SGBD B7 - UV 19786 (soir) - B6/B7 UV 21928 (ICPJ)

EXAMEN, 2e Session - 20 Septembre 2003

1 Modèle relationnel (7,5 points)

Le camping "Le Chassezac" gère l'occupation de ses places de camping avec un SGBD contenant les trois tables relationnelles $Places$, $Categories$ et $Clients$:

Places(noPl, noCat, surface, distance) (1)

Categories(noCat, prixAd, prixEnf) (2)

Clients(noPl, nomCl) (3)

Toutes les places de camping sont stockées dans la table *Places*. Chaque place est identifiée par un numéro (noPl) et appartient à une catégorie de prix (noCat) qui dépend de la surface de la place (en m^2) et de la distance de la plage (en m).

Les différentes catégories de prix sont stockées dans la table *Categories*. Chaque catégorie est identifiée par un numéro (noCat) et définit le prix journalier par adulte (prixAd) et par enfant (prixEnf) en euros.

La table *Clients* stocke pour chaque client son nom (nomCl) et la (les) place(s) qu'il occupe. Les places non-occupées n'apparaissent pas dans cette table.

Questions :

1. (1,5 points) Conception : Dessinez un schéma Entité-Association du schéma relationnel précédent. Indiquez également les cardinalités des associations.
   Solution :

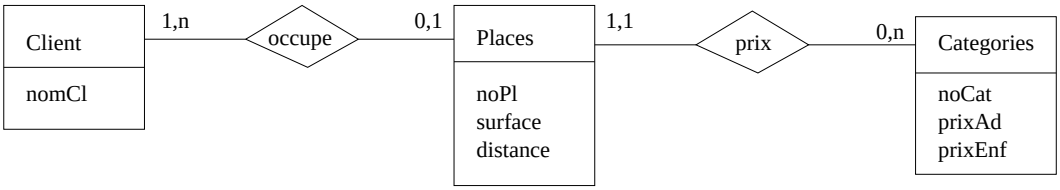

FIG. 1 – *Schéma entités-associations*

2. (6 points) Requêtes :
   (a) (1 point) Les numéros des places avec une surface supérieure à $20m^2$ et une distance inférieure à 100 mètres de la plage (algèbre relationnelle).
   Solution :

$$\pi_{noPl}(\sigma_{surface>20 \wedge distance<100} Places)$$

   (b) (2 points) Les numéros et la surface des places dont le prix est de 20 euros pour un adulte et qui se trouvent à moins de 100 mètres de la plage (SQL, calcul domaine).
   Solution :

```
select noPl, surface
from Places, Categories
where distance < 100 and
      prixAd = 20 and
      Places.noCat = Categories.noCat
```

$$\{P,S \mid \exists C,D,EPlaces(P,C,S,D) \wedge Categories(C,20,E) \wedge D < 100\}$$

(c) (1 point) Le numéro de la place qui est la plus près des la plage (SQL).
Solution :

```
select noPl
from Places A
where not exists (select *
                  from Places B
                  where B.distance < A.distance)
```

ou

```
select noPl
from Places
where distance =  (select min(distance)
                   from Places)
```

(d) (1 point) La surface totale des places non-occupées du camping (SQL).
Solution :

```
select sum(surface)
from Place
where noPl not in select noPl from Clients
```

(e) (1 point) Le prix moyen par adulte pour une place de $20m^2$ (SQL).
Solution :

```
select avg(prixAd)
from Place, Categorie
where Place.noCat = Categorie.noCat
and surface = 20
```

2 Organisation physique (2 points)

On veut stocker un fichier F avec 30000 articles d'une taille fixe de 1KO (kilo-octet) par article.

Questions :

1. (1 point) De combien de pages a-t-on besoin au minimum pour stocker tout le fichier si la taille d'une page est 32KO?
   Solution : $\lceil \frac{30000}{32} \rceil = 938$ *pages*
2. (1 point) Indexation par hachage : On suppose que les valeurs de l'attribut clé A du fichier sont des entiers avec une distribution uniforme. Définissez une fonction de hachage H pour indexer le fichier F. Expliquez votre choix.
   Solution : $H = MODULO(A,999)$*: On distribue le fichier dans 999 pages, ce qui correspond a un taux de remplissage de 94% (on suppose que* $A > 999$*).*

3 Optimisation (5,5 points)

Soit le schéma relationnel :

$$Journaliste(JID,Nom,Prenom) \quad (4)$$
$$Journal(Titre,Redaction,ReID) \quad (5)$$

La table $Journaliste$ stocke les informations (Nom, Prenom) sur les journalistes (JID est le numéro d'identification du journaliste). La table $Journal$ stocke pour chaque rédaction d'un journal le titre du journal (Titre), le nom de la rédaction (Redaction) et le JID de son rédacteur (ReID).

Soit la requête :

```
select Nom
from Journal, Journaliste
where Titre='Le Monde' And JID=ReID And Prenom='Jean'
```

1. (0,5 point) Que calcule cette requête ?
   Solution :
   Les noms des rédacteurs du journal le Monde qui s'appellent Jean.
2. (1 point) Voici deux requêtes algébriques :

$$\pi_{Nom}(\sigma_{Titre='LeMonde' \wedge Prenom='Jean'}(Journaliste \bowtie_{JID=ReID} Journal)) \quad (6)$$

$$\pi_{Nom}(\sigma_{Prenom='Jean'} Journaliste \bowtie_{JID=ReID} \sigma_{Titre='LeMonde'}(Journal)) \quad (7)$$

   Est-ce que les deux requêtes retournent le même résultat (sont équivalentes)? Est-ce qu'une requête est meilleure que l'autre si on considère les deux requêtes comme plans d'exécution? Expliquez votre réponse.

Le titre du journal est une clé au sens relationnel : il n'y a pas deux journaux ayant le même titre. On a seulement un index dense sur la table Journaliste sur l'attribut JID : I-Journaliste-JID.

1. (0,5 points) Donner le meilleur plan d'exécution physique sous forme arborescente ou sous forme d'une expression EXPLAIN. Les noeuds de l'arbre sont des opérateurs comme balayage séquentiel (BS), traversée d'index (TI), accès direct (par adresse) à une table (AD), tri, fusion, boucles imbriquées (BI), etc.
2. (1,5 points) Expliquez en détail ce plan.
   Solution :
   Réponse : par balayage séquentiel de la table Journal, on obtient les nuplets correspondant aux rédactions du Monde (sélection sur le titre). On fait ensuite une jointure par boucles imbriquées : la valeur de l'attribut ReID de ce nuplet sert de clé d'accès à l'index I-Journaliste-JID. La traversée de cet index donne les adresses des nuplets de la table Journaliste correspondant aux rédacteurs du Monde. Cette table est accédée directement (par adresse du nuplet correspondant au ReID du Monde). On projette le nuplet obtenu alors sur le champ Nom après avoir effectué une sélection sur le prénom.

Soit Bal le nombre de blocs de la table Journal et $Biste$ le nombre de blocs de la table Journaliste. Soit BTI le nombre de blocs lus lorsqu'on traverse l'index I-Journaliste-JID.

1. (1 point) Quel est le coût en nombre de lectures de blocs pour exécuter ce plan ?
2. (1 point) De combien de buffers a-t-on besoin en *mémoire centrale* pour exécuter ce plan (chaque buffer a pour taille la taille d'un bloc) ?
   Solution : *Réponse : Bal + BTI + 1. On a besoin d'un buffer pour lire un bloc de la table Journal, d'un buffer pour lire le contenu d'un nœud (feuille) d'index et d'un peu de mémoire pour stocker le nuplet de Journaliste.*

4 Concurrence (5 points)

L'exécution suivante est reçue par un SGBD utilisé pour des applications de commerce éléctronique.

H : $r_1[x]$ $r_2[y]$ $w_1[x]$ $r_2[z]$ $r_3[z]$ $r_3[x]$ $w_2[z]$ $r_1[z]$ c_2 $w_1[x]$ c_1 $w_3[x]$ c_3

1. (1 point) L'une des opérations dans l'application de commerce électronique est la modification du prix d'un produit suite à une réduction de prix d'un pourcentage donné (le prix du produit et le taux

de réduction sont stockés dans la base de données). Y a-t-il une ou plusieurs transactions de **H** qui puissent provenir de l'exécution de cette opération? Justifiez votre réponse.
Solution : *Les transactions de l'exécution* **H** *sont :*
T_1*:* $r_1[x]$ $w_1[x]$ $r_1[z]$ $w_1[x]$ c_1
T_2*:* $r_2[y]$ $r_2[z]$ $w_2[z]$ c_2
T_3*:* $r_3[z]$ $r_3[x]$ $w_3[x]$ c_3
La réduction du prix nécessite la lecture du taux de réduction et ensuite la mise-à-jour (lecture+écriture) du prix. Les transactions T_2 *et* T_3 *correspondent à cette suite d'actions. Mais les deux ne peuvent pas être en même temps des réductions de prix, car* z *ne peut pas être à la fois prix (pour* T_2*) et taux de réduction (pour* T_3*). Donc la réponse correcte est oui, soit* T_2*, soit* T_3*, mais pas les deux ensemble.*

2. (1 point) Vérifiez si **H** est sérialisable en identifiant les conflits et en construisant le graphe de sérialisation.
Solution : *Les conflits :*
sur x *:* $r_1[x]$-$w_3[x]$, $w_1[x]$-$r_3[x]$, $w_1[x]$-$w_3[x]$, $r_3[x]$-$w_1[x]$
sur y *: pas de conflit*
sur z *:* $r_3[z]$-$w_2[z]$, $w_2[z]$-$r_1[z]$
Le graphe de sérialisation contient un cycle T_1 T_3 T_2 *et un autre cycle* T_1 T_3*, donc* **H** *n'est pas sérialisable.*

3. (1,5 points) Montrez que l'exécution **H** n'évite pas les annulations en cascade. Est-elle recouvrable? Est-il possible, en modifiant seulement la position des **Commit** d'éviter les annulations en cascade?
Solution : *Après* $w_1[x]$*, on a plus tard* $r_3[x]$ *avant la fin de* T_1*, donc* **H** *n'évite pas les annulations en cascade. Pareil pour* $w_2[z]$ *suivi de* $r_1[z]$*. Par contre* **H** *est recouvrable, car dans les deux cas la transaction qui écrit est validée avant celle qui lit.*
Pour éviter les annulations en cascade, il faudrait déplacer c_1 *avant* $r_3[x]$*, respectivement* c_2 *avant* $r_1[z]$*. Le second déplacement est possible, par contre le premier est impossible, car* $r_1[z]$ *et* $w_1[x]$ *resteraient après* c_1*. Donc la réponse est non.*

4. (1,5 points) Quelle est l'exécution obtenue par verrouillage à deux phases à partir de **H**?
On considère qu'il existe deux verrous pour chaque enregistrement, un de lecture et un d'écriture. Le relâchement des verrous d'une transaction se fait au *Commit* et à ce moment on exécute en priorité les opérations bloquées en attente de verrou, dans l'ordre de leur blocage.
Solution : **H** *:* $r_1[x]$ $r_2[y]$ $w_1[x]$ $r_2[z]$ $r_3[z]$ $r_3[x]$ $w_2[z]$ $r_1[z]$ c_2 $w_1[x]$ c_1 $w_3[x]$ c_3
$r_1[x]$*,* $r_2[y]$ *s'exécutent, en prenant les verrous de lecture*
$w_1[x]$ *s'exécute, pas de conflit avec* $r_1[x]$ *(même transaction)*
$r_2[z]$ *prend le verrou de lecture sur* z *et s'exécute*
$r_3[z]$ *partage le verrou de lecture sur* z *avec* $r_2[z]$ *et s'exécute*
$r_3[x]$ *bloquée par* $w_1[x]$*, donc* T_3 *bloquée*
$w_2[z]$ *bloquée par* $r_3[z]$*, donc* T_2 *bloquée*
$r_1[z]$ *partage le verrou de lecture sur* z *avec* $r_2[z]$ *et* $r_3[z]$ *et s'exécute*
c_2 *bloquée car* T_2 *bloquée*
$w_1[x]$ *a déjà le verrou et peut s'exécuter*
c_1 *s'exécute et relâche les verrous de* $T_1 \Rightarrow$ $r_3[x]$ *peut obtenir le verrou sur* x *et s'exécuter, par contre* T_2 *reste bloquée*
$w_3[x]$ *prend le verrou d'écriture sur* x *et s'exécute*
c_3 *s'exécute et relâche les verrous de* $T_3 \Rightarrow$ *toutes les opérations de* T_2 *sont débloquées, donc* $w_2[z]$ *et* c_2 *s'exécutent*
Le résultat final est donc **H'** *:* $r_1[x]$ $r_2[y]$ $w_1[x]$ $r_2[z]$ $r_3[z]$ $r_1[z]$ $w_1[x]$ c_1 $r_3[x]$ $w_3[x]$ c_3 $w_2[z]$ c_2