NFP107 : Systèmes de gestion de bases de données
EXAMEN, juin 2007
(tout document écrit autorisé)

1 Modèle relationnel (12 points)

Un institut de sondages utilise une base de données relationnelle pour répertorier les intentions de vote pour les élections présidentielles. La base utilisée a le schéma suivant :

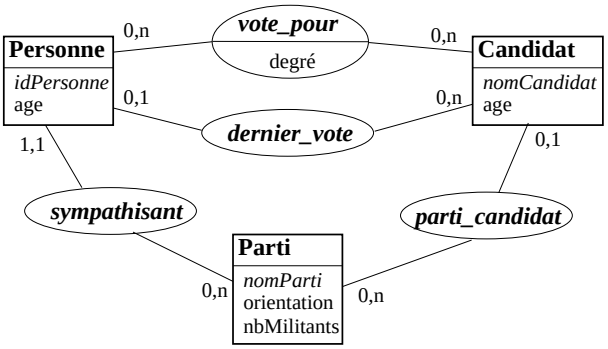
- `Candidat(nomCandidat, nomParti, âge);`
- `Personne(idPersonne, âge, dernierVote, sympathisant);`
- `Vote(idPersonne, nomCandidat, degré);`
- `Parti(nomParti, orientation, nbMilitants);`

Les clés primaires sont soulignées. Un candidat, identifié par son nom, est caractérisé par son âge et par le nom du parti qu'il représente (sauf s'il est candidat indépendant). Pour une personne dont on recueille l'intention de vote, on connaît son identifiant unique, son âge, le candidat pour lequel il/elle a voté aux dernières élections présidentielles (s'il/elle a voté) et le parti dont il/elle est sympathisant. Un parti est identifié par son nom et on connaît son orientation (gauche, droite) et le nombre de militants. Une intention de vote spécifie le nom de la personne et celui du candidat, ainsi que le degré de décision de la personne (entre 0 et 100%). Une personne peut donc être partagée dans son intention de vote entre plusieurs candidats.

Modèle conceptuel

1. (2,5 points) Construisez le schéma conceptuel (entité-association, Merise, etc.) de cette base de données en précisant bien les multiplicités des associations.

   Solution :



2. (1 point) Est-il possible qu'une personne indique à la fois l'intention de voter pour un candidat de gauche et pour un candidat de droite ? Peut-on avoir des partis sans candidat et sans sympathisant enregistré ?

   Solution : *(a) Oui, car une même personne peut exprimer des intentions de vote pour plusieurs candidats, avec des degrés de décision variés (la clé de* `Vote` *est le couple* `idPersonne-nomCandidat`*). Ensuite, il n'y a pas de contrainte concernant le lien entre un candidat et son orientation politique (celle de sont parti).*

   (b) Oui, voir les multiplicités dans le schéma conceptuel.

3. (1 point) Donnez un exemple de contrainte sur les données de la base, qui ne peut pas être imposée par le modèle relationnel.

   Solution : *Par exemple, la somme des pourcentages de décision d'une personne pour les différents candidats entre lesquels elle hésite ne doit pas dépasser 100%.*

Requêtes

1. (1 point) Les partis représentés par des candidats de moins de 50 ans (SQL et algèbre).

   Solution : *SQL :*

```
select distinct nomParti from Candidat
where âge<50;
```

Algèbre :

$$\Pi_{nomParti}(\sigma_{age<50}(Candidat)).$$

2. (1,5 point) Les partis ayant des sympathisants sûrs (100%) de leur vote (SQL et algèbre).
   Solution : *SQL :*

```
select distinct sympathisant from Personne, Vote
where Personne.idPersonne=Vote.idPersonne and
      degré = 100;
```

   Algèbre :

$$\Pi_{sympathisant}(Personne \bowtie (\sigma_{degre=100}(Vote))).$$

3. (2 points) Les noms des candidats aux dernières élections qui ne se présentent plus aux élections actuelles (par rapport aux données présentes dans la base) (SQL et algèbre).
   Solution : *SQL :*

```
select distinct dernierVote from Personne
where dernierVote not in (select nomCandidat from Vote);
```

   Algèbre :

$$\rho_{dernierVote/nomCandidat}(\Pi_{dernierVote}(Personne)) - \Pi_{nomCandidat}(Vote).$$

4. (1,5 points) L'identifiant et l'âge des sympathisants de gauche ayant l'intention à plus de 50% de voter pour le même candidat qu'au dernières élections (SQL).
   Solution :

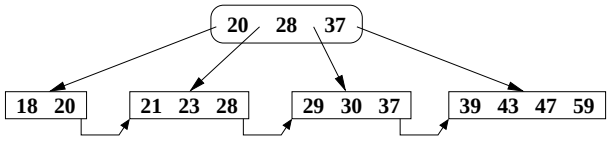
```
select Personne.idPersonne, âge
from Personne, Vote, Parti
where sympathisant=nomParti and
      orientation='gauche' and
      Personne.idPersonne=Vote.idPersonne and
      degré>50 and
      nomCandidat=dernierVote;
```

5. (1,5 points) Pour chaque candidat, son nom et l'âge moyen des électeurs qui votent pour lui, en considérant comme vote seulement une intention à au moins 50% (SQL).
   Solution : *SQL :*

```
select nomCandidat, avg(âge)
from Personne, Vote
where degré >= 50 and
      Personne.idPersonne = Vote.idPersonne
group by nomCandidat
```

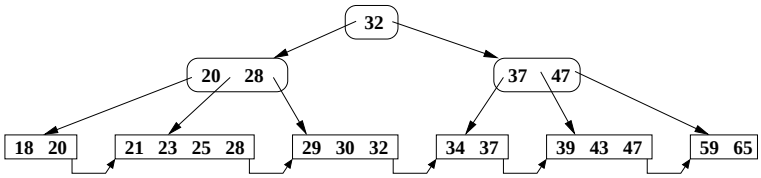
2 Organisation physique et optimisation (5 points)

Index La commande `create index âge_idx on Personne(âge)` crée un index de type arbre B+ d'ordre 2 sur l'âge des personnes interrogées. L'index ne stocke pas les n-uplets de *Personne*, mais seulement des pointeurs vers ces n-uplets sur disque. A un moment donné, l'index a la structure suivante :



1. (1,5 points) Montrer l'évolution de l'index après insertion successive de personnes ayant pour âge les valeurs 34, 65, 25 et 32.

   Solution :



2. (1 point) Quel est le coût de la lecture, à travers le nouvel index, de toutes les personnes ayant un âge entre 22 et 39 ans inclus. On considère que la lecture d'un n-uplet de *Personne*, étant donnée son adresse récupérée dans l'index, coûte une lecture de page.

   Solution : *La recherche de la feuille pour 22 coûte 3 lectures de page dans le nouvel index, ensuite il faut lire encore 3 feuilles pour trouver tous les âges entre 22 et 39. Il y a en tout 9 n-uplets dans l'intervalle. Le coût total est donc 3+3+9=15 lectures de pages.*

Optimisation Soit la requête suivante, qui donne les noms des candidats pour lesquels votent des personnes de moins de 21 ans :

```
select nomCandidat from Personne, Vote
where Personne.idPersonne=Vote.idPersonne and âge<21;
```

1. (1,5 points) Donnez le plan d'exécution physique pour cette requête si seul l'index du point précédent (sur l'âge des personnes) existe. Expliquez et justifiez ce plan, présenté sous forme d'arbre ou de plan EXPLAIN.

   Solution : *Accès à Personne à travers l'index sur l'âge (sélection) et jointure avec Vote par boucles imbriquées avec parcours séquentiel, suivi de la projection sur nomCandidat.*

   Un autre plan possible : comme ci-dessus, mais au lieu des boucles imbriquées on fait un tri-fusion.

2. (1 point) La même chose si en plus on dispose d'un index sur *idPersonne* dans la table *Vote*.

   Solution : *Pareil, que la première variante du point précédent, sauf que la jointure se fait avec traversée de l'index sur Vote.idPersonne.*

3 Concurrence (3 points)

Soit l'exécution concurrente de trois transactions suivante, où x et y sont des n-uplets des relations de la base de données.

$r_1[x]r_2[y]r_3[x]w_1[x]w_2[y]r_3[y]c_2r_1[y]c_3w_1[y]c_1$

1. (1,5 points) Laquelle des trois transactions pourrait provenir de l'exécution de la commande `update Candidat set âge=âge+1 where âge>70`?

   Solution : *Les trois transactions sont :*

   $T_1 : r_1[x]w_1[x]r_1[y]w_1[y]c_1$

   $T_2 : r_2[y]w_2[y]c_2$

   $T_3 : r_3[x]r_3[y]c_3$

   La commande `update` *lit les candidats de plus de 70 ans et incrémente leur âge, donc fait des lectures-écritures sur un ou plusieurs n-uplets de Candidat. Les transactions T_1 et T_2 satisfont cette condition.*

2. (1,5 points) Trouver l'exécution produite par verrouillage à deux phases si les verrous d'une transaction sont relâchés après son *Commit*. Les verrous de lecture sont partageables, ceux d'écriture sont exclusifs. Les opérations bloquées en attente de verrou s'exécutent en priorité quand le verrou devient disponible, en respectant l'ordre de blocage.

   Solution : $r_1[x]$ *s'exécute en prenant le verrou de lecture sur* x
   $r_2[y]$ *s'exécute en prenant le verrou de lecture sur* y
   $r_3[x]$ *partage le verrou de lecture sur* x *avec* $r_1[x]$ *et s'exécute*
   $w_1[x]$ *bloquée par* $r_3[x]$*, donc* T_1 *bloquée*
   $w_2[y]$ *s'exécute en prenant le verrou d'écriture sur* y*, pas de conflit avec* $r_2[y]$
   $r_3[y]$ *bloquée par* $w_2[y]$*, donc* T_3 *bloquée*
   c_2 *s'exécute et relâche les verrous de* $T_2 \Rightarrow r_3[y]$ *s'exécute en prenant le verrou de lecture sur* y
   $r_1[y]$ *bloquée, car* T_1 *bloquée*
   c_3 *s'exécute et relâche les verrous de* $T_3 \Rightarrow w_1[x]$ *et* $r_1[y]$ *sont débloquées et s'exécutent*
   $w_1[y]$ *et* c_1 *s'exécutent*
   Résultat : $r_1[x]r_2[y]r_3[x]w_2[y]c_2r_3[y]c_3w_1[x]r_1[y]w_1[y]c_1$