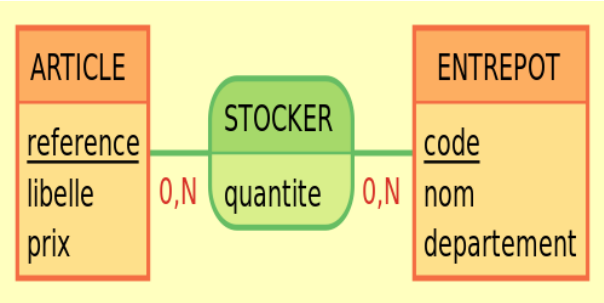
BD R3.07 : SQL dans un langage de programmation.

Feuille de TD/TP n°4

JDBC

Dans cette feuille nous allons utiliser la base de données `ENTREPOTS` déjà utilisée dans la feuille d'exercices précédente.




```sql
CREATE TABLE ARTICLE (
   reference INT(9),
   libelle VARCHAR(42),
   prix DECIMAL(6,2),
  PRIMARY KEY (reference)
) ;
```

```sql
CREATE TABLE ENTREPOT (
   code INT(9),
   nom VARCHAR(42),
   departement VARCHAR(42),
  PRIMARY KEY (code)
) ;

CREATE TABLE STOCKER (
   reference INT(9),
   code INT(9),
   quantite INT(5),
  PRIMARY KEY (reference, code)
) ;


ALTER TABLE STOCKER ADD FOREIGN KEY (code) REFERENCES
      ENTREPOT (code);

ALTER TABLE STOCKER ADD FOREIGN KEY (reference) REFERENCES
      ARTICLE (reference);
```

Exercice 1 *Connexion*

1. Récupérez le fichier `ConnexionMySQL.java` sur célène.
2. Créez une classe `TestJDBC` qui aura comme attribut `ConnexionMySQL connexion;` dont une instance permettra d'avoir une connexion sur votre base de donnée Mysql de l'IUT. Le nom du serveur Mysql à l'IUT est : `"servinfo-maria"`.
3. Testez !
   Pour compiler : `javac *.java`
   Pour exécuter : `java -cp .:/usr/share/java/mariadb-java-client.jar TestJDBC`

Exercice 2 *Préparation*

Ecrire les classes `Article`, `Entrepot` et `Stocker` comportant juste un constructeur, les observateurs (getters) et la méthode toString.

Exercice 3 *La classe `EntrepotBD` interrogation de la BD*

La classe contient un attribut de type ConnexionMySQL qui stocke la connexion avec laquelle nous allons travailler, la connexion sera passée en paramètre au constructeur. Par ailleurs cette classe devra implémenter les méthodes ci-dessous. Attention ! Testez bien vos méthodes dans la classe TestJDBC au fur et à mesure que vous les codez !

1. Ecrire une fonction pour obtenir le plus grand numéro utilisé pour identifier un article.
2. Ecrire une fonction qui prend en paramètre un numéro et retourne l'article de la base de données qui a ce numéro.

3. Ecrire une fonction pour obtenir l'article qui a le plus grand identifiant.
4. Ecrire une fonction qui retourne la liste des articles.
5. Ecrire une procédure pour afficher tous les entrepôts triés par département avec pour chaque département, le nombre d'entrepôts qu'a le département.
6. Ecrire une procédure qui pour un numéro d'article, affiche la liste des entrepôt disposant de cet article avec leur quantité disponible.
7. Ecrire une procédure qui pour un numéro d'entrepôt, affiche la liste des articles (avec leur quantité) disponibles dans l'entrepôt.
8. Ecrire une fonction qui retourne la valeur contenue dans un entrepôt donné.

Exercice 4 *La classe* `EntrepotBD` *mise à jour de la BD*

Nous allons voir maintenant des méthodes qui vont faire des mises à jour dans votre base de données.

1. Ecrire une fonction qui permet de stocker un nouvel article dans la table article ou de modifier le prix d'un article déjà existant. Par exemple si l'article est `a=(123, "tuile17x27", 3.55)` alors `majArticle(a)` modifiera le prix de l'article `123` s'il existe et qu'il correspond à `'tuile 17x27'`, si ne nom n'est pas `'tuile17x27'` afficher un message d'erreur, si l'article `123` n'est pas dans la base ajouter un nouvel article en prenant comme référence la plus grande des références présentes dans la base plus 1 à la place de `123`. La fonction retournera la référence de l'article créé ou modifié (-1 si erreur).
2. Ecrire une fonction qui ajoute un entrepôt dans la base de données. La fonction retournera l'identifiant de l'entrepôt ajouté, -1 si erreur. Attention aux contrainte : on ne veut pas avoir plusieurs entrepôts avec le même nom dans le même département, et on ne veut pas plus de trois entrepôts dans un même département.
3. Ecrire une fonction `entrerStock(refA int, codeE int, qte int)` qui augmente le stock de l'article `refA` dans l'entrepôt `codeE` de `qte`. Retourne la nouvelle quantité de l'article (-1) quand l'article ou l'entrepôt n'existe pas.
4. Ecrire une fonction `sortirStock(refA int, codeE int, qte int)` qui diminue le stock de l'article `refA` dans l'entrepôt `codeE` de `qte`. La quantité à sortir est limitée à la quantité présente. Retourne la quantité réellement sortie.

Exercice 5 *Bonus TP : En reprenant la base de données* `SALLES`

Proposer une application java avec un menu proposant différentes méthodes de visualisation ou de modification de votre BD.