NFA011 : corrigé de l'examen 2

Exercice n° 1 :

Soit la base de données suivante :

Immeuble (adresse, nbEtages, dateConstruction, nomPropriétaire)

Appartement (adresse, n°appartement, nomOccupant, type, superficie, étage)

Personne (nom, adresse, n°appartement, dateArrivée, dateDépart, âge, profession)

Ecrire les requêtes suivantes en SQL :

a) Donner l'adresse et le nom du propriétaire pour les immeubles ayant plus de 5 étages et construits avant 1950.

```
SELECT adresse, nomPropriétaire
FROM Immeuble
WHERE nbEtages > 5 AND dateConstruction < 1950;
```

b) Donner le nombre de personnes qui habitant un immeuble dont elles sont propriétaires

```
SELECT COUNT(nom)
FROM Personne, Immeuble
WHERE nom = nomPropriétaire
     AND
     Personne.adresse = Immeuble.adresse;
```

c) Donner les noms et adresses des personnes qui ne sont pas propriétaires

```
SELECT nom, adresse
FROM Personne
WHERE nom NOT IN (SELECT nomPropriétaire FROM Immeuble);
```

d) Donner le nom et la profession des propriétaires d'immeubles où il y a des appartements vides

```
SELECT DISTINCT nom, profession
FROM Appartement, Immeuble, Personne
WHERE nom = nomPropriétaire
     AND
     Immeuble.adresse = Appartement.adresse
     AND NOT EXISTS (SELECT *
                     FROM Personne P
                     WHERE P.adresse = Immeuble.adresse
                     AND
                     P.n°appartement = Appartement.n°appartement);
```

Exercice n° 2 :

Ecrivez une procédure stockée PL/SQL qui permet de vérifier, avant l'ajout d'une personne dans la base, si les contraintes suivantes sont vérifiées : dateConstruction de l'appartement occupé ≤ dateArrivée occupant < dateDépart occupant (la dateDépart peut avoir la valeur NULL).

Si une contrainte n'est pas vérifiée, la procédure doit l'indiquer par un message spécifique. Si les contraintes sont vérifiées, la personne est ajoutée dans la base.

La procédure prendra comme paramètres les attributs de la table **Personne** correspondant à la personne à ajouter (nom, adresse, n°appartement, dateArrivée, dateDépart, âge, profession).

Remarques :

1. D'autres solutions peuvent être correctes.
2. L'adresse est clé primaire de **Immeuble** et ne peut donc prendre plusieurs fois une même valeur !
3. Un traitement spécifique est prévu seulement pour l'exception `NO_DATA_FOUND`.
4. Un paramètre `doInsert` a été ajouté pour pouvoir utiliser la procédure dans le programme java du dernier exercice. Pour que l'insertion soit faite, `doInsert` doit être

```
CREATE OR REPLACE PROCEDURE
      VerifieContrainte(nomPersonne     IN Personne.nom%TYPE,
                        adressePersonne IN Personne.adresse%TYPE,
                        appartPersonne  IN Personne.noappartement%TYPE,
                        arriveePersonne IN Personne.dateArrivee%TYPE,
                        departPersonne  IN Personne.dateDepart%TYPE,
                        agePersonne     IN Personne.age%TYPE,
                        profPersonne    IN Personne.profession%TYPE,
                        doInsert        IN INTEGER DEFAULT 0) IS
DECLARE
      dateConstructionImmeuble Immeuble.dateConstruction%TYPE;
BEGIN
      IF arriveePersonne >= departPersonne THEN
            DBMS_OUTPUT.PUT_LINE(`Rejet ' || nomPersonne ||
            ` : arrivée doit être antérieure au depart !');
      ELSE
            SELECT dateConstruction INTO dateConstructionImmeuble
                  FROM Immeuble
                  WHERE adresse = adressePersonne;
            IF dateConstructionImmeuble IS NULL THEN
                  DBMS_OUTPUT.PUT_LINE(`Rejet ' || nomPersonne ||
                  ` : date construction immeuble inconnue !');
            ELSIF arriveePersonne < dateConstructionImmeuble THEN
                  DBMS_OUTPUT.PUT_LINE(`Rejet ' || nomPersonne ||
                  ` : arrivée doit être ultérieure à construction immeuble !');
            ELSIF doInsert != 0
                  INSERT INTO Personne
                        VALUES (nomPersonne, adressePersonne, appartPersonne,
                        arriveePersonne, departPersonne, agePersonne, profPersonne);
            END IF;
      END IF;
EXCEPTION
      WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE(`Aucun immeuble enregistré à cette adresse !');
      WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE(`Procédure VerifieContrainte abandonnée !');
END VerifieContrainte;
```

Exercice n° 3 :

Ecrivez un programme java qui fait les vérifications de dates pour toutes les personnes déjà présentes dans la base (table **Personne**). Pour chaque personne, le programme appellera la procédure stockée PL/SQL définie précédemment.

Remarques :

1. D'autres solutions peuvent être correctes, une version minimale est acceptée.
2. La procédure est appelée avec `doInsert = 0` (il est également possible de l'appeler sans donner de valeur à ce paramètre, car la valeur par défaut dans la procédure est 0) pour ne pas faire d'insertion (les personnes sont **déjà** dans la base, une tentative d'insertion d'une personne déjà présente lèverait une exception `DUP_VAL_ON_INDEX`).

```
import java.sql.*;
import oracle.jdbc.driver.*;

public class Examen2Jdbc {
      private final static String JDBC_DRIVER = "oracle.jdbc.driver.OracleDriver";
      private final static String JDBC_URL = "jdbc:oracle:thin:@kirov:1521:NFA011";
      private Connection connexion = null;
      public Examen2Jdbc() {}

      public void initConnection(String login, String password) throws SQLException {
            try {
                  Class.forName(JDBC_DRIVER);
                  connexion = DriverManager.getConnection(JDBC_URL, login, password);
            } catch (ClassNotFoundException e) {
                  throw new SQLException("Driver " + e.getMessage() + " absent !");
            }
      }
      public void closeConnection() {
            try {
                  if (connexion != null) { connexion.close(); }
            } catch (SQLException e) { // Exception ignoree
            }
      }
      public void verifierToutesPersonnes() throws SQLException {
            Statement stmt = null;
            CallableStatement callStmt = null;
            ResultSet rSet = null;
            String sStmt = "select * from Personne";
            String pCall = "{call VerifieContrainte(?,?,?,?,?,?,?,?)}";
            try {
                  stmt = connexion.createStatement();
                  rSet = stmt.executeQuery(sStmt);
                  callStmt = connexion.prepareCall(pCall);
                  while (rSet.next()) {
                        callStmt.setString(1, rSet.getString(1));
                        callStmt.setString(2, rSet.getString(2));
                        callStmt.setInt(3, rSet.getInt(3));
                        callStmt.setDate(4, rSet.getDate(4));
                        callStmt.setDate(5, rSet.getDate(5));
                        callStmt.setInt(6, rSet.getInt(6));
                        callStmt.setString(7, rSet.getString(7));
                        callStmt.setInt(8, 0);
                        callStmt.execute();
                  }
            } catch (SQLException e) {
                  throw e;
            } finally {
```

```
                try {
                      if (callStmt != null) callStmt.close();
                } catch (SQLException e) { }
                try { if (rSet != null) rSet.close(); } catch (SQLException e) { }
                try { if (stmt != null) stmt.close(); } catch (SQLException e) { }
          }
    }
    public static void main(String[] args) {
          if (args.length != 2) {
                System.err.println("Utilisation: java Examen2Jdbc [login] [pass]");
                System.exit(1);
          }
          Examen2Jdbc monExamen = new Examen2Jdbc();
          try {
                monExamen.initConnection(args[0], args[1]);
                monExamen.verifierToutesPersonnes();
          } catch (SQLException e) {
                e.printStackTrace();
          } finally {
                monExamen.closeConnection();
          }
    }
}
```