L2 Informatique - INF245 - DS - mars 2016

Durée : 2H **Document autorisé : une feuille A4, recto-verso, manuscripte.**
Le barème est donné à titre indicatif

1 Des relations et des requêtes

Soient R(A,B) et S(B, C, D) deux relations dont les valeurs sont données dans les tableaux suivants :

```
R   A          B
    - ----------
    a          10
    a          12
    b          11
    b           8
    c           8
```

```
S           B          C          D
    ---------- ---------- ----------
            10        100        124
            11        109        123
             8        110        400
            11        109        124
             8        122        400
            11        100        124
```

Question 1 (8 points) :
Donner, sous forme de tableau, le schéma et la valeur de la relation construite par chacune des requêtes ci-dessous (il peut s'agir d'un message d'erreur).

On considère la version de SQL supportée par Oracle.

1. `select C, D from S;`
2. `select B, C, D from S where D <> 124;`
3. ```
   select A, count(B)
   from R group by A;
   ```
4. ```
   select B, C, sum(A)
   from R natural join S
   group by C, B;
   ```
5. ```
   select B, C, count(D)
   from S group by C;
   ```
6. ```
   select A, B, C, D
   from R natural join S
   where A <> 'b';
   ```
7. ```
   select max(count(B))
   from R group by A;
   ```
8. ```
   select B, avg(D), max(D),
          count(distinct D), count(*)
   from S group by B;
   ```

2 A propos de vacances

On reprend la description de l'application Agence de voyages étudiée en TD/TP.

2.1 Position du problème

Une agence de voyage a informatisé la gestion des voyages qu'elle propose (itinéraires, monuments visités, réservations, etc.). La base de données a été construite à partir du cahier des charges suivant :

Les circuits :

Un circuit est identifié par un numéro, il est décrit par une ville de départ, une ville d'arrivée et une séquence d'étapes. Une étape se déroule pendant un nombre donné de jours, dans une ville donnée. Au cours de chaque étape, tous les monuments de la ville, lorsqu'il y en a, sont visités. Les villes de départ et d'arrivée n'étant pas considérées comme des étapes, leurs monuments ne sont pas visités.

Un même circuit ne repasse jamais plusieurs fois dans la même ville étape, mais il peut arriver qu'une ville départ ou arrivée figure aussi dans l'ensemble des villes étapes. Ceci permet de prendre en compte les situations où les villes de départ et/ou d'arrivée font l'objet d'une visite. Les villes sont identifiées par leur nom. Les monuments sont identifiés par leur nom, dans la ville où ils sont situés.

Un circuit peut être programmé plusieurs fois, à des dates différentes. A chacune de ces programmations, on associe un nombre de places proposées (c'est-à-dire le nombre maximal de places qu'il est possible de réserver). Deux programmations d'un même circuit peuvent avoir des nombres différents de places proposées. Par contre, le prix d'un circuit est fixé, toujours le même quelque soit sa programmation. Un circuit dure un nombre de jours égal à la somme des durées de chacune de ses étapes.

Les réservations :

Une réservation, identifiée par un numéro, est effectuée pour le compte d'un client (identifié par son nom) et concerne une programmation d'un circuit. Plusieurs places pour la même programmation du même circuit peuvent être réservées en une seule fois.

Pour chaque programmation, le nombre total de places réservées calculé sur toutes les réservations de cette programmation est inférieur ou égal au nombre de places offertes de la programmation.

2.2 Schéma des relations

Le schéma de la base de données est donné ci-dessous (les identifiants sont soulignés) :

LesVilles (nomV, pays)
/ (n, p) ∈ LesVilles ⟺ la ville dont le nom est n, est située dans le pays p. Le nom de la ville est un identifiant (clef de la relation). */*

LesMonuments (nomM, nomV, prix)
/ (nm, nv, p) ∈ LesMonuments ⟺ le monument de nom nm est situé dans la ville nv. Son prix de visite est p euros. */*

LesCircuits (numC, vDep, vArr, prix)
/ (n, vd, va, pr) ∈ LesCircuits ⟺ le circuit touristique identifié par le numéro n, part de la ville vd et se termine dans la ville va. Son prix est de pr, qui ne prend pas en compte le prix des monuments visités. La ville de départ représente le point de rendez-vous avec les accompagnateurs. */*

LesEtapes (numC, rang, vEtape, nbJours) */* numC,vEtape est une autre clef candidate. */*
/ (n, r, ve, nbj) ∈ LesEtapes ⟺ la r^e étape du circuit n se déroule dans la ville ve, où le séjour est de nbj jours. On fait comme hypothèse que lorsqu'une ville est dans un circuit, tous ses monuments sont visités. r ≥ 1. Les villes de départ et d'arrivée (vArr et vDép de LesCircuits) sont dans LesEtapes lorsqu'elles sont visitées. */*

LesProgrammations (numC, dateDep, nbPlaces)
/ (n, d, nbl) ∈ LesProgrammations ⟺ le circuit identifié par le numéro n, programmé à la date d est offert avec nbl places. Le même circuit peut être programmé à différentes dates. */*

LesReservations (numR, nomC, numC, dateDep, nbRes)
/ (nr, no, nc, d, nbr) ∈ LesRéservations ⟺ le client de nom no, a effectué une réservation identifiée par nr, sur le circuit nc programmé à la date d. Il a réservé nbr places. */*

Les domaines associés sont :

domaine (nomC) = {'Bonemine', 'Corto', etc.}
domaine (vEtape) = domaine (vDep) = domaine (vArr)
= domaine (nomV) = {'Paris', 'Florence', 'Briançon', etc.}
domaine (pays) = {'Italie', 'Finlande', 'France', etc.}
domaine (numC) = domaine (numR) = domaine (nbRes) = domaine (rang)

= domaine (nbJours) = domaine (prix) = domaine (nbPlaces) = entiers > 0
domaine (dateDep) = instant (unité jour)

Les contraintes d'intégrité référentielle sont :

LesReservations[numC, dateDep] $\subseteq$ LesProgrammations[numC, dateDep]
LesProgrammations[numC] = LesCircuits[numC]
LesEtapes[vEtape] $\subseteq$ LesVilles[nomV]
LesCircuits[vArr] $\subseteq$ LesVilles[nomV]
LesEtapes[numC] $\subseteq$ LesCircuits[numC]
LesCircuits[vDep] $\subseteq$ LesVilles[nomV]
LesMonuments[nomV] $\subseteq$ LesVilles[nomV]

Remarques :

- On dit qu'un circuit passe par une ville v, lorsque v est une de ses étapes, ou/et la ville arrivée, ou/et la ville de départ ;
- On dit qu'un circuit visite une ville lorsque celle-ci est une étape de ce circuit.
- On généralise ces définitions aux pays : un circuit passe dans un pays p lorsque l'une de ses étapes, et/ou sa ville arrivée, ou/et sa ville de départ sont situées dans p ; un circuit visite un pays lorsque celui-ci a une fois au moins une étape dans ce pays.

La définition, dans le langage SQL supporté par Oracle, des relations lesVilles, lesMonuments, lesEtapes, et LesReservations, est donnée ci-dessous :

```
create table lesVilles (nomV varchar(20), pays varchar (20),
        constraint villes_pk primary key (nomV) );
create table lesMonuments (nomM varchar(35), nomV varchar (20), prix real,
        constraint fk_nomv foreign key (nomv) references lesVilles (nomv),
        constraint pk primary key (nomm,nomv),
        constraint ck_mon check (prix > 0) ) ;
create table LesReservations(numR integer, nomC varchar (20) not null, numC integer not null,
        dateDep date not null, nbRes integer,
        constraint pk_r primary key (numR),
        constraint fk_nc_date_r foreign key (numC, dateDep) references LesProgrammations (numC, dateDep),
        constraint ck_nbRes check (nbRes > 0) );
create table lesEtapes (NumC integer, rang integer, vEtape varchar (20) not null, nbJours integer,
        constraint pk_ce primary key (numC, rang),
        constraint fk_nc_ce foreign key (numC) references lesCircuits (numC),
        constraint fk_vilet foreign key (vEtape) references lesVilles (nomV),
        constraint ck_nbjours check (nbJours > 0),
        constraint ck_rg check (rang > 0) ) ;
```

Question 2 (4 points) :
Donner dans le langage SQL supporté par Oracle, le code permettant la création des relations lesCircuits et lesProgrammations.

Question 3 (8 points) :
Donner en SQL l'expression des requêtes ci-dessous. Suivre rigoureusement les instructions données.
Les requêtes mal présentées, illisibles ou ne respectant les instructions données seront considérées comme incorrectes. Elles devront construire des résultats sans répétition de valeurs, la clause distinct ne sera utilisée que lorsque nécessaire. Les produits de relation seront exprimés dans la clause from des requêtes. Toute requête emboîtée dans une clause from devra être spécifiée avec soin (attributs, clef(s), prédicat).

1. Donner le numéro des circuits qui visitent la ville Briançon.

2. Donner le numéro des circuits qui passent par la ville Briançon.

3. Donner le numéro, les villes de départ et d'arrivée des circuits qui démarrent après le 01/01/2016.

   Instruction pour la requête 3 : la requête doit utiliser la fonction `to_date` avec le format `DD/MM/YYYY`.

4. Donner les nom des clients qui ont effectué au moins une réservation.

   Instruction pour la requête 4 : la requête ne doit pas utiliser de fonction d'agrégation, ni l'opérateur de partition.

5. Donner les nom des clients qui ont effectué exactement une réservation.

6. Pour chaque programmation de circuit, donner celle dont la date est la plus récente, ainsi que le nombre de places mises en vente.

   Instructions pour la requête 6 : la requête ne doit pas utiliser de fonction d'agrégation ni l'opérateur de partition. Pour comparer des dates on peut utiliser les opérateurs habituels de comparaison.

7. Donner le numéro des circuits qui ne visitent aucune ville de Belgique.

8. Donner le numéro des circuits qui visitent toutes les villes de Belgique.

9. Donner le numéro, le prix de base (sans tenir compte du prix des monuments visités), la date de départ, le nombre de places réservées et le nombre de places disponibles des programmations de circuits qui ont encore des places disponibles. Dans les programmations de circuits qui ont encore des places disponibles n'oublier par de considérer celles sans réservation !