

# JDBC : Interconnexion Java / SQL



Christian Soutou

<http://w3.univ-tlse2.fr/icare/soutou>

## JDBC

- Bibliothèques de classes (`java.sql.*` inclus dans le JDK 1.4 : JDBC 3.0)
- JDBC supporte le *multi-thread*
- Support de SQL92 *Entry Level*
- Drivers  
<http://java.sun.com/products/jdbc/jdbc.drivers.html>

Christian Soutou - 2005

2

## JDBC

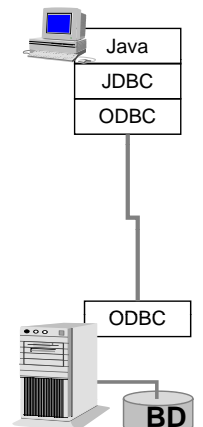
- Communication en mode client-serveur déconnecté :
  - connexion à la base de données
  - émissions d'instructions SQL et exploitation des résultats provenant de la base de données
  - déconnexion de la base
- Applicatif Java
  - Classe
  - Applet côté client ou Servlet,
  - EJB (*Enterprise Java Beans*)
  - Procédure cataloguée côté serveur

Christian Soutou - 2005

3

## Types de Pilotes JDBC

- Type 1 (*JDBC-ODBC Bridge*)
  - Client épais
    - ODBC + Pilote ODBC du SGBD
  - Utilisation
    - Application Java mais pas Applet
    - Test, 1<sup>ère</sup> implantation de JDBC
    - Base MS Access



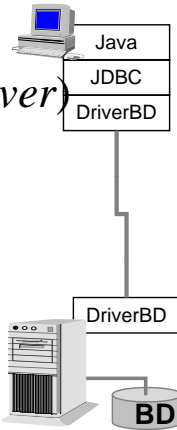
Christian Soutou - 2005

4

# Types de Pilotes JDBC

- Type 2 (*Native-API Partly-Java Driver*)

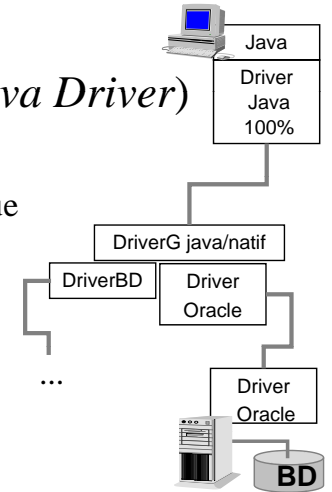
- Client épais
  - Pilote JDBC du SGBD
  - Bibliothèques client du SGBD
- Utilisation
  - Mieux que type 1 : code optimisé
  - Applications Java 2 tiers mais pas applet



# Types de Pilotes JDBC

- Type 3 (*Net Protocol All-Java Driver*)

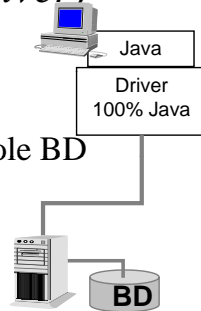
- Client léger
  - Pilote JDBC + Driver générique
- Utilisation
  - Multibases
  - Application Java ou Applets
  - Architectures 3 tiers



# Types de Pilotes JDBC

- Type 4 (*Native Protocol All-Java Driver*)

- Client léger
  - Pilote JDBC (300Ko pour Oracle *thin*)
  - Création de sockets pour émuler le protocole BD
- Utilisation
  - Applets
  - Architectures 2 tiers



# L'API JDBC (java.sql)

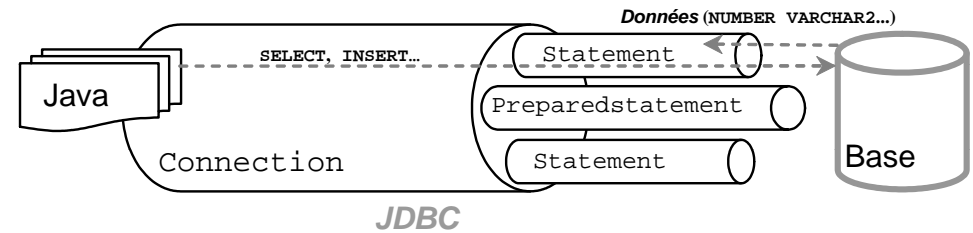
Interface-Classes-Exceptions	Description
Driver	Pilote appelé par le Driver Manager
Connection	Session avec une BD
Statement	Exécution d'ordres SQL standard
PreparedStatement	Gestion des ordres SQL avec paramètres
CallableStatement	Gestion des appels de procédures stockées
ResultSet	Tableau de lignes résultant d'un ordre
DriverManager	Envoie les demandes de connexion au driver
SQLException	Extension de java.lang.Exception, détail de l'erreur SQL
...	

# Variables d'environnement

- PATH : chemin de la machine virtuelle Java (en général installé dans C:\jdk1.4.0)
- CLASSPATH : paquetage Oracle en fonction du pilote choisi (*OracleHome\jdbc\lib\paquetage*)

Version du JDK utilisé	Paquetage JDBC Oracle
JDK 1.1	classes11.jar
JDK 1.2 et JDK 1.3	classes12.jar
JDK 1.4	ojdbc14.jar

# Mécanismes de JDBC

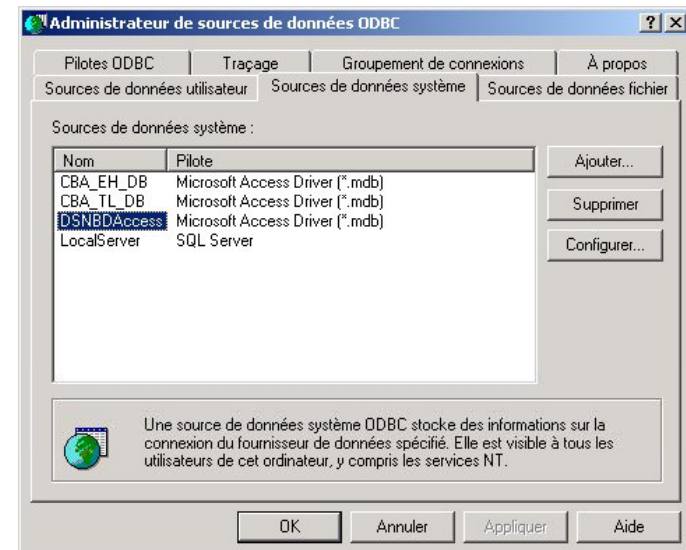


- Importation de paquetages (oracle.sql, ...)
- Enregistrement du pilote
- Ouverture de connexion(s) (Connection)
- Création d'un état (Statement, PreparedStatement ou CallableStatement)
- Gestion de ResultSet
- Fermeture des résultats, des états et connexions

# Enregistrement du pilote 1 (pont JDBC-ODBC)

```
try
{Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
try
{Connection cx =
DriverManager.getConnection("jdbc:odbc:DSNBDAccess", "", "");
Statement etat = cx.createStatement();
ResultSet rset = etat.executeQuery("SELECT * FROM Avion");
...
} catch(SQLException ex) { ... }
} catch (ClassNotFoundException ex) { ... };
```

# Data Source Name ODBC



# Enregistrement du pilote 4 Oracle

```
import oracle.jdbc.driver.*;
...
try
{
    DriverManager.registerDriver
        (new oracle.jdbc.driver.OracleDriver());
    Connection cx =
        DriverManager.getConnection
            ("jdbc:oracle:thin:@teline.cict.fr:1526:etu923",
            "soutou", "mdp");
    Statement etat = cx.createStatement ();
    ...
    // Fermeture de la connexion
    cx.close();
}
catch(SQLException ex) { ... }
```

# Méthodes de Connection

Méthodes	Description
Statement createStatement()	Création d'un objet destiné à recevoir un ordre SQL sans paramètre
PreparedStatement prepareStatement(String)	Précompile un ordre SQL avec paramètres
CallableStatement prepareCall(String)	Déclare l'appel d'une procédure cataloguée avec d'éventuels paramètres
void setAutoCommit(boolean)	Positionne le commit automatique
void commit()	Valide la transaction
void rollback()	Invalide la transaction
void close()	Ferme la connexion

# Méthodes de Statement

Méthodes	Description
ResultSet executeQuery(String)	Exécute un SELECT et renvoie un ensemble de lignes
int executeUpdate(String)	Exécute un ordre SQL LMD passé en paramètre et renvoie le nombre de lignes traitées
boolean execute(String)	Exécute un ordre SQL dynamique passé en paramètre et renvoie true si c'est un SELECT false si c'est un LMD
void setMaxRows(int)	Limite le nombre d'enregistrements à extraire par toute requête issue de cet état.
int getUpdateCount()	Nombre de lignes traitées par un ordre SQL dynamique ou -1 si c'est un SELECT
void close()	Ferme l'état

# Résumé de l'API JDBC

