HMIN103
Données du Web

Rendu TP NOTE - XML Relationnel

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. Stockage Monet

Question 1

```
0  CREATE TABLE batiment (
1      node int PRIMARY KEY,
2      txtval varchar(255),
3      numval int
4  )
5
6  CREATE TABLE batiment_etage (
7      node int PRIMARY KEY,
8      txtval varchar(255),
9      numval int
10 )
11
12 CREATE TABLE batiment_etage_bureau (
13     node int PRIMARY KEY,
14     txtval varchar(255),
15     numval int
16 )
17
18 CREATE TABLE batiment_etage_bureau_code (
19     node int PRIMARY KEY,
20     txtval varchar(255),
21     numval int
22 )
23
24 CREATE TABLE batiment_etage_description (
25     node int PRIMARY KEY,
26     txtval varchar(255),
27     numval int
28 )
29
30 CREATE TABLE batiment_etage_personne (
31     node int PRIMARY KEY,
32     txtval varchar(255),
33     numval int
34 )
35
36 CREATE TABLE batiment_etage_salle (
37     node int PRIMARY KEY,
38     txtval varchar(255),
39     numval int
40 )
41
42 CREATE TABLE batiment_etage_salle_nbplace (
43     node int PRIMARY KEY,
44     txtval varchar(255),
45     numval int
46 )
```

Question 2

Requête a

Prenons la requête XPath suivante :

```
/batiment//salle
```

Nous pouvons l'écrire comme suit en SQL :

```
SELECT node, txtval, numval FROM batiment_etage_salle;
```

Requête b

Prenons la requête XPath suivante :

```
/batiment/etage/salle/text()
```

Nous pouvons l'écrire comme suit en SQL :

```
SELECT txtval FROM batiment_etage_salle
```

Requête c

Prenons la requête XPath suivante :

```
/batiment/etage/salle[nbplace > 10]
```

Nous pouvons l'écrire comme suit en SQL :

```
SELECT node, numval FROM batiment_etage_salle_nbplace WHERE numval > 10
```

Question 3

On pourrait ajouter une clef étrangère parent qui référence la node parent.

2. Stockage schema-aware

Rappel de la DTD pour la presse vu au TP1 :

```
<!DOCTYPE presse [
    <!ELEMENT presse (journal,journalistes)>
    <!ELEMENT journal (nom, directeur, article*)>
    <!ELEMENT article (corps)>
    <!ATTLIST article titre CDATA #IMPLIED>
    <!ATTLIST article auteur IDREF #REQUIRED>
    <!ELEMENT corps (#PCDATA)>
    <!ELEMENT journalistes (journaliste+)>
    <!ATTLIST journaliste idJ ID #REQUIRED>
    <!ELEMENT journaliste ((nom,prenom)|pseudonyme)>
    <!ATTLIST journaliste anonymisation (oui|non) 'non'>
    <!ELEMENT pseudonyme (#PCDATA)>
    <!ELEMENT nom (#PCDATA)>
    <!ELEMENT prenom (#PCDATA)>
    <!ELEMENT directeur (nom,prenom)>
]>
```

Question 1

```
presse(presseID)
journal(presseID, journalID, nom : string)
directeur(journalID, directeurID, nom : string, prenom : string)
article(journalID, articleID, auteurIDREF, titre : string, corps : string)
journalistes(presseID, journalistesID)
journaliste(journalistesID, journalisteID, pseudo : string, nom : string,
    prenom : string, anonymisation : string)
```

Question 2

```
INSERT INTO presse (presseID) VALUES (1)
INSERT INTO journal (presseID, journalID, nom) VALUES (1, 1, "Le petit
    Thominou")
INSERT INTO directeur (journalID, directeurID, nom, prenom) VALUES (1, 1,
    "Boisunptivère", "Gérard")
INSERT INTO article (journalID, articleID, auteurIDREF, titre, auteur,
    corps) VALUES (1, 1, 1, "NEVER GONNA GIVE YOU UP", "Biographie de Rick
    Astley")
INSERT INTO journalistes(presseID, journalistesID) VALUES (1, 1)
INSERT INTO journaliste(journalistesID, journalisteID, pseudo, nom, prenom,
    anonymisation) VALUES (1, 1, "RA", "Astley", "Rick", "a007")
INSERT INTO journaliste(journalistesID, journalisteID, pseudo, nom, prenom,
    anonymisation) VALUES (1, 2, "Maire du Levallois-Perret", "Balkany",
    "Patrick", "adelargent")
INSERT INTO journaliste (journalistesID, journalisteID, pseudo, nom,
    prenom, anonymisation) VALUES (1,3,"Patoche", "Sébastien", "Patrick",
    "a1018");
```

Question 3

Requête a

Prenons la requête XPath suivante :

```
/presse/journal/directeur[prenom = "Gérard"]
```

Nous pouvons l'écrire comme suit en SQL :

```
SELECT * FROM directeur WHERE prenom="Gérard";
```

Requête b

Prenons la requête XPath suivante :

```
//journaliste[anonymisation = "adelargent"]
```

Nous pouvons l'écrire comme suit en SQL :

```
SELECT * FROM journaliste WHERE anonymisation = "adelargent";
```

Requête c

Prenons la requête XPath suivante :

```
//presse[@presseID = 1]//journaliste
```

Nous pouvons l'écrire comme suit en SQL :

```
SELECT j.journalisteID, j.journalistesID, j.pseudo, j.nom, j.prenom,
   j.anonymisation
FROM journaliste AS j, journalistes
WHERE j.journalistesID = journalistesID AND journalID = 1;
```

3. Interval-encoding avec SAX

Question 1

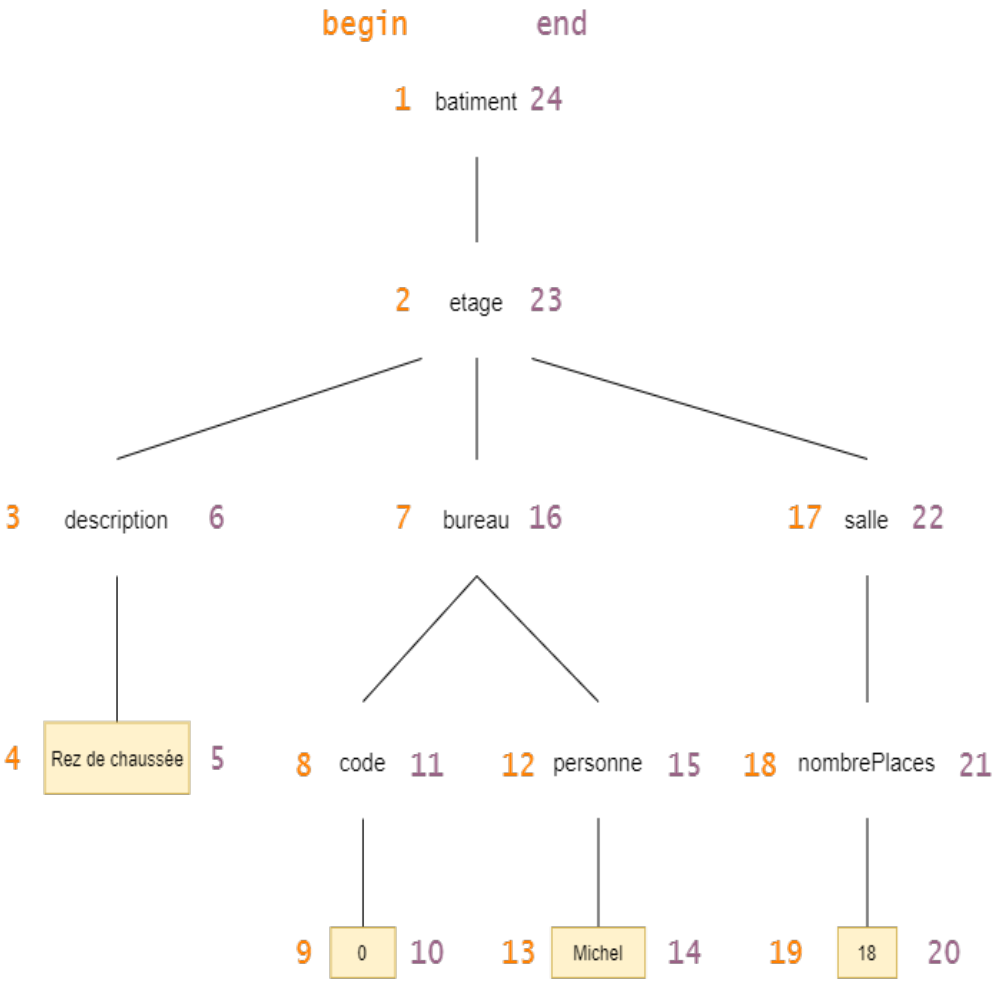

FIGURE 1 – Encodage begin/end sur le document XML des bâtiments

```
CREATE TABLE node(
    begin INT,
    end INT,
    parent INT,
    tag VARCHAR(30),
    type VARCHAR(10),
    CONSTRAINT FK_PARENT FOREIGN KEY (parent) REFERENCES node(begin)
);
```

Listing 1 – Encodage begin/end enregistrez dans une table NODE

Pour chaque extrait de code de SaxParser qui suit, vous retrouverez chaque fichier dans le dossier saxParsers et en annexe de ce rapport la méthode de compilation et d'exécution pour chacuns d'eux (voir annexe).

Question 2

```
0  ...
1  public void startElement(String namespaceURI, String localName, String
       qName, Attributes atts) throws SAXException {
2      cptrBegin.add(cptr);
3      cptr++;
4  }
5
6  public void endElement(String namespaceURI, String localName, String qName)
       throws SAXException {
7      int endNum = cptr;
8      cptr++;
9      int beginNum = cptrBegin.get(cptrBegin.size()-1);
10     cptrBegin.remove(cptrBegin.size()-1);
11     int parentNum = 0;
12     if ( cptrBegin.size()-1 < 0 ) {
13         parentNum = 0;
14     }
15     else {
16         parentNum = cptrBegin.get(cptrBegin.size()-1);
17     }
18
19     System.out.println("INSERT INTO node(begin,end,parent,tag,type)
           VALUES("+beginNum+ ","+endNum+ ","+parentNum
           +","+"\""+localName+"\""+","+"\"elt\");");
20 }
21
22 public void characters(char[] ch, int start, int length) throws
       SAXException {
23     String str = new String(ch, start, length);
24     if( !str.trim().isEmpty() ) {
25
26         System.out.print("INSERT INTO node(begin,end,parent,tag,type)
               VALUES("+this.cptr+" ,");
27
28         cptr++;
29         System.out.println(cptr+" ,"+ cptrBegin.get(cptrBegin.size()-1)
               +"," +"\""+str +"\""+ ","+"\"txt\");");
30
31         cptr++;
32     }
33 }
34 ...
```

Listing 2 – Partie du code SaxParserBeginEnd.java

Question 3

```
SELECT * FROM NODE WHERE parent = 29;
SELECT * FROM NODE WHERE type = "txt";
SELECT * FROM NODE WHERE tag = "pseudo";
```

Question 4

Nous avons tester le temps d'exécution avec et sans affichage car nous savons que celui-ci augmente grandement sa durée. Voici les résultats obtenus :

- Avec affichage ⇒ 9034 millisecondes.
- Sans affichage ⇒ 842 millisecondes.

Question 5

```
0  ...
1  public void startElementPost(String namespaceURI, String localName, String
       qName, Attributes atts) throws SAXException {
2      cptrBegin.add(begin);
3      begin++;
4  }
5
6  public void endElementPost(String namespaceURI, String localName, String
       qName) throws SAXException {
7      int endNum = end;
8      end++;
9      int beginNum = cptrBegin.get(cptrBegin.size()-1);
10     cptrBegin.remove(cptrBegin.size()-1);
11     int parentNum = 0;
12     if ( cptrBegin.size()-1 < 0 ) {
13         parentNum = 0;
14     }
15     else {
16         parentNum = cptrBegin.get(cptrBegin.size()-1);
17     }
18
19     System.out.println("INSERT INTO node (begin,end,parent,tag,nodtype)
           VALUES("+beginNum+ ","+endNum+ ","+parentNum
           +","+"\""+localName+"\""+","+"\"elt\");");
20
21 }
22
23 public void charactersPost(char[] ch, int start, int length) throws
       SAXException {
24     String str = new String(ch, start, length);
25     if( !str.trim().isEmpty() ) {
26
27         System.out.print("INSERT INTO node (begin,end,parent,tag,nodtype)
               VALUES("+begin+" ,");
28
29         begin++;
30         System.out.println(end+" ,"+ cptrBegin.get(cptrBegin.size()-1) +","
               +"\""+str +"\""+ ","+"\"txt\");");
31
32         end++;
33     }
34
35 }
36 ...
```

Listing 3 – Partie du code SaxParserPrePost.java

Question 6

```
0  ...
1  public void startElement(String namespaceURI, String localName, String
       qName, Attributes atts) throws SAXException {
2      // si y'a plus de frère on récupère le parent
3      if (frere.empty()) {
4          frere.push(1);
5          frere.push(1);
6      }
7
8      // sinon on continue de descendre
9      else {
10         int f = frere.pop();
11         int p = f;
12         if (!frere.empty()) {
13             p = frere.pop();
14         }
15         parent.push(p);
16         frere.push(p);
17         frere.push(f);
18         frere.push(1);
19     }
20 }
21
22 public void endElement(String namespaceURI, String localName, String qName)
       throws SAXException {
23
24     // on crée une pile temporaire qui permet de "renverser" notre pile parent
25     Stack<Integer> temp = new Stack<Integer>();
26     for(Integer e : parent) {
27         temp.push(e);
28     }
29
30     frere.pop();
31
32     // on commence à crée l'index en dépilant notre pile renversé
33     String indice="";
34     for(Integer e : temp) {
35         indice=indice+String.valueOf(e)+".";
36     }
37
38     // on ajoute le dernier élément
39     int f = frere.pop();
40     indice=indice+String.valueOf(f);
41
42     // on incrémente de 1 la pile des frères
43     frere.push(f+1);
44
45     System.out.println("INSERT INTO node (indice, tag, nodetype)
           VALUES("+indice+", "+"\""+localName+"\""+", elt);");
46
47     if(!parent.empty()) {
48         parent.pop();
49     }
50 }
51
52 public void characters(char[] ch, int start, int length) throws
       SAXException {
53
54     String str = new String(ch, start, length);
```

```
55      if (!str.trim().isEmpty()) {
56
57          // on réutilise le même code que start/endElement
58          if (frere.empty()) {
59              frere.push(1);
60              frere.push(1);
61          }
62
63          else {
64                  int f = frere.pop();
65                  int p = f;
66                  if (!frere.empty()) {
67                      p= frere.pop();
68              }
69                  parent.push(p);
70                  frere.push(p);
71                  frere.push(f);
72                  frere.push(1);
73          }
74
75          Stack<Integer> temp = new Stack<Integer>();
76          for (Integer e : parent) {
77              temp.push(e);
78          }
79
80          frere.pop();
81
82          String indice="";
83          for(Integer e : temp) {
84                  indice=indice+String.valueOf(e)+".";
85          }
86
87          int f = frere.pop();
88          indice=indice+String.valueOf(f);
89          frere.push(f+1);
90
91          System.out.println("INSERT INTO node (indice, tag, nodetype) VALUES
                ("+indice+", NULL, txt);");
92
93          if (!parent.empty()) {
94              parent.pop();
95          }
96      }
97  }
98  ...
```

Listing 4 – Partie du code SaxParserDewey.java

Question 7

On utilise un booléen nommé 'in' afin de savoir si on se trouve dans l'élément 'alpha'.

```
0  ...
1  public void startElement(String namespaceURI, String localName, String
       qName, Attributes atts) throws SAXException {
2      if(qName.equals(alpha)) {
3          in = true;
4      }
5      if(in) {
6          for(int i=0;i<profondeur;i++)
7              System.out.print("\t");
8          profondeur++;
9          System.out.println("<"+qName+">");
10     }
11 }
12
13 public void endElement(String namespaceURI, String localName, String qName)
       throws SAXException {
14     if(qName.equals(alpha)) {
15         in = false;
16         profondeur--;
17         for(int i=0;i<profondeur;i++)
18             System.out.print("\t");
19         System.out.println("</"+qName+">");
20     }
21     if(in) {
22         profondeur--;
23         for(int i=0;i<profondeur;i++)
24             System.out.print("\t");
25         System.out.println("</"+qName+">");
26     }
27 }
28
29 public void characters(char[] ch, int start, int length) throws
       SAXException {
30     if(in) {
31         for(int i=0;i<profondeur;i++)
32             System.out.print("\t");
33         String str = new String(ch, start, length);
34         System.out.println(str);
35     }
36 }
37 ...
```

Listing 5 – Partie du code SaxParserXPath.java

Question 8

Pour étendre ce programme afin qu'il supporte n'importe quelle combinaison des axes child et descendant, on pourrait parser la requête XPath passée en paramètre afin de descendre jusqu'à la node voulu avant d'afficher les balises. Nous pourrions même durant le parsing vérifier si il existe des conditions si l'on voit apparaître des crochets *"[ ]"*

Annexe

Compilation et exécution des parsers

Dans le dossier saxParsers, utiliser les commandes suivantes :

SaxParserBeginEnd.java

```
javac SaxParserBeginEnd.java # Compilation
java SaxParserBeginEnd xmls/tweet.xml # Exécution
```

SaxParserPrePost.java

```
javac SaxParserPrePost.java # Compilation
java SaxParserPrePost xmls/tweet.xml # Exécution
```

SaxParserDewey.java

```
javac SaxParserDewey.java # Compilation
java SaxParserDewey xmls/tweet.xml # Exécution
```

SaxParserXPath.java

```
javac SaxParserXPath.java # Compilation
java SaxParserXPath xmls/tweet.xml <element> # Exécution

# <element> : un élement de la structure xml
# exemple d'utilisation : java SaxParserXPath xmls/tweet.xml profiles
```

Pour supprimer TOUS les fichiers crées après la compilation on peut utiliser la commande suivante pour les supprimer, toujours dans le dossier saxParsers :

```
rm SaxParser*.class
```