Formations en Informatique de Lille – 2010/2011

Bases de Données Avancées

16 décembre 2010

Examen

Durée 3h. Documents autorisés, appareils mobiles de communication interdits.

Exercice 1 : Une société propose aux utilisateurs des autoroutes d'Europe, des badges qui leur permettent de passer aux péages sans attendre, et de payer chaque mois une somme calculée en fonction des autoroutes qu'ils ont empruntées et des kilomètres parcourus. Cette société a des clients (sociétés de transports ou simples usagers de l'autoroute), qui possèdent des véhicules. Lorsqu'un véhicule emprunte une autoroute, il parcourt des tronçons délimités par un péage d'entrée et un péage de sortie. Emprunter un tronçon a un certain coût de base, multiplié par un coefficient en fonction de l'autoroute et de la catégorie du véhicule qui a emprunté le tronçon. Chaque mois, la société envoie à ses clients une facture qui comprend (en plus de l'abonnement non pris en compte dans ce sujet) une somme totale calculée en fonction des différents trajets effectués par les véhicules de chaque client.
Après une phase d'analyse, la société a décidé de modéliser la gestion des péages avec les tables suivantes :

- Table `PEAGE(autoroute, km, ville, pays)`
  `autoroute` désigne le nom de l'autoroute (par exemple `'A11'`), ce nom est unique pour l'Europe. `km` est le kilomètre où se situe le péage sur cette autoroute. Les deux autres colonnes (`ville` et `pays`) désignent la ville la plus proche du péage. La figure 1 présente 3 péages sur l'autoroute `A26`, aux kilomètres 50, 62 et 86.
  Exemple de donnée dans la table `PEAGE` :
  `('A11', 156, 'Le Mans', 'FRANCE')`
  `('A26', 50, null, 'FRANCE')`

- Table `TRONCON(id_tron, autoroute, km_deb, km_fin, cout)`
  `id_troncon` est bien sûr l'identifiant du tronçon. Un tronçon se situe sur une autoroute, entre un péage d'entrée au kilomètre `km_deb` et un péage de sortie au kilomètre `km_fin`. Curieusement, selon le sens de circulation, `km_deb` peut être supérieur à `km_fin`. `cout` est le coût de base pour tout véhicule qui emprunte ce tronçon. Dans la figure 1, les flèches symbolisent 2 tronçons sur l'autoroute `A23`.

- Table `CLIENT(id_client, nom, adresse)`
  `id_client` est un identifiant. `nom` est le nom du client, nom d'un particulier ou d'une entreprise.
  Exemples de données dans la table `CLIENT` ;
  `(10, 'Legrand Transport', 'Z.I. de la plaine, 91000 Evry FRANCE')`
  `(30, 'Helmut Schmidt', 'PeterStrasse 25, 52062 Aachen GERMANY')`

- Table `VEHICULE(immatriculation, pays, id_client, categorie)`.
  Un véhicule est identifié par son immatriculation. Il est immatriculé dans un pays, qui sera représenté par 1, 2 ou 3 lettres (`F` pour France, `CH` pour Suisse, `IRL` pour l'Irlande, ...). `id_client` est l'identifiant du client qui possède le véhicule.
  Exemple de donnée dans `VEHICULE` : `('HJK-678-YU', 'SL', 10, 'semi-remorque')`

- Table `EMPRUNTE(immatriculation, id_tron, dateEtHeure, cout)`
  Une ligne de la table emprunte signifie que le véhicule identifié par une `immatriculation` a emprunté le tronçon `id_tron` au moment `dateEtHeure` et que ça lui a coûté `cout`. L'attribut `cout` est calculé en fonction de la catégorie du véhicule, et du tronçon.

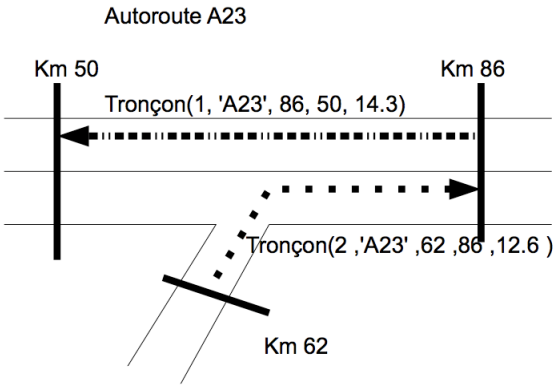

Figure 1: Exemple de tronçons

- Table `COEFFICIENT(catégorie, autoroute, coef)`
  Cette table donne pour chaque autoroute et chaque catégorie de véhicule le coefficient multiplicateur du coût de base. Par exemple, si un tronçon de l'autoroute `A1` a un coût de base de 6.50 euros, qu'une moto emprunte ce tronçon, et qu'il y a dans `COEFFICIENT` une ligne `('moto','A1',1.3)`, alors la moto devra payer 6.5×1.3 euros pour ce tronçon (soit 8.45 euros).

PL/SQL

Question 1.1 : Ecrire une procédure sans paramètre `calculer_cout_emprunte` qui calcule la colonne `cout` de la table `EMPRUNTE`, pour toutes les valeurs de cout à 0. Cette procédure manipule un curseur sur (une partie de) la table `EMPRUNTE`, curseur qui permet également la modification de la colonne coût.

JDBC

On écrit une application java qui utilise cette base de données. On définit une classe `Autoroute` qui possède une variable d'instance `connect` de type `java.sql.Connection`. On supposera que la connexion à la base a été initialisée dans le constructeur de la classe `Autoroute`.
Pour la question suivante, vous déclarerez en dehors de la méthode les variables de (super-)type `Statement` que vous utilisez et vous les initialiserez.

Question 1.2 : Ecrire une méthode d'instance `editerFactureMois(int mois, int client)` dans la classe `Autoroute`, qui affiche la facture d'un client pour un mois sous la forme de la liste des coûts attribués à chaque véhicule de ce client pour ce mois, puis affiche le total. Cette méthode commence par appeler la procédure stockée `calculer_cout_emprunte` pour être certain que les coûts de la table `EMPRUNTE` sont corrects.
Voici par exemple ce que donne l'exécution de cette méthode pour un client qui possède 3 véhicules dont 2 ont emprunté (plusieurs fois) l'autoroute en décembre avec un coût de 42 euros pour le premier véhicule et un coût de 56.1 euros pour le second :

```
facture :
YRT-672-Z : 42
DRF-67-REF : 56,1
total : 98,1
```

Si aucun véhicule de ce client n'a emprunté l'autoroute pour ce mois ci, cette méthode affiche un message approprié.

Indexation et calcul de coûts

Il y a (environ) 100 autoroutes, qui ont en moyenne 15 tronçons. Une ligne de la table `TRONCON` fait 20 octets.

Question 1.3 : Sachant qu'un bloc fait 8Ko, combien de blocs occupe la table `TRONCON` ?

Question 1.4 : On considère la requête SQL suivante, où a et k représentent des valeurs quelconques :

```
select * from TRONCON
where autoroute = a and km_deb = k
```

Pourquoi est-il intéressant d'utiliser un index B-arbre sur `(autoroute,km_deb)` ? On suppose qu'un nom d'autoroute prend 4 octets, qu'une valeur kilométrique prend 1 octet et qu'un pointeur prend 2 octets. Vous justifierez votre réponse en calculant le nombre de blocs lus pour répondre à la requête avec ou sans index.

Question 1.5 : On considère la requête SQL suivante, où a et k représentent des valeurs quelconques :

```
select * from TRONCON
where autoroute = a and km_deb >= k
```

Pourquoi n'est-il pas intéressant d'utiliser systématiquement un index sur `(autoroute,km_deb)` dans ce cas ? Là encore, vous justifierez votre réponse en calculant le nombre de blocs lus pour répondre à la requête avec ou sans index.

La table `VEHICULE` possède un index primaire[1] de type B-arbre, sur `immatriculation`.

Question 1.6 : Donner une requête `update` dont l'index permet d'accélérer le traitement.

Question 1.7 : Donner une requête `update` qui est ralentie par l'existence de l'index.

Exercice 2 : Pour gérer les inscriptions des enfants aux activités d'un centre de loisirs, on utilise le modèle Objet-relationnel. Les types et tables sont créés sous Oracle avec les instructions suivantes :

```
create type activite_type ;
/
create type ref_activite_type as object(
  ref_activite REF activite_type
);
create type ens_activite_type as TABLE OF ref_activite_type ;
/
create type enfant_type as object(
  id_enfant NUMBER,
  nom VARCHAR2(20),
  prenom VARCHAR2(30),
  les_activites ens_activite_type
);
/
create type ref_enfant_type as object(
  ref_enfant REF enfant_type
);
/
```

[1] i.e. la table est stockée avec l'index

```
create type ens_enfant_type as TABLE OF ref_enfant_type ;
/
create type activite_type as object(
  id_activite NUMBER,
  libelle VARCHAR2(50),
  les_inscrits ens_enfant_type
);
/

create table ACTIVITE of activite_type(
constraint activite_pkey primary key(id_activite),
constraint defaut_les_inscrits les_inscrits default ens_enfant_type()
)nested table les_inscrits store as tab_les_inscrits;

create table ENFANT of enfant_type(
constraint enfant_pkey primary key(id_enfant),
constraint defaut_les_activites les_activites default ens_activite_type()
)nested table les_activites store as tab_les_enfants;
```

Question 2.1 : Ecrire les requêtes qui permettent de donner les informations suivantes :

1. La liste des activités auxquelles est inscrite Léa Rollet
2. La liste des activités avec pour chacune le nombre d'inscrits
3. Les enfants qui ne sont inscrits à aucune activité

Question 2.2 : Ecrire `inscrire(enf enfant.id_enfant%type, act activite.id_activite%type)`, une procédure PL/SQL qui permet d'inscrire un enfant à une activité. On ne vérifiera pas si l'enfant et l'activité existent (on suppose que les paramètres sont corrects).

Exercice 3 : Question 3.1 : Ecrivez une fonction qui prend en paramètre un nom de table et une condition (chaîne de caractères), et qui renvoie une table avec les adresses (ROWID) des lignes de cette table qui vérifient cette condition. L'adresse d'une ligne peut s'obtenir par la pseudo-colonne `rowid`, cette colonne a pour type `VARCHAR2(18)`. Ainsi, une table de ROWID correspond au type :
`type tab_rowid_type as table of VARCHAR2(18)`
Voici le profil de cette fonction :

```
function requete_generique(la_table USER_TABLES.TABLE_NAME%type, condition VARCHAR2)
         return tab_rowid_type
```

Et quelques exemples d'exécution :

- `requete_generique('pays',null)` renvoie une table de 13 valeurs
  `TAB_ROWID_TYPE('AAAxStAAEAAAF50AAA', 'AAAxStAAEAAAF50AAB',...)`
- `requete_generique('pays','nom like ''c%''')` renvoie une table de 3 valeurs (pour le Cameroun, le Canada et la Chine)
  `TAB_ROWID_TYPE('AAAxStAAEAAAF50AAB', 'AAAxStAAEAAAF50AAF', 'AAAxStAAEAAAF50AAK')`

Question 3.2 : On suppose qu'on gère une base de données objet-relationnelle sous Oracle. Les tables sont des tables d'objets. Il y a une convention de nommage : Toute table `T` contient des objets de type `T_type`, et on a défini un type `tab_t_type`, table de `T_type`. De plus, tout type `T` dispose d'une méthode `afficher`, qui permet d'afficher un aperçu de l'objet. Définir une procédure `afficher_resultat_requete` qui prend en paramètre un nom de table et une condition (comme la question précédente) et affiche les lignes (objets) de cette table qui satisfont la condition.