L2 INF245 — Examen
19 mai 2016, session 1
des solutions

1 À propos de relations

Question 1 (4 points) :
Donner, sous forme de tableaux, le schéma et la valeur de la relation construite par chacune des requêtes ci-dessous appliquées aux deux relations R et S (il peut s'agir d'un message d'erreur Oracle).

1.
```
select C, B
from S where B <> 11;
```
Barême : 1 point par requête. si distinct mal utilisé -0.5 chaque fois.

C	B
100	10
110	8
101	10
100	12

2.
```
select R.A, R.B, S.B, S.C
from R join S on (R.B < S.B);
```

A	B	B	C
3	10	11	101
3	10	11	100
3	10	12	100
1	11	12	100

3.
```
select A, count(distinct B) as nbB,
          count(C) as nbC
from R natural join S group by A;
```

A	NBB	NBC
3	1	2
1	1	2

4.
```
select A, B, count(C) as nbC
from R natural join S group by A;
```
Error Oracle : "Not a group by expression".

5.
```
select A,B from R natural join S
minus
select A,C from T
```

A	B
1	11
3	10

6.
```
select A,B from R
minus
select  R1.A, R1.B
from R R1 join R R2 on (R1.B > R2.B)
```

A	B
3	10

2 À propos de JO

2.3 Expression de requêtes

Question 2 (6 points) :
Exprimer en SQL les requêtes ci-dessous.

1. Quelles sont les épreuves (numéro et date de l'épreuve) pour lesquelles aucune place ne plus être vendue ? C'est-à-dire les épreuves passées.

```
select nEpreuve, dateEpreuve
from LesEpreuves  swhere dateEpreuve < sysdate;
```

2. Quelles sont les numéros des épreuves dans lesquelles le pays Italie a gagné au moins une médaille d'or par équipe ou en couple ?

```
select nEpreuve
from LesResultats R join LesEquipes E on (R.gold=E.nEquipe)
     join LesSportifs S on (E.nsportif=S.nsportif)
where pays='Italie';
```

3. Pour chaque épreuve (donnée par son numéro), combien y a-t-il de billets vendus ?

```
select nEpreuve, count (nBillet)
from LesBillets
group by nEpreuve;
```

4. Donner pour chaque pays le nombre de médailles d'or, le nombre de médailles d'argent, et le nombre de médailles de bronze (lorsqu'une équipe gagne une médaille, celle-ci compte 1 pour le pays de l'équipe).

   Indication : introduire une relation T(pays, nInscrit) telle que <p,n> $\in$ T $\iff$ l'inscrit n, une équipe ou un sportif, est du pays p. Puis utiliser T dans la solution et fournir l'expression SQL qui construit T.

   L'expression qui construit T est :

```
create view T as
select pays, nInscrit
from LesSportifs join LesInscriptions on (nSportif=nInscrit)
union
select pays, nInscrit
from LesSportifs natural join LesEquipes
     join LesInscriptions on (nEquipe=nInscrit)
```

   La solution utilisant T est :

```
select pays, count(distinct gold) as nbOr, 'or' as Couleur
from LesResultats join T on (gold=nInscrit)
group by pays
union
select pays, count(distinct silver) as nbAr, 'argent' as Couleur
from LesResultats  join T on (silver=nInscrit)
group by pays
union
select pays, count(distinct bronze) as nbBr, 'bronze' as Couleur
from LesResultats  join T on (bronze=nInscrit)
group by pays
```

5. Chaque mois (du passé et du futur), pour chaque épreuve (donnée par son numéro), combien y a-t-il de billets vendus ? (un mois est donné par le nom du mois et le numéro de l'année : par exemple Mai 2016)

```
select to_char(dateEpreuve, 'Month, YYYY') as mois, nEpreuve, count(nBillet)
from LesEpreuves natural join LesBillets
group by nEpreuve, dateEpreuve;
```

6. Chaque mois (du passé et du futur), quelle est l'épreuve (donnée par son numéro et son nom) pour laquelle il y a le plus de billets vendus ?

   Soit R (mois, nEpreuve, nbBillet) la relation construite par la requête précédente. La solution utilisant R est :

```
select mois, nEpreuve, nomE
from R natural join LesEpreuves
where (mois, nbBillets) in
         (select mois, max(nbBillets)
         from R
         group by mois);
         -- <m,n> ap sous req. ssi le mois m, n est le nombre maximal de billets
         -- (vendus pour chacune des épreuves)
```

   Autre version :

```
select mois, nEpreuve, nomE
from R R1 natural join LesEpreuves
     join (select mois, max(nbBillets) as nbMax
           from R
           group by mois) R2
           -- <m,n> ap R2 ssi le mois m, n est le nombre maximal de billets
           -- (vendus pour chacune des épreuves)
     on (R1.mois=R2.mois and R1.nbBillets=nbMax)
```

   En remplaçant R par son expression, et en la simplifiant (suppression du produit naturel R natural join LesEpreuves devenu inutile), on obtient :

```
select mois, nEpreuve, nomE
from (select to_char(dateEpreuve, 'Month, YYYY') as mois, nEpreuve, nomE,
             count(nBillet) as nbBillets
      from LesEpreuves natural join LesBillets
      group by nEpreuve, dateEpreuve, nomE) R
where (mois, nbBillets) in
         (select mois, max(nbBillets)
         from (select to_char(dateEpreuve, 'Month, YYYY') as mois,
                      nEpreuve, count(nBillet) as nbBillets
               from LesEpreuves natural join LesBillets
               group by nEpreuve, dateEpreuve) R
         group by mois);
         -- <m,n> ap sous-req. ssi le mois m, n est le nombre maximal de billets
         -- (vendus pour chacune des épreuves)
```

   L'autre version :

```
select mois, nEpreuve, nomE
from (select to_char(dateEpreuve, 'Month, YYYY') as mois, nEpreuve, nomE,
             count(nBillet) as nbBillets
      from LesEpreuves natural join LesBillets
      group by nEpreuve, dateEpreuve, nomE) R1
  join
      (select mois, max(nbBillets) as nbMax
      from (select to_char(dateEpreuve, 'Month, YYYY') as mois, nEpreuve, nomE,
                   count(nBillet) as nbBillets
            from LesEpreuves natural join LesBillets
            group by nEpreuve, dateEpreuve, nomE) R
      group by mois) R2
      -- <m,n> ap R2 ssi le mois m, n est le nombre maximal de billets
      -- (vendus pour chacune des épreuves)
  on (R1.mois=R2.mois and R1.nbBillets=nbMax)
```

2.4 Mise en œuvre de l'application

Question 3 (3 points) :
Fournir le code SQL Oracle qui permet de définir les relations LesEpreuves, LesResultats, et LesSportifs.

```
create table LesEpreuves (nEpreuve number (3,0), nomE varchar2 (20),
    forme varchar2 (13), discipline varchar2 (25), categorie varchar2 (10),
    nbs number (2,0), dateEpreuve date, prix number (4,2),
    constraint ep_pk primary key (nEpreuve),
    constraint ep_fk foreign key (discipline) references LesDisciplines(discipline),
    constraint ep_ck1 check (forme in ('individuelle','par equipe','par couple')),
    constraint ep_ck2 check (categorie in ('feminin','masculin','mixte')),
    constraint ep_ck3 check (nEpreuve > 0),
    constraint ep_ck4 check (nbs > 0),
    constraint ep_ck5 check (prix >= 0),
    -- forme = 'par couple' <=> catégorie = 'mixte'
    constraint ep_ck6 check (forme='par couple' and catégorie='mixte')
);
create table lessportifs (nSportif number (4,0), nomS varchar2 (20), prenomS varchar2(20),
    pays varchar2(20), categorie varchar2(10), dateNais date,
    constraint sp_pk primary key (nSportif),
    constraint sp_un unique (nomS,prenomS),
    constraint sp_ck1 check (nSportif > 0),
    constraint sp_ck2 check (categorie in ('feminin','masculin'))
);
create table LesResultats (nepreuve number(4,0), gold number(4,0),
    silver number(4,0), bronze number(4,0),
    constraint res_pk primary key (nepreuve),
    constraint res_fk1 foreign key (nepreuve,gold)
                       references LesInscriptions (nEpreuve,nInscrit),
    constraint res_fk2 foreign key (nepreuve,silver)
                       references LesInscriptions (nEpreuve,nInscrit),
    constraint res_fk3 foreign key (nepreuve,bronze)
                       references LesInscriptions (nEpreuve,nInscrit),
    constraint res_ck1 check (gold<>silver and silver<>bronze and gold<>bronze)
);
```

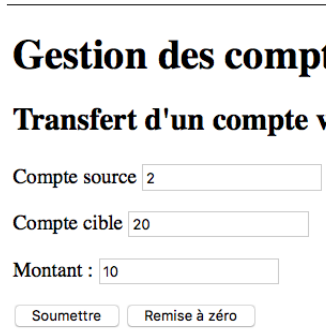
3 A propos de comptes bancaires

Question 4 (4 points) :
Indiquer dans chacun des cas d'interaction ci-dessous, la valeur de la relation LesComptes et la page retournée par l'exécution du programme Transferer-action.php (voir annexe **??**) après que l'utilisateur ait cliqué sur le bouton Soumettre, et interprétée par un navigateur compatible html. Pour la première interaction, considérer la base de données dans l'état initial ci-dessous. Puis, considérer la base de données obtenue à l'issue de l'interaction précédente.

1. Le transfert est impossible: la contrainte *check* est violée (découvert non autorisé). La relation n'a pas changé de valeur :

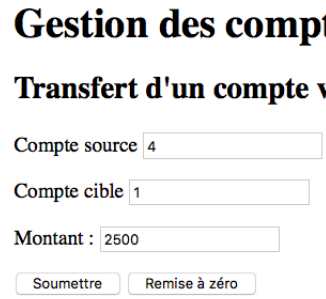
NOC	SOLDE
1	100
2	1
3	4701
4	5000

2.



Le transfert est impossible: la contrainte *check* est violée (découvert non autorisé). La relation n'a pas changé de valeur.

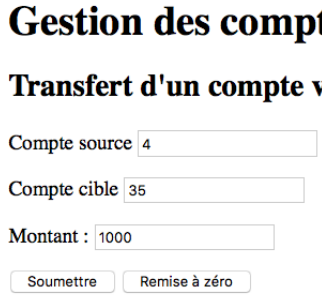
3.



Transfert autorisé. L'état de la relation est :

NOC	SOLDE
1	2600
2	1
3	4701
4	2500

4.



Transfert autorisé. La valeur de la relation est :

NOC	SOLDE
1	2600
2	1
3	4701
4	1500

Question 5 (3 points) :
Le programme php Transférer-action pose t-il problème ? Si oui, expliquer lequel et proposer une solution qui le corrige.

Oui, le programme Transférer-action pose problème : il ne vérifie pas que les comptes dont on modifie le solde existent dans la base de données.

Pour cela, il faut ajouter (lignes 26 pour le compte à débiter et 32 pour celui à créditer) un test qui vérifie que les instructions de mise à jour (*update*) modifient chacune un nuplet exactement.