RAMAIAH UNIVERSITY OF APPLIED SCIENCES

ASSIGNMENT

Course Code	**19CSC302A**
Course Name	**Database Systems**
Programme	**B. Tech**
Department	**Computer Science and Engineering**
Faculty	**Engineering and Technology**

Name of the Student	**Deepak R**
Reg. No	**18ETCS002041**
Semester/Year	**5th/2020**
Course Leader/s	**Ami Rai E.**

Declaration Sheet				
Student Name	**Deepak R**			
Reg. No	**18ETCS002041**			
Programme	**B. Tech**	**Semester/Year**	**5th/2020**	
Course Code	**19CSC302A**			
Course Title	**Computer Networks**			
Course Date		**to**		
Course Leader	**Ami Rai E.**			

Declaration

The assignment submitted herewith is a result of my own investigations and that I have conformed to the guidelines against plagiarism as laid out in the Student Handbook. All sections of the text and results, which have been obtained from other sources, are fully referenced. I understand that cheating and plagiarism constitute a breach of University regulations and will be dealt with accordingly.

Signature of the Student		**Date**	
Submission date stamp **(by Examination & Assessment Section)**			

Signature of the Course Leader and date	Signature of the Reviewer and date

Faculty of Engineering & Technology			
Ramaiah University of Applied Sciences			
Department	Computer Science and Engineering	Programme	B. Tech. Computer Science and Engineering
Semester/Batch	5th/2018		
Course Code	19CSC302A	Course Title	Database Systems
Course Leader(s)	A. Prabhakar, Gp Capt N Rath VSM, Ami Rai E.		

Assignment - 02

Register No.	18ETCS002041	Name of Student	Deepak R

Sections		Marking Scheme	Max Marks	First Examiner Marks	Second Examiner Marks
Part A	A1.1	Introduction to parallel database system	01		
	A1.2	Discussion on query processing in parallel database system	02		
	A1.3	Discussion on challenges	02		
		Part-A Max Marks	05		
Part B1	B1.1	Identification of candidate key(s)	02		
	B1.2	Highest normal form	02		
	B1.3	Decomposition of the tables	04		
		B1 Max Marks	08		
Part B2	B2.1	Design and implementation of GUI	03		
	B2.2	Connection of front end with the database	03		
	B2.3	Discussion on the results	03		
	B2.4	Concluding remarks (Summary, limitations, improvements	03		
		B2 Max Marks	12		
		Total Assignment Marks	25		

Course Marks Tabulation				
Component-1(B)Assignment	First Examiner	Remarks	Second Examiner	Remarks
A				
Marks (out of 10)				
Signature of First ExaminerSignature of Second Examiner				

Please note:

1. **Documental evidence for all the components/parts of the assessment such as the reports, photographs, laboratory exam / tool tests are required to be attached to the assignment report in a proper order.**
2. **The First Examiner is required to mark the comments in RED ink and the Second Examiner's comments should be in GREEN ink.**
3. **The marks for all the questions of the assignment have to be written only in the Component – CET B: Assignment table.**
4. **If the variation between the marks awarded by the first examiner and the second examiner lies within +/- 3 marks, then the marks allotted by the first examiner is considered to be final. If the variation is more than +/- 3 marks then both the examiners should resolve the issue in consultation with the Chairman BoE.**

Assignment - 02

Instructions to students:

1. **The assignment consists of 3questions: Part A –1 Question, Part B- 2Questions.**
2. **Maximum marks is25.**
3. **The assignment has to be neatly word processed as per the prescribed format.**
4. **Submission Date: 16/01/2020**
5. **Submission after the due date is not permitted.**
6. **IMPORTANT: It is essential that all the sources used in preparation of the assignment must be suitably referenced in the text.**
7. **Marks will be awarded only to the sections and subsections clearly indicated as per the problem statement/exercise/question**

Solution to Part A

Introduction to parallel database system

Parallel database management refers to the management of data in a multiprocessor computer and is done by a parallel database system, i.e., a full-fledge DBMS implemented on a multiprocessor computer. The basic principle employed by parallel DBMS is to partition the data across multiprocessor nodes, in order to increase performance through parallelism and availability through replication. This enables supporting very large databases with very high query or transaction loads.

Query processing in parallel database system

Parallel Query processing in databases systems means that the job of producing the results of the query gets divided between many processes which execute in parallel, thus leading to improvements in performance.

The underlying concept for achieving parallelism in data processing is partitioning the dataset and processing each partition in parallel such that results produced from each partition are (usually) independent of one another and can be simply merged and presented to the user.

Typically a SQL query gets broken down into a set of sequential operations. Some or all of such steps can be executed in parallel. A typical database system figures out which operations can be parallelized and how to parallelize them. Each operation can be parallelized in various ways and appropriate method is chosen depending on your system configuration and size of data etc.

Each database system has a different architecture and thus you will find as many ways of achieving parallel query execution as the number of architectures but data partitioning and redistribution is central is all such systems.
It is a way to get the other processors (or even the other computers) involved in the overall task.

It is a method of processing the data in a way that utilizes the computer capacities to the full by doing multiple works at the same time. Pretty much like having 4 loading people when packing the truck on the house move.

Discussion on challenges

Parallel loading of data from external sources is needed in order to handle large volumes of incoming data.

• Resilience to failure of some processors or disks.

– Probability of some disk or processor failing is higher in a parallel system.

– Operation (perhaps with degraded performance) should be possible in spite of failure.

– Redundancy achieved by storing extra copy of every data item at another processor.

On-line reorganization of data and schema changes must be supported.

– For example, index construction on terabyte databases can take hours or days even on a parallel system.

• Need to allow other processing (insertions/deletions/updates) to be performed on relation even as index is being constructed.

– Basic idea: index construction tracks changes and "catches up" on changes at the end.

• Also need support for on-line repartitioning and schema changes (executed concurrently with other processing).

Conclusion

Companies need to handle huge amount of data with high data transfer rate. The client server and centralized system is not much efficient to improve the efficiency the Companies Have to Depend on Parallel Database .With this Companies can improve Speed , reliable. and Efficiency although there is few disadvantages.

Solution to B1

Solution to B1.1 Identification of candidate key(s)

EMPLOYEE (ID, Name, Type, Salary, Designation, Address).Functional dependencies:-

ID -> Type, Address ----FD1

Type -> Salary. ----------FD2

Name ->Designation ---FD3

To find the candidate key closure of each Attribute should be find

CLOSURES :-+

ID -> {ID,Type,Address,salary}

Name -> {Name, Designation}

Type -> {Type,Salary}

Salary -> {Salary}

Designation -> {Designation}

Address -> {Address}

Closure of candidate key should be the relation (Candidate key)$^+$ ->Relation

closure of any single attribute does not gives full relation.

The combination of two or more attributes is a candidate key here

Closure of Id, Name gives the all the Attribute s in the relation

(Id,Name)$^+$ -> {ID, Name, Type, Salary, Designation, Address}.

{Id,Name} is a candidate key

No other candidate keys possible.

Solution to B1.2 Highest normal form

Checking for each NF :-

1NF(Simple Attributes):-The relation given is in 1NF by default .Because does not contain any composite and Multivalued Attributes.It only contain simple attributes. so it's in 1NF .

2NF(1NF + No partial dependency) :- It is in 1NF But it is not in 2NF.Beacause{Id, Name} are prime attributes means {Type,salary, destination,Address} are Non prime Attributes.

By the definition of 2NF every non-prime AttributeA in R should be fully functionally dependent on the primary key of Relation R.

In this relation {ID,NAME}is a primary key.

{Id ,Name} -> Address is a partial dependency because of functional dependency Id -> Address

Address is only functionally dependent on Id but not Name attribute.

{Id ,Name} -> Type is a partial dependency because of functional dependency Id -> Type

Type is only functionally dependent on Id but not on Name attribute.

{Id ,Name} -> Designation is a partial dependency because of functional dependency Name-> Address .Address is only functionally dependent on Name but not Id

By Considering all this this relation is not in 2NF .So by defaultit is also not in 3NF.

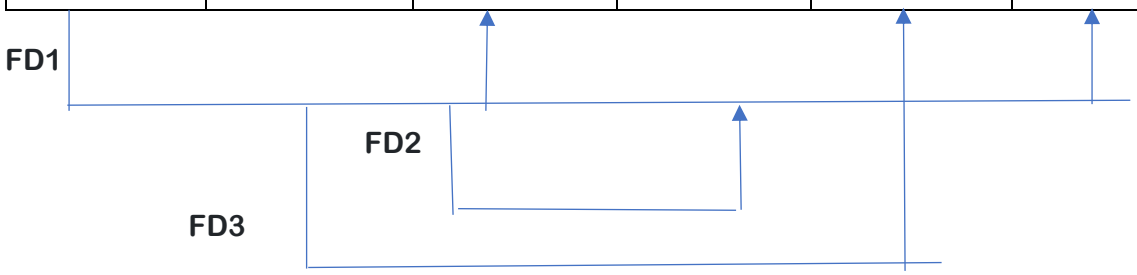
The Highest Normal Form of the relation is 1NF

Solution to B1.3 Decomposition of the tables

Highest Normal Form is 1NF So decomposing the relation into 2NF

EMPLOYEE

Id	Name	Type	Salary	Designation	Address



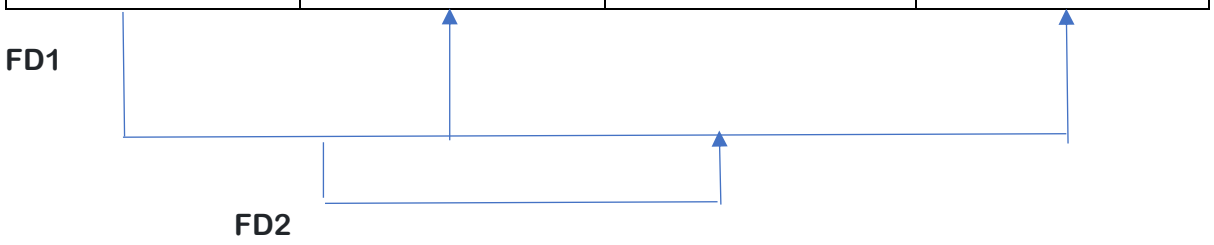
INTO 2NF

EMP_1

Id	Name

EMP_2

Id	Type	Salary	Address



EMP_3

Name	Designation

The relation is now decomposed

EMP_1(Id,Name). EMP_2(Id,Type,address,Salary). EMP_3(Name, Designation)

Now these relations are in 2NF.

EMP_2(Id,Type,address,Salary).

In this Relation Checking for 3NF

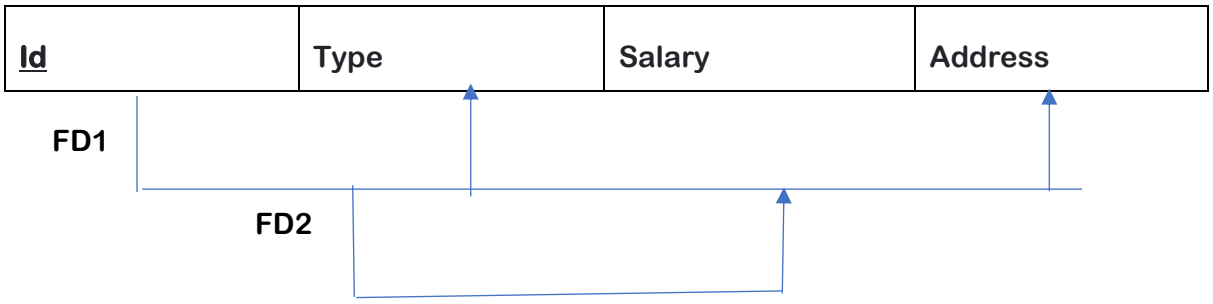
3NF Defnition :- No non-primary-key attribute is transitively dependent on the primary key,

ID -> Type -------from FD1

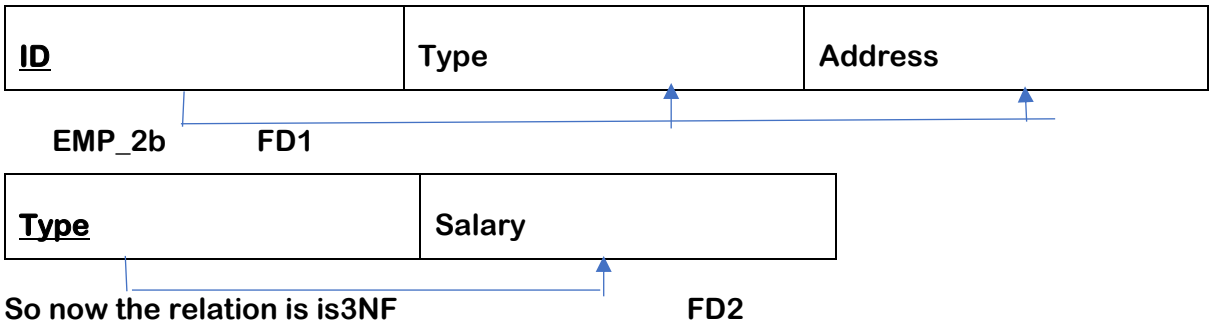
Type -> Salary----------FD2

In this Non prime attribute Salary transitively dependent on primary key ID So it violates the 3NF Property.So it is decomposed

EMP_2



EMP_2a.



The decomposed relations are

EMP_1(**Id,Name**)

EMP_2a(Id,Type,Address**)**

EMP_2b**(TYPE,**SALARY)

EMP_3(**Name,**Designation)

Now there is no Transition So The Highest Normal Form it can be decomposed is 3NF.

Solution to B2

Constructing Tables in Sql

```
mysql> Create database Online_Furnitures_shopping;
Query OK, 1 row affected (0.00 sec)
```

Fig 2.3.1 Created Database named Online_Furniture_System

```
mysql> Create table Customer(Customer_ID int primary key,Customer_IP int unique,Customer_name char(30) not null,Customer_email char(30),Customer_contact int,Customer_country char(30)  ,Customer_address char(30),
Customer_image char(30),Customer_city char(30), Customer_pass char(30));
Query OK, 0 rows affected (0.07 sec)

mysql> desc Customer;
+-----------------+---------+-----+----+---------+-------+
| Field           | Type    | Null | Key | Default | Extra |
+-----------------+---------+-----+----+---------+-------+
| Customer_ID     | int(11) | NO   | PRI | NULL    |       |
| Customer_IP     | int(11) | YES  | UNI | NULL    |       |
| Customer_name   | char(30) | NO   |     | NULL    |       |
| Customer_email  | char(30) | YES  |     | NULL    |       |
| Customer_contact | int(11) | YES  |     | NULL    |       |
| Customer_country | char(30) | YES  |     | NULL    |       |
| Customer_address | char(30) | YES  |     | NULL    |       |
| Customer_image  | char(30) | YES  |     | NULL    |       |
| Customer_city   | char(30) | YES  |     | NULL    |       |
| Customer_pass   | char(30) | YES  |     | NULL    |       |
+-----------------+---------+-----+----+---------+-------+
10 rows in set (0.09 sec)
```

Fig 2.3.2 Created table named Customer with its respective attributes

```
mysql> create table Admin(User_id int primary key,User_name char(30),User_email char(30));
Query OK, 0 rows affected (0.04 sec)

mysql> desc admin;
+------------+----------+------+-----+---------+-------+
| Field      | Type     | Null | Key | Default | Extra |
+------------+----------+------+-----+---------+-------+
| User_id    | int(11)  | NO   | PRI | NULL    |       |
| User_name  | char(30) | YES  |     | NULL    |       |
| User_email | char(30) | YES  |     | NULL    |       |
+------------+----------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

Fig 2.3.3 Created table named Admin with its respective attributes

```
mysql> create table Categories(Cat_id int primary key,Cat_tittle char(30));
Query OK, 0 rows affected (0.04 sec)

mysql> desc Categories;
+------------+----------+------+-----+---------+-------+
| Field      | Type     | Null | Key | Default | Extra |
+------------+----------+------+-----+---------+-------+
| Cat_id     | int(11)  | NO   | PRI | NULL    |       |
| Cat_tittle | char(30) | YES  |     | NULL    |       |
+------------+----------+------+-----+---------+-------+
2 rows in set (0.04 sec)
```

Fig 2.3.4 Created table named Categories with its respective attributes

```
mysql> create table product(product_id int primary key,product_date date,Product_tittle char(30),Product_price int,Product_imag char(30),Product_status char(30),Product_keyword int unique);
Query OK, 0 rows affected (0.03 sec)

mysql> desc product;
+----------------+---------+------+-----+---------+-------+
| Field          | Type    | Null | Key | Default | Extra |
+----------------+---------+------+-----+---------+-------+
| product_id     | int(11) | NO   | PRI | NULL    |       |
| product_date   | date    | YES  |     | NULL    |       |
| Product_tittle | char(30)| YES  |     | NULL    |       |
| Product_price  | int(11) | YES  |     | NULL    |       |
| Product_imag   | char(30)| YES  |     | NULL    |       |
| Product_status | char(30)| YES  |     | NULL    |       |
| Product_keyword| int(11) | YES  | UNI | NULL    |       |
+----------------+---------+------+-----+---------+-------+
7 rows in set (0.01 sec)

mysql>
```

Fig 2.3.5 Created table named Product with its respective attributes

```
mysql> create table cart(qty int,Ip_add char(30) unique,cust_id int,foreign key(cust_id) references customer(customer_id),P_id int ,foreign key(P_id) references Product(Product_id));
Query OK, 0 rows affected (0.02 sec)

mysql> desc cart;
+---------+---------+------+-----+---------+-------+
| Field   | Type    | Null | Key | Default | Extra |
+---------+---------+------+-----+---------+-------+
| qty     | int(11) | YES  |     | NULL    |       |
| Ip_add  | char(30)| YES  | UNI | NULL    |       |
| cust_id | int(11) | YES  | MUL | NULL    |       |
| P_id    | int(11) | YES  | MUL | NULL    |       |
+---------+---------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

Fig 2.3.6 Created table named cart with its respective attributes

```
nces Customer(customer_i" at line 1
mysql> Create table Selects(sCustomer_id int,foreign key(sCustomer_id) references Customer(Customer_id),sCat_id int,foreign key(sCat_id) references Categories(Cat_id));
Query OK, 0 rows affected (0.02 sec)

mysql> desc Selects;
+-------------+---------+------+-----+---------+-------+
| Field       | Type    | Null | Key | Default | Extra |
+-------------+---------+------+-----+---------+-------+
| sCustomer_id| int(11) | YES  | MUL | NULL    |       |
| sCat_id     | int(11) | YES  | MUL | NULL    |       |
+-------------+---------+------+-----+---------+-------+
2 rows in set (0.02 sec)
```

Fig 2.3.7 Created table named Selects with its respective attributes

```
mysql> Create table Buy(bCustomer_id int,foreign key(bCustomer_id) references Customer(Customer_id),bProduct_id int,foreign key(bProduct_id) references Product(Product_id));
Query OK, 0 rows affected (0.32 sec)

mysql> desc Buy;
+-------------+---------+------+-----+---------+-------+
| Field       | Type    | Null | Key | Default | Extra |
+-------------+---------+------+-----+---------+-------+
| bCustomer_id| int(11) | YES  | MUL | NULL    |       |
| bProduct_id | int(11) | YES  | MUL | NULL    |       |
+-------------+---------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

Fig 2.3.8 Created table named Buy with its respective attributes

```
mysql> Create table Managed_by(mProduct_id int,foreign key(mProduct_id) references Product(product_id),mUser_id int,foreign key(mUser_id) references admin(User_id));
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> desc Managed_by;
+-------------+---------+------+-----+---------+-------+
| Field       | Type    | Null | Key | Default | Extra |
+-------------+---------+------+-----+---------+-------+
| mProduct_id | int(11) | YES  | MUL | NULL    |       |
| mUser_id    | int(11) | YES  | MUL | NULL    |       |
+-------------+---------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

Fig 2.3.9 Created table named Managed_by with its respective attributes.

Java Code

```
    // End of variables declaration
}
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package onlineshopping;

/**
 *
 * @author Deepak
 */
public class Online_Shopping extends javax.swing.JFrame {
    Connection con = null;
    Statement stmt = null;
    ResultSet rs = null;


    /**
     * Creates new form Online_Shopping
     */
    public Online_Shopping() {
        initComponents();

    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jComboBox1 = new javax.swing.JComboBox();
        Customer = new javax.swing.JLabel();
        Customer_Id = new javax.swing.JLabel();
```

```
        Customer_IP = new javax.swing.JLabel();
        Customer_Name = new javax.swing.JLabel();
        Customer_Email = new javax.swing.JLabel();
        Customer_Country = new javax.swing.JLabel();
        Customer_Password = new javax.swing.JLabel();
        Customer_Contact = new javax.swing.JLabel();
        Customer_id = new javax.swing.JTextField();
        Customer_ip = new javax.swing.JTextField();
        jTextField3 = new javax.swing.JTextField();
        Customer_email = new javax.swing.JTextField();
        Customer_country = new javax.swing.JTextField();
        Customer_password = new javax.swing.JTextField();
        Customer_contact = new javax.swing.JTextField();
        jScrollPane1 = new javax.swing.JScrollPane();
        jTable1 = new javax.swing.JTable();
        Insert = new javax.swing.JButton();
        Delete = new javax.swing.JButton();

        jComboBox1.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "Item 1", "Item 2", "Item 3", "Item 4"
}));

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        Customer.setText("Customer");

        Customer_Id.setText("Customer ID");

        Customer_IP.setText("Customer IP");

        Customer_Name.setText("Customer Name");

        Customer_Email.setText("Customer email ");

        Customer_Country.setText("Customer Country");

        Customer_Password.setText("Customer Password");

        Customer_Contact.setText("Customer Contact");

        Customer_country.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                Customer_countryActionPerformed(evt);
            }
        });

        Customer_password.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                Customer_passwordActionPerformed(evt);
            }
        });

        Customer_contact.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                Customer_contactActionPerformed(evt);
            }
        });

        jTable1.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null, null, null, null, null, null},
                {null, null, null, null, null, null, null},
                {null, null, null, null, null, null, null},
```

```
            {null, null, null, null, null, null, null},
            {null, null, null, null, null, null, null}
         },
         new String [] {
            "Customer ID", "Customer IP", "Customer Name", "Customer Email", "Customer Country", "Customer Password",
"Customer Contact"
         }
      ) {
         Class[] types = new Class [] {
            java.lang.Integer.class, java.lang.Integer.class, java.lang.String.class, java.lang.String.class, java.lang.String.class,
java.lang.String.class, java.lang.Integer.class
         };

         public Class getColumnClass(int columnIndex) {
            return types [columnIndex];
         }
      });
      jScrollPane1.setViewportView(jTable1);

      Insert.setText("Insert");

      Delete.setText("Delete");

      javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
      getContentPane().setLayout(layout);
      layout.setHorizontalGroup(
         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
         .addGroup(layout.createSequentialGroup()
            .addGap(33, 33, 33)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
               .addComponent(Customer_Id)
               .addComponent(Customer_IP)
               .addComponent(Customer_Name)
               .addComponent(Customer_Email)
               .addComponent(Customer_Country)
               .addComponent(Customer_Password)
               .addComponent(Customer_Contact))
            .addGap(41, 41, 41)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
               .addComponent(Customer_contact, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 54, Short.MAX_VALUE)
               .addComponent(Customer_password, javax.swing.GroupLayout.Alignment.LEADING)
               .addComponent(Customer_country, javax.swing.GroupLayout.Alignment.LEADING)
               .addComponent(Customer_email, javax.swing.GroupLayout.Alignment.LEADING)
               .addComponent(jTextField3, javax.swing.GroupLayout.Alignment.LEADING)
               .addComponent(Customer_ip, javax.swing.GroupLayout.Alignment.LEADING)
               .addComponent(Customer_id))
            .addGap(235, 235, 235)
            .addComponent(Insert)
            .addGap(95, 95, 95)
            .addComponent(Delete)
            .addGap(0, 162, Short.MAX_VALUE))
         .addGroup(layout.createSequentialGroup()
            .addGap(155, 155, 155)
            .addComponent(Customer)
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jScrollPane1))
      );
      layout.setVerticalGroup(
         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                .addGroup(layout.createSequentialGroup()
                    .addComponent(Customer)
                    .addGap(32, 32, 32)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(Customer_Id)
                        .addComponent(Customer_id, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(Customer_IP)
                        .addComponent(Customer_ip, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(Insert)
                        .addComponent(Delete))
                    .addGap(18, 18, 18)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(Customer_Name)
                        .addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGap(18, 18, 18)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(Customer_Email)
                        .addComponent(Customer_email, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGap(18, 18, 18)
                    .addComponent(Customer_Country)
                    .addGap(0, 0, 0))
                .addComponent(Customer_country, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(Customer_Password)
                .addComponent(Customer_password, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                .addComponent(Customer_Contact)
                .addComponent(Customer_contact, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 96,
javax.swing.GroupLayout.PREFERRED_SIZE))
        );

        pack();
    }// </editor-fold>

    private void Customer_countryActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void Customer_contactActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void Customer_passwordActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }
```

```
    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(Online_Shopping.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(Online_Shopping.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(Online_Shopping.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(Online_Shopping.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Online_Shopping().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JLabel Customer;
    private javax.swing.JLabel Customer_Contact;
    private javax.swing.JLabel Customer_Country;
    private javax.swing.JLabel Customer_Email;
    private javax.swing.JLabel Customer_IP;
    private javax.swing.JLabel Customer_Id;
    private javax.swing.JLabel Customer_Name;
    private javax.swing.JLabel Customer_Password;
    private javax.swing.JTextField Customer_contact;
    private javax.swing.JTextField Customer_country;
    private javax.swing.JTextField Customer_email;
    private javax.swing.JTextField Customer_id;
    private javax.swing.JTextField Customer_ip;
    private javax.swing.JTextField Customer_password;
    private javax.swing.JButton Delete;
    private javax.swing.JButton Insert;
    private javax.swing.JComboBox jComboBox1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable jTable1;
    private javax.swing.JTextField jTextField3;
    // End of variables declaration
}
```

Front End

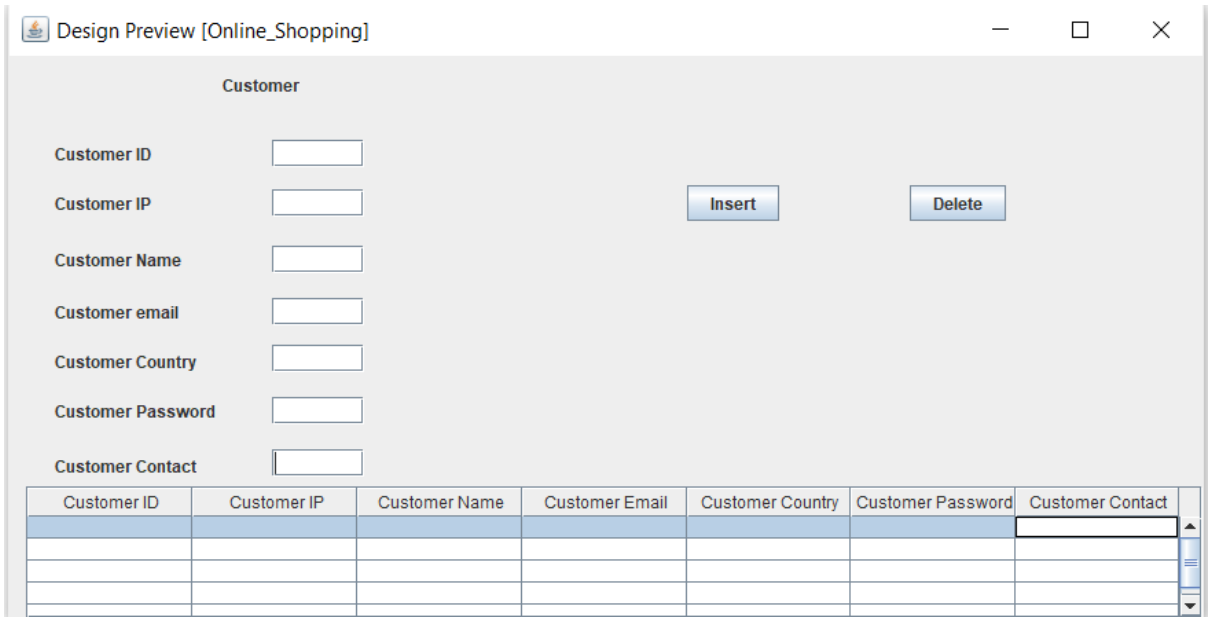


Fig 1 Table Showing Customer Registration form

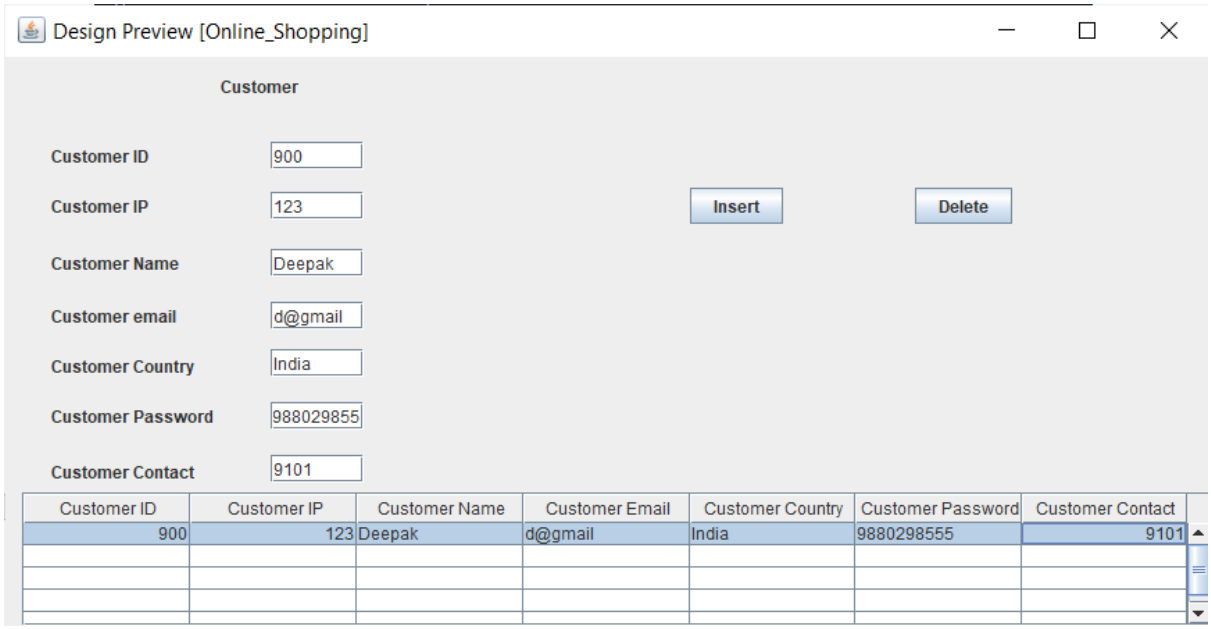


Fig 2 Table Showing insertion of Data in Customer Registration form

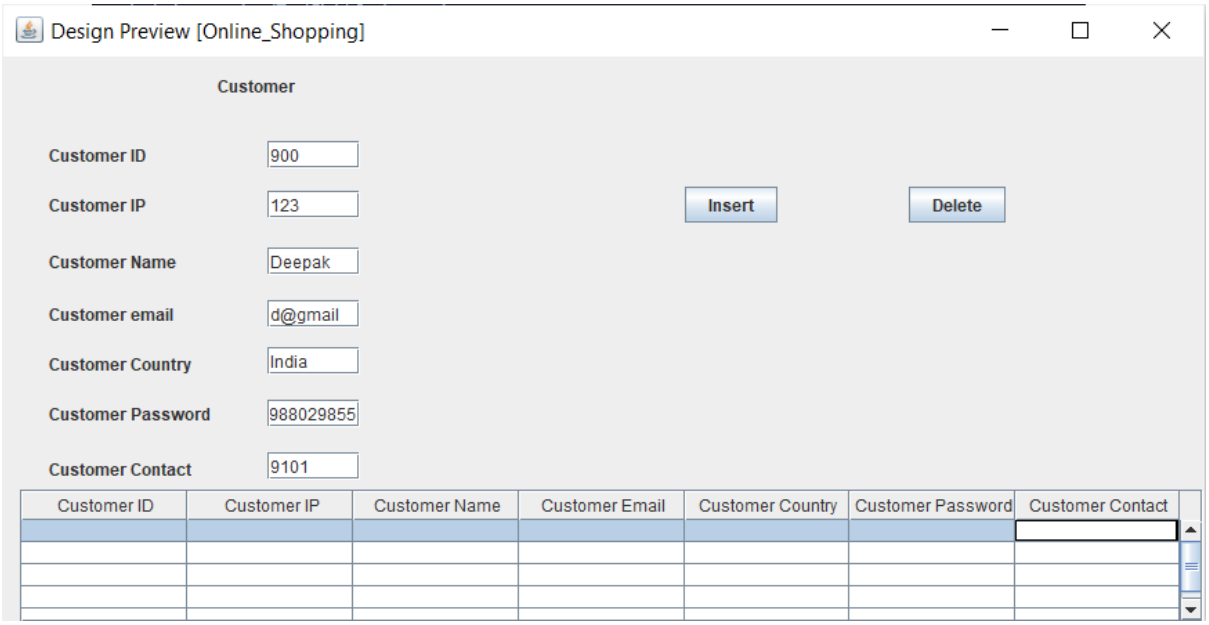


Fig 3 Table Showing Deletion of Data in Customer Registration form

Solution to 2.4

Conclusions

Java Swing is a part of Java Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java. The Java Foundation Classes (JFC) are a set of GUI components which simplify the development of desktop applications.

Unlike AWT, Java Swing provides platform-independent and lightweight components.

The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

JDBC stands for Java Database Connectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases.

The JDBC library includes APIs for each of the tasks mentioned below that are commonly associated with database usage.

- Making a connection to a database.
- Creating SQL or MySQL statements.
- Executing SQL or MySQL queries in the database.
- Viewing & Modifying the resulting records.

The java.sql and javax.sql are the primary packages for JDBC 4.0. This is the latest JDBC version at the time of writing this tutorial. It offers the main classes for interacting with your data sources.

1. **Comments**

1. Limitations of Experiments

- The database designed needs additional features like hashing for safely storing passwords
- The JDBC connector is not secured, and the transmission of data happened unsecurely

2**. Limitations of Results**

- Swing components that look like native components might not act exactly like native components
- To develop the application in swing, the individual has to be very careful with programming
- It can be slower than AWT

3**. Learning happened**

- We learnt how to use Java Swing to create GUI's in Java and connect it to the database using JDBC Driver.

4**. Recommendations**

Be careful when using Swing Components as they behave differently in the editor and when run

References

Ponnaiah,P.,2007,Data Modeling Fundamentals,Wiley.