# Bases de Données – Licence 1 – Semestre 2

Application de base de données

## 1 Objectifs

Dans ce TP, nous allons développer une application Java exploitant des données issues d'une base de données MySQL. Nous instancierons une base de données MySQL hébergée sur un serveur accessible via une URL et quelques classes Java. Celles-ci utiliseront une bibliothèque Java spécifique fournie pour communiquer avec la base.

## 2 Avant Propos

Pour accéder à la base de données, récupérez les fichiers :
- BD.class pour la version de 8 de Java ou BD.class pour une version plus récente de Java.
- mysql-connector-java.jar.

et placez les dans votre répertoire de travail, dans le dossier où vous allez écrire vos classes Java.

Ces fichiers vous permettront principalement :
- d'établir, exploiter puis enfin fermer les connexions aux bases de données (au moins une !)
- de manipuler les résultats des requêtes : ceux-ci seront vus comme des tableaux ; ces tableaux pourront être parcourus ligne-par-ligne (enregistrement-par-enregistrement) et dans chaque ligne on pourra récupérer la valeur d'un attribut.

## 3 Documentation sur la bibliothèque

### 3.1 Connexion à la base de données.

- `int BD.ouvrirConnexion(String adresse, String bd, String login, String password)`. Ouvre une connexion à la base de données nommée `bd` en se connectant au serveur défini pr `adresse`, avec le nom d'utilisateur `login` et le mot de passe `password`.
  Cette fonction renvoie un entier positif ou nul identifiant la connexion ouverte (à réutiliser par la suite). En cas d'erreur, la valeur `-1` est renvoyée.
- `void BD.fermerConnexion(int connexion)`. Ferme la connexion identifiée par `connexion` précédemment ouverte.

### 3.2 Exécution de requêtes et traitement des résultats

- `int BD.executerSelect(int connexion, String sql)`. Exécute une requête de type `SELECT` sur la base de données. Cette fonction renvoie un entier positif ou nul identifiant le résultat associé à la requête. En cas d'erreur, la valeur -1 est renvoyée.
- `int BD.executerUpdate(int connexion, String sql)`. Exécute une requête de type `INSERT`, `UPDATE` ou `DELETE` sur la base de données. Cette fonction renvoie un entier positif ou nul indiquant le nombre d'enregistrements impactés par l'exécution de la requête.
  Si la requête réalise une insertion (`INSERT`) d'un enregistrement dont la clef serait générée par le moteur SQL (auto-incrémentée par exemple), cette fonction renvoie le numéro automatique généré pour le nouvel enregistrement. En cas d'erreur, la valeur -1 est renvoyée.

— `boolean BD.suivant(int res)`. Passe à l'enregistrement suivant pour le résultat `res`. Cette fonction renvoie true si l'enregistrement suivant a pu être atteint (s'il existe), false en cas d'erreur.
— `boolean BD.reinitialiser(int res)`. Ré-initialise le parcours des enregistrements en se plaçant avant le premier enregistrement du résultat `res`. Cette fonction renvoie `true` si la réinitialisation a pu être effectuée, `false` sinon.
— `void BD.fermerResultat(int res)`. Libère la mémoire du résultat `res`, et supprime ce résultat ; son identifiant ne sera plus réutilisable par la suite (attention, il pourra être réattribué par la suite).
— `String BD.attributString(int res, String att)`. Renvoie la valeur de l'attribut `att` pour l'enregistrement courant du résultat `res` sous la forme d'une chaîne de caractères.
— `int BD.attributInt(int res, String att)`. Renvoie la valeur de l'attribut `att` pour l'enregistrement courant du résultat `res` sous la forme d'un entier.
— `long BD.attributLong(int res, String att)`. Renvoie la valeur de l'attribut `att` pour l'enregistrement courant du résultat `res` sous la forme d'un entier `long`.

### 3.3 Manipulation de dates

— `long BD.maintenant()`. Renvoie un entier long représentant la date du jour sous la forme du nombre de millisecondes écoulées depuis le 1er janvier 1970 à minuit.
— `long BD.date(int jour, int mois, int annee, int heures, int minutes, int secondes)`. Renvoie un entier long représentant la date spécifiée par les valeurs passées en paramètres.
— `int BD.jour(long d)`. Renvoie un entier représentant le numéro du jour (1-31) à partir d'une date `d` spécifiée par un entier long.
— `int BD.mois(long d)`. Renvoie un entier représentant le numéro du mois (1-12) à partir d'une date `d` spécifiée par un entier long.
— `int BD.annee(long d)`. Renvoie un entier représentant une année à partir d'une date `d` spécifiée par un entier long.
— `int BD.heures(long d)`. Renvoie un entier représentant les heures (0-23) à partir d'une date `d` spécifiée par un entier long.
— `int BD.minutes(long d)`. Renvoie un entier représentant les minutes (0-59) à partir d'une date `d` spécifiée par un entier long.
— `int BD.secondes(long d)`. Renvoie un entier représentant les secondes (0-59) à partir d'une date `d` spécifiée par un entier long.

### 3.4 Temporisation

— `void BD.pause(int m)`. Met le programme en pause pour une durée de `m` millisecondes.

### 3.5 Un exemple

La figure 1 donne un exemple de connexion à une base MySQL hébergée sur un serveur distant. Les identifiants de connexion seront donnés en TP. Ensuite est réalisée une requête avec jointures et projection. Toutes les lignes du résultat de la requête sont affichées les unes après les autres.

Pour exécuter ce code code il est d'abord nécessaire de le compiler

```
javac Cabinet.java
```

Si vous avez un interpréteur Java 8, pour l'executer il suffit de demander

```
java Cabinet
```

Si vous avez un interpréteur Java dont la version est plus récente, pour l'executer il suffit de demander

— Si vous êtes sur Windows : `java -cp .;mysql-connector-java.jar Cabinet`
— Si vous êtes sur Linux : `java -cp .:mysql-connector-java.jar Cabinet`

```java
public class Cabinet {
       public static String adresse = "...";
       public static String bd = "...";
       public static  String login = "...";
       public static String password = "...";

    public static void main(String[] args) {
        int connexion = BD.ouvrirConnexion(adresse, bd, login,password);
        // création de la requête
        String sql = "SELECT PATIENT.Nom, MEDECIN.Nom, NumCons FROM CONSULTATION, MEDECIN, PATIENT WHERE"
            +" CONSULTATION.Medecin = MEDECIN.Matricule AND CONSULTATION.Patient = NumSecu";
        // envoi de la requête
        int res = BD.executerSelect(connexion, sql);
        // parcours du résultat (ligne par ligne)
        while (BD.suivant(res)) {
            int numCons = BD.attributInt(res,"NumCons");
            String nomMedecin = BD.attributString(res,"MEDECIN.Nom");
            String nomPatient = BD.attributString(res,"PATIENT.Nom");
            System.out.println(""+numCons+ ": "+ nomPatient +" ("+nomMedecin+")");
        }
        BD.fermerResultat(res);
        BD.fermerConnexion(connexion);
    }
}
```

FIGURE 1 – Code Java d'interrogation et d'affichage d'une BD MySQL distante

# 4 Mise en application

## Exercice – Déploiement de l'exemple

**Question 1.1.** Faire en sorte que le code java proposé en figure 1 puisse être exécuté.

**Question 1.2.** En modifiant le programme précédent, pour chaque patient, afficher le numéro de sécurité sociale, son nom le nombre de consultations le concernant.

## Exercice – Insertion/Suppression de données

**Question 2.1.** Réalisez une application `Moi`, à partir de la classe `Moi.java` qui insère dans la base de données un patient avec votre nom, votre prénom et votre numéro de sécurité sociale dont vous modifierez quelques chiffres parmi les 6 derniers (sans la clef). Mémorisez ce numéro modifié *nss* pour une question ultérieure.

**Question 2.2.** Calculez le reste $r$ de la division de votre année de naissance par 3. Le médecin que vous êtes allé voir aujourd'hui est (126389,Hadley) si ce reste $r$ est 0, (526736,House) si c'est 1 et (943223, Foreman) sinon. Ce médecin vous a prescrit deux médicaments au choix parmi les trois et tel que le nombre de prise est compris entre 1 et 5. Insérez ces informations dans la base de données. On pourra constater que la méthode `BD.executerUpdate` retourne `NumCons`, clé de la table `CONSULTATION` et entier auto incrémenté. Ceci peut être utilisé ensuite dans la table `PRESCRIPTION`.

**Question 2.3.** Réalisez l'application Java `SupprimeMoi` qui supprime de la table `PATIENT` la personne identifiée par le numéro de sécurité sociale *nss*. Au moyen de l'interface PHPMyAdmin, constater la suppression de toutes les données relatives à *nss*. Pourquoi cela s'est-il propagé à `CONSULTATION` et à `PRESCRIPTION` ?