1. Objectif

Cet atelier vise à développer une application java centrée données – 2 tiers utilisant le SGBD MySQL et le pilote JDBC de MySQL. L'objectif de cet atelier est de :

- Créer une base de données sous MySQL
- Faire une connexion à la base de données depuis une application java
- Accès à la base de données (écrire des requêtes SQL) depuis une application java

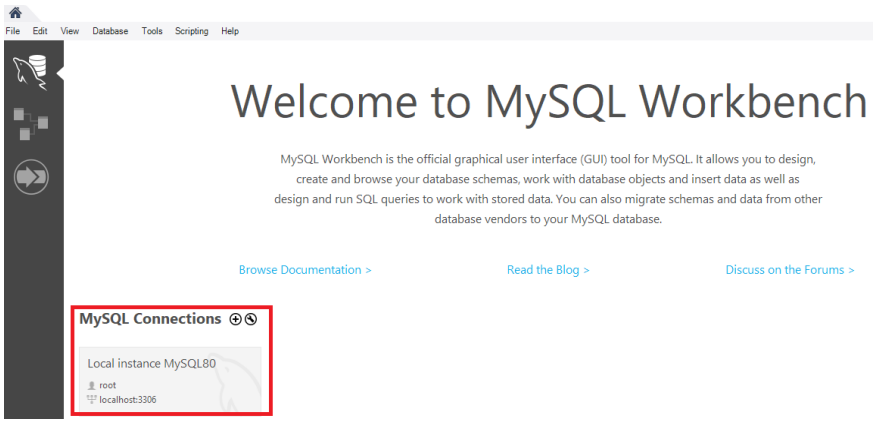
2. Prérequis

Vous avez besoin des outils suivants :

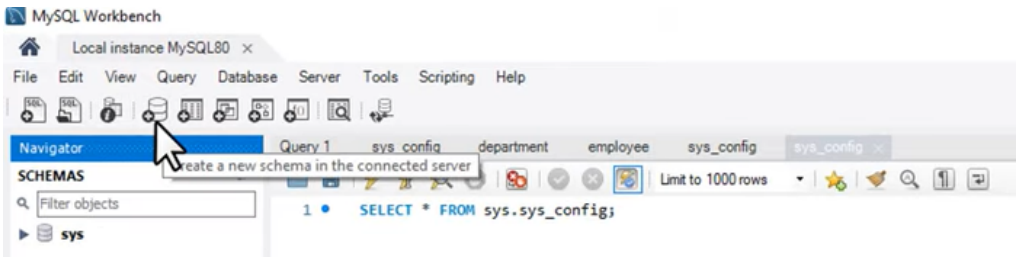
- **SGBD MySQL** : téléchargez *mysql-installer-web-community-8.0.36.0.msi* depuis ce lien https://dev.mysql.com/downloads/installer/. Une fois installé, cliquez deux fois là-dessus et suivre l'installation.
- **Le pilote JDBC** : téléchargez *mysql-connector-j-8.2.0.zip* depuis https://dev.mysql.com/downloads/connector/j/ (choisissez « Platform independent » pour le télécharger). Faites le unzipp du dossier et le garder dans votre espace de travail de java. Ce dossier contient un fichier jar qui correspond au pilote JDBC de MySQL.

3. Initiation avec MySQL Workbench

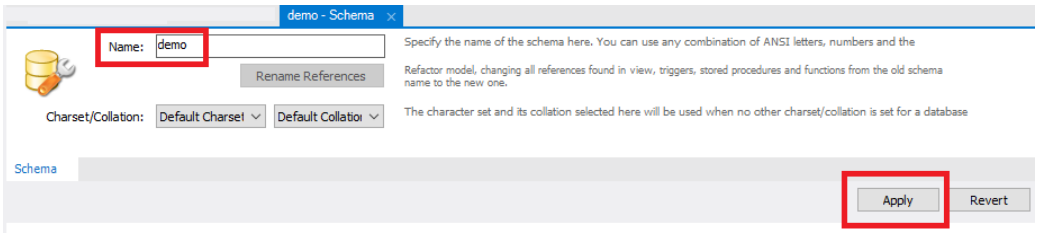
Dans cette section, nous allons créer la base de données « *demo* » sous MySQL contenant une seule table « *Employee* ». Pour cela, ouvrez votre MySQL workbench. Cliquez deux fois sur la partie encadrée en rouge pour accéder à votre espace MySQL.



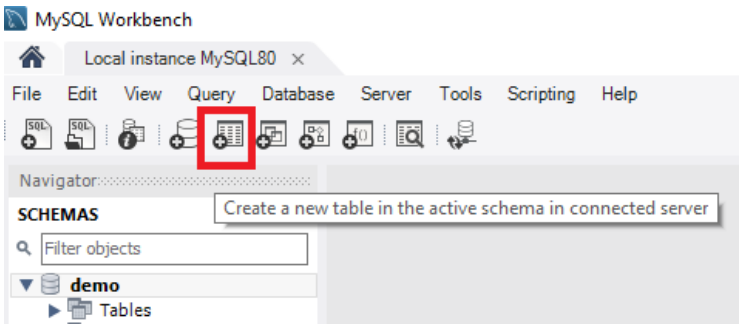
1. Pour créer une nouvelle base de données, cliquez-en haut sur le cylindre (voir figure ci-dessous) :



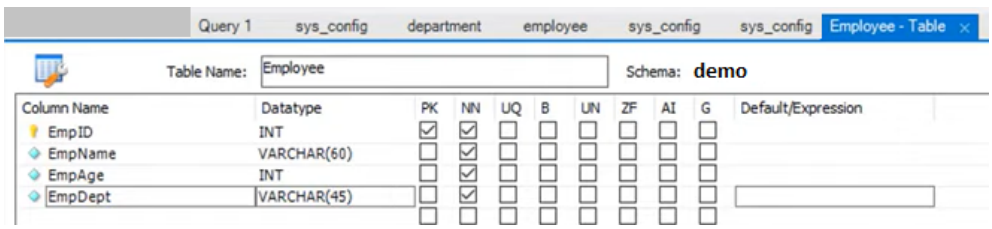
2. Tapez « demo » comme nom de base de données puis cliquez sur Apply.



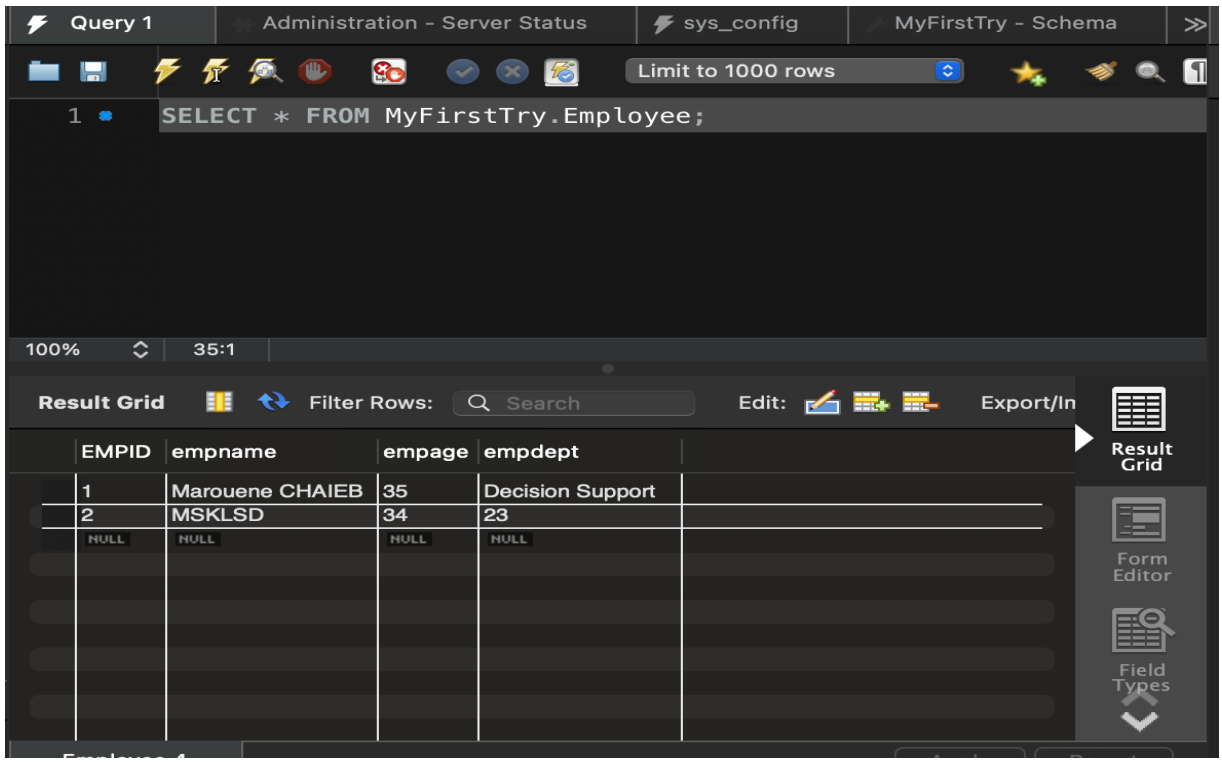
3. Une fois la base de données est créée, vous allez créer une nouvelle table « Employee ». Pour ce faire, cliquez sur l'icône tableau comme suit :



4. Créez les colonnes comme le montre la figure suivante puis cliquez sur Apply.



5. Rajoutez une ligne de données dans la table Employee :



Vous pouvez remplir votre table *employee* par d'autres données.

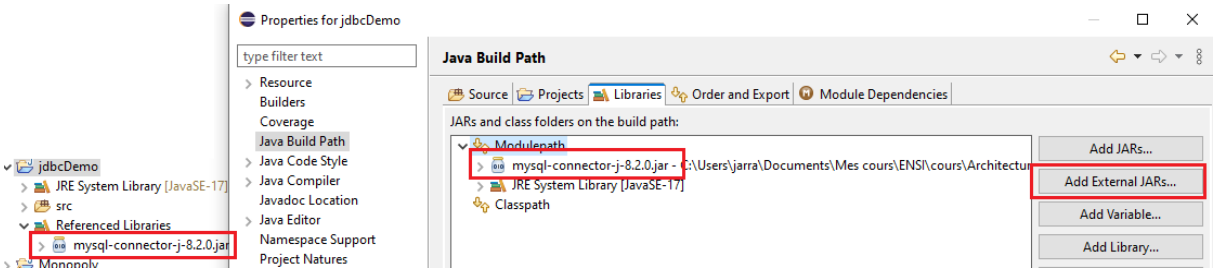
Votre première base de données *demo* est bien créée.

4. Se connecter à la base de données MySQL avec JDBC

Dans cette section, nous allons créer un projet java pour se connecter et manipuler la base de données *demo* en utilisant le pilote JDBC.

En effet, JDBC est l'acronyme de *Java DataBase Connectivity* et désigne une API pour permettre un accès aux bases de données avec Java.

1. Créez un nouveau projet java. Cliquez droit sur le nom de votre projet->properties-> onglet Libraries -> add External JARs. Parcourez et choisissez votre connecteur MySQL JDBC :

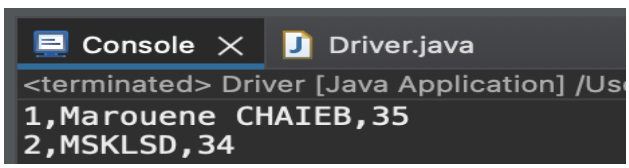


2. Créez une classe Driver dans le package jdbcDemo. Cette classe contiendra la méthode main :

```
package jdbcDemo;

import java.sql.Connection; \\ pour établir une connexion à une base de données.
import java.sql.DriverManager; \\ permet d'enregistrer et de gérer les pilotes de base de données.
import java.sql.ResultSet; \\ représente un ensemble de résultats de base de données après
l'exécution d'une requête SQL.
import java.sql.Statement; \\ utilisée pour exécuter des instructions SQL et retourner les résultats

public class Driver {

	public static void main(String[] args) {

		try {	// Début du bloc try, où les exceptions sont susceptibles d'être levées.

			//1. Get a connection to database : établit une connexion à la base de
données MySQL en utilisant JDBC. La méthode getConnection de la classe DriverManager prend l'URL de
connexion, le nom d'utilisateur et le mot de passe comme arguments.

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

			//2. Create a statement: Cette ligne crée un objet Statement à partir de la
connexion établie. Cet objet est utilisé pour envoyer des requêtes SQL à la base de données.


			Statement myState = myConn.createStatement();


			//3. Execute SQL Query : Cette ligne exécute une requête SQL. La méthode
executeQuery de l'objet Statement retourne un objet ResultSet contenant les résultats de la requête.


			ResultSet rs = myState.executeQuery("select * from employee");


			//4. Process the result set : Début d'une boucle while qui parcourt toutes
les lignes du ResultSet.

			while (rs.next()) {

				System.out.println(rs.getInt("id") + "," +
rs.getString("first_name")+","+rs.getString("last_name"));
			}


			//5. close connection: myConn.close(); : Cette ligne ferme la connexion à la
base de données après avoir terminé le traitement.


			myConn.close();

		}
\\ 6. Si une exception est levée dans le bloc try, elle est capturée ici et affichée dans la console. Cela
permet de détecter et de diagnostiquer les problèmes éventuels lors de l'exécution du programme.
		catch (Exception e) {
			e.printStackTrace();
		}
	}
}
```

```
package jdbcDemo; import java.sql.Connection; import java.sql.DriverManager; import java.sql.ResultSet;
import java.sql.Statement;
public class Driver {
	public static void main(String[] args) {
		try {
			//1. Get a connection to database
			Connection myConn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/demo", "root", "root");

			//2. Create a statement
			Statement myState = myConn.createStatement();

			//3. Execute SQL Query
			ResultSet rs = myState.executeQuery("select * from employee");

			//4. Process the result set
			while (rs.next()) {
				System.out.println(rs.getInt("id") + "," +
rs.getString("first_name")+","+rs.getString("last_name"));
			}

			//5. close connection
			myConn.close();
		}
		catch (Exception e) {
			e.printStackTrace();
		}
	}
}
```

3. Lancez l'exécution de votre projet. La sortie sera l'affichage de toutes les données contenant dans la table « employee ».

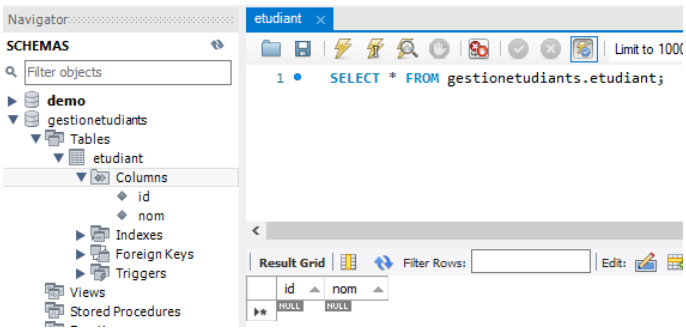


5. Exercice d'application – Gestion des étudiants

a. Création de la base de données

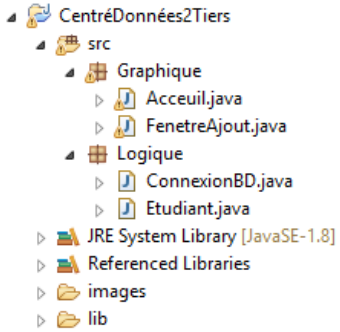
Sous MySql, créez la base de données « gestionetudiants » contenant une seule table « etudiant ». Cette table contient deux colonnes :

- id : int
- nom : varchar (45)



b. Création de l'application java

Vous allez créer une application java pour la gestion des étudiants.



La structure de votre projet se divise en deux packages :

- Package **Logique** : il comporte deux classes java :
  - La classe Etudiant.java
    Attributs privés : id(int) et nom (String)
    Constructeur pour initialiser les deux attributs
    Les getters/ les setters
  - La classe ConnexionBd.java
    Attributs privés : connexion(Connection), instruction(Statement), resultat(ResultSet)
    Constructeur ConnexionBD() : établir une connexion à la base de données
    Méthode arret() : ferme la connexion BD.
    Méthode getStudents() : affiche la liste des étudiants dans la table **Etudiant** de la base de données **GestionEtudiants**
    Méthode insertStudent() : permet d'insérer un nouvel étudiant dans la table **Etudiant** de la base de données **GestionEtudiants**

- Package **Graphique :** il contient deux interfaces graphiques :

- L'interface *Acceuil.java* : représente l'interface principale de votre application

La classe Acceuil utilise l'api **JFrame** pour développer l'interface d'acceuil.

```
package Graphique;

import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;

public class Acceuil extends JFrame{


public JLabel lab,id,nom;

        public JTextField idT,nomT;

        public JButton ajout = new JButton("Ajouter un étudiant");
        public JButton supp = new JButton("Supprimer un étudiant");

        public Acceuil()
        {
                this.lab = new JLabel("Gestion des étudiants");
                setTitle("Acceuil");
                setLayout(new FlowLayout(1,50,50));
                this.getContentPane().setBackground(Color.magenta);
                setSize(800,350);
                setResizable(true);
                setVisible(true);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

                this.add(lab);
                this.add(ajout);
                this.add(supp);

                this.ajout.addActionListener(new ActionListener() {

                        @Override
                        public void actionPerformed(ActionEvent arg0) {
                                // TODO Auto-generated method stub
                                FenetreAjout fenajout = new FenetreAjout();

                        }});

        }
        public static void main(String[] args) {
                // TODO Auto-generated method stub
                //Acceuil ihm = new Acceuil();
```

```
                Acceuil ihm = new Acceuil();
        }
}
```

- L'interface *FenetreAjout.java* : représente l'interface pour ajouter un nouvel étudiant (elle s'affiche en cliquant sur le bouton « ajouter un étudiant » de l'interface Acceuil)

La classe **FenetreAjout** présente l'implémentation de l'interface graphique permettant d'ajouter un nouvel étudiant.

```
package Graphique;

import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;
import Logique.ConnexionBD;
import Logique.Etudiant;

public class FenetreAjout extends JFrame{


                public JLabel id,nom;

                        public JTextField idT,nomT;

                        public JButton ok = new JButton("ok");
                        public JButton cancel = new JButton("cancel");

                        public FenetreAjout()
                        {
                                this.id = new JLabel("id");
                                this.nom = new JLabel("nom");
                                this.idT = new JTextField(20);
                                this.nomT = new JTextField(20);
                                setTitle("Ajout d'un étudiant");
                                setLayout(new FlowLayout(1,50,50));
                                setSize(700,400);
                                setResizable(true);
                                setVisible(true);
                                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

                                this.add(id);
                                this.add(idT);
                                this.add(nom);
                                this.add(nomT);
                                this.add(ok);
                                this.add(cancel);
```

```
                                this.ok.addActionListener(new ActionListener() {

                                        @Override
                                        public void actionPerformed(ActionEvent arg0) {
                                                // TODO Auto-generated method stub
                                                Etudiant e = new Etudiant
(Integer.parseInt(idT.getText()), nomT.getText());
                                                ConnexionBD gg = new ConnexionBD();
                                                gg.insertStudent("insert into Etudiant values
('"+e.getId()+"','"+ e.getNom()+"')");
                                                JOptionPane.showMessageDialog(null, "Ajout avec
succès!!", "Information",
                                                        JOptionPane.INFORMATION_MESSAGE) ;

                                        }});

                                this.cancel.addActionListener(new ActionListener() {

                                        @Override
                                        public void actionPerformed(ActionEvent arg0) {
                                                // TODO Auto-generated method stub
                                                setVisible(false);

                                        }});
                        }

}
```

Ce que vous allez faire :

1. Développez les deux classes de la logique métier : classe Etudiant.java et ConnexionBD.java.
2. Testez votre application et vérifiez si l'ajout d'un nouvel étudiant via l'interface graphique se fait correctement.
3. Développez une 3ème interface graphique pour supprimer un étudiant en donnant son id.
4. L'application gestion des étudiants est une application qui utilise le style architectural logique centrée donnée. Nous souhaitons utiliser le style architectural logique MVC. Quelles sont les modifications à apporter ? (Il est recommandé de créer un nouveau projet java MVC)