Systèmes de Gestion de Bases de Données SGBD B7 et ICPJ

EXAMEN - 18 Septembre 2002

Attention: Tous les élèves doivent traiter les problèmes 1, 2 et 5. Les élèves du cours du soir traitent le problème 3, ceux du jour le problème 4.

1 Langages de Requêtes (7 points)

Soit les tables relationnelles suivantes pour stocker des informations sur la programmation de différentes chaînes de télévision (les attributs qui forment une clé sont soulignés) :

Chaîne (nom, pays)

Attributs : `nom` de la chaîne, nom du `pays`;

Exemple : `[nom: 'Canal+', pays: 'France']`

Emission (numéro, titre, type, durée, description)

Attributs : `numéro`, `titre`, `type` (par exemple : 'film', 'variété', 'informations', 'jeu', ...), `durée` en minutes et `description` détaillée de l'émission;

Exemple : `[numéro: 234, titre: '20 heures', type: 'informations', durée: 30, description: 'Le journal présenté par...' ]`

Programme (nom, jour, début, numéro, langue, son)

Attributs : `nom` de la chaîne, `jour` de la semaine, heure de `début` (format `hh:mm`), `numéro`, `langue` de l'émission et `son` ('mono', 'stéréo', 'dolby stéréo');

Exemple : `[nom: 'Canal+', jour: 'jeudi', début: 19:45, numéro: 123, langue: 'Française', son: 'stéréo']`

Les heures peuvent être comparées avec les comparateurs $=, <, <=$, ... (exemple: $19:30 < 20:00$).

Questions : Écrivez les requêtes suivantes dans les langages indiqués (pour les requêtes en calcul relationnel vous pouvez choisir entre le calcul n-uplet et le calcul domaine) :

1. Les titres des émissions sur TF1 qui commencent entre 19:00 et 20:00 (en algèbre et SQL).

   Solution :

   $$\pi_{titre}(\sigma_{nom='TF1' \wedge debut>=19:00 \wedge debut<=20:00} Programme \bowtie Emissions)$$

   ```
   Select titre
   From Programme,Emission
   Where nom='TF1'
   And debut >=19:00
   And debut <=20:00
   And Programme.numero=Emission.numero
   ```

2. Toutes les chaînes qui ne diffusent pas d'émissions d'information (en algèbre et SQL).

   Solution :

   $$\pi_{nom} Chaine - \pi_{nom}(\sigma_{type='Informations'}(Emission) \bowtie Programme)$$

```
Select nom
From Chaine
Where nom not in (Select nom
                  from Programme, Emission
                  where type = 'Informations'
                  and Programme.numero = Emission.numero
```

3. Les chaînes françaises qui diffusent des émissions en allemand (SQL et calcul).

   Solution :

```
select distinct Chaîne.nom
  from Chaîne, Programme
 where Chaîne.nom = Programme.nom
   and langue = 'Allemand'
   and pays = 'France'
```

$$\{no \mid \exists p, t, j, h, nu, ti, ty, d\ Chaine(no,'France', t) \wedge Programme(no, j, h, nu,'Allemand', s) \wedge Emission(nu, ti, ty, d)\}$$

4. Pour chaque chaîne : son nom et le nombre d'émissions d'informations diffusées (SQL).

   Solution :

```
select nom, count(*)
  from Programme, Emission
 where Emission.numéro=Programme.numéro
   and Emission.type = 'informations'
 group by nom
```

5. Les pays avec au moins 5 chaînes (SQL).

   Solution :

```
select pays
  from Chaîne
 group by pays
having count(*) >= 5;
```

2 Arbres B+ (3 points)

Voici une instance de la table **Chaîne** :

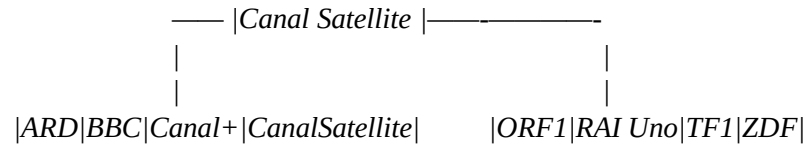
Chaîne	
nom	**pays**
page 1	
’Canal+’	’France’
’TF1’	’France’
’Canal Satellite’	’France’
page 2	
’ZDF’	’Allemagne’
’ARD’	’Allemagne’
’BBC’	’Angleterre’
page 3	
’RAI Uno’	’Italie’
’ORF 1’	’Autriche’

La table occupe trois pages disque dont les deux premières contiennent 3 n-uplets et la dernière contient 2 n-uplets.

Questions :

1. Dessinez l’arbre B+ d’ordre 2 sur l’attribut `nom` qu’on obtient si on considère que les n-uplets ont été insérés dans l’ordre de leur apparition dans la table.

   Solution :

2. Qu’est-ce qu’il faudrait faire avant de construire un index non-dense sur cette table?

   Solution : *Il faudrait trier la table par l’attribut ’nom’.*

3. Soit la requête suivante :

   ```
   select *
   from Chaîne
   where nom=’ZDF’ or pays=’France’;
   ```

   On suppose qu’il existe uniquement un index sur les noms des chaînes. Est-ce que cet index est utile pour évaluer cette requête ? Pourquoi ?

   Solution : *L’index n’est pas util car il faut parcourir la table à cause de la condition n “pays=’France”’?*

Solution :

1. *a faire*
2. *Trier la table sur la clé d’index*
3. *Non, car de toute façon il faut parcourir séquentiellement la table pour répondre au critère “pays=’France’ “*

3 SGBD B7: EXPLAIN (4 points)

On donne ci-dessous une requête SQL et le plan d’exécution fourni par Oracle :

Requête SQL :

```
select distinct titre
  from Chaine, Programme, Emission
 where Chaine.nom = Programme.nom
   and Programme.numero=Emission.numero;
```

Plan d'exécution :

```
0 SELECT STATEMENT
  1 SORT UNIQUE
    2 NESTED LOOPS
      3 MERGE JOIN
        4 SORT JOIN
          5 TABLE ACCESS FULL EMISSION
        6 SORT JOIN
          7 TABLE ACCESS FULL PROGRAMME
      8 TABLE ACCESS BY INDEX ROWID CHAINE
        9 INDEX UNIQUE SCAN PX23456
```

Questions :

1. **Existence d'index** : Existe-t-il un index ? sur quel(s) attribut(s) de quel(s) table(s) ?

   Solution : *Il existe un index sur l'attribut **nom** de la table **Chaîne**.*

2. **Algorithme de jointure** : Expliquer en détail le plan d'exécution (accès aux tables, sélections, jointure, projections)

   Solution :

   (a) parcours séquentiel des tables EMISSION et PROGRAMME

   (b) tri des deux tables sur l'attribut numéro et fusion ensuite -> relation R1

   (c) pour chaque n-uplet dans R1, chercher par l'index PX23456 la chaîne correspondante dans la table Chaîne et jointure -> R2

   (d) tri de la table R2 sur titre et projection

3. **Ajout d'index** : On crée un index sur l'attribut *numero* de la table **Programme**. Expliquez les améliorations en terme de plan d'exécution apportées par la création de cet index.

   Solution : *après création d'index le plan est :*

```
0 SELECT STATEMENT
  1 SORT UNIQUE
    2 NESTED LOOPS
      3 NESTED LOOPS
        4 TABLE ACCESS FULL EMISSION
        5 TABLE ACCESS BY INDEX ROWID PROGRAMME
          6 INDEX RANGE SCAN IY76896
      7 TABLE ACCESS BY INDEX ROWID CHAINE
        8 INDEX UNIQUE SCAN PX23456
```

4 ICPJ: Plan d'exécution (4 points)

Soit la requête SQL :

Requête SQL :

```
select distinct titre
  from Chaine, Programme, Emission
 where Chaine.nom = Programme.nom
   and Programme.numero=Emission.numero;
```

Il n'y a pas d'index sur le numéro ni dans la table Programme ni dans la table Emission. Il n'y a pas d'index non plus sur le nom ni dans la table Chaîne, ni dans la table Programme :

1. **Algorithme de jointure** : Donner un plan d'exécution éventuellement sous forme arborescente et l'expliquer en détail (accès aux tables, sélections, jointure, projections)

   Solution : *On projète Emission sur numéro et titre. On trie la table résultat sur l'attribut numéro. De même tri sur numéro de la table Programme après projection sur numero et nom. On fait la fusion et on obtient une première jointure. On trie le résultat sur le nom. Après projection sur nom, On trie la table Chaîne. On fait la fusion (2e jointure) et on projète le résultat sur le titre.*

2. **Ajout d'index** : On crée un index sur l'attribut *nom* de la table **Chaîne**. Donner le nouveau plan d'exécution et expliquez le en détail.

   Solution : *La deuxième jointure par tri fusion est remplacée par une boucle imbriquée avec traversée de l'index et accès direct à la table Chaîne.*

5 Concurrence (6 points)

L'histoire suivante est une exécution concurrente de trois transactions dans un système de location de DVD.

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

1. (1 point) L'un des programmes de ce système réalise la modification du prix de location d'un DVD en cas de promotion. Ce programme exécute les opérations suivantes : il demande à l'administrateur du système d'introduire le code du DVD et le taux de réduction du prix et diminue dans la base de données le prix avec le pourcentage donné.

   Laquelle des trois transactions de **H** représente l'exécution de ce programme? Pourquoi certaines opérations du programme n'apparaissent pas dans la transaction? Justifiez votre réponse.

   Solution : *Les trois transactions sont:*
   T_1: $r_1[x]$ $w_1[y]$ $w_1[z]$ c_1
   T_2: $r_2[y]$ $w_2[y]$ c_2
   T_3: $r_3[y]$ $w_3[z]$ c_3

   Le programme de promotion ne travaille qu'avec une seul enregistrement de la base de données: le prix de location du DVD donné. La diminution du prix avec le pourcentage donné signifie une lecture du prix, suivie de l'écriture du nouveau prix. La lecture du code du DVD et du taux de réduction ne sont pas des opérations sur la base de données.

   Seule la transaction T_2 correspond à ce programme.

2. (1 point) Vérifiez si **H** est sérialisable en identifiant les conflits et en construisant le graphe de sérialisation.

   Solution : *Les conflits sur x: aucun*
   Les conflits sur y: $r_2[y]$-$w_1[y]$, $w_1[y]$-$r_3[y]$, $w_1[y]$-$w_2[y]$, $r_3[y]$-$w_2[y]$.
   Les conflits sur z: $w_1[z]$-$w_3[z]$.

   Le graphe de sérialisation contient les arcs $T_2 \rightarrow T_1$, $T_1 \rightarrow T_3$, $T_1 \rightarrow T_2$ et $T_3 \rightarrow T_2$. Il y a deux cycles, donc **H** *n'est pas sérialisable.*

3. (1 point) L'exécution **H** est-elle stricte? Justifiez votre réponse.

   Solution : *Après $w_1[y]$, on a $r_3[y]$ avant la fin de T_1, donc* **H** *n'évite pas les annulations en cascade, donc elle n'est pas stricte non plus.*

4. (2 points) Quelle est l'exécution obtenue par verrouillage à deux phases à partir de **H**?

   On considère que le relâchement des verrous d'une transaction se fait au *Commit* et qu'à ce moment on exécute en priorité les opérations bloquées en attente de verrou, dans l'ordre de leur blocage.

   Solution : *$r_1[x]$, $r_2[y]$ s'exécutent*
   $w_1[y]$ est bloquée en attente de verrou sur y
   $r_3[y]$ s'exécute en partageant le verrou de lecture avec $r_2[y]$
   $w_1[z]$ est bloquée, car T_1 est bloquée
   $w_2[y]$ est bloquée à cause de $r_3[y]$
   c_1 est bloquée, car T_1 est bloquée
   $w_3[z]$ s'exécute
   c_3 s'exécute et relâche les verrous de T_3
   $\rightarrow$ $w_2[y]$ peut s'exécuter, mais pas les autres opérations bloquées
   c_2 s'exécute et relâche les verrous de T_2
   $\rightarrow$ $w_1[y]$, $w_1[z]$ et c_1 sont débloquées et s'exécutent.

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

5. (1 point) Notons **H'** l'histoire obtenue suite au verrouillage à deux phases. Peut-on dire, sans construire le graphe de sérialisation, que **H'** est sérialisable? Pourquoi? **H'** peut-elle être obtenue par estampillage simple? Justifiez votre réponse.

   Solution : *Oui, le verrouillage à deux phases garantit la sérialisabilité du résultat. Par contre,* **H'** *ne peut pas être obtenue par estampillage, car $w_2[y]$ ne pourrait pas être accepté après $r_3[y]$.*