Université Paris 7

Licence 3 Informatique et Master 1 ISIFAR, Bases de données.

16 mai 2009

Durée : 2 heures et 45 minutes. Documents manuscrits, notes de cours, notes de TD/TP autorisés. Livres interdits.

Le sujet comporte 6 pages.

Comprendre les requêtes

Exercice 1 Pour des tables R et S

```
R  a |b  | c              S  c | d
  ---+---+---               ---+---
   1 | 8 | 1                 2 | 6
   1 | 6 | 2                 2 | 8
   2 | 11| 1                 7 | 5
   2 | 6 | -5               13 | 8
   3 | 0 | 6                13 | 9
   3 | 3 | 0
   4 | 4 | 4
   4 | 7 | 2
   5 | 0 | 3
```

donner les résultats des requêtes

```
SELECT A, MIN(C) AS "minimum C"
FROM R
WHERE A <> ALL( SELECT B+1 FROM R )
GROUP BY A
HAVING MAX(B) <> 1;
```

et

```
SELECT  C, MAX(D) AS "max D"
  FROM S NATURAL LEFT OUTER JOIN R
  GROUP BY C;
```

Écrire les scripts SQL

Exercice 2 Une base de données de gestion de commandes est composée des tables :

```
client(client_id, nom, adresse)
commande(commande_id, date_commande, client_id, adresse_livraison)
produit(produit_id, prix)
commande_produit(commande_id, produit_id, quantite)
livraison(commande_id, produit_id, date_expedition, quantite)
```

(A) La table `client` contient tous les clients, `client_id` sert de clé.

(B) La table `commande` permet d'enregistrer toutes les commandes. `date_commande` nous donne la date de l'enregistrement de la commande. `commande_id` donne la clé de cette table. `client_id` fait référence au client qui a effectué la commande. `adresse_livraison` donne l'adresse où il faudra livrer la commande.

(C) La table `produit` donne tous les produits que notre entreprise peut livrer, `prix` c'est le prix d'une unité de produit. On suppose que si `prix` est non NULL alors `prix > 0` (le prix est donné à un centime près).

(D) Dans une seule commande on peut commander plusieurs produits (à condition que l'adresse de livraison soit la même). La table `commande_produit` nous indique pour chaque commande quels produits ont été commandés et en quelle quantité.
Le couple `commande_id, produit_id` forme la clé de cette table. L'attribut `commande_id` fait la référence vers la table de commandes tandis que `produit_id` est une référence vers un produit. On suppose que `quantite` (valeur entière) est > 0.

(E) La table `livraison` permet de stocker les informations concernant la réalisation d'une commande, en particulier `date_expedition` donne la date de l'envoi du produit. Les produits peuvent être expédiés au fur et à mesure, par exemple pour une commande de 25 pièces de produit A et 30 pièces de produit B il est possible d'avoir quatre livraisons pour réaliser toute la commande : 20 pièces A, 28 pièces B, et ensuite 5 pièces A suivi par 2 pièces B. Le triplet `commande_id, produit_id, date_expedition` est la clé de cette table. Le couple `commande_id, produit_id` fait référence au même couple de la table `commande_produit`.

On suppose que tous les attributs de tables sont non null, sauf l'attribut `prix` de la table ~~`livraison`~~ produit qui peut avoir la valeur `NULL` si le produit n'est pas disponible (noter que les contraintes non NULL sont déjà implicites pour les clés).
Les questions suivantes sont indépendantes.

Question 1: Écrire les scripts SQL qui construisent toutes ces tables avec les contraintes d'intégrité suivant la description ci-dessus.

Question 2: Écrire un script permettant d'insérer dans la table `livraison` deux lignes `(2,3,aujourd'hui,10)` et `(2,8,hier,20)` (l'ordre d'attributs est le même que dans la spécification de la table).
À la place de `aujourdhui` il faut mettre la date d'aujourd'hui et `hier` la date d'hier. Mais attention, on préfère que la date d'aujourd'hui soit obtenue par un appel approprié à une fonction SQL au lieu de `2009-05-16` mis en dur. Pareil pour la date d'hier on préfère une solution qui calcule la date d'hier à l'aide d'une expression SQL plutôt que `'2009-05-15'` écrit en dur.

Question 3: Mettre à jour un enregistrement dans la table `commande_produit` : pour la commande `commande_id=6` on modifiera la quantité du produit `produit_id=3` commandé en y mettant une nouvelle valeur 33 (on suppose la table `commande_produit` contient déjà un enregistrement avec `commande_id=6` et `produit_id=3`, il faut juste modifier la valeur de l'attribut `quantite` de cette ligne – ce n'est pas l'insertion d'un nouvel enregistrement).

Question 4: Combien d'unités du produit `produit_id=3` a-t-on déjà expédié (c'est-à-dire livré) en 2009 ?

Question 5: Supprimer de la table `produit` tous les produits dont le prix est NULL.

Question 6: Trouver tous les clients qui figurent dans la table de clients et qui n'ont jamais passé aucune commande.

Question 7: Trouver tous les clients qui ont passé au moins 2 commandes. Pour chacun de ces clients afficher `client_id`, l'adresse et le nombre de commandes qu'il a passé.

Question 8: Calculer la valeur totale de la facture pour la commande avec `commande_id=3`. La valeur totale de la facture est calculée comme la somme de `quantite*prix` sur tous les produits commandés.
Par exemple si cette commande concerne deux produits et pour le premier on a commandé 100 unités (c'est la quantité) à 2 euros l'unité (le prix dans la table de produit) et pour le deuxième 1000 unités à 5 euros l'unité alors le total de la facture s'élèvera à $100 * 2 + 1000 * 5 = 5200$ euros.

Question 9: Supposons qu'un des produits commandés dans la commande `commande_id=2` est le produit avec `produit_id=3`, c'est-à-dire la table `commande_produit` contient un enregistrement avec `commande_id=2` et `produit_id=3`.
Afficher combien d'unités de ce produit il reste encore à livrer pour cette commande.
Par exemple si cette commande concernait 100 pièces du produit `produit_id=3` et on a déjà effectué deux livraison, la première de 10 pièces et la deuxième de 20 pièces alors il reste encore à expédier 70 pièces et c'est cette valeur qu'il faudra calculer et afficher.

Question 10: Trouver le(s) produit(s) dont le prix unitaire est le plus élevé.

~~**Question 11:**~~ Trouver tous les produits dont le prix unitaire est supérieur à la moyenne des prix.

Question 12: Rappelons que chaque commande peut concerner plusieurs produits. Quel est le nombre moyen de produits différents par commande ?
Par exemple s'il y avait au total juste 3 commandes, la première pour 3 produits, la deuxième pour 5 et la troisième pour 2, alors en moyenne on aura $(3+5+2)/3$ produits par commande.
Ici s'agit bien de types de produits et pas de quantité, donc un produit est identifié par `produit_id` unique.

Question 13: Créer une vue `total_commande(produit_id, quantite)` qui pour chaque produit donne la quantité totale de toutes les commandes pour ce produit (le nombre de pièces commandées de ce produit).

Question 14: Pour chaque commande passée par le client `client_id=4` afficher : l'identifiant de la commande `commande_id`, la date la commande et le nombre de produits différents commandés (c'est le nombre de produits et pas le nombre d'unités de produits).

~~**Question 15:**~~ Trouver tous les clients qui ont passé au moins une commande avec l'adresse de livraison différente de l'adresse du client.

Question 16: Quel est le nombre total de commandes reçues par notre entreprise entre 2000 et 2005 ?

Question 17: Pour chaque produit trouver le nombre total d'unités commandées. Le résultat doit être trié par nombre d'unité décroissant.

Question 18: Créer une vue `reste(commande_id, produit_id, quantite_restante)` qui pour chaque couple `commande_id, produit_id` qui correspond à une commande d'un produit dont la livraison n'est pas encore achevée donne la quantité qui reste à expédier.
Par exemple si la commande `commande_id=3` concerne 100 unité du produit `produit_id=2` et on a déjà livré 70 unité (peut-être en plusieurs livraisons) alors la vue doit contenir l'enregistrement `(3,2,30)`.

JDBC

Exercice 3 Voilà un programme Java qui est censé être exécuté quand on effectue la livraison (on supposera que la date de la livraison est toujours la date d'aujourd'hui).

```
1   import java.sql.*;
2   public class Livraison{
3       public static void main(String[] args){
4
5           int commandeId, produitId, qty;
6
7           if( args.length != 3 ){
8               System.err.println("usage: Livraison commande_id produit_id quantite");
9               System.exit(1);
10          }
11
12          commandeId = Integer.parseInt(args[0]);
13          produitId = Integer.parseInt(args[1]);
14          qty = Integer.parseInt(args[2]);
15          if( qty <= 0 ){
16              System.err.println("quantite doir etre positive");
17              System.exit(0);
18          }
19
20          String url = "jdbc:postgresql:mai2009";
21          Connection conn=null;
22          try{
23              Class.forName("org.postgresql.Driver");
24              conn = DriverManager.getConnection(url,"zielonka","dupek");
25          }catch(ClassNotFoundException e){
26              System.err.println("Pb. pilote"); System.exit(1);
27          }
28          catch(SQLException e){
29              System.err.println("Pb " + e.getMessage()); System.exit(2);
30          }
31
32
33
34
35
36
37
```

```
38          Statement st = null;
39          ResultSet rs = null;
40          int qtyCommande = 0;
41          try{
42              st = conn.createStatement();
43              rs = st.executeQuery("SELECT quantite  " +
44                                   "FROM commande_produit " +
45                                   "WHERE commande_id = " + commandeId +
46                                   "      AND produit_id = " + produitId );
47              if( rs.next() )
48                  qtyCommande = rs.getInt(1);
49              else{
50                  System.err.println("rien a faire"); System.exit(2);
51              }
52              st.close();
53          }catch(SQLException e){
54              System.err.println("probleme : " + e.getMessage() ); System.exit(3);
55          }
56
57          int dejaLivre = 0;
58          try{
59              PreparedStatement ps = conn.prepareStatement("SELECT quantite " +
60                                                           "FROM livraison "+
61                                                           "WHERE commande_id = ? " +
62                                                           " AND produit_id = ? ");
63              ps.setInt(1,commandeId);
64              ps.setInt(2,produitId);
65              rs = ps.executeQuery();
66              while( rs.next() ){
67                  dejaLivre += rs.getInt(1);
68              }
69              ps.close();
70          }catch(SQLException e){
71              System.err.println("probleme : " + e.getMessage() ); System.exit(3);
72          }
73
74
75          if( dejaLivre + qty > qtyCommande ){
76              System.err.println("C'est trop. Abandon.");
77              System.exit(4);
78          }
79          /*****************************************************************************
80           *  Ici il manque le code qui enregistre la livraison dans la table livraison.
81           *  Ecrire le code manquant.
82           *****************************************************************************
83      }
84  }
```

Question 1: Expliquer brièvement le sens des instructions et le fonctionnement JDBC en commentant les lignes : 23, 24, 25 et 28.

Question 2: Expliquer ce qui est calculé par le programme dans le fragment à partir de la ligne 38 jusqu'à la ligne 55.
En particulier, qu'est-ce que fait l'instruction de la ligne 48 et qu'est-ce que contient la variable `qtyCommande` au moment où l'exécution du programme atteint la ligne 56.

Question 3: On peut remplacer l'argument entier `1` de `getInt()` de la ligne 48 par un argument approprié de type String et obtenir le même résultat. Réécrire l'instruction de la ligne 48 pour que `getInt` ait un argument String.

Question 4: Expliquer ce que fait le code entre les lignes 57 et 72 ? En particulier à quoi sert la variable `dejaLivre` ?

Question 5: Pourquoi la livraison est refusée dans la ligne 75 ?

Question 6: À la fin du programme il manque un bout de code (marqué par un commentaire). Écrire le code manquant.

Conception d'une BD

Exercice 4 Le but de cet exercice est de concevoir une base de données de gestion d'une prison. Les entités possibles : prisonniers (avec les attributs comme nom, prénom, numéro, date d'arrivée, date de sortie etc.), cellules avec les attributs : nombre de places, confort (entre 1 - le plus grand confort, cellule avec télé etc. jusqu'au 4 - le moins de confort pour le cachot), etc. Les prisonniers peuvent travailler dans différents services et ateliers : la cuisine, laverie, menuiserie, etc.
Vous pouvez ajouter d'autres entités de votre choix et d'autre relations, par exemple pour noter les couple de prisonniers qui ne doivent pas être mis dans la même cellule (par exemple en cas d'hostilité mutuelle).

Question 1: Proposez un MCD (UML) pour cette BD, avec une taille raisonnable d'au moins 3 entités avec des attributs appropriés, plusieurs (3 au minimum) relations. N'oubliez pas les cardinalités.

Question 2: En déduire le schéma de la base, c'est-à-dire quelles tables vous obtenez à partir de votre MCD. On ne demande pas d'écrire les commandes CREATE TABLE pour les tables mais vous devez indiquer pour chaque table que vous devez créer : le nom de la table, les attributs, la clé, les contraintes référentielles s'il y en a (clés étrangères).

Question 3: Est-ce que vous pouvez proposer des contraintes non référentielles (une ou deux suffisent) ?
