L3 MIAGE – 2014/2015

Bases de Données

le 19 décembre 2014

Examen

durée 3h, documents autorisés, appareils mobiles de communication interdits.

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. On ne range jamais un container vide dans un hangar. Un hangar possède une capacité, exprimée en litres, et représentant le volume maximal qui peut y être stocké. 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)
PRODUIT(id_produit, nom, categorie)
CONTAINER(id_container, volume, id_produit, lig, col)
-- On sait qu'un container est vide quand CONTAINER.id_produit vaut null
-- On sait qu'un container est rangé dans un hangar quand lig et col sont non null.
```

dans cet exercice, vous pouvez répondre à une question même si vous n'avez pas su répondre aux questions précédentes. Lisez entièrement l'énoncé avant de commencer.

Question 1.1 : Donner les commandes SQL qui permettent de créer les tables.

Question 1.2 : Ecrire les requêtes SQL qui donnent :

1. la liste de tous les containers non vides avec leur produit et la catégorie.
   Schéma `(id_container, volume, nom, categorie)`
2. la liste des produits avec le nombre de containers pour chaque produit. On ne s'intéresse pas aux produits qui ne sont stockés dans aucun container.
   Schéma `(id_produit, nom, nb_containers)`
3. les hangars avec le nombre de containers qu'ils contiennent (0 si hangar vide)
   Schéma `(lig, col, nb_containers)`
4. les hangars avec le volume stocké (0 si hangar vide)
   Schéma `(lig, col, volume_stocke)`

Question 1.3 : On veut écrire un trigger qui vérifie que les affectations des containers dans les hangars sont cohérentes du point de vue du volume, c'est à dire qu'un hangar ne contient jamais un volume de container supérieur à sa capacité.
Voici un premier trigger :

```
create or replace trigger verif_capacite
before insert or update of lig,col,volume
on container
for each row
declare
  somme NUMBER ;
  cap NUMBER ;
begin
  select sum(volume) into somme from container
  where :new.lig = lig and :new.col = col ;
  select capacite into cap from hangar
  where :new.lig = lig and :new.col = col ;
  if (somme > cap) then raise PAQ_STOCK.PB_CAPACITE ; end if;
end verif_capacite ;
```

On suppose que l'exception `PAQ_STOCK.PB_CAPACITE` existe bien, d'ailleurs il n'y a pas d'erreur à la compilation. Pourquoi ce trigger ligne peut provoquer une erreur à l'exécution ?

Pour résoudre le problème précédent et éviter des calculs récurrents, on ajoute une colonne `volume_reel` à la table `HANGAR`. Cette colonne représente le volume réellement stocké. On définit aussi une contrainte exprimant que cette nouvelle colonne `volume_reel` doit toujours être inférieure à la colonne `capacité`.

Question 1.4 : Comment en SQL ajouter cette colonne et cette contrainte à la table `HANGAR`.

Question 1.5 : Donner une instruction `update` qui permet de mettre à jour la colonne `volume_reel` de `HANGAR` à partir des données de la table `CONTAINER`.

Question 1.6 : 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 1.7 : 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 1.8 : Définir une vue `hangar_vide` qui donne la liste des hangars vides.
Le schéma de cette vue est `(lig, col, capacite)`.

Les vues `hangar_avec_categorie` *et* `hangar_vide` *peuvent être utilisées par la suite.*
On définit un paquetage `PAQ_STOCK` pour gérer le stockage des produits. 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 par exemple l'utiliser dans une requête SQL de la forme :

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

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. D'abord, cette 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, et une exception `CONTAINER_INCONNU` si le container n'existe pas.

Question 1.9 : Ecrire le corps de la procédure `affecter_container`.

Question 1.10 : Comment déclarer l'exception `PAS_HANGAR_DISPO` dans le paquetage, et l'associer au code d'erreur SQL -20001 ?

Exercice 2 : JDBC

Dans cette exercice, vous allez écrire quelques méthodes d'une classe `Stock` qui permet d'interroger la base de stockage des produits chimiques en java.

```
public class Stock {

  private Connection connect ;
  private CallableStatement ... ;
  private Statement ... ;
  private PreparedStatement ... ;

  ...
}
```

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.
On suppose que l'initialisation de la variable d'instance `connect` est faite dans le constructeur de la classe `Stock`.
Question 2.1 : Ecrire une méthode `remplirContainer` qui prend en paramètre un identifiant de produit et un identifiant de container. Cette méthode retire éventuellement 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 2.2 : Ecrire une méthode `affecterContainer` qui prend en paramètre un identifiant de container et appelle la procédure stockée `affecter_container` du paquetage `PAQ_STOCK`.

Question 2.3 : 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 ne déclenche pas d'exception si le produit n'existe pas.

Exercice 3 :

`R(numCmde, numProduit, numClient, dateCmde, quantité, marque, emailClient, nomClient, nomProd)` est une relation vérifiant l'ensemble suivant de dépendances fonctionnelles :

	numCmde	→	dateCmde
	numCmde	→	numClient
numCmde	numProduit	→	marque
	numProduit	→	nomProd
numCmde	numProduit	→	quantité
	numProduit	→	marque
	numCmde	→	nomClient
	numClient	→	nomClient
	numClient	→	emailClient

Question 3.1 : Quelles sont les clés de la relation R ? Justifiez votre réponse.

Question 3.2 : Est-ce que R est 3NF ? Si R n'est pas 3NF, appliquez l'algorithme de normalisation pour obtenir une décomposition 3NF de R.