Université de Nice-Sophia Antipolis École Polytech'Nice-Sophia

MAM4 SI4
2011–2012

Interrogation de Bases de données relationelles
8 Décembre ( Durée : deux heures)
Tous documents autorisés

# 1 Normalisation - 7 points

On donne la relation : R(A,B,C,D,E,F,G,H) et l'ensemble de dépendances fonctionnelles :

1. $H \longrightarrow A, D$
2. $A \longrightarrow B$
3. $D \longrightarrow E, F$
4. $A \longrightarrow C$
5. $A, D \longrightarrow H$
6. $E \longrightarrow F, G$

- Déterminer la fermeture transitive de chaque attribut et déterminer toutes les clés de R (2 points).
  - $A^+ = \{A, B, C\}$
  - $B^+ = \{B\}$
  - $C^+ = \{C\}$
  - $D^+ = \{D, E, F, G\}$
  - $E^+ = \{E, F, G\}$
  - $F^+ = \{F\}$
  - $G^+ = \{G\}$
  - $H^+ = \{A, B, C, D, E, F, G, H\}$

  Les seules clés sont donc $H$ et $AD$
- Quelles sont les contraintes qui violent la 2NF ? la 3NF ? (1 point) Les contraintes 2,3 et 4 violent la 2NF, la contrainte 6 viole la 3NF
- Décomposer R en table sous forme 3NF en utilisant l'algorithme vu en cours, en indiquant toutes les étapes ( (2 points)
  On commence par mettre la table en 2NF.
  Les contraintes 2 et 4 nous conduisent a introduire une relation $R_1 = (A, B, C)$ dont la seule clé est $A$ et qui est en 3NF et une relation$R_2 = (A, D, E, F, G, H)$ dont les deux clés sont $AD$ et $H$ et qui n'est toujours pas en 2NF a cause de la contrainte $D \longrightarrow EFG$.
  On décompose donc $R_2$ en deux relations $R_3 = (D, E, F, G)$ dont la clé est $D$, qui est en 2NF, mais pas en 3NF à cause de la contrainte 6 et $R_4 = (A, D, H)$ dont les deux clés sont H et AD et qui est en 3NF.
  il reste à décomposer $R_3$ en deux relations $R_5 = (E, F, G)$ dont la clé est $E$ et qui est en 3NF et $R_6 = (D, E)$ dont la clé est D et qui est en 3NF
- Supposons maintenant que nous avons unen relation (n,m) entre les attributs A et D Proposer une nouvelle décompostion en 3NF, sans perte d'information qui minimise la redondance des formations et les risques d'incohérences (2 points) C'est la table $R_4 = (A, D, H)$ qu'il faut modifier : on la remplace par une table $R_7 = (A, D)$ d'association et

deux tables $R_8 = (A)$ et $R_9 = (B)$ ( qui serviront pour mettre des contraintes de type clé étrangère dans $R_7$, et une table $R_{10}$=(A,H) ou (D,H).

# 2 SQL

## 2.1 Where et Having - 7 points

Soit la table *magasin* définie de la manière suivante

```
DROP TABLE ventes;
CREATE Table ventes (
       magasin VARCHAR,
      montant INT,
       laDate DATE);

INSERT INTO ventes VALUES('LaBoutique',400,'14-Jan-1995');
INSERT INTO ventes VALUES('LaCave',1500,'16-Jan-1995');
INSERT INTO ventes VALUES('LeGrenier',500,'17-Jan-1995');
INSERT INTO ventes VALUES('LaBoutique',1100,'16-Jan-1995');
INSERT INTO ventes VALUES('LaCave',800,'17-Jan-1995');
INSERT INTO ventes VALUES('LeGrenier',400,'18-Jan-1995');
INSERT INTO ventes VALUES('LaBoutique',2400,'24-Jan-1995');
INSERT INTO ventes VALUES('LaBoutique',2300,'25-Jan-1995');
```

1. Ecrire une requête SQL qui affiche les noms des magasins ,montant et dates pour les ventes de plus de 1000 euros effectuées après le 15 janvier 1995 (1 point)

   ```
   SELECT magasin,montant, LaDate
         FROM ventes
         WHERE laDate > '15-Jan-1995' and montant >1000;
   ```

2. Ecrire une requête SQL qui affiche la somme totale des montants des ventes par magasin (1 point)

   ```
   SELECT magasin, SUM(montant)
            FROM ventes
            GROUP BY magasin;
   ```

3. Ecrire une requête SQL qui affiche la somme totale des ventes par magasin pour les magasins qui ont effectué des ventes dont le total dépasse 2000 euros (1 point)

   ```
   SELECT magasin, SUM(montant)
            FROM ventes
            GROUP BY magasin
           HAVING SUM(montant) > 2000;
   ```

4. Ecrire une requête SQL qui affiche la somme totale des ventes par magasin pour les magasins qui ont effectué plus de 2000 euros de ventes après le 15 janvier 1995 (1 point)

```
SELECT magasin, SUM(montant)
         FROM ventes
WHERE laDate > '15-Jan-1995'
         GROUP BY magasin
        HAVING SUM(montant) > 2000;
```

5. Ecrire une requête SQL qui affiche la somme des ventes de plus de 2000 euros par magasin pour les magasins qui ont effectué au moins une vente de plus de 2000 euros (1 point)

```
SELECT magasin, SUM(montant)
         FROM ventes
WHERE montant> 2000
         GROUP BY magasin;
```

6. Ecrire une requête SQL qui affiche la somme des ventes de plus de 1000 euros par magasin pour les magasins qui ont effectué au moins 5000 euros de ventes dont le montant par vente est d'au moins 1000 euros (2 point)

```
SELECT magasin, SUM(montant)
         FROM ventes
WHERE montant> 1000
         GROUP BY magasin
HAVING SUM(montant)>=5000;
```

## 2.2 Récursivité– 8 pts

Soit la table *voisin_ droit* définie de la manière suivante :

```
CREATE TABLE voisin_droit(
          nom  VARCHAR,
          voisin_d VARCHAR
        );

INSERT INTO voisin_droit VALUES('pierre','francois');
INSERT INTO voisin_droit VALUES('pierre','jean');
INSERT INTO voisin_droit VALUES('andre','rose');
INSERT INTO voisin_droit VALUES('jean','bastien');
INSERT INTO voisin_droit VALUES('jean','rudolph');
INSERT INTO voisin_droit VALUES('rudolph','gwendoline');
INSERT INTO voisin_droit VALUES('Monique','andre');
INSERT INTO voisin_gauche VALUES('mathilde','francois');
INSERT INTO voisin_gauche VALUES('mathilde','rose');
INSERT INTO voisin_gauche VALUES('rose','bastien');
INSERT INTO voisin_gauche VALUES('rose','rudolph');
```

1. Ecrire une requête récursive SQL qui affiche le nom d'une personne et celui de chacun de ses voisins droits (le voisin droit d'un voisin droit est un voisin droit) (2 points).
Réponse attendue :

```
   nom    |  voisins
----------+------------
 rose     | bastien
 jean     | rudolph
 pierre   | francois
 Monique  | andre
 ...
 mathilde | gwendoline
 andre    | gwendoline
 Monique  | gwendoline

(26 lignes)
```

```
WITH RECURSIVE Lesvoisins(nom, voisins) AS (
          SELECT  nom ,voisin_d  FROM  voisin_droit
          UNION
          SELECT v.nom, vs.voisins
          FROM voisin_droit v,  Lesvoisins vs
          WHERE v.voisin_d = vs.nom)
        SELECT nom, voisins FROM Lesvoisins;
```

2. Ecrire une requéte récursive SQL qui affiche le nom d'une personne et celui de chacun de ses voisins droits ainsi que leur distance de voisinage ( la distance de voisinage étant 1 entre une personne et un voisin droit direct, et sinon 1 + nombre minimum de voisins directs intermediaires (2 points).
Réponse attendue :

```
   nom    |  voisins   |deistanc
----------+------------+---
 Monique  | bastien    | 3
 Monique  | rudolph    | 3
 pierre   | gwendoline | 3
 mathilde | gwendoline | 3
 andre    | gwendoline | 3
 Monique  | gwendoline | 4
(6 rows)
```

```
WITH RECURSIVE Lesvoisins(nom, voisins, N) AS (
          SELECT  nom ,voisin_d,1  FROM  voisin_droit
```

```
          UNION
          SELECT v.nom, vs.voisins, vs.N+1
          FROM voisin_droit v,  Lesvoisins vs
          WHERE v.voisin_d = vs.nom)
        SELECT nom, voisins, N FROM Lesvoisins where N >2;
```

3. Ecrire une requéte récursive SQL qui affiche le nom d'une personne et celui de chacun de ses voisins droits qui sont à une distance de voisinage d'au moins 3 (1 point).
   Réponse attendue :

```
   nom    |  voisins   |distance
----------+------------+---
 Monique  | bastien    | 3
 Monique  | rudolph    | 3
 pierre   | gwendoline | 3
 mathilde | gwendoline | 3
 andre    | gwendoline | 3
 Monique  | gwendoline | 4
(6 rows)
```

```
WITH RECURSIVE Lesvoisins(nom, voisins) AS (
          SELECT  nom ,voisin_d  FROM  voisin_droit
          UNION
          SELECT v.nom, vs.voisins
          FROM voisin_droit v,  Lesvoisins vs
          WHERE v.voisin_d = vs.nom)
        SELECT nom, voisins FROM Lesvoisins;
```

4. Ecrire une requête récursive SQL qui affiche le nom des voisins les plus éloignés (2 points).
   Réponse attendue :

```
  nom   |  voisins
---------+-----------
 Monique | gwendoline
(1 row)
```

```
WITH RECURSIVE Lesvoisins(nom, voisins, N) AS (
          SELECT  nom ,voisin_d, 1  FROM  voisin_droit
          UNION
          SELECT v.nom, vs.voisins, vs.N+1
          FROM voisin_droit v,  Lesvoisins vs
          WHERE v.voisin_d = vs.nom)
        SELECT L1.nom, L1.voisins, L1.N FROM Lesvoisins L1
              where N >= ALL (SELECT  L2.N FROM Lesvoisins L2);
```