L2 Informatique - INF245 - DS - mars 2016
des solutions

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

1 Des relations et des requêtes

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).

1. `select C, D from S;`

C	D
100	124
100	124
109	123
109	124
110	400
122	400

2. `select B, C, D from S where D <> 124;`

B	C	D
11	109	123
8	110	400
8	122	400

3. `select A, count(B) from R group by A;`

A	NBB
a	2
b	2
c	1

4. `select B, C, sum(A) from R natural join S group by C, B;`

   Impossible de réaliser la somme d'une liste de caractères.

5. `select B, C, count(D) from S group by C;`

   B is not in group by expression.

6. `select A, B, C, D from R natural join S where A <> 'b';`

A	B	C	D
a	10	100	124
c	8	110	400
c	8	122	400

7. `select max(count(B)) from R group by A;`

MAXNBB
2

8. `select B, avg(D), max(D), count(distinct D), count(*) from S group by B;`

B	AVG(D)	MAX(D)	count(distD)	count(*)
8	400	400	1	2
11	123.667	124	2	3
10	124	124	1	1

2 A propos de vacances

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

lesCircuits et lesProgrammations.

Remarques : la spécification de valuation obligatoire (*not null*) est inutile pour les attributs qui participent à la clef primaire. D'autre part, dans une expression constraint fk foreign key (A1,.., An) references S(U1,.., Un), les attributs U1,.., Un **doivent former** la clef primaire de S.

```
create table lesCircuits (numC integer, vDep varchar (20), vArr varchar (20), prix real,
    constraint pk_circuit primary key (numC),
    constraint fk_vdep foreign key (vDep) references lesVilles (nomV),
    constraint fk_varr foreign key (vArr) references lesVilles (nomV),
    constraint ck_cir_prix check (prix > 0)
    constraint ck_cir_numC check (numC > 0));
create table lesProgrammations (numC integer, dateDep date, nbPlaces integer,
    constraint pk_cd primary key (numC, dateDep),
    constraint fk_nc_cd foreign key (numC) references lesCircuits (numC),
    constraint ck_nbplaces check (nbPlaces > 0) ) ;
```

Question 3 (8 points) :
Donner en SQL l'expression des requêtes ci-dessous. Suivre rigoureusement les instructions données.

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

   L'énoncé précise qu'un circuit ne repasse jamais 2 fois dans la même ville étape. La clause select distinct est donc inutile.

   ```
   select numC from LesEtapes
   where vEtape='&v';
   ```

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

   ```
   select distinct numC from LesCircuits natural join LesEtapes
   where vEtape='&d' or vDep='&d' or vArr='&d' ;
   ```

   Une autre version :

   ```
   select numC from LesCircuits
   where vDep='&d' or vArr='&d'
   union
   select numC from LesEtapes where vEtape='&d' ;
   ```

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

   ```
   select distinct numC, vdep, varr
   from LesProgrammations natural join LesCircuits
   where datedep > to_date ('01/01/2016','DD/MM/YYYY') ;
   ```

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

   ```
   select distinct nomC from LesReservations
   ```

   Autre version :

   ```
   select nomC from LesReservations
   group by nomC having count(*) >= 1
   ```

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

   Principe : ce sont ceux qui ont au moins un réservation sauf ceux qui en ont au moins deux.

```
-- Le noms des clients qui ont au moins une réservation.
   select nomC from LesReservations
   minus
-- Le noms des clients qui ont au moins deux réservations.
   select R1.nomC
   from LesReservations R1 join LesReservations R2
   on (R1.nomC=R2.nomC and R1.numR<>R2.numR)
```

   Autre version :

```
select nomC from LesReservations
group by nomC having count(*) = 1
```

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

```
-- Toutes les programmations
   select numC, dateDep, nbPlaces
   from LesProgrammations
minus
-- Les programmations telles qu'il en existe une autre dont la date est plus récente
   select P1.numC, P1.dateDep, P1.nbPlaces
   from LesProgrammations P1 join LesProgrammations P2
        on (P1.numC=P2.numC and P1.dateDep<P2.dateDep) ;
```

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

```
-- tous les circuits
   select numC from LesCircuits
   minus
-- les circuits qui visitent au moins une ville de Belgique
   select numC
   from LesCircuits natural join LesEtapes
        join lesVilles vE on (vEtape=vE.nomV)
   where  vE.pays = '&p'
```

   La solution ci-dessous est incorrecte : elle ne retourne aucun nuplet (*no row selected*)

```
select numC
from LesCircuits natural join LesEtapes
        join lesVilles vE on (vEtape=vE.nomV)
where vE.pays = '&p'
group by numC
having count (*) = 0
```

   La condition introduite par la clause having est toujours fausse.

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

   On introduit deux relations intermédiaires R1(<u>numC</u>, nbVilles) et R2(nbVillesTot) telles que :

   <c, nbV> $\in$ R1 $\iff$ le circuit c visite nbV villes de la Belgique et

   <nbV> $\in$ R2 $\iff$ la Belgique compte nbV villes.

```
select numC
from (select numC, count(nomV) as nbVilles
     from LesEtapes join lesVilles on (vEtape=nomV)
     where pays = '&p'
     group by numC) R1
join
     (select count(nomV) as nbVillesTot from LesVilles
      where pays='&p') R2
on (R1.nbVilles=R2.nbVillesTot)
```

Une autre version : on introduit deux relations intermédiaires R11(numC, pays, nbVilles) et R21(pays, nbVillesTot) telles que :

<c, p, nbV> $\in$ R11 $\iff$ le circuit c visite nbV villes du pays p et

<p, nbV> $\in$ R21 $\iff$ le pays p compte nbV villes.

```
select numC
from (select numC, pays, count(nomV) as nbVilles
     from LesEtapes join lesVilles on (vEtape=nomV)
     group by numC, pays) R11
     join
     (select pays, count(nomV) as nbVillesTot from LesVilles
     group by pays) R21
     on (R1.pays=R2.pays and R1.nbVilles=R2.nbVillesTot)
where R1.pays ='&p';
```

Une autre version :

```
select numC, V1.pays
from LesEtapes join lesVilles V1 on (vEtape=nomV)
where V1.pays='&p'
group by numC, V1.pays
having (pays, count(nomV)) in
   --- pour chaque pays le nombre total de ses villes
   (select pays, count(nomV) from LesVilles group by pays);
```

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 !

   Le principe de la solution est de réaliser l'union des programmations de circuits sans réservation (le nombre de places disponibles est, dans ce cas, le nombre de places mis en vente, et le nombre de places réservées est 0) et des programmations avec réservation (le nombre de places disponibles est alors le nombre de places mis en ventes duquel on retranche la somme des nombres de places réservées) :

   On introduit R(numC,datedep) telle que : <n, d> $\in$ R $\iff$ la programmation du circuit de numéro n dont la date de départ est d n'a aucune réservation.

```
select numC, datedep, prix, nbPlaces as nbDispos
from R natural join LesCircuits natural join LesProgrammations
union
select numC, datedep, prix, nbPlaces - sum(nbRes) as NbDispos
from LesCircuits natural join LesProgrammations
     natural join LesReservations
group by numC, prix, dateDep, nbPlaces
having sum(nbRes) < nbPlaces ;
```

R est calculée par la requête suivante :

```
select numC, datedep
from LesProgrammations
minus
select numC, datedep
from lesReservations
```

En remplaçant R par son expression, on obtient ;

```
select numC, datedep, prix, nbPlaces as nbDispos, 0 as nbRes
from (select numC, datedep
      from LesProgrammations
      minus
      select numC, datedep
      from lesReservations) R
      natural join LesCircuits natural join LesProgrammations
union
select numC, datedep, prix, nbPlaces - sum(nbRes) as NbDispos, sum(nbRes) as NbRes
from LesCircuits natural join LesProgrammations
     natural join LesReservations
group by numC, prix, dateDep, nbPlaces
having sum(nbRes) < nbPlaces ;
```