SGBD B7 - UV 19786 (soir) et UV 21928 (ICPJ)

EXAMEN, 1ère Session - 26 Juin 2004

1 Modèle relationnel (7,5 points)

Dans le cadre de la réforme de la sécurité sociale le gouvernement veut créer un “dossier médical partagé” qui permettra en quelques minutes au médecin de savoir tous les actes médicaux, tous les traitements mais aussi tous les antécédents d’un patient. La gestion de tels dossiers peut être facilitée par un SGBD relationnel avec le schéma (simplifié) suivant (les attributs clés sont soulignés) :

- *Personne(Noss, Nom, Prénom, Adresse)*
- *Consultation(NoCons, Date, Symptome, NossPat, NossMed)*
- *Prescription(NoCons, NomMed)*
- *Medicament(NomMed, NomSubst, Prix)*

Une personne (patient ou médecin) a un numéro de sécurité social (Noss), un nom, un prénom et une adresse. Chaque consultation a un numéro unique (NoCons), une date, le numéro de sécurité social du patient et du médecin ainsi que le symptôme du patient (nous supposons que chaque patient n’a qu’un seul symptôme au moment d’une consultation). Les médicaments prescrit après une consultation sont stockés dans la table Prescription (un traitement est composé de toutes les prescriptions associées à la même consultation). Pour chaque médicament on connaît son nom, la substance active et le prix.

Questions :

1. (1,5 points) Conception : Dessinez un schéma Entité-Association du schéma relationnel précédent. Indiquez également les cardinalités des associations en prenant en compte les clés et la contrainte qu’une clé étrangère ne peut pas avoir la valeur NULL.
   Solution :

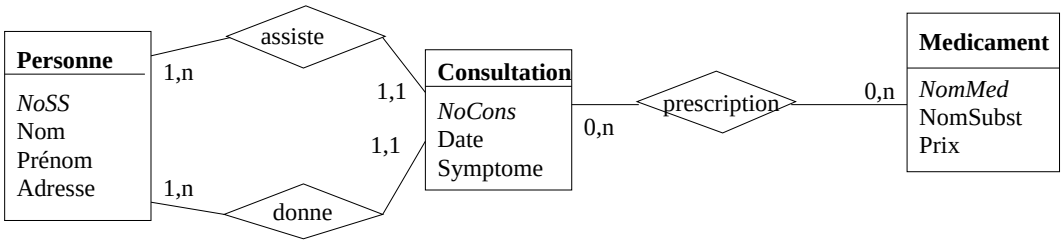

FIG. 1 – *Schéma entités-associations*

2. (6 points) Requêtes :
   (a) (1 point) On cherche les noms des patients avec une toux sèche et les noms des médicaments prescrits (algèbre relationnelle).
   Solution :

$$\pi_{Nom,NomMed}(\sigma_{Symptome=\text{"Toux seche"}} Consultation \bowtie Prescription \bowtie_{NossPat=Noss} Personne)$$

   (b) (2 points) On cherche les noms des patients qui ont consulté deux médecins différents pour une toux sèche le même jour. (SQL, calcul domaine).
   Solution :

```
select P.Nom
from  Consultation A, Consultation B, Personne P
where A.Date = B.Date
  and A.NossPat=B.NossPat
  and A.NossMed <> B.NossMed
  and A.Symptome="Toux Seche"
  and B.Symptome="Toux Seche"
  and P.Noss = P.NossPat
```

$$\{N \mid \exists C1,D,P,M1,,C2,M2,NN,A \quad (Consultation(C1,D,''Touxseche'',P,M1) \wedge Consultation(C2,D,"TouxSeche",P,M2) \wedge Personne(P,N,NN,A) \wedge M1! = M2)\}$$

(c) (1 point) On cherche les noms des médecins qui n'ont jamais prescrit de médicament avec la substance active NOSCAPINE contre la toux sèche (SQL).
Solution :

```
select distinct Nom
 from Personne, Consultation
where Noss=NossMed
  and NossMed not in (select NossMed
                        from Consultation, Prescription, Medicament
                       where Consultation.NoCons = Prescription.NoCons
                         and Medicament.NoMed = Prescription.NoMed
                         and Symptome="Toux Seche"
                         and NomSubst="NOSCAPINE")
```

(d) (1 point) On cherche le(s) médicament(s) le(s) moins cher(s) avec la substance NOSCAPINE (SQL)
Solution :

```
select A.NomMed
  from Medicament A
 where A.NomSubst="NOSCAPINE"
   and not exists (select *
                     from Medicament B
                      and B.NomSubst="NOSCAPINE"
                      and B.Prix < A.Prix)
```

(e) (1 point) On cherche pour chaque substance son nom ainsi que le prix moyen et le prix maximal des médicaments qui la contiennent (SQL)
Solution :

```
select NomSubst, avg(prix), max(prix)
  from Medicament
 group by NomSubst
```

2 Organisation physique (2 points)

La projection sur NoCons de la table Prescrition donne l'ensemble suivant:

{1, 3, 7, 15, 2, 8, 77, 6, 10, 105, 23, 13, 38, 44, 89, 19, 5, 18, 37, 49, 60, 153, 100, 28, 50, 78, 4, 9, 46}

Question : (2 points) On construit un index sur cet attribut: arbre B (variante B+ vue en cours) d'ordre 3. Donner l'arbre correspondant (ne montrer que les clés dans les feuilles).

Solution : *Figure 2.*

3 Optimisation (5,5 points)

Questions :

1. (1 point) Exprimer en SQL la requête 1.2.a : noms des patients avec une toux sèche et noms des médicaments prescrits.
   Solution :

```
select Nom,NomMed
from Consultation,Prescription,Personne
where Symptome='Toux sèche'
And NossPat=Noss
And Consultation.NoCons=Prescription.NoCons
```

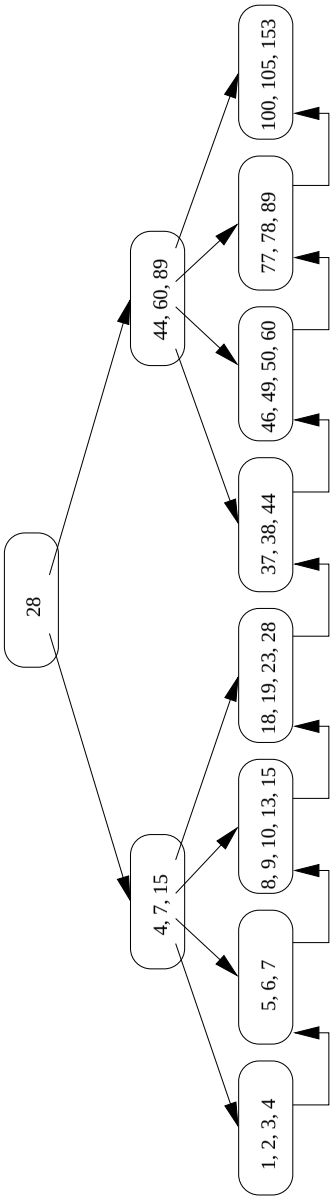
28
4, 7, 15
44, 60, 89
1, 2, 3, 4
5, 6, 7
8, 9, 10, 13, 15
18, 19, 23, 28
37, 38, 44
46, 49, 50, 60
77, 78, 89
100, 105, 153

FIG. 2 – *arbre B*

2. (0,5 point) Donner sous forme arborescente le plan d'exécution logique (arbre de requête).
   Solution : *Voir réponse à la requête 1.2.a*
3. (2 points) On suppose que chaque table est indexée sur sa clé primaire. Voici le plan d'exécution fourni par Oracle. Expliquer en *détail* ce plan.

```
Execution Plan
----------------------------------------------------------
    0      SELECT STATEMENT Optimizer=CHOOSE
    1       NESTED LOOPS
    2         NESTED LOOPS
    3           TABLE ACCESS (FULL) OF 'CONSULTATION'
    4           TABLE ACCESS (BY INDEX ROWID) OF 'PERSONNE'
    5             INDEX (UNIQUE SCAN) OF 'PK_PERSONNE' (UNIQUE)
    6         TABLE ACCESS (BY INDEX ROWID) OF 'PRESCRIPTION'
    7           INDEX (UNIQUE SCAN) OF 'PK_PRESCRIPTION' (UNIQUE)
```

   Solution : *L'algorithme choisi est celui d'une double boucle imbriquée (on a deux jointures). Le système commence par un parcours séquentiel de la table Consultation pour faire la sélection "toux sèche". Pour chaque nuplet retenu, la valeur de l'attribut NoCons est projetée. Elle servira pour la deuxième jointure. L'attribut NossPat sert de clé d'accès à l'index sur Noss de la table Personne (jointure entre Consultation et Personne). La traversée de l'index donne un rowid de personne. Le nuplet de la table Personne est obtenu par accès direct. On projette sur l'attribut "Nom". Pour chaque nuplet obtenu, la valeur de l'attribut NoCons sert de clé d'accès à l'index sur la table Prescription sur l'attribut "NoCons" (deuxième jointure). La traversée de l'index donne un rowid unique de Prescription. L'accès par rowid à la table Prescription donne un nuplet. On projette sur l'attribut NomMed.*
4. (2 points) On rajoute un index sur la table Consultation sur l'attribut Symptome par la commande:

```
SQL> create index sympt_idx on consultation(symptome);
```

   Que devient le plan d'exécution ? Donner le nouveau plan sous forme arborescente ou sous forme EXPLAIN et l'expliquer.
   Solution :

```
Execution Plan
----------------------------------------------------------
    0      SELECT STATEMENT Optimizer=CHOOSE
    1       NESTED LOOPS
    2         NESTED LOOPS
    3           TABLE ACCESS (BY INDEX ROWID) OF 'CONSULTATION'
    4             INDEX (RANGE SCAN) OF 'SYMPT_IDX' (NON-UNIQUE)
    5           TABLE ACCESS (BY INDEX ROWID) OF 'PERSONNE'
    6             INDEX (UNIQUE SCAN) OF 'PK_PERSONNE' (UNIQUE)
    7         TABLE ACCESS (BY INDEX ROWID) OF 'PRESCRIPTION'
    8           INDEX (UNIQUE SCAN) OF 'PK_PRESCRIPTION' (UNIQUE)
```

   On remplace le parcours séquentiel pour sélectionner les consultations avec pour symptôme "toux sèche" par une traversée d'index et un accès direct.

4 Concurrence (5 points)

Parmi les programmes qui s'exécutent dans le système de gestion des dossiers médicaux, il y a les trois suivants :

- *PrixMoyen(NoCons)*, qui calcule le prix moyen des médicaments pour une consultation de numéro donné.
- *ChangerPrix(NomMed, Prix)*, qui modifie le prix d'un médicament de nom donné, avec un nouveau prix donné.
- *Generiques(NoCons)*, qui substitue chaque médicament prescrit lors d'une consultation donnée, par son équivalent générique.

On considère que les enregistrements qui interviennent dans l'exécution sont des nuplets des relations présentées auparavant et que seuls les nuplets de ces relations sont des enregistrements de la base de données (par exemple, les informations sur les médicaments génériques équivalents ne sont pas dans la base de données). L'accès à un nuplet dont on connaît la clé peut se faire directement.

L'exécution suivante est reçue par le système de gestion des dossiers médicaux :

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

Questions :

1. (1,5 points) Montrer que cette exécution est compatible avec le résultat de l'exécution concurrente des trois transactions suivantes : *PrixMoyen(123), ChangerPrix('Aspirine', 1.6), Generiques(123)*. On sait que la consultation 123 prescrit de la pénicilline et de l'aspirine. Justifiez votre réponse et donnez la signification de chaque enregistrement qui intervient dans H.
   Solution : *Les transactions extraites de l'exécution* **H** *sont :*
   T_1*:* $r_1[x]$ $r_1[z]$ $r_1[u]$ $r_1[y]$ c_1
   T_2*:* $r_2[y]$ $w_2[y]$ c_2
   T_3*:* $r_3[x]$ $w_3[x]$ $r_3[u]$ $w_3[u]$ c_3
   Dans PrixMoyen(123)*, on lit les nuplets de la table* `Prescription` *pour la consultation 123 (il y a deux, car on y prescrit 2 médicaments) et pour chaque prescription on lit de* `Medicament` *le médicament en question pour récupérer le prix. Il y a donc dans cette transaction 4 lectures, il s'agit donc de la transaction* T_1.
   Dans ChangerPrix('Aspirine', 1.6)*, on accède à* `Medicament` *pour lire le nuplet pour 'Aspirine' et modifier le prix. Il y a donc une lecture et une écriture d'un même enregistrement, donc il s'agit de la transaction* T_2.
   Dans Generiques(123)*, on lit les 2 nuplets de la table* `Prescription` *pour la consultation 123 et on modifie pour chacun le médicament. Il y a donc lecture-écriture pour chacun des deux nuplets, il s'agit donc de la transaction* T_3.
   En conclusion, H est bien l'exécution concurrente des 3 transactions, avec : x=prescription pénicilline dans la consultation 123, y=médicament aspirine, z=médicament pénicilline, u=prescription aspirine dans la consultation 123
2. (1 point) Vérifiez si **H** est sérialisable en identifiant les conflits et en construisant le graphe de sérialisation.
   Solution : *Les conflits :*
   sur x *:* $r_1[x]$-$w_3[x]$
   sur y *:* $r_1[y]$-$w_2[y]$
   sur z *: pas de conflit*
   sur u *:* $r_1[u]$-$w_3[u]$
   Le graphe de sérialisation ne contient que des arcs T_1 T_2 *et* T_1 T_3*, donc il n'y a pas de cycle. En conclusion,* **H** *est sérialisable.*
3. (0,5 points) L'exécution **H** est-elle stricte? Justifiez votre réponse.
   Solution : *Après chacune des écritures de H il n'y a plus aucune opération sur le même enregistrement (ni lecture, ni écriture), donc H est stricte.*
4. (2 points) Quelle est l'exécution obtenue par verrouillage à deux phases à partir de **H**? L'algorithme d'estampillage simple accepte-t-il cette exécution sans rejets?
   On considère qu'il existe deux verrous pour chaque enregistrement, un de lecture et un d'écriture. On rappelle que les verrous de lecture peuvent être partagés entre autant de transactions que nécessaire, par contre on n'accepte jamais que le verrou de lecture et celui d'écriture d'un enregistrement soit accordés à des transactions différentes.
   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 *:* $r_1[x]$ $r_2[y]$ $r_3[x]$ $r_1[z]$ $r_1[u]$ $w_3[x]$ $r_1[y]$ $r_3[u]$ $w_2[y]$ c_2 c_1 $w_3[u]$ c_3
   $r_1[x]$*,* $r_2[y]$ *s'exécutent, en prenant les verrous de lecture*
   $r_3[x]$ *partage le verrou de lecture sur* x *avec* $r_1[x]$ *et s'exécute*
   $r_1[z]$*,* $r_1[u]$ *s'exécutent, en prenant les verrous de lecture*
   $w_3[x]$ *bloquée par* $r_1[x]$*, donc* T_3 *bloquée*
   $r_1[y]$ *partage le verrou de lecture sur* y *avec* $r_2[y]$ *et s'exécute*
   $r_3[u]$ *bloquée, car* T_3 *bloquée*
   $w_2[y]$ *bloquée par* $r_1[y]$*, donc* T_2 *bloquée*
   c_2 *bloquée car* T_2 *bloquée*
   c_1 *s'exécute et relâche les verrous de* T_1*.* $w_3[x]$ *peut obtenir le verrou sur* x *et s'exécuter, aussi* $r_3[u]$ *est débloquée, prend le verrou de lecture et s'exécute. Egalement,* $w_2[y]$ *peut obtenir le verrou sur* y *et s'exécuter, suivi de* c_2*, qui relâche tous les verrous de* T_2.
   $w_3[u]$ *et* c_3 *s'exécutent*
   Le résultat final est donc **H'** *:* $r_1[x]$ $r_2[y]$ $r_3[x]$ $r_1[z]$ $r_1[u]$ $r_1[y]$ c_1 $w_3[x]$ $r_3[u]$ $w_2[y]$ c_2 $w_3[u]$ c_3
   Pour l'estampillage simple: comme H ne contient que des conflits qui respectent l'ordre des estampilles, elle sera acceptée telle quelle par l'estampillage.