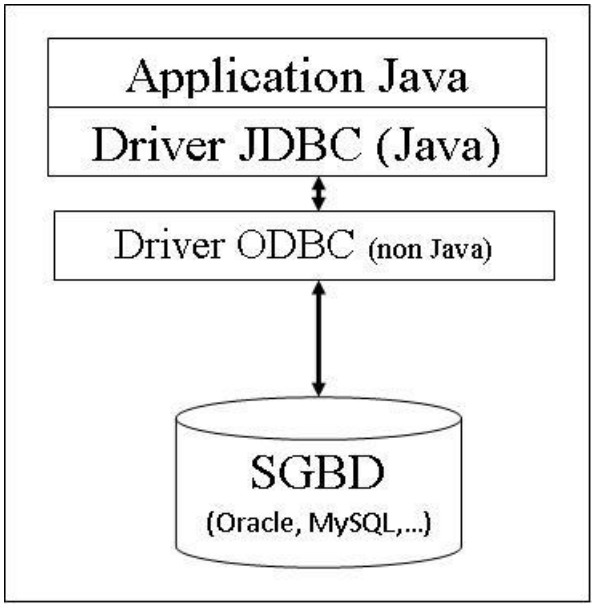
Introduction à JDBC

Définitions

- Un pilote ou driver JDBC est un ensemble de classes qui permettent d'établir une connexion entre un programme java et un système de gestion de bases de données. En fait, c'est une implémentation de l'interface Driver, du package java.sql
- Il existe 4 types de driver:
  - Driver de Type 1: Il permet l'accès au SGBD via le driver ODBC. Appelé aussi ODBC-JDBC bridges.

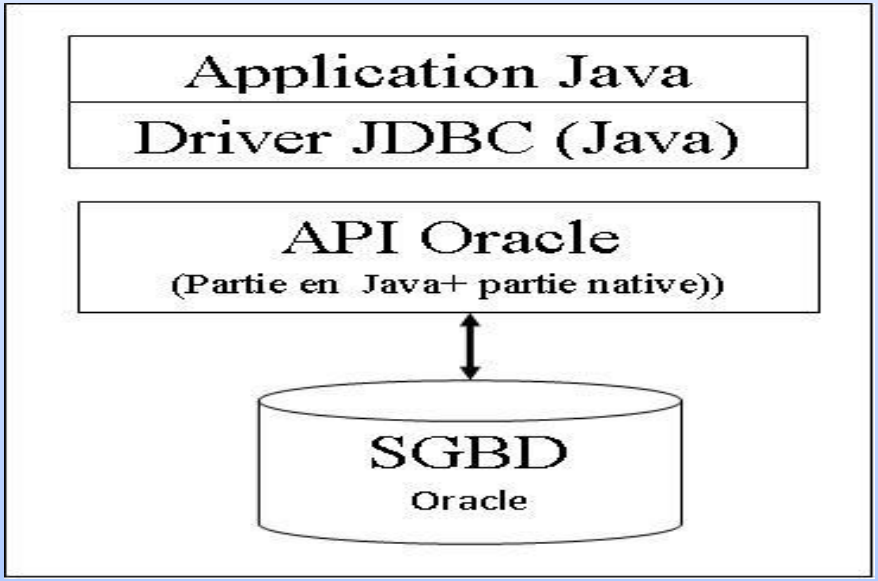
  Chaque requête JDBC est convertie par ce pilote en requête ODBC qui est par la suite convertie une seconde fois dans le langage spécifique de la base de donnée.

Introduction à JDBC



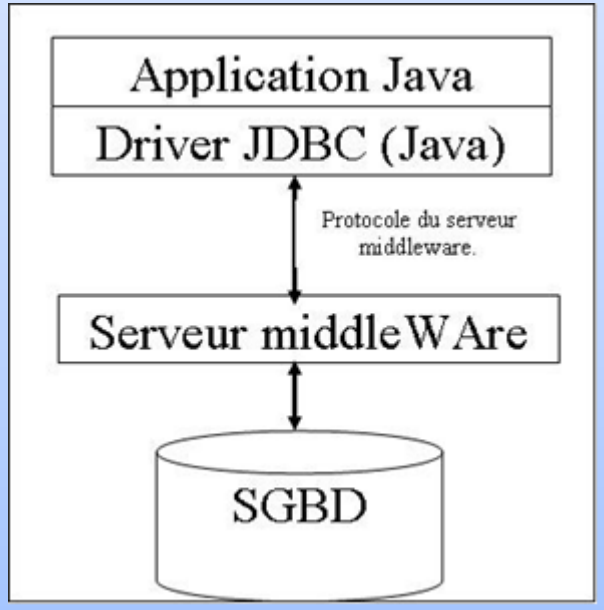
Introduction à JDBC

- Driver de Type 2: Ce type de driver traduit les appels de JDBC à un SGBD particulier, grâce à un mélange d'API java et d'API natives. Ce driver est fourni par le sgbd.(cas de OCI –Oracle Call Interface)



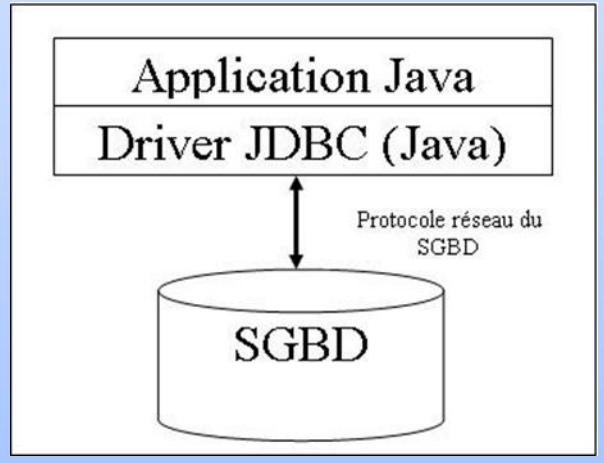
Introduction à JDBC

- Driver de type 3:Ce type de driver (écrit entièrement en java) passe par un serveur intermédiaire pour l'accès au SGBD. Adapté pour le web. – architecture 3tiers. (Idéal lorsqu'on connecte des SGBD différents)



Introduction à JDBC

- Driver de type 4:Ce type de driver (écrit entièrement en java) se connecte directement au SGBD. Connu sous le nom Direct Database Pure Java Driver
- C'est le type de driver utilisé par Oracle, et sans doute le plus performant.
- Chaque constructeur de SGBD fournit son propre Driver



Introduction à JDBC

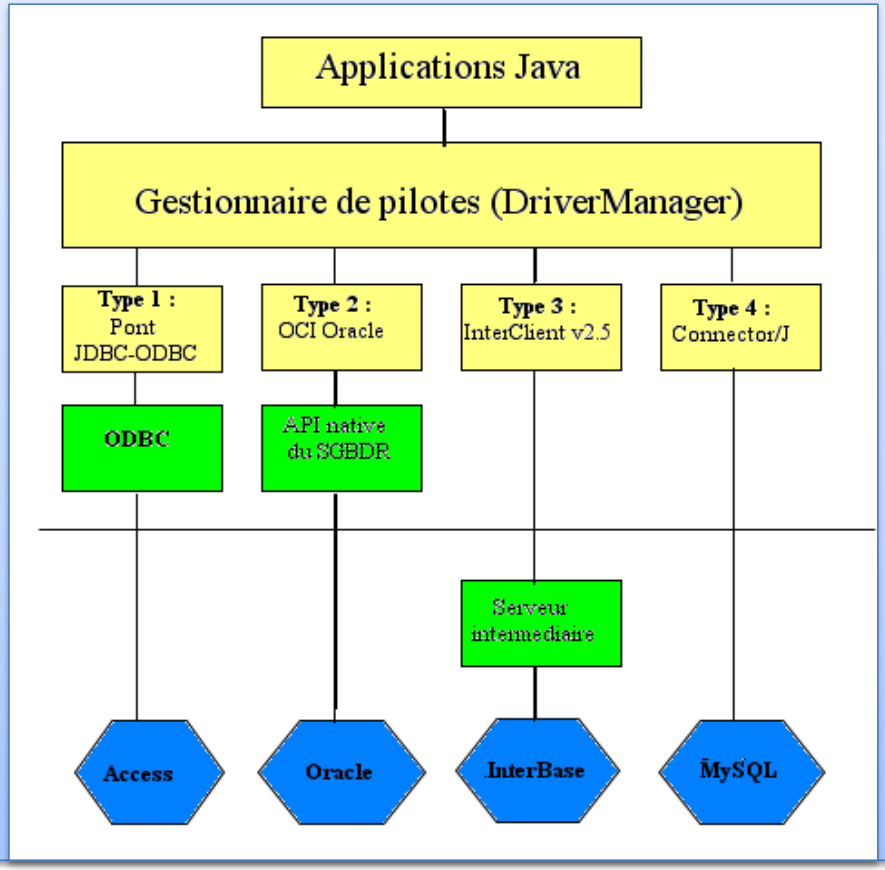
- Dans le driver de type 4 on retrouve le driver pour oracle (thin driver ou

oracle.jdbc.driver.OracleDriver) dont le format de la chaine de connexion à une base de

données est sous formes
:jdbc:oracle:thin:@paramètresdeconnexion

Introduction à JDBC

- Architecture



• En jaune: la technologie Java
• En vert: intermédiaires non java, comme les API natives et les serveurs intermédiaires.
•En bleu: les différents SGBD.

Introduction à JDBC

- Fonctionnement:
  - Connexion à la base de données.
    - Charger le Driver
    - Établir la connexion.
  - Traitement des commandes SQL
  - Traitement des résultats lors d'un SELECT
  - Fermeture de la connexion.

Introduction à JDBC

- Chargement du Driver.
- Pour oracle, les driver est téléchargeable à

http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-10201-088211.html

- Inclure le .JAR (ojdbc14.jar)qui contient le Driver dans le ClassPath .
- Charger le driver et enregistrer le auprès du DriverManager.
- Comment charger de Driver ?
  - Soit avec la méthode forName de la classe Class. Une exception de type ClassNotFoundException sera lancée si le driver est introuvable.
  - Soit enregistrer directement une instance du driver auprès du DriverManager. Une exception de type SQLException sera lancée si le driver est introuvable.

Introduction à JDBC

- Fonctionnement:
  - Importer les packages nécessaires.
  - Connexion à la base de données
    - Charger le Driver
    - Établir la connexion.
  - Traitement des commandes SQL
  - Traitement des résultats lors d'un SELECT
  - Fermeture des Statement et du ResultSet
  - Fermeture de la connexion.

Introduction à JDBC

- Chargement du Driver. (antérieur à JDBC 4)
- Pour oracle, les driver est téléchargeable à http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.htm
- Inclure le .JAR qui contient le Driver JDBC .(ojdbc6.jar ou ojdbc7.jar);
- Importer les packges:

import java.sql.*; et import oracle.jdbc.*;

- Charger le driver et enregistrer le auprès du DriverManager.

Comment charger de Driver ?

- Soit avec la méthode forName de la classe Class. Une exception de type ClassNotFoundException sera lancée si le driver est introuvable.

```
Class.forName("oracle.jdbc.OracleDriver");
```

- Soit enregistrer directement une instance du driver auprès du DriverManager. Une exception de type SQLException sera lancée si le driver est introuvable.

```
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
```

Introduction à JDBC

- Si on utilise la metode forName (String nom) de la classe Class alors le nom du Driver (de la classe) doit-être passé comme un string (chaine de caractères).
- Pour Oracle, cette chaine de caractère est: oracle.jdbc.driver.OracleDriver

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

Si on enregistre le driver avec la methode registreDriver (Driver driver) de la classe DriverManager alors il faut utiliser la methode new pour instancier un objet de type Driver.

```
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver())
```

Introduction à JDBC

Exemple 1: Utilisation de la méthode forName de la classe Class

```
try
        {
            Class.forName("oracle.jdbc.OracleDriver");

System.out.println("Driver chargé");

        }
catch (ClassNotFoundException    cnfe)
        {
            System.out.println("driver manquant");
        }
```

Introduction à JDBC

Exemple 2 :Avec le DriverManager

```
try
        {
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

System.out.println("Driver chargé");


        }
        catch (SQLException se)
        {
           System.out.println("driver manquant");
        }
```

Introduction à JDBC

- Pour une base de données Access

```
try
	{
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
	    System.out.println("Driver chargé");



	}
catch (ClassNotFoundException cnfe)
	{
	    System.out.println("driver manquant");
	}
```

Introduction à JDBC

- Établir une connexion à la base de données:

Une fois que le Driver chargé, alors on pourra établir une connexion à la base de données.

Une connexion est obtenue grâce à la méthode getConnection de la classe DriverManager.

Cette methode est de la forme:

getConnection (String url, String user, String password)

url, désigne la base de donnée,

user, désigne le nom de l'usager,

Password désigne le mot de passe

Pour Oracle, l'url est de la forme : jdbc:oracle:thin:@IP:port:orcl

getConnection retourne un objet Connection.

Introduction à JDBC

- Exemple :

```
String ur1= "jdbc:oracle:thin:@205.237.244.251:1521:orcl";
String ur2="user1";
String ur3="oracle1";
Connection connexion =null;
…..
 connexion =DriverManager.getConnection(ur1, ur2, ur3);
….
```

Introduction à JDBC

```
try
      {
         Class.forName("oracle.jdbc.OracleDriver");
System.out.println("Driver chargé");
      }
catch (ClassNotFoundException    cnfe)
      {
         System.out.println("driver manquant");
      }
try
   {
connexion =DriverManager.getConnection(ur1, ur2, ur3);
System.out.println("connecté");
    }
```

Introduction à JDBC

```
catch (SQLException se)
   {
      System.out.println(se);

   }


   finally
      {
      try
      {
       if (connexion != null )
       {
          connexion.close();}
      }
      catch (SQLException se){}
      }
```

Introduction à JDBC

À partir de la version 3 du JDBC:

On utilise un objet OracleDataSource qui définit un ensemble de méthodes permettant l'accès à la base de données Oracle

1. Importer les packages

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

2. Créer un objet OracleDataSource : ods
3. Faire le «setting » des propriétés pour le ods avec les méthodes setURL, setUser..
4. Appeler la méthode getConnection de OracleDataSource pour ods

L'exception renvoyée est de type SQLException.

Introduction à JDBC

getConnection()	Obtient une connexion à la base de données
setServerName	Définit le nom du serveur de la base de données utilisée. (ou son adresse IP)
setPortNumber	Définit le port du serveur de base s de données
setPassword	Définit le mot de passe avec lequel la connexion est obtenue.
setUser	Définit le username de l'usager
setServiceName	Définit le nom du service de la base de donnée : orcl
setDatabaseName	Définit le nom de la base de donnée : orcl
setURL	Définit la chaîne de connexion qui sera utilisée pour se connecter à la base de données. Elle est de la forme String url="jdbc:oracle:thin:@111.111.111.111:1521:orcl". Au lieu de définir l'IP , le protocole, le service name séparément, il est préférable d'utiliser l'URL.
getServerName	obtient le nom du serveur de la base de données utilisée. (ou son adresse IP)
getPortNumber	obtient le port du serveur de base s de données
getPassword	obtientt le mot de passe avec lequel la connexion est obtenue.
getUser	Obtient le username de l'usager
getServiceName	Obtient le nom du service de la base de donnée : orcl
getDatabaseName	Obtient le nom de la base de donnée : orcl

Introduction à JDBC

Avec la version 4 du JDBC (que nous allons utiliser)

Exemple:

```
String user1 ="user1";
String mdep ="oracle1";
String url="jdbc:oracle:thin:@205.237.244.251:1521:orcl";
Connection conn = null;
```

Dans le try

```
OracleDataSource ods =  new OracleDataSource();
ods.setURL(url);
ods.setUser(user1);
ods.setPassword(mdep);
conn = ods.getConnection();
```

catch(SQLException se)

Introduction à JDBC

Traitement des commandes SQL.

1- Passer la requête

Les requêtes SQL seront exécutées au travers :

Un Statement, pour les requêtes simples, exécutées une seule fois.

un PreparedStatement : pour des requêtes paramétrées, qui sont donc exécutées plusieurs fois .

un CallableStatement, pour les procédures stockées. (et fonctions).

Les Statement sont crées par la méthode CreateStatement() de l'objet Connection.

Introduction à JDBC

```
Exemple1:// on suppose que la connexion est déjà faite.
try
   {
connexion =DriverManager.getConnection(ur1, ur2, ur3);
System.out.println("connecté");

Statement stm=connexion.createStatement();

    }
```

Introduction à JDBC

2- Exécuter la requête:

Les méthodes ExecuteUpdate (sql), ExecuteQuerry(sql) et Execute(sql) permettent d'exécuter les requêtes SQL.

ExecuteUpdate (sql): est utilisée pour exécuter des requêtes DML. Elle retournent le nombre de lignes affectées par la requête.

int n = objetStatement.executeUpdate(String RequêteSQL);

ExecuteQuery(sql), permet d' exécuter une requête de type SELECT. Elle retourne un ResultSet qui contient un ensemble de tuples (enregistrements sélectionnés).

objetResultSet=objetStatement.executeQuery(String requeteSQL);

Execute(sql): retourne un booléen pour la présence ou non des résultats. C'est cette méthode qui est utilisée dans les appels de procédures et des fonctions.

Introduction à JDBC

```
String sql1="delete from employes where photo is null";
Statement stm = null;
try
   {
      stm=conn.createStatement();
      int total = stm.executeUpdate(sql1);
     System.out.println("nombre de lignes supprimées" + total);
    }

   catch(SQLException se){
   System.out.println (se);}

     finally {

     stm.close();
   }
```

Introduction à JDBC

Exploitation du ResultSet

L'interface ResultSet représente une table de lignes et de colonnes.

- Une ligne représente un enregistrement
- Une colonne représente un attribut particulier de la table
- Un objet de type ResultSet possède un pointeur sur l'enregistrement courant. À la réception de cet objet, le pointeur se trouve devant le premier enregistrement.
- On y accède ligne par ligne, puis colonne par colonne dans la ligne.

- À la création du ResultSet, le curseur de parcours est positionné avant la première occurrence à traiter.
- **Le premier indice étant 1**

Introduction à JDBC

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();** Renvoi un booléen indiquant la présence d'un élément suivant.
- Accéder à la valeur d'un attribut (par indice ou par nom) de l'enregistrement actuellement pointé par le curseur avec les méthodes getString(), getInt(), getDate() …..
  - public String getString(int indiceCol);
  - public String getString(String nomCol);
  - etc…

Introduction à JDBC

```
Connection conn = null;
Statement stm2= null;
ResultSet rst=null;
String sql2 = "select nom, prenom from employes";

   try {
       stm2=conn.createStatement();
       rst= stm2.executeQuery(sql2);
       while(rst.next())
                  {
      String noms = rst.getString(1);
      String prenoms = rst.getString("prenom");
      System.out.print(noms +" "+ prenoms);
      System.out.println();
                  }
     }
     catch(SQLException se1)    {System.out.println (se1); }

      finally
       {
      stm2.close();
      rst.close();
      }
```

Introduction à JDBC

Méthode du ResultSet	**Rôle**
boolean isBeforeFirst()	booléen qui indique si la position courante du curseur se trouve avant la première ligne
boolean isAfterLast()	booléen qui indique si la position courante du curseur se trouve après la dernière ligne
boolean isFirst()	booléen qui indique si le curseur est positionné sur la première ligne
boolean isLast()	booléen qui indique si le curseur est positionné sur la dernière ligne
boolean first()	déplacer le curseur sur la première ligne
boolean last()	déplacer le curseur sur la dernière ligne
boolean previous()	déplacer le curseur sur la ligne précédente. Le booléen indique si la première occurrence est dépassée.
Boolean next()	déplacer le curseur sur la ligne suivante. Le booléen indique si la dernière occurrence est dépassée.

Introduction à JDBC

Méthode du ResultSet	**Rôle**
void beforeFirst()	Déplace le curseur avant le premier enregistrement
void afterLast()	Déplace le curseur après le dernier enregistrement
String getString(int columnIndex)	Obtient la valeur de la colonne dont l'index est passé en paramètre sous forme de String dans le langage Java
int getInt(int columnIndex)	Obtient la valeur de la colonne dont l'index est passé en paramètre sous forme d'entier dans le langage Java
void close()	Fermer le ResultSet
ResultSet MetaData getMetaData()	Obtient les informations sur les méta-Données du ResultSet (nom de colonnes, nombre de colonnes, types des colonnes
Object getObject(int columnIndex)	Obtient la valeur de la colonne dont l'index est passé en paramètre. (Utilisé beaucoup dans les appeles de procédures)

Introduction à JDBC

Types de ResultSet

1. **Lecture du ResultSet** :

Il existe 3 types de ResultSet:

- TYPE_FORWARD_ONLY: c'est le type par défaut. Le ResultSet n'est pas «Scrollable». La lecture se fait du premier au dernier enregistrement.
- TYPE_SCROLL_INSENSITIVE: le ResultSet est «Scrollable». On le parcours du début à la fin ou de la fin vers le début. On peut aussi se positionner sur une ligne en particulier avec les méthodes absolute ou relative. Le ResultSet n'est pas sensible aux changements effectués dans la base de données lorsque celui-ci est ouvert.
- TYPE_SCROLL_SENSITIVE: identique au précèdent. Et en plus lorsqu'il est ouvert et que la BD est mise à jour, les changements apparaissent dans le ResultSet.

Introduction à JDBC

2. Modification du ResultSet

- CONCUR_READ_ONLY: c'est le type par défaut. Cela veut dire que le contenu du ResultSet ne peut être modifié par programmation
- CONCUR_UPDATABLE: des mises à jours peuvent se faire par le ResultSet.
- Les méthodes du ResultSet qui permettent de le modifier sont:
  - deleteRow();
  - updateRow();
  - insertRow(); dans ce cas, il faut se positionner à un emplacement vide du resultSet (en général au début) avec la méthode MoveToInsertRow()

Introduction à JDBC

Dans le cas d'une modification (updatable), voici les opérations pour une mise à jour ou une insertion

Modifier la valeur du type et de la colonne donnée (par indice ou par nom) de l'enregistrement actuellement **pointé** :

- public void updateString(int indiceCol, String value);
- public void updateString(String nomCol, String value);
- public void updateInt(int indiceCol, Int value);
- public void updateInt(String nomCol, Int value);
- public void updateDouble(int indiceCol, double value);
- Etc… (selon les type de colonnes voir le tableau TYPE de données JDBC à la page 21).

Introduction à JDBC

Utilisation du PréparedStatement:

Utilisé pour des requêtes avec paramètres. Dans ce cas on dit que la requête est pré-compilées.

- Les paramètres sont représentés par un « ?»
- Les paramètres sont passés dans l'ordre de leur présentation de la requête .
- La requête SQL est passée lors de la création du PréparedStatement.

```
PreparedStatement  stm = connexion.prepareStatement(sql);
stm.executeUpdate();
```

La methode setX(indice_de_colonne,Type_colonne) permet d'affecter les valeurs aux paramètres:

Introduction à JDBC

- Les paramètres et les valeurs sont passés comme suit :
- [Objet PreparedStatement].setString([index],[objet String]);
- [Objet PreparedStatement].setBoolean([index],[valeur]);
- [Objet PreparedStatement].setInt([index],[valeur]);
- [Objet PreparedStatement].setFloat([index],[valeur]);
- Etc…
- La requête est exécuté par executeUpdate() ou executeQuery
- On peut effacer le contenu des paramètres par la méthode clearParameters() du PreparedStatement.

Introduction à JDBC

Exemple

```
String sql ="update employes set salaire = ? where codeDep = ?"
PreparedStatement  stm = connexion.prepareStatement(sql);
        stm.setInt(1,5000 );
        stm.setInt(2, 12);
        int total = stm.executeUpdate();
        System.out.println("nombre de mise à jour pour Salaire " + total );
       stm.clearParameters();
```

Écrire le code JDBC qui permet d'effectuer des insertions dans la table «Employes». Les colonnes sont: num, nom, prenom et salaire.

Introduction à JDBC

Exemple1:

```
Resultat.absolute(2);
Resultat.deleteRow();
```

Exemple2:-- modification—

```
String sql2 = "select nom, prenom from employes";)

Statement stm
=connexion.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);
ResultSet rst4=stm.executeQuery(sql2);
rst.next();
rst.updateString("nom", "Coluche");
rst.updateString("prenom", "Moses");
rst.updateRow();
connexion.commit();
```

Introduction à JDBC

```
String sql = "select numemp, nomemp, prenomemp from employesbidon";
Statement stm1=connexion.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE);
  ResultSet Resultat = stm1.executeQuery(sql);
  Resultat.moveToInsertRow();
  Resultat.updateInt(" numemp ", 14);
  Resultat.updateString(" nom ", "Fafard");
  Resultat.updateString(3,  "Chantal");
  Resultat.insertRow();
  connexion.commit();
  Resultat.first();
// affichage.
```

Introduction à JDBC

```
String sql = "select numemp, nomemp, prenomemp from employesbidon";
Statement stm1=connexion.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE);
  ResultSet Resultat = stm1.executeQuery(sql);
  Resultat.absolute(3);
  Resultat.deleteRow();
// suite du code
```

Introduction à JDBC

Autres informations du ResultSet: Le ResultsSet présente une interface qui permet d'avoir des informations concernant les données qu'il contient. Ces informations sont appelées des méta-données.

L'interface qui obtient les méta-données du ResultSet est appelée: ResultSetMetaData.

Pour obtenir les «metadata» on utilise la methode getMetaData() du ResultSet.

Introduction à JDBC

```
String sql2="select * from employesclg";


stm = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_UPDATAB
rst = stm.executeQuery(sql2);

ResultSetMetaData structure = rst.getMetaData();
int nbcolones = structure.getColumnCount();
System.out.println("Nbre de colonnes" + " " + nbcolones);

for(int i=1; i<=nbcolones; i++)

{
    System.out.println(structure.getColumnName(i)+ "\t\t" +
structure.getColumnTypeName(i));

}
```