- Perform basic operations using arithmetic, comparison, and logical operators.
Software required: MySQL
Theory:-
Operator :

An operator manipulates individual data items and returns a result. The data items are called operands or arguments. Operators are either special words or characters that are used to execute an operation

For example, the multiplication operator is represented by an asterisk (*) and the operator that tests for nulls is represented by the keywords IS NULL .

Types of Operators:

There are three types of operators:

1. Arithmetic Operators.
2. Comparison Operators
3. Word Comparison Operators
4. Logical Operators.

1. Arithmetic Operators :

The Arithmetic operator is used in an arithmetic expression in SQL to perform calculations on attributes which have data type defined as number.

+ addition
- subtraction
* multiplication
/ division

Note: precedence of operators is as given below

1. ( )
2. *
3. /
4. +
5. –

2. Comparison Operators:

It is used to compare one expression with other using comparison operators.

Symbolic operators

= Equal to
!= Not Equal to
< Less than
> Greater than
<= Less than Equal to
>= Greater than Equal to

3. Word Comparison operators :

1. Between :

Betweenoperator is used to select values that are within the range of values specified inclusive of min value and max value. The range which is specified will contain min value and max value.

It is used for numeric and date data type.

2. Not Between:

Not Between operator is used to select values that are exclusive i.e. those not having values in the specified range and also not having values equal to min value and max value.

It is used for numeric and date data type.

3. Like :

Like operator is used for pattern searching pattern consist of characters to be matched. Pattern searching is done by using wild card characters. Wild card character Matches % (Percentage) It matches any sequence of string.

4. Not Like :

It will display all rows from table student_info where data does not match with the specified condition.

5. In :

In operator is used to find whether the attribute value is within specified list or not i.e. we check the single value against the multiple value in the list.

6. Not In :

Not In operator is used to find whether the attribute value is not within specified list. In and Not In operators can be used with any data type. If character or date is specified it should be enclosed in quotation.

7. Is Null :

Is Null operator checks for the value which is Null. Null value means no value or an unavailable, unassigned, unequal to any value or zero length of string in a data cell.

8. Is Not Null :

The operator checks for the value which is not null. Not Null value means the data cell consist of some logical value.

4) Logical Operators :

A Logical Operator is used to combine the results of two conditions to produce a single result.Boolean operators AND OR , NOT are used to combine logical expressions. Logical expression means it is either True or False.

Assume that A and B are two logical expression.

1. AND : AND means both

Then A AND B is True if and only both A and B are true.

2. OR : OR means any one or both

Thus A OR B is True if either A is true or B is true (or Both A and B are .True).

3. NOT : NOT is used to negate the value of expression

Thus if A is True , Not A is false .AND and OR are binary operators which operates on two expression. NOT is unary operator which operates on single operation.

Task to be performed:-

1) Display employee ID, name, designation, salary and incremented salary which is incremented by 20% to those employees who having salary greater than 10000 and having post as clerk.
2) Calculate double salary of employee from EMPLOYEE who lives in pune.
3) Find name of employee whose salary is not equal to 4000 and equal to 10000.
4) Retrieve data of employee whose salary is less than 20000 and greater than 5000
5) Retrieve data of employee whose salary is not greater than 7000 (including).
6) Display data of employee whose name starts with ‘a’ and his city name ends with ‘e’.
7) Display all data of employee whose name starts with ‘c’ and his district name not contain character ‘ag’.
8) Display employee data from employee table whose values are unknown.
9) Retrieve data of employee whose salary is greater than 7000 and designation as a clerk.

10) Retrieve employee name, designation and salary of employee, either their salary is greater than 10000 or designation as a clerk.

Viva questions:-

1. What is different comparison operators used?
2. What is the need of Operator?
3. What is word comparison operator?
4. What is the purposeword comparison operator?
5. What is difference between ‘IN’ and ‘BETWEEN’ operator?
6. What is difference between ‘AND’ and ‘OR’ logical operator?

Conclusion: -

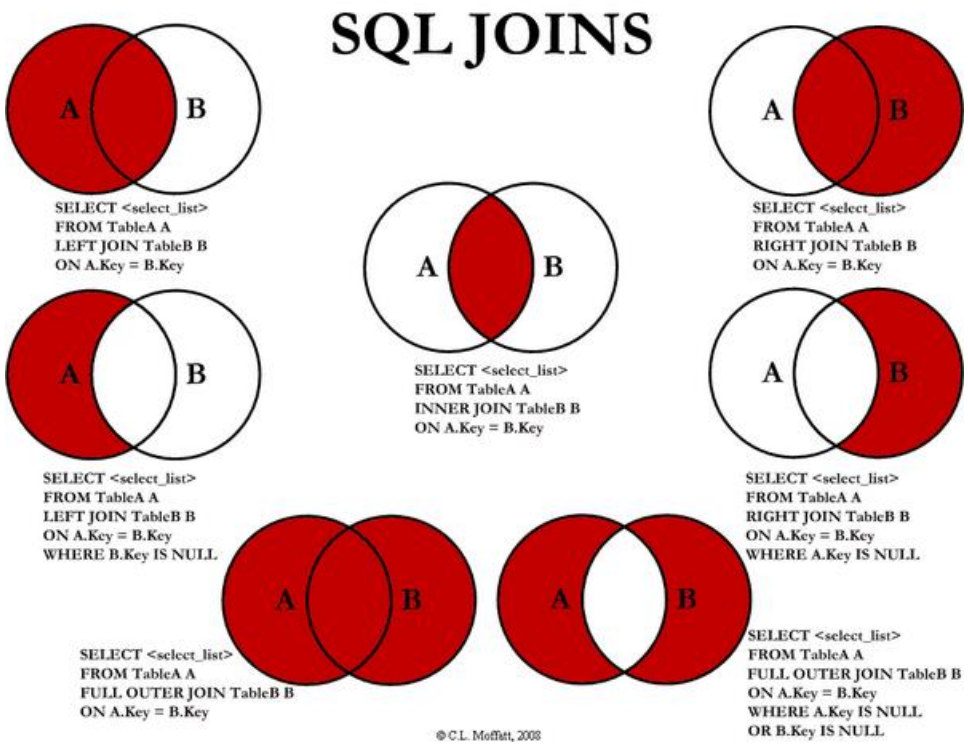
PRACTICAL NO. 5

Aim:- Perform the operation to combine rows from two or more tables, based on a related column between them using JOIN clause.

Software required: MySql

Theory:-

A join condition is a part of the sql query that retrieves rows from two or more tables. A SQL Join condition is used in the SQL WHERE Clause of select, update, delete statements.



Consider following tables

A) person

P-ID	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

B) order

O-ID	OrderNo	P-ID
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	15

Types of Join:-

1) CROSS JOIN

It joins two table and display result in multiple of both table rows.

Syntax. SELECT <column_name> FROM <table1> cross join <table2>;

-Select FirstName, orderno from person cross join order.

2) EQUI JOINS

It is a simple sql join condition which uses the equal sign as the comparison operator. Two types of equi joins are

i. **INNER JOIN** : joins tables with same data.

Syntax:-SELECT column_name(s) FROM table_name1 INNER JOIN table_name2 ON table_name1.column_name= table_name2.column_name;

-Select person.firstname,order.orderno from person inner join order on person.pid=order.pid;

FirstName	OrderNo
Ola	22456
Ola	24562
Kari	77895
Kari	44678

ii. **OUTER JOIN**

It avoids the data losing while joining the tables.

a) Left Outer Join:

The left join is a mechanism used to join tables before we add other conditions such as WHERE etc.The Left Outer Join keyword returns all rows from the left table, even if there are no matches in the right table.

Syntax.

SELECT <column_name> FROM <Table1>
LEFT JOIN <Table2> ON Table1.column = Table2.column;

-Selectperson.firstname,order.orderno from person left join order on person.pid=order.pid;

FirstName	OrderNo
Ola	22456
Ola	24562
Kari	77895
Kari	44678
Tove	Null

b) Right outer join:

The right join is a mechanism used to join tables beforewe add other conditions such as WHERE etc. The RIGHT JOIN keyword returns all the rows from the right table, even if there are no matches in the left table.

Syntax.

SELECT <column_name> FROM <Table1>
RIGHT JOIN <Table2> ON Table1.column = Table2.column;

-Select person.firstname, order.orderno from person right join order on person.pid=order.pid;

FirstName	OrderNo
Ola	22456
Ola	24562

Kari	77895
Kari	44678
Null	34764

Task to be performed:

Create following table and apply all join operation to retrieve data.

Order (ordered, customerid, orderdate)
Customer (customerid, customername, contactname, country)

Viva questions:-

1) What is mean by join operation?
2) What is equi join?
3) Differentiate between inner join and outer join.
4) What is natural join and cross join?
5) Give advantages of using full outer join over left/right outer join.

Conclusion: -

PRACTICAL NO. 6

Aim:- Create the database Trigger in SQL.

Software required: MySql

Theory:-

A trigger is a statement that the system executes automatically as a side effect of a modification to the database. To design a trigger mechanism, there are two requirements:

1.Specify when a trigger is to be executed. This is broken up into an event that causes the trigger to be checked and a condition that must be satisfied for trigger execution to proceed.

2. Specify the actions to be taken when the trigger executes.

The above model of triggers is referred to as the event-condition-action modelfor triggers.

The database stores triggers just as if they were regular data, so that they are persistent and are accessible to all database operations. Once we enter a trigger into the database, the database system takes on the responsibility of executing it whenever the specified event occurs and the corresponding condition is satisfied. Triggers are useful mechanisms for alerting humans or for starting certain tasks automatically when certain conditions are met. A trigger is a pl/sql block structure which is fired when a DML statements like Insert, Delete, Update is executed on a database table. A trigger is triggered automatically when an associated DML statement is executed.

The Syntax for creating a trigger is:

CREATE [OR REPLACE] TRIGGER trigger_name

{BEFORE | AFTER | INSTEAD OF }

{INSERT [OR] | UPDATE [OR] | DELETE}

[OF col_name]

ON table_name

[REFERENCING OLD AS o NEW AS n]

[FOR EACH ROW]

WHEN (condition)

BEGIN

--- sql statements

END;

- CREATE [OR REPLACE] TRIGGER trigger_name- This clause creates a trigger with the given name or overwrites an existing trigger with the same name.
- {BEFORE | AFTER | INSTEAD OF } - This clause indicates at what time should the trigger get fired. i.e for example: before or after updating a table. INSTEAD OF is used to create a trigger on a view. before and after cannot be used to create a trigger on a view.
- {INSERT [OR] | UPDATE [OR] | DELETE} - This clause determines the triggering event. More than one triggering events can be used together separated by OR keyword. The trigger gets fired at all the specified triggering event.
- [OF col_name] - This clause is used with update triggers. This clause is used when you want to trigger an event only when a specific column is updated.
- [ON table_name] - This clause identifies the name of the table or view to which the trigger is associated.
- [REFERENCING OLD AS o NEW AS n] - This clause is used to reference the old and new values of the data being changed. By default, you reference the values as :old.column_name or :new.column_name. The reference names can also be changed from old (or new) to any other user-defined name. You cannot reference old values when inserting a record, or new values when deleting a record, because they do not exist.
- [FOR EACH ROW] - This clause is used to determine whether a trigger must fire when each row gets affected (i.e. a Row Level Trigger) or just once when the entire sql statement is executed(i.e.statement level Trigger).
- WHEN (condition) - This clause is valid only for row level triggers. The trigger is fired only for rows that satisfy the condition specified.

Example: The price of a product changes constantly. It is important to maintain the history of the prices of the products. We can create a trigger to update the 'product_price_history' table when the price of the product is updated in the 'product' table.

1) Create the 'product' table and 'product_price_history' table

```
-CREATE TABLE product_price_history
(product_id number(5),
product_name varchar2(15),
supplier_name varchar2(15),
unit_price number(7,2) );
```

```
-CREATE TABLE product
    (product_id number(5),
    product_name varchar2(15),
    supplier_name varchar2(15),
    unit_price number(7,2) );
```

2) Create the price_history_trigger and execute it.

```
-CREATE or REPLACE TRIGGER price_history_trigger
BEFORE UPDATE OF unit_price
ON product
FOR EACH ROW
BEGIN
INSERT INTO product_price_history
VALUES
(:old.product_id,
:old.product_name,
:old.supplier_name,
:old.unit_price);
END;
```

3) Lets update the price of a product.

```
-UPDATE PRODUCT SET unit_price = 800 WHERE product_id = 100
```

Once the above update query is executed, the trigger fires and updates the 'product_price_history' table.

4) If you ROLLBACK the transaction before committing to the database, the data inserted to the table is also rolled back.

Disable a Trigger

The syntax for a disabling a Trigger is:

ALTER TRIGGER *trigger_name* DISABLE;

Enable a Trigger

The syntax for a enabling a Trigger is:

ALTER TRIGGER *trigger_name* ENABLE;

Drop a Trigger

The syntax for a dropping a Trigger is:

DROP TRIGGER *trigger_name*;

Task to be perform: -

Implement 2 types of triggers and understand the operation thoroughly.

Viva questions:

1. List all types of triggers.
2. Write difference between triggers and assertions

Conclusion: -

Practical No – 7

Aim: Implement Transaction Operations (Commit, Rollback, and Savepoint)

Software Requirement: MySQL

Theory:

SQL – Transactions

A transaction is a unit of work that is performed against a database. Transactions are units or sequences of work accomplished in a logical order, whether in a manual fashion by a user or automatically by some sort of a database program. A transaction is the propagation of one or more changes to the database. For example, if you are creating a record or updating a record or deleting a record from the table, then you are performing transaction on the table. It is important to control transactions to ensure data integrity and to handle database errors.

Properties of Transactions:

Transactions have the following four properties, usually referred by the acronym ACID:

- **Atomicity:** ensures that all operations within the work unit are completed successfully; otherwise, the transaction is aborted at the point of failure, and previous operations are rolled back to their former state.
- **Consistency:** ensures that the database properly changes states upon a successfully committed transaction.
- **Isolation:** enables transactions to operate independently of and transparent to each other.
- **Durability:** ensures that the result or effect of a committed transaction persists in case of a system failure.

Transaction Control:

The following statements provide control over use of transactions:

- **Start Transaction:** The START TRANSACTION or BEGIN statement begins a new transaction.
- **Commit:** commits the current transaction, making its changes permanent.
- **Rollback:** rolls back the current transaction, canceling its changes.
- **Savepoint:** creates points within groups of transactions in which to ROLLBACK

Transactional control commands are only used with the DML commands INSERT, UPDATE and DELETE only. They cannot be used while creating tables or dropping them because these operations are automatically committed in the database.

Consider the CUSTOMERS table having the following records:

Id	Name	Age	Address	Salary
1	Hansen	32	Pune	2000.00
2	Svendson	25	Delhi	1500.00
3	Pettersen	23	Mumbai	2000.00

4	Ahmad	25	Indore	6500.00
5	Hayes	27	Bhopal	8500.00
6	Johnson	22	MP	4500.00
7	Jones	24	Amravati	10000.00

1. The Start Transaction:

By default, MySQL runs with auto commit mode enabled. This means that as soon as you execute a statement that updates (modifies) a table, MySQL stores the update on disk to make it permanent. The change cannot be rolled back. To disable auto commit mode, use the START TRANSACTION statement. See the following

example:

SQL > START TRANSACTION;

2. The COMMIT Command:

The COMMIT command is the transactional command used to save changes invoked by a transaction to the database. The command saves all transactions to the database since the last COMMIT or ROLLBACK command.

Syntax: COMMIT;

*No*w delete records from the customer table having age = 25 and then COMMIT the changes in the database.

Example:

SQL > START TRANSACTION;

SQL > Delete from customers where age=25;

SQL > COMMIT;

The result-set will look like this:

Id	Name	Age	Address	Salary
1	Hansen	32	Pune	2000.00
3	Pettersen	23	Mumbai	2000.00
5	Hayes	27	Bhopal	8500.00
6	Johnson	22	MP	4500.00
7	Jones	24	Amravati	10000.00

3. The ROLLBACK Command:

The ROLLBACK command is the transactional command used to undo transactions that have not already been saved to the database. The command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued.

Syntax: ROLLBACK;

NOW delete records from the customer table having age = 25 and then ROLLBACK the changes in the database.

Example:

SQL > START TRANSACTION;

SQL > Delete from customers where age=25;

SQL > ROLLBACK;
The result-set will look like this:

Id	Name	Age	Address	Salary
1	Hansen	32	Pune	2000.00
2	Svendson	25	Delhi	1500.00
3	Pettersen	23	Mumbai	2000.00
4	Ahmad	25	Indore	6500.00
5	Hayes	27	Bhopal	8500.00
6	Johnson	22	MP	4500.00
7	Jones	24	Amravati	10000.00

4. The SAVEPOINT Command:

A SAVEPOINT is a point in a transaction when you can roll the transaction back to a certain point without rolling back the entire transaction.

***Syntax*:** SAVEPOINT SAVEPOINT_NAME;

This command serves only in the creation of a SAVEPOINT among transactional statements. The ROLLBACK command is used to undo a group of transactions.

The syntax for rolling back to a SAVEPOINT is as follows:

Syntax: ROLLBACK TO SAVEPOINT_NAME;

Now you plan to delete the three different records from the CUSTOMERS table. You want to create a SAVEPOINT before each delete, so that you can ROLLBACK to any SAVEPOINT at any time to return the appropriate data to its original state:

Example:

SQL>START TRANSACTION;
SQL> SAVEPOINT SP1;
Savepoint created;
SQL> Delete from Customers where id=1;
1 row deleted
SQL> SAVEPOINT SP2;
Savepoint created;
SQL> Delete from Customers where id=2;
1 row deleted
SQL> SAVEPOINT SP3;
Savepoint created;
SQL> Delete from Customers where id=3;
1 row deleted

Now the three deletions have taken place, you want to ROLLBACK to the SAVEPOINT that you identified as SP2. Because SP2 was created after the first deletion, the last two deletions are undone:

SQL> ROLLBACK TO SP2;

Rollback Complete

Notice:Only the first deletion took place since you rolled back to SP2.

The result-set will look like this:

Id	Name	Age	Address	Salary
2	Svendson	25	Delhi	1500.00
3	Pettersen	23	Mumbai	2000.00
4	Ahmad	25	Indore	6500.00
5	Hayes	27	Bhopal	8500.00
6	Johnson	22	MP	4500.00
7	Jones	24	Amravati	10000.00

Task to be performed:

Implement transaction operation considering any suitable example

Viva Questions:

1. What are the properties of transactions?
2. What are the commands used to control transactions and explain them with example?
3. What is the difference between rollback and savepoint command?
4. How to ensure data integrity using Transaction properties?
5. How to ensure accuracy using Transaction properties?

Conclusion:

Practical No – 8

Aim: Implement lock-based protocols for concurrency control.

Software Requirement: MySQL

Theory:

Concurrency control is an essential aspect of database management systems to ensure the consistency and correctness of data in multi-user environments. Lock-based protocols are a widely used technique for concurrency control. Here's a related theory on implementing lock-based protocols for concurrency control:

- Lock Types:

Shared Lock: Multiple transactions can acquire a shared lock simultaneously. It allows read access to the locked data item but prevents other transactions from acquiring an exclusive lock on the same data item.
Exclusive Lock: Only one transaction can acquire an exclusive lock at a time. It provides both read and write access to the locked data item, preventing other transactions from acquiring any type of lock on the same data item.

- Lock Granularity:

Locks can be applied at different granularities, such as the entire database, table, page, or individual rows. The choice of granularity affects both concurrency and performance.

- Lock-Based Concurrency Control:

Before accessing a data item, a transaction must request the appropriate lock on that item. When a transaction requests a lock, the lock manager checks if the lock conflicts with any existing locks held by other transactions. If a conflict exists, the requesting transaction may be blocked and placed in a waiting state until the conflicting lock is released. Once a transaction has acquired a lock, it can proceed with read or write operations on the locked data item. When the transaction is complete, it releases the lock, allowing other transactions to access the data.

- Deadlock Detection and Handling:

Deadlocks occur when two or more transactions are waiting for locks held by each other, resulting in a circular dependency. Deadlock detection algorithms periodically examine the lock graph to identify deadlocks. Deadlock handling techniques include aborting one or more transactions involved in the deadlock or using a timeout mechanism to break the deadlock.

- Lock Escalation and Granularity Control:

Lock escalation is the process of converting fine-grained locks to coarser-grained locks to reduce overhead and improve performance. Lock granularity control involves dynamically adjusting lock granularity based on the access pattern and contention level to optimize concurrency.

- Lock Duration and Transaction Isolation Levels:

The duration for which a lock is held impacts concurrency. Shorter lock durations allow more concurrent access but may lead to increased overhead.

Transaction isolation levels (e.g., Read Uncommitted, Read Committed, Repeatable Read, Serializable) define the visibility and behavior of locks during transaction execution.
It's important to note that while lock-based protocols provide concurrency control, they can also lead to issues like deadlocks and reduced throughput due to lock contention. Therefore, choosing an appropriate lock granularity and employing deadlock detection and resolution mechanisms are crucial.

Database systems like MySQL provide built-in support for lock-based concurrency control, offering features to acquire and release locks, manage lock compatibility, and handle deadlock situations. The specific commands and mechanisms may vary based on the database system being used.

Implementing lock-based protocols requires careful consideration of the application's requirements, transaction patterns, and performance goals. Thorough testing and performance evaluation are essential to ensure effective concurrency control and efficient database operations.

Task to be performed:

Create table student with column roll number and name and observe lock-based protocols for concurrency control.

Viva Questions:

1. What is concurrency control?
2. What are two types of locks?
3. Write note on compatibility of locks?

Conclusion:

PRACTICAL NO. 09

Aim: Develop a Applications to retrieve the information by connecting to the database using a host language (JAVA, C, C++) (Mini Project)

Software required: MySQL, JDK, NetBeans

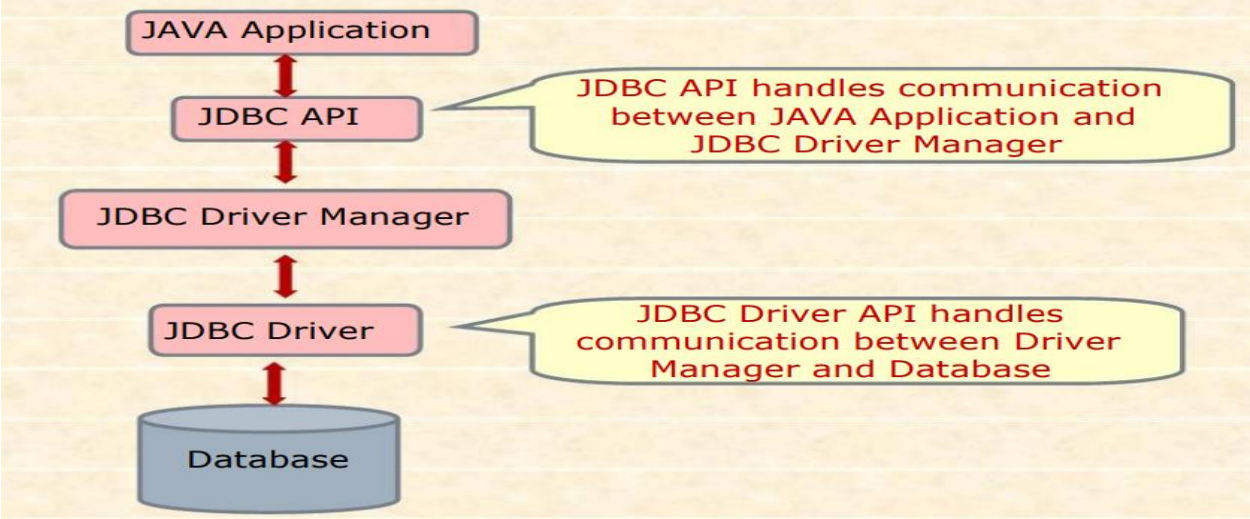
Theory:

JDBC

JDBC is JAVA's Database connection driverinterface which performs the following task for the application.

- Establish a connection with a Database.
- Send SQL request (Query) to a Database Server.
- Returns Result obtained against Query.

Some RDBMS like MS Access requires ODBC (Open Database Connection), which can be connect through JDBC-ODBC driver (jdbc.odbcbridge).



Classes used for Database Connect

The Core element of JDBC is JDBC API, which consists of a set of Java classes equipped with predefined methods to handle various data access functions such as Selecting appropriate database driver, establishing connection, submitting SQL query and processing results. JDBC API offers four main classes, which are-

- Driver Manager Class: It loads the JDBC driver to locate, logs and access a database.
- Connection Class: It manages communication between Java Client Application and Database, through SQL statements.
- Statement Class: It contains SQL commands which is submitted to the Database Server and returns ResultSet object containing the result of SQL statement.
- Result Set Class: It provides predefined methods to access and convert data values returned by the executed SQL statement.

A JDBC driver must be registered with JDBC Driver Manage using Class.forName() method before establishing a connection.

Installing JDBC Driver in NetBeans IDE

The Prerequisite for connecting a Java application to MySQL is JDBC driver (also called MySQL Connector/J). The MySQL Connector/J is freely available and can be

downloaded from the URL(dev.mysql.com/downloads/). After download it can be installed with NetBeans with help of following steps-

- Start NetBeans and Go to Tools->Libraries.
- Library Manager will be open, check MySQL JDBC Driver under Class libraries. If it is not present, you can add it by the following steps. Click on Add Jar Folder button.
- Specify downloaded uncompressed folder in the drive where JDBC is kept. Press Add Jar button and finally Click OK button.

Connecting MySQL from JAVA Application

After installing JDBC (MySQL Connector/J) Driver, you may access MySQL database through JAVA Application. The Following Six steps may be followed to establish a connection with MySQL database.

- Step 1: Import Required package/classes in the application.
- Step 2: Register the JDBC Driver to JDBC Driver Manager.
- Step 3: Open a Connection.
- Step 4: Execute a Query.
- Step 5: Extract data from Result set
- Step 6: Close Connection.

Step 1: Importing required package/classes

This step consists of two sub-steps.

- Import Java.sql Library package containing JDBC classes needed by following import statements.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.ResultSet;
```

Or import java.sql.*;

- Add MySQL JDBC connector in the application. In Project Window expand Libraries node by clicking + icon. If MySQL Connector is not present then Pres Add JAR/Folder and specify the location of Driver folder to add MySQL Driver.

Step 2: Registering the JDBC Driver

To open a Communacation channel, you require to initialize driver by registering the JDBC driver with JDBC river Manager. Java offers a Class.forName() method in java.lang package. Class.forName("java.sql.driver"); Or
Class.forName("com.mysql.jdbc.Driver");

Step 3: Opening a Connection

DriverManager.getConnection() method is used to create a connection object that represents a physical connection with database. DriverManager.getConnection() requires the complete address of the database (Database URL), user name and password as a parameter. A database URL can be formed asjdbc:mysql :// localhost/
Ex. Suppose School is a database designed in MySQL. jdbc:mysql://localhost/school
You can assign this string on a variable, which can be used later in DriverManager.getConnection() method.

```
String DB_URL = "jdbc:mysql://localhost/school";
Connection con = DriverManager.getConnection(DB_URL, "root" , "abc")
```

Step 4: Executing a Query:

You must create a Statement object for building and submitting a SQL query, using CreateStatement() method of Connection object created in Step 3.

```
Statement stmt = con.createStatement();
```

To execute a query executeQuery() method along with a valid SQL statement is used, which returns the record from the database (Result Set) on ResultSet type object.

ResultSetrs = stmt.executeQuery(“”);
The both statements can used as
Statement stmt = con.createStatement();
ResultSetrs = stmt.executeQuery(“select roll,name,class from student”);
Result Set refers to a logical set of records from the database by executing a query.
An executeUpdate() method is used in place of executeQuery() when query contains Insert, Delete or Update command.

Step 5: Extracting Data from ResultSet object

To retrieve the data from the ResultSet object, which contains records, You may use the following method.
<ResulSet object>.get<type>(<column name/number>);
Where <type>may be Int, Long, String, float etc depending on the column type of the table.
In general the data values are assigned on the variables and later used in the TextField controls of the Form using setText().
int r= rs.getInt(“roll”); String n= rs.getString(“name”);int c= rs.getInt(“class”);
That is,
int r= rs.getInt(1); String n= rs.getString(2); int c= rs.getInt(3);
Since a ResultSet object may contain more than one record, so a loop is required to process all the records. A while… loop is generally used to read all records. To break a loop .next() method is used, which returns false when all the records have been read from the Result set.

```
intr,c ; String n;
while (rs.next())
{ r= rs.getInt(“roll”); n= rs.getString(“name”); c= rs.getInt(“class”);
JOptionPane.showMessageDialog(null, “Name = “+n);
JOptionPane.showMessageDialog(null, “Roll = “+n);
JOptionPane.showMessageDialog(null, “Class = “+n); }
```

Step 6: Closing connection

After all the processing , the final step is to close the environment by closing the Connection by close() method of ResultSet, Statement and Connection objects.
rs.close(); stmt.close(); con.close();

Sample Database Connectivity Program:

```
import java.sql.*;
public class DemoJDBC
{   public static void main(String[] args) throws Exception
    {      String query= "insert into student values(9,'akash')";
Class.forName("com.mysql.jdbc.Driver");
                                        Connection con =
        DriverManager.getConnection("jdbc:mysql://localhost:3306/demo","root","root");
          Statement st = con.createStatement(); st.executeUpdate(query);
System.out.println("Inserted"); con.close();
    }
}
```

Task to be performed:

Design a mini project. Consider any front end and back end. Prepare mini project report. Attach report to this handbook properly.