**Term-Work -3: Date:26/03/2018**

**Problem definition:**

Write a Java program to simulate LOST UPDATE or INCONSISTENT READ Transaction issues of database using MULTITHREADING features of JAVA. Also write a java programto control the above concurrency issue. Output of the program to be displayed on the screen as well as to be written in a file of user choice.

**Description:**

INCONSISTENT READ occurs wherein one transaction is changing the tuple/record,

and a second transaction can read this tuple/record before the original change has

been committed or rolled back. This is known as a dirty read scenario because there

is always the possibility that the first transaction may rollback the change, resulting in

the second transaction having read an invalid value.

**Program-Case-1(unsynchronized):**

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.sql.\*;

import java.sql.SQLException;

import java.sql.Connection;

import java.util.logging.Level;

import java.util.logging.Logger;

class Termwork\_3\_1

{

public static void main (String args[])

{

try

{

Class.forName ("com.mysql.jdbc.Driver");

Connection con =DriverManager.getConnection ("jdbc:mysql://localhost:3306/termwork3","root", "");

//con.setAutoCommit (false);

//con.setTransactionIsolation (Connection.TRANSACTION\_READ\_COMMITTED);

Reader read = new Reader (con);

Update write = new Update (con);

try

{

write.start ();

read.start ();

}

catch (Exception e)

{

System.out.print (e);

}

}

catch (SQLException e)

{

e.printStackTrace ();

}

catch (ClassNotFoundException ex)

{

Logger.getLogger (Termwork\_3\_1.class.getName ()).log (Level.SEVERE, null,ex);

}

}

}

class Reader extends Thread

{

Connection con;

Reader (Connection con)

{

this.con = con;

}

public void run ()

{

try

{

Statement stmt = con.createStatement ();

ResultSet rs = stmt.executeQuery ("select \* from student");

//while (rs.next ())

//System.out.println (rs.getString (1) + " " + rs.getString (2) + " " + rs.getInt (3));

File file = new File("C:\\Users\\Ritesh\\Desktop\\ghanta\\xyz.txt");

FileWriter fstream = new FileWriter(file);

BufferedWriter out = new BufferedWriter(fstream);

while(rs.next()) {

String s1=rs.getString(1);

String s2=rs.getString(2);

int s3=rs.getInt(3);

System.out.println(s1+" "+s2+" "+s3);

out.write(s1+" ");

out.write(s2+" ");

out.write(Integer.toString(s3)+" ");

}

out.close();

}

catch (SQLException e)

{

e.printStackTrace ();

} catch (IOException ex) {

Logger.getLogger(Reader.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

class Update extends Thread

{

Connection con;

Update (Connection con)

{

this.con = con;

}

public void run ()

{

try

{

this.sleep (1000);

Statement stmt = con.createStatement ();

stmt.executeUpdate ("update student set usn='2sd15cs083' where sem=7");

//System.out.println("h");

} catch (SQLException e)

{

e.printStackTrace ();

} catch (InterruptedException ex)

{

Logger.getLogger (Update.class.getName ()).log (Level.SEVERE, null,ex);

}

}

}

**Program-Case-1(synchronized):**

import java.sql.\*;

import java.sql.SQLException;

import java.sql.Connection;

import java.util.logging.Level;

import java.util.logging.Logger;

import java.io.\*;

import java.util.\*;

class TWtest{

public static void main(String args[]){

try{

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/termwork3","root","");

Reader2 read=new Reader2(con);

Update2 write=new Update2(con);

try

{

write.start();

read.start();

}

catch(Exception e)

{

System.out.print(e);

}

}

catch (SQLException e) {

e.printStackTrace();

}

catch (ClassNotFoundException ex) {

Logger.getLogger(TWtest.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

class Reader2 extends Thread{

Connection con;

Reader2(Connection con)

{

this.con=con;

}

public void run(){

try

{

Statement stmt=con.createStatement();

ResultSet rs=stmt.executeQuery("select \* from student");

File file = new File("C:\\Users\\Ritesh\\Desktop\\ghanta\\xyz.txt");

FileWriter fstream = new FileWriter(file);

BufferedWriter out = new BufferedWriter(fstream);

while(rs.next()) {

String s1=rs.getString(1);

String s2=rs.getString(2);

int s3=rs.getInt(3);

System.out.println(s1+" "+s2+" "+s3);

out.write(s1);

out.write(s2);

out.write(Integer.toString(s3));

}

out.close();

}

catch (SQLException e) {

e.printStackTrace();

} catch (IOException ex) {

Logger.getLogger(Reader2.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

class Update2 extends Thread{

Connection con;

Update2(Connection con)

{

this.con=con;

}

public void run(){

synchronized(con){

try

{

this.sleep(1000);

Statement stmt=con.createStatement();

stmt.executeUpdate("update student set usn='2sd15005' where sem=7");

//String query="INSERT INTO student(name,usn,sem) VALUE ('ritesh','fg0',6);";

// stmt.executeUpdate(query);

}

catch (SQLException e) {

e.printStackTrace();

}

catch (InterruptedException ex) {

Logger.getLogger(Update.class.getName()).log(Level.SEVERE, null, ex);

}

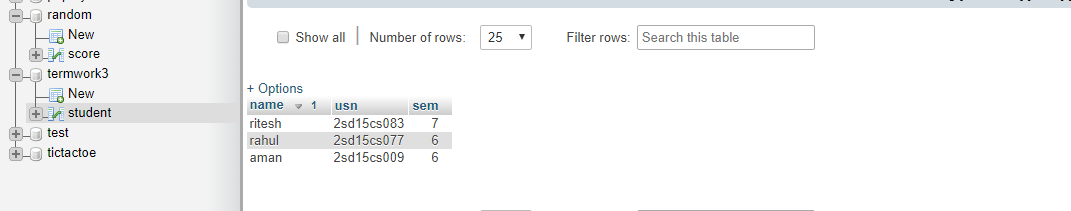
}

}

}

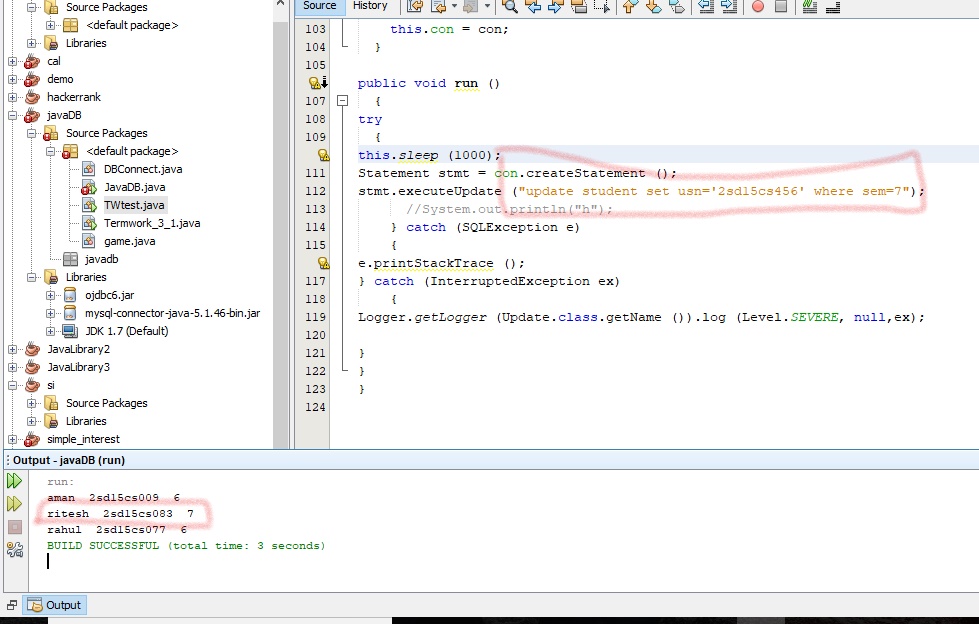
**Input/Output with screenshot:**

Current State of Database:



In first case after simultaneous Update and Read Statements.The previous value is read

,before update could take place:



In second case both the threads are synchronized and current updated value is read:

