TP2 BD

JDBC / Transactions

Partie 1 : JDBC

Les différentes fonctionnalités ont été mise en oeuvre de la manière suivante:

On commence par définir les différentes variables nécessaires à la connexion à la base:

```
public class Banque {

    static final String CONN_URL = "jdbc:oracle:thin:@im2ag-oracle.e.ujf-grenoble.fr:1521:ufrima";
    static final String USER = "lerchunt";
    static final String PASSWD = "bd2015";
```

Dans le main du programme, on lance ensuite établit la connexion (et on désactive l'auto-commit) :

```
public static void main(String args[]) {

    try {
    int action;
    boolean exit = false;

    // Enregistrement du driver Oracle
    System.out.print("Loading Oracle driver... ");
    Driver myDriver = new oracle.jdbc.driver.OracleDriver();
    DriverManager.registerDriver( myDriver );
    System.out.println("loaded");

    // Etablissement de la connection
    System.out.print("Connecting to the database... ");
    Connection  conn = DriverManager.getConnection(CONN_URL , USER , PASSWD);
    System.out.println("connected");

    // Desactivation de l'autocommit

    conn.setAutoCommit(false);
    System.out.println("Autocommit disabled");
```

Pour les 3 méthodes implémentées, on passe en paramètre la connexion. Voici le code de la méthode qui affiche les données de la table 'LesAnimaux' :

```
private static void listeAnimaux(Connection conn) throws SQLException {
    String nom,sexe,type_an,fonction_cage,pays,annais,nocage,nb_maladies;
    System.out.println("Affichage des animaux : (nom, sexe, type_an, fonction_cage, pays, annee, NO_cage, nb_maladies)");
    Statement  stmt = conn.createStatement ();
    ResultSet  rs = stmt.executeQuery("SELECT* from LesAnimaux");
    while (rs.next()){
        nom=rs.getString("NomA");
        sexe=rs.getString("SEXE");
        type_an=rs.getString("TYPE_AN");
        fonction_cage=rs.getString("FONCTION_CAGE");
        pays=rs.getString("PAYS");
        annais=rs.getString("ANNAIS");
        nocage=rs.getString("NOCAGE");
        nb_maladies=rs.getString("NB_MALADIES");
        System.out.println(nom+" "+sexe+" "+type_an+" " +fonction_cage+" "+pays+" "+annais+" "+nocage+" "+nb_maladies);
    }
}
```

C'est une requête simple de qui affiche à chaque ligne le résultat.

La seconde méthode consiste à déplacer un animal de cage.
Il s'agit donc de changer le Numero de cage de l'animal à partir de son nom. On demande d'abord à l'utilisateur le nom de l'animal concerné. Si et seulement si on trouve l'animal dans la table, on demande à l'utilisateur le nouveau numéro de cage de cet animal. puis on execute la requête de mise à jour.

```
private static void deplacerAnimal(Connection conn) throws SQLException {
    System.out.println("Quel est le nom de l'animal que vous souhaitez déplacer?");
    String action=LectureClavier.lireChaine();
    Statement  stmt = conn.createStatement ();
    String req =" SELECT * from LesAnimaux where NomA like '" + action + "'";
    ResultSet  rs = stmt.executeQuery(req);
    if(!rs.next())
        System.out.println("Animal non trouve");
    else
    {
        System.out.println("Animal trouve");
        int cage=LectureClavier.lireEntier("Dans quelle cage le placer ?");
        req= " Update LesAnimaux Set noCage =" + cage + "Where NomA like '" +action+"'";
        ResultSet  rs2 = stmt.executeQuery(req);
    }
}
```

Enfin , la dernière requête inséré une nouvelle maladie pour un animal. On demande ainsi à l'utilisateur le nom de l'animal, puis le nom de la maladie, avant de lancer la requête d'insertion.

```
private static void ajouterMaladie(Connection conn) throws SQLException {
    System.out.println("Quel est le nom de l'animal concerné?");
    String animal=LectureClavier.lireChaine();
    System.out.println("Quel est le nom de la maladie?");
    String maladie=LectureClavier.lireChaine();
    Statement  stmt = conn.createStatement ();
    String req ="INSERT into LesMaladies Values("+ "'" + animal +"'"+ "," + "'" + maladie +"' )";
    ResultSet  rs = stmt.executeQuery(req);
}
```

Les fonctions commit et rollback sont elles aussi implémentées.

```
private static void commit(Connection conn) throws SQLException {
    Statement  stmt = conn.createStatement ();
    String req = "commit";
    ResultSet  rs = stmt.executeQuery(req);
}

private static void rollback(Connection conn) throws SQLException {
    Statement  stmt = conn.createStatement ();
    String req = "rollback";
    ResultSet  rs = stmt.executeQuery(req);
}
```

On prend évidemment soin de fermer la connection à la fin du main, avec conn.close().

Partie 2 : Trigger

3 triggers ont été ajouté au script initial.

Le premier consiste à mettre à jour le nombre de maladies à chaque insertion ou suppression dans la table LesMaladies

```
create or replace trigger Q2_1
    after insert or delete
    on LesMaladies
    for each row
begin
    IF inserting then update LesAnimaux set NbMaladie= NbMaladie+1 Where NomA = :old.nomA;
    ELSIF deleting then update LesAnimaux set NbMaladie= NbMaladie -1 Where NomA= :old.nomA;
    end if;

end;
/
```

Pour chaque ligne, si on insère dans la table, alors on incrémente NbMaladie dans LesAnimaux, et on décrémente la valeur pour chaque suppression.

Le second trigger vérifie que pour chaque changement de cage d'un animal, la fonction de la cage est compatible avec cet animal

```
create or replace trigger Q2_2
before insert or update of noCage on LesAnimaux
for each row
Declare
fct varchar(20);
begin
    select fonction into fct from LesCages where nocage = :new.nocage;
    if fct != :new.fonction_cage then raise_application_error(-20001, 'cage incompatible');
    end if;
end;
/
```

Dans le cas où la nouvelle fonction n'est pas compatible, on retourne une erreur

Enfin, le dernier trigger vérifie qu'une cage est gardé par au moins un gardien. J'utilise donc une requête de type Sélect count(*) afin de trouver le nombre de gardien pour une cage donnée. Si ce nombre est à 0, on retourne une erreur.

```
create or replace trigger Q2_3
before insert or update  of noCage on LesAnimaux
for each row
declare
nb_grd number(3);
begin
    select count(*) noCage into nb_grd from LesGardiens where noCage = :new.nocage;
    if(nb_grd =0) then raise_application_error(-20002, 'Il n y a pas de gardien a cette cage');
    end if;
end;
/
```