Base de Données : Compte-rendu du TP2
Application "Zoo"

Line POUVARET, Mickaël TURNEL

2015-2016

5 - Transactions & JDBC

Etablissement de la connexion

```
1  /* Enregistrement du driver Oracle */
2  System.out.print("Loading␣Oracle␣driver...␣");
3  DriverManager.registerDriver(new oracle.jdbc.OracleDriver());
4  try {
5          Class.forName("oracle.jdbc.OracleDriver").newInstance();
6  } catch (InstantiationException ex) {
7          Logger.getLogger(Banque.class.getName()).log(Level.SEVERE, null, ex);
8  } catch (IllegalAccessException ex) {
9          Logger.getLogger(Banque.class.getName()).log(Level.SEVERE, null, ex);
10 } catch (ClassNotFoundException ex) {
11         Logger.getLogger(Banque.class.getName()).log(Level.SEVERE, null, ex);
12 }
13 System.out.println("loaded");
14
15 /* Etablissement de la connexion */
16 System.out.print("Connecting␣to␣the␣database...␣");
17 Connection conn = DriverManager.getConnection(CONN_URL, USER, PASSWD);
18
19 System.out.println("connected");
20
```

Question 1

1) Afficher la liste des animaux

```
1
2  /* Fonction d'affichage des animaux */
3  private static void listeAnimaux() throws SQLException {
4              Statement st = conn.createStatement();
5
6              ResultSet rs = st.executeQuery("SELECT␣*␣FROM␣LesAnimaux");
7              System.out.println("Liste␣de␣tous␣les␣animaux");
8
9              afficherRes(rs);
10                  System.out.println();
11         st.close();
12 }
13
14 /* Fonction d'affichage generale */
15 public static void afficherRes(ResultSet rs){
```

```
16              try {
17                      ResultSetMetaData rms = rs.getMetaData();
18                      int col = rms.getColumnCount();
19
20                      System.out.print("|");
21                      for(int i=1; i <= col; i++) {
22                              String nomCol = rms.getColumnName(i);
23                              System.out.print(" " + String.format("%15s",
    nomCol) + " |");
24                      }
25                      System.out.println();
26
27                      while(rs.next()) {
28                              System.out.print("|");
29                              for(int i=1; i <= col; i++) {
30                                      String type = rms.getColumnClassName(i);
31
32                                      if (type.equals("java.lang.String")) {
33                                              System.out.print(" " + String.
    format("%15s", rs.getString(i)) + " |");
34                                      } else if (type.equals("java.math.
    BigDecimal")) {
35                                              System.out.print(" " + String.
    format("%15s", String.valueOf(rs.getInt(i))) + " |");
36                                      } else if (type.equals("java.sql.Date")) {
37                                              System.out.print(" " + String.
    format("%15s", rs.getDate(i)) + " |");
38                                      }
39                              }
40                              System.out.println();
41                      }
42              } catch (SQLException e) {
43                      System.out.println("ERREUR : " + e.getMessage());
44              }
45  }
46
```

2) Déplacer un animal de cage

```
1   /* Fonction qui deplace un animal vers une autre cage */
2   private static void deplacerAnimal() throws SQLException {
3               String animal, cage;
4               System.out.println("Quel animal voulez-vous deplacer ?");
5               animal = LectureClavier.lireChaine();
6               System.out.println("Vers quelle cage voulez-vous deplacer "+animal+
    " ?");
7               cage = LectureClavier.lireChaine();
8
9               Statement st = conn.createStatement();
10
11              ResultSet rs = st.executeQuery("SELECT noCage FROM LesCages WHERE
    fonction='"+cage+"'");
12              if(!(rs.next()))
13                      System.out.println("Cette cage n'existe pas");
14              else{
15                  int nb = st.executeUpdate("UPDATE LesAnimaux SET fonction_cage
    ='"+cage+"', noCage="+rs.getInt(1)+" WHERE noma='"+animal+"'");
16                  System.out.println(nb + " ligne (s) modifiee (s).");
17              }
```

```
18              System.out.println();
19              st.close();
20  }
21
```

3) Ajouter une maladie a un animal

```
1   /* Fonction qui ajoute une maladie a un animal */
2   private static void ajouterMaladie() throws SQLException {
3               String animal, maladie;
4               System.out.println("A␣quel␣animal␣souhaitez-vous␣ajouter␣une␣
        maladie␣?");
5               animal = LectureClavier.lireChaine();
6               System.out.println("Quelle␣maladie␣souhaitez-vous␣ajouter␣?");
7               maladie = LectureClavier.lireChaine();
8
9               Statement st = conn.createStatement();
10
11              int nb = st.executeUpdate("INSERT␣INTO␣LesMaladies␣VALUES('"+animal
        +"',␣'"+maladie+"')");
12              System.out.println(nb + "␣ligne␣(s)␣ajoutee␣(s)␣dans␣LesMaladies.")
        ;
13              System.out.println();
14              st.close();
15  }
16
```

4) Valider/Annuler une transaction

```
1   /* Fonction de validation d'une transaction */
2   private static void commit() throws SQLException {
3               conn.commit();
4               System.out.println("Transaction␣validee");
5   }
6
7   /* Fonction d'annulation d'une transaction */
8   private static void rollback() throws SQLException {
9               conn.rollback();
10              System.out.println("Transaction␣annulee");
11  }
12
```

5) Obtenir/Modifier le niveau d'isolation

```
1   /* Fonction permettant d'obtenir le niveau d'isolation */
2   private static void getIsolation() throws SQLException {
3               int level = conn.getTransactionIsolation();
4
5               System.out.print("Niveau␣d'isolation␣:␣");
6               switch(level){
7                   case Connection.TRANSACTION_READ_COMMITTED:
8                       System.out.println("READ␣COMMITTED");
9                       break;
10                  case Connection.TRANSACTION_READ_UNCOMMITTED:
11                      System.out.println("READ␣UNCOMMITTED");
12                      break;
13                  case Connection.TRANSACTION_REPEATABLE_READ:
14                      System.out.println("REPEATABLE␣READ");
```

```
                    break;
                case Connection.TRANSACTION_SERIALIZABLE:
                    System.out.println("SERIALIZABLE");
                    break;
                default:
                    break;
            }
}

/* Fonction permettant de modifier le niveau d'isolation */
private static void setIsolation() throws SQLException {
            System.out.println("*** Choisir un niveau d'isolation : ***");
            System.out.println("0 : READ_UNCOMMITTED");
            System.out.println("1 : READ_COMMITTED");
            System.out.println("2 : REPEATABLE_READ");
            System.out.println("3 : SERIALIZABLE");

            int level = LectureClavier.lireEntier("Entrez un entier");
            boolean ok = true;

            switch(level){
                case 0: level=Connection.TRANSACTION_READ_UNCOMMITTED; break;
                case 1: level=Connection.TRANSACTION_READ_COMMITTED; break;
                case 2: level=Connection.TRANSACTION_REPEATABLE_READ; break;
                case 3: level=Connection.TRANSACTION_SERIALIZABLE; break;
                default: System.out.println("Pas le bon numero");ok=false;
    break;

            }
            if(ok)
                conn.setTransactionIsolation(level);
}

```

6 - Gestion des contraintes

Question 2

1) Le calcul du nombre de maladies pour chaque animal doit être automatisé en fonction de l'ajout ou de la suppression des maladies.

```
CREATE or REPLACE TRIGGER MaladiesTrig
AFTER INSERT or DELETE on LesMaladies
FOR EACH ROW
BEGIN
        IF INSERTING THEN UPDATE LesAnimaux SET nb_maladies = nb_maladies+1
    WHERE nomA=:new.nomA;
        ELSIF DELETING THEN UPDATE LesAnimaux SET nb_maladies = nb_maladies-1
    WHERE nomA=:old.nomA;
        END IF;
END;
/
```

On constate qu'à la suite de la création du déclencheur, lorsqu'on veut ajouter une maladie à un animal dans la table LesMaladies, la valeur nb_maladies associée à cet animal dans la table LesAnimaux est bien incrémentée (et décrémentée lors de la suppression d'une maladie).

2) Des animaux ne peuvent pas être placés dans une cage dont la fonction est incompatible avec ces animaux. On prendra en compte le fait que des animaux peuvent être ajoutés, mais aussi déplacés d'une cage.

```
1  CREATE OR REPLACE TRIGGER DeplacerCageTrig
2  BEFORE INSERT OR UPDATE OF noCage on LesAnimaux
3  FOR EACH ROW
4  DECLARE
5          fct VARCHAR2(20);
6  BEGIN
7          SELECT fonction into fct FROM LesCages WHERE noCage = :new.noCage;
8          IF (fct != :new.fonction_cage) THEN
9                  raise_application_error(-20001, 'cage incompatible');
10         END IF;
11 EXCEPTION
12         WHEN NO_DATA_FOUND THEN
13                 raise_application_error(-20002, 'cage inexistante');
14 END;
15 /
```

3) Des animaux ne peuvent pas être placés dans une cage non gardée. On prendra en compte le fait que des animaux peuvent être ajoutés, mais aussi déplacés d'une cage.

```
1  CREATE OR REPLACE TRIGGER CageGardeeTrig
2  BEFORE INSERT OR UPDATE OF noCage on LesAnimaux
3  FOR EACH ROW
4  DECLARE
5          nb INTEGER;
6  BEGIN
7          SELECT count(*) into nb FROM LesGardiens WHERE noCage = :new.noCage;
8          IF (nb = 0) THEN
9                  raise_application_error(-20003, 'Cage non gardee');
10         END IF;
11 EXCEPTION
12         WHEN NO_DATA_FOUND THEN
13                 raise_application_error(-20002, 'cage inexistante');
14 END;
15 /
```

4) Des animaux de type différent ne peuvent pas cohabiter dans une même cage. On prendra en compte le fait que des animaux peuvent être ajoutés, mais aussi déplacés d'une cage.

Jeux de tests pertinents

7 - Contraintes & JDBC

Question 3

Reprenez les fonctionnalités développées sous forme de programme Java et modifier le code pour prendre en compte les erreurs d'intégrités renvoyés par le SGBD via vos triggers. Il s'agit ici d'annuler les transactions qui violent certaines contraintes métiers.