Nom :	Prénom :	*page 1*

Module Bases de Données et Web
Examen du 21 décembre 2006
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 : Questions diverses 2 pts

Question 1

Enumérer au moins deux avantages de la répartition d'une base de données sur plusieurs sites.

Question 2

Enumérer deux limites du modèle DTD par rapport au modèle XSchéma.

Question 3

Enumérer deux avantages d'un SGBD orienté objet par rapport à un langage de programmation orienté objet.

Question 4

Enumérer deux avantages du modèle XML par rapport aux modèles de données classiques, tels que le relationnel ou l'orienté objet.

Exercice 2 : BD réparties 4 pts

La base de données d'un institut de recherche en Biologie a le schéma global suivant:

LABORATOIRE (codeL, nom, sigle, région, adresse, directeur)
CHERCHEUR (#chercheur, codeL, nom, prénom, spécialité, salaire, prime)
PROJET (codeP, codeL, titre, chef, budget, durée, date_début)
PARTICIPE (#chercheur, codeP, rôle, taux_participation)

L'institut est formé de trois centres de recherche situés dans trois régions Françaises : Paris, Lille, et Grenoble. Chaque centre est formé de plusieurs laboratoires. En supposant que la base de données de l'institut est répartie sur trois sites informatiques correspondant aux centres de Paris, de Lille, et de Grenoble, proposer une bonne décomposition de la base sur ces trois sites en se basant sur les hypothèses suivantes.

- Chaque centre gère ses propres données.
- L'attribut *région* de LABORATOIRE prend une des valeurs suivantes: "Paris", "Lille", "Grenoble".
- L'attribut *spécialité* de CHERCHEUR prend une des valeurs suivantes : "Virologie", "Parasitologie", "Neuroscience", "Microbiologie", "Epidémiologie", "Génétique", " Immunologie".
- codeL (resp. #chercheur, codeP) est clé primaire de LABORATOIRE (resp. CHERCHEUR, PROJET).
- Le chef d'un projet et le directeur d'un laboratoire sont des chercheurs désignés par leurs numéros.
- Chaque chercheur est rattaché à un laboratoire donné, et peut participer à plusieurs projets de son laboratoire.

Question 1

Donner la définition des différents fragments en utilisant les opérateurs de l'algèbre relationnelle.

Réponse:

$i \in \{Paris, Lille, Grenoble\}$

$P_i \in \{(région = 'Paris), (région = 'Lille), (région = 'Grenoble')\}$

$LABORATOIRE_i = \sigma_{P_i}$ (LABORATOIRE)

$CHERCHEUR_i$ = CHERCHEUR $\ltimes$ $LABORATOIRE_i$

$PROJET_i$ = PROJET $\ltimes$ $LABORATOIRE_i$

$PARTICIPE_i$ = PARTICIPE $\ltimes$ $PROJET_i$

Question 2

Dans cette question, le centre de la région Parisienne tient également lieu de siège pour l'institut. Afin de séparer les données scientifiques des données administratives, supposons maintenant que la relation CHERCHEUR est répartie sur les trois sites informatiques de Paris, de Lille, et de Grenoble comme suit:

$i \in \{Paris, Lille, Grenoble\}$

$P_i \in \{(région = 'Paris), (région = 'Lille'), (région = 'Grenoble')\}$

$CHERCHEUR\_SC_i = \pi$#chercheur, codeL, nom, prénom, spécialité (σ_{P_i} (CHERCHEUR))

$CHERCHEUR\_ADMIN_{Paris} = \pi$#chercheur, nom, prénom, salaire, prime (CHERCHEUR)

$CHERCHEUR\_ADMIN_{Lille} = CHERCHEUR\_ADMIN_{Grenoble} = \varnothing$

Proposez un plan d'exécution réparti de la requête permettant de retrouver, pour chaque laboratoire, le revenu le plus élevé (salaire + prime) d'un chef de projet du laboratoire. On suppose que la requête est émise par le siège de l'institut. Le plan proposé doit optimiser le coût de communication inter sites.

Réponse:

Soit R1 la sous-requête suivante:

```
SELECT   #chercheur, C.codeL
FROM     CHERCHEUR_SC C, PROJET P
WHERE    #chercheur = chef;
```

-- Le site S_{Paris} execute:

SEND R1 **TO** S_i; // i ∈ {Paris, Lille, Grenoble}

-- Les sites S_i exécutent:

RECEIVE R1 **FROM** S_{Paris};

$(R1) \rightarrow T_i$;

SEND T_i **TO** S_{Paris};

-- Le site S_{Paris} execute:

RECEIVE T_i **FROM** S_i;

(**SELECT** * **FROM** T_{Paris}) **UNION**

(**SELECT** *

```
 FROM      T_Lille ) UNION
( SELECT    *
 FROM      T_Grenoble ) → Temp

SELECT      T.codeL, Max(salaire + prime)
FROM        CHERCHEUR_ADMIN CA, Temp T
WHERE     CA.#chercheur = T.#chercheur
GROUP BY  T.codeL;
```

Exercice 3 : DTD et XMLSchema 6 pts

L'objectif de cet exercice est d'étudier les fichiers *XMLSchema.dtd* et *XMLSchema.xsd,* qui décrivent les éléments de XMLSchema.

Question 1. **(2pts)**

On demande d'écrire la DTD des éléments *xs:key et xs :keyref* de XMLSchema.

On pourra utiliser l'entité *%XPathExpr ;* , qui décrit l'ensemble des expressions XPath, pour définir les expressions XPath. On suppose cette entité déjà définie.

```
Réponse :

<!ELEMENT xs:key    (xs:selector, (xs:field)+)>
<!ATTLIST xs:key
          name ID  #REQUIRED>

<!ELEMENT xs:keyref (xs:selector, (xs:field)+)>
<!ATTLIST xs:keyref
          name  CDATA  #REQUIRED
         refer  IDREF  #REQUIRED>

<!ELEMENT xs:selector >
<!ATTLIST xs:selector
          xpath %XPathExpr; #REQUIRED>

<!ELEMENT xs:field >
<!ATTLIST xs:field
          xpath %XPathExpr; #REQUIRED>
```

Question2 (2pts)

Soit la DTD suivante, décrivant l'élément *xs :attribute* de XMLSchema.

```
<!ELEMENT xs:attribute ((xs:simpleType)?)>
<!ATTLIST xs:attribute
          name      CDATA     #IMPLIED
          id        ID            #IMPLIED
          type      CDATA       #IMPLIED
          use       (prohibited|optional|required) #IMPLIED
          default   CDATA         #IMPLIED
          fixed     CDATA         #IMPLIED>
```

On demande d'écrire la définition de cet élément *xs:attribute* en XMLSchema, en définissant le type TypeAttribute indépendamment.

On suppose que l'élément *simpleType* est de type *TypeSimpleType*

```
Réponse :
<xs:complexType name="TypeAttribute">
  <xs:complexContent>
          <xs:sequence>
     <xs:element name="simpleType" minOccurs="0" type="xs:localSimpleType"/>
    </xs:sequence>
    <xs:attribute name="type" type="xs:string"/>
    <xs:attribute name="use" use="optional" default="optional">
     <xs:simpleType>
      <xs:restriction base="xs:string">
       <xs:enumeration value="prohibited"/>
       <xs:enumeration value="optional"/>
       <xs:enumeration value="required"/>
      </xs:restriction>
     </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="default" type="xs:string"/>
    <xs:attribute name="fixed" type="xs:string"/>
        </xs:complexContent>
 </xs:complexType>

<xs :element name= "attribute" type="TypeAttribute">
```

Question 3 (1pt)

Quelles contraintes, sur un schéma XMLSchema, impliquent les extraits suivants, du fichier XMLSchema.xsd ?

1)

```
<xs:key name="type">
        <xs:selector xpath="xs:complexType|xs:simpleType"/>
        <xs:field xpath="@name"/>
 </xs:key>
```

Réponse : unicité des noms de type (parmi l'ensemble des types complexes et simples non anonymes)

2)

```
<xs:key name="identityConstraint">
        <xs:selector xpath=".//xs:key|.//xs:unique|.//xs:keyref"/>
        <xs:field xpath="@name"/>
 </xs:key>
```

Réponse : unicité des noms de contraintes

Question 4. (1 pt)

Dans la définition de l'élément *xs:element*, les noms d'éléments utilisés comme valeur de l'attribut *ref* doivent mentionner des éléments existants (c'est-à-dire la valeur de l'attribut *name* d'un autre élément existant).

On demande de définir cette contrainte à l'aide de *keyref*, en se basant sur la contrainte de clé appelée "element" définie ci dessous:

```
<xs:key name="element">
 <xs:selector xpath="xs:element"/>
 <xs:field xpath="@name"/>
</xs:key>
```

Réponse :

```
<xs:keyref name=”RefConstraint” refer=”element”>
<xs:selector xpath=”xs:element”/>
<xs:field xpath=”@ref”/>
```

Exercice 4 : XPath et XQuery 8 pts

Soit le fichier XML Atlas.xml suivant :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE atlas SYSTEM "atlas.dtd">
<atlas>
  <pays n="p2" population="82" continent="c1">
    <nom>Allemagne</nom>
    <langue>Allemand</langue>
    <frontiere pays="p1"/>
  </pays>
  <pays n="p1" population="60" continent="c1" >
    <nom>France</nom>
    <langue pourcentage="100">Français</langue>
    <langue pourcentage="1">Corse</langue>
    <frontiere pays="p2"/>
    <frontiere pays="p3"/>
  </pays>
  <pays n="p3" population="40" continent="c1">
    <nom>Espagne</nom>
```

```
    <langue pourcentage="74">Espagnol</langue>
    <langue pourcentage="17">Catalan</langue>
    <langue pourcentage="7">Galicien</langue>
    <frontiere pays="p1"/>
  </pays>
  <pays n="p4" population="76" continent="c2">
    <nom>Egypte</nom>
    <langue>Arabe</langue>
  </pays>

  <continent n="c1" nom="Europe" superficie="10"/>
  <continent n="c2" nom="Afrique" superficie="30"/>

  <mer n="m1" nom="Mer Mediterranee" profondeur="5120">
    <situation pays="p1"/> <situation pays="p3"/> <situation pays="p4"/>
  </mer>

  <montagne n="M1" nom="Alpes" altitude="4810">
    <situation pays="p1"/> <situation pays="p2"/>
  </montagne>
    <montagne n="M2" nom="Cevennes" altitude="1700">
    <situation pays="p1"/>
  </montagne>
</atlas>
```

Question 1. Exprimez en XPath les requêtes suivantes :

1. Langues qui ne sont pas parlées en Europe.

//pays[@continent=//continent[not(@nom=’europe’)]/langue

2. Nom des pays méditerranéens (ayant un accès à la mer Méditerranée)

//pays[@n=//mer[@nom=’Mer méditerrannée’]/situation/@pays

3. Nom des pays ayant des frontières avec plus de 3 autres pays.

//pays[count(frontiere)>3]/nom

4. Pays de plus de 40 millions d'habitants n'ayant pas de montagne de sommet supérieur à 2000.

//pays[@population>40][not(@n=//montagne[@altitude>2000]/situation/@pays)]

Autres questions éventuellement :

Langues des pays d'Europe parlées par moins de 40% de la population du pays.

//pays[@continent=//continent[@nom='europe']/langue[pourcentage<=40]

Question 2. Expliquer en une phrase en français ce que signifient les expressions XPath suivantes, et donner le résultat de leur évaluation sur le document Atlas.xml.

1. //pays[population<80 and position()=2]/nom

Renvoie le 2eme élément pays s'il a une population inférieure à 80.

Renvoie France.

2. //pays[population<80][position()=2]/nom

Sélectionne tous les éléments pays qui ont une population inférieure à 80 et renvoie le 2eme.

Renvoie Espagne

Question 3. Ecrire en XQuery les requêtes suivantes :

1 . Donner les noms des pays qui sont voisins du pays d'identificateur p1 (n='p1').

Le résultat doit être :

<root>

<nom>**Allemagne**</nom>

<nom>**Espagne**</nom>

</root>

1 point

```
<root>
 { for $p in document("atlas.xml")//pays
   where $p/frontiere/@pays='p1'
  return $p/nom }
</root>
```

2. Donner les noms des pays dont toutes les langues sont parlées par au moins 2% de la population. Le résultat doit être :

<root>

<nom>**Espagne**</nom>

</root>

1.25 point

```
<root>
 { for $p in document("atlas.xml")//pays
   where every $l in $p/langue satisfies $l/@pourcentage > 2
  return $p/nom
 }
</root>
```

3. Donner, dans l'ordre alphabétique, tous les noms de continent, suivi chacun par les noms des pays composant le continent, dans l'ordre alphabétique également. On obtient une hiérarchie à trois niveaux suivante :

<monde>

<continent nom="**Afrique**">

<pays>**Egypte**</pays>

</continent>

<continent nom="**Europe**">

<pays>**Allemagne**</pays>

<pays>**Espagne**</pays>

<pays>**France**</pays>

</continent>

</monde>

1.25 point

```
<root>
 { for $c in document("atlas.xml")//continent
        order by $c/@nom
        return  <continent>
                     {$c/@nom}
                     { for $p in document("atlas.xml")//pays
                        where $p/@continent=$c/@n
                        order by $p/nom
                        return <pays> {$p/nom/text()} </pays>
                     }
                </continent>
 }
</root>
```

4. Donner tous les noms de montagne, suivi chacun par les noms des pays dans lesquels est située la montagne. On obtient le résultat suivant :

<root>

<montagne nom="**Alpes**">

<pays>**Allemagne**</pays>

<pays>**France**</pays>

</montagne>

<montagne nom="**Cévennes**">

<pays>**France**</pays>

</montagne>

</root>

1.25 point

```
<root>
 { for $m in document("atlas.xml")//montagne
        return  <montagne>
                     {$m/@nom}
                     { for $p in document("atlas.xml")//pays,
```

```
                        $s in $m/situation
                        where $s/@pays=$p/@n
                        return <pays> {$p/nom/text()} </pays>
                    }
                </montagne>
 }
</root>
```

5. Donner la population totale, pour le fichier Atlas.xml, de chaque continent.

Le résultat doit être :

<root>

  <continent nom="**Europe**">182</continent>

  <continent nom="**Afrique**">76</continent>

</root>

1.25 point

```
<root>
{ for $c in document("atlas.xml")//continent
let $pops := document("atlas.xml")//pays[@continent=$c/@n]
       return  <continent>
                    {$c/@nom}
                    {count($pops/@population)}
                    }
               </continent>
 }
</root>
```