Ing1 - Examen de Base de Données – EISTI 2008-2009 – RATTRAPAGE - CORRIGE

Mercredi 4 mars 2009

1ère Partie – QCM – 6 points

Répondre aux questions suivantes. Chacune rapporte 0,75 point si elle est entièrement juste.

Q1 – Le modèle entité-association corresponde à quel modèle de données ?

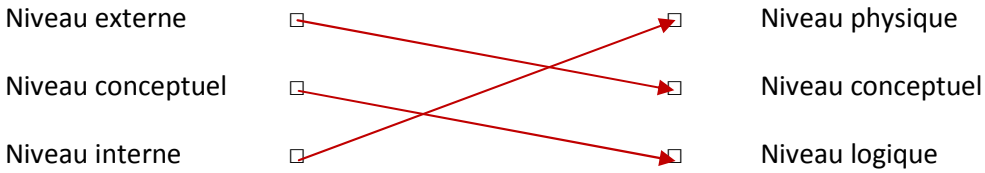
□ modèle hiérarchique

□ modèle réseau

√□ modèle relationnel

□ modèle objet

Q2 – Mettre en correspondance les niveaux de description de donnée ANSI/SPARC (colonne 1) avec les niveaux d'abstraction de MERISE (colonne 2):



Q3 – Une relation présentant plusieurs clés candidates peut être en 3ème forme normale :

vrai □ √□ faux

Q4 – Parmi ces opérations de l'algèbre relationnelle, quelle opération n'a pas une traduction directe en langage SQL ?

□ UNION

□ INTERSECTION

√□ DIVISION

□ DIFFERENCE

Q5 – On considère la relation Evaluation(num_etudiant, code_cours, annee, note) qui stocke les notes des étudiants dans une université. Par exemple, le n-uplet (123, 'BDD1', 2008, 5.0) signifie que l'étudiant 123 avait suivi le cours BDD1 en 2008 recevait la note de 5.0. Nous avons défini la vue V suivante :

```
CREATE VIEW V AS
    SELECT *
    FROM Evaluation e1
    WHERE NOT EXISTS (  SELECT *
                        FROM Evaluation e2
                        WHERE e2.code_cours =  e1.code_cours
                        AND e2.note > e1.note);
```

Ci-après on trouve une requête, impliquant V:

```
SELECT DISTINCT code_cours FROM V WHERE note = 19.0;
```

Laquelle des deux requêtes suivantes donnera le même résultat que la requête ci-dessus?

a.
```
SELECT DISTINCT e1.code_cours FROM Evaluation e1
WHERE e1.note = 19.0
AND NOT EXISTS (    SELECT * FROM Evaluation e2
                    WHERE e2.code_cours = e1.code_cours
                    AND e2.note > 19.0);
```

b.
```
SELECT DISTINCT code_cours FROM Evaluation
GROUP BY code_cours HAVING MAX(note) = 19.0;
```

Cochez la bonne réponse :

□ seulement a

□ seulement b

√□ a et b

□ ni a ni b

Q6 – Supposons que nous souhaitions trouver la note moyenne du cours BDD1 de l'année 2008. Partons du fait qu'il n'y a aucun index sur Evaluation. Laquelle des possibilités ci-dessous va améliorer de façon significative le temps de réponse de notre requête?

□ construire un index sur num_etudiant

□ construire un index sur num_etudiant et un autre sur note

□ construire un index sur code_cours et un autre sur année

√□ construire un index sur le couple (code_cours, année)

Q7– Que signifie ACID ?

Atomicité, Cohérence, Isolation, Durabilité

Q8 – Quels sont les opérations que l'on peut voir dans une transaction ?

√□ SELECT

√□ INSERT

□ DELETE TABLE

□ ALTER

√□ UPDATE

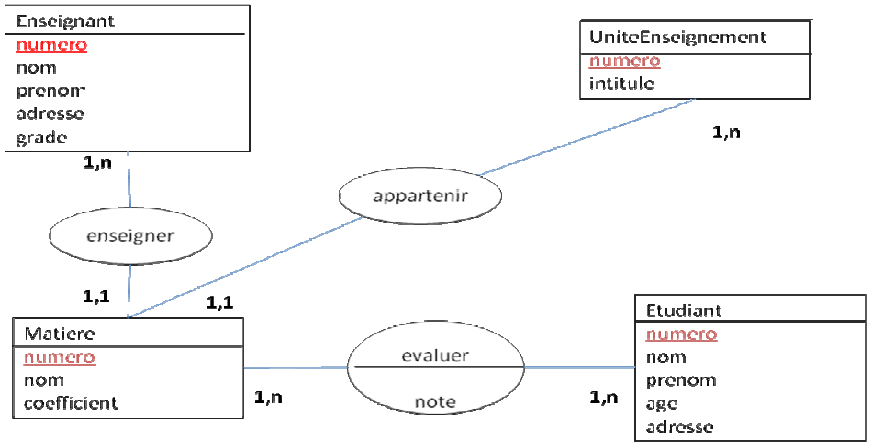
2èmes Partie – Exercices – 14 points

Exercice 1 – 4 points

On se propose de modéliser le programme d'études d'un département dans un institut universitaire de technologie. L'analyse de l'existant a dégagé les informations suivantes:

- Le programme contient plusieurs unités d'enseignement qui sont numérotées. Ces unités peuvent être intitulées Statistique, Outils scientifiques, Environnement économique, Communication ou Stage, etc.
- Les différentes matières sont regroupées en unités d'enseignement. Chaque matière a un numéro unique, un nom, un coefficient et est gérée par un enseignant.
- On connaît le nom, le prénom, l'adresse et le grade de chaque enseignant. Un enseignant peut donner des cours dans plusieurs matières différentes, mais dans même domaine d'expertise.
- Chaque étudiant possède un numéro d'étudiant, un nom, un prénom, un âge et une adresse.
- Les étudiants suivent les cours et ils ont une note pour chaque matière.

Question 1.1 : Proposer le MCD correspondant - **2,5 points**.



Question 1.2 : Donner le MLD associé – **1,5 points**.

Enseignant(numero, nom, prenom, adresse, grade)

Etudiant (numero, nom, prenom, age, adresse)

UniteEnseignement(numero, intitule)

Matiere(numero, *#numeroEnseignant, #numeroUE*, nom, coefficient)

Evaluer(*#numeroMatiere, #numeroEtudiant*, note)

Exercice 2 – 10 points

Soit une base de données relationnelle décrivant les vols d'une compagnie aérienne :

Vol (numVol , villeDepart , villeArrivee , duree, nbPlaces)

Liaison (*numVol , numPilote* , jourDepart, heureDepart , heureArrivee, nbPlacesOccupees)

Pilote (<u>numPilote</u>, nom, prenom, nbHeures)

La table Vol contient le numéro de vol (numVol), la ville de départ, la ville d'arrivée, la durée de vol et l'attribut nbPlaces corresponde au nombre total de places proposées.

La table Pilote contient le numéro du pilote (numPilote), le nom, le prénom et le nombre d'heures de vol effectuées par lui-même.

La table Liaison contient le numéro de vol, le numéro du pilote, le jour de départ, l'heure de départ et l'heure d'arrivée, ainsi que le nombre de places occupées réellement.

Question 2.1 – 4 points

Pour chaque question, proposer une requête en algèbre relationnelle et une requête équivalente en SQL.

a) Quels sont les noms et prénoms des pilotes qui ont déjà atterri à Hong Kong et effectué plus de 1000 heures de vol ?

Algèbre relationnelle :

$\Pi_{nom,prenom}((\sigma_{villeArrivee = 'Hong\ Kong'}(Vol) \bowtie Liaison) \bowtie (\sigma_{nbHeures > 1000}(Pilote)))$

Requête SQL :

```
SELECT nom, prenom
FROM Pilote P, Vol V, Liaison L
WHERE P.numPilote  = L.numPilote
AND V.numVol = L.numVol
AND villeArrivee = 'Hong Kong'
AND nbHeures > 1000;
```

b) Quelles sont la ville de départ, la ville d'arrivée et la durée du vol le plus long assuré par la compagnie ?

Algèbre relationnelle :

$\Pi_{villeDepart,\ villeArrivee,\ duree}((\sigma_{durée = Fmax(duree)(Vol)}(Vol))$

Requête SQL :

```
SELECT villeDepart, villeArrivee, duree
FROM Vol
WHERE duree = (SELECT MAX(duree) FROM Vol ) ;
```

c) Quels sont les liaisons Paris-Madrid (jour et heure de départ) à venir pour lesquelles des places sont disponibles ?

Algèbre relationnelle :

$\Pi_{jourDepart,\ heureDepart}(\sigma_{nbPlaces > nbPlacesOccupées}(\sigma_{villeDepart = 'Paris' \text{ AND } villeArrivee = 'Madrid'}(Vol) \bowtie \sigma_{jourDepart > '04/03/2009'}(Liaison)))$

Requête SQL :

```
SELECT jourDepart, heureDepart
FROM Vol V, Liaison L
WHERE V.numVol = L.numVol
AND villeDepart = 'Paris'
AND villeArrivee = 'Madrid'
AND jourDepart > '04/03/2009'
AND nbPlaces > nbPlacesOccupées;
```

d) Quelles lignes sont desservies par au moins trois vols ? Une ligne est un couple (ville de départ - ville d'arrivée) tel que Paris- Madrid ou Hanoi – Paris.

Algèbre relationnelle :

$\Pi_{villeDepart,\ villeArrivee}(\sigma_{count(numVol) >= 3}(_{villeDepart,\ villeArrivee}F_{villeDepart,\ villeArrivee,count(numVol)}(Vol))$

Requête SQL :

```
SELECT villeDepart, villeArrivee
FROM Vol
GROUP BY villeDepart, villeArrivee
HAVING COUNT(numVol) >= 3;
```

Question 2.2 – 4 points

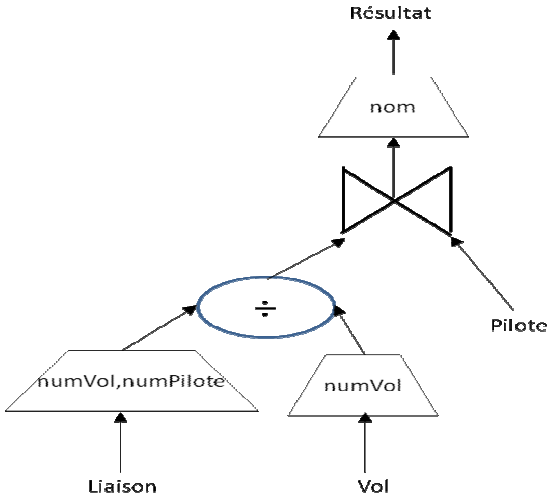
Pour chaque question, proposer une requête en algèbre relationnelle, l'arbre algébrique correspondant et l'équivalent en SQL.

a) Quels sont les noms des pilotes qui ont déjà piloté sur tous les vols de la compagnie ?

Algèbre relationnelle :

$\Pi_{nom}((\Pi_{numVol,\ numPilote}(Liaison) \div \Pi_{numVol}(Vol)) \bowtie Pilote)$

Arbre algébrique :



Requête SQL :

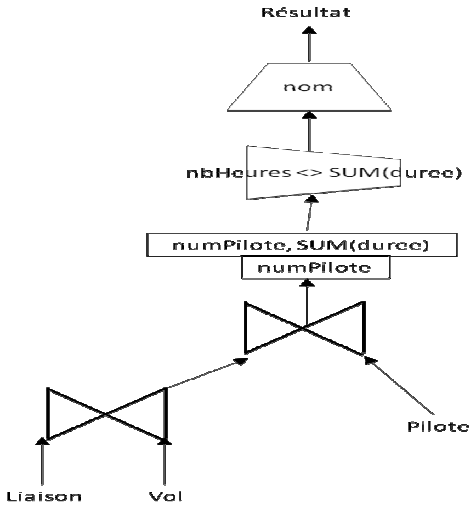
```
SELECT nom
FROM Pilote
WHERE numPilote IN (  SELECT numPilote
                      FROM Liaison
                      GROUP BY numPilote
                      HAVING COUNT(DISTINCT numVol) = (SELECT COUNT(*)
                                                       FROM Vol));
```

b) Afin de détecter les informations manquantes ou erronées, on souhaite trouver les noms des pilotes dont le nombre d'heures de vol enregistré est incohérent par rapport au volume globale des heures calculé à partir de la durée des vols assuré par le pilote.

Algèbre relationnelle :

$$\Pi_{nom}(\sigma_{nbHeures <> SUM(duree)}({}_{numPilote}F_{numPilote,SUM(duree)}(Vol \bowtie (Liaison \bowtie Pilote)))$$

Arbre algébrique :



Requête SQL :

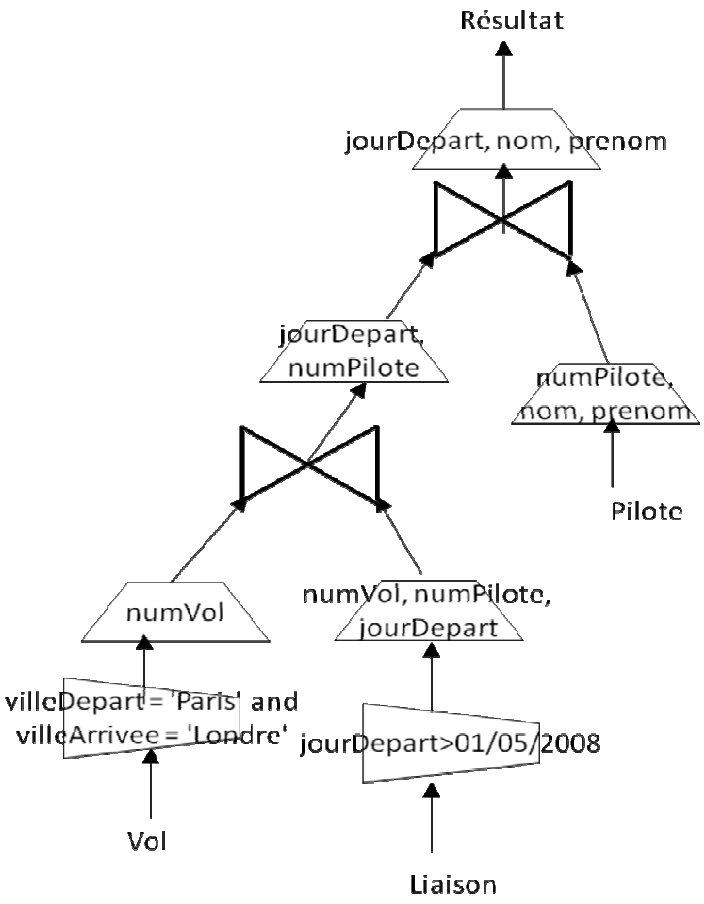
```
SELECT nom
FROM Pilote P
WHERE nbHeures <> (  SELECT SUM(duree)
                     FROM Liaison L, Vol V
                     WHERE P.numPilote  = L.numPilote
                     AND V.numVol = L.numVol
                     GROUP BY L.numPilote);
```

Question 2.3 – 2 points

Donner un arbre algébrique optimisé en suivant l'heuristique du cours pour la requête SQL suivante qui permet de trouver la date et le pilote de toutes les liaisons Paris-Londres postérieures au 1er mai 2008 ?

```
SELECT    jourDepart, nom, prenom
FROM      Liaison L, Pilote P, Vol V
WHERE     L.numPilote = P.numPilote
AND       L.numVol = V.numVol
AND       villeDepart = 'Paris'
AND       villeArrivee = 'Londre'
AND       jourDepart > '01/05/2008'
```

Voici l'arbre le plus optimisé possible : on effectue des restrictions d'abord et puis des projections intermédiaires qui permettent de réduire le nombre de colonnes de chaque table avant de faire des jointures.


Résultat
jourDepart, nom, prenom
jourDepart, numPilote
numPilote, nom, prenom
Pilote
numVol
numVol, numPilote, jourDepart
villeDepart = 'Paris' and villeArrivee = 'Londre'
jourDepart>01/05/2008
Vol
Liaison