JAVA

JDBC : Traitement des requêtes SQL : Récupération des résultats

Récupération des résultats

Lors de la récupération des résultats, ceux-ci sont formatés d'une certaine manière. Il est alors nécessaire d'accéder à la structure des enregistrements. Pour cela, ***ResultSet*** fournit une méthode permettant de récupérer une instance de ***ResultSetMetaData***, qui comprend toutes les propriétés de la structure (nom de colonne, type, accès…) :

public ResultSetMetaData getMetaData();

ATTENTION : l'indice de la première colonne est 1, il n'y a pas de colonne 0.

Exemple

```
ResultSet rs = state.executeQuery("SELECT * FROM vendeur");
ResultSetMetaData rsmd = rs.getMetaData();
int colomnCount = rsmd.getColumnCount();
for (int i = 1; i <= colomnCount; i++) {
System.out.print(rsmd.getColumnName(i) + "\t"); }
 while (rs.next) {
System.out.print("-");
for (int i = 1; i <= colomnCount; i++)
{ System.out.print(rs.getObject(i) + "\t");
 }
 System.out.println(""); }
```

Lecture des données

Pour pouvoir récupérer les données contenues dans l'instance de ResultSet, celui-ci met à disposition des méthodes permettant de :

- Positionner le curseur sur l'enregistrement suivant : **public boolean next();**
- Accéder à la valeur du type donné et à la colonne donnée (par indice ou par nom) de l'enregistrement actuellement pointé par le curseur :
  - public Xxxx getXxxx(int indiceCol);
  - public Xxxx getXxxx(String nomCol); où Xxxx est le type de la valeur
    - getString(int indiceCol);
    - getInt(int indiceCol);

Exemple

```
ResultSet rs = state.executeQuery("SELECT *
FROM vendeur");
while (rs.next) {
 System.out.println("[" +
rs.getString("nomvendeur") + ", " +
rs.getInt("agevendeur") + ", " +
rs.getDate("dateembvendeur") + "]");
 }
```

Modification des données

Pour pouvoir modifier les données contenues dans l'instance de ResultSet, celui-ci met à disposition des méthodes permettant de :

- Modifier la valeur du type donné et à la colonne donnée (par indice ou par nom) de l'enregistrement actuellement pointé par le curseur :
  - public void updateXxxx(int indiceCol, Xxxx value) ; public void updateXxxx(String nomCol, Xxxx value) ; où Xxxx est le type de la valeur
  - updateString(int indiceCol, String value) ; updateInt(int indiceCol, int value) ;

  Appliquer dans la base de données les changements effectués sur l'enregistrement actuellement pointé par le curseur : public void updateRow();
- Insérer dans la base de données l'enregistrement actuellement pointé par le curseur : public void insertRow();
- Aller sur un emplacement vide permettant d'insérer un nouvel enregistrement : public void moveToInsertRow();

```
ResultSet rs = state.executeQuery("SELECT * FROM vendeur");
 rs.next();
...
 rs.updateString("nomvendeur", "Dupont");
rs.updateInt("agevendeur", 28);
rs.updateDouble("salvendeur", 1300);
updateRow(); rs.moveToInsertRow();
 rs.updateString("nomvendeur", "Robert");
rs.updateInt("agevendeur", 32);
rs.updateDouble("salvendeur", 1215);
rs.insertRow();
```

Fermeture de connexion

Après toutes les actions effectuées sur la base de données, il faut fermer les instances qui permettent la connexion à celle-ci. Cette action est effectuée par l'appel de la fonction close() des objets Statement, PreparedStatement, CallableStatement, Connection, ResultSet.

Fermeture de connexion

```
 try {
 rs.close();
state.close();
 connect.close(); }
catch (Exception e)
 { ... }
```

REMARQUE : l'instance d'un Statement doit être fermée avant de relancer une autre requête SQL.

Tester