Java :: Teradata

Utilisation de Teradata depuis java

<http://lotfi-chaari.net/ens/NFP121/Chap_Bases_donnees.pdf>

<https://forgetcode.com/Teradata/435-TERADATA-with-JAVA>

1. Establishing connection

1. private void buildCon()
2. {
3. try
4. {
5. Class.forName("com.ncr.teradata.TeraDriver");
6. con = DriverManager.getConnection(serverURL, username, password);
7. }
8. catch (Exception e)
9. {
10. e.printStackTrace();
11. }
12. }

2. Execute query

1. Statement st = con.createStatement();
2. ResultSet rset = null;
3. String sql = "Your query !!!";
4. rset = st.executequery(sql);

3. Close connection

1. private void closeCon()
2. {
3. if(con != null)
4. {
5. try
6. {
7. this.con.close();
8. }
9. catch (SQLException e)
10. {
11. e.printStackTrace();
12. }
13. }

<http://blog.paumard.org/cours/jdbc/chap02-apercu-exemple.html>

D’autres liens celui-ci ☺

**3. Un exemple simple**

**3.1. Chargement du pilote de la base**

Le pilote d’une base de données est tout simplement une classe Java instance de l’interface java.sql.Driver. Comme nous l’avons déjà vu, il est possible de charger une classe Java explicitement dans le code, par appel de la méthode Class.forName().

Chaque éditeur de base de données fournit une telle classe, généralement dans un JAR qui constitue le pilote complet. Le nom complet de cette classe varie donc d'un éditeur à l'autre. Pour chaque base, il faut donc trouver ce nom. Dans le cas de MySQL, il s'agit de com.mysql.jdbc.Driver. L'ouverture d'une connexion sur une base MySQL aura donc la forme suivant :

* Tout d’abord chargement du pilote MySQL, récupération d’un objet de type Class : Class.forName("com.mysql.jdbc.Driver") ;
* Invocation de la méthode newInstance() sur cet objet, qui nous retourne une instance de ce driver, de type java.sql.Driver ;
* Enregistrement de ce pilote auprès du gestionnaire de pilotes JDBC de la machine Java : DriverManager.registerDriver(pilote).

Ensuite ouverture de la connexion proprement dite :

Connection con = DriverManager.getConnection(chaineDeConnexion, user, password) ;

L’objet chaîneDeConnexion est de la classe String. C’est une concaténation des éléments suivants :

* jdbc:mysql : indique le protocole utilisé pour accéder à la base de donnée. Ce protocole est propre à chaque base.
* localhost:3306 : indique le nom d'hôte qui héberge la base de données, ainsi que le port d'accès.
* nom\_de\_la\_base : le nom de la base de données à laquelle on souhaite se connecter.

Ces différents éléments se concatènent en une chaîne unique qui a la forme suivante : jdbc:mysql://localhost:3306/db\_test

Ces deux méthodes jettent bien sûr leurs propres exceptions, notamment si la classe du pilote n’est pas trouvée, si une erreur s’est glissée dans l’URL de connexion, si l’hôte est inaccessible, ou s’il se trouve une erreur dans les identifiants de connexion.

**3.2. Création d’une requête**

Pour pouvoir passer une requête à la base, il faut un objet de type Statement. Cet objet s’obtient en faisant appel à la méthode createStatement() de notre objet de type Connection.

Statement smt = con.createStatement() ;

On peut envoyer ensuite trois types de requêtes :

* Des sélections : smt.executeQuery(chaîneSQL) ;
* Des mises à jour : smt.executeUpdate(chaîneSQL) ;
* Des exécutions de procédures stockées : smt.execute(chaîneSQL).

**3.3. Récupération d’un résultat**

On récupère le résultat dans un objet de type ResultSet. Dans le cas d’une mise à jour de la base, ce qui correspond aux commandes SQL insert, update, create, drop, delete, on utilise la commande :

int i = smt.executeUpdate("insert into … values …") ;

L’entier de retour représente le nombre de lignes affectées dans le cas d’un insert, d’un update ou d’un delete.

Dans le cas d’une requête de sélection, un objet de type ResultSet est renvoyé.

ResultSet resultSet = smt.executeQuery("select nom, age from Marins") ;

L’objet resultSet contient les lignes renvoyées par la requête.

Enfin, pour parcourir ces lignes, il faut utiliser les méthodes de l’interface ResultSet. Voici un exemple.

while (rs.next()) {

String nom = rs.getString("nom") ;

int age = rs.getInt("age") ;

}

Les champs des lignes sélectionnées sont maintenant disponibles sous forme d’objets Java, ce qui était bien notre objectif.

Nous reverrons toutes ces méthodes et cette procédure en détails dans les paragraphes suivants.

**3.4. Exemple complet**

Voici un exemple complet du code que nous venons de présenter très rapidement.

**Exemple 1. Un premier exemple de connexion**

**import** java.sql.\* ;

**import** java.io.\* ;

**import** com.mysql.jdbc.driver.\* ;

**public** **class** TestDB {

**public** **static** **void** main (String[] args) {

**try** {

// chargement de la classe par son nom

Class c = Class.forName("com.mysql.jdbc.Driver") ;

Driver pilote = (Driver)c.newInstance() ;

// enregistrement du pilote auprès du DriverManager

DriverManager.registerDriver(pilote);

// Protocole de connexion

String protocole = "jdbc:mysql:" ;

// Adresse IP de l’hôte de la base et port

String ip = "localhost" ; // dépend du contexte

String port = "3306" ; // port MySQL par défaut

// Nom de la base ;

String nomBase = "test\_db" ; // dépend du contexte

// Chaîne de connexion

String conString = protocole + "//" + ip + ":" + port + "/" + nomBase ;

// Identifiants de connexion et mot de passe

String nomConnexion = "user" ; // dépend du contexte

String motDePasse = "passwd" ; // dépend du contexte

// Connexion

Connection con = DriverManager.getConnection(

conString, nomConnexion, motDePasse) ;

// Envoi d’un requête générique

String sql = "select \* from Marins" ;

Statement smt = con.createStatement() ;

ResultSet rs = smt.executeQuery(sql) ;

**while** (rs.next()) {

System.out.println(rs.getString("nom")) ;

}

} **catch** (Exception e) {

// gestion des exceptions

}

}

}