MLBDA – MU4IN801- 1ER EXAMEN REPARTI

DU 10 NOVEMBRE 2021

Version avec SOLUTIONS

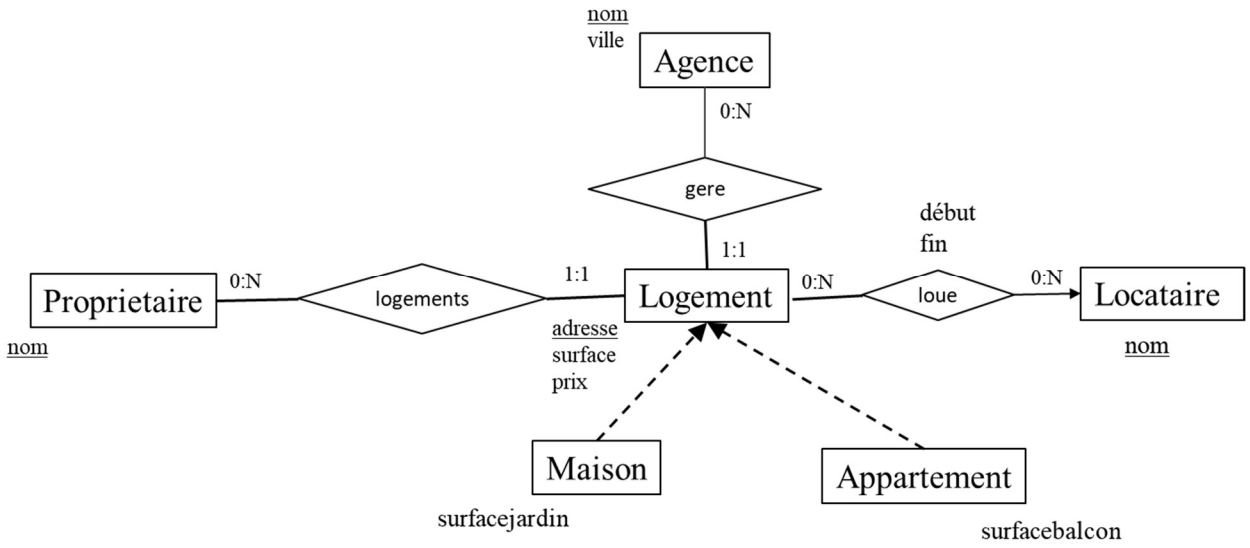
Ex 1 :	Ex 2 :	Ex 3 :	

Les documents sont autorisés – Durée totale : 2h.

Répondre aux questions sur la feuille du sujet dans les cadres appropriés. Utiliser le dos de la dernière feuille si la réponse déborde du cadre. Le barème est donné à titre indicatif. La qualité de la rédaction sera prise en compte dans la note.

Exercice 1. Modélisation SQL3 — 10 pts

On considère le schéma Entité-Association suivant, modélisant une base de données de locations.



Les logements ont une adresse (identifiant), une surface et un prix de location (euros/mois) et chaque logement est géré par une seule agence et a un seul propriétaire. On fait la distinction entre les maisons avec une surface de jardin et les appartements avec une surface de balcon. Les agences ont un nom et se situent dans une ville. Elles peuvent gérer plusieurs logements. Les propriétaires ont un nom et peuvent posséder plusieurs logements. Un logement peut être loué par différents locataires à différentes périodes de temps. Un locataire a un nom et peut louer plusieurs logements.

On veut modéliser ce schéma en SQL3. On définit les associations N-N dans le sens des flèches (ex : *Logement – loue* → *Locataire*). Chaque entité *E* correspond à un type SQL3 `T_E` et il peut exister d'autres types en plus.

Les types suivants sont déjà définis (le schéma est incomplet):

```
create type T_Locataire as object (
        nom varchar(16)
) final instantiable;

create type T_Proprietaire as object (
        nom varchar(16),
        logements Ens_Logements
) final instantiable;
```

```
create type T_Logement as object (
        adresse varchar(32),
        surface number(3),
        prix number(6,2),
        proprietaire ref T_Proprietaire,
        agence ref T_Agence,
        locations Ens_Locations
) not final not instantiable;
```

Il existe **quatre tables** (racines de persistance) `LesProprietaires`, `LesLocataires`, `LesLogements`, `LesAgences` pour stocker des objets de type `T_Proprietaire`, `T_Locataire`, `T_Logement` et `T_Agence`.

1.1 Définition des types et des tables

Définissez les types et les tables suivantes (il est possible que vous deviez définir d'autres types en plus).

Question 1. Définissez le type `T_Agence`.

1 pt

```
create type Ens_Logements as table of ref T_Logement;
create type T_Agence as  object(
  nom  varchar(16),
  ville varchar(16),
  logements Ens_Logements,
) final instantiable;
```

Question 2. Définissez le type `T_Appartement`.

1 pt

```
create type T_Appartement under T_Logement (
  surfacebalcon decimal(3)
 ) final  instantiable;
```

Question 3. Définissez le type `Ens_Locations` utilisé par le type `T_Logement`.

2 pts

```
create type T_Location as object (
     debut Date,
     fin Date,
     locataire ref T_Locataire
) final instantiable;
/
create type Ens_Locations as table of T_Location;
```

Question 4. Définissez les tables `LesProprietaires`, `LesLocataires`, `LesLogements` et `LesAgences`.

1pt

```
create table LesProprietaires of T_Proprietaire
   nested table logements store as T367;;
create table LesLocataires of T_Locataire;
create table LesLogements of T_Logement
```

```
  nested table locations store as T476;
create table LesAgences of T_Agence
    nested table logements store as T267;
```

2.1 Insertion de données :

Question 5. Ecrivez les instruction SQL3 pour insérer

- l'agence 'Louer Pas Cher' à Paris,
- un appartement au 'App 3 - 1, rue de Paris, Gentilly', avec 75 m2 du surface loué à 500€/mois. La surface du balcon est 20 m2.
- un propriétaire qui s'appelle Max Dupont

1pt

```
insert into LesAgences values ('Louer Pas Cher','Paris',Ens_Logements());

insert into LesLogements values (T_Appartement('App 3 - 1, rue de Paris,
Gentilly',75,500,null,null,Ens_Locations(),20));

insert into LesProprietaires values (T_Proprietaire ('Max Dupont',
Ens_Logements()));
```

Question 6. Ecrivez les instructions SQL3 qui déclarent Max Dupont comme propriétaire de l'appartement 'App3 - 1, rue de Paris, Gentilly'.

2pt

```
     update LesLogements
     set proprietaire = (select ref(p) from LesProprietaires p
                         where p.nom='Max')
     where adresse='App 3 - 1, rue de Paris, Gentilly';

     insert into table (select logements
                        from LesProprietaires where nom='Max Dupont')
           select ref(l) from LesLogements l
           where adresse='App 3 - 1, rue de Paris, Gentilly';
```

ou (si logements IS NULL)

```
   update LesProprietaires e
   set e.logements = Ens_logements ()
   where nom='Max Dupont';

   insert into table (select logements
                        from LesProprietaires where nom='Max Dupont')
         select ref(l) from LesLogements l
           where adresse='App 3 - 1, rue de Paris, Gentilly';
```

La commande suivant ne fonctionne pas -1 pt) :

```
update LesProprietaires e
     set e.logements = Ens_logements (
                           select ref(l) from LesLogements l
                           where adresse='App 3 - 1, rue de Paris,
Gentilly'))
   where nom='Max Dupont';
```

2.2 Méthodes SQL3

Question 7. On veut ajouter à ce schéma la méthode **getLocataires(dateloc Date)** qui retourne, pour une agence, l'ensemble des locataires des logements qui sont gérés par l'agence et loués à une date donnée.

Complétez la définition suivante.

```
create or replace type body …… as
  member function getLocataires(…….) return ……. is









end;
```

2pt

```
create or replace type body T_Agence as
  member function getLocataires(dateloc Date) return Ens_Locataires is
  resultat Ens_Locataires;
  begin
     select value(l).locataire bulk collect into resultat
     from table(self.logements) v, table (value(v).locations) l
     where value(l).debut <= dateloc and value(l).fin >= dateloc;
     return resultat;
  end;
 end;
```

Exercice 3 : SQL3 requêtes 7 pts

Soit le schéma :

```
create type T_Service ;
create type T_Contrat as  object(
        salaire integer,
        statut varchar(16) -- stagiaire, CDD, CDI
);
create type T_Personne as object (
        nom varchar(32),
        prenom varchar(32),
        ville varchar(32),
        contrat T_Contrat);
create type Ens_Personnes as table of ref T_Personne;
create type T_Service as object (
        nom varchar(32),
        responsable ref T_Personne,
        membres Ens_Personnes
)
```

```
create type Ens_Services as table of ref T_Service;
create type T_Departement as object (
   nom varchar(32),
   responsable ref T_Personne,
   services Ens_Services
  )
create table LesPersonnes of T_Personne;
create table LesServices of T_Service
   nested table membres store as T789;
create table LesDepartements of T_Departement
   nested table services store as T356;
```

Répondre en SQL3 en complétant les « trous » dans les modèles de réponse.

Question 1: Les noms des membres des services du département 'Informatique'.

```
1pt
Select value(m).nom
From LesDepartements d, table(d.services) s, table(value(s).membres) m
Where d.nom = 'Informatique';
```

Question 2 : Le nom du responsable du service 'Logiciels' du département 'Informatique'.

```
Select value(s).responsable.nom
From LesDepartements d, table(d.services) s
Where d.nom = 'Informatique'
And value(s).nom = 'logiciel'
```

Question 3 : Les noms des responsables de département avec un contrat CDI, et des responsables de service avec un contrat CDD. Afficher le nom de la personne, le type de son contrat et son salaire

1 pt

~~Select value(m).nom~~
~~From LesDepartements d, table(d.services) s, table(value(s).membres) m~~
~~Where d.nom = 'Management'~~
~~And value(m).contrat.statut = 'CDI'~~
~~Union~~

```
Select value(s).responsable.nom, value(s).responsable.contrat.statut,
value(s).responsable.contrat.salaire
From LesDepartements d, table(d.services) s
Where value(s).responsable.contrat.statut = 'CDI'
Union
Select d.responsable.nom, d.responsable.contrat.statut,
d.responsable.contrat.salaire
From LesDepartements d
Where d.responsable.contrat.statut = 'CDD'
```

Question 4 : Les noms des employés qui sont responsables d'un département et membre d'un service dans un autre département.

2 pts

```
Select distinct d.responsable.nom
From LesDepartements d, LesDepartements d2, table(d2.services) s2,
table(value(s2).membres) m2
Where d.responsable = value(m2)
And ref(d) <> ref(d2)


Select d.responsable.nom
From LesDepartements d
Where d.responsable in (
     Select value (m2)
     From LesDepartements d2, table(d2.services) s2,
table(value(s2).membres) m2
     Where ref(d) <> ref(d2) )
```

Question 5 : Pour chaque service, afficher le nom du service et les noms des employés avec le plus grand salaire.

1 pts

```
Select s.nom, value(m).nom
From LesServices s, table(s.membres) m
Where value(m).contrat.salaire = (
          select max(value(m2).contrat.salaire)
          from table(s.membres) m2);

Select s.nom, value(m).nom
From LesServices s, table(s.membres) m
Where value(m).contrat.salaire >= ALL (
          select value(m2).contrat.salaire
          from table(s.membres) m2);
```

autre réponse avec not exist un salaire plus grand

Question 6: Pour chaque département et service, le salaire moyen des membres du service. La requête affiche le nom du département, le nom du service et le salaire moyen.

2 pts

```
Select d.nom, value(s).nom, avg( value(m).contrat.salaire) as salmoyen
From LesDepartements d, table(d.services) s, table(value(s).membres) m
Group by d.nom, value(s).nom;
```

~~Select d.nom, value(s).nom, avg(value(m).contrat.salaire) as salmoyen~~
~~From LesDepartements d, table(d.services) s, table(value(s).membres) m~~
~~Group by ref(d), value(s);~~

Exercice 2. XML et DTD — 3 pts

On considère la DTD suivante pour stocker des graphes dirigés dont les nœuds et les arcs sont étiquetés :

<!ELEMENT graphe (noeud*,arc*) >

<!ELEMENT noeud (label) >

<!ELEMENT arc (label) >

<!ELEMENT label (#PCDATA)>

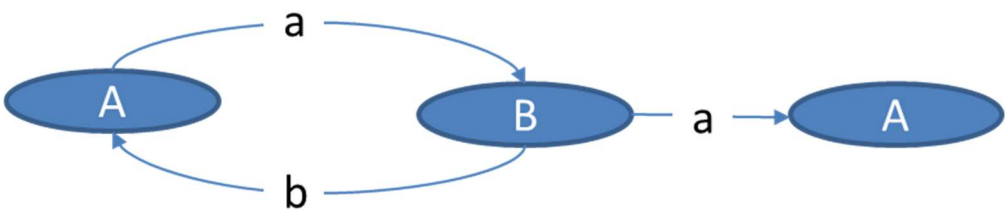
<!ATTLIST noeud nid ID #REQUIRED>

<!ATTLIST arc source IDREF #REQUIRED

dest IDREF #REQUIRED>

Les éléments label correspondent aux étiquettes des nœuds et des arcs.

Question 1. Transformer les deux graphes suivant en un seul document XML valide par rapport à la DTD.




```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE graphe SYSTEM "dtd2.dtd">
```

1 pt

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE graphe SYSTEM "dtd2.dtd">
```

```
<graphe>
<noeud nid="n1"><label>A</label></noeud>
<noeud nid="n2"><label>B</label></noeud>
<noeud nid="n3"><label>A</label></noeud>
<arc source="n1" dest="n2"><label>a</label></arc>
<arc source="n2" dest="n1"><label>b</label></arc>
<arc source="n2" dest="n3"><label>a</label></arc>
</graphe>
```

Question 2. Est-ce qu'il est possible de définir une DTD pour empêcher qu'il y ait des nœuds sans arcs sortants (le graphe précédent n'est pas valide, car le nœud de droite n'a pas d'arc sortant) ? Si oui, donnez la DTD correspondante. Sinon, expliquez pourquoi ?

2 pts

```
<!ELEMENT graphe (noeud*) >
<!ELEMENT noeud (label, arc+) >
<!ELEMENT arc (label) >
<!ELEMENT label (#PCDATA)>
<!ATTLIST noeud nid ID #REQUIRED>
<!ATTLIST arc dest IDREF #REQUIRED>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE graphe SYSTEM "dtd2.dtd">
<graphe>
<noeud nid="n1">
  <label>A</label>
  <arc dest="n2">
     <label>a</label>
  </arc>
</noeud>
<noeud nid="n2">
  <label>B</label>
  <arc dest="n1">
     <label>b</label>
  </arc>
</noeud>
```

```
</graphe>
```