Données du Web : XQuery - Partie 2

TP en binômes à rendre le 11 octobre sur Moodle.
Déposer un seul document contenant aussi les solutions à la partie XPath

0) Warm-up BaseX

On utilisera BaseX pour interroger des documents XML avec XQuery.

1) XQuery : Tweets

Reprenez votre DTD pour les Tweets, et créez un document XML valide contenant au moins 3 utilisateurs et 5 tweets. Attention : pour le bon déroulement de l'exercice, il sera peut être nécessaire d'apporter des légères modifications à votre DTD afin de permettre les interrogations !

Donner les requêtes XQuery correspondants aux expressions suivantes et évaluer ces expressions dans le document XML pour les Tweets.

1. Créez une liste de paires tweet-auteur, avec chaque paire contenue dans un element `result`.
2. Pour chaque utilisateur, listez le nom de l'utilisateur et la date de tous ses tweets, le tout regroupé dans un élément `result`.
3. Listez les utilisateurs qui ont publié un tweet qui a été retwitté au moins deux fois.
4. Pour chaque tweet, listez son contenu et la date de ses deux premières réponses. Rajoutez un element vide `<nonRetwitted/>` s'il n'a pas été retwitté.
5. Listez les utilisateurs de la plateforme en ordre alphabétique.
6. Listez les tweets contenants l'hashtag "#I<3XML".
7. Trouvez le tweet le plus ancien ainsi que le plus recent.
8. Pour chaque tweet ayant des hashtags, retournez le tweet avec la liste de ses hashtag.
9. Pour chaque tweet ayant des références utilisateur, retournez le tweet avec la liste des références utilisateur.
10. Déclarez la fonction `local:retwittePar`, qui, étant donné un tweet, retourne tous les utilisateurs qui l'ont retwitté.

2) XQuery : Trains (suite)

Considérez le document XML suivant, représentant des informations ferroviaires.

```
<gare>
  <train numero="t5560" type="TGV">
    <voiture numero="v1">
      <resa numero="r17" id="u55"/>
      <resa numero="r18" id="u52"/>
    </voiture>
    <voiture numero="v2"/>
    <voiture numero="v3"/>
    <voiture numero="v4">
      <bar service="froid uniquement"/>
    </voiture>
  </train>
  <train numero="t6731">
    <voiture numero="v1"/> 2
```

```
    <voiture numero="v2">
      <resa numero="r15" id="u55"/>
    </voiture>
  </train>
  <usager id="u55" nom="Jean" prenom="Dufour"/>
  <usager id="u52" nom="Brigitte" prenom="Lefebvre"/>
  <usager id="u56" nom="Patrick" prenom="Subiran"/>
</gare>
```

Donner des expressions XQuery pour les requêtes suivantes.

1. Le numéro des trains possédant une voiture-bar.
2. Le nom des usages ayant au moins une réservation.
3. La reservation avec le plus grand identifiant (dans l'ordre lexicographique).
4. Le numéro des trains dont au moins 2 places sont réservées.
5. Le nom des personnes ayant réservé exactement deux fois.
6. Les usagers n'ayant effectué aucune réservation.

3) Propriétés des requêtes XPath et XQuery

1. Reformuler les requêtes suivantes en utilisant exclusivement les axes `child`, `descendant`, `descendant-or-self`, `following` et `following-sibling`
   — `//b[parent::a]`
   — `//a/preceding-sibling::c`
   — `//c[preceding::d]`
   — `//b/a/preceding-sibling::c/preceding::d`
   — `/a/b/..//*/./../preceding::d`
   — `//a/ancestor::b/parent::c/child::d/parent::e`
2. Reformuler les requêtes `//a/following::b` et `//a/preceding::b` en utilisant les axes `descendant-or-self`, `ancestor`, `following-sibling` et `preceding-sibling`.
3. Pour chaque requête définie aux points 1 et 2, proposer un document XML pour lequel la réponse à la requête n'est pas vide, sinon expliquer pourquoi un tel document n'existe pas.
4. Pour chaque requête définie aux points 1 et 2, proposer un document XML *valide par rapport à la DTD suivante* pour lequel la réponse à la requête n'est pas vide.
   `<!DOCTYPE a [<!ELEMENT a (c)> <!ELEMENT c ANY>]>`
5. Soient X, Y, Z des séquence d'éléments XML. Est il vrai que, dans le cadre du langage XPath, si $X = Y$ et $Y = Z$ alors $X = Z$? Est-ce le cas pour XQuery ?
6. Donner un document XML pour lequel la requête `//r[a[1] = a[2]]` n'est pas vide.