Blin Sébastien
Collin Pierre-Henri

ECOLE SUPERIEURE
D'INGENIEURS DE RENNES

École supérieure d'ingénieurs de Rennes

1ère Année
Parcours Informatique

BDD

Compte-Rendu TP n° 4\5\6

Sous l'encadrement de :

Zoltan Miklos

1 Analyse des besoins

1.1 *Fonctionnalité essentielle*

— Emploi du temps d'un étudiant
— Emploi du temps d'un enseignant
— Enseignant d'un cours
— Liste des étudiants qui suivent un cours spécifique
— Horaires d'un cours

1.2 *Fonctionnalité importante*

— Emploi du temps d'un parcours
— Salles occupées/non-occupées

1.3 *Fonctionnalité utile*

— Occupation moyenne des salles

2 Conception du schéma

Nous avons défini 5 entités (Enseignant, Cours, Salle, Etudiant et Parcours) et 3 associations (Enseigne, estDans et estInscrit) pour réaliser les fonctionnalités essentielles.

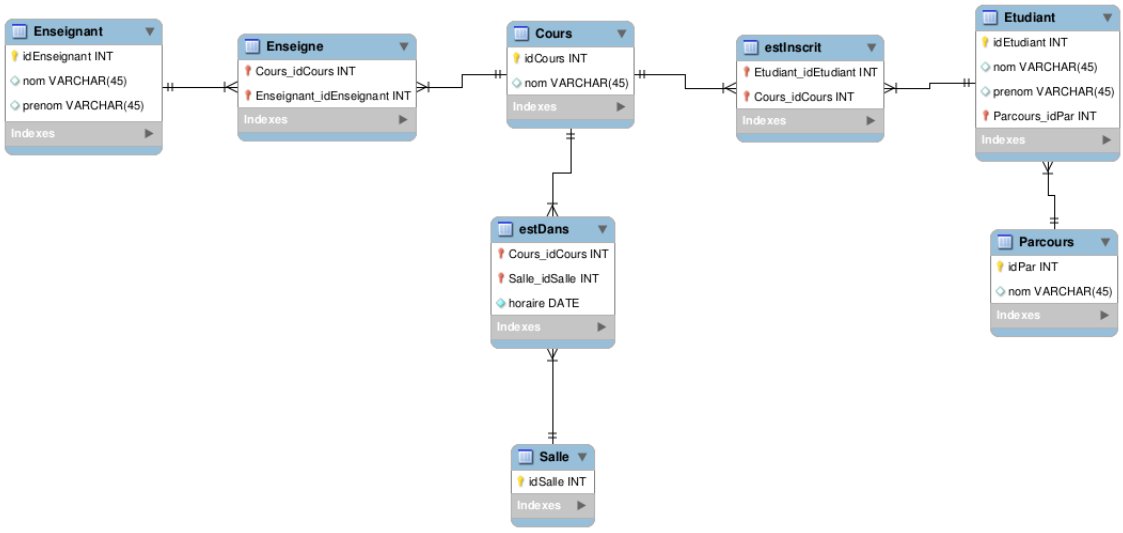

FIGURE 1 – Schéma réalisé à l'aide de *MySQL Workbench*

3 Quelques requêtes

Voici une liste non-exhaustive de requêtes SQL correspondant aux besoins énoncés ci-dessus :

— Lister l'emploi du temps (d'un créneau spécifique) pour un parcours.

```
SELECT DISTINCT horaire FROM estDans eD, Cours C, estInscrit eI, Etudiant E
WHERE eD.Cours_idCours=C.idCours
AND C.idCours=eI.idCours AND eI.idEtudiant=E.idEtudiant AND E.Parcours_idPar=2;
```

— Trouver le nom des enseignants pour un cours spécifique.

```
SELECT Enseignant_idEnseignant FROM Enseigne WHERE Cours_idCours = 1;
```

— Horaires d'un cours spécifique (par ex. la date du prochain cours de bases de données).

```
SELECT horaire FROM estDans WHERE Cours_idCours = 2;
```

— Obtenir une liste des étudiants (d'un parcours)

```
SELECT * FROM Etudiant WHERE Parcours_idPar = 2;
```

— Occupation d'une salle spécifique.

```
SELECT * FROM estDans WHERE Salle_idSalle = 20;
```

— Lister l'emploi du temps (d'un créneau spécifique) pour un étudiant particulier.

```
SELECT horaire FROM estDans
WHERE Cours_idCours =
(SELECT idCours FROM estInscrit WHERE idEtudiant = 1) AND horaire > '2010';

SELECT horaire FROM estDans
WHERE Cours_idCours =
(SELECT idCours FROM estInscrit WHERE idEtudiant = numEtudiant)
AND horaire > creneau_inf AND horaire < creneau_sup;
```

— Lister l'emploi du temps (d'un créneau spécifique) pour un enseignant.

```
SELECT horaire FROM estDans WHERE Cours_idCours =
(SELECT Cours_idCours FROM Enseigne WHERE Enseignant_idEnseignant = 1)
```

— Obtenir une liste des étudiants (d'un parcours) qui suivent un cours spécifique.

```
SELECT nom, prenom FROM estInscrit EI, Etudiant E
WHERE  EI.idEtudiant = E.idEtudiant AND E.Parcours_idPar = 1
AND EI.idCours = 1;
```

— Liste des salles (de TP/ou autre type) non-occupées (dans une période spécifique)

```
SELECT DISTINCT Salle_idSalle FROM estDans
WHERE Salle_idSalle !=
(SELECT DISTINCT Salle_idSalle FROM estDans
```

```
WHERE horaire > '2014' AND horaire < '2016');
```

— Déterminer l'occupation moyenne des salles de TP.

```
Soit X le nombre de creneaux sur une periode T. A raison de 8 creneaux par jour.
Pour une semaine, on a donc X = 40;
SELECT COUNT(horaire)/40 FROM estDans
WHERE horaire > '2015-05-04' AND horaire < '2015-05-11' AND Salle_idSalle = 1;
SELECT COUNT(horaire)/X FROM estDans
WHERE horaire > min AND horaire < sup AND Salle_idSalle = salle
```

4 Interface JDBC

4.1 Partie graphique

L'interface graphique que nous avons développé est composée de deux parties :

— Une rangée de boutons qui permettent de se connecter à la base de données, de se déconnecter, et d'envoyer des requêtes en fonction de ce que l'utilisateur souhaite obtenir comme information. Ainsi, il peut obtenir la liste de tous les enseignants, des étudiants, des horaires de cours, des horaires de parcours et des salles.

— Un ensemble de zones de texte qui permettent de filtrer les résultats et d'obtenir une recherche plus affinée. Par exemple, lorsque qu'on souhaite connaître les horaires d'un cours, on précise le numéro du cours dans le champ correspondant.

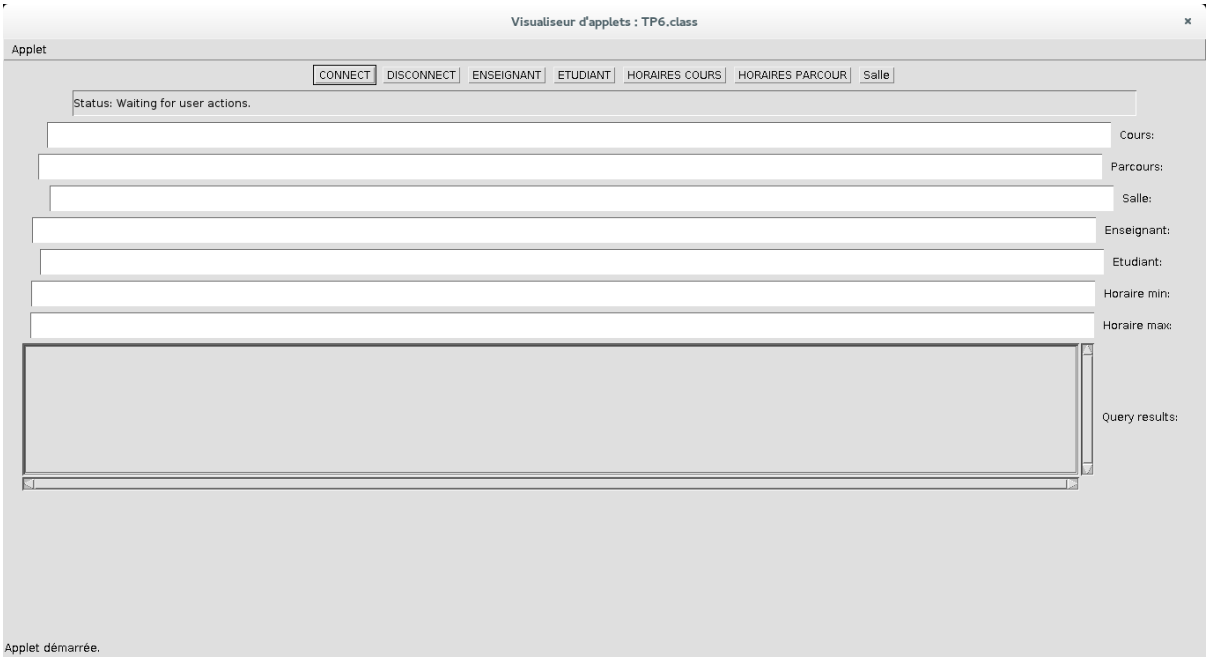

FIGURE 2 – Aperçu de l'applet

4.2 Code source

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;

/**
 * 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 actionPerforme
 * is called. Depending on his actions, one can implement the database connection (disconnection)
 * querying or insert.
 *
 * @author zmiklos
 *
 */
public class TP6 extends java.applet.Applet implements ActionListener {

 private TextField mStat, mCours, mParcours, mSalle,
 mEnseignant, mEtudiant, mHoraireMin, mHoraireMax;
 TextArea mRes;
 private Button b1, b2, bEnseignant, bEtudiant,
 bHoraireCours, bHoraireParcours, bSalle;
 private static final long serialVersionUID = 1L;

 // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://mysql.istic.univ-rennes1.fr:3306/base_13002155";

   //  Database credentials
   static final String USER = "user_13002155";
   static final String PASS = "lolgithub";

Connection conn = null;
Statement stmt = null;


 /**
  * This procedure is called when the Applet is initialized.
  *
  */
 public void init ()
 {
 /**
  * Definition of text fields
  */
     mStat = new TextField(150);
     mStat.setEditable(false);
```

```
mRes = new TextArea(10,150);
mRes.setEditable(false);
mCours = new TextField(150);
mParcours = new TextField(150);
mSalle = new TextField(150);
mEnseignant = new TextField(150);
mEtudiant= new TextField(150);
mHoraireMin = new TextField(150);
mHoraireMax = new TextField(150);



/**
 * 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");
bEnseignant = new Button("ENSEIGNANT");
bEtudiant = new Button("ETUDIANT");
bHoraireCours = new Button("HORAIRES COURS");
bHoraireParcours = new Button("HORAIRES PARCOUR");
bSalle = new Button("Salle");
add(b1) ;
add(b2) ;
add(bEnseignant) ;
add(bEtudiant) ;
add(bHoraireCours);
add(bHoraireParcours);
add(bSalle);
b1.addActionListener(this);
b2.addActionListener(this);
bEnseignant.addActionListener(this);
bEtudiant.addActionListener(this);
bHoraireCours.addActionListener(this);
bHoraireParcours.addActionListener(this);
bSalle.addActionListener(this);


add(mStat);
setLayout(new FlowLayout());
add(mCours);
add(new Label("Cours: ",Label.CENTER));
add(mParcours);
add(new Label("Parcours: ", Label.CENTER));
add(mSalle);
add(new Label("Salle: ", Label.CENTER));
add(mEnseignant);
```

```
    add(new Label("Enseignant: ", Label.CENTER));
    add(mEtudiant);
    add(new Label("Etudiant: ", Label.CENTER));
    add(mHoraireMin);
    add(new Label("Horaire min: ", Label.CENTER));
    add(mHoraireMax);
    add(new Label("Horaire max: ", Label.CENTER));
    add(mRes);
    add(new Label("Query results: ", Label.CENTER));


    setStatus("Waiting for user actions.");
}


/**
 * This procedure is called upon a user action.
 *
 *  @param event The user event.
 */
 public void actionPerformed(ActionEvent event)
{

// Extract the relevant information from the action
// (i.e. which button is pressed?)
Object cause = event.getSource();

// Act depending on the user action
// Button CONNECT
    if (cause == b1)
    {
       connectToDatabase();
    }

    // Button DISCONNECT
    if (cause == b2)
    {
     disconnectFromDatabase();
    }

    if (cause == bEnseignant)
    {
     QueryEnseignant();
    }

    if (cause == bEtudiant)
    {
     QueryEtudiant();
    }
```

```
     if (cause == bHoraireCours)
     {
      QueryHoraireCours();
     }

     if (cause == bHoraireParcours)
     {
      QueryHoraireParcours();
     }

     if (cause == bSalle)
     {
      QuerySalle();
     }
 }


/**
 * Set the status text.
 *
 * @param text The text to set.
 */
private void setStatus(String text){
    mStat.setText("Status: " + text);
  }

/**
 * Procedure, where the database connection should be implemented.
 */
private void connectToDatabase(){
try{
//STEP 2: Register JDBC driver
Class.forName("com.mysql.jdbc.Driver");

//STEP 3: Open a connection
System.out.println("Connecting to database...");
conn = DriverManager.getConnection(DB_URL,USER,PASS);
if(conn != null)
setStatus("Connected to the database");
} catch(Exception e){
System.err.println(e.getMessage());
setStatus("Connection failed" + e.toString());
}
}


/**
 * Procedure, where the database connection should be implemented.
```

```
 */
private void disconnectFromDatabase(){
try{
setStatus("Disconnected from the database");
} catch(Exception e){
        if(conn!=null)
try {
conn.close();
} catch (SQLException e1) {
e1.printStackTrace();
}
System.err.println(e.getMessage());
setStatus("Disconnection failed");
}
}

/**
 * Search Enseignant in Enseigne
 */
private void QueryEnseignant(){
try{
stmt = conn.createStatement();
    String sql;
    sql = "SELECT prenom, nom FROM Enseigne as E, Enseignant
    WHERE Enseignant.idEnseignant = E.Enseignant_idEnseignant";
    if(!mCours.getText().isEmpty())
     sql += " AND Cours_idCours = " + mCours.getText();
    System.out.println(sql);
    ResultSet rs = stmt.executeQuery(sql);
    mRes.setText("");
    //STEP 5: Extract data from result set
    while(rs.next()){
       //Retrieve by column name
       String prenom = rs.getString("prenom");
       String nom = rs.getString("nom");

       //Display values

       mRes.append("Enseignant: " + nom + " " + prenom + "\n");
    }

    stmt.close();
    rs.close();
} catch(Exception e){
System.err.println(e.getMessage());
setStatus("Requete ratee");
}

}
```

```
/**
 * Recherche un etudiant
 */
private void QueryEtudiant(){
try{
stmt = conn.createStatement();
    String sql;
    sql = "SELECT nom, prenom FROM estInscrit EI, Etudiant E
    WHERE  EI.idEtudiant = E.idEtudiant";
    if(!mParcours.getText().isEmpty())
     sql += " And E.Parcours_idPar = " + mParcours.getText();
    if(!mCours.getText().isEmpty())
     sql += " And EI.idCours = " + mCours.getText();
    System.out.println(sql);
    ResultSet rs = stmt.executeQuery(sql);
    //mRes.setText("");
    //STEP 5: Extract data from result set
    while(rs.next()){
       //Retrieve by column name
       String prenom = rs.getString("prenom");
       String nom = rs.getString("nom");

       //Display values

       mRes.append("Etudiant: " + nom + " " + prenom + "\n");
    }

    stmt.close();
    rs.close();
} catch(Exception e){
System.err.println(e.getMessage());
setStatus("Requete ratee");
}

}


/**
 * Horaire
 */
private void QueryHoraireCours(){
try{
stmt = conn.createStatement();
    String sql;
    sql = "SELECT * FROM estDans WHERE";
    System.out.println(sql);
    if(!mCours.getText().isEmpty())
     sql += " Cours_idCours= " + mCours.getText();
```

```
    else if(!mEnseignant.getText().isEmpty())
     sql += " Cours_idCours = (SELECT Cours_idCours
     FROM Enseigne
     WHERE Enseignant_idEnseignant = "+mEnseignant.getText()+")";
    else if(!mEtudiant.getText().isEmpty())
     sql += " Cours_idCours = (SELECT idCours
     FROM estInscrit
     WHERE idEtudiant= "+mEtudiant.getText()+")";
    else
     sql += " Salle_idSalle = " + mSalle.getText();

    if(!mHoraireMax.getText().isEmpty() && !mHoraireMin.getText().isEmpty())
     sql += "AND horaire > '"+mHoraireMin.getText()+"'
     AND horaire < '"+mHoraireMax.getText()+"'";
    System.out.println(sql);
    ResultSet rs = stmt.executeQuery(sql);
    mRes.setText("");
    //STEP 5: Extract data from result set
    while(rs.next()){
       //Retrieve by column name
   String cours = rs.getString("Cours_idCours");
   String salle = rs.getString("Salle_idSalle");
       String horaire = rs.getString("horaire");

       //Display values

       mRes.append("Horaire: " + cours + "-" + salle + "-" + horaire + "\n");
    }

    stmt.close();
    rs.close();
} catch(Exception e){
System.err.println(e.getMessage());
setStatus("Requete ratee");
}

}

/**
 * Horaire
 */
private void QueryHoraireParcours(){
try{
stmt = conn.createStatement();
    String sql;
    sql = "SELECT DISTINCT horaire FROM estDans eD, Cours C, estInscrit eI, Etudiant E
     WHERE eD.Cours_idCours=C.idCours AND C.idCours=eI.idCours
     AND eI.idEtudiant=E.idEtudiant AND E.Parcours_idPar=" + mParcours.getText();
    System.out.println(sql);
```

```
    ResultSet rs = stmt.executeQuery(sql);
    mRes.setText("");
    //STEP 5: Extract data from result set
    while(rs.next()){
       //Retrieve by column name
       String horaire = rs.getString("horaire");

       //Display values

       mRes.append("Horaire: " + horaire + "\n");
    }

    stmt.close();
    rs.close();
} catch(Exception e){
System.err.println(e.getMessage());
setStatus("Requete ratee");
}

}

/**
 * Salle
 */
private void QuerySalle(){
try{
stmt = conn.createStatement();
    String sql;
    sql = "SELECT DISTINCT Salle_idSalle FROM estDans ";
    if(!mHoraireMax.getText().isEmpty() && !mHoraireMin.getText().isEmpty())
     sql += "WHERE horaire > '"+mHoraireMin.getText()+"' AND horaire < '"
     + mHoraireMax.getText()+"'";
    System.out.println(sql);
    ResultSet rs = stmt.executeQuery(sql);
    mRes.setText("");
    //STEP 5: Extract data from result set
    while(rs.next()){
       //Retrieve by column name
       String Salle = rs.getString("Salle_idSalle");

       //Display values

       mRes.append("Salle: " + Salle + "\n");
    }

    stmt.close();
    rs.close();
} catch(Exception e){
System.err.println(e.getMessage());
```

```
setStatus("Requete ratee");
}

}


}
```