

Gestion de Masse de Données (GMD)

Cours 2

Adrien Coulet
adrien.coulet@loria.fr

I. Accéder et extraire des données

② dans un système de gestion de base de données

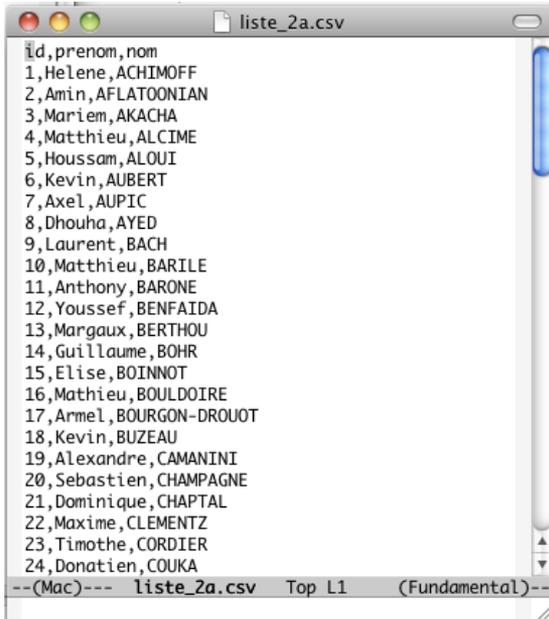
• INSERT

a) insert

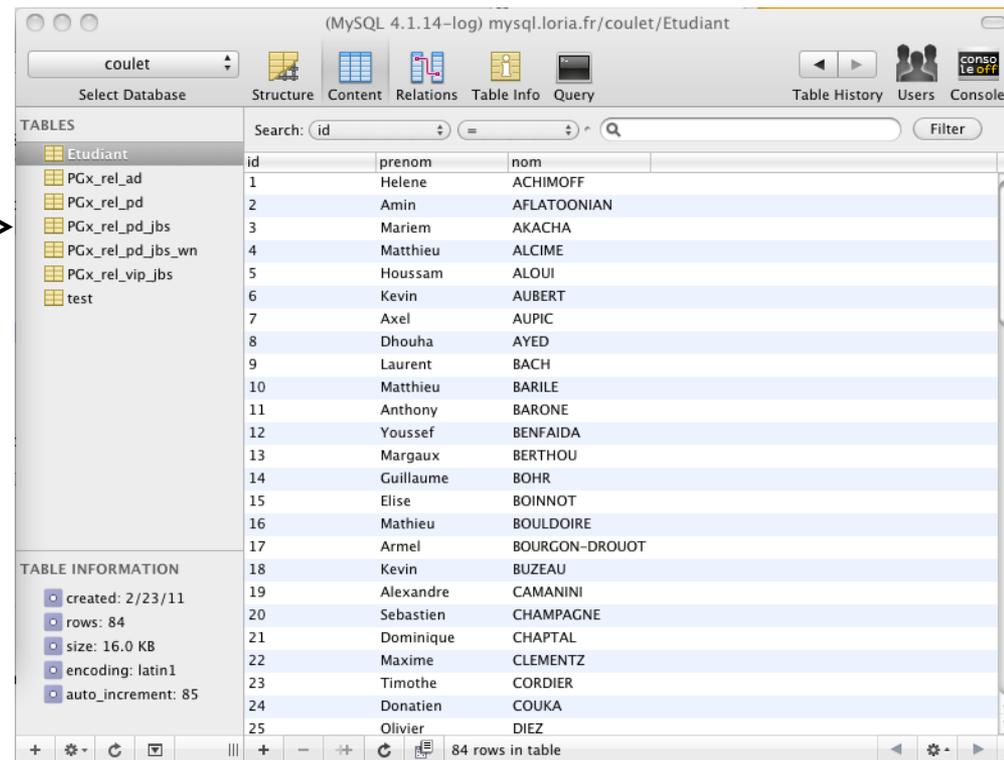
- **unique** : `INSERT INTO nom_de_table (col1,col2,col3) VALUES(15,Elise,BOINNOT);`
- **multiple** : `INSERT INTO nom_de_table (col1,col2,col3) VALUES(15,Elise,BOINNOT), (16,Mathieu,BOULDOIRE);`

• LOAD DATA INFILE (plus rapide)

```
LOAD DATA INFILE '/Users/coulet/teaching/esial/GMD/dossier_test/liste_2a.csv' INTO TABLE Etudiant
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\r'
IGNORE 1 LINES;
```



id	prenom	nom
1	Helene	ACHIMOFF
2	Amin	AFLATOONIAN
3	Mariem	AKACHA
4	Matthieu	ALCIME
5	Houssam	ALLOUI
6	Kevin	AUBERT
7	Axel	AUPIC
8	Dhouha	AYED
9	Laurent	BACH
10	Matthieu	BARILE
11	Anthony	BARONE
12	Youssef	BENFAIDA
13	Margaux	BERTHOU
14	Guillaume	BOHR
15	Elise	BOINNOT
16	Mathieu	BOULDOIRE
17	Armel	BOURGON-DROUOT
18	Kevin	BUZEAU
19	Alexandre	CAMANINI
20	Sebastien	CHAMPAGNE
21	Dominique	CHAPTAL
22	Maxime	CLEMENTZ
23	Timothe	CORDIER
24	Donatien	COUKA



id	prenom	nom
1	Helene	ACHIMOFF
2	Amin	AFLATOONIAN
3	Mariem	AKACHA
4	Matthieu	ALCIME
5	Houssam	ALLOUI
6	Kevin	AUBERT
7	Axel	AUPIC
8	Dhouha	AYED
9	Laurent	BACH
10	Matthieu	BARILE
11	Anthony	BARONE
12	Youssef	BENFAIDA
13	Margaux	BERTHOU
14	Guillaume	BOHR
15	Elise	BOINNOT
16	Mathieu	BOULDOIRE
17	Armel	BOURGON-DROUOT
18	Kevin	BUZEAU
19	Alexandre	CAMANINI
20	Sebastien	CHAMPAGNE
21	Dominique	CHAPTAL
22	Maxime	CLEMENTZ
23	Timothe	CORDIER
24	Donatien	COUKA
25	Olivier	DIEZ

I. Accéder et extraire des données

② dans un système de gestion de base de données

b) `select`

- Faire une simple requête
 - Rappel
 - Important
 - TP
 - Projet
 - votre vie professionnelle
 - Questions

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.SQLException;
```

```
public class ExecuteOneSimpleQuery2 {
```

```
    static String DB_SERVER = "jdbc:mysql://mysql.loria.fr:3306/";
    static String DB = "coulet";
    static String DRIVER = "com.mysql.jdbc.Driver";
    static String USER_NAME = "couletadm";
    static String USER_PSWD = "blabla";
```

Les paramètres de connexions

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

```
        try{
            // Set up the connection
            Class.forName(DRIVER);
            Connection con = DriverManager.getConnection(DB_SERVER+DB, USER_NAME, USER_PSWD);
```

Chargement driver de connexion MySQL : le jar mysql-connector-java-5.0.7-bin.jar doit être dans le path et pas importé
Ouverture de la connexion

```
            // Build and execute the SQL query
            String myQuery = "SELECT * " +
                "FROM Etudiant " +
                "WHERE prenom='Clement'";
```

Création d'un Statement

```
            Statement st = con.createStatement();
            ResultSet res = st.executeQuery(myQuery);
```

Exécution de la requête via le statement

```
            while (res.next()) {
                // if we know attributes names
                int id = res.getInt("id");
                String prenom = res.getString("prenom");
                String nom = res.getString("nom");

                System.out.println(" ");
                System.out.println("Etudiant:"+id+"\n\tNom: "+nom+" "+prenom);
            }
```

Itération sur le ResultSet

```
            res.close();
            st.close();
            con.close();
```

Fermeture de tout

```
        } catch (ClassNotFoundException e){
            System.err.println("Could not load JDBC driver");
            System.out.println("Exception: " + e);
            e.printStackTrace();
        }
```

```
        } catch (SQLException ex){
            System.err.println("SQLException information");
            while(ex!=null) {
                System.err.println ("Error msg: " + ex.getMessage());
                System.err.println ("SQLSTATE: " + ex.getSQLState());
                System.err.println ("Error code: " + ex.getErrorCode());
                ex.printStackTrace();
                ex = ex.getNextException(); // For drivers that support chained exceptions
            }
        }
```

Gestion des erreurs

```
    }
```

```
}
```

```

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class ExecuteOneSimpleQuery2 {

    static String DB_SERVER = "jdbc:mysql://mysql.loria.fr:3306/";
    static String DB = "coulet";
    static String DRIVER = "com.mysql.jdbc.Driver";
    static String USER_NAME = "couletadm";
    static String USER_PSWD = "blabla";

    public static void main(String[] args) {
        try{
            // Set up the connection
            Class.forName(DRIVER);
            Connection con = DriverManager.getConnection(DB_SERVER+DB, USER_NAME, USER_PSWD);
            // Build and execute the SQL query
            String myQuery = "SELECT * " +
                "FROM Etudiant " +
                "WHERE prenom='Clement'";
            Statement st = con.createStatement();
            ResultSet res = st.executeQuery(myQuery);

            while (res.next()) {
                // if we know attributes names
                int id = res.getInt("id");
                String prenom = res.getString("prenom");
                String nom = res.getString("nom");

                System.out.println(" ");
                System.out.println("Etudiant:"+id+"\n\tNom: "+nom+" "+prenom);
            }
            res.close();
            st.close();
            con.close();
        }
        catch (ClassNotFoundException e){
            System.err.println("Could not load JDBC driver");
            System.out.println("Exception: " + e);
            e.printStackTrace();
        }
        catch(SQLException ex){
            System.err.println("SQLException information");
            while(ex!=null) {
                System.err.println ("Error msg: " + ex.getMessage());
                System.err.println ("SQLSTATE: " + ex.getSQLState());
                System.err.println ("Error code: " + ex.getErrorCode());
                ex.printStackTrace();
                ex = ex.getNextException(); // For drivers that support chained exceptions
            }
        }
    }
}

```

```

Etudiant:35
    Nom: FOUQUERAY Clement

Etudiant:66
    Nom: PEYRESAUBES Clement

Etudiant:71
    Nom: RIVELLINI Clement

```

```

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class ExecuteOneSimpleQuery2 {

    static String DB_SERVER = "jdbc:mysql://mysql.loria.fr:3306/";
    static String DB = "coulet";
    static String DRIVER = "com.mysql.jdbc.Driver";
    static String USER_NAME = "couletadm";
    static String USER_PSWD = "blabla";

    public static void main(String[] args) {
        try{
            // Set up the connection
            Class.forName(DRIVER);
            Connection con = DriverManager.getConnection(DB_SERVER+DB, USER_NAME, USER_PSWD);
            // Build and execute the SQL query
            String myQuery = "SELECT * " +
                "FROM Etudiant " +
                "WHERE prenom='Clement'";
            Statement st = con.createStatement();
            ResultSet res = st.executeQuery(myQuery);

            while (res.next()) {
                // if we know attributes names
                int id = res.getInt("id");
                String prenom = res.getString("prenom");
                String nom = res.getString("nom");

                System.out.println(" ");
                System.out.println("Etudiant:"+id+"\n\tNom: "+nom+" "+prenom);
            }
            res.close();
            st.close();
            con.close();
        }
        catch (ClassNotFoundException e){
            System.err.println("Could not load JDBC driver");
            System.out.println("Exception: " + e);
            e.printStackTrace();
        }
        catch(SQLException ex){
            System.err.println("SQLException information");
            while(ex!=null) {
                System.err.println ("Error msg: " + ex.getMessage());
                System.err.println ("SQLSTATE: " + ex.getSQLState());
                System.err.println ("Error code: " + ex.getErrorCode());
                ex.printStackTrace();
                ex = ex.getNextException(); // For drivers that support chained exceptions
            }
        }
    }
}

```

```

while (res.next()) {
    // if we don't know attributes names, we have to know their positions
    int id = res.getInt(1);
    String prenom = res.getString(2);
    String nom = res.getString(3);

    System.out.println(" ");
    System.out.println("Etudiant:"+id+"\n\tNom: "+nom+" "+prenom);
}

```

```

Etudiant:35
Nom: FOUQUERAY Clement

Etudiant:66
Nom: PEYRESAUBES Clement

Etudiant:71
Nom: RIVELLINI Clement

```

```

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class ExecuteOneSimpleQuery2 {

    static String DB_SERVER = "jdbc:mysql://mysql.loria.fr:3306/";
    static String DB = "coulet";
    static String DRIVER = "com.mysql.jdbc.Driver";
    static String USER_NAME = "couletadm";
    static String USER_PSWD = "blabla";

    public static void main(String[] args) {
        try{
            // Set up the connection
            Class.forName(DRIVER);
            Connection con = DriverManager.getConnection(DB_SERVER+DB, USER_NAME, USER_PSWD);
            // Build and execute the SQL query
            String myQuery = "SELECT * " +
                "FROM Etudiant " +
                "WHERE prenom='Clement'";
            Statement st = con.createStatement();
            ResultSet res = st.executeQuery(myQuery);

            while (res.next()) {
                // if we know attributes names
                int id = res.getInt("id");
                String prenom = res.getString("prenom");
                String nom = res.getString("nom");

                System.out.println(" ");
                System.out.println("Etudiant:"+id+"\n\tNom: "+nom+" "+prenom);
            }
            res.close();
            st.close();
            con.close();
        }
        catch (ClassNotFoundException e){
            System.err.println("Could not load JDBC driver");
            System.out.println("Exception: " + e);
            e.printStackTrace();
        }
        catch(SQLException ex){
            System.err.println("SQLException information");
            while(ex!=null) {
                System.err.println ("Error msg: " + ex.getMessage());
                System.err.println ("SQLSTATE: " + ex.getSQLState());
                System.err.println ("Error code: " + ex.getErrorCode());
                ex.printStackTrace();
                ex = ex.getNextException(); // For drivers that support chained exceptions
            }
        }
    }
}

```

```

// if you don't know anything about the content of the ResultSet
ResultSetMetaData rsmd = res.getMetaData();
int nbOfColumns = rsmd.getColumnCount();
HashMap<String, String> columnMap = new HashMap<String, String>();
for(int c=1;c<=nbOfColumns;c++){
    columnMap.put(rsmd.getColumnLabel(c), rsmd.getColumnType(c)+"");
}
while (res.next()) {
    for(String columnName : columnMap.keySet()){
        if(columnMap.get(columnName).equals(java.sql.Types.INTEGER+"")){
            System.out.print(columnName+"="+res.getInt(columnName)+" ");
        }
        if(columnMap.get(columnName).equals(java.sql.Types.VARCHAR+"")){
            System.out.print(columnName+"="+res.getString(columnName)+" ");
        }
    }
    System.out.println("");
}

```

```

prenom=Clement id=35 nom=FOUQUERAY
prenom=Clement id=66 nom=PEYRESAUBES
prenom=Clement id=71 nom=RIVELLINI

```

I. Accéder et extraire des données

② dans un système de gestion de base de données

b) select

- Les procédures stockées (simples)
 - on enregistre dans la BD la procédure
 - on lance son exécution à partir du code java
 - disponible uniquement si \geq MySQL **5**

```
CREATE PROCEDURE getEtudiantClement()  
BEGIN  
    SELECT *  
    FROM Etudiant  
    WHERE prenom='Clement';  
END;
```

```

CREATE PROCEDURE getEtudiantClement()
BEGIN
    SELECT *
    FROM Etudiant
    WHERE prenom='Clement';
END;

```

```

import java.sql.Connection;

public class ExecuteOneSimpleProcedure {

    static String DB_SERVER = "jdbc:mysql://mysql.loria.fr:3306/";
    static String DB = "coulet";
    static String DRIVER = "com.mysql.jdbc.Driver";
    static String USER_NAME = "couletadm";
    static String USER_PSWD = "blabla";

    public static void main(String[] args) {
        try{
            // Set up the connection
            Class.forName(DRIVER);
            Connection con = DriverManager.getConnection(DB_SERVER+DB, USER_NAME, USER_PSWD);
            // Build and execute the SQL query
            ResultSet res = null;

            CallableStatement cStmt = con.prepareCall("{call getEtudiantClement()}");
            boolean hadResult = cStmt.execute();
            if(hadResult){
                res = cStmt.getResultSet();
                while (res.next()) {
                    String id = res.getString("id");
                    String prenom = res.getString("prenom");
                    String nom = res.getString("nom");

                    System.out.println(" ");
                    System.out.println("Etudiant:"+id);
                    System.out.println("\tNom: "+nom+" "+prenom);
                }
            }else{
                System.out.println("La procédure stockée ne retourne aucun n-uplets.");
            }
            res.close();
            cStmt.close();
            con.close();
        }
        catch (ClassNotFoundException e){
            System.err.println("Could not load JDBC driver");
            System.out.println("Exception: " + e);
            e.printStackTrace();
        }
        catch(SQLException ex){
            System.err.println("SQLException information");
            while(ex!=null) {
                System.err.println ("Error msg: " + ex.getMessage());
                System.err.println ("SQLSTATE: " + ex.getSQLState());
                System.err.println ("Error code: " + ex.getErrorCode());
                ex.printStackTrace();
                ex = ex.getNextException(); // For drivers that support chained exceptions
            }
        }
    }
}

```

I. Accéder et extraire des données

② dans un système de gestion de base de données

b) `select`

- Les procédures stockées (simples)

- intérêt

- centralisées, sécurisées...et
 - rapidité

I. Accéder et extraire des données

② dans un système de gestion de base de données

b) select

• Les procédures stockées avec des paramètres

```
CREATE PROCEDURE getEtudiantParam(IN inputAttribute VARCHAR(80),
  IN inputValue VARCHAR(80))
BEGIN
  DECLARE inputAttribute VARCHAR(80);
  DECLARE inputValue VARCHAR(80);

  IF inputAttribute = 'nom' THEN
    SELECT * FROM Etudiant WHERE nom=inputValue ;
  END IF;

  IF inputAttribute = 'prenom' THEN
    SELECT * FROM Etudiant WHERE prenom=inputValue ;
  END IF;

  IF inputAttribute = 'id' THEN
    SELECT * FROM Etudiant WHERE id=inputValue ;
  END IF;
END;
```

```

CREATE PROCEDURE getEtudiantParam(IN inputAttribute
VARCHAR(80), (IN inputValue VARCHAR(80))
BEGIN
    DECLARE inputAttribute VARCHAR(80);
    DECLARE inputValue VARCHAR(80);

    IF inputAttribute = 'nom' THEN
        SELECT * FROM Etudiant WHERE nom=inputValue ;
    END IF;

    IF inputAttribute = 'prenom' THEN
        SELECT * FROM Etudiant WHERE prenom=inputValue ;
    END IF;

    IF inputAttribute = 'id' THEN
        SELECT * FROM Etudiant WHERE id=inputValue ;
    END IF;
END;

```

```

public static void main(String[] args) {
    try{
        // Set up the connection
        Class.forName(DRIVER);
        Connection con = DriverManager.getConnection(DB_SERVER+DB, USER_NAME, USER_PSWD);
        // Build and execute the SQL query
        ResultSet res = null;

        CallableStatement cStmt = con.prepareCall("{call ParamSp(?,?)}");

        System.out.println(" ");
        cStmt.setString(1, "prenom");
        cStmt.setString(2, "Clement");
        boolean hadResult = cStmt.execute();
        if(hadResult){
            res = cStmt.getResultSet();
            while (res.next()) {
                String prenom = res.getString("prenom");
                String nom = res.getString("nom");
                System.out.println("\tNom: "+nom+" "+prenom);
            }
        }

        System.out.println(" ");
        cStmt.setString(1, "nom");
        cStmt.setString(2, "DIEZ");
        hadResult = cStmt.execute();
        if(hadResult){
            res = cStmt.getResultSet();
            while (res.next()) {
                String prenom = res.getString("prenom");
                String nom = res.getString("nom");
                System.out.println("\tNom: "+nom+" "+prenom);
            }
        }
        res.close();
        cStmt.close();
        con.close();
    }
    catch (ClassNotFoundException e){
        System.err.println("Could not load JDBC driver");
        System.out.println("Exception: " + e);
        e.printStackTrace();
    }
    catch(SQLException ex){
        System.err.println("SQLException information");
        while(ex!=null) {
            System.err.println ("Error msg: " + ex.getMessage());
            System.err.println ("SQLSTATE: " + ex.getSQLState());
            System.err.println ("Error code: " + ex.getErrorCode());
            ex.printStackTrace();
            ex = ex.getNextException(); // For drivers that support chained exceptions
        }
    }
}
}

```

I. Accéder et extraire des données

② dans un système de gestion de base de données

b) `select`

- Les PreparedStatement

- correspond à un "pattern" de requête que l'on veut exécuter souvent lors d'une même connexion
- on ouvre un PreparedStatement dans la BD
- on envoie ensuite simplement les paramètres manquants (?)
- intérêt : plus rapide que de créer plusieurs Statement car le pattern est déjà envoyé au SGBD et précompilé

```
SELECT id, nom, prenom  
FROM Etudiant  
WHERE nom=?;
```

```
SELECT id, nom, prenom
FROM Etudiant
WHERE nom=?;
```

```
public class ExecuteOnePreparedStt {

    static String DB_SERVER = "jdbc:mysql://mysql.loria.fr:3306/";
    static String DB        = "coulet";
    static String DRIVER    = "com.mysql.jdbc.Driver";
    static String USER_NAME = "couletadm";
    static String USER_PSWD = "blabla";

    public static void main(String[] args) {
        try{
            // Set up the connection
            Class.forName(DRIVER);
            Connection con = DriverManager.getConnection(DB_SERVER+DB, USER_NAME, USER_PSWD);
            String preparedQuery = "SELECT id, nom, prenom " +
                                   "FROM Etudiant " +
                                   "WHERE nom=?";
            PreparedStatement pStmt = con.prepareStatement(preparedQuery);

            HashSet<String> noms = new HashSet<String>();
            noms = getListNoms(); // une méthode qui retourne une liste de noms d'étudiants

            for(String nom:noms){
                pStmt.setString(1,nom);
                pStmt.executeQuery();
                ResultSet res = pStmt.getResultSet();
                while (res.next()) {
                    String id    = res.getString("id");
                    String prenom = res.getString("prenom");
                    System.out.println(" ");
                    System.out.println("Etudiant:"+id);
                    System.out.println("\tNom: "+nom+" "+prenom);
                }
                res.close();
            }
            pStmt.close();
            con.close();

        }catch (ClassNotFoundException e){
```

```
SELECT id, nom, prenom
FROM Etudiant
WHERE nom=?;
```

```
public class ExecuteOnePreparedStt {

    static String DB_SERVER = "jdbc:mysql://mysql.loria.fr:3306/";
    static String DB       = "coulet";
    static String DRIVER   = "com.mysql.jdbc.Driver";
    static String USER_NAME = "couletadm";
    static String USER_PSWD = "blabla";

    public static void main(String[] args) {
        try{
            // Set up the connection
            Class.forName(DRIVER);
            Connection con = DriverManager.getConnection(DB_SERVER+DB, USER_NAME, USER_PSWD);
            String preparedQuery = "SELECT id, nom, prenom " +
                "FROM Etudiant " +
                "WHERE nom=?";

            PreparedStatement pStmt = con.prepareStatement(preparedQuery);

            HashSet<String> noms = new HashSet<String>();
            noms = getListNoms(); // une méthode qui retourne une liste de noms d'étudiants

            for(String nom:noms){
                pStmt.setString(1,nom);
                pStmt.executeQuery();
                ResultSet res = pStmt.getResultSet();
                while (res.next()) {
                    String id    = res.getString("id");
                    String prenom = res.getString("prenom");
                    System.out.println(" ");
                    System.out.println("Etudiant:"+id);
                    System.out.println("\tNom: "+nom+" "+prenom);
                }
                res.close();
            }
            pStmt.close();
            con.close();
        }catch (ClassNotFoundException e){
```

Si

```
SELECT id, nom, prenom
FROM Etudiant
WHERE nom=?
AND prenom=?;
```

alors

```
pStmt.setString(1,nom)
pStmt.setString(2,prenom)
```

I. Accéder et extraire des données

① dans un système de gestion de base de données

a) java

- Un PreparedStatement est valable le temps d'une connexion
- Si les requêtes sont longues...
 - si l'on enchaîne des requêtes longues lors d'une même connexion,
 - la connexion se ferme automatiquement au bout d'un certain temps,
 - il faut donc détecter la fermeture et ouvrir une nouvelle connexion
 - Cela peut impliquer de devoir ré-ouvrir les PreparedStatement

```
private void openMyPreparedStatement(){
    StringBuffer queryb = new StringBuffer();
    queryb.append("SELECT id, nom, prenom ");
    queryb.append("FROM Etudiant ");
    queryb.append("WHERE prenom=?");
    pstmt = con.prepareStatement(queryb.toString());
}

public HashSet<Student> printStudentByFirstName(String prenom){
    HashSet<Student> studentList = new HashSet<Student>;
    try {
        pstmt.setString(1, prenom);
        ResultSet rSet = pstmt.executeQuery();
        while (rSet.next()) {
            studentList.add(new Student(rSet.getString("id"),rSet.getString("nom"),rSet.getString("prenom")));
        }
        rSet.close();
    }
    catch (MySQLNonTransientConnectionException e) {
        if(con.isClosed()) {
            con = DriverManager.getConnection(DB_SERVER+DB, USER_NAME, USER_PSWD);
            this.openMyPreparedStatement();
            return this.printStudentByFirstName(prenom);
        }
    }
    catch (SQLException e) {
        e.printStackTrace();
        System.out.println("** PROBLEM ** Cannot get student from table Etudiant for prenom="+prenom+". Empty set returned.");
    }
    return studentList;
}
```

```
private void openMyPreparedStatement(){
    StringBuffer queryb = new StringBuffer();
    queryb.append("SELECT id, nom, prenom ");
    queryb.append("FROM Etudiant ");
    queryb.append("WHERE prenom=?");
    pstmt = con.prepareStatement(queryb.toString());
}

```

```
public HashSet<Student> printStudentByFirstName(String prenom){
    HashSet<Student> studentList = new HashSet<Student>;
    try {
        pstmt.setString(1, prenom);
        ResultSet rSet = pstmt.executeQuery();
        while (rSet.next()) {
            studentList.add(new Student(rSet.getString("id"),rSet.getString("nom"),rSet.getString("prenom")));
        }
        rSet.close();
    }
}

```

```
catch (MySQLNonTransientConnectionException e) {
    if(con.isClosed()) {
        con = DriverManager.getConnection(DB_SERVER+DB, USER, PASSWORD);
        this.openMyPreparedStatement();
        return this.printStudentByFirstName(prenom);
    }
}
catch (SQLException e) {
    e.printStackTrace();
    System.out.println("** PROBLEM ** Cannot get student from table Etudiant for prenom="+prenom+". Empty set returned.");
}
return studentList;
}

```

Si lorsque l'on veut exécuter une nouvelle requête cela échoue car la connexion est fermée on ré-ouvre la connexion on relance la méthode de déclaration du PreparedStatement on relance la méthode elle-même

I. Accéder et extraire des données

② dans un système de gestion de base de données

a) java

- InnoDB vs MyISAM vs MEMORY

- Trois types de "storage engine"

- InnoDB par défaut dans MySQL

- sinon CREATE TABLE [...] ENGINE=MyISAM

1. InnoDB stocke physiquement les lignes dans l'ordre de la clé primaire vs MyISAM les stocke dans l'ordre de leur ajout – par conséquent InnoDB retourne plus vite les lignes avec moins de besoin de mémoire.

2. La taille des tables InnoDB < MyISAM

3. Une BD de table InnoDB ne peut pas être copié-collé d'un serveur à un autre (si le serveur tourne)

4. MyISAM permet de créer des *indexes "full text"* c'ad des indexes qui permettent des recherches rapides sur le contenu de champs texte longs.

5. et plus...

I. Accéder et extraire des données

② dans un système de gestion de base de données

- Quelques éléments pour améliorer l'accès au contenu de bases de données ^{a) java}
 - utiliser les PreparedStatements
 - utiliser le bon type de table (InnoDB vs. MyISAM vs.Memory)
 - limiter le nombre de connexions (en parallèle et successives)
 - exécuter les requêtes directement sur la machine du serveur de BD
 - utiliser des types de données et tailles de champs adaptés
 - indexer les bons champs des tables
 - utiliser des clés numériques
 - faire les jointures sur des clés numériques
 - créer des tables d'index pour les jointures
 - partition horizontale / partition verticale

I. Accéder et extraire des données

② à partir d'un service REST

a) Client REST

- Service Web
 - ensemble de fonctionnalités exposées sur Internet
- REST est un type de service Web
 - REpresentational State Transfer
 - Les services REST exposent leurs fonctionnalités comme un ensemble de ressources (URI) identifiables et accessibles par la syntaxe et la sémantique du protocole HTTP. Les Services Web de type REST sont donc basés sur l'architecture du web et ses standards de base : HTTP et URI
 - client/serveur sont remplacé par agent/ressource

I. Accéder et extraire des données

② à partir d'un service REST

• Exemple de serveur REST

a) Client REST

– le NCBO et son serveur d'ontologies

The screenshot shows the NCBO Ontology Browser interface. On the left is a tree view of the ontology hierarchy, with 'Human' selected. On the right is a detailed view of the 'Human' concept, showing its ID, full URI, synonyms, definitions, code, and other metadata.

ID:	Human
Full Id:	http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Human
Synonyms:	Human, General Homo sapiens
Definitions:	The bipedal primate mammal, Homo sapiens; belonging to man or mankind; pertaining to man or to the race of man; use of man as experimental subject or unit of analysis in research.
Code:	C14225
Legacy Concept Name:	Human
Ncbi Taxon:	9606
Preferred Name:	Human
Label:	Human
Semantic Type:	Human
Umls Cui:	C0086418
Sub Class Of:	Hominidae

GUI

I. Accéder et extraire des données

② à partir d'un service REST

a) Client REST

- 2 modes d'interrogation (comme http)

- GET <http://rest.bioontology.org/bioportal/virtual/rootpath/1032/Human?email=example@example.org>

- POST : les paramètres ne sont pas visibles dans l'URL

```
import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.methods.GetMethod;

public class GetPathGet2 {

    public static final String PROD_URL = "http://rest.bioontology.org/bioportal/virtual/rootpath";
    public static final String ONTOLOGY_ID = "/1032";
    public static final String TERM_NAME = "/Human";
    public static final String PARAM1 = "?email=example@example.org";
    //public static final String PARAM2 = "&pagenum=";
    //public static final String PAGE_SIZE = "50";

    public String call(){
        String answer="";

        try {
            HttpClient client = new HttpClient();
            GetMethod method = new GetMethod(PROD_URL+ONTOLOGY_ID+TERM_NAME+PARAM1);

            // Execute the GET method
            int statusCode = client.executeMethod(method);
            if( statusCode != -1 ) {
                answer = method.getResponseBodyAsString();
                method.releaseConnection();
            }
        }
        catch( Exception e ) {
            e.printStackTrace();
        }

        return answer;
    }

    public static void main( String[] args ) {
        System.out.println("***** NCBO all concept CLIENT TEST ***** \n");

        GetPathGet2 myClient = new GetPathGet2();
        System.out.println(myClient.call());
    }
}
```

```
package esial.gmd.tp2;
```

```
import org.apache.commons.httpclient.HttpClient;
```

```
import org.apache.commons.httpclient.methods.PostMethod;
```

```
/**
```

```
 * @author Adrien Coulet
```

```
 * @date 2010
```

```
 */
```

```
public class GetAnnotPost {
```

```
    public static final String PROD_URL = "http://rest.bioontology.org/obs/annotator?email=example@example.org";
```

```
    private static String text = "Melanoma is a malignant tumor of melanocytes " +  
        "which are found predominantly in skin but also in the bowel and the eye";
```

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

```
        System.out.println("***** NCBO REST CLIENT TEST ***** \n");
```

```
        try {
```

```
            HttpClient client = new HttpClient();
```

```
            PostMethod method = new PostMethod(PROD_URL);
```

```
            // Configure the form parameters
```

```
            method.addParameter("longestOnly", "true");
```

```
            method.addParameter("scored", "true");
```

```
            method.addParameter("mappingTypes", "null");
```

```
            method.addParameter("textToAnnotate", text);
```

```
            // Execute the POST method
```

```
            int statusCode = client.executeMethod(method);
```

```
            if( statusCode != -1 ) {
```

```
                String contents = method.getResponseBodyAsString();
```

```
                method.releaseConnection();
```

```
                System.out.println(contents);
```

```
            }
```

```
        }
```

```
        catch( Exception e ) {
```

```
            e.printStackTrace();
```

```
        }
```

```
    }
```

```
}
```

I. Accéder et extraire des données

② à partir d'un service REST

b) Serveur REST

- Déployer un Serveur REST :
Non abordé dans ce module.

I. Accéder et extraire des données

② à partir d'un service REST

c) Parser le XML

- Rappels

- les API SAX et DOM

- DOM : chargement de l'intégralité de l'arbre XML en mémoire
 - *SAX scales* :
 - SAX traite l'arborescence élément par élément (un peu comme awk finalement !)
 - SAX est dirigé par l'utilisateur (notion d'événement)

```

import java.io.*;
import org.jdom.*;
import org.jdom.input.*;
import org.jdom.filter.*;
import java.util.List;
import java.util.Iterator;

public class JDOM2
{
    static org.jdom.Document document;
    static Element racine;

    public static void main(String[] args)
    {
        //On crée une instance de SAXBuilder
        SAXBuilder sxb = new SAXBuilder();
        try
        {
            //On crée un nouveau document JDOM avec en argument le fichier XML
            //Le parsing est terminé ;)
            document = sxb.build(new File("Exercice2.xml"));
        }
        catch(Exception e){}

        //On initialise un nouvel élément racine avec l'élément racine du document.
        racine = document.getRootElement();

        //Méthode définie dans la partie 3.2. de cet article
        afficheALL();
    }
    static void afficheALL()
    {
        //On crée une List contenant tous les noeuds "etudiant" de l'Element racine
        List listEtudiants = racine.getChildren("etudiant");

        //On crée un Iterator sur notre liste
        Iterator i = listEtudiants.iterator();
        while(i.hasNext())
        {
            //On recrée l'Element courant à chaque tour de boucle afin de
            //pouvoir utiliser les méthodes propres aux Element comme :
            //selectionner un noeud fils, modifier du texte, etc...
            Element courant = (Element)i.next();
            //On affiche le nom de l'element courant
            System.out.println(courant.getChild("nom").getText());
        }
    }
}

```

```

<?xml version="1.0" encoding="UTF-8"?>
  <personnes>
    <etudiant classe="P2">
      <nom>Cynoc</nom>
      <prenoms>
        <prenom>Nicolas</prenom>
        <prenom>Laurent</prenom>
      </prenoms>
    </etudiant>
    <etudiant classe="P1">
      <nom>Superwoman</nom>
    </etudiant>
    <etudiant classe="P1">
      <nom>Don Corleone</nom>
    </etudiant>
  </personnes>

```

I. Accéder et extraire des données

3

avec JSON

```
{
  "firstName": "Adrien",
  "lastName" : "Coulet",
  "age"       : 32,
  "address"   :
  {
    "streetAddress": "625 Rhodes Island Street",
    "city"          : "San Francisco",
    "state"         : "CA",
    "postalCode"   : "94107"
  },
  "phoneNumber":
  [
    {
      "type" : "home",
      "number": "650 344-4505"
    },
    {
      "type" : "work",
      "number": "650 283-4910"
    }
  ]
}
```

I. Accéder et extraire des données

3

avec JSON

- format pour l'échange de données
- moins verbeux que l'XML
- plus facile à lire que l'XML

I. Accéder et extraire des données

3

avec SQLite



- SQLite
 - pas d'installation
 - pas de configuration
 - pas de serveur
 - un seul fichier avec les données
 - copiable-collable

I. Accéder et extraire des données

3

avec SQLite

- SQLite
 - limites :
 - accès concurrent
 - gros volumes de données
 - architecture client - serveur
 - simple et bon pour
 - application locale
 - petit site Web
 - logiciel embarqué
 - testing

I. Accéder et extraire des données

3

avec SQLite

- C'est une bibliothèque C
- qui implémente un moteur de base de données
- accessible par le langage SQL

N★SQL

N★SQL

- *Not only SQL* but more *NoREL*
- "The term covers a wide range of technologies and data architectures and priorities"
- "it represents as much a movement or a school of thought as it does any particular technology"
- databases (diff from RDBMS) mostly addressing some of the points:
 - being non relational
 - distributed
 - open-source and
 - horizontally scalable

N★SQL

- Various types
 - key-value store
 - document store
 - wide column store
 - graph store

- illustration in Harry Kauhanen slides