Rapport du projet de l'UE MLBDA

Nom : BENSLIMANE **Prénom** : Amine **Groupe** : 1

Exercice 1 :

DTD 1

Pour cette DTD, on crée les types suivants :

(La dénomination des types est assez clair, il n'y a pas d'ambiguité)

T_Airport, T_Continent, T_Island, T_Desert, T_Mountain, T_Province, T_Country, T_mondial

Ainsi que leurs tables respectives :

LesAirports, LesContinents, LesIslands, LesDeserts, LesMountains, LesProvinces, LesCountrys, Mondial.

On stock les montagnes, les deserts et les islands dans la table LesProvinces à l'aide du type "AS TABLE OF".

Lors de la définition du "toXML()" de T_Province, il convient alors de récupérer toutes les données nécessaires des trois sous-tables respectives.

Le cas de Country étant analogue pour les éléments Continents, provinces et airports.

On finit par créer un type T_mondial et sa table respective qui englobera tout les countrys.

Lors des insertions, on établit des jointures pour récupérer des attributs qui ne sont pas dans la même table dans la base de données Mondial.

Par exemple, on récupère les provinces par GEO_ISLAND pour les Islands, GEO_DESERT pour les déserts etc.

On finit par faire un seul objet ligne dans notre table Mondial pour englober le tout.

--> Après validation, on observe que le fichier XML généré respecte la DTD.

Vous trouverez le contenu SQL sous le nom : projetMLBDA-ex1-BENSLIMANE-AMINE-DTD1.sql

DTD 2

On considère headquarter comme étant l'élément "city" d'organization.

Pour cette DTD, on crée les types suivants :

T_borders , T_langage, T_Country, T_organization, T_mondial

Ainsi que leurs tables respectives

LesBorders, LesLangages, LesCountrys, LesOrganizations, Mondial

Comme pour la DTD1, on observe la nécessité des stocker des éléments Langages et borders dans Country, On crée alors les types nécessaires (AS TABLE OF).

L'élément "headquarter" est annoté comme "child" à l'élément organization.

On finit par récupérer les éléments Organizations dans le toXML() de T_mondial comme pour la DTD1.

Lors des insertions, on remarque que la table Borders dans la base de données MONDIAL n'est pas symétrique, on insère alors alors deux fois en permutant "country1" et "country2".

Les autres insertions sont triviales.

On exporte le résultat xml et on vérifie avec la DTD2, --> Validé.

EXERCICE 2

Pour cet exercice, nous avons choisi de le partager en trois fichiers SQL, (Question 1 et 2 ensemble, question 3 et question 4) Pour éviter de créer les mêmes types différemment et éviter toute erreur et surtout avoir plus de lisibilité.

Q1:

Pour cette question, on reprend les types montagnes, deserts et islands déjà définis et on ajoute l'élément geo comme un "child" de country ;

Il suffit alors de rajouter les éléments montagnes, deserts et islands comme les fils de "geo".

Les insertions auront comme jointure GEO_MOUNTAIN, GEO_DESERT et GEO_ISLAND.

Le fichier XML obtenu vérifie la DTD.

Q2 : (élément peak)

L'élément peak est défini comme la plus haute altitude d'une montagne dans un pays

On calcule alors en initialisant peak à -1 ;

Et pour chaque montagne que l'on trouve, si notre peak actuel est inférieur à son altitude, alors peak reçoit la nouvelle valeur.

Lorsqu'on parcours toutes les montagnes, on est sûr qu'à la fin de la boucle, peak contient l'altitude la plus élevée pour un country.

(Bien évidemment, on répétera cette opération pour chaque Country trouvé, c'est pour cela que la méthode doit être implémentée dans T_country)

Q3 : (élément contCountries)

Il s'agit d'un simple filtrage en continents, les éléments country doivent maintenant avoir leur continent comme attribut, contCountries est un "child" de country.

Pour border, on définit son type T_border ayant countrycode1, countrycode2 et length afin de faire des insertions simples à l'aide de la table BORDER présente dans la BDD.

Mais afin de joindre contCountries et les éléments border, on définit le country courant comme countrycode1 et donc aura pour voisin countrycode2.

On obtient donc les pays voisins et en même temps qui sont dans le même continent que countrycode1 (car jointure avec contCountries).

Le fichier XML obtenu est conforme à la DTD imposée.

Q4 : (attribut blength)

L'attribut blength contient la frontière totale pour un pays courant, on le calcul donc dans la méthode toXML() de T_Country comme suit :

On l'initialise à 0;

À chaque nouvel élément border rencontré (et donc nouvelle frontière)

Rajouter blength := blength + length

Lorsqu'on parcours tout les voisins d'un pays, blength contiendra la somme de toutes ses frontières ;

EXERCICE 3

Q1 :

Le but étant de trouver la séquence des pays les plus peuplés de chaque continent, on propose la DTD suivante :

<!ELEMENT ex3 (continent+)>

<!ELEMENT continent (country+)>

<!ATTLIST continent name CDATA #REQUIRED>

<!ELEMENT country EMPTY>

<!ATTLIST country name CDATA #REQUIRED population CDATA #REQUIRED>

Ainsi le fichier XML aura comme structure :

```
<ex3>
  <continent name="America">
    <country name="Antigua and Barbuda" population="65647"/>
          ….
          …
</continent>
  <continent name="Australia/Oceania">
    <country name="American Samoa" population="65628"/>
    <country name="Australia" population="18260863"/>
          …
          …
```

La requete xPath est :

//country[not(@population <= preceding-sibling::country/@population) and not(@population <=following-sibling::country/@population)]

Il suffit d'imposer que la population du pays courant ne doit pas être inférieur ou égale à tout ses frères (precedings sibling et following sibling)

Q2 :

Pour cette question, on définit la DTD suivante :

<!ELEMENT ex3 (country+)>

<!ELEMENT country (organization*)>

<!ATTLIST country name CDATA #REQUIRED>

<!ELEMENT organization EMPTY>

<!ATTLIST organization name CDATA #REQUIRED date CDATA #REQUIRED>

Q3 :

Pour cette question, la DTD sera la suivante :

<!ELEMENT ex3 (province+)>

<!ELEMENT province (montagne*)>

<!ATTLIST province name CDATA #REQUIRED>

<!ELEMENT montagne EMPTY>

<!ATTLIST montagne name CDATA #REQUIRED altitude CDATA #REQUIRED latitude CDATA #REQUIRED longitude CDATA #REQUIRED>

Q4 :

La dtd sera comme suit :

<!ELEMENT ex3 (country+)>

<!ELEMENT country (riviere*)>

<!ATTLIST country name CDATA #REQUIRED>

<!ELEMENT riviere EMPTY>

<!ATTLIST riviere name CDATA #REQUIRED>

L'expression xPath est : //country/riviere

Q5 :

Pour cette question, on reprend la DTD de l'exercice 2 question 4, c'est à dire :

<!ELEMENT ex2 (country+) >

<!ELEMENT country (contCountries) >

<!ATTLIST country name CDATA #REQUIRED blength CDATA #REQUIRED >

<!ELEMENT contCountries (border*) >

<!ELEMENT border EMPTY>

<!ATTLIST border countryCode CDATA #REQUIRED

length CDATA #REQUIRED >

L'expression xPath est :

//country[not(@blength<= preceding-sibling::country/@blength) and not(@blength <=following-sibling::country/@blength)]