2020
-
2021

ADAVACED DATABASE TECHNOLOGY LABORATORY MANUAL (MC5114)

PREPARED BY

AARTHI SIVAKUMAR. S – AARTHY SUBBURAJ. S – AAKASH. S.K – RENGA PRAVEEN KUMAR. B – SIVARANJANI. S – MUTHU SIVASANKARI. M – BALAJI. R – HAJIRAH RASHMIA. M.L– KARTHICK. G.K – NAVEEN KUMAR. U

INDEX

1. MongoDB CURD operations

AIM:

To perform CURD operations using MongoDB shell.

ALGORITHM:

1. **CREATE THE DATABASE:**

Create a database named student in the shell prompt by executing the following command.

```
> use admin switched to db admin >
show collections system.version > use
student switched to db student
```

2. **INSERT THE COLLECTIONS INTO THE DATABASE:**

Insert document into collections named student by executing the following command in the MongoDB Shell prompt.

```
> db.student.insert({"name":"gayathri","roll
no":001,"dept_name":"Maths","college":"holy cross
college","extra_curriclr":"book reading"}); WriteResult({ "nInserted" : 1 })

> db.student.insert({"name":"malini","roll no":002,"dept_name":"computer
science","college":"HCC
college","extra_curriclr":"dance"}); WriteResult({ "nInserted" : 1 })

> db.student.insert({"name":"Malathi","roll
no":003,"dept_name":"MCA","college":"ABC
college","extra_curriclr":"drawing"}); WriteResult({ "nInserted" : 1 })
```

The above document is executed succesfully,we assume it as true.

3. **READ COLLECTIONS FROM THE DATABASE:**

Read the document from the collections named student by executing the following command in the MongoDB Shell prompt.

```
> db.student.find().pretty();
{
        "_id" : ObjectId("603c0d13371e685b222cd0de"),
        "name" : "gayathri",
        "roll no" : 1,
        "dept_name" : "Maths",
        "college" : "holy cross college",
        "extra_curriclr" : "book reading"
}
{
        "_id" : ObjectId("603c0d9a371e685b222cd0df"),
        "name" : "malini",
        "roll no" : 2,
        "dept_name" : "computer science",
        "college" : "HCC college",
        "extra_curriclr" : "dance"
}
{
        "_id" : ObjectId("603c0e74371e685b222cd0e0"),
        "name" : "Malathi",
        "roll no" : 3,
        "dept_name" : "MCA",
        "college" : "ABC college",
        "extra_curriclr" : "drawing"
}

> db.student.findOne({"name":"gayathri"});
{
        "_id" : ObjectId("603c0d13371e685b222cd0de"),
        "name" : "gayathri",
        "roll no" : 1,
        "dept_name" : "Maths",
        "college" : "holy cross college",
        "extra_curriclr" : "book reading"
}
```

4. UPDATING THE COLLECTIONS INTO THE DATABASE:

We can also use the update keyword to update and alter collections using set and unset collections.

```
> db.student.update({"college":"ABC college"},{$set:{"college":"GTN college"}});

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.student.findOne({"name":"Malathi"});
{
        "_id" : ObjectId("603c0e74371e685b222cd0e0"),
        "name" : "Malathi",
        "roll no" : 3,
        "dept_name" : "MCA",
        "college" : "GTN college",
        "extra_curriclr" : "drawing"
}
> db.student.findOne({"name":"malini"});
{
        "_id" : ObjectId("603c0d9a371e685b222cd0df"),
        "name" : "malini",
        "roll no" : 2,
        "dept_name" : "computer science",
        "college" : "HCC college",
        "extra_curriclr" : "dance"
}
>db.student.update({"name":"malini"},{$unset:{"extra_curriclr":"dance"}
});

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.student.findOne({"name":"malini"});
{
        "_id" : ObjectId("603c0d9a371e685b222cd0df"),
        "name" : "malini",
        "roll no" : 2,
        "dept_name" : "computer science",
        "college" : "HCC college",
        "extra_curriclr" : "dance"
}
```

5. DELETING THE COLLECTIONS FROM THE DATABASE:

The following MongoDB delete command is used to remove the documents which belongs to the collection. db.student.deleteOne({"name":"malini"});

```
{ "acknowledged" : true, "deletedCount" : 1 }

> db.student.find().pretty();
{
        "_id" : ObjectId("603c0d13371e685b222cd0de"),
        "name" : "gayathri",
        "roll no" : 1,
        "dept_name" : "Maths",
        "college" : "holy cross college",
        "extra_curriclr" : "book reading"
}
{
        "_id" : ObjectId("603c0e74371e685b222cd0e0"),
        "name" : "Malathi",
        "roll no" : 3,
        "dept_name" : "MCA",
        "college" : "GTN college",
        "extra_curriclr" : "drawing"
}
```

RESULT:

Thus the above program is executed successfully.

2. CASSANDRA CRUD OPERATION

AIM:

To perform CRUD operation using CASSANDRA shell.

ALGORITHM:

1. To startup CASSANDRA, open the command prompt and then type Cassandra in the shell.

```
C:\Users\BALAJI>cassandra
WARNING! Powershell script execution unavailable.
   Please use 'powershell Set-ExecutionPolicy Unrestricted'
   on this user-account to run cassandra with fully featured
   functionality on this platform.
Starting with legacy startup options
Starting Cassandra Server
```

   Where the Cassandra server has startup.

2. Now open another command prompt and then type cqlsh for interacting with CASSANDRA
   QUERY LANGUAGE(CQL).

```
C:\Users\BALAJI>cqlsh

WARNING: console codepage must be set to cp65001 to support utf-8 encoding on Windows platforms.
If you experience encoding problems, change your console codepage with 'chcp 65001' before starting cqlsh.

Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.9 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
WARNING: pyreadline dependency missing.  Install to enable tab completion.
cqlsh>
```

   Where cqlsh has opened.

3. select a keyspace where the created tables are stored.

```
cqlsh> desc keyspaces;

system_schema  system       system_distributed
system_auth    sample_demo  system_traces

cqlsh> use sample_demo;
```

4. create a table named stud in the shell prompt by executing the following command in the CASSANDRA shell.

```
cqlsh:sample_demo> CREATE TABLE stud(
               ... reg int,
               ... name text,
               ... place text,
               ... ph int,
               ... PRIMARY KEY(reg));
```

Where the table is created with the field of reg,name,place,ph with their required data type were PRIMARY KEY is used in reg to avoid duplicate value.

5. Insert a values for the created table by executing the following command in the CASSANDRA shell.

```
cqlsh:sample_demo> INSERT INTO stu(reg,name,place,ph)VALUES(007,'balaji','dindigul',787878788);
cqlsh:sample_demo> INSERT INTO stu(reg,name,place,ph)VALUES(019,'haji','dindigul',868686868);
cqlsh:sample_demo> INSERT INTO stu(reg,name,place,ph)VALUES(036,'ruthran','palani',9383734353);
InvalidRequest: Error from server: code=2200 [Invalid query] message="Unable to make int from '9383734353'"
cqlsh:sample_demo> INSERT INTO stu(reg,name,place,ph)VALUES(020,'bala','natham',865686868);
cqlsh:sample_demo>
```

Where the values are inserted into the table.

6. Now read the table by using the following in the CASSANDRA shell.

```
cqlsh:sample_demo> SELECT * FROM stu;

 reg | name    | ph        | place
-----+---------+-----------+----------
  19 |    haji | 868686868 | dindigul
  20 |    bala | 865686868 |   natham
   7 |  balaji | 787878788 | dindigul
  36 | ruthran | 938373433 |   palani

(4 rows)
cqlsh:sample_demo>
```

Where the inserted values in the table has displayed.

7. To update table use the following command in the CASSANDRA shell.

```
cqlsh:sample_demo> UPDATE stu SET place='chennai' WHERE reg=7;
cqlsh:sample_demo> SELECT * FROM stu;

 reg | name    | ph        | place
-----+---------+-----------+----------
  19 |    haji | 868686868 | dindigul
  20 |    bala | 865686868 |   natham
   7 |  balaji | 787878788 |  chennai
  36 | ruthran | 938373433 |   palani

(4 rows)
```

Where the reg=007 has updated the place from dindigul to Chennai were the updated table is also displayed.

8. To delete a specified value from the table use the following command in the CASSANDRA shell.

```
cqlsh:sample_demo> DELETE ph FROM stu WHERE reg=20;
cqlsh:sample_demo> SELECT * FROM stu;

 reg | name    | ph        | place
-----+---------+-----------+----------
  19 |    haji | 868686868 | dindigul
  20 |    bala |      null |   natham
   7 |  balaji | 787878788 |  chennai
  36 | ruthran | 938373433 |   palani

(4 rows)
```

Where the ph value of reg=20 has been removed.

RESULT:

Thus the above program is executed successfully.

3. ORIENT DB

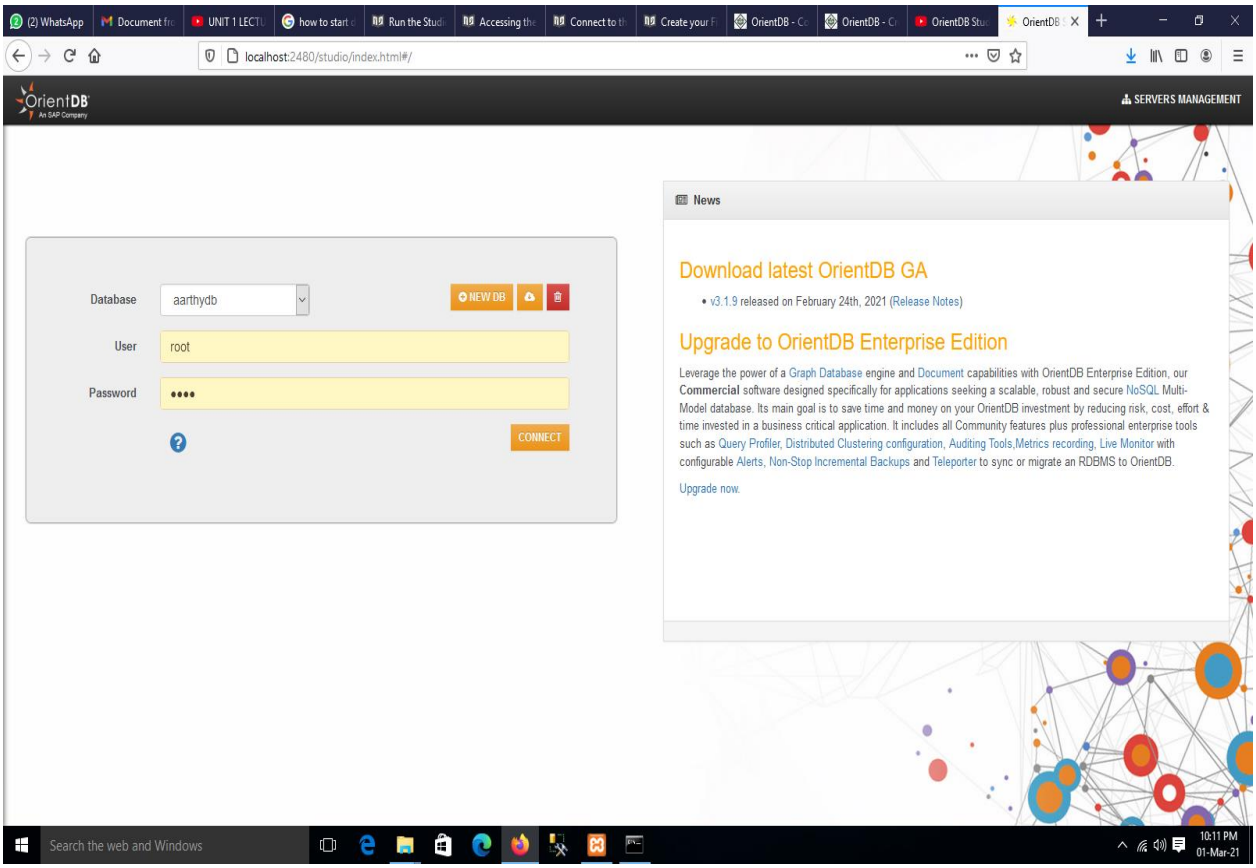
Aim:

To create a `table in Orient DB Graph Database and insert records and to make relationship between vectors.

ALGORITHM:

1. **Creating a Orient DB graph Database:**

   We can create a data base by entering a valid database and entering user name and password.

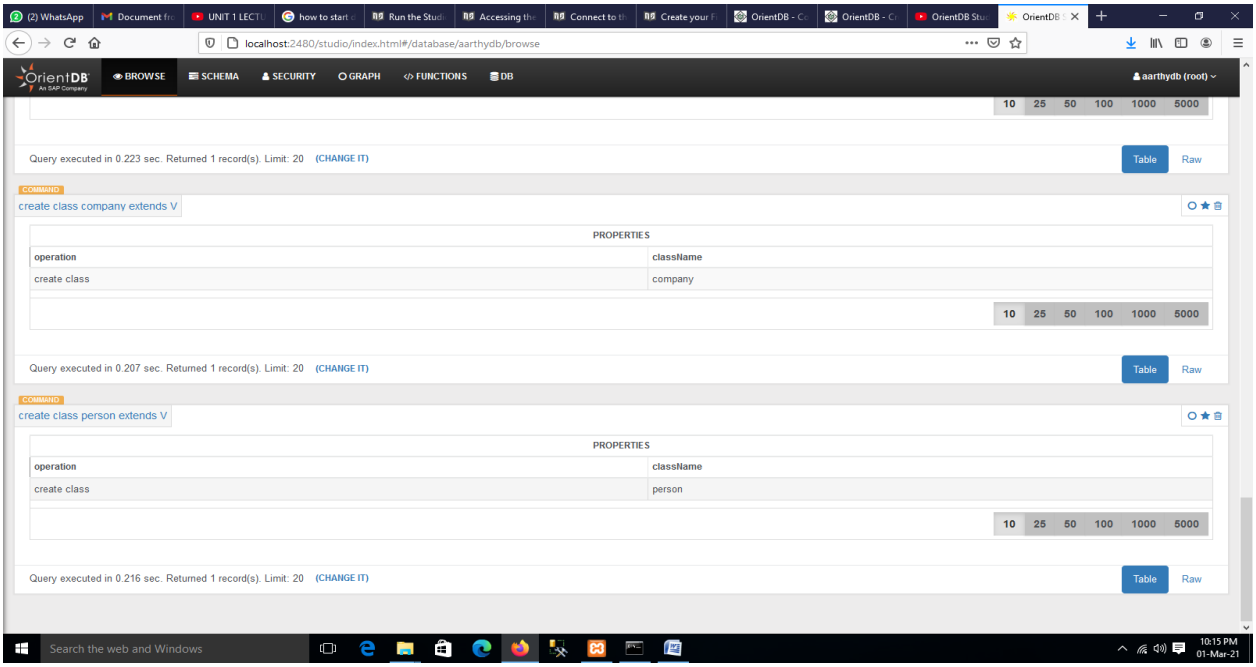


2. **Creating vertex :**

   We can create a vertex in OrientDB by using the query

   Query: Create class company extends V

   Create class person extends V



The two vertex named person and company has been created successfully.

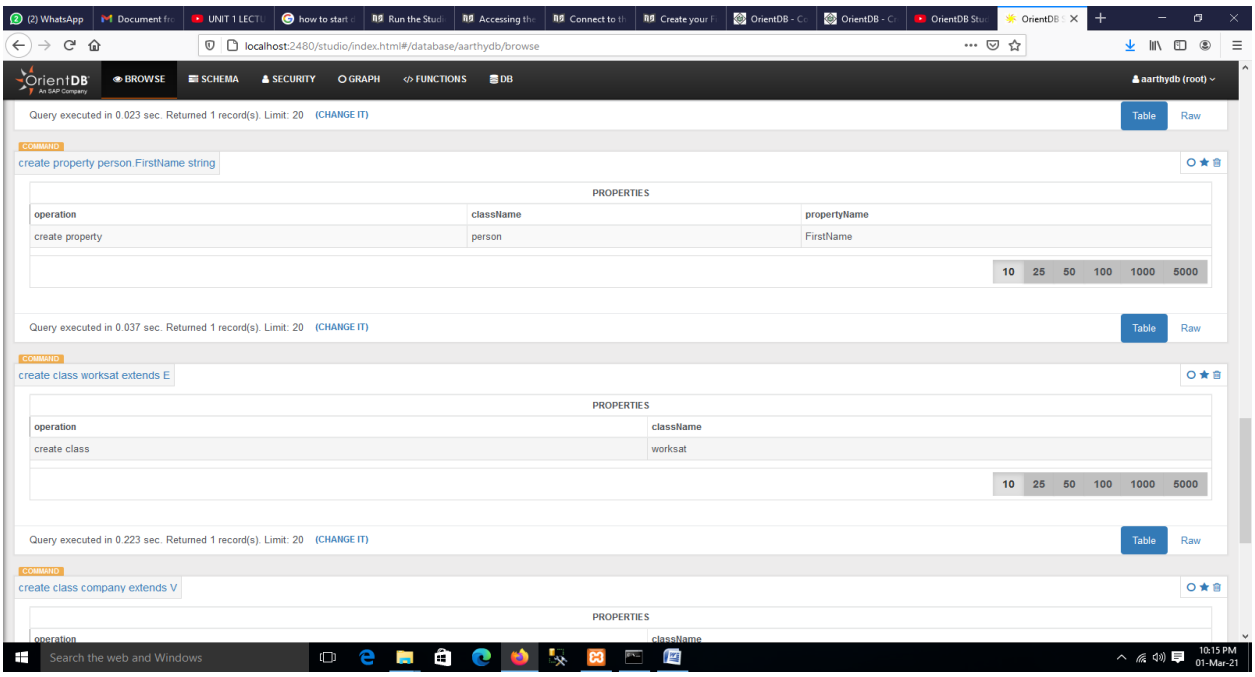
3. Creating property and edge:

The property in the OrientDB graph database is created by the following query

Query : Create property person FirseName.string

The edge in the OrientDB graph database is created by the following query

Query : Create class worksat extends E

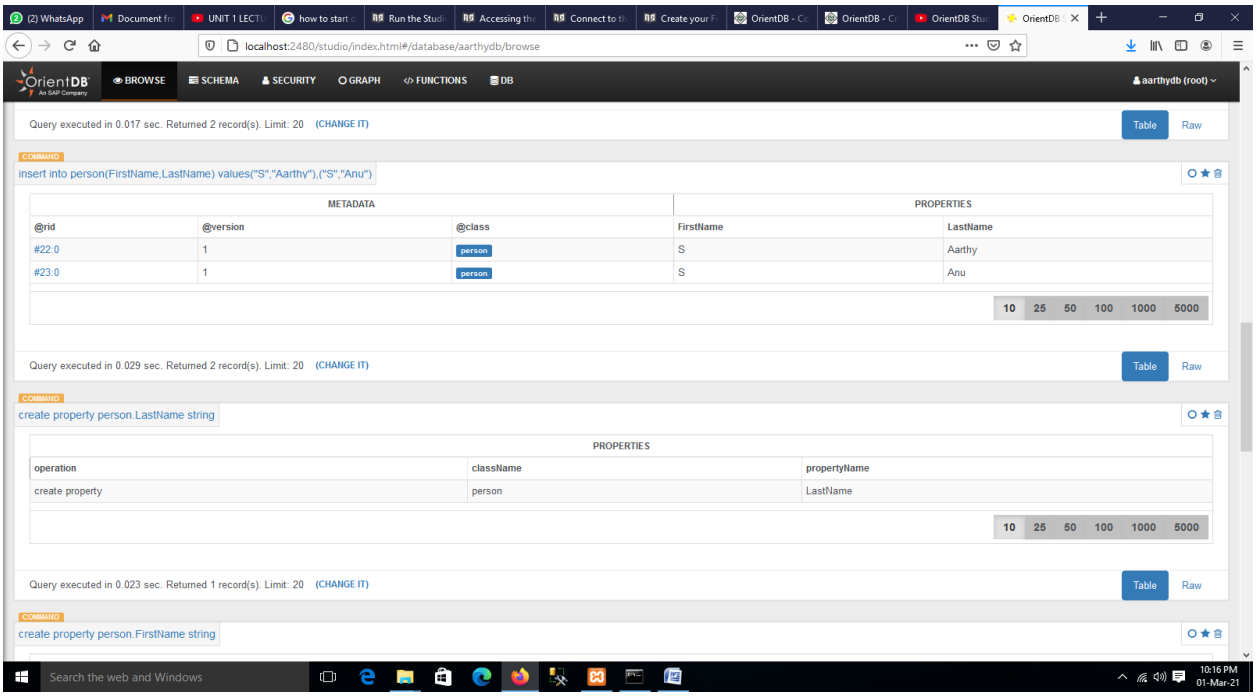


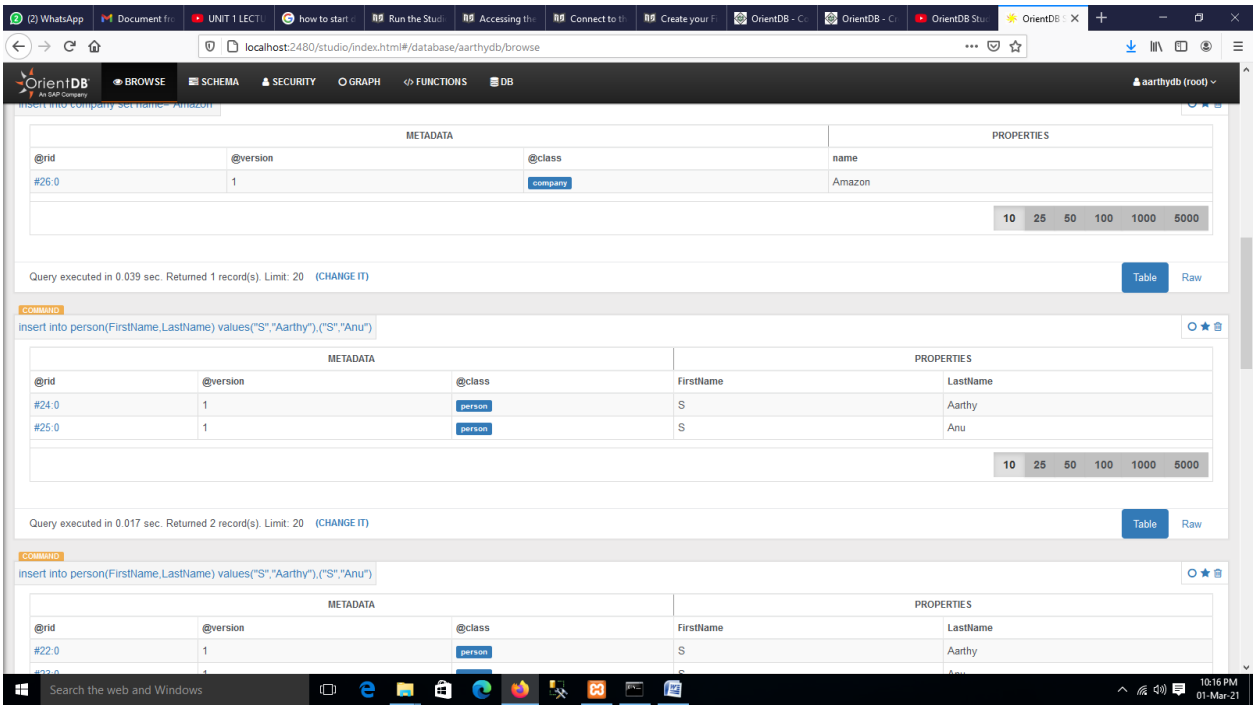
We have successfully created one property and 2 edges.

4. Inserting records:

The following query in the OrientDB graph database is used to enter the data or records

Query : Insert into person (FirstName,LastName) values (“S”,”Aarthy”),(“S”,”Anu”)





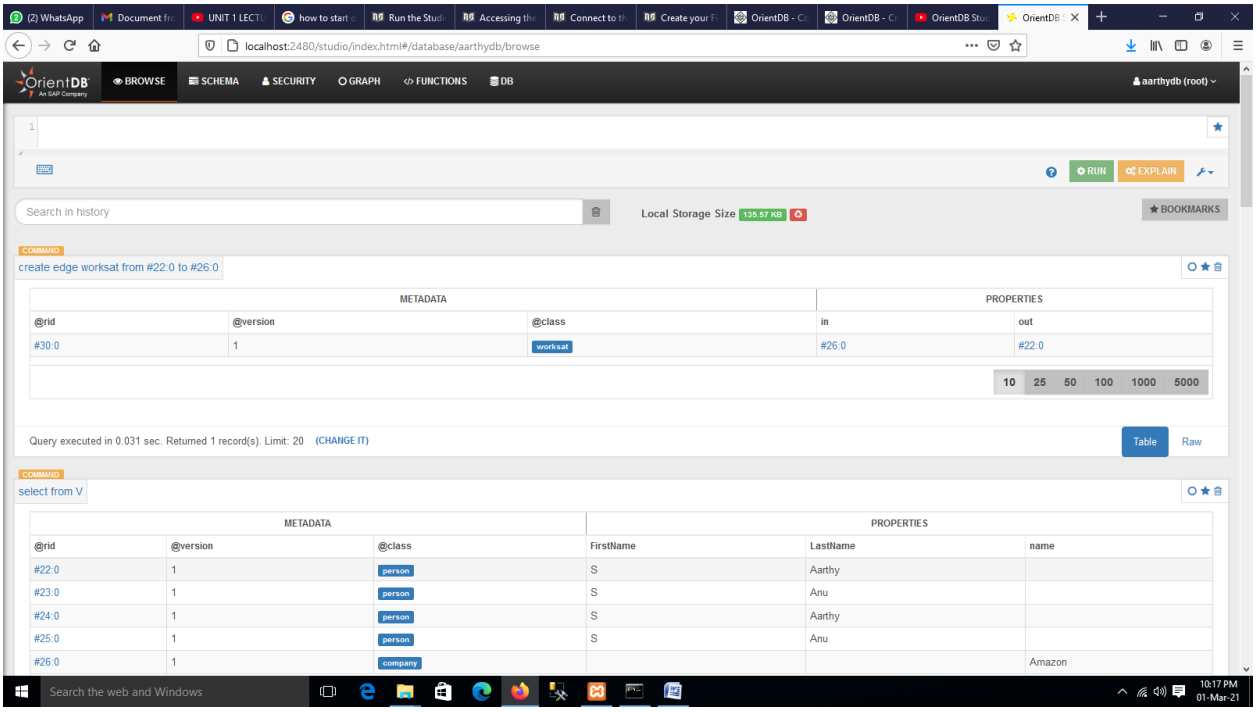
5. Creating Relationship:

The query to create a relationship in the OrientDB graph database by using the row id is as follows

Query : Create edge worksat from #22:0 to #26:0

To view the graph use the following query

Query : Select from V



Result:

Thus the OrientDB graph database has been created successfully and the OrientDB features has been executed successfully.

4. MySQL Database creation, Table creation and Query

AIM:

To create a database and a table in MySQL and to perform various queries in the created table.

ALGORITHM:

1. CREATING A DATABASE:

- To create a database in MySQL, click on the Database tab.
- In the create database, enter the appropriate name (**adbt**) for the database in the input field.
- We will get a message that **Database adbt has been created.**

2. CREATING A TABLE:

- In the created Database click on the 'Structure tab'.
- At the end of the tables list, we can see a Create Table option.
- Fill the input fields titled 'Name' as **library** and 'Number of Columns' as **5** and hit the 'Go' button.

BY USING QUERY:

We can also create a MySQL table using the following query

```
CREATE TABLE `adbt`.`library` ( `Book_ID` INT(10) NOT NULL , `Book_Name` VARCHAR(20) NOT NULL , `Author` VARCHAR(20) NOT NULL , `Edition` VARCHAR(10) NOT NULL , `Price` VARCHAR(10) NOT NULL ) ENGINE = InnoDB;
```

The table has been created successfully

3. INSERTING RECORDS IN THE TABLE:

We can insert records in the table **library** by using the query

```
INSERT INTO `library` (`Book_ID`, `Book_Name`, `Author`, `Edition`, `Price`) VALUES
('20048', 'PL/SQL', 'Deshpande', 'Second', '250'),
('36517', 'OSConcepts', 'Silberschertz', 'Sixth','650'),
```

('11023', 'RDBMS', 'RaghuRamakrishnan', 'Third', '700'),('32115', 'C', 'Balagurusamy', 'Fourth', '450') , ('1023', 'MobileCommunication', 'Schiller', 'Second', '375')

OUTPUT:

Book_ID	Book_Name	Author	Edition	Price
20048	PL/SQL	Deshpande	Second	250
36517	OS Concepts	Silberschertz	Sixth	650
11023	RDBMS	Raghu Ramakrishnan	Third	700
32115	C	Balagurusamy	Fourth	450
1023	Mobile Communication	Schiller	Second	375

4. UPDATING A RECORD IN THE TABLE:

We can update a particular record in the table **library** by using the update query

```
UPDATE `library` SET `Book_ID`='4000' WHERE Book_Name='C'
```

OUTPUT:

Book_ID	Book_Name	Author	Edition	Price
36517	OS Concepts	Silberschertz	Sixth	650
11023	RDBMS	Raghu Ramakrishnan	Third	700
4000	C	Balagurusamy	Fourth	450
1023	Mobile Communication	Schiller	Second	375

5. DELETE A RECORD FROM THE TABLE:

We can delete a record from the table **library** by using the **DELETE QUERY**

```
DELETE FROM `library` WHERE Book_ID='20048'
```

OUTPUT:

Book_ID	Book_Name	Author	Edition	Price
36517	OS Concepts	Silberschertz	Sixth	650
11023	RDBMS	Raghu Ramakrishnan	Third	700
4000	C	Balagurusamy	Fourth	450
1023	Mobile Communication	Schiller	Second	375

6. SORTING A PARTCULAR FIELD IN THE TABLE:

We can insert records in the table **library** by using the query

```
SELECT Book_Name FROM `library` ORDER BY Book_Name ASC
```

OUTPUT:

Book_Name ▲ 1
C
Mobile Communication
OS Concepts
RDBMS

RESULT:

Thus the creation of database and table in MySQL and various queries was preformed successfully.

5. MySQL Replication – Distributed Database

AIM:

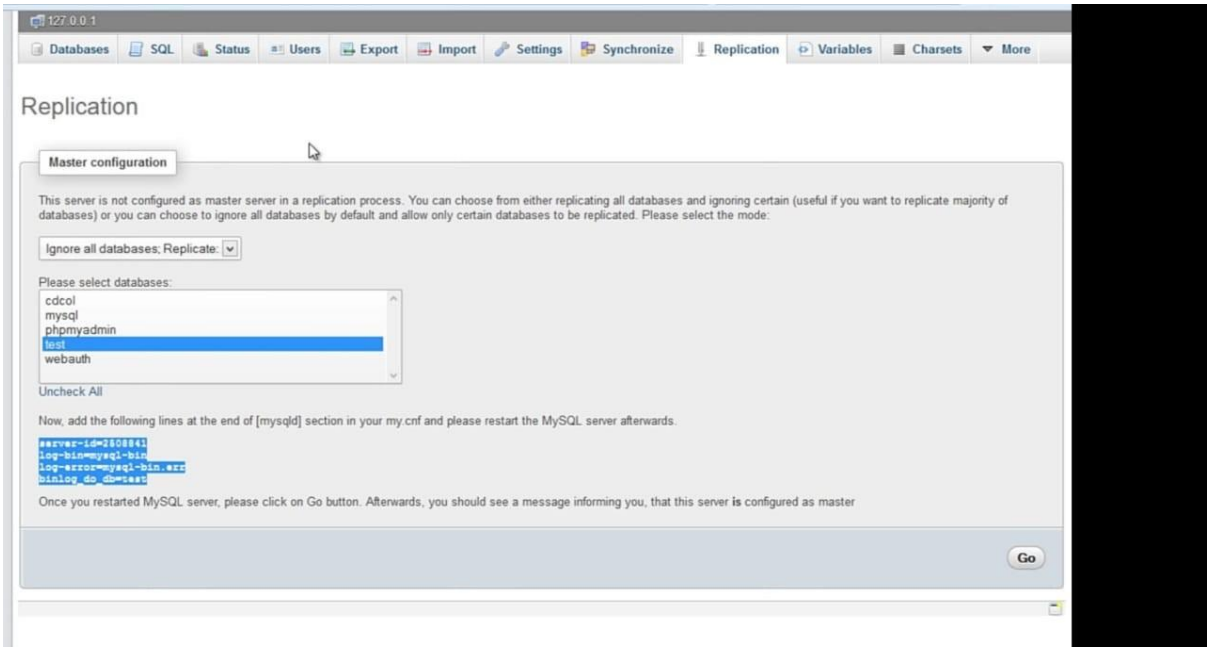
To perform MySQL replication by using php my admin.

ALGORITHM:

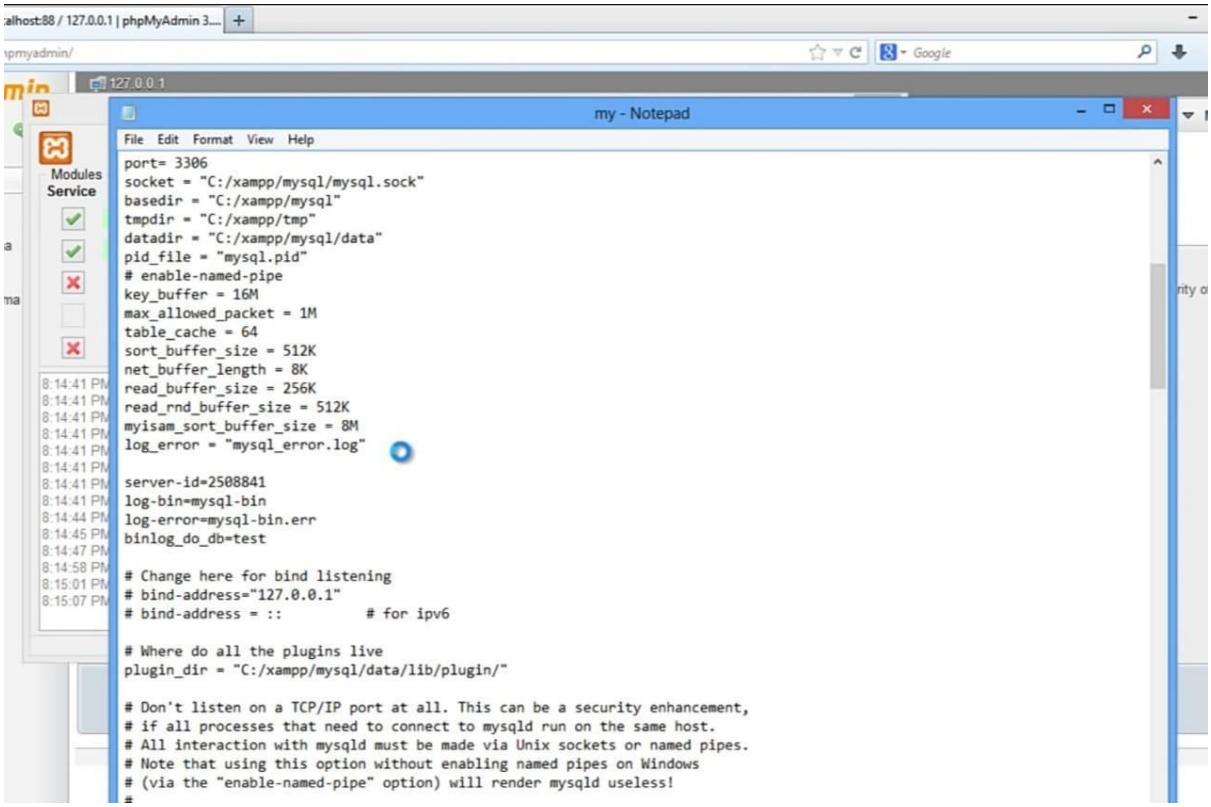
STEP1: Login to “phpmyadmin” panel.

STEP 2: Click on the replication tab. There shows configure under master replication section.

STEP 3: Master configuration will expand. click “ignore all databases: Replicate”. List of databases will display. Select “test” here. Copy the list of code over here. These have to be added in the MySQL “my.ini” file

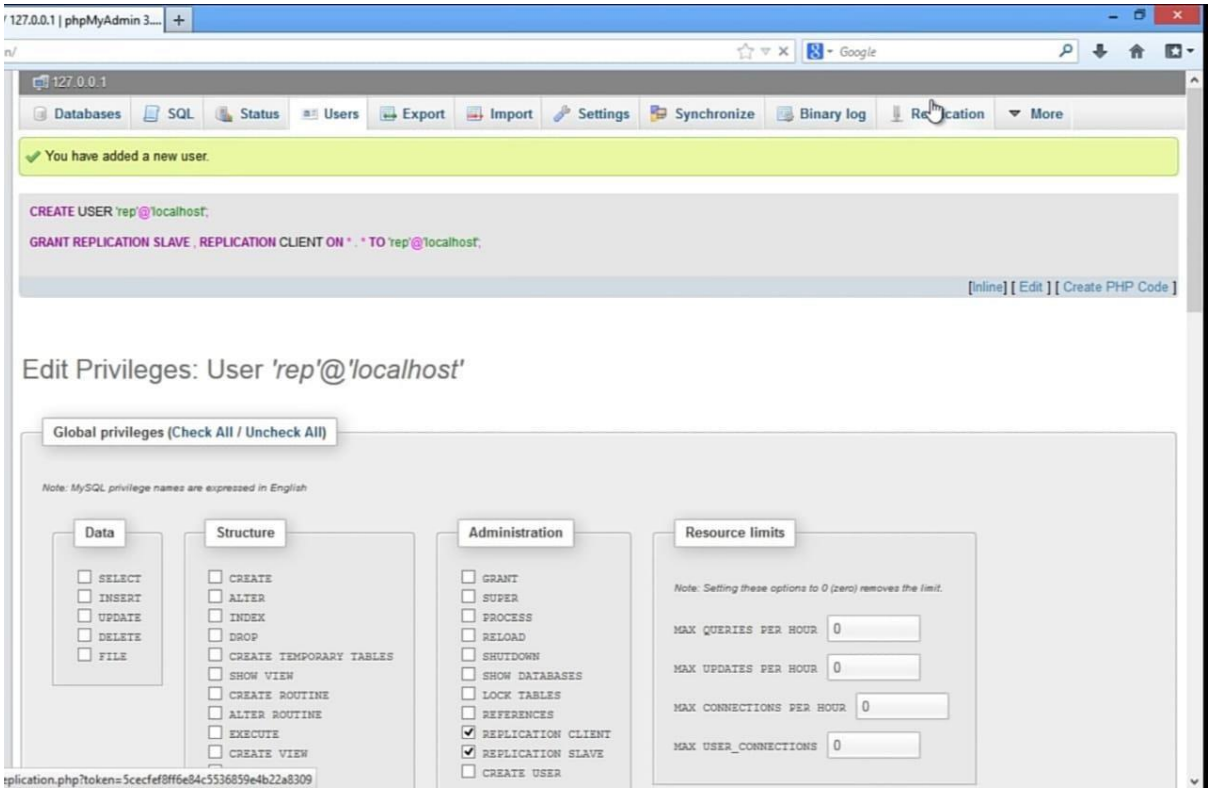


STEP 4: Open the XAMPP server to fetch the **my.ini** file by click the config button. After the files open up, paste the code under **log_error =”my sql_error.log”** and change the max_allowed_packet=16M and put # for server id=1. Then save it.

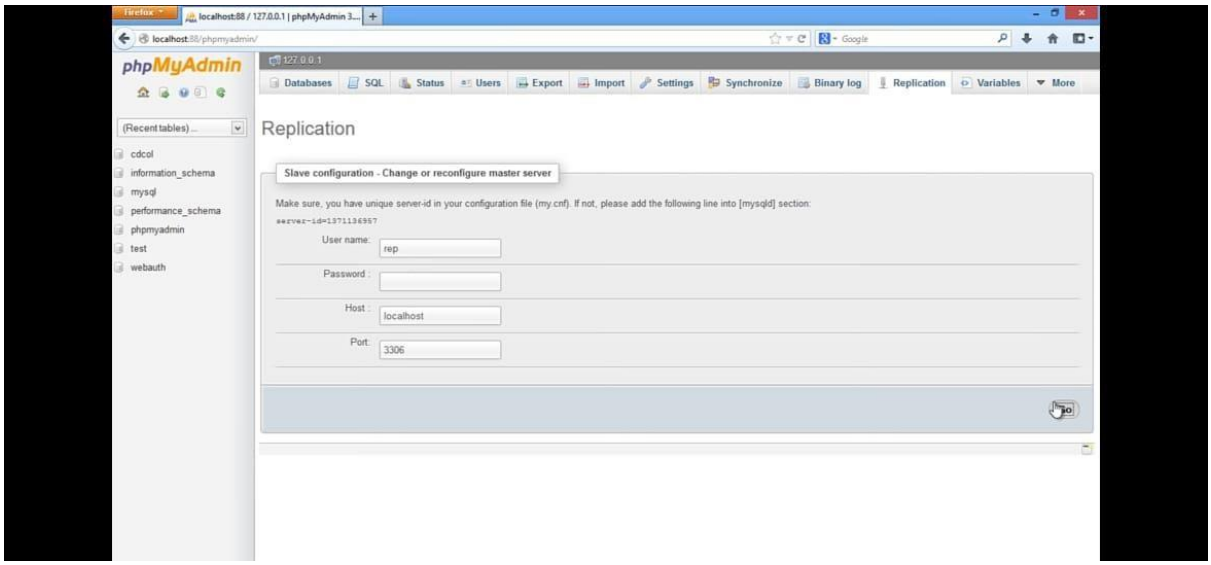


STEP 5: Start the MySQL and apache. Refresh the page and move on to the replication page again. The master replication has been setup successfully. Click on "show master status" to see the status.

STEP 6: Now click the "add slave replication user" to add a replication to the user. Then specify the user name in the host and click on the go button.



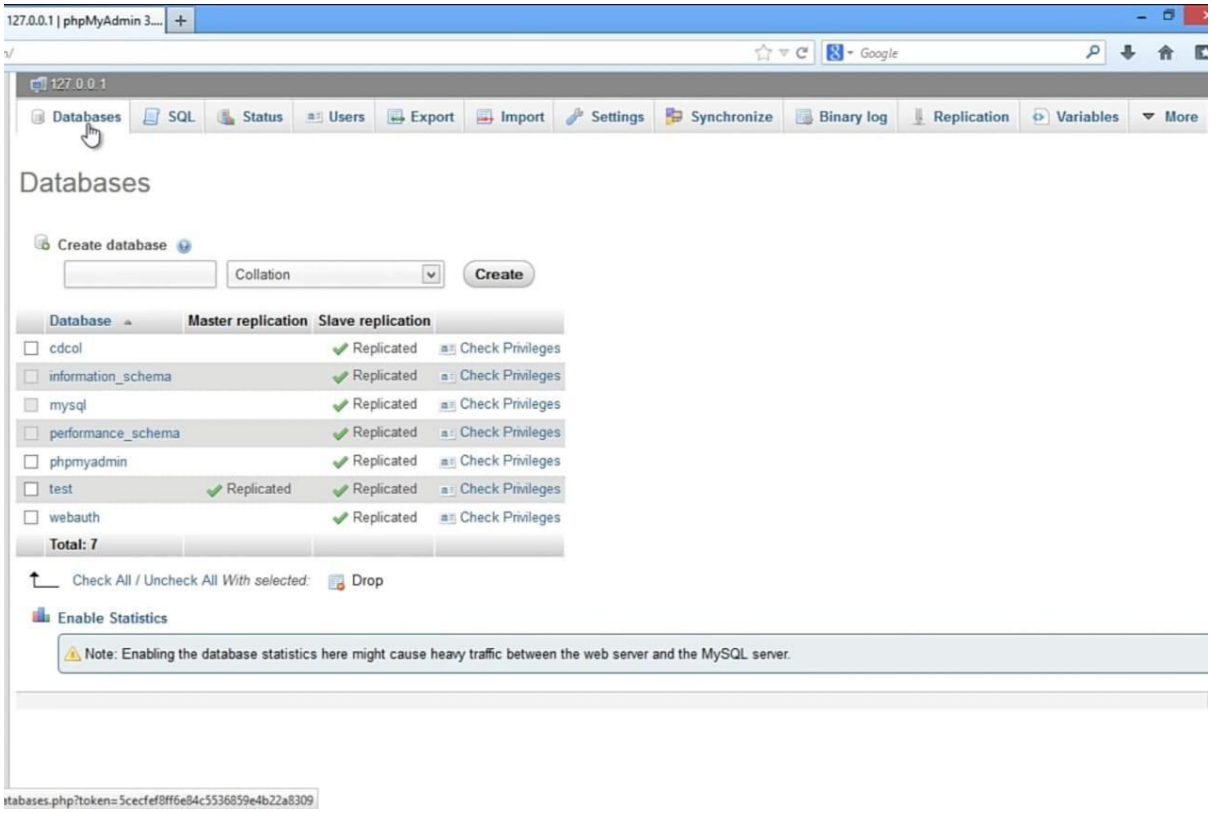
STEP 7: Open the replication tab again and click on the configure under the slave replication section. Another page will be open up , copy the server id to paste above #server_id=1 in “my.ini” file and save it. And restart the MySQL services. Then specify the user details given below and click on the go button.



STEP 8: A message will appear “A master changed successfully to localhost”. Now if we open replication table once more, it shows two warning messages in slave replication.

STEP 9: To resolve these errors, click start SQL thread only and start IO thread only under control slave one after another.

STEP 10: Now click on the databases tab, here we can see the test databases has been replicated successfully.



RESULT:

Thus the replication process has been successfully replicated.

6. Spatial data storage and retrieval in MySQL

AIM:

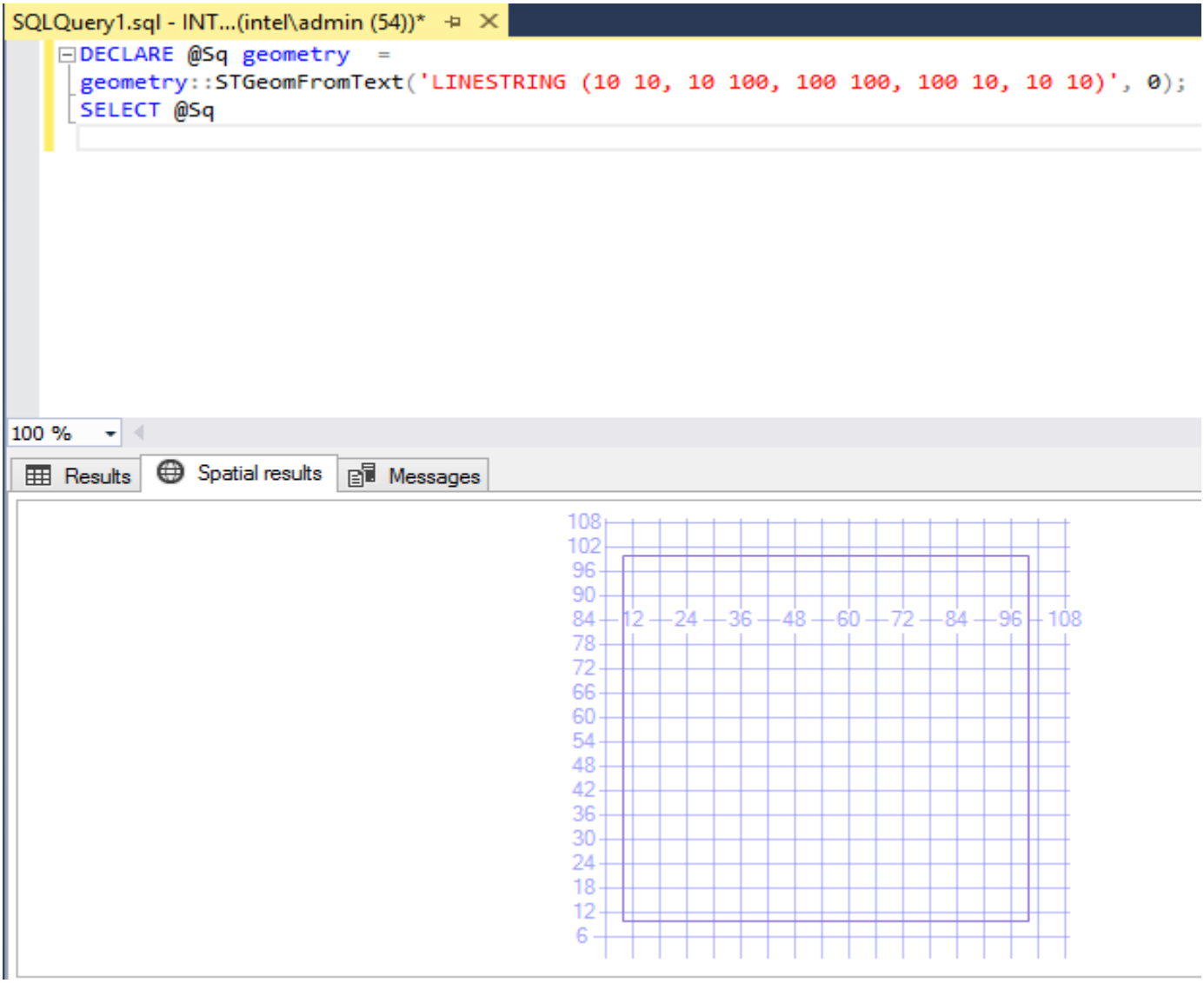
To Create and Store Spatial data and Retrieve it using MySQL.

ALGORITHM:

1. To store values for square and retrieve it as a image. A square is a regular quadrilateral with 4 lines, in which it has four equal angles with four sides.

DECLARE @Sq geometry = geometry::STGeomFromText('LINESTRING (10 10, 10 100, 100 10

0, 100 10, 10 10)', 0); SELECT @Sq

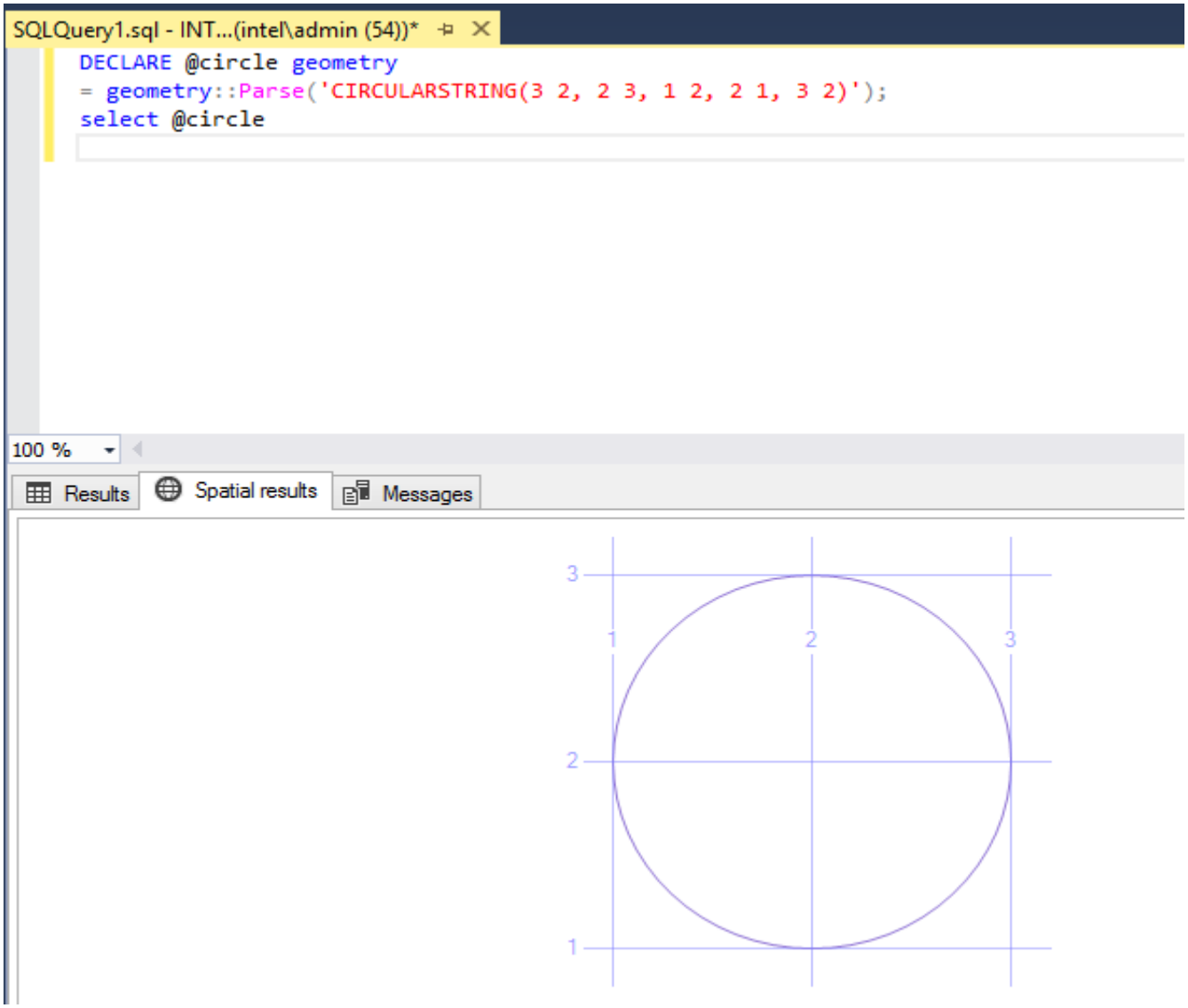


To store value for a circle with geographical coordinates and retrieve it as image.

A circle is a line enclosed, end to end, in which distance from any given point to another is constant

DECLARE @circle geometry = geometry::Parse('CIRCULARSTRING(3 2, 2 3, 1 2, 2 1, 3 2)');

select @circle

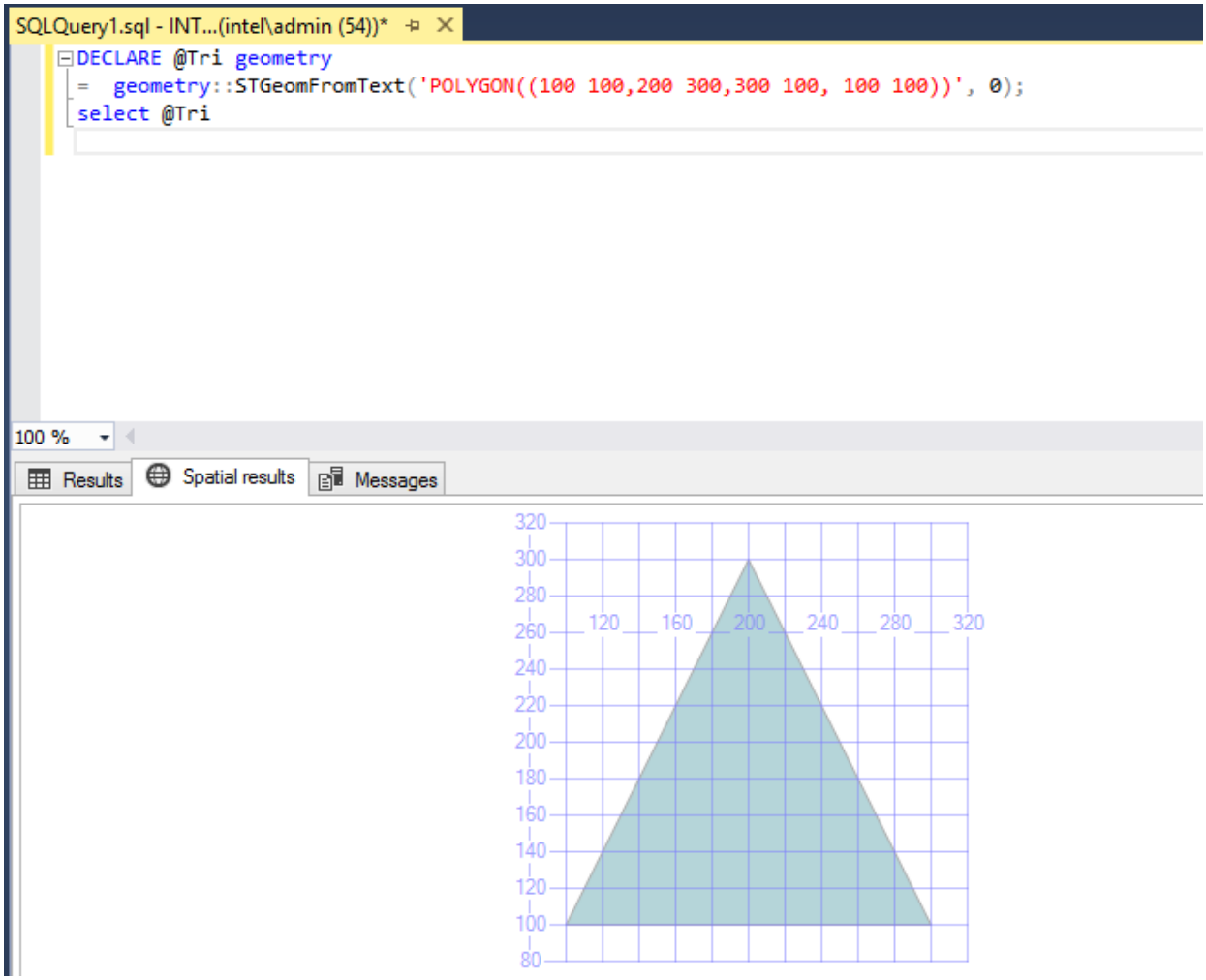


2. To store values for 3 vertices of Triangle and retrieve it as an image. A triangle is a polygon with three vertices and three edges.

DECLARE @Tri geometry = geometry::STGeomFromText('POLYGON((100 100,200 300,300 1

00, 100 100))', 0);

select @Tri



3. To store values for layer and retrieve it as an combined image. Here we would like to combine two objects and add some interesting effects. The below query will evaluate a couple of objects, a square and triangle, however, both objects overlap and give a unique image, although they are independent in shape

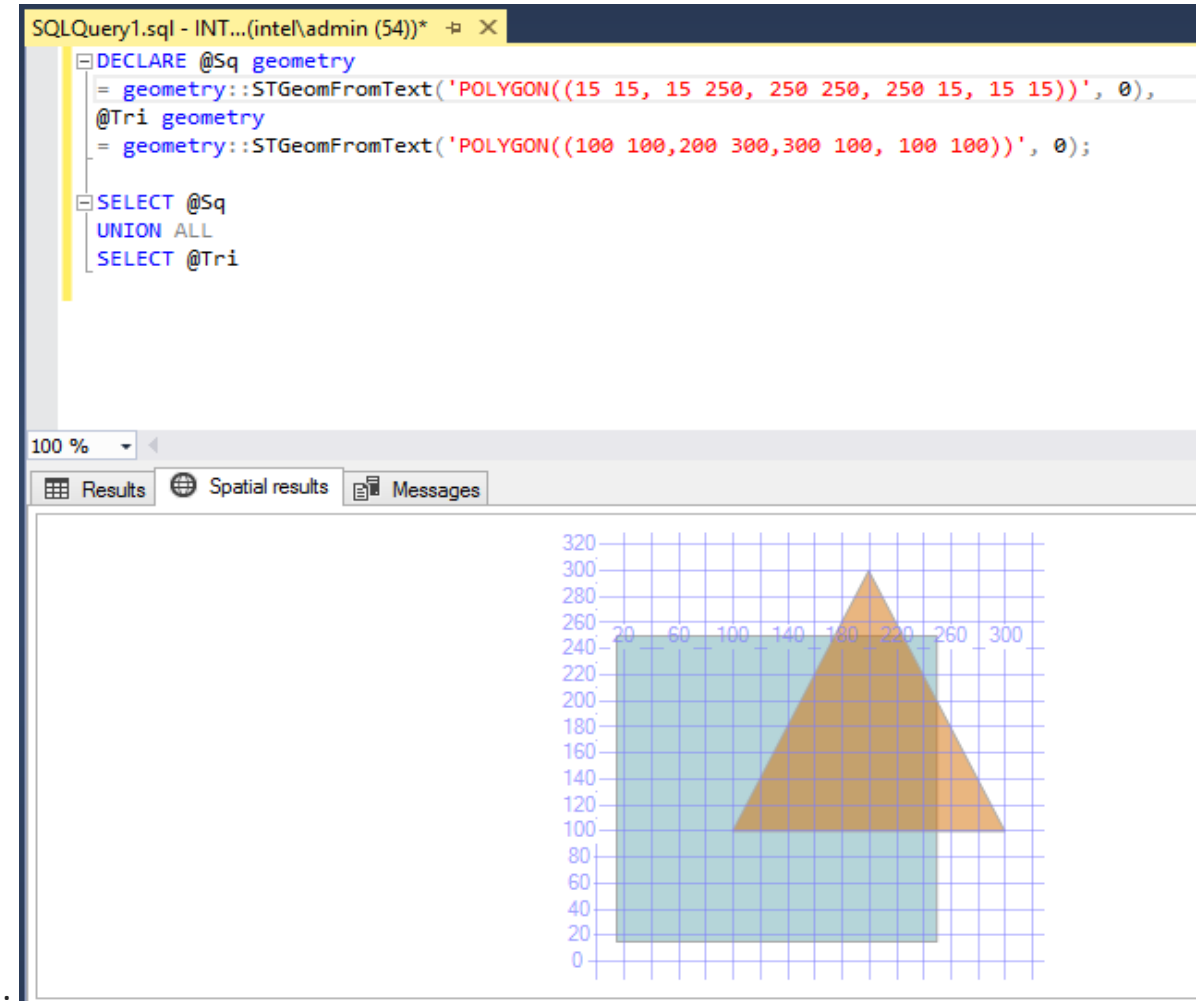
DECLARE @Sq geometry = geometry::STGeomFromText('POLYGON((15 15, 15 250, 250 250,

250 15, 15 15))', 0),

@Tri geometry = geometry::STGeomFromText('POLYGON((100 100,200 300,300 100, 100 1

00))', 0);

SELECT @Sq UNION ALL SELECT @Tri



4. To store values for layer intersection and retrieve as an image. An intersect returns the portion of the object that is in common between two objects

If the portion of the object exists in one query and not in other, it will be removed from the results. So the part that overlaps from object1 to object 2 is returned by the method STIntersection

DECLARE @Sq geometry = geometry::STGeomFromText('POLYGON((15 15, 15 250, 250 250,

250 15, 15 15))', 0),

@Tri geometry = geometry::STGeomFromText('POLYGON((100 100,200 300,300 100, 100 1

00))', 0);

SELECT @Sq.STIntersection(@Tri)

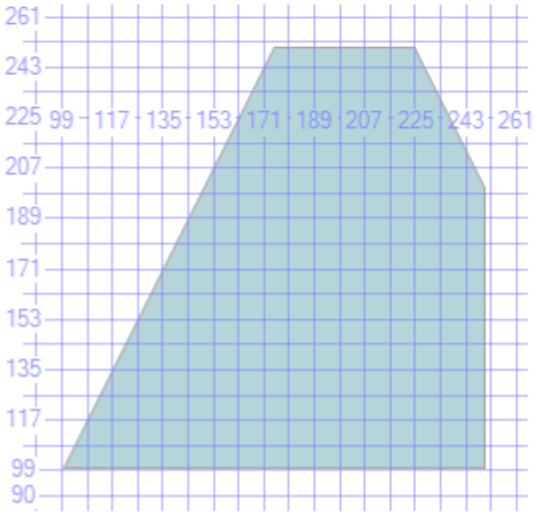
SQLQuery1.sql - INT...(intel\admin (54))*

```
DECLARE @Sq geometry
= geometry::STGeomFromText('POLYGON((15 15, 15 250, 250 250, 250 15, 15 15))', 0),
@Tri geometry
= geometry::STGeomFromText('POLYGON((100 100,200 300,300 100, 100 100))', 0);

SELECT @Sq.STIntersection(@Tri)
```

100 %

Results | Spatial results | Messages



RESULT:

Thus the program to store spatial data and retrieve in My SQL was executed successfully.

7. Temporal data storage and retrieval in MySQL

AIM:

To create a database and a table in MySQL to store temporal data and to retrieve data from the MySQL database.

ALGORITHM:

1. CREATING A DATABASE:

- To create a database in MySQL, click on the Database tab.
- In the create database, enter the appropriate name (mysql) for the database in the Input field.
- We will get a message that Database mysql has been created.

2. CREATING A TABLE:

- In the created Database click on the 'Structure tab'.
- At the end of the tables list, we can see a Create Table option.
- Fill the input fields titled 'Name' as emprecords and 'Number of Columns' as 6 and hit the 'Go' button.

BY USING QUERY:

We can also create a MySQL table using the following query

CREATE TABLE `emprecords` ('ID` INT(3) NOT NULL , `emp_name` VARCHAR(30) NOT NULL , 'Batch' INT(4) NOT NULL `Joining_Date` DATE, `Ending_Date` DATE) ENGINE = InnoDB;

The table has been created successfully

3. INSERTING RECORDS IN THE TABLE:

We can insert records in the table library by using the query

INSERT INTO `emprecords`(`Id`, `emp_name`, `Batch`, `Joining_Date`, `Ending_Date`) VALUES (['1','Ashwin','2020','2020-01-01','2022-12-30'],['2','Praveen','2020','2020-01-06','2022-12-23'],['3','Mani','2018','2018-02-05','2020-02-28'],['4','Aravindh','2019','2019-07-16','2021-07-30'],['5','Vijai','2018','2018-01-01','2020-12-25'],['6','Sherif','2018','2018-12-25','2020-11-30'])

OUTPUT:

Id	emp_name	Batch	Joining_Date	Ending_Date
1	Ashwin	2020	2020-01-01	2022-12-30
2	Praveen	2020	2020-01-06	2022-12-23
3	Mani	2018	2018-02-05	2020-02-28
4	Aravindh	2019	2019-07-16	2021-07-30
5	Vijai	2018	2018-01-01	2020-12-25
6	Sherif	2018	2018-12-25	2020-11-30

4. VIEW A PARTICULAR RECORD IN THE TABLE:

We can view a record from the table emprecords by using the where query

```
SELECT Id,emp_name,Batch,Joining_Date FROM `emprecords` WHERE Id=1
```

OUTPUT:

Id	emp_name	Batch	Joining_Date
1	Ashwin	2020	2020-01-01

```
SELECT Id,Batch,Joining_Date FROM `emprecords` WHERE emp_name='Ashwin'
```

OUTPUT:

Id	Batch	Joining_Date
1	2020	2020-01-01

5. VIEW BETWEEN & NOT BETWEEN RECORDS IN THE TABLE :

We can view a record from the table emprecords by using the BETWEEN query and NOT BETWEEN query

```
SELECT * FROM `emprecords` WHERE (Joining_Date BETWEEN '2018-01-01' AND '2018-12-25')
```

OUTPUT:

Id	emp_name	Batch	Joining_Date	Ending_Date
3	Mani	2018	2018-02-05	2020-02-28
5	Vijai	2018	2018-01-01	2020-12-25
6	Sherif	2018	2018-12-25	2020-11-30

```
SELECT emp_name,Joining_Date,Ending_Date FROM `emprecords` WHERE Batch NOT BETWEEN 2018 and 2019
```

OUTPUT:

emp_name	Joining_Date	Ending_Date
Ashwin	2020-01-01	2022-12-30
Praveen	2020-01-06	2022-12-23

6. SORTING AND DETERIORATING A PARTICULAR FIELD IN THE TABLE

We can insert records in the table emprecords by using the query

```
SELECT emp_name FROM `emprecords` ORDER BY emp_name asc
```

OUTPUT:

emp_name ▲ 1
Aravindh
Ashwin
Mani
Praveen
Sherif
Vijai

```
SELECT emp_name FROM `emprecords` ORDER BY emp_name DESC
```

OUTPUT:

emp_name ▼ 1
Vijai
Sherif
Praveen
Mani
Ashwin
Aravindh

RESULT:

Thus the creation of database and table in MySQL and various queries was performed successfully.

8. Object storage and retrieval using MySQL

AIM:

To perform crud operation on JSON object in MySql.

ALGORITHM:

1. Open Xampp control panel . Then start the Apache and MySql server.

2. **Creating Database:**
   In the MySql server , Create the Database by the following command.

   Code: CREATE DATABASE emp;

3. **Creating Table:**
   Then , create the table and their column including json type by the following command.

   Code: CREATE TABLE emprec(id int auto_increment primary key, name

   varchar(255), courses json);

4. **Inserting Record:**
   Inserting Table values including JSON values: Now, we have to insert the records (including json) into created table. The below command is used to insert record.

   Code: INSERT INTO

   emprec(name,courses)VALUES('Ramakrishnan','

   {"Java":"CSC","Dot net":"sisi"}');

5. Using the above code, insert 2 or more records to table.

6. **Fetching Records in the Table:**
   The Below code is used to fetching all the records in the table.

   Code : SELECT * FROM emprec;

7. **Updating json record in the Table:**
   The below command is used to update the json object of first id.

   Code: UPDATE emprec SET

   courses=JSON_SET(courses,'$.Java','Mpn Institute') WHERE id=1;

name	courses
Ramakrishnan	{"java": "Mpn Institute", "dot net": "sisi"}

8. **Deleting Json record in the table:**
   The below command is used to update the json object of first id.

   Code: SELECT JSON_REMOVE('{"java": "Mpn

   Institute", "dot net": "sisi"}', '$.java') AS

   'courses'

courses
{"dot net": "sisi"}

RESULT:

Thus the given program has been successfully executed.

9. Inserting XML data to MySQL database

AIM:

To perform the process of inserting xml data to MySQL database.

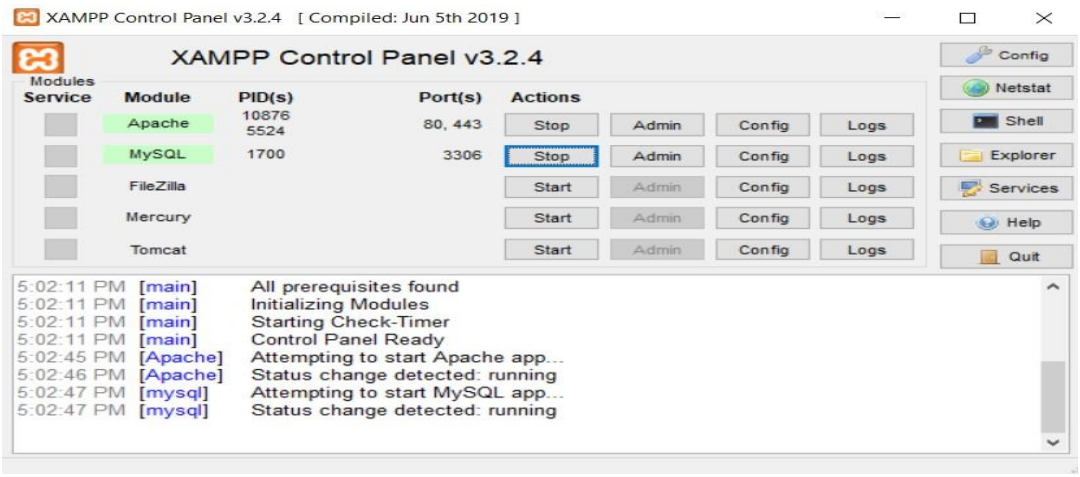
ALGORITHM:

1. Open notepad and enter the xml data in the format of parent child relationship.

Coding:

```
<college>
    <staff>
        <id>1</id>
        <name>Ganesh K</name>
        <dept>Mca</dept>
    </staff>
    <staff>
        <id>2</id>
        <name>Manikandan R</name>
        <dept>Mca</dept>
    </staff>
</college>
```

2. Save the above file in the .xml extension.
3. Now open the XAMPP Control Panel and start the Apache and MySql server.



4. In the MySql server , Create the database
   Coding: CREATE DATABASE college.
5. Then create the table and number of columns in the same structure of xml document.
   Coding: CREATE TABLE STAFF (id INT, name VARCHAR(20), dept VARCHAR(20));
6. Now, open the command prompt.
7. Then , enter the MySql path “C:\xampp\mysql\bin>”.
8. In that path enter the “**mysql.exe –u root –p** ” command. This command allows to access MySQL in command prompt .
9. Then enter the command to access the database “college”. **Coding :** use college

```
MariaDB [(none)]> use college
Database changed
MariaDB [college]>
```

10. The below command is used to load xml data to MySql database.

    Coding: LOAD XML INFILE
    "C:/Users/NithiyaPCcare/Desktop/college.xml"
    INTO TABLE staff ROWS IDENTIFIED BY '<college>';

```
MariaDB [college]> LOAD XML INFILE "C:/Users/NithiyaPCcare/Desktop/college.xml" INTO TABLE staff ROWS IDENTIFIED BY '<college>';
Query OK, 1 row affected (0.149 sec)
Records: 1  Deleted: 0  Skipped: 0  Warnings: 0

MariaDB [college]>
```

11. Now the xml data is loaded to MySql database. The below command is used to check the inserted record in loaded to database or not.

    Coding: select * from staff;

```
MariaDB [college]> select * from staff;
+----+--------------+------+
| id | name         | dept |
+----+--------------+------+
|  2 | Manikandan R | Mca  |
+----+--------------+------+
1 row in set (0.000 sec)
```

RESULT:

Thus the given program has been successfully executed.