



HMIN103

Données du Web

Rendu TP/TD5 - XPath/Xquery

Auteur :

Gracia-Moulis Kévin (21604392)
Canta Thomas (21607288)

Master 1 - AIGLE/DECOL
Faculté des sciences de Montpellier
Année universitaire 2020/2021



Table des matières

1. XQuery : Tweets	2
Question 1	2
Question 2	2
Question 3	2
Question 4	2
Question 5	2
Question 6	2
Question 7	3
Question 8	3
Question 9	3
Question 10	3
Question 11	3
Question 12	3
2. Génération de Pages HTML via XQuery	5
3. Propriétés des requêtes XPath	6
Question 1	6
i	6
ii	6
iii	6
iv	6
v	6
vi	7
Question 2	7
i	7
ii	7
4. L'égalité dans XQuery	8
Question 1	8
Question 2	8

1. XQuery : Tweets

Les documents associés à cette partie se trouvent dans le dossier "tweet". Le fichier tweet.dtd est la DTD de notre base XML. Les requêtes XML ci-dessous sont à utiliser avec le fichier tweet.xml.

Question 1

```
let $tweets := //tweets/tweet return count($tweets)
```

Question 2

```
//hashtag
```

Question 3

```
for $user in //profile
  for $tweet in //tweet where $tweet/@idUser = $user/@idUser
    return <result> {$user,$tweet} </result>
```

Question 4

```
for $user in //profile
  let $res :=
    for $tweet in //tweet
      where $tweet/@idUser = $user/@idUser
      return $tweet/date
  return <result>{$user/pseudo,$res}</result>
```

Question 5

```
let $id := //profile/@idUser/string()
for $x in //tweet[@idUser=$id and nbrRt/text() > 0]
  return $x
```

Question 6

```
for $x in //tweet/responses
  return
    if ( count($x//response) = 0 )
    then <responses><nonRetwitted/></responses>
    else
      if ( count($x//response) = 1 )
      then $x//response
      else ($x//response[position() = last()-1], $x//response[last()])
```

Question 7

```
for $user in //profile/pseudo/text() order by $user
  return $user
```

Question 8

```
for $x in //tweet
  where exists($x//corps/hashtag[contains(., '#I&lt;3XML')])
  return $x
```

Question 9

```
for $x in //tweets order by $x//tweet/date/timestamp
  return ($x[position() = 1], $x[last()])
```

Question 10

```
for $user in //profile
  let $hashtag := for $tweet in //tweet
    where $tweet/@idUser = $user/@idUser
    return $tweet//hashtag
  return ($user/@idUser, $hashtag)
```

Question 11

```
for $tweet in //tweet
  where exists($tweet//corps/ref-user)
  let $tweetref := for $x in $tweet
    return $x//corps/ref-user
  return ($tweet, $tweetref)
```

Question 12

```
declare function local:aReponduAuTweet($ptweet as element(tweet)) {  
    if ( count($ptweet//response) = 0 )  
    then  
        <noresponse/>  
    else (  
        let $users := $ptweet//response/@idUser/string()  
        return distinct-values($users)  
    )  
};
```

```
(: exemple d'exécution :)  
local:aReponduAuTweet(//tweet[@id="tweet4"])
```

2. Génération de Pages HTML via XQuery

Les documents associés à cette partie se trouve dans le dossier "tam". La requête permettant d'afficher le code HTML se trouve dans le fichier *req.xqy*. *file.html* est le résultat final obtenus.

NB : Le résultat affiché n'est que le body car nous avons rajouter dans notre fichier html une balise de style pour rendre l'affichage final plus lisible.

3. Propriétés des requêtes XPath

Question 1

i

(: La requête :)

```
//d/preceding-sibling::c
```

(: Peut être réécrit de la façon suivante :)

```
//c[following-sibling::d]
```

ii

(: La requête :)

```
//c/a/preceding-sibling::a/preceding::e
```

(: Peut être réécrit de la façon suivante :)

```
descendant-or-self::*[following::c/a/following-sibling::a]/e
```

iii

(: La requête :)

```
//d[parent::b/c]
```

(: Peut être réécrit de la façon suivante :)

```
descendant-or-self::*/b[child::c]/d
```

iv

(: La requête :)

```
/r/b/.../*/. .../preceding::d
```

(: Peut être réécrit de la façon suivante :)

```
descendant-or-self::*[/r/b]/d[following::*]
```

v

(: La requête :)

//a/ancestor::c/child::d/parent::e

(: Peut être réécrit de la façon suivante :)

descendant-or-self::*/*e[child::d = /descendant-or-self::*/*c[descendant::a]/d]

vi

(: La requête :)

//c[preceding::d]

(: Peut être réécrit de la façon suivante :)

descendant-or-self::*/*d/following::c

Question 2

i

(: La requête :)

//a/following::b

(: Peut être réécrit de la façon suivante :)

//a/ancestor::*/*/*[preceding-sibling::a]//b

ii

(: La requête :)

//a/preceding::b

(: Peut être réécrit de la façon suivante :)

//a/ancestor::*/*/*[following-sibling::a]//b

4. L'égalité dans XQuery

Question 1

Nous voulons vérifier le cas de la transitivité (*i*) dans XPath et XQuery :

$$(X = Y \text{ et } Y = Z) \Rightarrow (X = Z) \text{ (i)}$$

Dans **XPath** et **XQuery** la transitivité n'est pas possible car l'égalité entre deux nœuds renvoie vrai si, sous forme de chaînes de caractères, ils contiennent un élément en commun.

 exemple :

$X = "a"$

$Y = "ab"$

$Z = "b"$

Dans l'exemple ci-dessus on a $X = Y$ et $Y = Z$ mais $X \neq Z$. Donc la transitivité n'est pas vérifiée dans tout les cas.

Question 2

```
declare function local:isEgal($a,$b)
{
  let $res :=
    (: si séquences vides :)
    if (empty($a) and empty($b)) then true()

  else
    (: si l'une des séquences est vide :)
    if (empty($a) or empty($b)) then false()

  else (
    (: tête de la séquence de a et b :)
    let $x := $a/*[position() = 1]
    let $y := $b/*[position() = 1]

    (: queue de la séquence de a et b :)
    let $q := $a/*[position() > 1]
    let $r := $b/*[position() > 1]
```

```
        (: on test l'égalité entre la tête et la queue de a et b:)
        return $x = $y and local:isEgal($q,$r)
    )
    return $res
};
```