Conservatoire National des Arts et Métiers

292 Rue St Martin 75141 Paris Cedex 03

Informatique - CNAM, Paris

Bases de données relationnelles
NFP 107 et NFP 107J

Exercices dirigés

C. Crochepeyre, M. Ferecatu, P. Rigaux, V. Thion et N. Travers

10 mars 2011

Chapitre 8

Optimisation de Requêtes

Exercice A : Soit la base STATION DE SKI contenant les relations suivantes :

- `hotel (noms,nomh,categorie,adresse,tel,nb_chambres)` (**5** catégories différentes,**1000** pages, chaque page contient **10** tuples).
- `station (noms,gare)` (**100** pages, chaque page contient **10** tuples).
- `activite (type_activite,noms)` (**100** pages, chaque page contient **10** tuples).

On suppose de plus l'existence des index suivants (arbres B+) :

- Index sur le couple `(noms,nomh)` de la relation `hotel` (**26 pages**).
- Index sur l'attribut `categorie` de `hotel` (**33 pages**).

Hypothèses :

- Chaque catégorie comprend un nombre égal d'hôtels.
- Chaque station comprend un nombre égal d'hôtels.
- Le nombre d'E/S lors du parcours de l'arborescence des index est *négligeable*.

Requête 1 : adresse, numéro de téléphone et nombre de chambres des hôtels de catégorie 3' dans la station de nom 'pesey' .

```
SELECT adresse, tel, nb_chambres
FROM   hotel
WHERE  noms='pesey' AND categorie=3;
```

1. Calculer le nombre d'E/S effectuées lors de l'évaluation de cette requête de sélection par un parcours séquentiel de la relation `hotel`.
2. Calculer le nombre d'E/S effectuées lors de l'évaluation de cette requête en utilisant l'index sur `categorie`.
3. Calculer le nombre d'E/S effectuées lors de l'évaluation de cette requête en utilisant l'index sur `noms,nomh`.
4. Conclure

Requête 2 : Soit le schéma suivant :

```
CREATE TABLE FILM (
       TITRE   VARCHAR2(32),
       REALISATEUR VARCHAR2(32),
       ACTEUR VARCHAR2(32)
);

CREATE TABLE VU (
       SPECTATEUR      VARCHAR2(32),
       TITRE           VARCHAR2(32)
);
```

Soit la requête SQL :

```
SELECT ACTEUR, REALISATEUR
  FROM FILM, VU
 WHERE FILM.TITRE=VU.TITRE
```

Dans chacun des cas suivants, donner l'algorithme de jointure de ORACLE (par EXPLAIN, un arbre d'exécution commenté, une explication textuelle ou tout autre moyen de votre choix) :

1. Il n'existe pas d'index sur TITRE ni dans FILM ni dans VU,
2. Il existe un index sur TITRE dans FILM seulement.
3. Il existe un index sur TITRE dans les deux relations.

Requête 3 : Soit la requête :

```
SELECT  e.enom, d.dnom
FROM    emp e, dept d
WHERE   e.dno = d.dno
AND     e.sal = 10000
```

sur la relation **EMP** de schéma *(EMPNO, SAL, MGR, DNO)*. Cette requête affiche le nom des employés dont le salaire (SAL) est égal à 10000, et celui de leur département. Indiquez le plan d'exécution dans chacune des hypothèses suivantes.

1. Index sur $DEPT(Dno)$ et sur $EMP(Sal)$
2. Index sur $EMP(Sal)$ seulement.
3. Index sur $EMP(Dno)$ et sur $EMP(Sal)$
4. Voici une autre requête, légèrement différente. Plan d'exécution s'il n'y a pas d'index.

```
SELECT e.enom
FROM emp e, dept d
WHERE  e.dno = d.dno
AND    d.ville = 'Paris'
```

5. Que pensez-vous de la requête suivante par rapport à la précédente ?

```
SELECT e.enom
FROM emp e
WHERE  e.dno IN (SELECT d.dno
                 FROM Dept d
                 WHERE d.Ville = 'Paris')
```

Voici le plan d'exécution donné par ORACLE :

```
0 SELECT STATEMENT
  1 MERGE JOIN
    2 SORT JOIN
      3 TABLE ACCESS FULL EMP
    4 SORT JOIN
      5 VIEW
        6 SORT UNIQUE
          7 TABLE ACCESS FULL DEPT
```

Qu'en dites vous ?

Requête 4 : Sur le même schéma, voici maintenant la requête suivante.

```
SELECT *
FROM EMP X WHERE X.SAL IN (SELECT SAL
                           FROM EMP
                           WHERE EMP.EMPNO=X.MGR)
```

Cette requête cherche les employés dont le salaire (SAL) est égal à celui de leur patron (MGR). On donne le plan d'exécution avec Oracle (outil EXPLAIN) pour cette requête dans deux cas : (i) pas d'index, (ii) un index sur le salaire et un index sur le numéro d'employé. Expliquez dans les deux cas ce plan d'exécution (éventuellement en vous aidant d'une représentation arborescente de ce plan d'exécution).

1. Pas d'index.

```
Plan d'exécution
----------------------------------------------------------------------------
0 FILTER
  1 TABLE ACCESS FULL EMP
  2 TABLE ACCESS FULL EMP
```

2. Index empno et index sal.

```
Plan d'exécution
----------------------------------------------------------------------------
0 FILTER
  1 TABLE ACCESS FULL EMP
  2 AND-EQUAL
    3 INDEX RANGE SCAN I-EMPNO
    4 INDEX RANGE SCAN I-SAL
```

3. Dans le cas où il y a les deux index (salaire et numéro d'employé) et où la requête est :

```
SELECT *
  FROM EMP X
 WHERE X.SAL = (SELECT SAL
                  FROM EMP
                 WHERE EMP.EMPNO=X.MGR)
```

on a le plan d'exécution suivant :

```
Plan d'exécution
----------------------------------------------------------------------------
0 FILTER
  1 TABLE ACCESS FULL EMP
  2 TABLE ACCESS ROWID EMP
    3 INDEX RANGE SCAN I-EMPNO
```

Expliquez-le.

Requête 5 : On reprend le schéma CINEMA donné dans le cours, **mais on ne sais plus quels index existent**.

Questions :

1. Donner l'ordre SQL pour la requête : *Quels sont les films d'Hitchcock visibles après 20h00 ?*

2. Donner l'expression algébrique correspondante et proposez un arbre de requête qui vous paraît optimal.

3. Sous ORACLE, l'outil EXPLAIN donne le plan d'exécution suivant :

```
0 SELECT STATEMENT
  1 MERGE JOIN
    2 SORT JOIN
      3 NESTED LOOPS
        4 TABLE ACCESS FULL ARTISTE
```

```
            5 TABLE ACCESS BY ROWID FILM
              6 INDEX RANGE SCAN IDX-ARTISTE-ID
        7 SORT JOIN
         8 TABLE ACCESS FULL SEANCE
```

Commentez le plan donné par EXPLAIN. Pourrait-on améliorer les performances de cette requête ?

Requête 6 : Soit le schéma suivant :

```
CREATE TABLE Artiste (
       ID-artiste     NUMBER(4),
       Nom            VARCHAR2(32),
       Adresse        VARCHAR2(32)
);

CREATE TABLE Joue (
       ID-artiste     NUMBER(4),
       ID-film        NUMBER(4)
);
```

```
CREATE TABLE Film (
       ID-film        NUMBER(4),
       Titre          VARCHAR2(32),
       Année          NUMBER(4),
       ID-réalisateur NUMBER(4)
);
```

Questions :

1. Donner l'ordre SQL pour la requête : *Afficher le nom des acteurs et le titre des films où ils ont joué.*
2. Donner l'expression algébrique correspondante.
3. Quel est à votre avis le plan d'exécution dans s'il n'existe que deux index, un sur FILM(ID-réalisateur), et un sur ARTISTE(ID-artiste)ÿ ?
4. Idem, avec un index sur $FILM(ID - Film)$, et un sur $JOUE(ID - Artiste)$.
5. Idem, avec un index sur $FILM(ID - Film)$, et un sur $JOUE(ID - Film)$.

Chapitre 9

Concurrence

9.1 Sérialisabilité et recouvrabilité

9.1.1 Graphe de sérialisation et équivalence des exécutions

Construisez les graphes de sérialisation pour les trois exécutions (histoires) suivantes. Indiquez les exécutions sérialisables et vérifiez si parmi les trois exécutions il y a des exécutions équivalentes.

1. $H_1 : w_2[x]\ w_3[z]\ w_2[y]\ c_2\ r_1[x]\ w_1[z]\ c_1\ r_3[y]\ c_3$
2. $H_2 : r_1[x]\ w_2[y]\ r_3[y]\ w_3[z]\ c_3\ w_1[z]\ c_1\ w_2[x]\ c_2$
3. $H_3 : w_3[z]\ w_1[z]\ w_2[y]\ w_2[x]\ c_2\ r_3[y]\ c_3\ r_1[x]\ c_1$

9.1.2 Recouvrabilité

Parmi les exécutions (histoires) suivantes, lesquelles sont recouvrables, lesquelles évitent les annulations en cascade et lesquelles sont strictes ? Indiquez s'il y a des exécutions sérialisables.

1. $H_1 : r_1[x]\ w_2[y]\ r_1[y]\ w_1[x]\ c_1\ r_2[x]\ w_2[x]\ c_2$
2. $H_2 : r_1[x]\ w_1[y]\ r_2[y]\ c_1\ w_2[x]\ c_2$
3. $H_3 : r_1[y]\ w_2[x]\ r_2[y]\ w_1[x]\ c_2\ r_1[x]\ c_1$

9.2 Contrôle de concurrence

9.2.1 Verrouillage à 2 phases

Un ordonnanceur avec verrouillage à 2 phases reçoit la séquence d'opérations ci-dessous.

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

Indiquez l'ordre d'exécution établi par l'ordonnanceur, en considérant que les opérations bloquées en attente d'un verrou seront exécutées en priorité dès que le verrou devient disponible, dans l'ordre de leur blocage. On suppose que les verrous d'une transaction sont relâchés au moment du Commit.

9.2.2 Verrouillage, interblocage

On considère l'exécution concurrente suivante :

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

1. Trouvez les conflits dans H et montrez que H n'est pas sérialisable.
2. Trouvez l'exécution obtenue à partir de H par verrouillage à deux phases. On considère que les verrous d'une transaction sont relâchés au Commit et qu'à ce moment les opérations bloquées en attente de verrou sont exécutées en priorité.
3. Le raisonnement suivant mène à une conclusion paradoxale.
   Considérons une histoire quelconque H non-sérialisable, traitée par verrouillage à deux phases. H contient donc un cycle dans son graphe de sérialisation. Chaque arc $T_i \rightarrow T_j$ de ce cycle provient d'une paire d'opérations conflictuelles de type "a_i[x] avant b_j[x]". Quand a_i[x] s'exécute, elle prend le verrou sur x, ce qui bloquera b_j[x].
   Donc T_i bloque T_j, mais comme les arcs forment un cycle, il y aura un blocage circulaire, donc un interblocage. Conclusion : toute histoire non-sérialisable traitée par verrouillage à deux phases produit un interblocage ! Cependant, l'exemple précedent montre que cette conclusion est fausse, car H n'est pas sérialisable et elle ne produit pas d'interblocage. Expliquez pourquoi il n'y a pas d'interblocage et pourquoi le raisonnement ci-dessus est faux.
4. Montrez que H n'évite pas les annulations en cascade. Montrez qu'en avançant des Commit, H peut éviter les annulations en cascade.

9.2.3 Verrouillage hiérarchique

Considérons une relation **Compte**(*Numéro, Titulaire, Solde*) et trois opérations qui s'exécutent sur cette relation à deux niveaux de granularité : relation et enregistrement. Supposons que dans cette relation il y a deux comptes pour le titulaire "Dupont". Les trois opérations sont les suivantes :
– un programme qui lit tous les comptes ;
– un programme qui réalise un transfert entre les deux comptes de Dupont ;
– un programme qui lit tous les comptes, afin de trouver les comptes de Dupont et de créditer leur solde avec les intérêts de l'année.

1. Montrer que l'exécution suivante est une exécution concurrente des trois opérations ci-dessus. Identifier les données qui correspondent à chaque variable.
   $H : r_1[z]\ r_2[z]\ w_1[x]\ r_3[x]\ w_3[x]\ c_2\ w_1[y]\ r_3[y]\ c_1\ w_3[y]\ c_3$
2. Trouver l'exécution obtenue par verrouillage hiérarchique à partir de H. On considère que le verrouillage hiérarchique utilise des verrous de type SIX (lecture et intention d'écriture), dans le cas présent pour le rajout des intérêts sur les comptes de Dupont.
   Les verrous sont relâchés au Commit et les opérations bloquées en attente de verrou sont traitées en priorité, dans l'ordre de leur blocage.
3. Que se passe-t-il dans le verrouillage hiérarchique de H si on n'utilise pas de verrou SIX au niveau de la relation (mais seulement du S et du IX séparemment) ?

9.3 Reprise après panne

9.3.1 Journalisation

Soit le journal physique ci-dessous, dans lequel on a marqué les opérations Commit et Abort réalisées :

$[T_1, x, 3], [T_2, y, 1], [T_1, z, 2], c_1, [T_2, z, 4], [T_3, x, 2], a_2, [T_4, y, 3], c_3, [T_5, x, 5]$

1. Indiquez le contenu de *liste_commit*, *liste_abort*, *liste_active*.

2. En supposant qu'une nouvelle écriture vient de s'ajouter au journal, lesquelles des écritures suivantes sont compatibles avec une exécution stricte : $[T_5, y, 4]$, $[T_4, z, 3]$ ou $[T_6, x, 1]$?

3. Quelles sont les entrées récupérables par l'algorithme de ramasse-miettes ?

4. Si les valeurs initiales des enregistrements étaient $x = 1$, $y = 2$,$z = 3$, et si une panne survenait à ce moment, quelles seraient les valeurs restaurées pour x, y et z après la reprise ?

5. En pratique, on stocke souvent dans chaque entrée aussi l'image avant de l'écriture (la valeur de l'enregistrement avant l'écriture). Aussi, les positions des Commit et Abort ne sont pas connues, mais seulement les listes de transactions *liste_commit*, *liste_abort* et *liste_active* au moment courant (la fin du journal). Quel est dans ce cas le contenu du journal ? Les valeurs initiales sont celles précisées ci-dessus ($x = 1$, $y = 2$,$z = 3$), ainsi que les listes courantes de transactions.

6. Expliquez le fonctionnement de l'algorithme de reprise avec annulation et avec répétition sur ce dernier journal.

9.4 Concurrence : Gestion Bancaire

Les trois programmes suivants peuvent s'exécuter dans un système de gestion bancaire. *Débit* diminue le solde d'un compte `c` avec un montant donné `m`. Pour simplifier, tout débit est permis (on accepte des découverts). *Crédit* augmente le solde d'un compte `c` avec un montant donné `m`. *Transfert* transfère un montant `m` à partir d'un compte source `s` vers un compte destination `d`. L'exécution de chaque programme démarre par un **Start** et se termine par un **Commit** (non montrés ci-dessous).

```
Débit (c:Compte;      |    Crédit (c:Compte;      |    Transfert (s,d:Compte;
       m:Montant)     |            m:Montant)     |               m:Montant)
begin                 |    begin                  |    begin
  t := Read(c);       |      t = Read(c);         |      Débit(s,m);
  Write(c,t-m);       |      Write(c,t+m);        |      Crédit(d,m);
end                   |    end                    |    end
```

Le système exécute en même temps les trois opérations suivantes :
(1) un transfert de montant 100 du compte A vers le compte B
(2) un crédit de 200 pour le compte A
(3) un débit de 50 pour le compte B

1. Écrire les transactions T_1, T_2 et T_3 qui correspondent à ces opérations. Montrer que l'histoire **H** : $r_1[A]$ $r_3[B]$ $w_1[A]$ $r_2[A]$ $w_3[B]$ $r_1[B]$ c_3 $w_2[A]$ c_2 $w_1[B]$ c_1 est une exécution concurrente de T_1, T_2 et T_3.

2. Mettre en évidence les conflits dans **H** et construire le graphe de sérialisation de cette histoire. **H** est-elle sérialisable ? **H** est-elle recouvrable ?

3. Quelle est l'exécution **H'** obtenue à partir de **H** par verrouillage à deux phases ? On suppose que les verrous d'une transaction sont relâchés après le Commit de celle-ci. Une opération bloquée en attente d'un verrou bloque le reste de sa transaction. Au moment du relâchement des verrous, les opérations en attente sont exécutées en priorité, dans l'ordre de leur blocage.
   Si au début le compte A avait un solde de 100 et B de 50, quel sera le solde des deux comptes après la reprise si une panne intervient après l'exécution de $w_1[B]$?

9.5 Concurrence : Agence environnementale

Une agence environnementale utilise une base de données relationnelle pour gérer des informations sur les véhicules. Le schéma est le suivant (les clés sont soulignées) :

propriétaire(nom, âge)
véhicule(immatriculation, kilométrage, nom, idmodèle) (nom = nom propriétaire)
modèle(idmodèle, crashtest, consommation)
carburant(désignation, cours, pollution)
utilise(idmodèle, désignation)

L'exécution suivante est reçue par le système de l'agence environnementale :

$H : r_1[x]r_2[y]r_3[x]w_2[y]w_1[x]r_3[y]r_1[z]w_2[z]w_1[y]c_1c_2w_3[z]c_3$

1. Parmi les programmes qui s exécutent dans le système, il y a *KilométrAge(i, k, a)*, qui fixe pour le véhicule d'immatriculation i le kilométrage à k et l'age du propriétaire à a. Montrez quelle transaction de H pourrait provenir de *KilométrAge*.
   On considère que les enregistrements (variables) de H sont des nuplets des relations de la base de données. On suppose que tout nuplet est accessible directement si l'on connaît sa clé.

2. Identifiez tous les conflits dans H et vérifiez si l'exécution est sérialisable en construisant le graphe de sérialisation.

3. A quel niveau de recouvrabilité se situe H (recouvrable, évitant les annulations en cascade, stricte) ?

4. Quelle est l'exécution obtenue par verrouillage à deux phases à partir de H ? 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.