Master 2 Humanités numériques – Bases de données semi-structurées

Examen sur machine – 20/01/2025

J. Darmont – https://eric.univ-lyon2.fr/jdarmont/

Durée : 2 heures – Documents autorisés – Barème fourni à titre indicatif

Rendu : Fichiers DTD (.dtd) et XQuery (.xq) sur Moodle.
https://moodle.univ-lyon3.fr/course/view.php?id=20834

Soient les trois documents XML locations.xml (adresses), countries.xml (pays) et regions.xml (continents) annexés dans Moodle.

Exercice 1 : Schéma XML (10 points)

Définir pour chacun des trois documents XML une DTD permettant de valider les documents avec https://www.xmlvalidation.com.

Exercice 2 : XQuery (10 points)

Formuler les requêtes suivantes, en privilégiant la lisibilité du code XQuery. Tout résultat de requête doit être un fragment de code XML bien formé.

1. Nom (*name*) des pays (*country*) triés par ordre alphabétique.

2. Adresses (*location*) complètes du Royaume-Uni (*UK*).

3. Adresses qui n'ont pas de code postal (*postal_code*).

4. Noms des villes (*city*) d'Italie (*Italy*).

5. Couples [continents (*region*), pays] triés par ordre alphabétique de continent et de pays.

6. Couples [continents, villes] triés par ordre alphabétique de continent et de villes.

RSVP

7. Nombre d’adresses, de pays et de continents.

8. Nombre d’adresses par identifiant de pays (*country_id*).

9. Nombre d’adresses par nom de pays, triées par ordre inverse de nombre d’adresses et par nom de pays.

10. Noms des villes par continents, triées par continents.

Correction Exercice 1

```
<!-- locations -->
<!ELEMENT locations (location)+>
  <!ELEMENT location (street_address, postal_code?, city, state_province?)>
    <!ATTLIST location id ID #REQUIRED country_id IDREF #REQUIRED>
    <!ELEMENT street_address (#PCDATA)>
    <!ELEMENT postal_code (#PCDATA)>
    <!ELEMENT city (#PCDATA)>
    <!ELEMENT state_province (#PCDATA)>

<!-- countries -->
<!ELEMENT countries (country)+>
  <!ELEMENT country (name)>
    <!ATTLIST country id ID #REQUIRED region_id IDREF #REQUIRED>
    <!ELEMENT name (#PCDATA)>

<!-- regions -->
<!ELEMENT regions (region)+>
  <!ELEMENT region (name)>
    <!ATTLIST region id ID #REQUIRED>
    <!ELEMENT name (#PCDATA)>
```

Correction Exercice 2

```
(: 1 :)
for $n in //country/name
order by $n
return $n

(: 2 :)
for $l in //location
where $l/@country_id = "UK"
return $l

(: 3 :)
for $l in //location
where not(exists($l/postal_code))
return $l

(: 4 :)
for  $c in //country,
     $l in //location
where $c/@id = $l/@country_id
and $c/name = "Italy"
return   <res>
             {$l/city}
         </res>
```

```
(: 5 :)
for $r in //region,
    $c in //country
where $r/@id = $c/@region_id
order by $r/name, $c/name
return   <res>
             {$r/name}
             {$c/name}
         </res>

(: 6 :)
for $r in //region,
    $c in //country,
    $l in //location
where $r/@id = $c/@region_id
and $c/@id = $l/@country_id
order by $r/name, $l/city
return   <res>
             {$r/name}
             {$l/city}
         </res>

(: 7 :)
let $l := count(//location),
    $c := count(//country),
    $r := count(//region)
return   <res>
             <numloc>{$l}</numloc>
             <numcou>{$c}</numcou>
             <numreg>{$r}</numreg>
         </res>

(: 8 :)
for $l in //location
group by $c := $l/@country_id
let $n := count($l)
return <res country_id="{$c}" number="{$n}" />

(: 9 :)
for $c in //country,
    $l in //location
where $c/@id = $l/@country_id
group by $name := $c/name
let $n := count($l)
order by $n descending, $name
return <res name="{$name}" number="{$n}" />
```

```
(: 10 :)
for $r in //region,
    $c in //country,
    $l in //location
where $r/@id = $c/@region_id
and $c/@id = $l/@country_id
group by $name := $r/name
order by $name
return   <res region="{$name}">
             {$l/city}
         </res>
```