NFA011 : corrigé de l'examen 1

Exercice n° 1 :

Tables :

Poisson (n° poisson, nom poisson, âge, poids)

Internaute (n°internaute, nom, prénom, adresse email)

Propriété (n°poisson, n°internaute)

Repas (n°poisson, jour, heure, minute)

Garderie (n°poisson, date_début, date_fin)

a) Combien de poissons ont plus de 40 jours ?

```
SELECT COUNT(NOPOISSON)
FROM Poisson
WHERE AGE > 40;
```

b) Combien de noms de poissons peut-on répertorier sur ce site ?

```
SELECT COUNT(DISTINCT NOMPOISSON)
FROM Poisson;
```

c) Quel est le poids moyen des poissons laissés à la garderie ?

```
SELECT AVG(POIDS)
FROM Poisson, Garderie
WHERE Poisson.NOPOISSON = Garderie.NOPOISSON;
```

d) Quel est l'internaute vainqueur, c'est-à-dire celui dont le poisson a le poids le plus élevé (il peut y en avoir plusieurs en cas d'égalité de poids) ?

```
SELECT NOINTERNAUTE
FROM Internaute, Propriete, Poisson
WHERE Internaute.NOINTERNAUTE = Propriete.NOINTERNAUTE
     AND
     Propriete.NOPOISSON = Poisson.NOPOISSON
     AND
     Poisson.POIDS >= ALL(SELECT POIDS FROM Poisson);
```

ou

```
SELECT NOINTERNAUTE
FROM Internaute, Propriete, Poisson
WHERE Internaute.NOINTERNAUTE = Propriete.NOINTERNAUTE
     AND
     Propriete.NOPOISSON = Poisson.NOPOISSON
```

```
        AND
        Poisson.POIDS = (SELECT MAX(POIDS) FROM Poisson);
```

e) Quels sont les internautes ayant tous leurs poissons à la garderie ?

(requête reformulée : quels sont les internautes qui n'ont aucun poisson qui ne soit pas à la garderie ?)

```
SELECT Internaute.NOINTERNAUTE
FROM Internaute
WHERE NOT EXISTS
          (SELECT Poisson.NOPOISSON
          FROM Poisson, Propriete
          WHERE Poisson.NOPOISSON = Propriete.NOPOISSON
          AND
          Propriete.NOINTERNAUTE = Internaute.NOINTERNAUTE
          AND
          Poisson.NOPOISSON NOT IN
                    (SELECT Garderie.NOPOISSON
                    FROM Garderie));
```

f) Donner tous les couples d'internautes ayant donné le même nom à leur poisson.

```
SELECT I1.NOINTERNAUTE, I2.NOINTERNAUTE
FROM Internaute I1, Internaute I2,
     Propriete PR1, Propriete PR2,
     Poisson PO1, Poisson PO2
WHERE I1.NOINTERNAUTE = PR1.NOINTERNAUTE
     AND
     I2.NOINTERNAUTE = PR2.NOINTERNAUTE
     AND
     PR1.NOPOISSON = PO1.NOPOISSON
     AND
     PR2.NOPOISSON = PO2.NOPOISSON
     AND
     PO1.NOPOISSON > PO2.NOPOISSON
     AND
     PO1.NOMPOISSON = PO2.NOMPOISSON;
```

Exercice n° 2 :

Remarques :

1. D'autres solutions peuvent être correctes. Le traitement des exceptions est minimal.
2. Le n°poisson est clé primaire de **Poisson** et ne peut donc prendre plusieurs fois une même valeur !
3. Tous les SGBD n'implémentent pas CURRENT_DATE, pour que la solution soit correcte il suffisait d'indiquer d'une façon ou d'une autre que ce critère était nécessaire.
4. Les suppressions (DELETE) sont propagées, l'énoncé ne demandait pas d'étudier plus cet aspect.

```
CREATE OR REPLACE PROCEDURE
      NourirPoisson(numeroPoisson IN Poisson.NOPOISSON%TYPE,
                    maxRepas      IN INTEGER            DEFAULT 2,
                    prisePoids    IN Poisson.POIDS%TYPE DEFAULT 2,
                    pertePoids    IN Poisson.POIDS%TYPE DEFAULT 10) IS
DECLARE
      poidsPoisson Poisson.POIDS%TYPE;
      nbRepas INTEGER;
BEGIN
      SELECT POIDS INTO poidsPoisson
            FROM Poisson
            WHERE NOPOISSON = numeroPoisson;
      SELECT COUNT(NOPOISSON) INTO nbRepas
            FROM Repas
            WHERE Repas.NOPOISSON = numeroPoisson
                  AND
                  Repas.JOUR = EXTRACT(DAY FROM CURRENT_DATE);
      IF nbRepas <= maxRepas THEN
            poidsPoisson := poidsPoisson + prisePoids;
      ELSE
            poidsPoisson := poidsPoisson – pertePoids;
      END IF;
      IF poidsPoisson > 0 THEN
            INSERT INTO Repas VALUES (numeroPoisson, EXTRACT(DAY FROM CURRENT_DATE),
                                                     EXTRACT(HOUR FROM CURRENT_TIME),
                                                     EXTRACT(MINUTE FROM CURRENT_TIME));
            UPDATE Poisson SET POIDS = poidsPoisson
                  WHERE NOPOISSON = numeroPoisson;
      ELSE
            DELETE FROM Poisson   WHERE NOPOISSON = numeroPoisson;
            DELETE FROM Propriete WHERE NOPOISSON = numeroPoisson;
            DBMS_OUTPUT.PUT_LINE(`Suppression poisson ' || numeroPoisson);
      END IF;
EXCEPTION
      WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE(`Procedure NourirPoisson abandonnee');
END NourirPoisson;
```

Exercice n° 3 :

Remarques : d'autres solutions peuvent être correctes, une version minimale est acceptée.

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

public class ExamenJdbc {
      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 ExamenJdbc() {}
```

```
	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 nourrirTousPoissons() throws SQLException {
		int numeroPoisson;
		Statement stmt = null;
		CallableStatement callStmt = null;
		ResultSet rSet = null;
		String sStmt = "select distinct nopoisson from Garderie where date_debut
	<= current_date AND date_fin >= current_date";
		String pCall = "{call NourirPoisson(?,?,?,?)}";
		try {
			stmt = connexion.createStatement();
			rSet = stmt.executeQuery(sStmt);
			callStmt = connexion.prepareCall(pCall);
			while (rSet.next()) {
				numeroPoisson = rSet.getInt(1);
				callStmt.setInt(1, numeroPoisson);
				callStmt.execute();
				System.out.println("Poisson " + numeroPoisson + " nouri");
			}
		} 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 ExamenJdbc [login] [pass]");
			System.exit(1);
		}
		ExamenJdbc monExamen = new ExamenJdbc();
		try {
			monExamen.initConnection(args[0], args[1]);
			monExamen.nourrirTousPoissons();
		} catch (SQLException e) {
			e.printStackTrace();
		} finally {
			monExamen.closeConnection();
		}
	}
}
```