TD1 Exercice 1 :

Voir « ex1.xml »

Exercice 2:

1)

```
┌──(root㉿cb89a520012b)-[/nosql]
└─# xmllint --valid cd-1.xml --noout

┌──(root㉿cb89a520012b)-[/nosql]
└─#
```

Le fichier cd1-xml est validé contre le dtd cd.dtd

2)

```
┌──(root㉿cb89a520012b)-[/nosql]
└─# xmllint --valid cd-2-old.xml --noout
cd-2-old.xml:12: element performance: validity error : Element performance content does not follow the DTD, expecting (composition , soloi
st? , (orchestra , conductor)?), got (orchestra soloist composition )
    </performance>
                  ^
cd-2-old.xml:16: element minutes: validity error : No declaration for element minutes
      <minutes>105</minutes>
                           ^
cd-2-old.xml:17: element length: validity error : Element length was declared #PCDATA but contains non text nodes
    </length>
             ^
cd-2-old.xml:18: element CD: validity error : Element CD content does not follow the DTD, expecting (composer , performance+ , publisher ,
 length?), got (composer performance publisher publisher length )
  </CD>
       ^
```

Le fichier cd2.xml n'est pas validé contre le même dtd.

Il faut mettre la balise composition en première. Aussi, le couple (orchestra, conductor) est optionnel mais les 2 doivent exister ensemble.
Ainsi on rajoute une balise <conductor></conductor> pour satisfaire cette contrainte.
De plus on a 2 publisher, on en supprime un en commentant la balise.

Voir le fichier cd2.xml

3)

```
┌──(root㉿cb89a520012b)-[/nosql]
└─# xmllint --schema note.xsd note-1.xml --noout
note-1.xml validates

┌──(root㉿cb89a520012b)-[/nosql]
└─#
```

Le fichier note-1.xml est validé contre le schéma note.xsd

4)

```
┌──(root㉿cb89a520012b)-[/nosql]
└─# xmllint --schema note.xsd note-2.xml --noout
note-2.xml:4: element from: Schemas validity error : Element 'from': This element is not expected. Expected is ( to ).
note-2.xml:13: element body: Schemas validity error : Element 'body': This element is not expected.
note-2.xml:15: element notes: Schemas validity error : Element 'notes': This element is not expected. Expected is ( note ).
note-2.xml fails to validate
```

Les balises « from » et « to » sont inversé dans la première note, on les remet dans le bon ordre.

On a un élément <body> en trop, on le commente.

Pour finir, une balise est nommée <notes> au lieu de <note>. On la renomme correctement.

Voir le fichier note-2.xml modifié

Exercice 3

```
<!ELEMENT famille (nom, personne+)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT personne (prenom, age, (poids-kg|poids-lb), taille?)>
<!ATTLIST personne pnumber ID #REQUIRED
          mere IDREF #IMPLIED
          pere IDREF #IMPLIED>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT age (#PCDATA)>
<!ELEMENT poids-kg (#PCDATA)>
<!ELEMENT poids-lb (#PCDATA)>
<!ELEMENT taille (#PCDATA)>
```

Explication des lignes

1. Une famille a effectivement un nom et une personne
2. le nom est une chaine de charactère
3. l'élement personne est constitué d'un prenom (chaine de char), d'un age, obligatoirement un poids soit en kg soit en lb, et possiblement une taille (?)

Voir le fichier famille.dtd

TD2:

Exercice 1:

Commande: xmllint --xpath "Q" contacts.xml

1. /Contacts -> On s'attend à obtenir le nœud contact ainsi que tous ses descendant.
2. /Contacts/Person -> On va obtenir les 2 nœuds Person, fils de /Contacts, et leur descendants.
3. //Person[Firstname="John"] -> Cherche toute les Person dont le fils Firstname a pour valeur « John ».
   Ici la 1ère personne.
4. //Person[Email] -> retourne toutes les nœuds Person qui ont un champ email.
   Ici les 2 nœuds Person.
5. /Contacts/Person[1]/Firstname/child::text()-> retourne la Valeur des Text, fils de Firstname, du 1er nœud Person, fils du nœud Contacts. Concrètement, cela retourne le prénom de la 1ère personne (ici « John »).
6. /Contacts/Person[1]/Firstname/text() -> Il n'y a pas de différence dans le résultat dans notre cas. text() donne tout les Text descendant de Firstname, pas seulement les Text, fils de Firstname.
7. /Contacts//Address[@type="home"]//Street/child::text()-> On regarde les nœuds « Address », descendant des Contacts pour lequels l'attribut « type » a pour valeur « home ». On récupère les nœuds Street des ces Address et retourne leurs fils Text.
   Ainsi on retourne le nom de la rue des adresses de domicile des contacts.
   Dans notre cas, cela retourne « 23 Main St », l'adresse du domicile de John.
8. /Contacts//Address[@type="home" and City="London"]
   On descend dans le noeud contact jusqu'a obtenir les noeud Address avec comme attribut type "home" et ayant comme noeud city "London" , il n'y en a qu'un donc il retourne tout le noeud address

9. /Contacts//Address[@type="work" and City="Dublin"]/parent::node()/Lastname/text()

   On descend dans le noeud contact jusqu'a obtenir les noeud Address avec comme attribut type "work" et ayant comme noeud city "Dublin", on remonte

dans son parent donc le noeud Person et on prend tout les noeuds en dessous, on selectione le noeud Lastname et son texte.

10. /Contacts//Address[@type="work" and City="Dublin"]/../Lastname/text()
    On descend dans le noeud contact jusqu'a obtenir les noeud Address avec comme attribut type "work" et ayant comme noeud city "Dublin", on remonte dans son parent donc le noeud Person on obtient le texte du noeud Lastname. La difference ici est que on prend uniquement le parent Person et pas tout les noeuds et sous noeuds associés à cause de parent::node()

11. /Contacts[.//Address[@type="work" and City="Dublin"]]//Lastname/text()
    On choisit les noeuds contacts qui ont un noeud address avec comme attribut type work and ville dublin, or le seul noeud contact satisfait cette condition donc c'est equivalent à /Concacts/Lastname/text() et comme il y a 2 noeud Lastname dans contact avec valeur Smith et Dune, ça retourne Smith et Dune, ce n'est pas la meme chose que la requete precedente car on prend le lastname des deux noeuds Address.

    C'est équivalent a dire qu'on affiche le texte des nœuds Lastname, si le nœud contact a une adresse de type 'work' et une ville Dublin

12. /Contacts//Address[@type="work"]/ancestor::node()

On descend dans le noeud contact et on selection les noeuds Address avec comme attribut type work (Ce qui nous retourne un seul noeud adresse). On prend les ancetre de ce noeud c'est à dire Person, ensuite l'ancetre de Person qui est le noeud Contact et finalement le noeud de Contact, la racine document. On a donc bien 3 noeud retournés

13. /Contacts/Person[Lastname="Smith"]/following-sibling::node()/Lastname/text()
    On retourne le nom de famille de toutes les Person suivant une personne dont le nom de famille est « Smith ».
    Ici John Smith a le nom de famille « Smith » donc on retourne le nom de

famille de toutes les Person après. Il n'y a que Tom Dunne après John Smith donc on retourne « Dunne ».

14. /Contacts/Person[following-sibling::node()/Lastname="Dunne"]/Lastname/text()
    On retourne les nom de famille (noeud Text fils de Lastname) des Person dont, parmi les prochains voisins (tous nœuds compris), il y a une personne dont le nom de famille est « Dunne ».
    Ici Tom Dunne a le nom de famille Dunne. Donc on va retourner le nom de famille de toute les Person avant lui. Ici on retourne seulement le nom de famille de John Smith, soit « Smith ».

Exercice 2 :

1. Toutes les compositions.
   //composition
2. Toutes les compositions ayant un soloist
   //performance[soloist]/composition
   -> si il n'y a pas de soloist, la performance ne sera pas ajoutée au contexte.
3. Toutes les performances avec un seul orchestra mais pas de soloist.
   //performance[orchestra and not(soloist)]
4. Tous les soloists ayant jou´e avec le London Symphony Orchestra sur un CD publié par Deutsche Grammophon.
   //CD[publisher = "Deutsche Grammophon"]/performance[orchestra = "London Symphony Orchestra"]/soloist/child::text()
5. Tous les CDs comportant des performances du London Symphony Orchestra.
   //CD[performance[orchestra = "London Symphony Orchestra"]]

Exercice 3 :

1. Le titre du cinquième livre dans la liste
   //award[5]/title/child::text()
2. L'auteur du sixième livre dans la liste
   //award[6]/author/child::text()

3. Le titre du livre qui a gagné en 2000
   //award[year = "2000"]/title/child::text()
4. Le nom de l'auteur du livre intitulé Possession.
   //award[title = "Possession"]/author/child::text()
5. Le titre des livres dont J M Coetzee est l'auteur.
   //award[author = "J M Coetzee"]/title/child::text()
6. Le nom de tous les auteurs qui ont obtenu un prix depuis 1995
   //award[year[.>=1995]]/author/child ::text()
7. Le nombre total de prix décernés
   count(//award)

Exercice 4 :

Recettes1

1. Les éléments titres des recettes.
   //titre
2. Les noms des ingrédients.
   //nom_ing/child::text()
3. L'élément titre de la deuxième recette.
   //recette[2]/titre
4. La dernière étape de chaque recette.
   //recette//etape[last()]/child::text()
5. Le nombre de recettes.
   count(//recette)
6. Les éléments recette qui ont strictement moins de 7 ingrédients
   //recette[count(.//ingredient)<7]
7. Les titres de recette qui ont strictement moins de 7 ingrédients.
   //recette[count(.//ingredient)<7]/titre/child::text()
8. Les recettes qui utilisent de la farine.
   //recette[.//nom_ing = "farine"]/titre/child::text()
9. Les recettes de la catégorie entrée.
   //recette[categorie[child::text() = "Entrée"]]/titre/child::text()

Recette2

1. //titre

2. //ingredient/child::text()
3. //recette[2]/titre
4. //recette//etape[last()]/child::text()
5. count(//recette)
6. //recette[count(.//ing-recette) < 7]
7. //recette[count(.//ing-recette) < 7]/titre/child ::text()
8. //ing-recette[@ingredient = "farine"]/../../titre/child::text()
9. //recette[contains(@categ, "entree")]/titre/child ::text()

Exercice 5 :

1. Le nombre de morceaux (tracks hors PlayLists) de la bibliothèque
count(/plist/dict/dict/dict)

Les musiques sont des balises dict dans plist/dict/dict. On récupère toute ces balises et on les compte.

2. Tous les noms d'albums.
/plist/dict/dict/dict/key[child::text() = "Album"]/following-sibling::node()[1]/child::text()

On trouve toute les musiques (/plist/dict/dict/dict) comme précédemment. On récupère la balise key dont le contenu est « Album ».
Cela nous donne beaucoup de doublons mais on peut les éliminer en utilisant cette commande :
xmllint --xpath '/plist/dict/dict/dict/key[child::text() = "Album"]/following-sibling::node()[1]/child::text()' iTunes-Music-Library.xml | sort | uniq

3. Tous les genres de musique (Jazz, Rock, . . .)
/plist/dict/dict/dict/key[child::text() = "Genre"]/following-sibling::node()[1]/child::text()

On récupère la valeur du noeud après la <key> "Genre" dans chaque musique.
De même on peut éliminer les doublons:

xmllint --xpath '/plist/dict/dict/dict/key[child::text() = "Genre"]/following-sibling::node()[1]/child::text()' iTunes-Music-Library.xml | sort | uniq

4. Le nombre de morceaux de Jazz.
count(/plist/dict/dict/dict/key[child::text() = "Genre"]/following-sibling::node()[1][contains(child::text(), "Jazz")])'

5. Tous les genres de musique mais en faisant en sorte de n'avoir dans le résultat qu'une seule occurrence de chaque genre.

/plist/dict/dict/dict/key[child::text() = "Genre"]/following-sibling::node()[1][not(child::text() = ../following-sibling::node()/key[child::text() = "Genre"]/following-sibling::node()[1]/child::text())]/child::text()

Après avoir trouvé le nom d'un genre du musique, on cherche dans les musiques suivantes si ce genre existe. Ainsi on sélectionne seulement le dernier élément de chaque genre de musique.

6. Le titre (Name) des morceaux qui ont été écoutés au moins 1 fois.
/plist/dict/dict/dict[key[child::text() = "Play Count"]]/key[child::text() = "Name"]/following-sibling::node()[1]/child::text()

On regarde pour chaque musique si une balise <key>Play Count</key> existe. Si c'est le cas, on retourne le nom du morceau. On fait cela pour chaque morceau.

7. Le titre des morceaux qui n'ont jamais été écoutés
/plist/dict/dict/dict[not(key[child::text() = "Play Count"])]/key[child::text() = "Name"]/following-sibling::node()[1]/child::text()

On fait l'inverse de la question précédente. On retourne la musique si la balise <key>Play Count</key> n'est pas présente dans la musique.

8) Le titre du (ou des) morceaux les plus anciens (renseignement Year) de la bibliothèque.
/plist/dict/dict/dict[key[child::text() = "Year"]/following-

sibling::node()[1]/child::text()[not(. > /plist/dict/dict/dict/key[child::text() = "Year"]/following-sibling::node()[1]/child::text())]]/key[child::text() = "Name"]/following-sibling::node()[1]/child::text()

On regarde pour chaque morceau s'il n'existe pas de morceau avec une année strictement supérieure à la sienne. Ainsi cela permet de déterminer ceux avec l'année la plus faible. On retourne le titre de ces musiques.