EISTI Ecole Internationale des Sciences du Traitement de l'Information	**ING2-SIE : BASES DE DONNEES AVANCEES : EXAMEN DE 1ERE SESSION S3**
Rédigé par : Matthias Colin, Rachid Chelouah et Hervé de Milleville	Ref : *ING2-SIE-EX-BDD-AVA*
A l'intention de : Elèves-ingénieurs du parcours SIE2A	Créé le : 25/10/2012

1. Préambule

C'est un examen sur machine avec documents électroniques autorisés ainsi qu'une feuille papier A4 recto-verso.
L'examen dure 3 heures.
Rendu électronique sous la forme d'une archive au format de nom : NomPrenomBDDA.(tgz |zip) contenant les fichiers demandés dans les exercices.

Outils autorisés :
- Oracle XE.
- SQL Developer

Avertissement : Ne perdez pas temps à compiler et à tester. Vous serez plutôt évalués sur vos capacités de réflexion et pas tant sur les capacités de programmation.

2. Problème étudié

Nous sommes dans un centre de formation sur 3 sites, Paris avec 8 bureaux, Marseille avec 5 bureaux et Lyon avec 2 bureaux, et on suppose que les bureaux sont de même taille. Les informations gérées sont les formateurs, les inscrits, les formations, les modules des formations, les inscriptions à des sessions de formations. Dans l'un des sites, en plus d'assurer des sessions de formations, les équipes de formateurs définissent de nouvelles formations et/ou font évoluer le contenu de formations existantes. Sur un site donné, 70% des requêtes portent sur les inscrits et les formations sur ce site et 35% sur les formateurs et 15% sur les modules, et on suppose que les formateurs peuvent intervenir sur plusieurs sites.

Le modèle logique de données associé à une base de données centralisées est

Formateur (idFormateur, nom, prenom, #idSite)

Site (idSite, nom, adresse, ville)

Inscrit (idInscrit, nom, prenom, adresse, ville, #idSite)

Formation(idFormation, nom, descriptif)

Module(idModule, nom, descriptif, nbHeures)

FormationModule(idFormation, IdModule)

FormationSession(idFS, idFormation, dateDebut, dateFin, nbInscrits, nbMaximum)

InscriptionSession(idFS, idInscrit)

1. Proposer un modèle de données réparti sur les 3 sites, et justifier votre choix
2. Que faut il faire pour qu'un employer d'un site puisse accéder à la base de données sur un autre site (donner la procédure complète)

3. Si pour des raisons économiques, on souhaite fusionner une ou deux tables reparties, lesquelles choisiriez vous et ou placeriez vous la ou les tables fusionnées.
4. Supposant que la table Inscrit est fragmenté horizontalement sur les 3 sites, donner la requête que doit écrire un employé de Paris pour voir afficher la liste des inscrites sur les 3 sites
5. Supposant que nous souhaitons avoir une instance du résultat de cette requête à Paris, comment doit on faire ?

2.1 PL/SQL

Dans un premier temps, on suppose qu'on est dans une base de données centralisées.

2.1.1

On considère une application de gestion de questionnaires posés par des professeurs à leurs étudiants sur les matières enseignées. Chaque questionnaire est associé à un certain nombre de questions. Ces questions peuvent avoir des réponses prédéfinies ou être à réponse libre ; dans le 1er cas, la réponse peut-être à choix unique ou multiple. On ne gère pas dans cet exercice le remplissage des questionnaires par les étudiants.

Les données sont gérées en XML dont on vous donne la modélisation en annexe (**scolarite.xsd**). Seuls les extraits de code correspondant aux exercices de l'examen sont demandés.

2.2 Partie XSL

2.2.1 Exercice 1

Ecrire la feuille XSL ***scolariteReport.xsl*** qui prend en entrée un fichier XML au format scolarite.xsd et qui affiche la liste des professeurs avec pour chacun :

1. son nom et son prénom ;
2. le nombre total de questionnaires le concernant ;
3. la liste des matières (libellé) pour lesquelles il a posé au moins un questionnaire, en précisant le nombre de questionnaires entre parenthèses.

Vous définirez pour ce travail deux templates pour l'élément professeur :

1. le premier simple qui permet d'obtenir le point 1 de l'affichage ;
2. le deuxième complet qui permet d'obtenir les points 1 à 3 de l'affichage.

Le format de sortie est libre entre texte, html ou xhtml.

2.2.2 Exercice 2

Compléter la feuille ***scolariteReport.xsl*** en ajoutant une fonction qui prend en paramètre une séquence d'entiers et renvoie celle-ci triée et sans doublons.
Tester cette fonction en ajoutant à l'affichage de l'exercice précédent une ligne avec la liste triée sans doublons des ECTS du fichier d'entrée

2.2.3 Exercice 3

Pour un nombre de questionnaires important, on souhaite placer ceux-ci dans des fichiers séparés du fichier complet de la scolarité. On propose de remplacer la liste des questionnaires dans l'élément scolarité par une liste de références vers des fichiers, chacun contenant un questionnaire.

1. Que faut-il changer au schéma XSD pour intégrer cette gestion des fichiers ?
2. Ecrire la feuille XSL ***triQuestionnaires.xsl*** qui permet de :
   - prendre en entrée un fichier au format XSD modifié de la question 1 (par exemple, scolarite.xml lié à une série de fichiers questionnaires : q1.xml, q2.xml, questionnaireAB.xml, etc.).
   - créer un fichier par matière ayant des questionnaires ; le nom des fichiers de sortie sera obtenu à partir du libellé de chaque matière (par exemple, java.xml, bdd.xml, etc.) ; chaque fichier suit le format scolarite.xsd non modifié, avec uniquement le professeur et la matière concernés et la liste des questionnaires de cette matière.

2.3 Partie FOP

On désire dans le cadre d'une documentation technique générer des documents imprimables décrivant le schéma xsd scolarité.xsd. Pour être clair, le document xml que l'on fournira en entrée de FOP sera scolarité.xsd.

2.3.1 Exercice 1

Ce document xsd est un ensemble de types simples ou complexes. Ce document doit décrire l'ensemble de ces types par ordre alphabétique de leurs noms. Il commence par le titre "Description des types" et il est suivi des descriptions de chaque type.

Une description d'un type complexe doit faire apparaître dans l'ordre :

1. son nom suivi de la mention "type complexe".
2. En dessous, ses attributs sous forme d'une liste (la puce étant un tiret). Le contenu de chaque item décrivant un attribut est composé de son nom suivi de son type. Ces deux renseignements doivent être séparés par le symbole ":".S'il n'y a pas d'attribut, on doit écrire "pas d'attribut".
3. En dessous, ses éléments sous forme d'un tableau :
   a. dans la première colonne le nom de l'élément,
   b. dans la deuxième colonne le type de l'élément,
   c. dans la troisième colonne la cardinalité minimum,
   d. dans la quatrième colonne la cardinalité maximum (on écrit "infinie" si ce n'est pas borné)

Une description de type simple doit faire apparaître dans l'ordre :

1. son nom suivi de la mention de la mention "type simple" suivi du nom du type de base
2. En dessous, les valeurs possibles :
   a. S'il s'agit une énumération alors on écrit "Valeurs : " suivi des valeurs séparées par des virgules et le tout délimité par des accolades (Ex **Valeurs : {ChoixUnique, ChoixMultiple, Ouverte}**)
   b. S'il s'agit d'une inclusion alors on écrit "Valeurs : " suivi des deux valeurs séparées par ", …," et le tout séparé par des crochets (Ex **Valeurs : [1, …, 5]**)

Dans chaque cas, le premier élément de description (nom et type) doit avoir des attributs graphiques (police de caractères, …) différents des autres éléments pour le faire apparaître comme un titre. De même, le titre "Description des types" doit apparaître comme un super titre.

2.3.2 Exercice 2

On vous demande d'ajouter à l'exercice précédent des bookmarks qui pointent vers chaque description des types. Le libellé du bookmark est le libellé du type.

2.4 Partie JAXP

2.4.1 Exercice 1

On considère la classe suivante :

class Scolarite {
private Document doc;
….
}

A tout moment de la vie d'un objet de cette classe, **doc** réfère vers un objet qui contient un arbre XML associé au schéma décrit dans l'annexe.

On vous demande d'écrire la méthode suivante dont le prototype est :

public void upsertMatiere(int pid, String libelle, int nbEcts);

Cette méthode insère une nouvelle matière dans l'arbre **doc** si la valeur du paramètre **pid** ne correspond à aucune valeur de l'attribut id des éléments **tMatiere** de l'objet **doc** sinon elle modifie l'élément correspondant en changeant le libellé et le nombre d'ECTS de cette matière.

2.4.2 Exercice 2

A l'aide de la technologie SAX, écrire un parser SAX prenant en entrée un document conforme au schéma de l'annexe et un nom de questionnaire. Le programme valide le format du questionnaire et affiche la liste des questions posées (libellé uniquement).

3. Annexes

3.1 Modèle des données XML : scolarite.xsd

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!-- L'élément Racine -->
<xsd:element name="scolarité" type="tScolarité"/>

<!-- type Liste de Modules -->
<xsd:complexType name="tScolarité">
<xsd:sequence>
<xsd:element name="professeur" type="tProfesseur" maxOccurs="unbounded" />
<xsd:element name="matière" type="tMatière" maxOccurs="unbounded" />
<xsd:element name="questionnaire" type="tQuestionnaire" maxOccurs="unbounded" />
</xsd:sequence>
</xsd:complexType>

<!-- type Professeur -->
<xsd:complexType name="tProfesseur">
<xsd:sequence>
<xsd:element name="nom" type="xsd:string" />
<xsd:element name="prénom" type="xsd:string" />
</xsd:sequence>
<xsd:attribute name="id" type="xsd:ID" />
</xsd:complexType>


<!-- type Matière -->
<xsd:complexType name="tMatiere">
```

```
<xsd:sequence>
<xsd:element name="libellé" type="xsd:string" />
<xsd:element name="nombreECTS" type="tECTS" />
</xsd:sequence>
<xsd:attribute name="id" type="xsd:ID" />
</xsd:complexType>

<xsd:simpleType name="tECTS">
<xsd:restriction base="xsd:integer">
<xsd:minInclusive value="1" />
<xsd:maxInclusive value="5" />
</xsd:restriction>
</xsd:simpleType>

<!-- type Questionnaire -->
<xsd:complexType name="tQuestionnaire">
<xsd:sequence>
<xsd:element name="libellé" type="xsd:string" />
<xsd:element name="question" type="tQuestion" minOccurs="0" maxOccurs="unbounded" />
</xsd:sequence>
<xsd:attribute name="idMatiere" type="xsd:ID" use="required" />
<xsd:attribute name="idProfesseur" type="xsd:ID" use="required" />
</xsd:complexType>

<!-- type Question -->
<xsd:complexType name="tQuestion">
<xsd:sequence>
<xsd:element name="libellé" type="xsd:string" />
<xsd:element name="réponse" type="xsd:string" minOccurs="0" maxOccurs="unbounded" />
</xsd:sequence>
<xsd:attribute name="typeQuestion" type="tTypeQuestion" use="required" />
</xsd:complexType>

<xsd:simpleType name="tTypeQuestion">
<xsd:restriction base="xsd:string">
<xsd:enumeration value="ChoixUnique" />
<xsd:enumeration value="ChoixMultiple" />
<xsd:enumeration value="Ouverte" />
</xsd:restriction>
</xsd:simpleType>

</xsd:schema>
```

.