Université Nice - Sophia Antipolis / Polytech – SI3

Controle final de Base de données – 11 Janvier 2018

(Seul les supports de cours et de TD sont autorisés)
Durée : 1h30 minutes

Nom : **Prénom :** **Groupe :** **Note:**

Barème : 1: 3 pts 2: 4 pts 3: 8 pts 4: 4 pts 5: 2 pts

On veut mettre en place une application de traitement des ventes en ligne à partir des relations suivantes:
`client` : (nomC, Prénom, sexe, dateNaissance, villeC)
`produit` :(refP, libellé, prix)
`achat` : (nomC,refP,nomM, quantité)
`magasin` : (nomM, villeM)
`mag-prod` : (nomM, refP)

Les attributs soulignés sont des identifiants uniques, c'est à dire, il n'y a pas deux clients qui portent le même nom. Un magasin peut vendre de nombreux produits et être implanté dans différentes villes; le même produit peut être vendu dans plusieurs magasins.

1. Exprimez en algèbre relationnelle les requêtes suivantes :
   (a) Les noms des clients qui ont acheté tous les produits.

   $\Pi_{\texttt{nomC,refP}}\ \texttt{achat} \div \Pi_{\texttt{refP}}\ \texttt{produit}$

   (b) Les noms des magasins dont aucun produit n'est vendu à moins de 2 euros, et qui ont au moins vendu un produit.

   $\Pi_{\texttt{nomM}}\ \texttt{achat} \setminus \Pi_{\texttt{nomM}}\ [\sigma_{\texttt{prix}<2}\ (\texttt{magasin}) \bowtie \texttt{produit})]$

   ---

2. Ecrire en SQL les tables correspondantes On imposera que les quantités achetées soient strictement positives et que le sexe des clients ait pour valeur 'M', 'F' ou 'X'.

```
DROP TABLE IF EXISTS Client, Produit, Magasin, Achat, MagProd CASCADE;

CREATE Table Client (
       NomC VARCHAR(30) PRIMARY KEY,
       Prenom VARCHAR(30),
       Sexe CHAR CHECK (Sexe ='M' OR Sexe ='F' OR Sexe ='X'),
       DateN Date,
       VilleC VARCHAR(30));

CREATE Table Produit (
       RefP INTEGER PRIMARY KEY,
       Libelle VARCHAR(30),
       Prix INTEGER );

CREATE Table Magasin (
```

```
        NomM VARCHAR(30) PRIMARY KEY,
        VilleM VARCHAR(30) );

CREATE Table Achat (
        NomC VARCHAR(30) REFERENCES Client,
        RefP INTEGER REFERENCES Produit,
        NomM VARCHAR(30) REFERENCES Magasin,
        Quantite INTEGER CHECK ( Quantite > 0),
        PRIMARY KEY (NomC,RefP,NomM)) ;

CREATE Table MagProd (
        NomM VARCHAR(30) REFERENCES Magasin,
        RefP INTEGER REFERENCES Produit,
        PRIMARY KEY (RefP,NomM));
```

---

3. Ecrire les requêtes qui affichent :

   (a) Les noms des clients qui n'ont acheté que des produits vendus à plus de 30 euros.

```
SELECT NomC FROM Client -- ou FROM Achat (pour éliminer ceux qui n'ont rien acheté)
EXCEPT
SELECT NomC FROM  Achat  WHERE RefP IN
           (SELECT RefP FROM Produit P WHERE P.Prix < 30) ;
```

   (b) Les noms des clients ayant effectué au moins un achat dans chaque magasin de la ville où ils sont domiciliés.

```
SELECT  NomC FROM Client C
WHERE VilleC IN (SELECT VilleM FROM Magasin M WHERE M.NomM IN
             (SELECT NomM FROM Achat A  WHERE A.NomC =C.NomC));
```

   (c) Le nom des magasins dont le prix moyen des articles est le plus élevé, c'est à dire supérieur au prix moyen des articles vendus dans l'ensemble des magasins. On proposera deux solutions : l'une qui utilise une vue, l'autre qui utilise la clause *having*.

```
DROP VIEW IF EXISTS PM_Mag;
CREATE view PM_Mag AS
       (SELECT NomM, AVG(Prix) AS P_Moy FROM MagProd MP, Produit P
                     WHERE P.RefP=MP.RefP
       GROUP BY NomM);
SELECT NomM, P_Moy AS PMM  FROM PM_Mag WHERE P_Moy=(SELECT Max(P_Moy) FROM PM_Mag);
```

```
-- Avec HAVING
SELECT NomM, AVG(Prix) AS PM_Max FROM MagProd MP, Produit P
                      WHERE P.RefP=MP.RefP
       GROUP BY NomM
       HAVING AVG(Prix)  >= all
                       (SELECT AVG(Prix)  FROM MagProd MP, Produit P
                             WHERE P.RefP=MP.RefP
                               GROUP BY NomM);
```

---

4. Tables en 3NF

   On veut créer les tables nécessaires pour la gestion de compétitions de courses à pied. Les courses se déroulent dans différentes villes et à des dates différentes. A une date donnée, il n'y a qu'une seule compétition, mais plusieurs compétitions peuvent avoir lieu à des dates différentes dans la même ville.
   Chaque coureur a un numéro d'identification unique et on doit stocker son nom, son prénom, sa date de naissance et l'ensemble des courses auxquelles il a participé. Une ville est définie par son code (identifiant unique) mais il faut aussi stocker son nom et son pays. Une compétition a aussi un identifiant unique et il faut stocker pour chaque compétition, la ville où elle s'est déroulée, sa date, les participants et le nombre de participants.
   On demande de définir l'ensemble des relations nécessaires pour cette base de données; ces relations doivent être sous forme 3NF et la clé doit être spécifiée.

   Les relations suivantes, où les attributs soulignés représentent la clé, sont nécesssaires:
   `compétition` : (CodeComp,CodeVille, Date, NbParticipants)
   `ville` : (CodeVille, Nom, Pays)
   `coureur` : (IdCoureur, Nom,Prénom, DateNaissance)
   `compétition-coureur` : (CodeComp, IdCoureur)

   ---

5. Soit la relation `produit` définie par la table:

```
CREATE TABLE Produit1  (
    Ref INT,
    Prix INT,
    DateFin Date) ;
--Et la requête
SELECT Ref FROM Produit1
    WHERE Prix >= (SELECT MAX(Prix) FROM Produit1)
    AND to_char(DateFin, 'DD Mon YYYY') like '21\%';
```

   Expliquez pour quels attributs il peut être utile de créer un index pour accélérer la recherche, et sur quels attributs cela est inutile, voire pénalisant. Justifiez votre réponse.

   Cela peut être utile sur le prix car le calcul de Max(Prix) est fait avant la comparaison; mais c'est totalement inutile sur DateFin car la date doit être convertie en une chaine de caractères et l'index ne pourra donc pas être utilisé.
