Nom :	Prénom :	*page 1*

Module Bases de Données et Web
Examen du 25 janvier 2007
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 : DTD et XSchema 6 pts

Question 1. Complétez la DTD ci-dessous décrivant cette application. Les associations sont représentées par des attributs.

1. <?xml version ="1.0" encoding="ISO-8859-1" ?>
2. < !ELEMENT Planification ……..
3. < !ELEMENT Machine …….
4. < !ATTLIST Machine ……
5. < !ELEMENT Tache ……
6. < !ATTLIST Tache ……

```
< ?xml version ="1.0" encoding="ISO-8859-1" ?>
< !ELEMENT Planification((Machine)+|(Tache)*)>
< !ELEMENT Machine EMPTY>
< !ATTLIST Machine numero ID #REQUIRED
                   Description CDATA #IMPLIED
                  Realise IDREFS #IMPLIED>
< !ELEMENT Machine EMPTY>
< !ATTLIST Tache  numero ID #REQUIRED
                   Duree CDATA #IMPLIED
                   Est-realisee-par IDREF  #REQUIRED
                  précède IDREFS #IMPLIED
                  est-précédée-par IDREFS #IMPLIED>
```

Question 2. Pour chacune des contraintes suivantes, indiquez la ligne à modifier et les modifications à faire pour exprimer la contrainte (si c'est possible).

a) Le système comprend au maximum deux machines

Ligne à modifier :

Modifications :

Ligne à modifier : 2

Modif : < !ELEMENT Planification(Machine|(Machine ?)|(Tache)*)>

b) Le numéro d'une tâche est un entier dans l'intervalle [1, 100]

Ligne à modifier :

Modifications :

Pas possible, ou alors énumérer toutes les valeurs (ce qui devient problématique à partir d'une certaine taille)

c) Une tâche est réalisée par une et une seule machine

Ligne à modifier :

Modifications :

Ligne à modifier : 6

Modif : Est-realisee-par IDREF #REQUIRED (IDREF doit être au singulier + required)

d) Le numéro d'une machine est unique et obligatoire

Ligne à modifier :

Modifications :

Ligne à modifier : 4

Modif : < !ATTLIST Machine numero ID #REQUIRED

e) Le champ description d'une machine est une chaîne de caractères de longueur 50.

Ligne à modifier :

Modifications :

Modif : pas possible

f) Une tâche précède une ou plusieurs tâches

Ligne à modifier :

Modifications :

Ligne à modifier : 6

Modif : précède IDREFS #REQUIRED (Mettre IDREFS au pluriel et required)

Question 3. On souhaite maintenant modéliser cette application en XSchema.

3.1 Définir en Xschema, l'élément `machine`.

<xs:element name="machine"……

```
<xs:element name="machine">
     <xs:complexType>
     <xs:attribute name="numero" type="xs:integer" use="required"/>
     <xs:attribute name="description" type="xs:string" use="optional"/>
     <xs:attribute name="realise" type="xs:integer" use="optional"/>
```

```
     </xs:complexType>
    </xs:element>
```

3.2 Définir en Xschema, l'élément `tache`.

<xs:element name="tache"……

```
<xs:element name="tache"/>
     <xs:complexType>
     <xs:attribute name="numero" type="xs:integer" use="required"/>
     <xs:attribute name="duree" type="xs:integer" use="optional"/>
     <xs:attribute name="precede" type="xs:integer" use="optional"/>
     <xs:attribute name="est-realisee-par" type="xs:integer" use="required"/>
     <xs:attribute name="est-precedee-par" type="xs:integer" use="optional"/>
     </xs:complexType>
    </xs:element>
```

3.3. En utilisant ces deux définitions, complétez la description du schéma ci-dessous. Pour la lisibilité de la suite, pensez à numéroter les lignes.

1.<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"

targetNamespace="http://www.planification.org"

xmlns= "http://www.planification.org"

elementFormDefault="qualified">…….

2.<xs:element name= "planification" ……

3. <xs:complexType …….

4. <xs:…….

5..

< /xs:complexType>

</xs:schema>

```
1. <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.planification.org"
xmlns= "http://www.planification.org"
elementFormDefault="qualified">

2. <xs:element name= "planification" >
3.    <xs:complexType >
4.    <xs:choice…minOccurs="0" maxOccurs="unbounded">
5.          <xs:element ref="machine" />
6.          <xs:element ref="tache"/>
7.    </xs:choice>
8.   < /xs:complexType>

</xs:schema>
```

Question 4. Exprimez chacune des contraintes suivantes, en indiquant, s'il y a lieu, ce qu'il faut modifier, et le numéro de ligne où insérer la contrainte.

a) Le système comprend au maximum deux machines

Modifications :

Contrainte :

Ligne où insérer la contrainte :

Ligne à modifier : 5
```
Modif :  <xs:element ref="machine" minOccurs="0" maxOccurs="2"/>
```

b) Le numéro d'une tâche est un entier dans l'intervalle [1, 100]

Modifications :

Contrainte :

Ligne où insérer la contrainte :

Ligne à modifier : 10
```
Modif :
<xs:attribute name="numero" use="required">
     <xs:simpleType>
      <xs:restriction base="xs:integer">
      <xs:minInclusive value="1"/>
      <xs:maxInclusive value="100"/>
      </xs:restriction>
      </xs:simpleType>
</xs:attribute>
```

c) L'attribut `numéro` est un identifiant unique pour les éléments `machine` et `tache`.

Modifications :

Contrainte :

Ligne où insérer la contrainte :

```
<xs:key name="cléMachine">
     <xs:selector xpath="machine">
     <xs:field xpath="@numero">
</xs:key>

xs:key name="cléTache">
     <xs:selector xpath="tache">
     <xs:field xpath="@numero">
```

d) Une tâche est réalisée par une et une seule machine

Modifications :

Contrainte :

Ligne où insérer la contrainte :

Plusieurs solutions : mettre type = IDREF use=required

On peut aussi modifier la ligne 10. Supprimer l'attribut Est-réalisé par et mettre :
```
10.<xs:element name="tache"/>
     <xs:complexType>
     <xs:sequence minOccurs="0" maxOccurs="1">
          <xs:element name=est-realisé-par type=xs:integer/>
     </xs:sequence>
     <xs:attribute name="numero" type="xs:integer" use="required"/>
     <xs:attribute name="duree" type="xs:integer" use="optional"/>
     <xs:attribute name="precede" type="xs:integer" use="optional"/>
     <xs:attribute name="est-precedee-par" type="xs:integer" use="optional"/>
     </xs:complexType>
    </xs:element>
```

e) Le champ description d'une machine est une chaîne de caractères de longueur 50.

Modifications :

Contrainte:

Ligne où insérer la contrainte :

Ligne à modifier : élément machine

```
Modif :
<xs:attribute name="description"  use="optional">
     <xs:simpleType>
     <xs:restriction base="xs:string">
     <xs:length value="50"/>
     </xs:restriction>
     </xs:simpleType>
</xs:attribute>
```

f) Une tâche précède une ou plusieurs tâches

Contrainte :

Ligne où insérer la contrainte :

Ligne à insérer : après les clefs

```
Modif :  <xs:keyref name="précède" refer="cleTache">
     <xs:selector xpath="tache">
     <xs:field xpath="@précède" >
     </xs:keyref>
```

Question 5. On souhaite maintenant distinguer les tâches initiales des autres tâches. Une tâche initiale n'est précédée d'aucune autre tâche.

a) Définir le type `TypeTacheInitiale`

```
<xs:complexType name="TypeTacheInitiale">
     <xs:attribute name="numero" type="xs:integer" use="required"/>
     <xs:attribute name="duree" type="xs:integer" use="optional"/>
     <xs:attribute name="precede" type="xs:integer" use="optional"/>
     <xs:attribute name="est-realisee-par" type="xs:integer" use="required"/>
</xs:complexType>
```

b) définir le type `TypeTache`, en utilisant le type `TypeTacheInitiale.`

```
<xs:complexType name="TypeTache" >
<xs:complexContent>
     <xs:extension base="TypeTacheInitiale">
     <xs:attribute name="est-precedee-par" type="xs:integer" use="optional"/>
     </xs:extension>
</xs:complexContent>
</xs:complexType>
```

Exercice 2 : ODMG et OQL — 3 pts

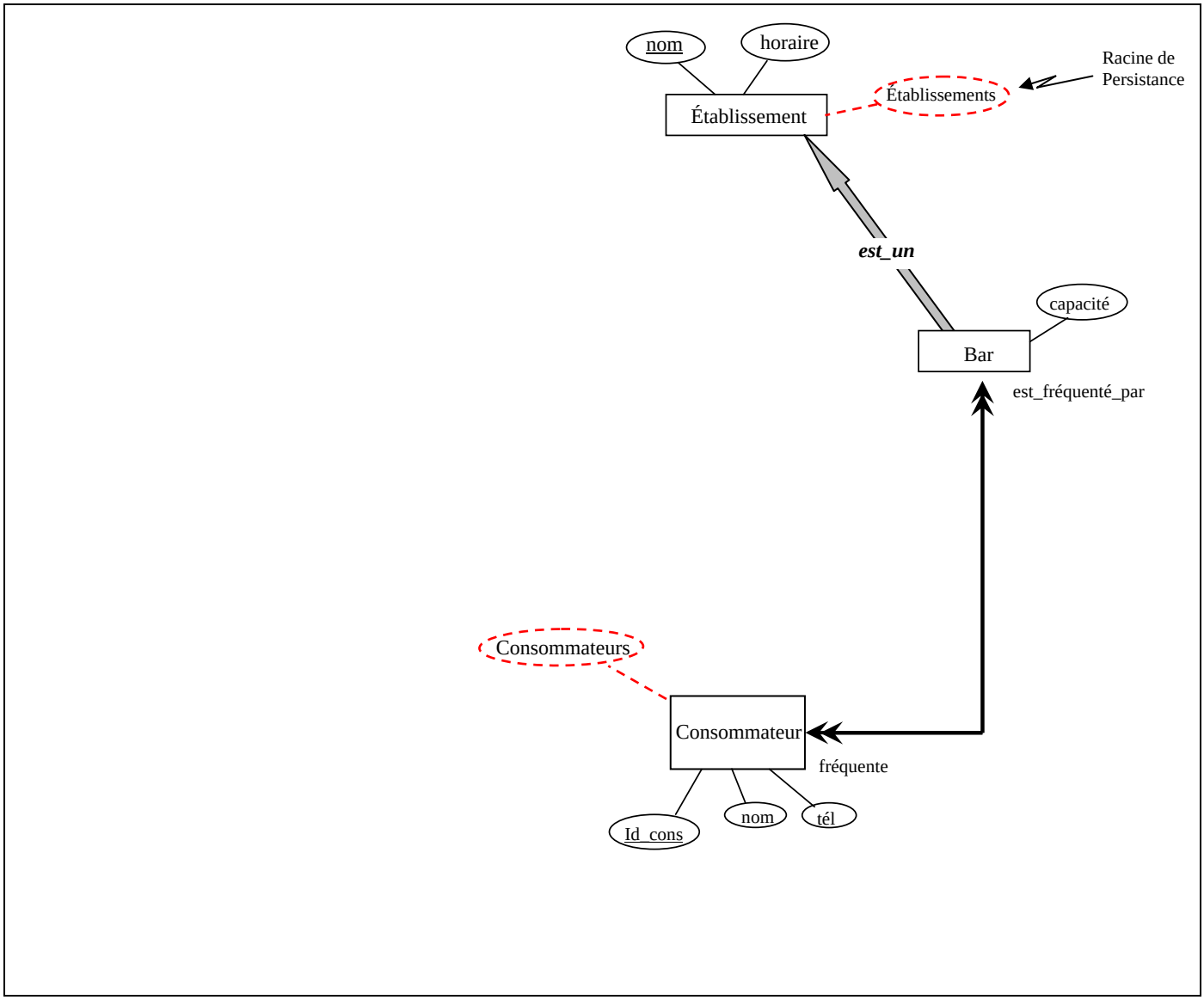
L'application qu'on souhaite modéliser concerne une étude sur la consommation et la vente de bières dans les différentes villes de France. Dans ce but, on souhaite représenter et gérer les données concernant les bières et les activités liées à leur consommation et à leur vente dans divers établissements.

Une ville est désignée par un nom (identifiant), un certain nombre d'habitants, et peut contenir plusieurs établissements de bières. Chaque établissement est désigné par un nom (identifiant), un horaire d'ouverture, un numéro de téléphone, et propose une sélection de bières. On distingue deux types d'établissements: les *magasins*, et les *bars*.

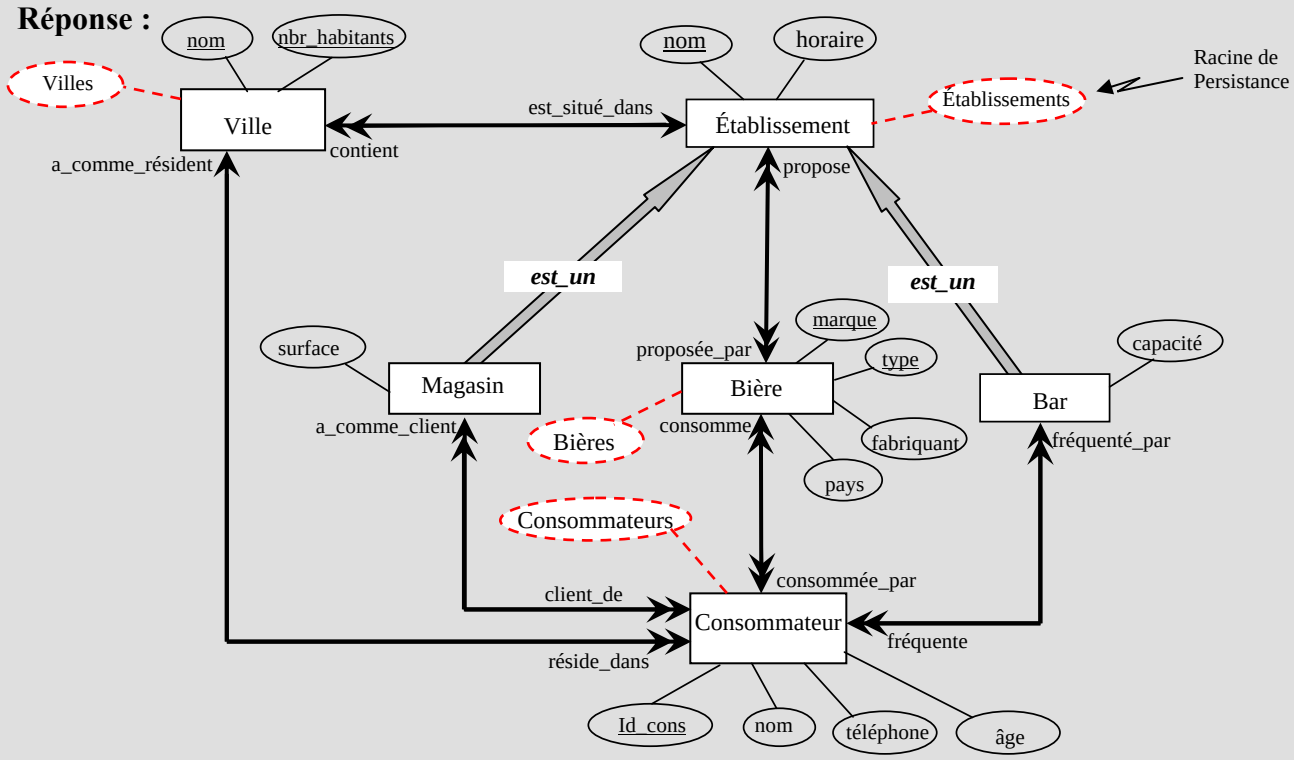
- Les *magasins* proposent des bières à l'achat, et sont caractérisés par une surface.
- Les bars proposent des bières à la consommation (sur place), et disposent d'une certaine capacité d'accueil.

Chaque bière est identifiée par une marque et un type, et est caractérisée par un prix, un fabriquant, un pays de fabrication, et est consommée par un certain nombre de consommateurs. Un consommateur est caractérisé par un id (identifiant), un nom, un numéro de téléphone, un âge, une ville de résidence, et peut consommer une variété de bières en les achetant dans différents magasins ou en les consommant sur place dans différents bars.

Question 1. Compléter le diagramme du schéma de cette application en utilisant la représentation graphique de la norme **ODMG**. Déterminer les racines de persistance correspondantes.



Réponse :



La définition ODL du sous-schéma de la Question 1 étant la suivante:

```
interface Établissement                    // extent définit une racine de persistance
        (extent Établissements key nom)
{           attribute string nom;
            attribute string horaire;
            boolean est_un_Bar();                  // cette méthode renvoie vrai si l'objet cible sur lequel elle
                                                   // est invoquée est  de type Bar, et faux autrement.
}

interface Bar : Établissement                      // " : " définit le lien d'héritage entre Bar et Établissement
{           attribute integer capacité;
            relationship set<Consommateur> est_fréquenté_par
            inverse Consommateur::fréquente;
}

interface Consommateur
          (extent Consommateurs key Id)
{           attribute string Id;
            attribute string nom;
            attribute string adresse;
            attribute string tél;
            attribute integer âge;
            relationship set<Bar> fréquente
               inverse Bar::est_fréquenté_par;
}
```

Question 2. Exprimer en OQL les requêtes suivantes sur ce sous-schéma :

R1. Trouver le nom des bars ayant plus de 100 clients (consommateurs) de moins de 25 ans.

Réponse:

```
SELECT    E.nom
FROM      E
IN        Etablissements
WHERE     E → est_un_Bar() AND COUNT (SELECT   C
                                      FROM     C
                                      IN       E.fréquenté_par
                                      WHERE    C.âge < 25) > 100;
```

R2. Trouver le nom des bars dont tous les clients (consommateurs) ont moins de 25 ans.

Réponse:

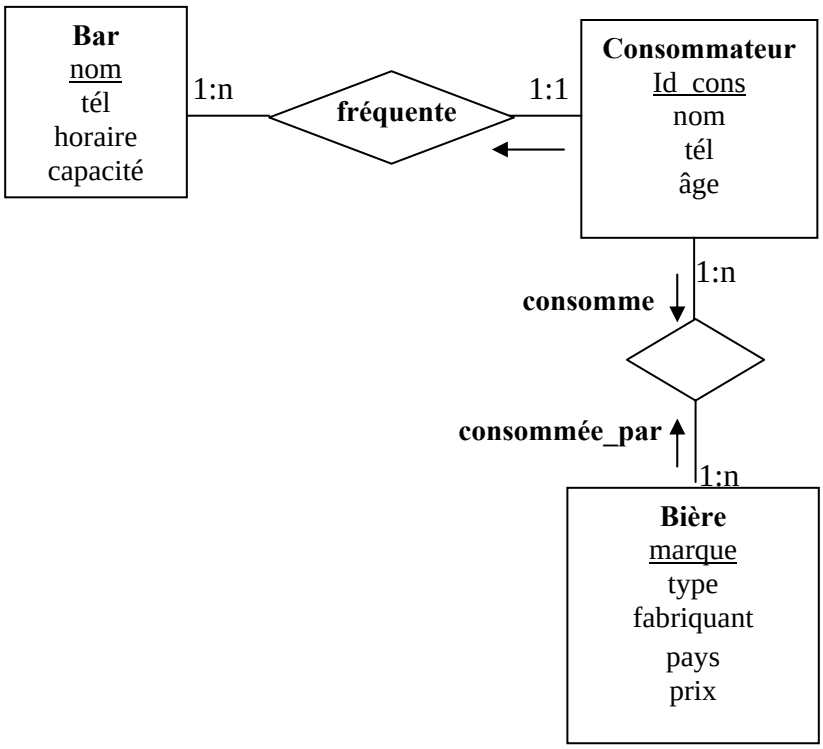
```
1.  SELECT      STRUCT (Nom:            E.nom,
                        Moyenne_Age:    AVG (SELECT   C.âge
                                             FROM     C
                                             IN       E.fréquenté_par))
    FROM        E
```

IN Etablissements
WHERE E → est_un_Bar() **AND NOT EXISTS** C **IN** E.est_fréquenté **:** C.âge >= 25;

Rem: l'utilisation du STRUCT n'est oas obligatoire…

Exercice 3 : SQL3 — 4 pts

Soit le schéma entité-association suivant:



Question 1. Traduire en SQL3 le schéma entité-association ci-dessus en implémentant les associations dans le sens désigné par les flèches noires.

Compléter les instructions suivantes :

```
CREATE TYPE Bar AS OBJECT(
nom          VARCHAR(20),
téléphone    VARCHAR(10),
horaire          VARCHAR(115),
capacité     NUMBER(4)
);
```

```
CREATE TYPE Bar AS OBJECT(
nom           VARCHAR(20),
téléphone     VARCHAR(10),
horaire             VARCHAR(115),  //hh:mn – hh:mn
capacité      NUMBER(4)
);

CREATE TYPE Bière;

CREATE TYPE obj_Bière AS OBJECT(
ptr  REF Bière;
);

CREATE TYPE EnsBière AS TABLE OF obj_Bière;

CREATE TYPE Consommateur AS OBJECT(
Id_cons       VARCHAR(10),
nom           VARCHAR(20),
téléphone     VARCHAR(10),
âge           NUMBER(3),
fréquente     REF Bar,
consomme      EnsBière
);
```

```
CREATE TYPE obj_Cons AS OBJECT(
ptr  REF Consommateur;
);

CREATE TYPE EnsCons AS TABLE OF obj_Cons;

CREATE TYPE Bière AS OBJECT(
Marque       VARCHAR(10),
Type         VARCHAR(10),
Fabriquant   VARCHAR(10),
Pays         VARCHAR(10),
Prix         NUMBER(4,2),
consommé_par  EnsCons
);
```

Question 2. Créer les tables nécessaires au stockage des objets Bar, Bière, et consommateur.

Compléter les instructions suivantes :

```
CREATE TABLE LesBars OF Bars;
```

```
CREATE TABLE LesBars OF Bars ;

CREATE TABLE LesBières OF Bière
NESTED TABLE consommé_par STORE AS table_consommé_par ;

CREATE TABLE LesConsommateurs OF consommateur
```

```
NESTED TABLE consomme STORE AS table_consomme ;
```

Question 3. Ecrire les requêtes suivantes en SQL3

R1. Trouver le nom des bars ayant plus de 100 clients (consommateurs) de moins de 25 ans.

```
Select   C.fréquente.nom
From     LesConsommateurs C
Where    C.âge < 25
Group    by C.fréquente
Having   count(*) >= 100;
```

R2. Trouver le nom des consommateurs qui ont consommé au moins une fois la bière de la marque Chimay.

Réponse:

```
Select   C.nom
From     LesConsommateurs C, table (C.consomme) B
Where    B.ptr.marque = "Chimay";
```

Exercice 4 : XPath et XQuery 7 pts

Soit le fichier XML family.xml suivant :

```
 <?xml version="1.0" encoding="UTF-8"?>
<base>
   <personne id = "p1" genre = "m">
       <prenom>Pierre</prenom>
       <nom>Rao</nom>
       <age>58</age>
       <conjoint idref = "p2"/>
   </personne>

   <personne id = "p2" genre = "f">
        <prenom>Isabelle</prenom>
        <nom>Rao</nom>
        <age>61</age>
        <conjoint idref = "p1"/>
   </personne>

   <personne id = "p3" genre = "m">
        <prenom>Fernand</prenom>
        <nom>Daile</nom>
        <age>32</age>
        <conjoint idref = "p4"/>
   </personne>

   <personne id = "p4" genre = "f">
        <prenom>Fernande</prenom>
        <nom>Ehle</nom>
        <age>27</age>
        <conjoint idref = "p3"/>
        <pere idref = "p1"/>
        <mere idref = "p2"/>
   </personne>

   <personne id = "p5" genre = "f">
        <prenom>Elise</prenom>
        <nom>Lettra</nom>
        <age>27</age>
        <conjoint idref = "p7"/>
        <pere idref = "p1"/>
       <mere idref = "p2"/>
   </personne>

   <personne id = "p6" genre = "f">
        <prenom>Lucie</prenom>
        <nom>Daile</nom>
        <age>6</age>
    <personne id = "p8" genre = "m">
         <prenom>Louis</prenom>
         <nom>Daile</nom>
         <age>6</age>
         <pere idref = "p3"/>
         <mere idref = "p4"/>
    </personne>

  <personne id = "p9" genre = "f">
         <prenom>Marie</prenom>
         <nom>Daile</nom>
         <age>5</age>
         <pere idref = "p3"/>
         <mere idref = "p4"/>
    </personne>

    <famille id = "f1">
       <epoux idref = "p1"/>
       <epouse idref = "p2"/>
       <mariage>
           <lieu>Bandol</lieu>
           <date>12 fevrier 1975</date>
       </mariage>
     </famille>

     <famille id = "f2">
       <epoux idref = "p3"/>
       <epouse idref = "p4"/>
       <mariage>
           <lieu>Lyon</lieu>
           <date>12 mars 1999</date>
       </mariage>
     </famille>

     <famille id = "f3">
       <epoux idref = "p7"/>
       <epouse idref = "p5"/>
       <mariage>
           <lieu>Toulouse</lieu>
           <date>3 avril 2001</date>
       </mariage>
     </famille>

</base>
```

```
        <pere idref = "p3"/>
        <mere idref = "p4"/>
    </personne>

    <personne id = "p7" genre = "m">
        <prenom>Jean</prenom>
        <nom>Lettra</nom>
        <age>30</age>
        <conjoint idref = "p5"/>
    </personne>
```

Question 1. Exprimez en XPath les requêtes suivantes :

1. Les petits-enfants de Pierre Rao. Résultat : les personnes d'id p6, p8, p9

0.75 point

//personne[pere/@idref=//personne[pere/@idref=//personne[nom='Rao' and prenom='Pierre']/@id]/@id or mere/@idref=//personne[pere/@idref=//personne[nom='Rao' and prenom='Pierre']/@id]/@id]

2. Nom des personnes dont le père s'est marié le 12 février 1975. Résultat : Ehle, Lettra.

0.75 point

//personne[//famille[mariage/date='12 fevrier 1975']/epoux/@idref=pere/@idref]

3. Prénom des célibataires. Résultat : Lucie, Louis, Marie.

0.75 point

//personne[not(conjoint)]/prenom

4. Ages des sœurs (mêmes père et mère) de Lucie Daile. Résultat : 5

0.75 point

```
//personne[pere/@idref=//personne[nom='Daile' and
prenom='Lucie']/pere/@idref and mere/@idref=//personne[nom='Daile' and
prenom='Lucie']/mere/@idref and @genre='f' and @id!=//personne[nom='Daile' and
prenom='Lucie']/@id]/age
```

Question 2. Ecrire en XQuery les requêtes suivantes :

1. Donner, lorsque c'est possible, la liste des hommes (nom et prénom) et la différence d'âge avec leur père. Le résultat doit être :

```
<root>
  <Personne>
    <nom>Daile</nom>
    <prenom>Louis</prenom>
    <diff-age>26</diff-age>
  </Personne>
</root>
```

```
1 point

<root>
 { for $p in document("family.xml")//personne[@genre='m']
  let $nomp:=$p/nom
   let $prenomp:=$p/prenom
   let $pere:=document("family.xml")//personne[@id=$p/pere/@idref]
   where exists($pere)
   return
     <Personne>
     {$nomp}
     {$prenomp}
     <diff-age>
      {$pere/age - $p/age}
     </diff-age>
     </Personne>}
</root>
```

2 .Donner la liste des hommes (nom et prénom), avec, s'il existe, le prénom du père.

Le résultat doit être :

```
<root>
  <Personne><nom>Rao</nom><prenom>Pierre</prenom></Personne>
  <Personne><nom>Daile</nom><prenom>Fernand</prenom></Personne>
  <Personne><nom>Lettra</nom><prenom>Jean</prenom></Personne>
  <Personne>
    <nom>Daile</nom>
```

```
   <prenom>Louis</prenom>
   <prenompere>Fernand</prenompere>
  </Personne>
</root>
```

1 point

```
<root>
 { for $p in document("family.xml")//personne[@genre='m']
  let $nomp:=$p/nom
   let $prenomp:=$p/prenom
   let $pere:=document("family.xml")//personne[@id=$p/pere/@idref]
   return
     <Personne>
     {$nomp}
     {$prenomp}
     {if ($pere) then
     <prenompere>
      {$pere/prenom/text()}
     </prenompere>
     else () }
     </Personne>}
</root>
```

4. Donner la moyenne d’âge des gendres (maris des filles) de Pierre Rao.

On obtient le résultat suivant : <root>31</root>

1 point

```
<root>
 { let $pr:=document("family.xml")//personne[nom='Rao' and prenom='Pierre']
let $fpr:=document("family.xml")//personne[pere/@idref=$pr/@id and @genre='f']
let $gpr:=document("family.xml")//personne[conjoint/@idref=$fpr/@id and @genre='m']
return avg($gpr/age)
 }
</root>
```