TP4 : Connexion à une base de données distante

De nombreux programmes utilisent des données situées sur base de données séparée du programme lui-même. Le programme qui effectue les requêtes est appelé le “client”, le programme qui répond aux requêtes est le “serveur”. Le serveur est en général situé sur une machine distante, accessible par son adresse IP. Ici, pour des raisons de simplicité, nous travaillerons sur une base locale gérée par le programme sqlite.

Le but de ce TP est d’écrire un programme client (en Java) qui se connecte à une base de données et affiche le résultat de quelques requêtes.

1. Lecture de la base

Nous travaillerons sur la base `foodmart.db` contenant la description d’environ 1200 employés d’un magasin de grande distribution.

Les employés sont décrits par les attributs :

Employee (employee_id, full_name, first_name, position_id, position_title, store_id, department_id, birth_date, hire_date, end_date, salary, supervisor_id, education_level, marital_status, gender, management_role)[1)]

Téléchargez foodmart.db, placez-vous dans le bon répertoire et lancez:

```
sqlite3 foodmart.db
```

La base contient 2 tables : `employee` et `department`.

tapez par exemple :

```
  .schema employee
```

Vous voyez apparaître les commandes sql qui définissent la structure de la table. tapez ensuite :

```
  SELECT * FROM employee;
```

la liste de employés s’affiche.

Testez les requêtes suivantes dans la base :

Par exemple :

- donner le nombre total d’employés
- donner le nombre d’employés et le salaire moyen par magasin
- donner les salaires min et max et moyen par catégorie managériale
- donner la liste des magasins employant plus de 30 employés
- donner le nom complet, la fonction et la catégorie managériale des employés supervisant plus de 5 employés

- donner le nom complet, la fonction et la catégorie managériale des 10 employés ayant les plus hauts salaires

etc…

2. Création d’un nouveau projet Java

Le TP sera réalisé en Java. Lancez le programme Eclipse.

Nous allons apprendre aujourd’hui à utiliser la librairie JDBC. Il s’agit d’un ensemble de fonctionnalités permettant d’envoyer des requêtes vers un serveur de bases de données.

Pour créer le nouveau projet, nous allons utiliser l’outil d’importation d’Eclipse :

- récupérez le fichier S7-TP4-2016.tgz
- allez dans File → Import → General → Existing Projects into Workspace
- sélectionnez “select archive file” et cliquez sur “Browse”
- sélectionnez le fichier S7-TP4-2016.tgz

Un nouveau projet S7 TP4 doit apparaître dans votre espace de travail contenant le programme principal suivant:

```
import java.sql.*;
public class S7_TP4 {
    public static void main(String[] args) throws ClassNotFoundException,
SQLException{
        Class.forName("org.sqlite.JDBC");
        Connection c = null;
        try{
            c = DriverManager.getConnection("jdbc:sqlite:foodmart.db");
        }
        catch(Exception e){
            System.out.println("problème d'accès au fichier foodmart.db : "
+ e.getMessage());
        }
        Statement m = c.createStatement();
        ResultSet r = null;
    }
}
```

Explications :

La classe principale gère un objet de type “Connection” permettant d’interagir avec le serveur.

La création du “connecteur” c nécessite 2 opérations:

- création d’un gestionnaire de pilote (“driver manager”) à l’aide de la commande :

```
Class.forName("org.sqlite.JDBC");
```

- définition d'un objet de type Connection

```
Connection c = DriverManager.getConnection("jdbc:sqlite:foodmart.db");
```

Pour communiquer avec le serveur de bases de données, nous avons créé un "messager" qui transmet la requête au serveur, et un "récepteur" qui récupère le résultat de la requête :

- création du messager :

```
Statement m = c.createStatement();
```

- création du récepteur :

```
ResultSet r;
```

Lancez le programme et vérifiez qu'il n'y a pas d'erreur.

3. Premières requêtes

Le but de ce TP est d'écrire un programme capable de récupérer des informations de la table "employee", afin de les stocker dans des objets de type "Employe", et d'afficher certaines informations concernant un ou plusieurs employés.

Recopiez le code suivant:

```
r = m.executeQuery("SELECT * FROM employee");
int num_employe;
String nom;
while (r.next()) {
        num_employe = r.getInt("employee_id");
        nom = r.getString("full_name");
        System.out.println(num_employe + " : " + nom);
    }
```

Explications :

L'objet `r` est un ensemble de tuples correspondant au résultat de la requête (il y a souvent plusieurs réponses à une requête particulière). L'objet `r` a une structure de flux : chaque appel à `r.next()` donne accès au tuple suivant.

Pour parcourir toutes les réponses, il faut parcourir les éléments de `r` du premier au dernier, en utilisant une boucle comme suit:

```
while (r.next()) {
    …
    }
```

Avant de lancer une requête, nous devons connaître les principaux types de données contenues dans la table des employés. Si nous regardons la définition de la table dans sqlite3, nous voyons des types entiers (`INTEGER`), chaîne de caractères (`VARCHAR`), date (`DATE`), date et heure (`DATETIME`), et enfin nombre décimal (`DECIMAL`) définis selon la syntaxe SQL. Les types correspondant en Java seront les types `int`, `String`, `Date`, `Time`, `double`, … Lorsque le programme récupérera des valeurs résultant d'une requête, les variables devront être correctement typées.

A chaque appel à `r.next()`, il est possible de récupérer les données du tuple à l'aide d'une des fonctions suivantes :

- `getInt()` pour récupérer une valeur d'attribut de type entier
- `getString()` pour récupérer une valeur d'attribut de type chaîne de caractère
- `getDouble()` pour récupérer une valeur décimale
- etc…

Testez cette requête ainsi que quelques autres :

3.1 - Comptez et affichez le nom, le prénom et le salaire des employés qui gagnent plus de 10000 dollars.

3.2 - Comptez et affichez le nom complet (full_name), la fonction (position_title), la date de naissance (birth_date) et le salaire (salary) de tous les employés supervisés par Mona Jaramillo.

4. Classe Employe

Définir une classe `Employe` contenant les informations suivantes :

- nom complet
- fonction
- date de naissance
- salaire

Définir un constructeur (prenant en paramètres 2 chaînes de caractères, une date et un décimal)

Définir une méthode `toString` permettant de d'afficher les informations concernant l'employé

4.1 - Au niveau du programme principal, reprenez la question 3.2 en stockant les résultats de la requête dans un objet de type Employe et en affichant cet objet.

4.2 - Vous définirez également un tableau d'employés contenant la liste des employés supervisés par Mona Jaramillo (en utilisant par exemple une ArrayList : `import java.util.ArrayList;`) (voir par exemple http://www.siteduzero.com/tutoriel-3-10409-les-collections-d-objets.html#ss_part_2)

4.3 - Définir une méthode `mieux_paye_que()` qui compare le salaire de deux employés (retourne `true` ou `false`) Affichez le nom et le prénom du mieux payé des employés de Mona Jaramillo.

5. Classe Departement

Définir une classe `Departement` contenant les informations suivantes :

- `numDep` (numéro du département)
- `description` (description du département)
- `listeEmployes` (liste des employés appartenant à ce département)
- définir un constructeur (prenant en paramètres un entier et une chaînes de caractères)
- définir une méthode `ajouteEmploye()` qui ajoute un employé à la liste des employés.

5.1 - Au niveau du programme principal, créez puis remplissez une liste de départements à partir de la table department.

5.2 - Définir une méthode `coutDep()` calculant le coût salarial d'un département (la somme des salaires de ses personnels). Afficher le coût de chacun des départements.

5.3 - Définir une méthode `plusCouteuxQue()` comparant le coût de deux départements. Afficher la description et le coût du département le moins coûteux et du département le plus coûteux.

[1]
autrement dit en français : **Employé**(id_employé, nom_complet, prénom, id_fonction, titre_fonction, id_magasin, id_département, date_naiss, date_embauche, date_fin, salaire, id_supérieur, niveau_educatif, statut_marital, sexe, rôle_managérial)