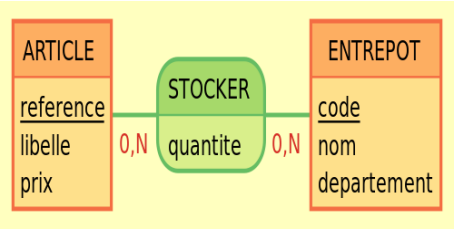
BD R3.07 : SQL dans un langage de programmation.

Feuille de TD/TP n°2

PL-SQL

Soient le MCD suivant :



et le script de création de la base de données `ENTREPOTS` :

```
-- CREATE DATABASE IF NOT EXISTS ENTREPOT DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
-- USE ENTREPOT;

drop table STOCKER;
drop table ARTICLE;
drop table ENTREPOT;

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

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 *TD : Requêtes paramétrées*

1. Ecrire une requête paramétrée pour avoir la liste des articles valant moins d'une certaine somme.
2. Ecrire une requête paramétrée pour avoir, en fonction du libellé d'un article et d'un département, les quantité en stock de cet article dans les entrepôts du département choisi.

Exercice 2 *TD :Procédures et fonctions*

1. Ecrire une fonction `maxRefArticle` qui retourne la plus grande référence utilisée pour identifier un article. (0 si la table article est vide).
2. Ecrire une fonction `deptEntrepot(codeEnt int)` qui retourne le département ou se trouve l'entrepôt de code `codeEnt`.
3. Ecrire une fonction `valEntrepot(codeEnt int)` qui retourne la valeur des marchandises contenues dans l'entrepôt `codeEnt`.

4. Ecrire une procédure pour afficher tous les entrepôts.
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. Ajouter à l'affichage précédent, la valeur contenue dans chaque entrepôt.
7. 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 `majArticle(123, 'tuile17x27', 3.55)` 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).
8. 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.
9. 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 3 *TD : Triggers*

Proposez une solution pour imposer à la base de données les nouvelles contraintes.

1. On ne veut pas avoir plusieurs entrepôts avec le même nom dans le même département.
2. On ne veut pas plus de trois entrepôts dans un même département.
3. A chaque fois que le stock d'un article est modifié (à la hausse ou à la baisse), on veut conserver une trace de la mise à jour.

Exercice 4 *TP : En reprenant la base de données* `SALLES`

1. Proposez des requêtes paramétrées.
2. Proposez des les procédures et fonctions stockées.
3. Proposez des triggers.