05/05/2017

Compte Rendu n°2 : TP 4-5-6

« J'atteste que ce travail est original, qu'il indique de façon appropriée tous les emprunts, et qu'il fait référence de façon appropriée à chaque source utilisée »

GUILPAIN Léo & LEGRIS Thomas

ESIR 1 / OPTION TICB

TP n°4 :

Il fallait concevoir une base de données. Nous avons choisi d'utiliser le schéma du TD n°3 « La base Cling Clang & Co ». On a d'abord réalisé sur papier le schéma entité association pour pouvoir le simplifier au maximum.

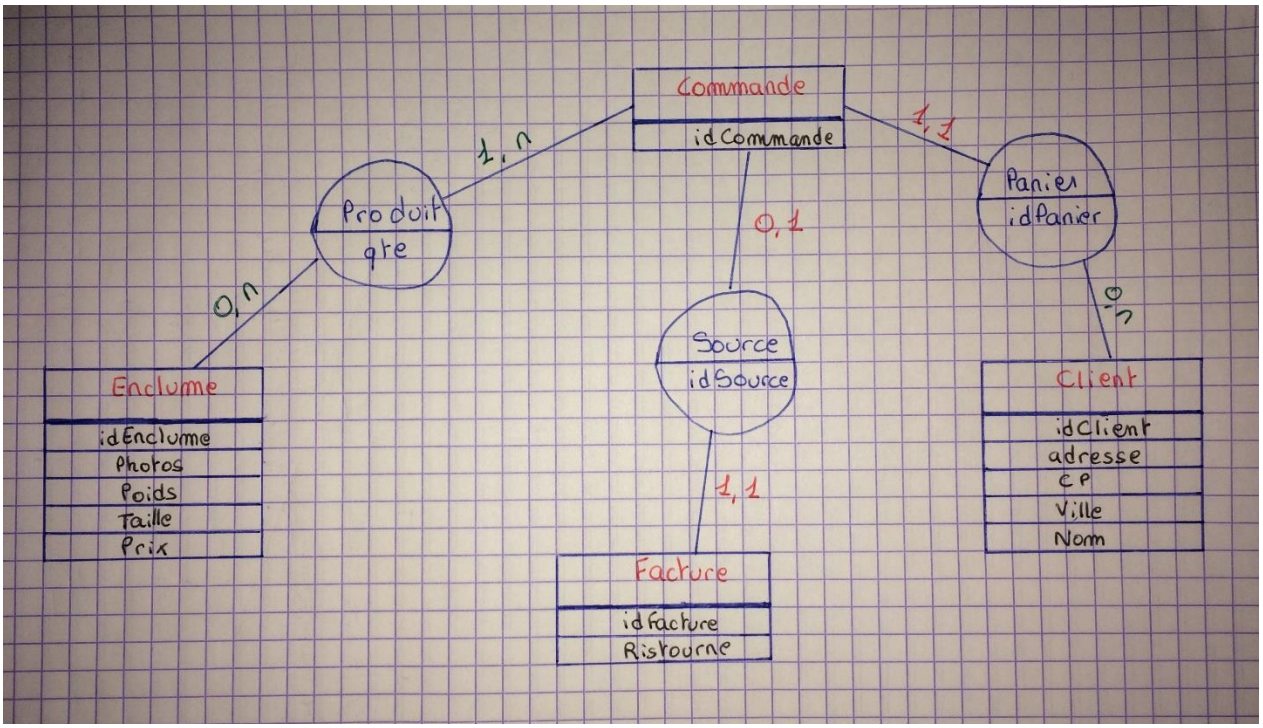


Figure 1: Schéma illustrant notre table

En effet on sait que lorsque les cardinalités sont *,1, ici en rouge. On peut fusionner entité/association.

On fusionne les associations et les tables. Si l'on fait ça, deux entités vont être reliées par un traits, or dans un schéma entité/association on ne peut pas relié 2 entités ensembles. C'est pour cela que l'on utilise EER.

Exercice n°3 :

2. Dans cet exercice, le but était d'apprendre à manipuler MySQL Workbench. Pour cela, nous avons fait simple et nous avons créé seulement une table.

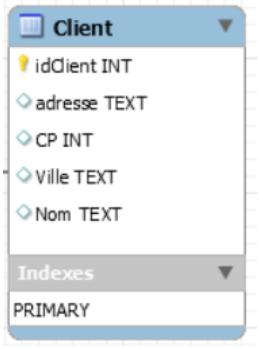


Figure 2: Table "Client"

3. Après avoir créé ce schéma, il fallait faire engendre le code SQL permettant de construire la base de données. Pour cela il suffit de faire :

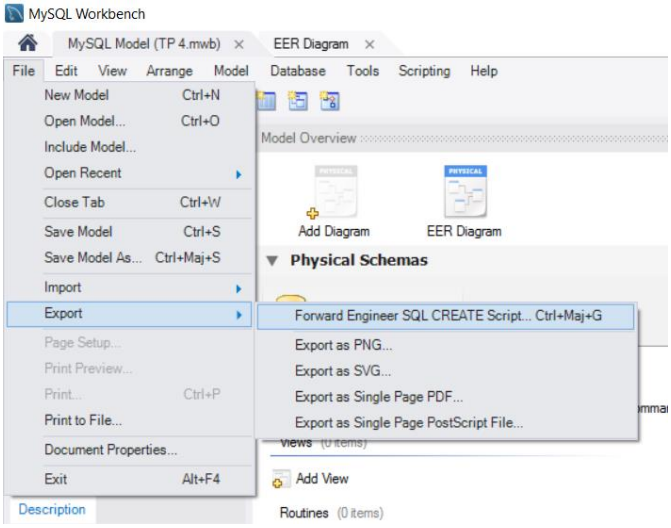


Figure 4 : 1ère étape pour construire le code

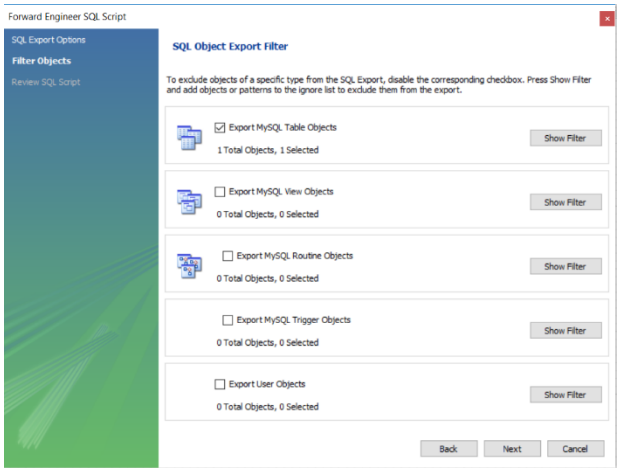


Figure 3 : 2ème étape pour construire le code

Après avoir cliqué sur next, le code MySQL apparait et pour la première table client, il est le suivant :

```
-- MySQL Script generated by MySQL Workbench
-- Tue Mar 28 11:58:51 2017
-- Model: New Model    Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
```

```
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';
-- -----------------------------------------------------
-- Schema base_17011240
-- -----------------------------------------------------
-- -----------------------------------------------------
-- Schema base_17011240
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `base_17011240` DEFAULT CHARACTER SET utf8 ;
USE `base_17011240` ;
-- -----------------------------------------------------
-- Table `base_17011240`.`Client`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `base_17011240`.`Client` (
  `idClient` INT NOT NULL,
  `adresse` TEXT NULL,s
  `CP` INT NULL,
  `Ville` TEXT NULL,
  `Nom` TEXT NULL,
  PRIMARY KEY (`idClient`))
ENGINE = InnoDB;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

4.5.6. Désormais notre code SQL est créé, il faut donc connecter notre base de données au serveur anteros.

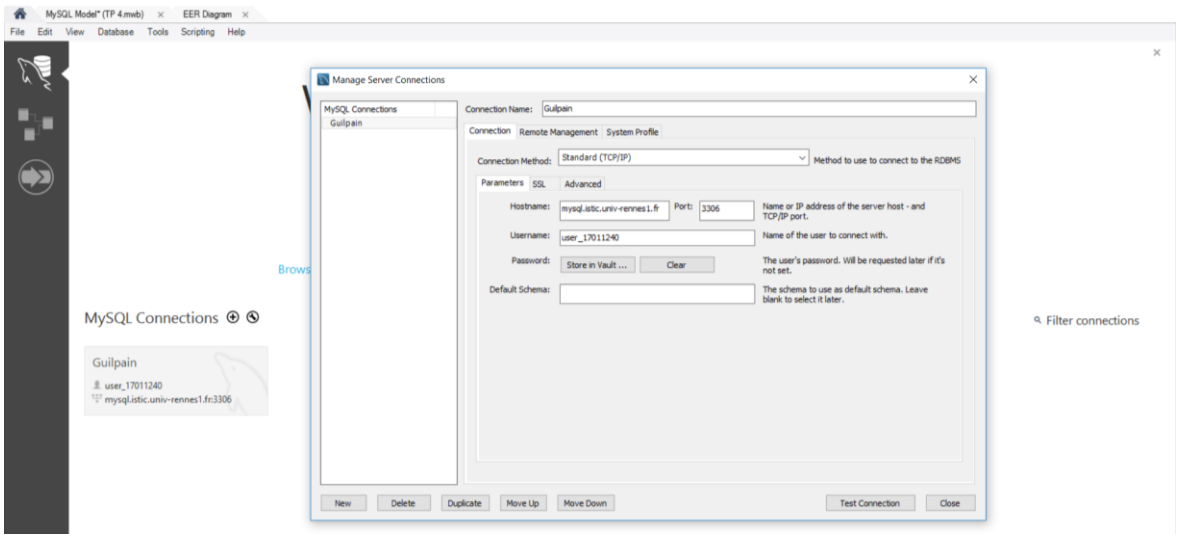


Figure 5 : Connexion au serveur

Voilà comment nous nous sommes connectés.

Maintenant que la base est créée et connectée, il suffit de faire les requêtes. Nous avons choisi d'effectuer les requêtes au moment où la base est complète. Vous les trouverez donc dans la suite de notre compte rendu.

Exercice 4 :

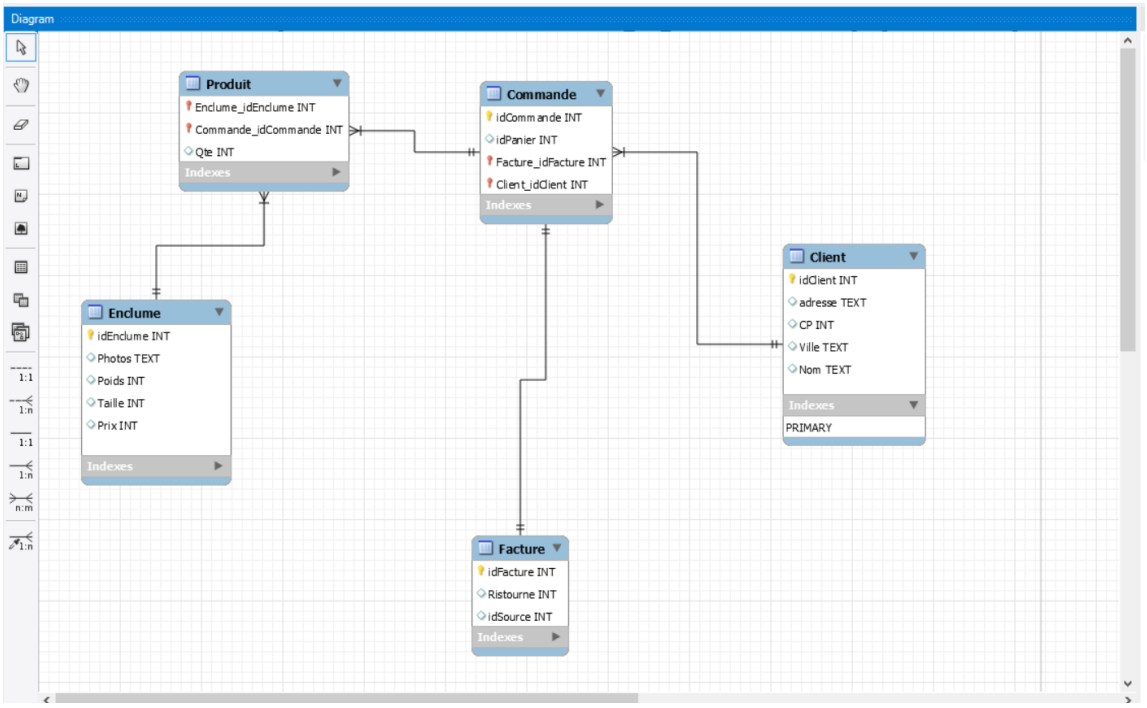


Figure 6 : Schéma final

Voilà le résultat de notre schéma créé suite aux simplifications.

Comme précédemment, une fois le schéma créé il faut faire engendrer le code, ce qui nous donne :

```
-- MySQL Script generated by MySQL Workbench
-- Tue Apr  4 14:44:30 2017
-- Model: New Model    Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----------------------------------------------------
-- Schema base_17011240
-- -----------------------------------------------------
-- -----------------------------------------------------
-- Schema base_17011240
```

```
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `base_17011240` DEFAULT CHARACTER SET utf8 ;
USE `base_17011240` ;
-- -----------------------------------------------------
-- Table `base_17011240`.`Client`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `base_17011240`.`Client` (
  `idClient` INT NOT NULL,
  `adresse` TEXT NULL,
  `CP` INT NULL,
  `Ville` TEXT NULL,
  `Nom` TEXT NULL,
  PRIMARY KEY (`idClient`))
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `base_17011240`.`Facture`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `base_17011240`.`Facture` (
  `idFacture` INT NOT NULL,
  `Ristourne` INT NULL,
  `idSource` INT NULL,
  PRIMARY KEY (`idFacture`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `base_17011240`.`Commande`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `base_17011240`.`Commande` (
  `idCommande` INT NOT NULL,
  `idPanier` INT NULL,
  `Facture_idFacture` INT NOT NULL,
  `Client_idClient` INT NOT NULL,
  PRIMARY KEY (`idCommande`, `Facture_idFacture`, `Client_idClient`),
  INDEX `fk_Commande_Facture1_idx` (`Facture_idFacture` ASC),
  INDEX `fk_Commande_Client1_idx` (`Client_idClient` ASC),
  CONSTRAINT `fk_Commande_Facture1`
    FOREIGN KEY (`Facture_idFacture`)
    REFERENCES `base_17011240`.`Facture` (`idFacture`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Commande_Client1`
    FOREIGN KEY (`Client_idClient`)
    REFERENCES `base_17011240`.`Client` (`idClient`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
```

```
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `base_17011240`.`Enclume`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `base_17011240`.`Enclume` (
  `idEnclume` INT NOT NULL,
  `Photos` TEXT NULL,
  `Poids` INT NULL,
  `Taille` INT NULL,
  `Prix` INT NULL,
  PRIMARY KEY (`idEnclume`))
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `base_17011240`.`Produit`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `base_17011240`.`Produit` (
  `Enclume_idEnclume` INT NOT NULL,
  `Commande_idCommande` INT NOT NULL,
  `Qte` INT NULL,
  PRIMARY KEY (`Enclume_idEnclume`, `Commande_idCommande`),
  INDEX `fk_Enclume_has_Commande_Commande1_idx` (`Commande_idCommande` ASC),
  INDEX `fk_Enclume_has_Commande_Enclume_idx` (`Enclume_idEnclume` ASC),
  CONSTRAINT `fk_Enclume_has_Commande_Enclume`
    FOREIGN KEY (`Enclume_idEnclume`)
    REFERENCES `base_17011240`.`Enclume` (`idEnclume`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Enclume_has_Commande_Commande1`
    FOREIGN KEY (`Commande_idCommande`)
    REFERENCES `base_17011240`.`Commande` (`idCommande`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Exercice n° 5 :

Après avoir récupéré le code SQL, il a fallu remplir notre base, voici la liste des différentes insertions effectués :

```
INSERT INTO `base_17011240`.`Client` (`idClient`, `adresse`, `CP`, `Ville`, `Nom`) VALUES ('1', 'rue dantrain', '35700', 'Rennes', 'Guilpain');
INSERT INTO `base_17011240`.`Client` (`idClient`, `adresse`, `CP`, `Ville`, `Nom`) VALUES ('2', 'rue des champs', '69000', 'Lyon', 'Greco');
INSERT INTO `base_17011240`.`Client` (`idClient`, `adresse`, `CP`, `Ville`, `Nom`) VALUES ('3', 'rue de la ville', '49000', 'Angers', 'Robin');
INSERT INTO `base_17011240`.`Client` (`idClient`, `adresse`, `CP`, `Ville`, `Nom`) VALUES ('4', 'rue de la foret', '14000', 'Caen', 'Legris');

INSERT INTO `base_17011240`.`Enclume` (`idEnclume`, `Photos`, `Poids`, `Taille`, `Prix`) VALUES ('1', 'Rouge', '20', '60', '200');
INSERT INTO `base_17011240`.`Enclume` (`idEnclume`, `Photos`, `Poids`, `Taille`, `Prix`) VALUES ('2', 'Bleu', '30', '70', '300');

INSERT INTO `base_17011240`.`Facture` (`idFacture`, `Ristourne`, `idSource`) VALUES ('1', '0', '1');
INSERT INTO `base_17011240`.`Facture` (`idFacture`, `Ristourne`, `idSource`) VALUES ('2', '0', '2');
INSERT INTO `base_17011240`.`Facture` (`idFacture`, `Ristourne`, `idSource`) VALUES ('1', '0', '1');
INSERT INTO `base_17011240`.`Facture` (`idFacture`, `Ristourne`, `idSource`) VALUES ('2', '0', '2');
INSERT INTO `base_17011240`.`Facture` (`idFacture`, `Ristourne`, `idSource`) VALUES ('3', '0', '3');
INSERT INTO `base_17011240`.`Facture` (`idFacture`, `Ristourne`, `idSource`) VALUES ('4', '0', '4');

INSERT INTO `base_17011240`.`Commande` (`idCommande`, `idPanier`, `Facture_idFacture`, `Client_idClient`) VALUES ('1', '1', '1', '1');
INSERT INTO `base_17011240`.`Commande` (`idCommande`, `idPanier`, `Facture_idFacture`, `Client_idClient`) VALUES ('2', '2', '2', '2');
INSERT INTO `base_17011240`.`Commande` (`idCommande`, `idPanier`, `Facture_idFacture`, `Client_idClient`) VALUES ('3', '3', '4', '4');
INSERT INTO `base_17011240`.`Commande` (`idCommande`, `idPanier`, `Facture_idFacture`, `Client_idClient`) VALUES ('4', '4', '3', '3');
```

Nos tables sont désormais remplies de différentes données.

Nous avons donc fait des requêtes pour voir si notre base fonctionnait correctement.

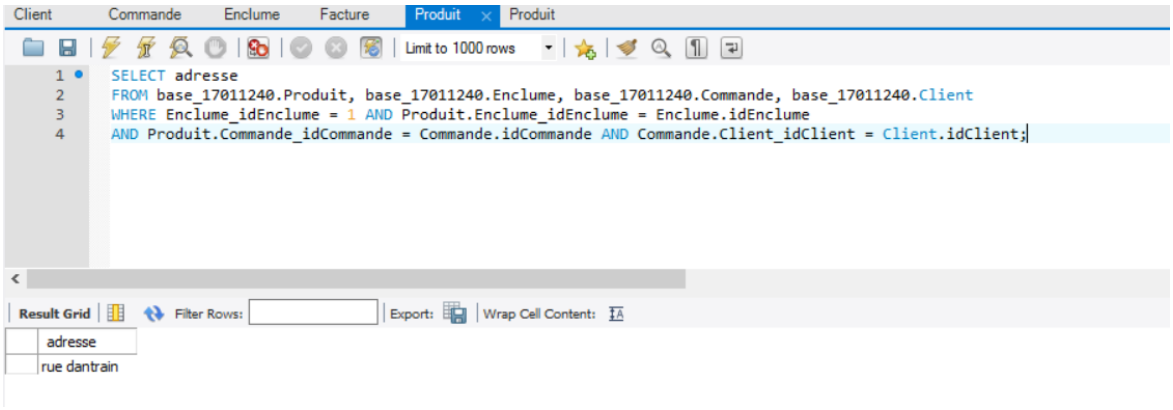


Figure 7: Requête permettant de renvoyer l'adresse du client qui a acheter l'enclume n°1

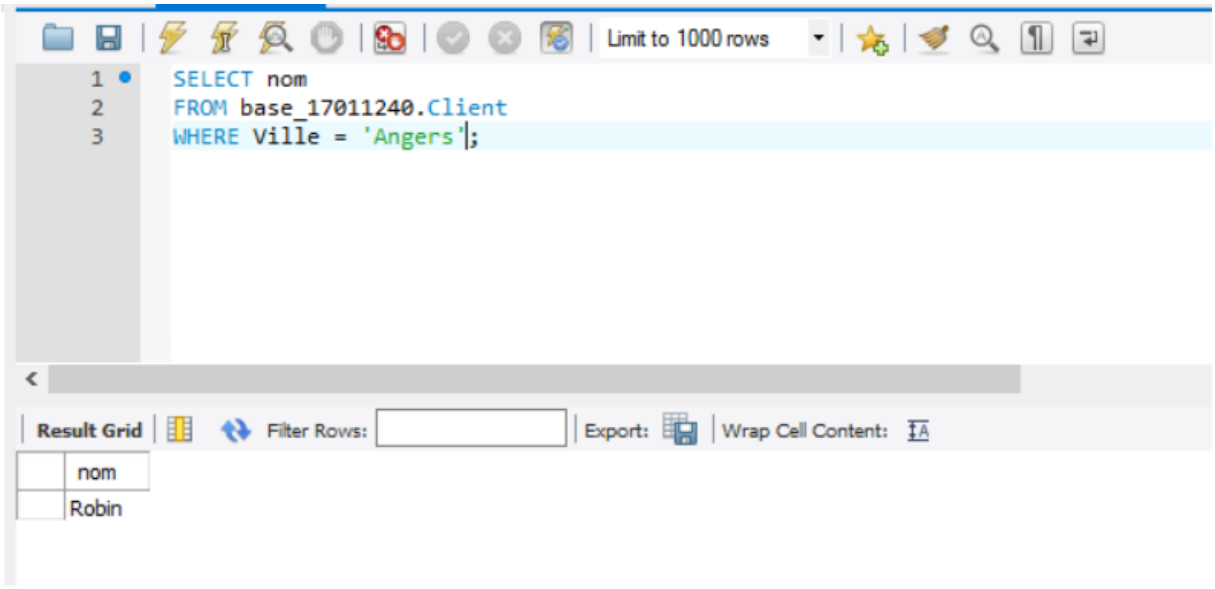


Figure 8 : Requête permettant de donner le nom du client qui vient d'Angers

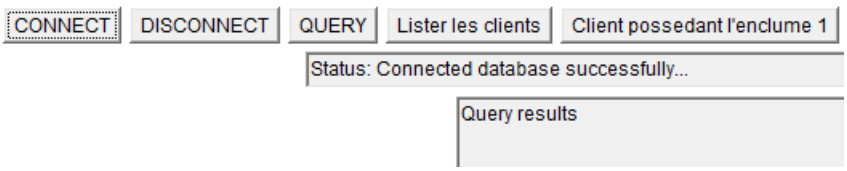
TP n° 5 & 6 :

Dans ce TP, avec notre base nous devons concevoir une interface JDBC afin de pouvoir créer des requêtes.

La première étape est de se connecter à la base de données.

```
private void connectToDatabase(){
    // Connection à la base de donnée

    // JDBC driver name and database URL
    final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    final String DB_URL = "jdbc:mysql://anteros.istic.univ-rennes1.fr:3306/base_17011240";

    //  Database credentials
    final String USER = "user_17011240";
    final String PASS = "leoguilpain11";

    try{
        //STEP 2: Register JDBC driver
        Class.forName(JDBC_DRIVER);

        //STEP 3: Open a connection
        System.out.println("Connecting to database...");
        conn = DriverManager.getConnection(DB_URL,USER,PASS);
        System.out.println("Connected database successfully...");
        setStatus("Connected database successfully...");
    }
    catch (Exception e){
        setStatus("Connexion failed");
        System.out.println("Connexion failed");
    }

}
```

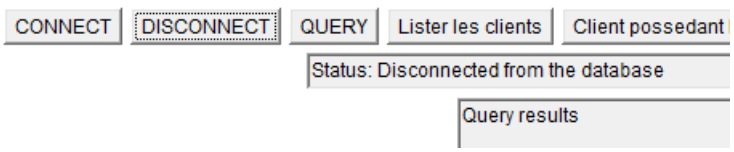
A l'aide de cette fonction, on se connecte à la base_17011240.



Après avoir cliqué sur le bouton Connect, on voit bien que le statut est « connected database successfully.. » donc la connexion a bien été effectuée.

Après la connexion, il faut être capable de se deconnecter.

```
private void disconnectFromDatabase(){
    try{
    setStatus("Disconnected from the database");
    System.out.println("Disconnected from the database");
    } catch(Exception e){
        System.err.println(e.getMessage());
        setStatus("Disconnection failed");
    }
}
```

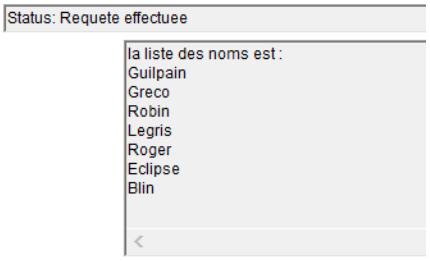


Après avoir cliqué sur le bouton Disconnect, on voit bien que le statut est « Disconnected from the database » donc la déconnexion a bien été effectuée.

Nous avons donc réussi à se connecter et à se déconnecter de la base, il faut maitenant l'utiliser.
Par la suite on va réaliser des requêtes et des insertions dans les tables.
Voici à quoi ressemble notre interface.

Pour le bouton « Lister les clients » , on a :

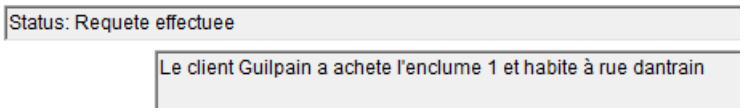
```
private void queryDatabase1(){
    try{
        System.out.println("Creating statement...");
        stmt = conn.createStatement();
        String sql;
        sql = "SELECT nom FROM Client";
        ResultSet rs = stmt.executeQuery(sql);
        mRes.setText("la liste des noms est :\n");
        //STEP 5: Extract data from result set
        while(rs.next()){
            //Retrieve by column name
            String nom = rs.getString("nom");
            System.out.println("nom : " + nom);
            mRes.append(nom + "\n");
            setStatus("Requete effectuee");
        }
        rs.close();
        stmt.close();
        conn.close();
    }
    catch(SQLException se){
        //Handle errors for JDBC
        se.printStackTrace();
    }   catch(Exception e){
        //Handle errors for Class.forName
        e.printStackTrace();
    }   finally{
        //finally block used to close resources
        try{
            if(stmt!=null)
                stmt.close();
        }   catch(SQLException se2){
        }// nothing we can do
        try{
            if(conn!=null)
                conn.close();
        }   catch(SQLException se){
            se.printStackTrace();
        }//end finally try
    }//end try
}
```



La requête permet de lister le nom des clients présent dans la base Client. On voit bien d'après l'interface que la requête est bien correctement effectuée.

Pour le bouton « Client possedant l'enclume 1 », on a :

```
private void queryDatabase2(){
    try{
        System.out.println("Creating statement...");
        stmt = conn.createStatement();
        String sql;
        sql = "SELECT adresse,nom FROM Produit, Enclume, Commande, "
                + "Client WHERE Enclume_idEnclume = 1 AND Produit.Enclume_idEnclume = Enclume.idEnclume "
                + "AND Produit.Commande_idCommande = Commande.idCommande AND Commande.Client_idClient = Client.idClient";
        ResultSet rs = stmt.executeQuery(sql);

        //STEP 5: Extract data from result set
        while(rs.next()){
            //Retrieve by column name
            String nom = rs.getString("nom");
            String adresse = rs.getString("adresse");
            mRes.setText("Le client " + nom + " a achete l'enclume 1 et habite à " + adresse);
            setStatus("Requete effectuee");
        }
        rs.close();
        stmt.close();
        conn.close();
    }
    catch(SQLException se){
        //Handle errors for JDBC
        se.printStackTrace();
    }   catch(Exception e){
        //Handle errors for Class.forName
        e.printStackTrace();
    }   finally{
        //finally block used to close resources
        try{
            if(stmt!=null)
                stmt.close();
        }   catch(SQLException se2){
        }// nothing we can do
        try{
            if(conn!=null)
                conn.close();
        }   catch(SQLException se){
            se.printStackTrace();
        }//end finally try
    }//end try
```



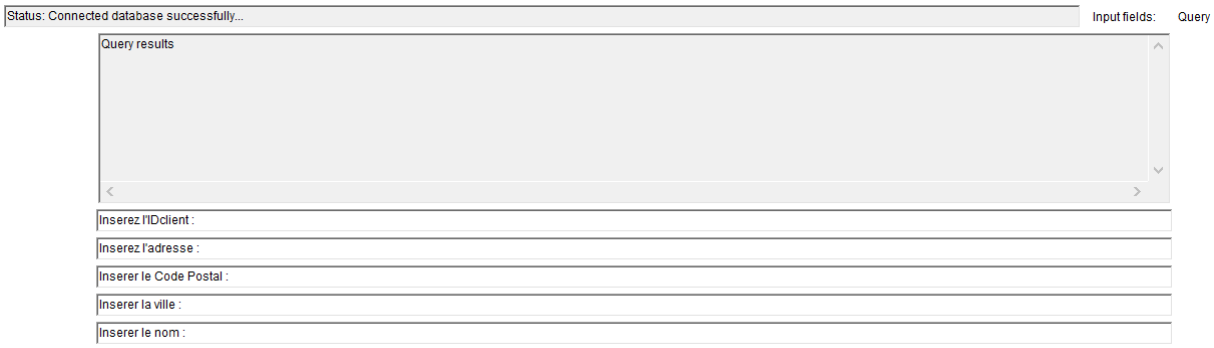
La requête permet de retrouver le nom du client ayant acheté l'enclume 1 et de savoir à quelle adresse il se situe. On voit bien d'après l'interface que la requête est bien correctement effectuée.

Maintenant que nous avons fait les différentes requêtes, nous allons passer aux insertions dans les tables.
L'interface possède les différentes tables, les boutons permettent de faire apparaitre les différents champs pour faire les insertions. Nous allons vous présenter une insertion car la méthode est la même pour tous.

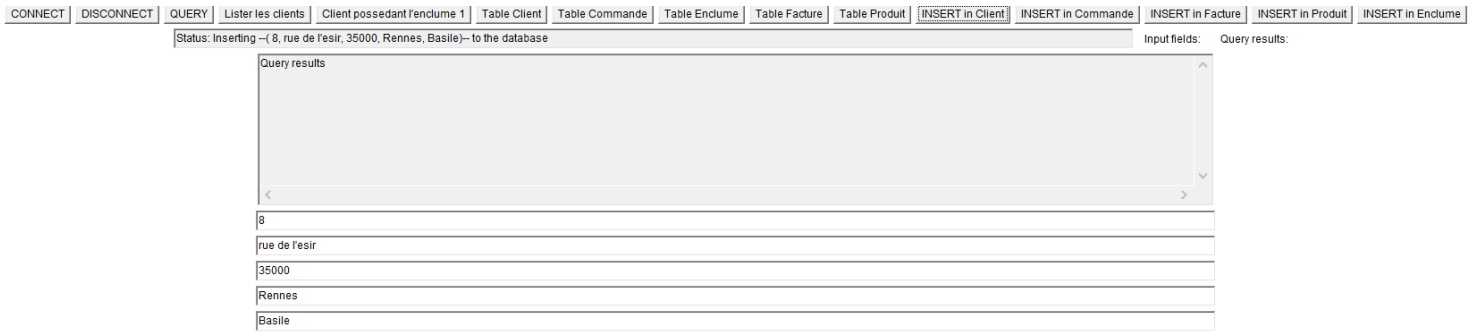
Insertion dans la table « Client » :

```
private void insertDatabaseClient(){
    try{
        stmt = conn.createStatement();
        int idClient = Integer.parseInt(m1.getText());
        String adresse = m2.getText();
        int CP = Integer.parseInt(m3.getText());
        String Ville = m4.getText();
        String Nom = m5.getText();
        String sql = "INSERT INTO Client " +
                     "VALUES( " + idClient + ", \"" + adresse + "\", " + CP + ", \""+ Ville + "\", \"" + Nom + "\")";
        System.out.println(sql);
        stmt.executeUpdate(sql);
        setStatus("Inserting --( " + idClient + ", " + adresse + ", " + CP + ", " + Ville + ", " + Nom + ")-- to the database");
        } catch(Exception e){
            System.err.println(e.getMessage());
            setStatus("Insertion failed");
        }
}
    //Selectionne la table Client
    if (cause == b6){
         add(m1);
         add(m2);
         add(m3);
         add(m4);
         add(m5);
         m1.setText("Inserez l'IDclient :");  //According to the database schema
         m2.setText("Inserez l'adresse : "); //According to the database schema
         m3.setText("Inserer le Code Postal : ");//According to the database schema
         m4.setText("Inserer la ville : ") ;
         m5.setText("Inserer le nom : ");
    }
```

Avec ce code, lorsque nous cliquons sur le bouton « Table Client », nous obtenons ce resultat sur l'interface :



Après avoir rentré les différentes valeurs dans les champs correspondants il suffit de cliquer sur « INSERT in Client ».



On voit bien que l'insertion a bien été effectuée.

Conclusion :

Les séances de la séquence 2 ont eu pour but de créer des bases de données, mais aussi, et c'est cela qui différencie les 2 séquences, de connecter une base à un serveur en ligne et de fabriquer une interface graphique. Nous nous sommes d'abord habitués au logiciel MySQL Workbench, puis nous avons créé un schéma entité association en utilisant l'exercice sur les enclumes traité durant le TD3, nous l'avons dessiné sur ce logiciel puis nous l'avons connecté au serveur Anteros. Seulement, pour qu'un utilisateur puisse accéder aux différentes tables, requêtes, nous avons créé une interface Java pour l'utilisateur, reliée à cette base de données. Nous avons donc réalisé un programme permettant à l'utilisateur de se connecter à la base de données, de cliquer sur des boutons afin d'obtenir les requêtes qui lui sont utiles. Nous nous sommes mis à la place du développeur et de l'utilisateur, pour faciliter la vie des personnes qui utiliseront cet interface JDBC, si c'est une personne de l'entreprise elle peut retrouver à n'importe quel instant la commande du client X par exemple, ou bien son adresse. Nous pouvons appliquer les méthodes suivies dans cette séquence de TP pour n'importe quelles entreprises afin de leur proposer un outil stable et fonctionnel.

Annexe :

DatabaseUserInterface.java | test.java | DatabaseUserInterface.java

```java
package tp5;


import java.awt.*;

/**
 * This is a skeleton for realizing a very simple database user interface in java.
 * The interface is an Applet, and it implements the interface ActionListener.
 * If the user performs an action (for example he presses a button), the procedure actionPerformed
 * is called. Depending on his actions, one can implement the database connection (disconnection),
 * querying or insert.
 *
 * @author zmiklos
 *
 */
public class DatabaseUserInterface extends java.applet.Applet implements ActionListener {

  private TextField mStat, m1, m2, m3, m4, m5;
  private TextArea mRes;
  private Button b1, b2, b3, b4, b5, b6, b7, b8, b9, b10, b11, b12, b13, b14, b15;
  private static final long serialVersionUID = 1L;
  private Statement stmt;
  private Connection conn;



  /**
   * This procedure is called when the Applet is initialized.
   *
   */
  public void init ()
  {
      /**
       * Definition of text fields
       */
      //m1 = new TextField(80);
      //m1.setText("What are you going to do when the light is:");
      //m1.setEditable(false);
      mStat = new TextField(150);
      mStat.setEditable(false);
      m1 = new TextField(150);
      m2 = new TextField(150);
      m3 = new TextField(150);
```

DatabaseUserInterface.java | test.java | DatabaseUserInterface.java

```java
      m3 = new TextField(150);
      m4 = new TextField(150);
      m5 = new TextField(150);
      mRes = new TextArea(10,150);
      mRes.setEditable(false);



      /**
       * First we define the buttons, then we add to the Applet, finally add and ActionListener
       * (with a self-reference) to capture the user actions.
       */
      b1 = new Button("CONNECT");
      b2 = new Button("DISCONNECT");
      b3 = new Button("QUERY");
      b4 = new Button("Lister les clients");
      b5 = new Button("Client possedant l'enclume 1");
      b6 = new Button("Table Client");
      b7 = new Button("Table Commande");
      b8 = new Button("Table Enclume");
      b9 = new Button("Table Facture");
      b10 = new Button("Table Produit");
      b11 = new Button("INSERT in Client");
      b12 = new Button("INSERT in Commande");
      b13 = new Button("INSERT in Facture");
      b14 = new Button("INSERT in Produit");
      b15 = new Button("INSERT in Enclume");
      add(b1) ;
      add(b2) ;
      add(b3) ;
      add(b4);
      add(b5);
      add(b6);
      add(b7);
      add(b8);
      add(b9);
      add(b10);
      add(b11);
      add(b12);
      add(b13);
      add(b14);
      add(b15);
      b1.addActionListener(this);
```

DatabaseUserInterface.java | test.java | DatabaseUserInterface.java

```
90      add(b15);
91      b1.addActionListener(this);
92      b2.addActionListener(this);
93      b3.addActionListener(this);
94      b4.addActionListener(this);
95      b5.addActionListener(this);
96      b6.addActionListener(this);
97      b7.addActionListener(this);
98      b8.addActionListener(this);
99      b9.addActionListener(this);
100     b10.addActionListener(this);
101     b11.addActionListener(this);
102     b12.addActionListener(this);
103     b13.addActionListener(this);
104     b14.addActionListener(this);
105     b15.addActionListener(this);
106     add(mStat);
107     add(new Label("Input fields: ", Label.CENTER));
108     add(new Label("Query results: ", Label.CENTER));
109     add(mRes);
110     mRes.setText("Query results");
111
112     setStatus("Waiting for user actions.");
113 }
114
115
116 /**
117  * This procedure is called upon a user action.
118  *
119  *  @param event The user event.
120  */
121  public void actionPerformed(ActionEvent event)
122 {
123
124      // Extract the relevant information from the action (i.e. which button is pressed?)
125      Object cause = event.getSource();
126
127      // Act depending on the user action
128      // Button CONNECT
129      if (cause == b1)
130      {
131          connectToDatabase();
132      }
```

DatabaseUserInterface.java | test.java | DatabaseUserInterface.java

```
134         // Button DISCONNECT
135         if (cause == b2)
136         {
137             disconnectFromDatabase();
138         }
139
140         //Button QUERY
141         if (cause == b3)
142         {
143             queryDatabase1();
144         }
145
146         if (cause == b4)
147         {
148             queryDatabase1();
149         }
150
151         if (cause == b5)
152         {
153             queryDatabase2();
154         }
155
156         //Selectionne la table Client
157         if (cause == b6){
158             add(m1);
159             add(m2);
160             add(m3);
161             add(m4);
162             add(m5);
163             m1.setText("Inserez l'IDclient :");  //According to the database schema
164             m2.setText("Inserez l'adresse : "); //According to the database schema
165             m3.setText("Inserer le Code Postal : ");//According to the database schema
166             m4.setText("Inserer la ville : ") ;
167             m5.setText("Inserer le nom : ");
168         }
169
170         //Sélectionne la table Commande
171         if (cause == b7){
172             add(m1);
173             add(m2);
174             add(m3);
175             add(m4);
```

DatabaseUserInterface.java | test.java | DatabaseUserInterface.java

```
175             add(m4);
176             m1.setText("Inserez l'IDcommande :");
177             m2.setText("Inserez l'IDpanier : ");
178             m3.setText("Inserez Facture_idFacture : ");
179             m4.setText("Inserez Client_idClient : ");
180         }
181
182         //Sélectionne la table Enclume
183         if (cause == b8){
184             add(m1);
185             add(m2);
186             add(m3);
187             add(m4);
188             add(m5);
189             m1.setText("Inserez l'IDenclume :");
190             m2.setText("Inserez la photo : ");
191             m3.setText("Inserez le poids : ");
192             m4.setText("Inserez la taille : ");
193             m5.setText("Inserez le prix : ");
194         }
195
196         //Sélectionne la table Facture
197         if (cause == b9){
198             add(m1);
199             add(m2);
200             add(m3);
201             m1.setText("Inserez l'IDfacture :");
202             m2.setText("Inserez la Ristourne : ");
203             m3.setText("Inserez l'Idsource : ");
204         }
205
206         //Sélectionne la table Produit
207         if (cause == b10){
208             add(m1);
209             add(m2);
210             add(m3);
211             m1.setText("Inserez l'Enclume_idEnclume :");
212             m2.setText("Inserez la Commande_idCommande : ");
213             m3.setText("Inserez la Qte : ");
214         }
215
216       if (cause == b11)
217       {
```

DatabaseUserInterface.java | test.java | DatabaseUserInterface.java

```
213             m3.setText("Inserez la Qte : ");
214         }
215
216       if (cause == b11)
217       {
218           insertDatabaseClient();
219       }
220
221       if (cause == b12)
222       {
223           insertDatabaseCommande();
224       }
225
226       if (cause == b13)
227       {
228           insertDatabaseFacture();
229       }
230
231       if (cause == b14)
232       {
233           insertDatabaseProduit();
234       }
235
236       if (cause == b15)
237       {
238           insertDatabaseEnclume();
239       }
240
241   }
242
243
244 /**
245  * Set the status text.
246  *
247  * @param text The text to set.
248  */
249 private void setStatus(String text){
250         mStat.setText("Status: " + text);
251   }
252
253 /**
254  * Procedure, where the database connection should be implemented.
255  */
```

DatabaseUserInterface.java | test.java | DatabaseUserInterface.java

```java
private void connectToDatabase(){
    // Connection à la base de donnée

    // JDBC driver name and database URL
    final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    final String DB_URL = "jdbc:mysql://anteros.istic.univ-rennes1.fr:3306/base_17011240";

    //  Database credentials
    final String USER = "user_17011240";
    final String PASS = "leoguilpain11";

    try{
        //STEP 2: Register JDBC driver
        Class.forName(JDBC_DRIVER);

        //STEP 3: Open a connection
        System.out.println("Connecting to database...");
        conn = DriverManager.getConnection(DB_URL,USER,PASS);
        System.out.println("Connected database successfully...");
        setStatus("Connected database successfully...");
    }
    catch (Exception e){
        setStatus("Connexion failed");
        System.out.println("Connexion failed");
    }

}


/**
 * Procedure, where the database connection should be implemented.
 */
private void disconnectFromDatabase(){
    try{
    setStatus("Disconnected from the database");
    System.out.println("Disconnected from the database");
    } catch(Exception e){
        System.err.println(e.getMessage());
        setStatus("Disconnection failed");
    }
}

```

DatabaseUserInterface.java | test.java | DatabaseUserInterface.java

```java
private void queryDatabase1(){
    try{
        System.out.println("Creating statement...");
        stmt = conn.createStatement();
        String sql;
        sql = "SELECT nom FROM Client";
        ResultSet rs = stmt.executeQuery(sql);
        mRes.setText("la liste des noms est :\n");
        //STEP 5: Extract data from result set
        while(rs.next()){
            //Retrieve by column name
            String nom = rs.getString("nom");
            System.out.println("nom : " + nom);
            mRes.append(nom + "\n");
            setStatus("Requete effectuee");
        }
        rs.close();
        stmt.close();
        conn.close();
    }
    catch(SQLException se){
        //Handle errors for JDBC
        se.printStackTrace();
    }   catch(Exception e){
        //Handle errors for Class.forName
        e.printStackTrace();
    }   finally{
        //finally block used to close resources
        try{
            if(stmt!=null)
                stmt.close();
        }   catch(SQLException se2){
        }// nothing we can do
        try{
            if(conn!=null)
                conn.close();
        }   catch(SQLException se){
            se.printStackTrace();
        }//end finally try
    }//end try
}

```

DatabaseUserInterface.java | test.java | DatabaseUserInterface.java

```
346 private void queryDatabase2(){
347     try{
348         System.out.println("Creating statement...");
349         stmt = conn.createStatement();
350         String sql;
351         sql = "SELECT adresse,nom FROM Produit, Enclume, Commande, "
352                 + "Client WHERE Enclume_idEnclume = 1 AND Produit.Enclume_idEnclume = Enclume.idEnclume "
353                 + "AND Produit.Commande_idCommande = Commande.idCommande AND Commande.Client_idClient = Client.idClient";
354         ResultSet rs = stmt.executeQuery(sql);
355 
356         //STEP 5: Extract data from result set
357         while(rs.next()){
358             //Retrieve by column name
359             String nom = rs.getString("nom");
360             String adresse = rs.getString("adresse");
361             mRes.setText("Le client " + nom + " a achete l'enclume 1 et habite à " + adresse);
362             setStatus("Requete effectuee");
363         }
364         rs.close();
365         stmt.close();
366         conn.close();
367     }
368     catch(SQLException se){
369         //Handle errors for JDBC
370         se.printStackTrace();
371     }   catch(Exception e){
372         //Handle errors for Class.forName
373         e.printStackTrace();
374     }   finally{
375         //finally block used to close resources
376         try{
377             if(stmt!=null)
378                 stmt.close();
379         }   catch(SQLException se2){
380         }// nothing we can do
381         try{
382             if(conn!=null)
383                 conn.close();
384         }   catch(SQLException se){
385             se.printStackTrace();
386         }//end finally try
387     }//end try
```

DatabaseUserInterface.java | test.java | DatabaseUserInterface.java

```
388 }
389 
390 
391 
392 private void insertDatabaseClient(){
393     try{
394         stmt = conn.createStatement();
395         int idClient = Integer.parseInt(m1.getText());
396         String adresse = m2.getText();
397         int CP = Integer.parseInt(m3.getText());
398         String Ville = m4.getText();
399         String Nom = m5.getText();
400         String sql = "INSERT INTO Client " +
401                         "VALUES( " + idClient + ", \"" + adresse + "\", " + CP + ", \""+ Ville + "\", \"" + Nom + "\")";
402         System.out.println(sql);
403         stmt.executeUpdate(sql);
404         setStatus("Inserting --( " + idClient + ", " + adresse + ", " + CP + ", " + Ville + ", " + Nom + ")-- to the database");
405         } catch(Exception e){
406             System.err.println(e.getMessage());
407             setStatus("Insertion failed");
408         }
409 }
410 
411 private void insertDatabaseCommande(){
412     try{
413         stmt = conn.createStatement();
414         int idCommande = Integer.parseInt(m1.getText());
415         int idPanier = Integer.parseInt(m2.getText());
416         int Facture_idFacture = Integer.parseInt(m3.getText());
417         int Client_idClient = Integer.parseInt(m4.getText());
418 
419         String sql = "INSERT INTO Commande " +
420                         "VALUES( " + idCommande + ", " + idPanier + ", " + Facture_idFacture + ", " + Client_idClient + ")";
421         System.out.println(sql);
422         stmt.executeUpdate(sql);
423         setStatus("Inserting --( " + idCommande + ", " + idPanier + ", " + Facture_idFacture + ", " + Client_idClient + ")-- to the database");
424         } catch(Exception e){
425             System.err.println(e.getMessage());
426             setStatus("Insertion failed");
427         }
428 }
429 
```

DatabaseUserInterface.java | test.java | DatabaseUserInterface.java

```
private void insertDatabaseFacture(){
    try{
        stmt = conn.createStatement();
        int idFacture = Integer.parseInt(m1.getText());
        int Ristourne = Integer.parseInt(m2.getText());
        int idSource = Integer.parseInt(m3.getText());
        String sql = "INSERT INTO Facture " +
                           "VALUES( " + idFacture + ", " + Ristourne + ", " + idSource + ")";
        System.out.println(sql);
        stmt.executeUpdate(sql);
        setStatus("Inserting --( " + idFacture + ", " + Ristourne + ", " + idSource + ")-- to the database");
        } catch(Exception e){
            System.err.println(e.getMessage());
            setStatus("Insertion failed");
        }
}

private void insertDatabaseEnclume(){
    try{
        stmt = conn.createStatement();
        int idEnclume = Integer.parseInt(m1.getText());
        String Photos = m2.getText();
        int Poids = Integer.parseInt(m3.getText());
        int Taille = Integer.parseInt(m4.getText());
        int Prix = Integer.parseInt(m5.getText());
        String sql = "INSERT INTO Enclume " +
                "VALUES( " + idEnclume + ", \"" + Photos + "\", " + Poids + ", " + Taille + ", " + Prix + ")";
        System.out.println(sql);
        stmt.executeUpdate(sql);
        setStatus("Inserting --( " + idEnclume + ", " + Photos + ", " + Poids + ", " + Taille + ", " + Prix + ")-- to the database");
    } catch(Exception e){
        System.err.println(e.getMessage());
        setStatus("Insertion failed");
    }
}

private void insertDatabaseProduit(){
    try{
        stmt = conn.createStatement();
        int Enclume_idEnclume = Integer.parseInt(m1.getText());
        int Commande_idCommande = Integer.parseInt(m2.getText());
        int Qte = Integer.parseInt(m3.getText());
```

DatabaseUserInterface.java | test.java | DatabaseUserInterface.java

```
        setStatus("Inserting --(  " + idFacture + ",  " + Ristourne + ",  " + idSource + ")-- to the database");
        } catch(Exception e){
            System.err.println(e.getMessage());
            setStatus("Insertion failed");
        }
}

private void insertDatabaseEnclume(){
    try{
        stmt = conn.createStatement();
        int idEnclume = Integer.parseInt(m1.getText());
        String Photos = m2.getText();
        int Poids = Integer.parseInt(m3.getText());
        int Taille = Integer.parseInt(m4.getText());
        int Prix = Integer.parseInt(m5.getText());
        String sql = "INSERT INTO Enclume " +
                "VALUES( " + idEnclume + ", \"" + Photos + "\", " + Poids + ", " + Taille + ", " + Prix + ")";
        System.out.println(sql);
        stmt.executeUpdate(sql);
        setStatus("Inserting --( " + idEnclume + ", " + Photos + ", " + Poids + ", " + Taille + ", " + Prix + ")-- to the database");
    } catch(Exception e){
        System.err.println(e.getMessage());
        setStatus("Insertion failed");
    }
}

private void insertDatabaseProduit(){
    try{
        stmt = conn.createStatement();
        int Enclume_idEnclume = Integer.parseInt(m1.getText());
        int Commande_idCommande = Integer.parseInt(m2.getText());
        int Qte = Integer.parseInt(m3.getText());
        String sql = "INSERT INTO Produit " +
                "VALUES( " + Enclume_idEnclume + ", " + Commande_idCommande + ", " + Qte + ")";
        System.out.println(sql);
        stmt.executeUpdate(sql);
        setStatus("Inserting --( " + Enclume_idEnclume + ", " + Commande_idCommande + ", " + Qte + ")-- to the database");
    } catch(Exception e){
        System.err.println(e.getMessage());
        setStatus("Insertion failed");
    }
}
}
```