Nom :	**Prénom :**	*page 1*

MLBDA – 4I801- Examen réparti du 5 novembre 2014

Version CORRIGEE

Ex1 :	
Ex2 :	
Ex3 :	

Seuls les documents de cours et de TD sont autorisés – Durée : 2h.

Répondre aux questions sur la feuille du sujet dans les cadres appropriés. Utiliser le dos de la feuille précédente si la réponse déborde du cadre. Le barème est donné à titre indicatif. La qualité de la rédaction sera prise en compte. Ecrire à l'encre bleue ou noire. Ne pas dégrafer le sujet. Eteindre et ranger tout téléphone et autre appareil électronique.

Exercice 1. OQL — 7 pts

On considère le schéma ODL suivant décrivant le contenu du musée du Louvre.

```
Interface Artiste {keys id ; extent LesArtistes ;
        Attribute long id ;
        Attribute string nom ;
        Relationship set<Œuvre> oeuvres inverse Œuvre :: realiséPar ;
        } ;
Interface Œuvre { extent LesOeuvres ;
        Attribute string nom ;
        Attribute integer annee ;
        Relationship Artiste realiséPar inverse Artiste :: œuvres ;
        Relationship Salle estDans inverse Salle :: contient
        Relationship set<Document> description ;
} ;
Interface Sculpture : Œuvre {
        Attribute string matière ;      -- marbre, bois, ..
        Attribute long hauteur;
} ;
Interface Peinture : Œuvre {
        Attribute string technique ;    -- huile sur toile, aquarelle, …
} ;
Interface Salle {
        Attribute integer numero ;
        Attribute integer étage ;
        Attribute string collection;
        Relationship set<Oeuvre> contient inverse Oeuvre :: estDans;
} ;
Interface Document {
        Attribute string auteur ;
        Attribute string titre ;
        Attribute integer annee ;
} ;
```

Question 1. Racines de persistance

Peut-on stocker toutes les instances de la base avec la racine de persistance *LesArtistes* ? Justifiez votre réponse.

Entourer : OUI NON Justifier : .....

Si oui, pourquoi a-t-on défini la racine *LesOeuvres* ? Justifiez votre réponse à l'aide d'un exemple.

Si non, est-il possible de stocker toutes les instances de la base avec une seule racine de persistance autre que *LesArtistes* ? Préciser de quelle racine il s'agit et justifier votre réponse.

1pt ;

Oui, grâce aux associations œuvres, estDans, description.

Pour simplifier l'expression de requêtes, par exemple celles qui ne mentionnent pas les artistes (ex : titre des œuvres exposées dans la salle 3 du 1er étage.) On ne pourrait pas non plus avoir des salles sans œuvre.

Oui, LesOeuvres, grâce à l'association realiséPar qui donne accès aux artistes et lesSalles, avec l'association Contient qui donne accès aux œuvres (puis au reste).

Question 2. L'association entre Œuvre et Salle est déclarée comme étant inverse. Il est possible de déclarer cette association sans inverse, de la façon suivante :

```
Interface Œuvre {…. Relationship Salle estDans ; …}
Interface Salle { …. Relationship Set<Œuvre> contient ;}
```

a) Quel est l'intérêt de déclarer cette association avec inverse ?

0,5pt

permet de contrôler l'intégrité référentielle (une œuvre est dans une salle que si l'œuvre existe, et la salle contient une œuvre que si elle existe). Permet de garantir la cohérence : l'œuvre X est dans la salle Y, la salle Y contient l'œuvre X).

b) Peut-on écrire le même ensemble de requêtes sur la base dans les deux cas ?

0,5pt Oui

Question 3. Ecrivez en OQL les requêtes suivantes :

a) Quelles sont les œuvres de Leonard de Vinci ?

0,5pt

```
Select o
From o in lesOeuvres
Where o.realiséPar.nom = 'Leonard de Vinci';
```

b) Liste des auteurs ayant écrit un ouvrage sur la ‘Joconde’

0,5pt

```
Select d.auteur
From o in LesOeuvres, d in o.description
Where o.nom=’Joconde’;
```

c) Liste des salles présentant les œuvres de la collection ‘peinture italienne’ triées par étage et par numéro (sans doublon).

1pt

```
Select distinct o.estDans
From o in LesOeuvres
Where o.estDans.collection=’peinture italienne’
Order by o.estDans.etage, o.estDans.numero
```

d) Utilisez la question précédente pour lister les œuvres de la collection ‘peinture italienne’ classées par étage et numéro de salle. Le résultat est un triplet (étage, salle, œuvres)

1pt

```
Select struct (etage : o.estDans.etage,
               Salle : o.estDans.num,
               Œuvres : (select c  from c.in s.contient)
       )
From s  in (  Select distinct o.estDans
               From o in LesOeuvres
               Where o.estDans.collection='peinture italienne'
               Order by o.estDans.etage, o.estDans.numero);
```

Question 4.

a) Ecrire une méthode *sculpt()* permettant de distinguer le type d'une œuvre (sculpture ou peinture). *sculpt ()* retourne *true* s'il s'agit d'une sculpture, *false* s'il s'agit d'une peinture.

1pt

```
boolean Œuvre :: sculpt() {return()} ;
boolean Sculpture ::sculpt() {return true};
boolean Peinture::sculpt() {return false};
```

b) Utilisez cette méthode pour écrire en OQL la requête suivante :
Quels sont les artistes qui ont réalisé des peintures **et** des sculptures ?

1pt

```
Select a
```

```
From a in LesArtistes
        o1 in a.oeuvres, o2 in a.oeuvres
Where o1.sculpt() and  not o2.sculpt();
```

Exercice 2. XML DTD — 3 pts

1) Les définitions suivantes sont-elles équivalentes ? (Les éléments B, C et D ont un contenu EMPTY). Entourez la bonne réponse. Justifiez votre réponse si les définitions ne sont pas équivalentes.

<! ELEMENT A (B|C*)> et <! ELEMENT A (B|C)*>

OUI NON
Si NON, pourquoi ?

<! ELEMENT A (B*,B*)> et <! ELEMENT A (B*)>

OUI NON
Si NON, pourquoi ?

<! ELEMENT A (B ? | (C, D)*)> et <! ELEMENT A ((B? | C*), (B? | D*)) >

OUI NON
Si NON, pourquoi ?

<! ELEMENT A ((B*,C) | D+)> et <! ELEMENT A (B*, (C| D+)) >

OUI NON
Si NON, pourquoi ?

```
1pt
Non pour tous, sauf le 2eme
```

2) On considère la DTD suivante :

<! ELEMENT A ((B | D+), (C?, D)*, ((C, (B | D)) | B*), (B | C)) >
<! ELEMENT B EMPTY >
<! ELEMENT C (B, D?) >
<! ELEMENT D EMPTY >

a) Quel est le plus petit document conforme à la définition ci-dessus?

<A>….._ _ _ _

_ _ _ _ _

```
0,5pt
<A><B/><B/></A> ou <A><D/><B/></A>
```

b) Les documents suivants sont-ils conformes à la définition de A ?

<A> <D/> <D/><D/><C><B/><D/></C><D/><B/><B/><B/></A>

Conforme : entourer OUI NON

Si non pourquoi ?:

```
0,5pt
oui
```

<A><B/><D/><C><B/><D/></C><B/><B/><B/></A>

Conforme : entourer OUI NON

Si non pourquoi ?:

```
0,5pt
Non, on ne peut pas avoir 3 B de suite après le C
```

<A><B/><C><B/></C><D/><C><B/></C><B/><D/><C><B/><D/></C></A>

Conforme : entourer OUI NON

Si non pourquoi ?:

```
0,5pt
Non, il y a un D en trop avant le dernier C
```

Exercice 3. SQL3 — 10 pts

On considère le schéma SQL3 suivant décrivant des jeux de tirage (loto, euro million, ...). Lire la 1ère colonne puis la 2ème.

```
create type Cases as table of Number(2);
/
create type Ticket as object (
  jeu Varchar2(30),
  combinaison Cases,
  jour Date,  // la date du ticket ex. '13-6-2014'
  prix Number   // le prix de vente
);
/
create type Tickets as table of Ticket;
/
```

```
create type Joueur as object (
  prenom Varchar2(30),   // le prénom est unique
  achat Tickets
);
/
// Asso est un organisme (culturel, sportif, autre ...)
create type Asso as object (
  nom Varchar2(30),
  ville Varchar2(30)) not final;
/
```

1) A partir du type Asso, définir le type plus spécifique **Club** ayant plusieurs adhérents. Un joueur adhère à un Club en se faisant parrainer par un joueur. On connait la date et le parrain pour chaque adhésion. Un joueur peut adhérer à aucun ou plusieurs clubs. On connait le nom et la ville d'un club. Utiliser les termes de l'énoncé autant que possible.

```
CREATE _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
```

```
create type Adhesion as object(
 j ref Joueur,
 parrain ref Joueur,
 date_adhesion Date
);
@compile

create type Adhesions as table of Adhesion;
@compile

create type Club under Asso (
 joueurs Adhesions);
@compile
```

2) Compléter la définition des tables LesJoueurs et LesClubs stockant les joueurs et les clubs respectivement.

```
Create table LesJoueurs




Create table LesClubs
```

```
create table LesJoueurs of Joueur
nested table achat store as t
 (nested table combinaison store as c); double niveau d'imbrication

create table LesClubs of Club
nested table adherents store as a;
```

3a) Insérer, en SQL3, le joueur Alan qui a joué au loto le vendredi 13-12-2013 la combinaison (1,3,5,7,9) pour 2 euros, et le vendredi 13-6-2014 la combinaison (1,2,4,6,8,10) pour 12 euros.

```
INSERT INTO   _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

_ _ _ _ _ _ __ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
```

```
_ _ _ _ _ _ __ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
```

```
insert into LesJoueurs values(
 Joueur('Alan', Tickets(Ticket('loto', Cases(1,3,5,7,9), '13-12-2013', 2),
                        Ticket('loto', Cases(1,2,4,6,8,10), '13-6-2014', 12))));

insert into LesJoueurs values(
 Joueur('Bob', Tickets(Ticket('loto', Cases(5,7,9,13,17,19), '27-10-2014', 12),
                       Ticket('loto', Cases(13,40,41,42,43), '27-10-2014', 2),
                       Ticket('EuroM', Cases(13,50,51,52,53), '27-10-2014', 12))));
```

3b) Alan a aussi joué à l'EuroM le 14-12-2013 la combinaison (5,7,9,13,17) pour 5 euros. Ecrire cela en SQL3.

```
INSERT INTO _ _ _ _ _ _ _ _ _ _ _ _

                      _ _ _ _ _ _ _

                      _ _ _ _ _ _ _

VALUES ( _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

               _ _ _ _ _ _ _ _ _ _ _ _ _

               _ _ _ _ _ _ _ _ _ _ _ _ _

)
```

```
insert into table(select j.achat
                  from LesJoueurs j
                  where j.prenom='Alan')
       values(Ticket('euroM', Cases(3,4,5,6,7,8), '28-10-2014', 12));
```

4a) En SQL3 : Quel est le prix total dépensé par Bob pour acheter ses tickets ?

```
Select _ _ _ _ _ _

From _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Where _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _
```

```
select sum(t.prix)
from LesJoueurs j, table(j.achat) t
where j.prenom ='Bob'
;
```

4b) En SQL3 : Lister, dans l'ordre croissant, tous les numéros déjà joués par Carole sur des tickets à 2 euros.

```
Select _ _ _ _ _ _

From _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Where _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _
```

```
select distinct value(c)
from LesJoueurs j, table(j.achat) t, table(t.combinaison) c
where t.prix = 2
and j.prenom=’Carole’
order by value(c);
```

5) On complète le schéma avec le type **Stat** représentant des statistiques d'utilisation des numéros.

```
create type Stat as object (
  numero Number(2),      // un numéro déjà joué
  nbusage Number(3));    // le  nombre de fois que le numéro a été joué
/
```

En SQL3, lister les numéros déjà joués au Loto, avec pour chacun leur nombre d'utilisations. Le résultat de la requête doit être une liste d'objets **Stat**. Trier la liste par ordre décroissant du nombre d'utilisation.

```
Select _ _ _ _ _ _

From _ _ _ _ _ _ _ _ _    _ _ _ _ _ _ _ _ _

Where _ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _
```

```
select Stat(value(c), count(*))
from LesJoueurs j, table(j.achat) t, table(t.combinaison) c
where t.nom='loto'
group by value(c)
order by count(*) desc
;
```

6) On complète le schéma avec le type **EnsStat** suivant :

```
create type EnsStat as table of Stat;
```

On complète le type Joueur avec la méthode *mesStat* pour calculer les statistiques d'usage d'un joueur. La signature est :

```
member function mesStat return EnsStat
```

Compléter le corps de la méthode :

```
member function mesStat return EnsStat is
   resultat _ _ _ _ _ _;
begin
SELECT _ _ _ _ _ _ _ _ _

_ _ _ _ _ _  into resultat

FROM _ _ _ _ _ _ _ _ _  _ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _  _ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _  _ _ _ _ _ _ _ _ _

return resultat;
end;
```

```
create or replace type body Joueur as
  member function mesStat return EnsStat is
  resultat EnsStat;
  begin
   select Stat(value(c), count(*))
   bulk collect into resultat
   from table(self.achat) a, table(a.combinaison) c
   group by value(c)
--   order by count(*) desc // ne pas trier
   ;
   return resultat;
 end;
```

```
end;
```

7) Afficher les statistiques d'utilisation de chaque Joueur. Le résultat de la requête doit être une liste contenant des couples dont le type est (prenom Varchar2, e EnsStat).

```
SELECT  _ _ _ _ _ _ _ _ _ _ _

FROM _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
```

```
select j.prenom, j.mesStat() as e
from LesJoueurs j
;
```

8) On suppose que chaque joueur a un seul numéro favori (c'est le numéro qu'il a joué le plus souvent). Afficher le numéro favori de chaque joueur. Le résultat de la requête doit être une liste contenant des couples dont le type est (prenom Varchar2, favori Number).

```
SELECT  _ _ _ _ _ _ _ _

FROM _ _ _ _ _ _ _ _    _ _ _ _ _ _ _ _

WHERE _ _ _ _ _ _ _ _ (SELECT  _ _ _ _ _

_ _ _ _ _ _ _ _    _ _ _ FROM _ _ _ _

_ _ _ _ _ _ _ _    _ _ _ _ _ _ _ _ _
```

```
select j.prenom, s.numero as favori
from LesJoueurs j, table(j.mesStat()) s
where s.nbusage= (select max( s2.nbusage)
                  from table(j.mesStat()) s2)
order by j.prenom, s.numero
;
```

9) (bonus). On complète le type Club avec la méthode **statAdherents** pour calculer les statistiques d'usage de tous les adhérents du club

```
member function mesStat return EnsStat
```

Compléter le corps de la méthode, en invoquant la méthode mesStat() de Joueur.

```
member function statAdherents return EnsStat is
   resultat _ _ _ _ _ _;
begin
SELECT _ _ _ _ _ _ _ _

_ _ _ _ _ _  into resultat

FROM _ _ _ _ _ _ _ _  _ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _  _ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _  _ _ _ _ _ _ _ _ _

return resultat;
```

```
end;
create or replace type body Club as
  member function statAdherents return EnsStat is
     resultat EnsStat
     begin

     select Stat(s.numero, sum(s.nbusage))
     bulk collect into resultat
     from table(self.adherents) a, table(a.j.mesStat()) s
     group by s.numero;
```