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

MAM4 SI4
2009–2010

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

1 Normalisation

On donne la relation : R(C,P,H,S,E) et l'ensemble de dépendances fonctionnelles :
$DF = (C \rightarrow P; HS \rightarrow C; HP \rightarrow S; CE \rightarrow N; HE \rightarrow S)$

– Déterminer toutes les clés de R (2 points)

Seul HE constitue une clé : Ni H ni E n'apparaissent en partie droite de DF, ils doivent donc appartenir à toutes les clés, et comme par transitivité on a $HE \rightarrow SCNP$, HE est une clé.

– Quelle est la forme normale de R (1 points) ?

R est en 2NF (Ni H ni E ne determinent fonctionnelllement quoi que ce soit indépendament l'un de l'autre mais pas en 3NF car $C \rightarrow P$ et C n'appartient pas à la clé.

– Comment peut-on transformer R pour la mettre sous forme normale juste supérieure en appliquant l'algorithme du cours (2 points) ?

Oui, en définissant deux relations:
$R_1(C,P)$ avec pour clé C et $R_2(H,E,S,C,N)$ avec pour clé H,E. On pourrait aussi décomposer R_2 en deux relations : $R_{2_1}(C,E)$ et $R_{2_1}(H,E,D,C)$

---

2 SQL

La base de données associée à une messagerie comporte les tables DNS, ADRESSE_EMAIL, COMPTE_EMAIL, EMAIL,PIECE_JOINTE et DESTINATAIRES_MAIL_ENVOYES.

Les adresses emails sont décomposées en deux parties:

– la partie nom de domaine (exemple, unice.fr) stockée dans la table DNS

– la partie spécifique (nom ou pseudonyme de la personne) dans la table ADRESSE_EMAIL.

La table COMPTE_EMAIL liste les différents comptes mails de la boîte aux lettres. La table ADRESSE_EMAIL contient à la fois les adresses mails des comptes de la boîte aux lettres ainsi que toutes les adresses mail du carnet d'adresse, en particulier toutes les adresses mail de tous les expéditeurs ou destinataires de mail de cette boite aux lettres

```
create table DNS (
   DNS_ID               int                  PRIMARY KEY,
   DNS_NAME             char(128)            not null;


create table ADRESSE_EMAIL (
   AdresseMail_ID          int            PRIMARY KEY,
```

```
   DNS_ID                  int                  REFERENCES DNS,
   AdresseMail_COMPTE      char(128)            not null,
   AdresseMail_NOM         varchar(256)         default null;

create table COMPTE_EMAIL (
   CompteMail_ID         int                PRIMARY KEY,
   AdresseMail_ID        int                REFERENCES Adresse_Email,
   CML_USER_NAME        char(64)            not null,
   CML_PASSWORD         char(32)            not null,
   CML_SERVER           char(64)            not null;
```

La table EMAIL contient tous les mails envoyés ou reçus de tous les comptes relatifs à l'application cliente.

```
create table EMAIL (
   Mail_GUID                 char(36)  PRIMARY KEY,
   Mail_GUID_ORIGINE         char(36) REFERENCES EMAIL,
   AdresseMail_ID            int      REFERENCES ADRESSE_EMAIL, -- adresse de l'expediteur
   CompteMail_ID_Recepteur   int      REFERENCES COMPTE_EMAIL, -- compte recepteur si message reçu
   CompteMail_ID_Expediteur  int      REFERENCES COMPTE_EMAIL, -- compte expediteur si message envoyé
   Mail_DATEHEURE            timestamp    default null,
   Mail_TITRE                varchar(256) default null,
   Mail_CORPS                text         default null; -- text: string of any length
```

La table PIECE_JOINTE contient les différentes pièces jointes relatives à un mail.

```
create table PIECE_JOINTE (
   PCJ_ID               int                  PRIMARY KEY,
   Mail_GUID            char(36)             not null REFERENCES EMAIL,
   PCJ_NOM_FICHIER      varchar(256)         not null,
   PCJ_TYPE_CONTENU     char(32)             default null;
```

La table DESTINATAIRES_MAIL_ENVOYES sert à stocker le ou les destinataires d'un mail envoyé depuis l'un des compte mail.

```
create table DESTINATAIRES_MAIL_ENVOYES (
   AdresseMail_ID        int          not null REFERENCES ADRESSE_EMAIL ,
   Mail_GUID             char(36)     not null REFERENCES EMAIL ,
   constraint PK_MEA primary key  (AdresseMail_ID, Mail_GUID));
```

1. Afficher toutes les adresses mails correspondant aux comptes mail de la boite aux lettres. (La concaténation de deux chaînes se fait avec l'opérateur ||, et peut être utiliser dans le select pour concaterner des attributs ou des constantes)

   ```
   select AdresseMail_Compte||'@'||DNS_NAME  from COMPTE_EMAIL
             join  ADRESSE_EMAIL  using  (AdresseMail_ID) join DNS using (DNS_ID);
   ```

   ---

2. Ecrire les instructions `GRANT` qui accordent les droits suffisants (mais pas davantage) à l'utilisateur `invited` pour qu'il puisse créer la table EMAIL ci-dessus suivante [1]

```
GRANT REFERENCES (CompteMail_ID) ON COMPTE_EMAIL TO  invited;
GRANT REFERENCES (AdresseMail_ID) on COMPTE_EMAIL TO invited;
```

---

3. Ecrire la requête qui calcule le nombre de courriels qui ont été reçus pour l'adresse "Jane.Doe@wanadoo.fr" au cours du mois de juillet 2008.

```
SELECT COUNT(*)
FROM   EMAIL AS E
       INNER JOIN COMPTE_EMAIL AS C
             ON E.CompteMail_ID_Recepteur = C.CompteMail_ID
       INNER JOIN ADRESSE_EMAIL AS A
             ON C.CompteMail_ID = A.CompteMail_ID
                AND C.AdresseMail_ID = A.AdresseMail_ID
       INNER JOIN DNS AS D
             ON A.DNS_ID = D.DNS_ID
WHERE  D.DNS_NAME = 'wanadoo.fr'
  AND  AdresseMail_COMPTE = 'Jane.Doe'
  AND  Mail_DATEHEURE BETWEEN '20080701' AND '20080731 23:59:59.999';

ou


SELECT count(*)
FROM   EMAIL AS E, COMPTE_EMAIL AS C, ADRESSE_EMAIL AS A, DNS AS D
WHERE   E.CompteMail_ID_Recepteur = C.CompteMail_ID
  AND C.AdresseMail_ID = A.AdresseMail_ID
  AND A.DNS_ID = D.DNS_ID
  AND D.DNS_NAME = 'wanadoo.fr'
AND A.AdresseMail_COMPTE = 'Jane.Doe'
  AND E.Mail_DATEHEURE BETWEEN '20080701' AND '20080731 23:59:59.999';
```

---

4. Ecrire la requête qui calcule le nombre moyen de personnes à qui ont été envoyé les mails expédiés à plus d'un destinataire.

```
SELECT AVG(NOMBRE) AS MOYENNE
FROM  ( SELECT M.Mail_GUID, COUNT(*) AS NOMBRE
        FROM   EMAIL AS M
               INNER JOIN DESTINATAIRES_MAIL_ENVOYES AS E
                     ON M.Mail_GUID = E.Mail_GUID
        GROUP  BY M.Mail_GUID
        HAVING COUNT(*) > 1
      ) AS T;
```

---

1. On suppose qu'il a déjà le droit de créer une table dans la base de données dans laquelle se trouve les tables ci-dessus.

5. Comptez le nombre d'adresses mail du carnet d'adresses par rapport aux différents serveurs DNS (tous sans exception, même si il y a 0 adresses pour ce serveur)

```
SELECT DNS_NAME, COUNT(A.DNS_ID)
FROM   DNS AS D
       LEFT OUTER JOIN ADRESSE_EMAIL AS A
            ON D.DNS_ID = A.DNS_ID
GROUP  BY D.DNS_ID, DNS_NAME;
```

---

6. Créez la contrainte CHECK de la table EMAIL afin de faire en sorte de respecter la règle suivante : un et un seul des deux attributs CompteMail_ID_Recepteur CompteMail_ID_Recepteur contient la valeur null.

```
ALTER TABLE EMAIL
   ADD CONSTRAINT CK_EML_RCUENV
   CHECK (    (    CompteMail_ID_Recepteur IS NOT NULL
                AND   CompteMail_ID_Recepteurr IS NULL        )
           OR (    CompteMail_ID_Recepteur IS NULL
                AND     CompteMail_ID_Expediteur IS NOT NULL)
         );
```

---

7. Compte tenu que les réponses à un mail peuvent s'enchaîner de manière arborescente, écrire une requete récursive qui calcule la profondeur maximale d'un fil de discussion.

```
WITH  RECURSIVE
FIL AS
(SELECT *, 1 AS PROFONDEUR
 FROM   EMAIL
 UNION  ALL
 SELECT E.*, PROFONDEUR + 1
 FROM   EMAIL AS E
        INNER JOIN FIL AS F
              ON E.Mail_GUID = F.Mail_GUID_ORIGINE)
SELECT MAX(PROFONDEUR)
FROM   FIL;
```

---

8. Ecrire la requête qui affiche le(s) expéditeur(s) dont provien(nen)t le (ou les) courriel(s) contenant le plus de pièces jointes. Vous pouvez si vous le souhaitez utiliser des vues.

```
CREATE VIEW TC AS
 (SELECT E.Mail_GUID, COUNT(*) AS NOMBRE
  FROM   EMAIL AS E
       INNER JOIN PIECE_JOINTE AS P
              ON E.Mail_GUID = P.Mail_GUID
```

```
  GROUP  BY E.Mail_GUID);

CREATE VIEW TM AS  (SELECT MAX(NOMBRE) AS MAX_NOMBRE  FROM   TC) ;

SELECT A.*
FROM   EMAIL AS E
       INNER JOIN ADRESSE_EMAIL AS A
             ON E.AdresseMail_ID = A.AdresseMail_ID
       INNER JOIN TC
             ON E.Mail_GUID = TC.Mail_GUID
       INNER JOIN TM
             ON TC.NOMBRE = TM.MAX_NOMBRE;

-- ou encore :

SELECT A.*
FROM   EMAIL AS E
       INNER JOIN ADRESSE_EMAIL AS A
             ON E.AdresseMail_ID = A.AdresseMail_ID
       INNER JOIN (SELECT E.Mail_GUID, COUNT(*) AS NOMBRE
                   FROM   EMAIL AS E
                          INNER JOIN PIECE_JOINTE AS P
                                ON E.Mail_GUID = P.Mail_GUID
                   GROUP  BY E.Mail_GUID) AS TC
             ON E.Mail_GUID = TC.Mail_GUID
       INNER JOIN (SELECT MAX(NOMBRE) AS MAX_NOMBRE
                   FROM   (SELECT E.Mail_GUID, COUNT(*) AS NOMBRE
                           FROM   EMAIL AS E
                                  INNER JOIN PIECE_JOINTE AS P
                                        ON E.Mail_GUID = P.Mail_GUID
                           GROUP  BY E.Mail_GUID) AS T) AS TM
             ON TC.NOMBRE = TM.MAX_NOMBRE;
```
