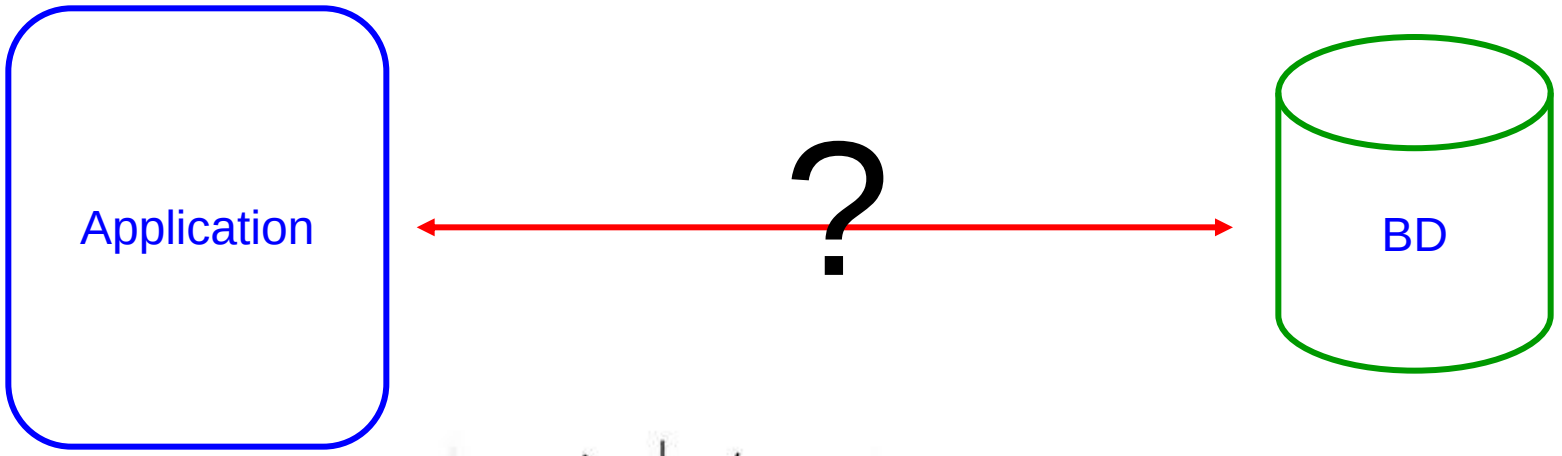
Formation Session 4 2018

Formateur: AMEUR .K
E-Mail: ameur.khadidja@univ-ouargla.dz

Université Kasdi Merbah Ouargla
Faculté des nouvelles technologies de l'information et de la communication
Département d'informatique et technologie d'information

comment manipuler une base de données en Java ?

Introduction : le problème



JDBC pour exécuter, depuis un programme Java, l'ensemble des ordres SQL reconnus par la base de données cible.

Introduction: JDBC

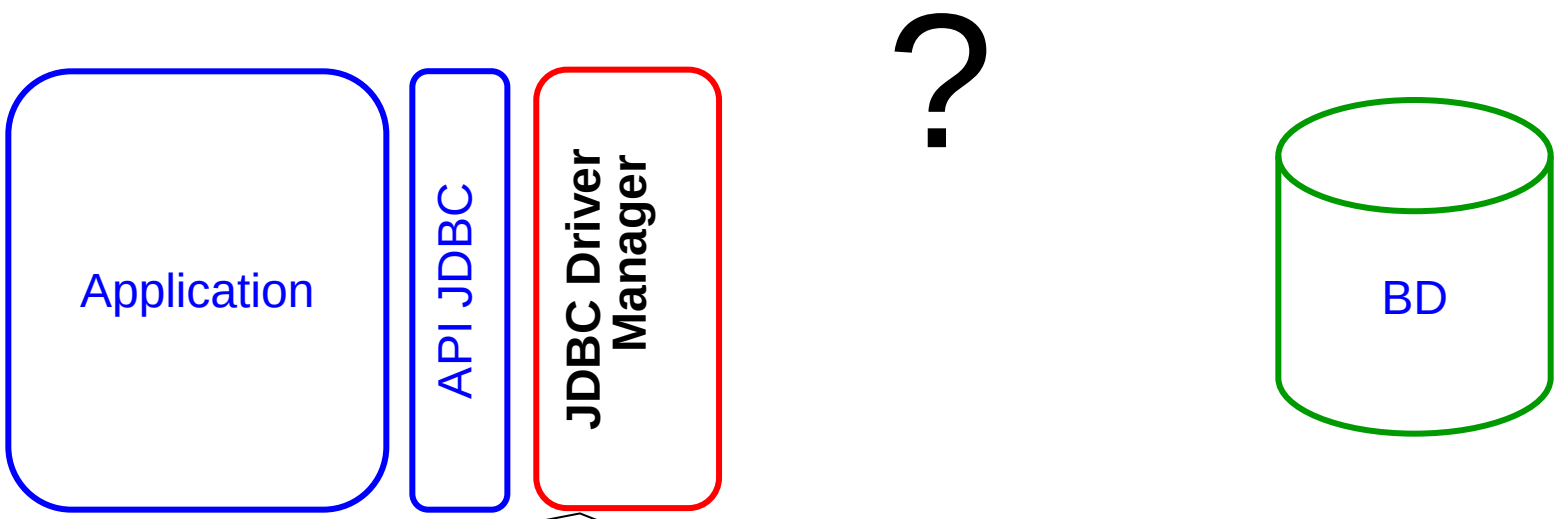
JDBC (*Java DataBase Connectivity*) est une API (*Application Programming Interface*) qui permet d'exécuter des instructions SQL.

JDBC fait partie du JDK (*Java Development Kit*).

Paquetage **java.sql** :

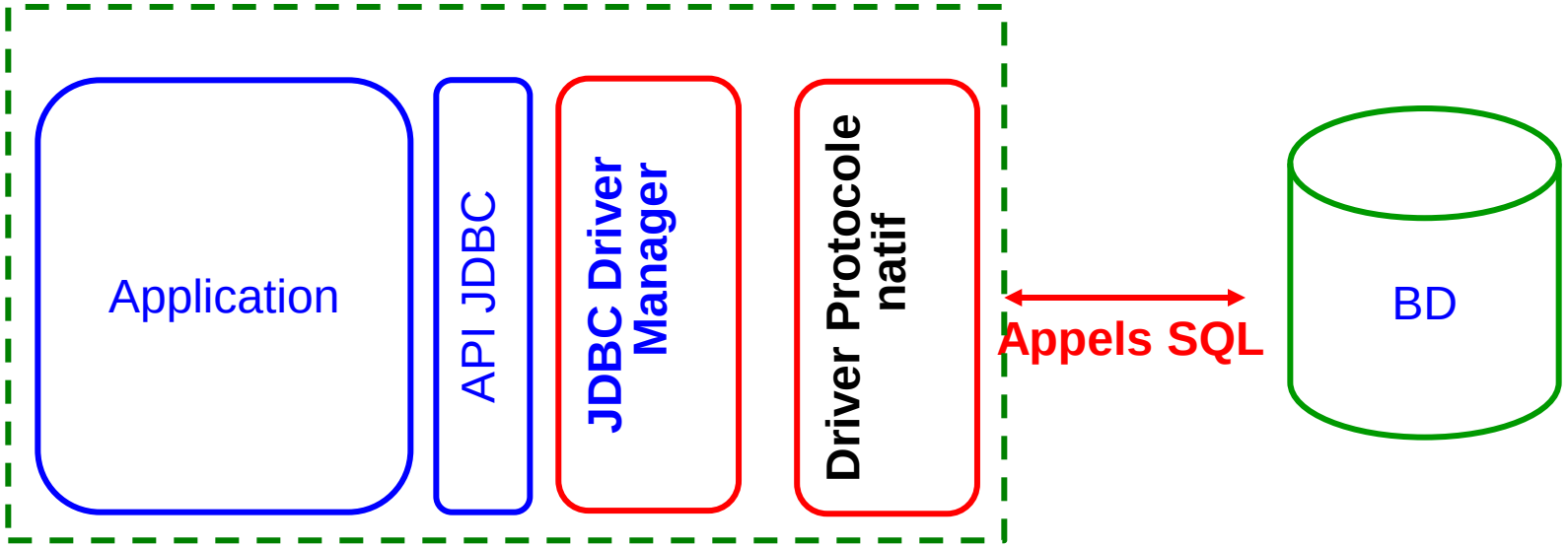
```
import java.sql.*;
```

Introduction: Gestion du pilote



Le pilote...
Il va établir le lien avec la base de données, en sachant \lui parler".
Dans JDBC : des classes chargées de gérer un pilote...
Des pilotes existent pour mySQL, postGresSQL, ACCESS,...

Introduction: Gestion du pilote



Préparatifs

- D’abord installer un driver JDBC
  - E.g. SQL server 2000 de Microsoft

    http://msdn2.microsoft.com/en-us/sql/aa336272.aspx
  - MySQL : MySQL Connector : mysql-connector-java-5.0.8-bin.jar (le plus récent ou le plus adapté à votre version java)

Les étapes principales

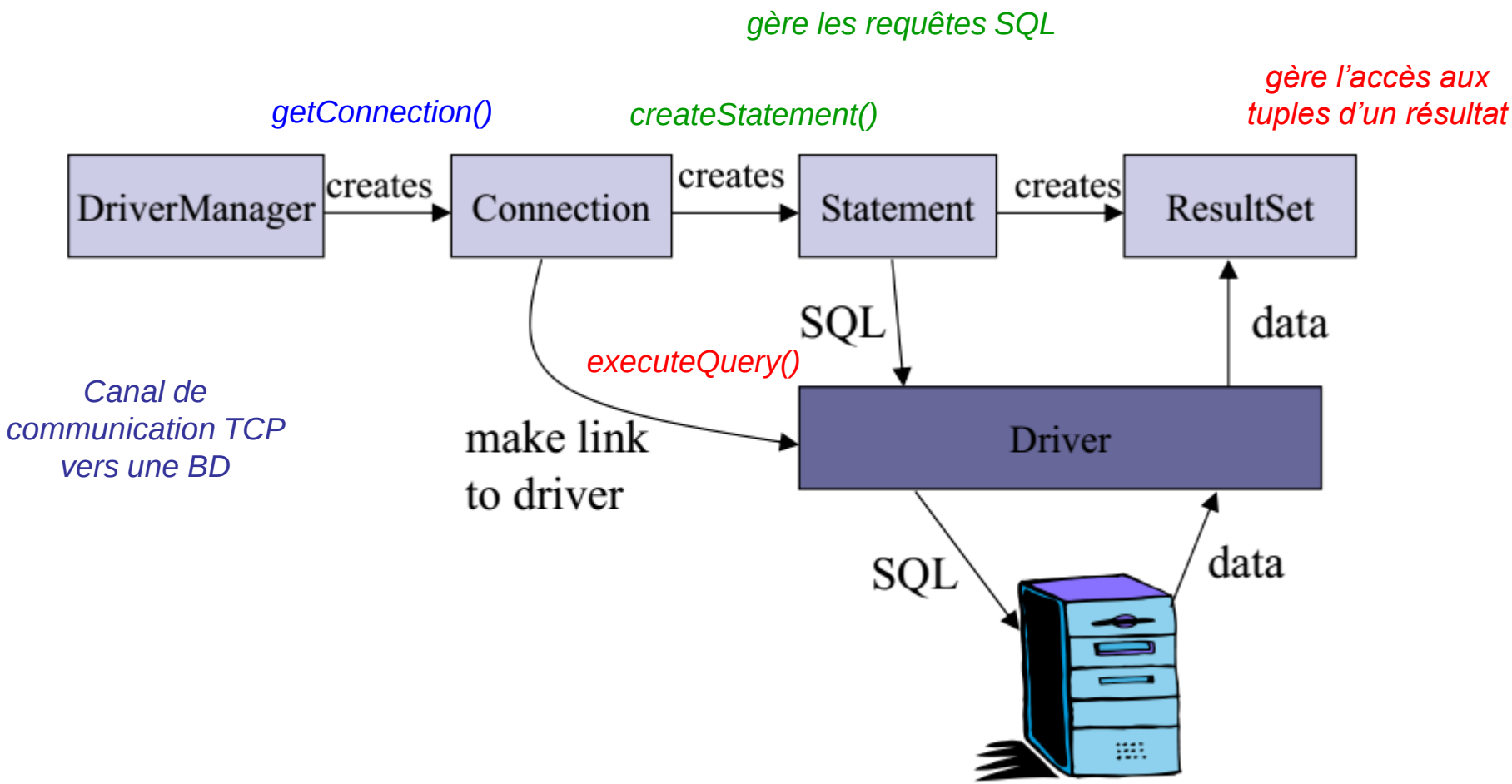
Étape 1: charger le pilote

Étape 2: établir une connexion

Étape 3: Requête SQL

- Envoyer une requête SQL
- Manipuler le résultat

Étape 4 Déconnexion


gère les requêtes SQL
gère l'accès aux tuples d'un résultat
getConnection()
createStatement()
DriverManager
creates
Connection
creates
Statement
creates
ResultSet
SQL
data
executeQuery()
Canal de communication TCP vers une BD
make link to driver
Driver
SQL
data

Étape 1: charger le pilote

- Charger le pilote (driver)
  - contient toutes les classes nécessaire pour communiquer avec une base de données
  - utiliser la méthode forName de la classe Class
  - EX :
    - MySQL:
      Class.forName("com.mysql.jdbc.Driver");
    - Pont ODBC-JDBC
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  - Cette méthode charge en mémoire la classe demandée et exécute son éventuel bloc static.
    - static { BaseDriver.registerDriver(new SQLServerDriver()); }
  - Pour que cela fonctionne, il faut définir la variable d'environnement CLASSPATH et inclure le répertoire contenant les classes du driver

Étape 2: établir une connexion

- Pour établir la connexion avec SQL Server, il faut préciser
  - le nom de la machine (ou son numéro IP),
  - le port où le service SQL est démarré (par ex : 3306 pour localhost),
  - le nom de la base de données,
  - le login utilisé ainsi que son mot de passe.

```
try {
String strClassName = "com.mysql.jdbc.Driver  "  ;
Class.forName(strClassName);
String strUrl = "jdbc:mysql://localhost:3306/" + dbName;
Connection conn = DriverManager.getConnection(strUrl, login="root", passwd="");
    // . . .
conn.close();
}
catch(ClassNotFoundException e) {
    System.err.println("Driver non chargé !");
    e.printStackTrace();
} catch(SQLException e) {
    //
}
```

Charger le pilote

Établir la connexion

opérations

Étape 2: établir une connexion

- Établir la connexion avec MySQL
  - DriverManager: sa méthode statique getConnection va créer un objet de connexion de la classe Connection.

Étape 3: Requête SQL

- L’exécution d’une requête SQL passe par l'utilisation d'une classe, spécifique au pilote utilisé, implémentant l'interface **Statement**

- Un objet de type **Statement** se doit d'être adapté à la base manipulée. JDBC ne fournit que l'interface Statement, qui est implantée par différentes classes du pilote chargé

- Obtenir un objet Statement: avec la méthode **createStatement**.

Exemple

```
try { String strClassName = "com.mysql.jdbc.Driver";
Class.forName(strClassName);

String strUrl = "jdbc:mysql://localhost:3306/MaBase" ;
Connection conn = DriverManager.getConnection(strUrl, "root", "");

String strInsert = "INSERT INTO T_Users (Login, Password, ConnectionNumber)
VALUES ('Toto', 'Titi', 0);";

Statement st0 = conn.createStatement();

st0.executeUpdate(strInsert);

conn.close();
}
catch(ClassNotFoundException e
    // . . .
} catch(SQLException e) {
    // . . .
}
```

Créer un Statement

Exécuter un ordre SQL

Id	Login	Password	ConnectionNumber
1	SkyWalker	Luc	30
2	Plisken	Snake	18
3	Ripley	Helen	19
4	Gordon	Flash	12
5	Kent	Clark	1

Exécuter une requête SELECT

- l'ordre SQL **"SELECT * FROM T_Users;"**
- L'appel à **"executeQuery"** renvoie au final un objet de type **ResultSet**

```
try {
    String strClassName = "com.mysql.jdbc.Driver";
    Class.forName(strClassName);

    String strUrl = "jdbc:mysql://localhost:3306/MaBase;"
    Connection conn = DriverManager.getConnection(strUrl,"root", "");

    Statement st1 = conn.createStatement();
    String strQuery = "SELECT * FROM T_Users;";
    String strQuery2 = « select login from T_users »;
    ResultSet rs1 = st1.executeQuery(strQuery);
    ResultSet rs2 = st1.executeQuery(strQuery2);

    // . . . Utilisation du ResultSet . . .
    conn.close();
}
catch(ClassNotFoundException e) {
```

Requête

Exécuter la requête et stocker le résultat

Manipuler le résultat

- On peut identifier chaque colonne de la base de donnée
  - Par son index
  - Par son nom

```
String Query = "SELECT * FROM T_Users;";
ResultSet rs1 = st1.exexcuteQuery(Query);
while(rs1.next()) {
    String pass = rs1.getString(« Password »);

    System.out.print("Id[" + rs1.getInt(1) + "]"
            + rs1.getString(2)
            + "[" + rs1.getString("Password") + "] "
            + rs1.getInt("ConnectionNumber") ); }
conn.close();
```

Modifier le résultat ou la base

- Se positionner sur le premier enregistrement
  - rs1.first();
- Ou avancer jusqu'à l'élément voulu : rs1.next()
- Modifie la valeur du Password dans le résultat
  - rs1.updateString("Password", "toto");
- Pour appliquer les modifications dans la base de données:
  - rs1.updateRow();

JDBC

Quelques notions de Meta Données (Metadata)

1. Les meta données sont des données sur les données.

2. Chaque ResultSet possède ses propres MetaDonnées.

3. Elles sont utilisées pour obtenir les noms des colonnes dans un ResultSet ainsi que le type des données qui se trouvent dans chacune d'elles.

```
try
{
   // Obtenir les en-têtes de colonne.
   ResultSetMetaData rsmd = rs.getMetaData();

   for ( int i = 1; i <= rsmd.getColumnCount(); ++i )
      enTeteColonne.addElement( rsmd.getColumnName( i ) );
```

Obtenir les meta-données

Utiliser les meta-données

Les méta données sont stockées dans un ResultSetMetaData.