NFP107 : Systèmes de gestion de bases de données
EXAMEN, juillet 2008
(tout document écrit autorisé)

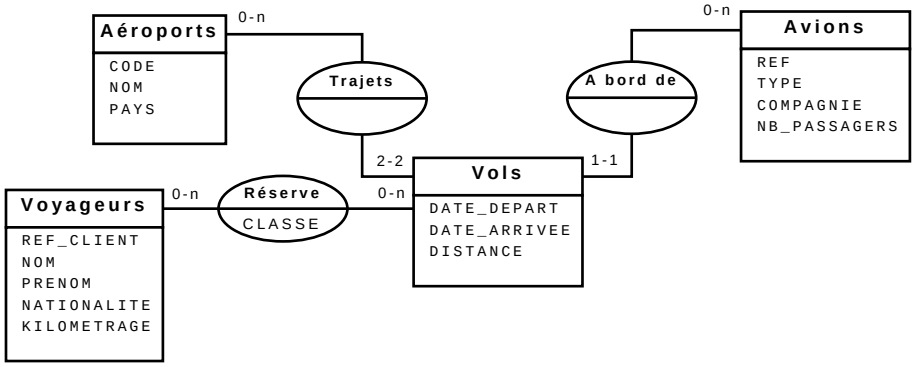
1 Modèle relationnel (10 points)

Une application de gestion des voyages aériens utilise une base de données relationnelle ayant le schéma suivant, où les clés primaires sont soulignées :

- **AEROPORTS**(CODE, NOM, PAYS)
  A chaque aéroport, identifié par un code, correspond son nom et son pays.
- **AVIONS**(REF, TYPE, COMPAGNIE, NB_PASSAGERS)
  A chaque avion, identifié par une référence, est associé un type (A320, Boeing 747, ...), une compagnie (Air France, Alitalia,...) et le nombre maximum de passagers qu'il peut transporter.
- **VOLS**(REF, DATE_DEPART, DEPART, ARRIVEE, DATE_ARRIVEE, DISTANCE)
  A chaque vol, identifié par un avion et une date/heure de départ, sont associés un aéroport de départ et d'arrivée, une date/heure de départ et d'arrivée, ainsi que la distance parcourue.
  Note : Les valeurs des attributs DATE_DEPART et DATE_ARRIVEE contiennent à la fois la date et l'heure, en format 'année-mois-jour heure-minutes-secondes', par exemple la valeur '2008-07-02 18 :21 :14' signifie le 2 juillet 2008, 18 heures, 21 minutes et 14 secondes.
- **VOYAGEURS**(REF_CLIENT, NOM, PRENOM, NATIONALITE, KILOMETRAGE)
  A chaque voyageur, identifié par un code client, sont associés ses nom et prénom, la nationalité et le kilométrage effectué en vols.
- **RESERVATIONS**(REF, DATE_DEPART, REF_CLIENT, CLASSE)
  Une réservation sur un vol est identifiée par les informations d'identification du vol et du passager et précise également la classe du siège réservé.

Modèle conceptuel (3 points) Construisez le schéma conceptuel entité-association de cette base de données en précisant bien les multiplicités des associations.

Solution :



L'association Trajets *pourrait être remplacée par deux associations de multiplicité 1-n*, Départ *et* Arrivée.

Requêtes (7 points)

1. (1 point) Donnez les voyageurs de nationalité française ayant effectué plus de 10 000 km de vol (SQL et algèbre).
   Solution : *SQL :*

```
SELECT * From VOYAGEURS
Where NATIONALITE='Française' AND KILOMETRAGE>10000;
```

   Algèbre :

$$\sigma_{NATIONALITE='Francaise' \wedge KILOMETRAGE>10000}(VOYAGEURS)$$

2. (1,5 points) Les pays de destination pour tous les départs de l'aéroport de code *'CDG'* le 14 juillet 2008 (SQL et algèbre).
   Solution : *SQL :*

```
SELECT A.PAYS From VOLS V, AEROPORTS A
Where V.ARRIVEE=A.CODE AND V.DEPART='CDG' AND
      V.DATE_DEPART>='2008-07-14 00:00:00' AND V.DATE_DEPART<'2008-07-15 00:00:00';
```

   Algèbre :

$$\pi_{PAYS}(\sigma_{DEPART='CDG' \wedge '2008-07-14\ 00:00:00' \leq DATE\_DEPART < '2008-07-15\ 00:00:00'}(VOLS) \bowtie_{CODE=ARRIVEE} AEROPORTS)$$

3. (2 points) Nationalités des voyageurs n'ayant jamais voyagé avec 'Air France' (SQL et algèbre).
   Solution : *SQL :*

```
SELECT DISTINCT V.NATIONALITE From VOYAGEURS V
Where V.REF_CLIENT NOT IN
  (SELECT R.REF_CLIENT From RESERVATIONS R, AVIONS A
   Where R.REF=A.REF AND A.COMPAGNIE='Air France');
```

   Algèbre :

$$R_1 = \pi_{REF\_CLIENT}(VOYAGEURS) - \pi_{REF\_CLIENT}(RESERVATIONS \bowtie \sigma_{COMPAGNIE='AirFrance'}(AVIONS))$$

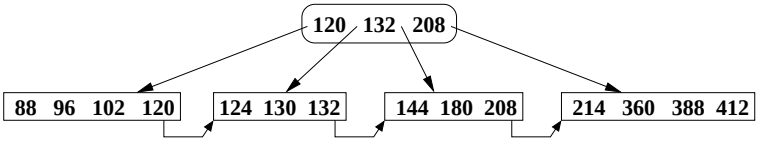
   Résultat = $\pi_{NATIONALITE}(VOYAGEURS \rhd\!\!< R_1)$

4. (1 point) La référence de chaque avion de type 'A320' et la distance totale qu'il a parcourue (SQL).
   Solution : *SELECT A.REF, SUM(V.DISTANCE)*
   FROM VOLS V, AVIONS A
   WHERE V.REF=A.REF AND A.TYPE='A320'
   GROUP BY A.REF ;

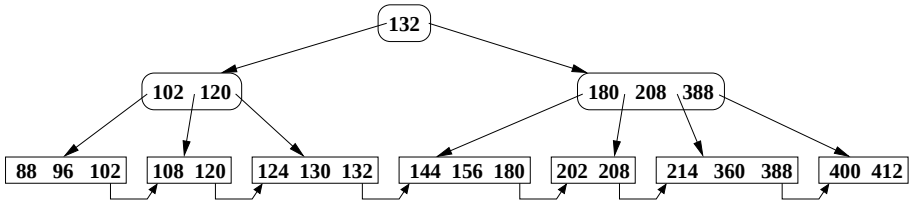
5. (1,5 points) Date/heure de départ des vols complets (toutes les places sont réservées) au départ de 'CDG' (SQL).
   Solution : *SELECT V.DATE_DEPART From VOLS V*
   Where V.DEPART='CDG'
   AND V.NB_PASSAGERS = (SELECT COUNT(*) From RESERVATIONS R
   Where R.REF=V.REF AND R.DATE_DEPART=V.DATE_DEPART)

2 Organisation physique et optimisation (5 points)

Index (2,5 points) Un index de type arbre B+ d'ordre 2 sur l'attribut NB_PASSAGERS de la relation AVIONS a, à un moment donné, la structure suivante :



1. (1,5 points) Montrer l'évolution de l'index après insertion successive dans la relation AVIONS d'avions ayant pour capacités maximales respectivement 202, 400, 108 et 156 places. Il faut montrer tous les éclatements de noeuds de l'index suite à ces insertions.
   Solution :



2. (1 point) Quel est le coût (en nombre de lectures de pages disque) de la recherche dans l'index (après insertions) des avions ayant une capacité de plus de 200 places ? En supposant que l'index stocke seulement les adresses sur disque ('rowid') des nuplets d'AVIONS, quel est le coût pour lire ensuite tous les nuplets ainsi identifiés dans l'index ?

   Solution : *La recherche de la feuille pour 200 (la feuille avec 202, 208) coûte 3 lectures de page dans le nouvel index, ensuite il faut lire les feuilles à droite de celle-ci (encore 2 feuilles). Le coût total de la recherche dans l'index est donc de 5 lectures de pages.*

   Il y a en tout 7 valeurs de plus de 200 ainsi trouvées, il faut donc encore 7 lectures de pages.

Optimisation (2,5 points) Soit la requête suivante, qui donne les distances parcourues par les vols sur des avions de plus de 400 places :

```
SELECT V.DISTANCE From VOLS V, AVIONS A
Where V.REF=A.REF AND A.NB_PASSAGERS>400;
```

1. (1,5 points) Quel serait en principe le meilleur plan d'exécution physique pour cette requête si seul l'index du point précédent existe (sur NB_PASSAGERS) - à noter que les clés des relations ne sont donc pas indexées ici. On considère que le nombre d'avions de plus de 400 places est très faible par rapport au nombre total d'avions. Expliquez et justifiez ce plan, présenté sous forme d'arbre ou de plan EXPLAIN.

   Solution : *A cause de la bonne sélectivité de la condition sur le nombre de passagers, on peut utiliser l'index sur AVIONS.NB_PASSAGERS pour réaliser la sélection. Dans ce cas, une jointure par boucles imbriquées se justifie, car le résultat de la sélection peut tenir en mémoire. Donc le plan contient une jointure par boucles imbriquées, avec à gauche la sélection par index sur AVIONS et à droite le parcours séquentiel de VOLS. La projection sur DISTANCE après la jointure donne le résultat final.*

   Une autre variante acceptable : remplacer les boucles imbriquées par un tri-fusion entre VOLS et le résultat de la sélection par l'index sur AVIONS.

2. (1 point) Que peut-on faire pour améliorer les performances de cette requête ? Justifiez votre réponse.

   Solution : *On peut rajouter un index pour la jointure, sur l'attribut VOLS.REF, ce qui permettrait de remplacer le parcours séquentiel de VOLS par un parcours utilisant cet index. Par contre, un index sur AVIONS.REF n'améliore pas les performances.*

3 Concurrence (5 points)

Soit l'exécution concurrente suivante :
$r_1[x]w_2[y]r_3[x]w_1[x]r_1[z]w_2[z]w_3[y]c_2w_1[z]r_3[z]c_3c_1$

1. (1,5 points) Vérifier si l'exécution est sérialisable en construisant son graphe de sérialisation. Cette exécution est-elle stricte ?

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

   En dessinant le graphe de sérialisation, on retrouve plusieurs cycles, donc l'exécution n'est pas sérialisable. Elle n'est pas stricte non plus à cause de la lecture sale $r_3[z]$ à partir de $w_1[z]$, ce qui peut produire des annulations en cascade.

2. (1,5 points) Trouver l'exécution produite par verrouillage à deux phases simple si les verrous d'une transaction sont toujours relâchés après sa validation. 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
   $w_2[y]$ *s'exécute en prenant le verrou d'écriture 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*
   $r_1[z]$ *bloquée, car* T_1 *bloquée*

$w_2[z]$ *s'exécute en prenant le verrou d'écriture sur* z
$w_3[y]$ *bloquée par* $w_2[y]$*, donc* T_3 *bloquée*
c_2 *s'exécute et relâche les verrous de* $T_2 \Rightarrow w_1[x]$ *ne peut pas être débloquée, mais* $w_3[y]$ *si et s'exécute*
$w_1[z]$ *bloquée, car* T_1 *bloquée*
$r_3[z]$ *s'exécute en prenant le verrou de lecture sur* z
c_3 *s'exécute et relâche les verrous de* $T_3 \Rightarrow w_1[x]$*,* $r_1[z]$ *et* $w_1[z]$ *sont débloquées et s'exécutent*
c_1 *s'exécute*
Résultat : $r_1[x]w_2[y]r_3[x]w_2[z]c_2w_3[y]r_3[z]c_3w_1[x]r_1[z]w_1[z]c_1$

3. (1 point) En considérant que x, y et z sont des n-uplets des relations de la base de données, lesquelles des trois transactions *ne peuvent pas* provenir de l'exécution de la commande
`update AVIONS set NB_PASSAGERS=104 where TYPE='A318'`?
Justifiez votre réponse.

   Solution : *Les trois transactions sont :*
   T_1 : $r_1[x]w_1[x]r_1[z]w_1[z]c_1$
   T_2 : $w_2[y]w_2[z]c_2$
   T_3 : $r_3[x]w_3[y]r_3[z]c_3$
   La commande `update` *réalise une suite de lectures de nuplet, éventuellement suivies de leur modification. La transaction* T_2 *ne contient pas de lecture, tandis que* T_3 *contient une écriture sans lecture du même nuplet. Seule* T_1 *peut provenir de cette commande.*

4. (1 point) Dans un verrouillage hiérarchique à trois niveaux (relation-nuplet-attribut), la lecture d'une relation bloque-t-elle l'écriture d'un attribut de ses nuplets ? Justifiez votre réponse.

   Solution : *Pour écrire un attribut, il faut obtenir les verrous d'intention d'écriture à tous les niveaux supérieurs, y compris au niveau relation. Or ce verrou est en conflit avec le verrou de lecture de la relation, donc la lecture de la relation bloque l'écriture d'un attribut de ses nuplets.*