Nom :	Prénom :	*page 1*

Module Bases de Données et Web
Examen réparti du 4 novembre 2011
Version CORRIGEE

Les documents sont autorisés – Durée : 2h.

Répondre aux questions sur la feuille du sujet dans les cadres appropriés. La taille des cadres suggère celle de la réponse attendue. 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.

Exercice 1. SQL3 — 10 pts

On considère le schéma SQL3 suivant :

```
Create type  Personne as object (
     Nom varchar2(30),
     Affiliation varchar2(30)
) ;
Create type  EnsPersonnes as table of  ref Personne ;
Create type  Article as object (
     Titre varchar2(50),
     Auteurs EnsPersonnes
);
Create type  EnsArticles as table of ref Article ;
Create type  Conference as object (
      Titre varchar2(50),
      Lieu varchar2(30),
      Dateconf number(4),
      Participants EnsPersonnes,
      Contient EnsArticles
) ;
```

Question 1. Définir les tables LesPersonnes, LesArticles et LesConferences

Create table LesPersonnes of Personne ;
Create table LesArticles of Article nested table auteurs store as T-auteurs;
Create table LesConferences of Conference
Nested table participants store as T-participants,
Nested table contient store as T-contient;

Question 2. On effectue les instructions suivantes :

```
Insert into LesPersonnes values( Personne('Max', 'UPMC') );
Insert into LesPersonnes values( Personne( 'Léa', 'P6') );
Insert into LesArticles values ( Article('art1', EnsPersonnes()) );
```

Ecrivez en SQL3 l'instruction permettant d'insérer la conférence de titre 'BD et Web', qui a lieu à Lyon en 2010, à laquelle Max participe, et qui contient l'article 'art1'.

Insert into LesConferences values ('BD et Web', 'Lyon', 2010, ensPersonnes ((select ref(p) from LesPersonnes p where p.nom='Max')), ensArticles((select ref(a) from LesArticles a where a.titre='art1')));

Question 3. Ecrivez l'instruction SQL3 qui permet d'insérer Léa comme auteur de l'article 'art1'.

Insert into table (select a.auteurs from LesArticles a where a.titre='art1') values ((select ref(p) from lesPersonnes p where p.nom='Léa'));

Question 4. Ecrivez en SQL3 les requêtes suivantes :

1. Noms des participants à la conférence 'BD et Web'.

```
select
```

Select value(p).nom from lesCOnferences c, table(c.participants) p where c.titre='BD et Web';

2. Nombre de participants à la conference 'BD et Web';

```
select
```

Select count(value(p)) from LesConferences c, table(c.participants) p where c.titre =’BD et Web’;

3. Nom des auteurs des articles qui ne participent pas à la conférence où est publié leur article.

```
select
```

```
Select  value(a).nom
from LesConferences c, table(c.contient) cont, table(value(cont).auteurs) a
where value(a) not in (select value(p) from table(c.participants) p);
```

REQUETE NON POSEE

Titre de la conférence qui a eu lieu en 2010 et qui contient l’article ‘art1’.

Réponse :

```
Select c.titre from LesConferences c, table (c.contient) cont
where c.dateconf=2010 and   value(cont).titre=’art1’;;
```

Question 5. Avant qu’un article soit publié, il est évalué par plusieurs relecteurs (5 au maximum) qui lui attribuent une note. Modifier le schéma initial pour avoir, pour chaque article, les relecteurs et la note que chacun a attribuée. Par exemple, l’article ‘art1’ a été relu par Luc, qui a donné la note de 5 et par Marie, qui a donné la note de 4.

```
Drop type article ;
Create type evaluation as object (
relecteur ref Personne,
note number(2)
) ;
Create type ensevaluation as varray(5) of evaluation ;
Create type  Article as object (
     Titre varchar2(50),
```

```
    Auteurs EnsPersonnes,
    Eval ensevaluation
);
Comme il s'agit d'un varray, on n'a pas besoin de modifier la table LesArticles en
ajoutant une nested table. Si  ensevaluation est defini comme une table, il faut
aussi modifier la table en ajoutant une nested table.
```

Exercice 2 : OQL — 10 pts

```
Références:  issu de l'ex OQL du partiel 2010
```

Soit le schéma ODL d'une base pour gérer les utilisateurs d'un réseau social. Un internaute possède un mur sur lequel il partage ses photos. Un internaute peut commenter les photos des autres internautes. Un internaute peut voter pour (avis positif) ou contre (avis négatif) une photo. Une photo peut être associée avec les personnes qui apparaissent en portrait sur la photo. Un internaute a des amis directs, toujours mutuels. Les amis indirects, éloignés d'un chemin de longueur d, sont appelés les proches, *cf.* la méthode proches(d).

interface **Personne** { keys mail;
attribute string **mail**;
attribute string **nom**;
attribute string **prénom**;
relationship set<Photo> **paraît_sur** inverse Photo::portrait_de
};

interface **Mur** { extent **Murs**; keys id;
attribute string **id**;
attribute string **nom;**
relationship Internaute **propriétaire** inverse Internaute::mur;
relationship set<Photo> **contient** inverse Photo::sur;
Photo **meilleur_avis()**;
set<Photo> **x()** ;
};

interface **Internaute** : Personne { extent **Internautes**;
attribute string **login**;
attribute long **age**;
attribute string **ville**;
relationship set<Internaute> **amis** inverse Internaute::amis;
relationship Mur **mur** inverse Mur::propriétaire;
set<Internaute> **proches(**long d**)**;
};

interface **Photo** { extent **Photos**; keys numéro;
attribute long **numéro** ;
attribute string **nom**;
attribute long **hauteur**;
attribute long **largeur**;
attribute long **avis_positifs** ;
attribute long **avis_négatifs** ;
relationship Mur **sur** inverse Mur::contient;
relationship set<Personne> **portrait_de** inverse Personne::paraît_sur;
relationship set<Commentaire> **commentaires** inverse Commentaire::sujet;
};

interface **Commentaire** { keys numéro;
attribute long **numéro**;
attribute string **nom**;
attribute string **texte**;
relationship Internaute **écrit_par**;
relationship Photo **sujet**
inverse Photo::commentaires;
} ;

Question 1. Compléter le schéma en ajoutant une relation inverse pour la relation *écrit_par* de l'interface *Commentaire*.

```
Dans l'interface Commentaire :
  Relationship Internaute écrit_par inverse Internaute::commente

Dans l'interface Internaute :
  Relationship set<Commentaire> ecrit inverse Commentaire::ecrit_par
```

Question 2. Dans cette question et les suivantes, on ignore la modification de schéma faite à la question 1. On considère le schéma donné dans l'énoncé.
a) Combien de racines de persistance sont définies ?

```
3 racines Inernautes,Murs,Photos
```

b) La base peut-elle contenir des personnes qui ne sont pas des internautes ?

```
Oui, bien qu'il n'y ait pas de racine pour les Personnes,
il est possible d'atteindre toute les personnes qui apparaissent sur au moins une
photo.
```

Question 3. Ecrire en OQL les requêtes suivantes :
R1 : Afficher le nom des photos ayant au moins 5 avis positifs, et se trouvant sur le mur de l'internaute dont le mail est '123@etu.upmc.fr'

```
select
from    in
where
and
```

```
Select p.nom
From p in Photos
Where p.avis_positifs >5
And p.sur.propriétaire.mail= '123@etu.upmc.fr'
```

R2 : Afficher le mail des amis de l'internaute ayant mis une photo nommée 'Tour Eiffel' sur leur mur. Donner deux réponses équivalentes, la première en utilisant la racine *Internautes*, la deuxième utilisant la racine *Photos*.

```
select
from i in Internautes
where


select
from p in Photos
where
```

```
Select a.mail
From i in Internautes, a in i.amis, p in i.mur.contient
Where p.nom = 'Tour Eiffel'



Select a.mail
From p in Photos, a in p.sur.propriétaire.amis
Where p.nom = 'Tour Eiffel'
```

R3: Afficher le nombre moyen de photos par mur. Rmq : la fonction avg() calcule la moyenne.

```
avg(select count(m.contient)
    from m in Murs )

OU

select avg( select count(*)
            from p in m.contient)
from m in Murs

OU

Select avg(count(partition))
From m in Murs, p in m.contient
Group by m
```

R4: Quels sont les internautes qui n'ont commenté aucune photos ?

```
select

from

where
```

```
Select
From i in Internautes
Where i not in (select p.écrit_par
                From p in Photos, c in p.commentaires)


Autre possibilité avec un for all
```

R5 : Quel est le format (hauteur et largeur) de toutes les photos partagées sur les murs des amis de Jean Dupont ?

```
select p.largeur, p.hauteur
from i in Internautes, a in i.amis, p in a.mur.contient
where i.nom= 'Dupont' and i.prénom='Jean'

Autre solution (en utilisant 2 racines
Select p.largeur, p.hauteur
from m in Murs, p in m.contient, i in Internautes, a in i.amis
where i.nom= 'Dupont' and i.prénom='Jean'
and m.propriétaire = a
```

Question 4

Ecrire le corps de la méthode *top_commentaire()* qui retourne la ou les photos ayant le plus grand nombre de commentaires, parmi les photos mur en question.

```
Set<Photo> Mur::top_commentaire () {









}
```

```
Photo Mur::top_commentaire() {

return (select p
        from p in this.contient
        where count(p.commentaire) = (
                select max(p1.commentaire)
                from p1 in this.contient)
       )
}

Autre solution avec >= ALL
```

Question 5

a) Expliquer ce que retourne la méthode y() définie ci-dessous. Quel est l'inconvénient majeur de cette méthode ?

```
set<Photo> Mur::y() {
  return (this.contient
           union
          select distinct p
          from a in this.propriétaire.amis, p in a.mur.y());
}
```

La méthode y retourne les photos d'un mur d'un internaute et celles de ses amis directs et indirects. Problème : fin de la récursion ?

b) Ecrire le corps de la méthode *proches(d)* qui retourne l'ensemble des amis (et amis d'amis par transitivité) atteignables par un chemin de longueur inférieure ou égale à d. Remarque : si nécessaire vous pouvez utiliser une instruction conditionnelle *if then else endif*.

```
set<Internaute> Internaute::proches(long d) {









}
```

```
Set<Internaute> Internaute::proche(int d)

if(d>1) then
Return (  this.amis
          union
          Select distinct b
          From a in this.amis, b in a.proches(d-1)
        )
```

```
Else
     Return (this.amis)
End if;
```

QUESTION NON POSEE

c) Etant donné une personne *A*, on veut déterminer l'ensemble *E* des personnes qui apparaissent soit sur la même photo que *A*, soit sur la même photo que quelqu'un appartenant à *E*. Expliquer brièvement comment obtenir *E*.

Réponse :
Fermeture transitive de la relation : p_i ' est sur la même photo que p_j

On ajoute la méthode auxilliaire m() pour implémenter la relation « est sur la même photo»

```
Set<Personne> Personne::m() {
  return (select distinct pers
          from ph in this.paraît_sur, pers in ph.portrait_de
}
```

On détermine l'ensemble E à l'aide de la méthode récursive e()

```
Initialisation: Set<Personne> visité = (A);
Invocation: A.e(visité) ;

Attention, il faut détecter les cycles sinon ça ne se termine pas.

Set<Personne> Personne::e( set<Personne> visitées) {

// les personnes à visiter (moins celles qui ont déjà été visitées)
Set <Personne> à_visiter = select p
                           From p in m()
                           Where p not in visitées;

              Ou à visiter = m() minus visitées;

// compléter les personnes visitées
Visitées = Visitées union m() ;

return à visiter
         union (select distinct q
                from p in à_visiter, q in p.e(visitées) )
}
```