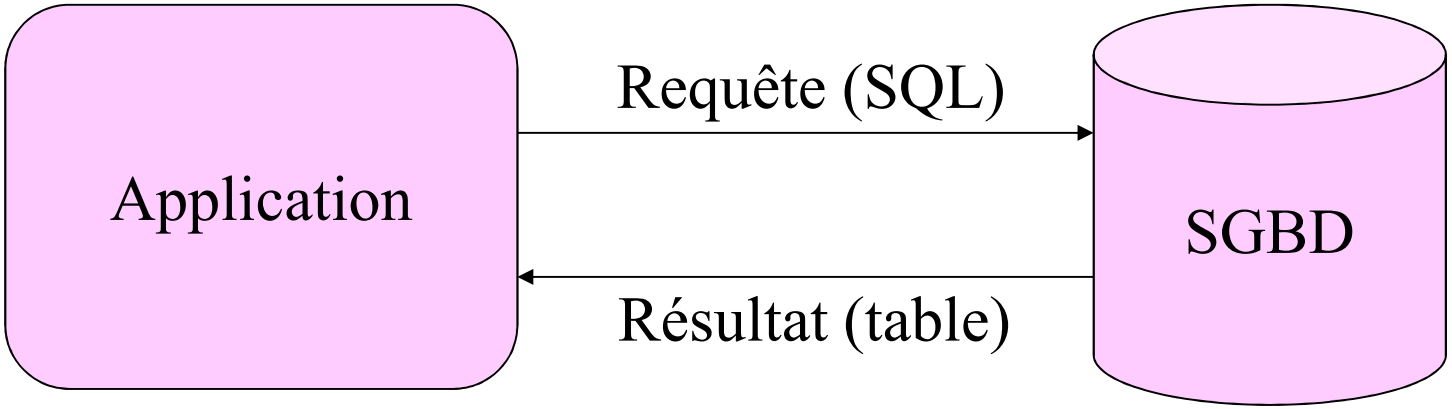
JDBC

ENSIMAG 2ème année

Architecture Client-Serveur

- Le SGBD est ici vu comme un Serveur de Requêtes
- Communication directe entre l'application et le SGBD



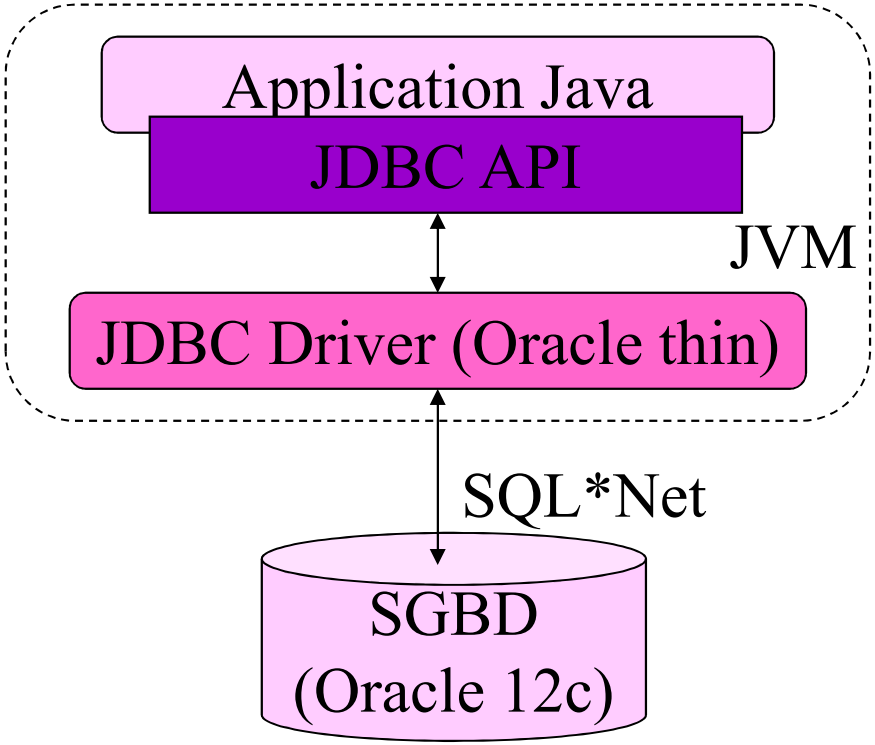
JDBC : Description

- **Ensemble de classes et d'interfaces permettant à des applications Java d'utiliser les services proposés par un ou plusieurs SGBD de manière uniforme**

- **Une API pur Java (package *java.sql*)**
- **Supporte SQL2**
  - Indépendamment d'un SGBD particulier

Que peut-on faire avec JDBC ?

- **Se connecter à un ou plusieurs SGBD**
  - Driver et modèle de connexion
- **Envoyer des requêtes SQL**
- **Récupérer une structure contenant le résultat.**



Manipuler une BD avec JDBC

- Initialiser la JVM (Driver JDBC)
- Ouvrir une connexion
- Créer des requêtes SQL
- Exécuter les requêtes
  - Sélection
  - Insertion/Modification/Suppression
- Fermer la connexion

Drivers JDBC

- **Indispensable pour pouvoir interagir avec un SGBD**
- **Différents drivers, même pour un seul SGBD**
- **Charger plusieurs drivers permet de se connecter à plusieurs SGBD**
- **Chargement d'un driver :**
  - Création d'une instance du driver
  - Enregistrement auprès du *DriverManager*
- **Connexion à un SGBD**
  - *DriverManager* utilise le premier driver compatible

Exemple (driver Oracle 12c)

```
import java.sql.*;
…
try {
  DriverManager.registerDriver(
  new oracle.jdbc.driver.OracleDriver());
  …
} catch (SQLException e) {
  e.printStackTrace();
}
…
```

Connexion à un SGBD

- **Une connexion est représentée par une instance de la classe *Connection*.**
- **Pour se connecter, il faut :**
  - Une URL : « jdbc:<sous-protocole>:<base> »
  - Un login
  - Un mot de passe
- **Une application peut avoir plusieurs connexions sur la même ou sur différentes bases de données.**

Exemple de connexion

```
import java.sql.*;
…
static final String url =
  « jdbc:oracle:thin:@ensioracle1.imag.fr:1521:ensioracle1 »;
…
try {
  …
  Connection con = DriverManager.getConnection(url, « login »,
« passwd »);
  …
} catch (SQLException e) {
  e.printStackTrace();
}
…
```

Sous-protocole

Base

Connexions et Transactions…

- **A l'ouverture d'une connexion, une transaction démarre**
- **Par défaut, JDBC est en *autocommit* : commit après chaque requête.**
  - Peut (doit !) être changé par la méthode *void setAutoCommit*(*boolean*) de la classe *Connection* (**FORTEMENT** recommandé)
- **Terminaison de transactions**
  - Méthodes *void commit*() et *void rollback*() de la classe *Connection*.
  - Toute terminaison entraîne le démarrage d'une nouvelle transaction.

…Connexions et Transactions

- **Points de sauvegarde**
  - Création
    - Méthode *Savepoint setSavepoint(String name)* de la classe *Connection*
  - Abandon partiel
    - Méthode *void rollback(Savepoint savepoint)* de la classe *Connection*

Requêtes SQL simples

- **L'interface *Statement* permet d'envoyer des requêtes au SGBD**
- **Une requête SQL simple est représentée par une instance d'objet implantant l'interface *Statement,* créé à partir d'une instance de *Connection*.**
- **L'exécution d'une requête est typée :**
  - Sélection
  - Mise à jour

Sélection de données

```
import java.sql.*;
…
try {
  …
  Statement stmt = con.createStatement();
  ResultSet rs = stmt.executeQuery(« select * from emp »);
  …
} catch (SQLException e) {
  e.printStackTrace();
}
…
```

Résultats de requêtes

- Permet de récupérer les différents attributs (méthodes getInt(i), getFloat(i), getString(i), …, où i est le numéro ou le nom d'un attribut)
- Permet la navigation dans le résultat (méthodes next(), previous(), etc).

```
import java.sql.*;
…
try {
  …
  ResultSet rs = stmt.executeQuery(« select * from emp »);
  while (rs.next()) {
    System.out.println(« Employe «  + rs.getString(« ename »)
                  + «  -> salaire : » + rs.getFloat(« sal »));
  }
  …
```

Mise à jour de données

```
import java.sql.*;
…
try {
  …
  Statement stmt = con.createStatement();
  int nb = stmt.executeUpdate(« insert into… »);
  …
```

- La méthode *executeUpdate*() permet l'exécution de requêtes de mise à jour (*insert*, *update*, *delete*, *create*, *alter* et *drop*)
- La valeur de retour contient le nombre de tuples réellement mis à jour.

Requêtes paramétrées

- Intéressant si les paramètres de la requête sont calculés par l'application

```
import java.sql.*;
…
try {
   …
   PreparedStatement pstmt = con.prepareStatement(
     « update emp set sal = ? where ename = ? »);
   …
   pstmt.setFloat(1, 10000.0);
   pstmt.setString(2, « Dupont »);
   int nb = pstmt.executeUpdate();
   …
```

Fermeture des objets

- **Le ramasse-miettes de la JVM ferme automatiquement les objets *Statement* et *PreparedStatement*, *Resultset* et *Connection*.**
- **Néanmoins, il vaut mieux les fermer soi-même (méthode *close()*)**
  - Libération des ressources du SGBD
  - Libération rapide de mémoire