TD sur JDBC

I. Accès à une BD via JDBC

Le programme suivant utilise l'API java JDBC pour accéder à une base de données PostgreSQL distante.

1. Soit une table nommée "personnes" qui comporte 4 colonnes "nom" (chaîne), "prenom" (chaîne), "age" (entier) et "compte" (flottant). Compléter le code en /* 1. */ pour afficher le contenu complet de cette table sur la console de la façon suivante :

nom	prenom	age	compte
Dupont	Jean	30	3510.0
Martin	Pierre	45	16510.5
Durand	Jacques	53	832.0

…

2. On veut afficher la somme de tous les comptes et le nombre de ceux qui sont renseignés (non null) en partant de la requête SQL déjà exécutée en /* 1. */ Compléter le code en /* 2. */

Remarque : on aurait pu faire directement "SELECT SUM(compte), COUNT(*) from personnes WHERE compte IS NOT NULL".

3. On veut afficher le contenu complet d'une une table nommée "divers" de la même façon qu'en 1. sauf qu'on ne connait rien de cette table (ni le nom des colonnes ni même leur nombre). Compléter le code en /* 3. */

4. Revenons à la table "personnes". Soit la requête de mise à jour (update) écrite en /* 4 */. On veut exécuter cette requête mais en l'annulant ensuite si le nombre de lignes mises à jour est supérieur à 1. Compléter le code en /* 4. */

5. On veut permuter les comptes en banque des personnes "toto" et "titi" de manière transactionelle (si une des deux personnes n'existe pas ou qu'un des deux comptes n'est pas renseigné (null), la transaction est annulée). Compléter le code en /* 5. */

6. Modifier le /* 5. */ pour regrouper/remplacer les requêtes SELECT et UPDATE (qui sont chacunes appelées 2 fois) en requêtes paramétrées (puis initialiser ces paramètres).

7. Avant les mises à jours, on s'est bien assuré que les personnes "toto" et "titi" existaient bien et que leurs comptes n'étaient pas null. Mais on est jamais trop prudent et on veut s'assurer que les deux mises à jour ont bien été effectuées (sinon, encore une fois, on annule la transaction). Que faut-il modifier le code en /* 5. */ ?

```
import java.sql.*;

public class Test{

   public static void main(String[] args) throws SQLException, ClassNotFoundException {
      try {

         String DBurl ="";

         /* Exemple avec SGBD MySql local
         Class.forName("com.mysql.jdbc.Driver");
         DBurl="jdbc:mysql://localhost:3306/BDTEST";
         */

         /* Exemple avec SGBD PostgreSQL local
         Class.forName("org.postgresql.Driver");
```

```
DBurl="jdbc:postgresql://localhost:3306/BDTEST";
*/

/* Exemple avec Microsoft Access en local */
String path="C:/MaBase.mdb";
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
DBurl="jdbc:odbc:DRIVER={Microsoft Access Driver (*.mdb)};DBQ="+path;

/* Connexion avec un login et un password factices
String user = "user";
String password = "pass";
Connection con = DriverManager.getConnection(DBurl, user, password); */

Connection con = DriverManager.getConnection(DBurl);

Statement stmt = con.createStatement(); // pour exécution d'une requête SQL
String requete=""; // requête SQL
ResultSet rset = null; // pour contenir le résultat d'une requête SQL de type SELECT

ResultSetMetaData rsmd = null; // pour les métadonnées d'un ResultSet (question 3)
String ligneAffichage=""; // ligne d'un ResultSet (formatée de telle ou telle façon) pour affichage console (question 3)
int nbColonnes=-1; // pour le nombre de colonnes d'un ResultSet (question 3)

int nbRes=-1; // pour le nombre de mises à jour de la requête SQL de type UPDATE de la question 4

ResultSet rset2 = null; // pour besoin d'un deuxième ResultSet (questions 5 et 6)
float compteToto=-1.0f; // pour la valeur du compte de toto (questions 5 et 6)
float compteTiti=-1.0f; // pour la valeur du compte de titi (questions 5 et 6)
PreparedStatement pstmt=null; // pour exécution requête SQL paramétrée (question 6)

int nbMaj=-1; // pour le nombre de mises à jour des deux requêtes SQL de type UPDATE de la question 7


/* 1. Question 1. */

/* 1.1. Requête SQL */
……..

/* 1.2. Exécution de la requête */
……..

System.out.println("nom  \t"+"prénom  \t "+"age  \t" +"compte" );

/* 1.3. Parcours et affichage de chaque ligne */
……..
……..


/* 2. Question 2. */

/* 2.1. Variables pour contenir la somme de tous comptes et le nombre de comptes non null */
float sommeComptes=0.0f;
int nbComptesNotNull=0 ;

/* 2.2. On repositionne le curseur sur la première ligne de résultat */
……..

/* 2.3. Parcours de chaque ligne en mettant à jour les deux variables précédentes */
……..
……..
……..
……..
……..

/* 2.4. Affichage du résultat */
System.out.println("Somme des comptes: "+sommeComptes+" Nombre de comptes valides: "+nbComptesNotNull);


/* 3. Question 3. */
```

```
/* 3.1. Requête SQL et exécution de celle-ci */
……
……

/* 3.2. Récupérer le nombre de colonnes du résultat */
……
……

/* 3.3. Afficher le nom des colonnes sur une ligne */
……
……
……

/* 3.4. Parcours et affichage de chaque ligne */
……
……
……
……
……
……

/* 4. Question 4. */

/* 4.1 Requête de mise à jour */
requete= "update personnes set age=age+1 where age >100";

/* 4.2. Exécution de la requête et annulation de celle-ci si le nombre de personnes concernés par la mise à jour est supérieur à 1*/
……
……
……
……
……
……


/* 5. Question 5. */

con.setAutoCommit(false);

/* 5.1. Requêtes pour récupérer le compte de "toto" et le compte de "titi" ; remarque : SELECT FOR UPDATE : verrouillage "Row Share Table Locks (RS)" sur une une BD Oracle */

requete="select compte from personnes where nom = 'toto' for update of personnes";
rset = stmt.executeQuery(requete);
requete="select compte from personnes where nom = 'titi' for update of personnes";
rset2 = stmt.executeQuery(requete);

/* 5.2. Si les personnes "toto" ou "titi" n'existent pas, on annule la transaction  */
……
……

else {

   /* 5.3. Récupération des comptes ; si l'un d'eux n'est pas renseigné (null), on annule la transaction  */
   ……
   ……
   ……
   ……

   else{

      /* 5.4. Ecrire, exécuter et valider les deux requêtes de mise à jour */
      ……
      ……
      ……
      ……
      ……

      /* Remarque : une autre manière de faire aurait été de mettre à jour les lignes directement dans le ResultSet
```

```
                => ligne correspondant à la personne "toto"
                rset.updateFloat("compte", compteTiti);
                rset.updateRow();
                => ligne correspondant à la personne "titi"
                rset2.updateFloat("compte", compteToto);
                rset2.updateRow();
                con.commit(); */

            }
          }

      }
      catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```