M1 MIAGE FC/FA – 2011/2012

# Bases de Données Avancées

octobre 2011

# TP PLSQL-JDBC

**Exercice 1 :** Une société commercialisant des produits chimiques informatise le stockage de ses produits. Chaque produit chimique possède un nom, un identifiant, et appartient à une catégorie, décrite par un nombre entier entre 1 et 50. La catégorie est une indication de dangerosité du produit. Un produit est stocké dans un container. Chaque container possède un identifiant, et un volume (en litre). Un container ne contient qu'un seul produit (pas de mélange). On suppose qu'un container est soit vide, soit rempli au maximum de ce que son volume permet.
La société possède des hangars, disposés en allées perpendiculaires, à la manière d'un quadrillage. Chaque hangar est donc identifié par son numéro de ligne et de colonne dans ce quadrillage, qui forme un rectangle de 5 lignes et 9 colonnes. Dans chaque hangar, on stocke des containers. Un container peut éventuellement ne pas être rangé dans un hangar. Un hangar possède une capacité, exprimée en litres, et représentant le volume maximal qui peut y être stocké. On mémorise aussi le volume réel stocké dans ce hangar, qui est la somme des volumes des containers rangés dans ce hangar (cet attribut est redondant puisqu'on retrouve cette information à partir de la table `CONTAINER`) Pour des raisons de sécurité, un hangar ne peut stocker que des produits de la même catégorie. Voici le MLD qui a été défini pour modéliser le stockage des produits :

```
HANGAR(lig, col, capacite, volume_reel)
CONTAINER(id_container, volume, id_produit, lig, col)
PRODUIT(id_produit, nom, categorie)
```

Vous trouverez sur le portail dans la rubrique document le fichier `stockage.sql` contenant les ordres de création des tables ainsi que des instructions d'insertion de données dans ces tables.

Question 1.1 : Définir un trigger qui met à jour automatiquement la colonne `volume_reel` en fonction des affectations faites dans la table `CONTAINER`. On ne s'occupera pas de l'éventuelle erreur déclenchée si la contrainte (`volume_reel` inférieur à `capacite`) n'est pas satisfaite.

Question 1.2 : Définir une vue `hangar_avec_categorie` qui donne la liste des hangars non vides avec la catégorie des produits qu'ils contiennent. Rappelons que dans un hangar, il n'y a qu'une seule catégorie de produits.
Le schéma de cette vue est `(lig, col, capacite, volume_reel, categorie)`.

Question 1.3 : Définir une vue `hangar_vide` qui donne la liste des hangars vides.
Le schéma de cette vue est `(lig, col, capacite)`.

On définit un paquetage `PAQ_STOCK` pour gérer le stockage des produits. Vous trouverez la spécification de ce paquetage dans `stockage.sql`.

Question 1.4 : Dans ce paquetage, on définit une fonction `identif_hangar` qui permet de fabriquer un identifiant de type nombre à partir des valeurs de `lig` et `col`. On peut par exemple prendre `10*lig + col`. L'avantage de cette fonction est qu'elle permet à une requête de renvoyer un seul nombre pour un hangar. On peut donc l'utiliser dans une requête SQL de la forme :

```
select max(identif_hangar(lig,col)) from hangar where ...
```

Ecrire le corps de `identif_hangar`.

Question 1.5 :
Dans ce paquetage, on définit également une procédure `affecter_container` qui affecte un container à un hangar. Cette procédure prend en paramètre un numéro de container.

- Si ce container n'existe pas, on déclenche `CONTAINER_INCONNU`.
- Si ce container est vide, on déclenche `CONTAINER_VIDE`.
- Si le container est déjà dans un hanger, la procédure met à `NULL` les colonnes `lig` et `col` pour le container. Elle recherche ensuite un hangar que l'on peut compléter avec le container. Si ce n'est pas possible, elle recherche un hangar vide dans lequel ranger le container. Rappelons qu'un hangar contient des containers dont les produits appartiennent tous à la même catégorie. Cette procédure déclenche l'exception `PAS_HANGAR_DISPO` si aucun hangar ne peut accueillir ce container.

Ecrire le corps de la procédure `affecter_container`.

**Exercice 2 :** *suite de l'exercice PL/SQL*
L'application de gestion des stocks de produits chimiques est écrite en java. Nous allons voir comment utiliser la base relationnelle avec JBDC.
Vous renommerez `BaseACompleter.java` (fichier sur le portail) en `Base.java`.

### Développement Java avec JDeveloper

Ouvrez `JDeveloper`, et cochez toute les cases de la première fenêtre.
Créez un `Workspace` et un `Project` dans cet espace de travail. Pour cela, dans l'onglet Applications (à gauche), sélectionnez `Applications` puis avec le clic droit, `new Application`. Le nom de votre application est `Application1` et choisissez son répertoire sur H, par exemple `H:\tpjdbc`. Suivez les fenêtres de dialogue pour créer un projet `Project1` dans le répertoire `H:\tpjdbc\Project1`

Sous windows, dans le poste de travail, allez dans le répertoire `H:\tpjdbc\Project1`, créez-y un sous-répertoire `src` et recopiez-y le fichier `Base.java`.

Sous `JDeveloper`, sélectionnez le projet `Project1` puis avec le clic droit, choisissez `Add to project content`. Ajoutez le répertoire `src`.

Pour pouvoir utiliser le pilote JDBC d'oracle, il faut l'ajouter dans le `CLASSPATH` du projet. Pour cela, sélectionnez votre projet, avec le clic droit choisissez `Project Properties` puis la rubrique `Libraries`. Appuyez sur le bouton `Add jar/directory` et suivez les instructions pour ajouter le chemin `C:\Oracle\product\10.2.0\client_1\jdbc\lib\ojdbc14.jar`

### Accès à la base

Question 2.1 : La classe `Base` permet de créer une connexion, et d'obtenir des ressources pour interroger la base. Complétez le fichier `Base.java` et écrivez un `main` pour tester la connexion à votre base de données.

Question 2.2 : Définir une classe java qui accède au schéma et possède une variable d'instance de type `Base` :

```
public class Stock {
  private Base laBase;
  private CallableStatement ... ;
  private Statement ... ;
  private PreparedStatement ... ;
  ...
}
```

Toutes les questions suivantes concernent des méthodes d'instance de la classe `Stock`. Pour tester ces méthodes au fur et à mesure, vous écrirez un `main` (sommaire) dans la classe `Stock`. Si, dans les questions suivantes, vous avez besoin d'utiliser dans une méthode un objet de type `Statement`, ou d'un de ses sous-types, écrivez l'initialisation de cet objet en dehors de la méthode.

Question 2.3 : Ecrire une méthode `remplirContainer` qui prend en paramètre un identifiant de produit et un identifiant de container. Cette méthode retire le container de son hangar (on met à `null` les colonnes `lig` et `col` pour ce container) et ajoute dans la base le fait que ce container est rempli avec ce produit. Cette méthode déclenche une SQLException si le produit ou le container n'existent pas.

Question 2.4 : Ecrire une méthode `affecterContainer` qui prend en paramètre un identifiant de container et appelle la procédure stockée `affecter_container` du paquetage `PAQ_STOCK`. Cette méthode déclenche une SQLException avec un message d'erreur adapté, à chaque fois que la procédure stockée déclenche une exception (dont vous récupérez le code par `e.getErrorCode()`).

Question 2.5 : Ecrire une méthode `voirStockProduit` qui prend un identifiant de produit en paramètre et qui affiche la liste de tous les containers contenant ce produit avec leur hangar de stockage. Cette méthode utilisera un `PreparedStatement`. On déclenchera une `SQLException` si le produit n'existe pas.

## A Rendre

Le paquetage PL/SQL (fichier texte) et la classe `Stock`, par mail, avant le jeudi 13 octobre soir.