L2 INF245 — Examen mai 2015, session 1
des solutions

Université Joseph Fourier GRENOBLE UFR IM²AG

1 A 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	R.B	S.B	C
1	11	12	100
3	10	11	101
3	10	11	100
3	10	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
1	1	2
3	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".

Question 2 (2 points) :
Reformuler en SQL chacune des requêtes listées ci-dessous. Suivre très rigoureusement les instructions fournies.

1. select A, B from R where B in
   (select B from S where C >100);

```
select R.A, R.B
from R natural join
     ( select B from S where S.C > 100 ) X;
      -- <b> ap X ssi il existe c > 100
      -- tel que <b, c> ap S
```

   Ou encore :

```
select A, B
from R natural join S
where C > 100;
```

   Autre version :

```
select R.A, R.B
from R join S on (R.B = S.B)
where S.C > 100;
```

2. select B from S group by B having count(distinct C) = 1;

```
select B from S
minus
select  S1.B
from S S1 join S S2
     on (S1.B = S2.B and S1.C <> S2.C)
```

2 A propos de théâtre

2.3 Expression de requêtes

Question 3 (8 points) :
Exprimer en SQL les requêtes ci-dessous.

1. Donner l'expression SQL associée à la vue LesDossiers.

```
create vue LesDossiers as
    select noDossier, sum (prix - R.reduc*prix*B.reduc) as montant
    from LesBillets B natural join LesRepresentations R
         natural join LesZones natural join LesCategories C
    group by noDossier;
```

2. Quelles sont les places (numéro du rang et numéro de la place dans le rang) qui ne peuvent pas être vendues (ou réservées) pour le spectacle numéro 10 ?

```
--- Toutes les places de la salle où ont lieu les représentations
select noRang, noPlace from LesPlaces natural join LesSpectacles
where noSpec = 10
minus
--- Sauf toutes les places déjà réservées pour ce spectacle
select noRang, noPlace from LesBillets where noSpec = 10
--- Sauf toutes les places qui ne sont pas dans la configuration du spectacle
select noRang, noPlace from LesPlaces
where noSpec = 10 and nbZone not in (
      select noZone from LesZones natural join LesConfigurations
      where noSpec = 10 )
```

3. Pour chaque représentation, combien y a t-il de places libres (c'est-à-dire qui n'ont pas été réservées, et qu'il est encore possible de réserver/d'acheter) ?

   Indication : bien qu'intuitive, l'expression SQL select count(A, B)... est incorrecte.

```
select dateRep, count(*)
from (select noPlace, noRang, dateRep
      from LesRepresentations natural join LesConfigurations natural join LesPlaces
      where dateRep > sysdate ) X
      -- <p, r, d> ap X ssi la place <p,r> peut être vendues ou réservées
      -- pour la représentation d planifiée dans le futur.
      minus
      (select noPlace, noRang, dateRep
      from LesBillets
      where dateRep >  sysdate ) Y
      -- <p, r, d> ap Y ssi la place <p,r> est vendue pour la représentation d planifiée
      -- dans le futur.
group by dateRep
```

4. Chaque mois, quelle est la représentation (date et numéro du spectacle) pour laquelle le plus grand nombre de billets est ou a été vendu ? (un mois est donné par le nom du mois et le numéro de l'année : par exemple Janvier 2015)

   Soit R construite par la requête :

```
select mois, dateRep, noSpec, count(noSerie) as nbBillets
from ( select to_char(dateRep, 'Month-YYYY') as Mois, dateRep, noSpect, noSerie
       from LesBillets ) X
       --- <m, d, s, n> ap. X ssi le billet de numéro n a été vendu pour la représentation
       --- du spectacle s. Cette vente a été faite le mois m.
group by mois, dateRep, noSpec
```

<m, d, s, n> $\in$ R $\iff$ n billets ont été vendus pour la représentation d du spectacle s. Cette représentation est prévue ou bien a eu lieu le mois m.

La requête est finalement :

```
select dateRep, noSpec from R
where (mois, nbBillets) in (select mois, max(nbBillets) from R group by Mois )
```

En remplaçant R par son expression, on obtient :

```
select dateRep, noSpec
from (select mois, dateRep, noSpec, count(noSerie) as nbBillets
      from ( select to_char(dateRep, 'Month-YYYY') as Mois, dateRep, noSpect, noSerie
             from LesBillets ) X
             --- <m, d, s, n> ap. X ssi le billet de numéro n a été vendu pour la
             --- représentation du spectacle s. Cette vente a été faite le mois m.
      group by mois, dateRep, noSpec) R2
where (mois, nbBillets) in (
      select mois, max(nbBillets)
      from ( select mois, dateRep, noSpec, count(noSerie) as nbBillets
             from ( select to_char(dateRep, 'Month-YYYY') as Mois, dateRep, noSpect, noSerie
                    from LesBillets ) Y
             group by mois, dateRep, noSpec) R1
      group by mois ) Z
```

2.4 Implantation de l'application

Nous considérons la base de données telle que spécifiée Section 2.1. Le code SQL permettant de définir les relations sauf LesSpectacles, LesZones, LesPlaces et LesCategories est donné en annexe

Question 4 (2 points) :
Fournir le code SQL Oracle qui permet de définir les relations LesRepresentations, LesConfigurations et LesBillets.

```
create table LesConfigurations (noSalle number (4), noZone number (4), noSpec number (4),
      constraint conf_c1 primary key (noSalle, noZone, noSpec),
      constraint conf_c2 (noSalle, noZone) references LesZones(noSalle, noZone),
      constraint conf_c2 (noSpec) references LesSpectacles(noSpec)  );

create table LesRepresentations (dateRep date not null, noSpec number (4), reduc number (4,2),
      constraint rep_c1 primary key (dateRep),
      constraint rep_c2 check (reduc >= 0 and reduc < 100)),
      constraint rep_c3 foreign key noSpec references LesSpectacles(noSpec) );

create table LesBillets (noSerie number (4), noSpec number (4) not null,
      dateRep date not null, noPlace number (4) not null, noRang number (4) not null,
      dateEmission date not null, noDossier number (4) not null,  reduc number (4,2),
      constraint billets_c1 primary key (noSerie),
      constraint billets_c2 unique (noSpec, dateRep, noPlace, noRang),
      constraint billets_c3 foreign key (noSpec, dateRep)
```

```
            references LesRepresentations (noSpec, dateRep),
    constraint billets_c4 foreign key (noPlace, noRang)
            references LesPlaces (noPlace, noRang),
    constraint billets_c6 check (dateEmission < dateRep),
    constraint billets_c7 check (reduc >= 0 and reduc < 100)),
    constraint billets_c8 check (noSerie > 0),
    constraint billets_c9 check (noDossier > 0)  ) ;
```

Question 5 (2 points) :
Parmi toutes les contraintes d'intégrité spécifiées dans l'énoncé lesquelles sont à la charge des applications ?

Les contraintes qui sont à la charge des applications :

- domaine (dateRep) = date(heure) et domaine (dateEmission)= date (seconde)
- Pour chaque représentation, il y a 70 places disponibles jusqu'à t - 1 heure (t est l'instant de début de la représentation).
- Le nombre de places liées à une configuration est ≥ 70
- Chaque jour, dans chaque salle il y a 0 ou 1 représentation
- LesCatégories[nomC] $\subseteq$ LesZones[nomC]
- LesZones[noZone, noSalle] $\subseteq$ LesPlaces[noZone, noSalle]
- LesSpectacles[noSalle] = LesZones[noSalle]
- LesSpectacles[noSpec] $\subseteq$ LesConfigurations[noSpec]
- LesSpectacles[noSpec] $\subseteq$ LesReprésentations[noSpec]
- Pour chaque billet, la place associée (noPlace, noRang) est une place de la configuration de la salle affectée au spectacle :
  LesBillets[noSpec, dateRep, noPlace, noRang] $\subseteq$
  (LesReprésentations * LesConfigurations * LesPlaces)[noSpec, dateRep, noPlace, noRang]
  (* dénote le produit naturel.)

Question 6 (3 points) :
Le programme nouveau-spectacle-action.php écrit en PhP et dont le squelette est donné page **??** en annexe permet d'enregistrer un nouveau spectacle et de construire un message destiné à l'utilisateur qui l'informe de l'issue de cette création. Compléter le programme nouveau-spectacle-action.php (utiliser la page fournie avec le sujet).

```
<?php
include('entete.php');
$noSpec = $_POST['noSpec']; $noSalle = $_POST['noSalle'];
$nomS = $_POST['nomS']; $dateRep = $_POST['dateRep'];
$reduc = $_POST['reduc'];

   $requete = (" select greatest (sysdate, to_date(:d, 'DD-MM-YYYY')) - sysdate from dummy ");
   $curseur = oci_parse ($lien, $requete) ;
   oci_bind_by_name ($curseur, ':d', $dateRep);
   $ok = oci_execute ($curseur, OCI_NO_AUTO_COMMIT) ;
   if (!$ok) {
```

```
   $error_message = oci_error($curseur);
   echo "<p class=\"erreur\">{$error_message['message']}</p>";
       oci_rollback($lien);
   } else {
      $res = oci_fetch ($curseur);
      $diff = (double) oci_result ($curseur, 1);
      if ($diff<=0) {
         echo "Problème d'accès à la base de données. Spectacle non ajouté.";
         oci_rollback($lien);
      } else {
         $reqInsert = (" insert into LesSpectacles values (:n, :s, :a) ");
         $reqInsertRep (" insert into LesRepresentations values (to_date(:d, 'DD-MM-YYYY'), :n, :r) ");
         $curInsert = oci_parse ($lien, $reqInsert);
         $curInsertRep : oci_parse ($lien, $reqInsertRep);
         oci_bind_by_name ($curInsert, ':s', $nomS);
         oci_bind_by_name ($curInsert, ':a', $noSalle);
         oci_bind_by_name ($curInsert, ':n', $noSpec);
-->      oci_bind_by_name ($curInsertRep, ':r', $reduc);
-->      oci_bind_by_name ($curInsertRep, ':d', $dateRep);
-->      oci_bind_by_name ($curInsertRep, ':s', $noSpec);
         $ok = oci_execute ($curInsert, OCI_NO_AUTO_COMMIT) ;
         if (!$ok) {
            $error_code = oci_error($curInsert);
            $constraint = getConstraintName($error_code);
            switch ($constraint) {
                case "spec_c1": $message="Le numéro du spectacle est déjà affecté." ; break;
                default : $message="Problème avec la base de données"; break;
            }
            echo $message ; echo "Spectacle non ajouté.";
-->         oci_rollback($lien);
         } else {
            $ok = oci_execute ($curInsertRep, OCI_NO_AUTO_COMMIT) ;
            if (!$ok) {
               $error_message = oci_error($curInsert);
               $constraint = getConstraintName($error_code);
               switch ($constraint) {
-->                  case "rep_c1": $message="Il y a déjà une représentation à la date $dateRep. ;
                         break;
-->                  case "rep_c2": $message="La valeur du taux de remise doit être dans [0, 100[. ;
                         break;
-->                  case "rep_c3": $message="Impossible de créer une représentation pour un
                                        spectacle inconnu.";
                         break;
-->                 default: $message = "Problème avec la base de données";
                         break;
                }
                echo $message; echo "Spectacle non ajouté";
-->             oci_rollback($lien);
             } else {
                echo "Spectacle ajouté." ;
-->             oci_commit($lien);
}      } } }    }
```