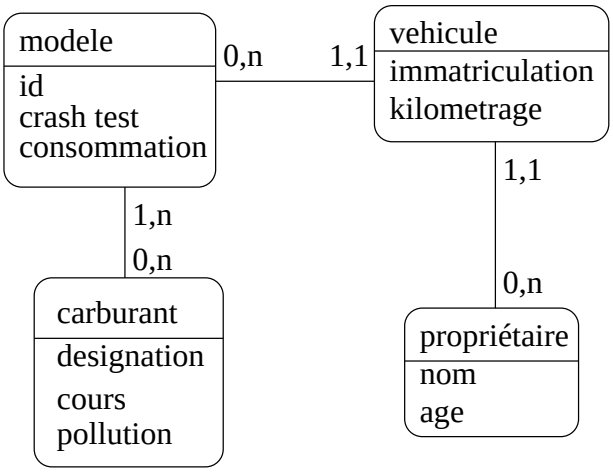
SGBD B7 - UV 19786 (soir) et UV 21928 (ICPJ)
EXAMEN, 2e session – septembre 2005
(tout document écrit autorisé)

1 Modèle relationnel (6,5 points)

Une agence environnementale s'intéresse à la problématique des transports. Pour cela, les informations concernant les véhicules d'un échantillon de la population ont été collectées dans une base de données. Elle respecte le schéma Entité/Relation suivant :



Compréhension du schéma :

1. (0,5 point) Dans ce schéma, un véhicule peut-il être de plusieurs modèles à la fois ?
   Solution : *Non (cardinalité maximale 1 sur* `véhicule-modèle`*).*
2. (0,5 point) Un propriétaire possède-t-il nécessairement un véhicule ?
   Solution : *Non (cardinalité minimale 0 sur* `propriétaire-véhicule`*).*
3. (0,5 point) Un modèle de véhicule peut-il utiliser plusieurs carburants à la fois ?
   Solution : *Oui (cardinalité maximale n sur* `modèle-carburant`*).*
4. (0,5 point) Le "Glubozol"[1] est un carburant qui n'est utilisé par aucun véhicule. Peut-on le représenter dans la base ?
   Solution : *Oui (cardinalité minimale 0 sur* `carburant-modèle`*).*
5. (0,5 point) La "bicyclette" est un modèle de véhicule n'utilisant pas de carburant. Peut-on la représenter dans ce modèle ?
   Solution : *Non (cardinalité minimale 1 sur* `véhicule-carburant`*).*

Requêtes : Le schéma E/R précédent est traduit par le schéma relationnel suivant (les clés primaires sont soulignées) :

- `propriétaire(nom,âge)`
- `véhicule(immatriculation,kilométrage,nom,idmodèle)` (`nom` est le nom du propriétaire)
- `modèle(idmodèle,crashtest,consommation)`
- `carburant(désignation,cours,pollution)`
- `utilise(idmodèle,désignation)`

Donnez les requêtes suivantes, dans le bon language :

1. (1 point) Le nom des propriétaires possédant un véhicule dont le kilométrage est strictement supérieur à 100 000 kilomètres (en SQL, algèbre, calcul domaine et n-uplet) ;
   Solution : *SQL :*

```
select propriétaire.nom from propriétaire, véhicule
where véhicule.nom=propriétaire.nom
and véhicule.kilométrage>100 000.
```

[1] inspiré de "Z comme Zorglub", Franquin, Éditions Dupuis, 1966.

Algèbre :

$$\Pi_{nom}(\textit{propriétaire} \bowtie \sigma_{\textit{kilométrage}>100\,000}(\textit{véhicule})).$$

Calcul domaine :

$$\{n | \exists a, i, k, m\, \textit{propriétaire}(n, a) \wedge \textit{véhicule}(i, k, n, m) \wedge k > 100\,000\}.$$

Calcul n-uplet :

$$\{p.nom | \exists p, v\, \textit{propriétaire}(p) \wedge \textit{véhicule}(v) \wedge p.\texttt{nom} = v.\texttt{nom} \wedge v.\texttt{k} > 100\,000\}.$$

2. (1 point) La pollution et la consommation des véhicule de modèle "4x4" (en algèbre et calcul domaine, **pas** en SQL ni en calcul n-uplet).
   Solution : *Algèbre :*

$$\Pi_{\textit{pollution,consommation}}(\textit{utilise} \bowtie \textit{carburant} \bowtie \sigma_{\textit{idmodèle='4x4'}}(\textit{modèle})).$$

   Calcul domaine :

$$\{p, cons | \exists i, d, cours, crash\, \textit{utilise}(i, d) \wedge \textit{carburant}(d, cours, p) \wedge \textit{modèle}(i, crash, cons) \wedge i = \textit{'4x4'}\}.$$

3. (2 points) La consommation moyenne des modèles de véhicule, groupée par type de carburant, et dont la pollution est supérieur à 100 g de CO_2 (en SQL).
   Solution :

```
select mean(consommation), carburant.désignation
from véhicule, carburant, utilise
where véhicule.idmodèle=utilise.idmodèle
and   carburant.désignation=utilise.désignation
and pollution > 100
group by carburant.désignation.
```

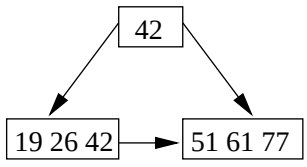
2 Organisation physique (3 points)

On considère un index sur l'attribut `âge` de la table `propriétaire`, de type arbre B+ d'ordre 2.

1. (1 point) Dessinez l'arbre correspondant à l'insertion des valeurs suivantes, en respectant l'ordre donné (ne montrer que les clés dans l'arbre) :

$$\{42,51,19,61,26,77\}.$$

   Solution :



2. (1 point) Construire un nouvel index pour les mêmes données mais dans l'ordre suivant :

$$\{51,19,42,26,77,61\}.$$

   Solution : *L'arbre est identique.*

3. (1 point) Pour un même ensemble de valeur, les arbres fabriqués sont-ils toujours les mêmes quel que soit l'ordre d'insertion ? Argumentez votre réponse.
   Solution : *Non, par exemple l'ordre* $\{19, 26, 51, 61, 77, 42\}$ *donne un arbre différent, avec 51 en racine.*

3 Optimisation (5,5 points)

Soit les commandes SQL suivantes :

```
create table propriétaire(nom varchar2(30) PRIMARY KEY, âge number(2));

create table véhicule(immatriculation varchar2(10) PRIMARY KEY,
     kilométrage number(5), nom varchar2(30), idmodèle number(4));
```

Que font ces commandes (1 point) ?

Solution : *Elles créent deux tables ainsi que les index sur les clés primaires :* `nom` *et* `immatriculation`*.*

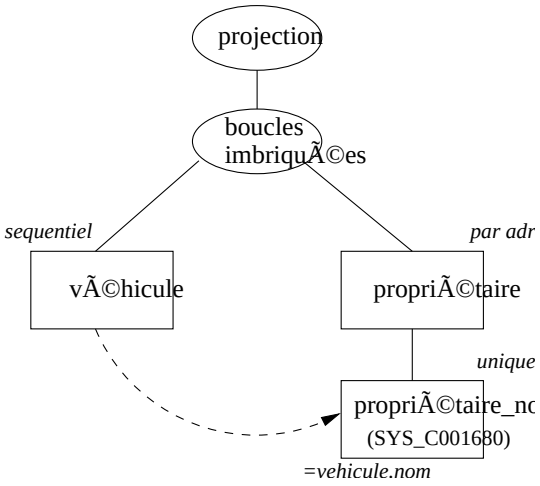
Soit la requête SQL :

```
1 select propriétaire.nom, âge
2  from propriétaire, véhicule
3  where véhicule.nom = propriétaire.nom
4  and véhicule.kilométrage > 50000;
```

Il n'y a pas d'index sur l'attribut `kilométrage` dans la table `véhicule`. Donnez un plan d'exécution sous forme arborescente ou sous forme explain (1,5 point) et expliquez ce plan en détail (1,5 point).

Solution :

```
Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE
   1    0   NESTED LOOPS
   2    1     TABLE ACCESS (FULL) OF 'VÉHICULE'
   3    1     TABLE ACCESS (BY INDEX ROWID) OF 'PROPRIÉTAIRE'
   4    3       INDEX (UNIQUE SCAN) OF 'SYS_C001680' (UNIQUE)
```



On rajoute la commande SQL

```
create index idx_1234 on véhicule(kilométrage);
```

Que fait cette commande (0,5 point) et quelle est la conséquence pour le plan d'exécution physique (1 point) ? (en quelques mots ou à l'aide d'un arbre indiquant le plan d'exécution physique ou à l'aide de commandes explain.)

Solution : *Création d'un index sur le kilométrage.*

```
Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE
   1    0   NESTED LOOPS
   2    1     TABLE ACCESS (BY INDEX ROWID) OF 'VÉHICULE'
   3    2       INDEX (RANGE SCAN) OF 'IDX_1234' (NON-UNIQUE)
   4    1     TABLE ACCESS (BY INDEX ROWID) OF 'PROPRIÉTAIRE'
   5    2       INDEX (UNIQUE SCAN) OF 'SYS_C001680' (UNIQUE)
```

Au lieu de balayer séquentiellement la table `véhicule`*, on traverse l'index sur le kilométrage. Pour chaque véhicule obtenu on obtient le nom du propriétaire par accès (par rowid) à la table* `véhicule` *et on traverse l'index sur le nom de la table* `propriétaire`*. On accède à la table* `propriétaire` *par rowid, pour projeter le* n*-uplet.*

4 Concurrence (5 points)

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

1. (1 point) 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 (en justifiant votre réponse) quelle transaction de **H** pourrait provenir de *KilométrAge*.

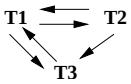
   On considère que les enregistrements (variables) de **H** sont **des nuplets** des relations de la base de données. On suppose que tout nuplet est accessible directement si l'on connaît sa clé.

   Solution : *Les transactions extraites de l'exécution* **H** *sont :*
   T_1 *:* $r_1[x]\ w_1[x]\ r_1[z]\ w_1[y]\ c_1$
   T_2 *:* $r_2[y]\ w_2[y]\ w_2[z]\ c_2$
   T_3 *:* $r_3[x]\ r_3[y]\ w_3[z]\ c_3$
   Dans KilométrAge, *on accède au nuplet de clé* i *de* `véhicule`. *On lit ce nuplet pour récupérer le nom du propriétaire et on l'écrit pour actualiser le kilométrage. Ensuite, avec le nom du propriétaire (clé) et l'age* a, *on écrit le nuplet de* `propriétaire` *(pas besoin de lecture).*

   Il y a donc une mise-à-jour (lecture-écriture) d'un enregistrement et une écriture d'un autre enregistrement. La seule transaction qui correspond est T_2.

2. (1 point) Identifiez tous les conflits dans **H** et vérifiez si l'exécution est sérialisable en construisant le graphe de sérialisation.

   Solution : *Les conflits :*
   sur x *:* $r_3[x]$-$w_1[x]$
   sur y *:* $r_2[y]$-$w_1[y]$, $w_2[y]$-$r_3[y]$, $w_2[y]$-$w_1[y]$, $r_3[y]$-$w_1[y]$
   sur z *:* $r_1[z]$-$w_2[z]$, $r_1[z]$-$w_3[z]$, $w_2[z]$-$w_3[z]$

   Le graphe de sérialisation contient plusieurs cycles, donc **H** *n'est pas sérialisable.*

3. (1 point) A quel niveau de recouvrabilité se situe **H** (recouvrable, évitant les annulations en cascade, stricte) ? Justifiez votre réponse.

   Solution : *On regarde pour chaque écriture de* **H** *si entre l'écriture et sa validation il y a d'autres opérations sur le même enregistrement. La seule écriture dans ce cas est* $w_2[y]$, *car on a* $r_3[y]$ *avant* c_2. *Donc* **H** *n'évite pas les annulations en cascade, mais comme l'écriture* $w_2[y]$ *est validée avant* $r_3[y]$, **H** *est recouvrable.*

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 *:* $r_1[x]\ r_2[y]\ r_3[x]\ w_2[y]\ w_1[x]\ r_3[y]\ r_1[z]\ w_2[z]\ w_1[y]\ c_1\ c_2\ w_3[z]\ c_3$
   $r_1[x]$, $r_2[y]$ *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*
   $w_2[y]$ *prend le verrou d'écriture et s'exécute (pas de conflit avec* $r_2[y]$*)*
   $w_1[x]$ *bloquée par* $r_3[x]$, *donc* T_1 *bloquée*
   $r_3[y]$ *bloquée par* $w_2[y]$, *donc* T_3 *bloquée*
   $r_1[z]$ *bloquée, car* T_1 *bloquée*
   $w_2[z]$ *prend le verrou d'écriture et s'exécute*
   $w_1[y]$, c_1 *bloquées, car* T_1 *bloquée*
   c_2 *s'exécute et relâche les verrous de* T_2, *mais seule* $r_3[y]$ *peut obtenir le verrou et s'exécuter, donc* T_3 *est débloquée*
   relâche les verrous de T_2, *donc* $w_3[y]$ *et* $r_3[z]$ *s'exécutent*
   $w_3[z]$ *s'exécute*
   c_3 *s'exécute et relâche les verrous de* T_3, *donc* $w_1[x]$, $r_1[z]$, $w_1[y]$, c_1 *s'exécutent*
   Le résultat final est donc **H'** *:* $r_1[x]\ r_2[y]\ r_3[x]\ w_2[y]\ w_2[z]\ c_2\ r_3[y]\ w_3[z]\ c_3\ w_1[x]\ r_1[z]\ w_1[y]\ c_1$