SGBD B7 - UV 19786 (soir) et UV 21928 (ICPJ)
EXAMEN, 1re session – 25 juin 2005
(tout document écrit autorisé)

1 Modèle relationnel (6,5 points)

Un institut de sondage a réalisé une enquête sur les choix électoraux des français. Les résultats de cette enquête sont modélisés dans une base de données contenant les relations suivantes (les attributs clés primaires sont soulignés) :

- `PERSONNE(numPers,age,sexe,numCat)` : chaque personne est identifiée par son numéro (`numPers`), possède un `age`, un `sexe` et un numéro de catégorie socio-professionnelle (`numCat`).
- `CATEGORIE(numCat,intitule)` : chaque catégorie socio-professionnelle est identifiée par un numéro (`numCat`) et est décrite par un `intitule` (par exemple «ouvrier»).
- `QUESTION(numQ,description)` : chaque question de l'enquête est identifiée par un numéro (`numQ`) et est décrite par l'attribut `description`.
- `AVIS(numA,numQ,numPers,reponse)` : chaque avis d'une personne (`numPers`) sur une question (`numQ`) est représenté dans l'attribut `reponse`. Les avis sont identifiés par l'attribut `numA`.

Questions :

1. (0,5 point) Donnez deux commandes `insert` sur la relation `PERSONNE` qui, ensemble, ne peuvent être acceptées par le SGBD. Expliquez pourquoi.

   Solution : *Les commandes suivantes :*

   ```
   insert into PERSONNE values(1,53,'feminin',3);
   insert into PERSONNE values(1,21,'feminin',2);
   ```

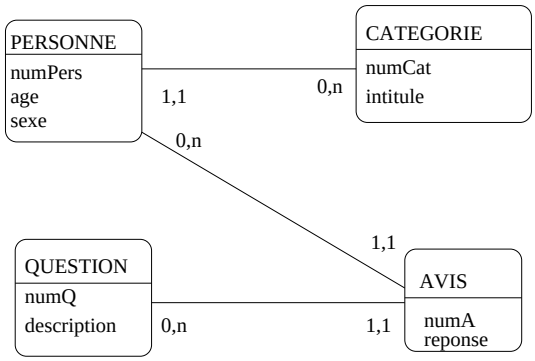
   ne respectent pas la contrainte de clé primaire sur l'attribut numPers.

2. (0,5 point) Une personne peut-elle avoir plusieurs avis sur la même question ?

   Solution : *Oui, car il n'y a pas de contrainte de clé primaire sur les attributs `numQ` et `numPers` dans la relation `AVIS`.*

3. (1 point) Dessinez un schéma Entité-Association du schéma relationnel précédent. Indiquez également les cardinalités minimales et maximales des associations en prenant en compte les clés et la contrainte qu'une clé étrangère ne peut pas avoir la valeur NULL.

   Solution :



Donnez les requêtes suivantes, dans le bon language :

1. (1,5 point) Les numéros des personnes dans la catégorie "cadre" de sexe "féminin" (en SQL, en algèbre et en calcul domaine).

   Solution : *SQL :*

   ```
   select numPers from PERSONNE, CATEGORIE
   where sexe="feminin"
   and PERSONNE.numCat=CATEGORIE.numCat
   and intitule="cadre".
   ```

Algèbre :

$$\Pi_{numPers}((\sigma_{sexe="feminin"}(PERSONNE)) \bowtie (\sigma_{intitule="cadre"}(CATEGORIE))).$$

Calcul domaine :

$$\{n \mid \exists a \exists c\, PERSONNE(n, a, "feminin", c) \wedge CATEGORIE(c, "cadre")\}.$$

2. (1 point) Les numéros des personnes de strictement plus de 25 ans n'ayant jamais donné leur avis (en SQL et en calcul n-uplet).
   Solution : *SQL :*

```
select numPers from PERSONNE
where age > 25
and not exists
          (select * from AVIS
           where PERSONNE.numPers=AVIS.numPers).
```

   Calcul n-uplet :

$$\{p.numPers | PERSONNE(p) \wedge p.age > 25 \wedge \neg \exists a(AVIS(a) \wedge a.numPers = p.numPers)\}.$$

3. (1 point) Pour chaque numéro de question, le nombre d'avis pour cette question, uniquement si ce nombre d'avis est strictement supérieur à 30 (en SQL).
   Solution :

```
select numQ,count(numA) from AVIS
group by numQ
having count(numA)>30.
```

4. (1 point) Sans préjuger de la réponse à la question 2 de l'exercice précédent, on supposera ici qu'une personne peut répondre plusieurs fois à la même question. Donnez alors le nombre de personnes ayant répondu "oui" à la question "referendum européen", en ne comptant pas plusieurs fois la même personne (en SQL).
   Solution :

```
select count(distinct numPers)
from AVIS, QUESTION
where QUESTION.intitule="referendum européen"
and AVIS.reponse="oui"
and AVIS.numQ=Question.numQ.
```

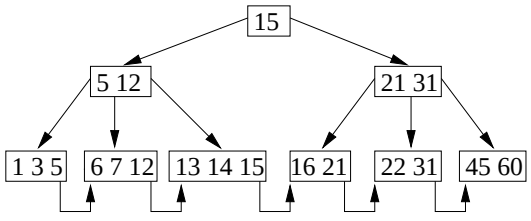
2 Organisation physique (3,5 points)

La projection sur `numPers` de la table `PERSONNE` donne l'ensemble suivant :

{1,5,3,6,22,31,7,12,45,21,60,13,14,15,16}.

Question :

1. (2 points) On construit un index sur cet attribut, de type arbre B+ d'ordre 2 (l'arbre appelé simplement arbre B dans le cours ICPJ). **On utilise une variante de l'arbre B+ dans laquelle les n-uplets sont stockés dans les feuilles de l'arbre**. Dessinez l'arbre correspondant à l'insertion des valeurs précédentes, en respectant l'ordre donné (ne montrer dans les feuilles de l'arbre que les clés).
   Solution :



2. (1 point) Combien de pages le SGBD doit-il lire pour trouver les attributs de la personne 12 ? Indiquez précisément les pages lues.
   Solution : *3 : (15), (5,12) puis (6,7,12), où se trouvent les attributs recherchés.*
3. (0,5 point) Même question pour la personne 55, qui ne figure pas dans la base.
   Solution : *3 également : (15), (21,31) puis (45,60), pour se rendre compte que 55 n'existe pas.*

3 Optimisation (5 points)

Questions : Soit la requête SQL suivante, calculant l'âge des personnes de strictement plus de 40 ans ayant répondu "oui" à la question "baisse des impôts" :

```
select age from PERSONNE, QUESTION, AVIS
where AVIS.numPers=PERSONNE.numPers
and AVIS.numQ=QUESTION.numQ
and AVIS.reponse="oui"
and PERSONNE.age > 40
and QUESTION.description="baisse des impôts".
```

1. (2 points) Donnez une requête algébrique optimisée correspondante, en expliquant vos choix.
   Solution : *Ne sachant rien sur la taille des relations, aucun choix particulier ne peut être effectué sur l'ordre des opérations de jointure. Par contre, il est naturel d'effectuer le plus tôt possible les sélections. Une requête algébrique optimisée serait alors :*

   $$\Pi_{age}(((\sigma_{age>40}(PERSONNE)) \bowtie (\sigma_{reponse="oui"}(AVIS))) \bowtie (\sigma_{description="baisse..."}(QUESTION))).$$

   ***Non demandé** : on peut également réduire le nombre d'attributs pendant le calcul en projettant sur les seuls attributs utiles :*

   $$\Pi_{age}((\Pi_{age,numPers}(\sigma_{age>40}(PERSONNE))) \bowtie (\Pi_{numQ,numPers}(\sigma_{reponse="oui"}(AVIS))) \bowtie (\Pi_{numQ}(\sigma_{description="baisse..."}(QUESTION)))).$$

2. On considère la requête SQL suivante :

   ```
   select PERSONNE.numPers,reponse from PERSONNE, AVIS
   where AVIS.numPers=PERSONNE.numPers
   and PERSONNE.age > 40.
   ```

   Voici le plan d'exécution EXPLAIN donné par le SGBD :

   ```
   Execution Plan
   ----------------------------------------------------------
       0      SELECT STATEMENT
       1      NESTED LOOPS
       2        TABLE ACCESS (BY ROWID) PERSONNE
       3          INDEX (RANGE SCAN) PERSONNE_IDX
       4        TABLE ACCESS (BY ROWID) AVIS
       5          INDEX (RANGE SCAN) AVIS_IDX
   ```

   (a) (1 point) Indiquez pour chaque relation si un index est utilisé et, le cas échéant, sur quel attribut porte l'index.

   (b) (2 points) Expliquez le fonctionnement de ce plan d'exécution, en précisant les algorithmes utilisés.

   Solution : *D'après le plan d'exécution, on voit qu'un index est disponible (ligne `INDEX ...`) pour les relations `PERSONNE` et `AVIS`. Pour la relation `PERSONNE`, l'accès est un `RANGE SCAN` donc plusieurs résultats sont attendus. Il s'agit probablement de la recherche des personnes ayant un âge strictement supérieur à 40 ans. Pour la relation `AVIS`, l'accès est un `RANGE SCAN`, donc plusieurs résultats sont attendus. Comme de plus la requête est une jointure, il s'agit probablement de l'attribut `numPers`.*

   – *`PERSONNE` : index sur l'attribut `age`.*
   – *`AVIS` : index sur l'attribut `numPers`.*

   L'algorithme choisi pour effectuer la jointure est une boucle imbriquée (`NESTED LOOPS`). Le système commence par un interrogation de l'index sur `PERSONNE` pour obtenir les adresses des n-uplets dont l'âge est strictement plus grand que 40 ans. Pour une adresse donnée, le n-uplet p correspondant est obtenu. Le système extrait la valeur de l'attribut `numPers` puis va interroger l'index de `AVIS`. Il obtient ainsi les adresses des n-uplets de `AVIS` en jointure avec le n-uplet p. Pour chaque adresse, le n-uplet correspondant a est obtenu. Enfin, la jointure des n-uplets p et a est calculée, et le résultat est projeté sur les attributs `numPers` et `reponse`.

4 Concurrence (5 points)

L'exécution suivante est reçue par le système de l'institut de sondage :
H : $r_1[x]\ r_2[y]\ w_1[x]\ r_3[y]\ r_2[x]\ w_3[y]\ r_2[z]\ c_1\ r_3[z]\ w_2[z]\ c_2\ w_3[z]\ c_3$

1. (1 point) Parmi les programmes qui s'exécutent dans le système, il y a *ModifierAvis(description_question, avis_donné, nouvel_avis)*, qui modifie pour la question *description_question* seulement les avis *avis_donné* en *nouvel_avis*. Si les enregistrements de **H** sont des nuplets des relations de la base de données, montrez (en justifiant votre réponse) quelles transactions de **H** pourraient provenir de *ModifierAvis*.

   On suppose qu'une description de question donne accès direct aux nuplets QUESTION qui lui correspondent. Aussi, un numéro de question donne accès direct aux nuplets AVIS sur la question.

   Solution : *Les transactions extraites de l'exécution* **H** *sont :*
   T_1 *:* $r_1[x]\ w_1[x]\ c_1$
   T_2 *:* $r_2[y]\ r_2[x]\ r_2[z]\ w_2[z]\ c_2$
   T_3 *:* $r_3[y]\ w_3[y]\ r_3[z]\ w_3[z]\ c_3$
   Dans ModifierAvis*, on lit les nuplets de QUESTION ayant* description_question *et on récupère les numéros de question. Ensuite, pour chaque numéro de question on lit les nuplets AVIS et pour ceux ayant* avis_donné *on modifie le nuplet. Toutes les transactions de* **H** *ont des mises-à-jour, donc il faut au moins une lecture de question avant. La seule transaction qui correspond est* T_2.

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

   Solution : *Les conflits :*
   sur x *:* $w_1[x]$-$r_2[x]$
   sur y *:* $r_2[y]$-$w_3[y]$
   sur z *:* $r_2[z]$-$w_3[z]$, $r_3[z]$-$w_2[z]$, $w_2[z]$-$w_3[z]$
   Le graphe de sérialisation contient des arcs $T_1 \rightarrow T_2$, $T_2 \rightarrow T_3$ *et* $T_3 \rightarrow T_2$. *Il y a un cycle* T_2 - T_3 - T_2, *donc* **H** *n'est pas sérialisable.*

3. (1 point) Montrez que **H** n'évite pas les annulations en cascade à cause de la position d'un des *Commit*. En déplaçant ce *Commit* peut-on obtenir une exécution stricte ? Justifiez votre réponse.

   Solution : *La seule écriture suivie d'une lecture du même enregistrement est* $w_1[x]$. *La lecture* $r_2[x]$ *intervient avant la validation de* $w_1[x]$, *donc* **H** *n'évite pas les annulations en cascade.*

   En avançant c_1 *avant* $r_2[x]$, **H** *évitera les annulations en cascade. Pour qu'elle soit stricte, les écritures d'un même enregistrement doivent être séparées par des validations. Le seul cas de ce type est celui des écritures sur* z, $w_2[z]$ *et* $w_3[z]$, *qui sont séparées dans* **H** *par* c_2, *donc l'exécution est stricte.*

4. (2 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. 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 *:* $r_1[x]\ r_2[y]\ w_1[x]\ r_3[y]\ r_2[x]\ w_3[y]\ r_2[z]\ c_1\ r_3[z]\ w_2[z]\ c_2\ w_3[z]\ c_3$
   $r_1[x]$, $r_2[y]$ *s'exécutent, en prenant les verrous de lecture*
   $w_1[x]$ *prend le verrou d'écriture et s'exécute (pas de conflit avec* $r_1[x]$*)*
   $r_3[y]$ *partage le verrou de lecture sur* y *avec* $r_2[y]$ *et s'exécute*
   $r_2[x]$ *bloquée par* $w_1[x]$, *donc* T_2 *bloquée*
   $w_3[y]$ *bloquée par* $r_2[y]$, *donc* T_3 *bloquée*
   $r_2[z]$ *bloquée, car* T_2 *bloquée*
   c_1 *s'exécute et relâche les verrous de* T_1, *donc* $r_2[x]$ *et* $r_2[z]$ *peuvent s'exécuter, tandis que* $w_3[y]$ *reste bloquée*
   $r_3[z]$ *bloquée, car* T_3 *bloquée*
   $w_2[z]$ *s'exécute*
   c_2 *s'exécute et relâche les verrous de* T_2, *donc* $w_3[y]$ *et* $r_3[z]$ *s'exécutent*
   $w_3[z]$ c_3 *s'exécutent*
   Le résultat final est donc **H'** *:* $r_1[x]\ r_2[y]\ w_1[x]\ r_3[y]\ c_1\ r_2[x]\ r_2[z]\ w_2[z]\ c_2\ w_3[y]\ r_3[z]\ w_3[z]\ c_3$