Donnes du Web - TD3 - XML-Relationnel

Basil Dali - Yannis Naidja

Octobre 2019

Chapitre 1

Stockage schema-unaware : Verical-Edge vs Monet

1.1 Considrons le document XML pour la presse que vous avez propos en rponse la question1du TD1

1.2 laide du langage SQL, implmenter les schmas de stockage Vertical-Edge et Monet associs au docu-ment. Ensuite, peupler les tables correspondantes

Voir document XML presse en annexe.

1.2.1 Vertical-Edge

```
CREATE TABLE PRESSE (
        source int,
        target int NOT NULL PRIMARY KEY,
        ordinal int,
        txtval  varchar(30),
        numval int
);

CREATE TABLE JOURNAL (
        source int,
        target int NOT NULL PRIMARY KEY,
        ordinal int,
```

```
        txtval  varchar(30),
        numval int
);

CREATE TABLE ARTICLE (
        source int,
        target int NOT NULL PRIMARY KEY,
        ordinal int,
        txtval  varchar(30),
        numval int
);
CREATE TABLE TITRE (
        source int,
        target int NOT NULL PRIMARY KEY,
        ordinal int,
        txtval  varchar(30),
        numval int
);

CREATE TABLE AUTEUR (
        source int,
        target int NOT NULL PRIMARY KEY,
        ordinal int,
        txtval  varchar(30),
        numval int
);

CREATE TABLE CORPS (
        source int,
        target int NOT NULL PRIMARY KEY,
        ordinal int,
        txtval  varchar(30),
        numval int
);
CREATE TABLE JOURNALISTES (
        source int,
        target int NOT NULL PRIMARY KEY,
        ordinal int,
        txtval  varchar(30),
        numval int
);
CREATE TABLE JOURNALISTE (
        source int,
        target int NOT NULL PRIMARY KEY,
        ordinal int,
        txtval  varchar(30),
```

```
        numval int
);
CREATE TABLE JOURNALISTEID (
        source int,
        target int NOT NULL PRIMARY KEY,
        ordinal int,
        txtval  varchar(30),
        numval int
);

CREATE TABLE ANONYMOUS (
        source int,
        target int NOT NULL PRIMARY KEY,
        ordinal int,
        txtval  varchar(30),
        numval int
);
CREATE TABLE PSEUDO (
        source int,
        target int NOT NULL PRIMARY KEY,
        ordinal int,
        txtval  varchar(30),
        numval int
);

CREATE TABLE NOM (
        source int,
        target int NOT NULL PRIMARY KEY,
        ordinal int,
        txtval  varchar(30),
        numval int
);

CREATE TABLE PRENOM (
        source int,
        target int NOT NULL PRIMARY KEY,
        ordinal int,
        txtval  varchar(30),
        numval int
);

CREATE TABLE DIRECTEUR (
        source int,
        target int NOT NULL PRIMARY KEY,
        ordinal int,
        txtval  varchar(30),
```

```
105         numval int
106 );
107
108 INSERT INTO PRESSE (target)                              VALUES
    ↪  (0);
109 INSERT INTO JOURNAL (source, target, ordinal)           VALUES (0,
    ↪  1, 1);
110 INSERT INTO NOM (source, target, ordinal, txtval)       VALUES (1,
    ↪  2, 1, 'CNEWS');
111 INSERT INTO DIRECTEUR (source, target, ordinal)         VALUES (1,
    ↪  3, 1);
112 INSERT INTO NOM (source, target, ordinal, txtval)       VALUES (3,
    ↪  4, 1, 'Pepega');
113 INSERT INTO PRENOM (source, target, ordinal, txtval)    VALUES (3,
    ↪  5, 2, 'Kekw');
114 INSERT INTO ARTICLE (source, target, ordinal)           VALUES (1,
    ↪  7, 1);
115 INSERT INTO CORPS (source, target, ordinal, txtval)     VALUES (7,
    ↪  10, 1, 'Des fake news');
116 INSERT INTO TITRE (source, target, txtval)             VALUES (7,
    ↪  8, 'fake');
117 INSERT INTO AUTEUR(source, target, txtval)             VALUES (7,
    ↪  9, 'j1');
118 INSERT INTO ARTICLE (source, target, ordinal)          VALUES (1,
    ↪  11, 1);
119 INSERT INTO CORPS (source, target, ordinal, txtval)    VALUES
    ↪  (11, 14, 1,' Encore des fake news');
120 INSERT INTO TITRE (source, target, txtval)             VALUES
    ↪  (11, 12, 'news');
121 INSERT INTO AUTEUR(source, target, txtval)             VALUES
    ↪  (11, 13, 'j1');
122 INSERT INTO JOURNALISTES (source, target, ordinal)     VALUES (0,
    ↪  15, 2);
123 INSERT INTO JOURNALISTE (source, target, ordinal)      VALUES
    ↪  (15, 16, 1);
124 INSERT INTO JOURNALISTEID (source, target, txtval)     VALUES
    ↪  (16, 19, 'j1');
125 INSERT INTO NOM (source, target, ordinal, txtval)      VALUES
    ↪  (16, 17, 1, 'Vuillard');
126 INSERT INTO PRENOM (source, target, ordinal, txtval)   VALUES
    ↪  (16, 18, 1, 'Eric');
127 INSERT INTO JOURNALISTE (source, target, ordinal)      VALUES
    ↪  (15, 20, 2);
128 INSERT INTO JOURNALISTEID (source, target, txtval)     VALUES
    ↪  (20, 23, 'j2');
```

```
129 INSERT INTO NOM (source, target, ordinal, txtval)      VALUES
    ↪  (20, 21, 1, 'Dupont');
130 INSERT INTO PRENOM (source, target, ordinal, txtval)   VALUES
    ↪  (20, 22, 1, 'Jean');
131 INSERT INTO ANONYMOUS(source, target, ordinal, txtval) VALUES
    ↪  (20, 24, 1, 'non');
```

1.2.2 Monet

```
1  CREATE TABLE PRESSE (
2          node int,
3          txtval  varchar(30),
4          numval int
5  );
6  CREATE TABLE PRESSE_JOURNAL (
7          node int,
8          txtval  varchar(30),
9          numval int
10 );
11
12 CREATE TABLE PRESSE_JOURNAL_NOM (
13         node int,
14         txtval  varchar(30),
15         numval int
16 );
17 CREATE TABLE PRESSE_JOURNAL_DIRECTEUR (
18         node int,
19         txtval  varchar(30),
20         numval int
21 );
22
23 CREATE TABLE PRESSE_JOURNAL_DIRECTEUR_NOM (
24         node int,
25         txtval  varchar(30),
26         numval int
27 );
28
29 CREATE TABLE PRESSE_JOURNAL_DIRECTEUR_PRENOM (
30         node int,
31         txtval  varchar(30),
32         numval int
33 );
34
35 CREATE TABLE PRESSE_JOURNAL_ARTICLE  (
36         node int,
37         txtval  varchar(30),
```

```
        numval int
);

CREATE TABLE PRESSE_JOURNAL_ARTICLE_CORPS  (
        node int,
        txtval  varchar(30),
        numval int
);

CREATE TABLE PRESSE_JOURNAL_ARTICLE_TITRE  (
        node int,
        txtval  varchar(30),
        numval int
);

CREATE TABLE PRESSE_JOURNAL_ARTICLE_AUTEUR  (
        node int,
        txtval  varchar(30),
        numval int
);

CREATE TABLE PRESSE_JOURNALISTES (
        node int,
        txtval  varchar(30),
        numval int
);

CREATE TABLE PRESSE_JOURNALISTES_JOURNALISTE (
        node int,
        txtval  varchar(30),
        numval int
);

CREATE TABLE PRESSE_JOURNALISTES_JOURNALISTE_NOM (
        node int,
        txtval  varchar(30),
        numval int
);

CREATE TABLE PRESSE_JOURNALISTES_JOURNALISTE_PRENOM (
        node int,
        txtval  varchar(30),
        numval int
);

CREATE TABLE PRESSE_JOURNALISTES_JOURNALISTE_IDJ (
```

```
84          node int,
85          txtval  varchar(30),
86          numval int
87  );
88
89  CREATE TABLE PRESSE_JOURNALISTES_JOURNALISTE_ANONYMISATION (
90          node int,
91          txtval  varchar(30),
92          numval int
93  );
94
95  INSERT INTO PRESSE(node)
    ↪  VALUES(0);
96  INSERT INTO PRESSE_JOURNAL(node)
    ↪  VALUES(1);
97  INSERT INTO PRESSE_JOURNAL_NOM(node,txtval)
    ↪  VALUES(2,'CNEWS');
98  INSERT INTO PRESSE_JOURNAL_DIRECTEUR(node)
    ↪  VALUES(3);
99  INSERT INTO PRESSE_JOURNAL_DIRECTEUR_NOM(node,txtval)
    ↪  VALUES(4,'Pepega');
100 INSERT INTO PRESSE_JOURNAL_DIRECTEUR_PRENOM(node,txtval)
    ↪  VALUES(5,'KEKW');
101 INSERT INTO PRESSE_JOURNAL_ARTICLE(node)
    ↪  VALUES(7);
102 INSERT INTO PRESSE_JOURNAL_ARTICLE_TITRE(node, txtval)
    ↪  VALUES(8, 'fake');
103 INSERT INTO PRESSE_JOURNAL_ARTICLE_AUTEUR(node, txtval)
    ↪  VALUES(9, 'j1');
104 INSERT INTO PRESSE_JOURNAL_ARTICLE_CORPS(node,txtval)
    ↪  VALUES (10,'fakenews');
105 INSERT INTO PRESSE_JOURNAL_ARTICLE(node)
    ↪  VALUES(11);
106 INSERT INTO PRESSE_JOURNAL_ARTICLE_TITRE(node, txtval)
    ↪  VALUES(12, 'news');
107 INSERT INTO PRESSE_JOURNAL_ARTICLE_AUTEUR(node, txtval)
    ↪  VALUES(13, 'j1');
108 INSERT INTO PRESSE_JOURNAL_ARTICLE_CORPS(node,txtval)
    ↪  VALUES (14,' more fakenews');
109 INSERT INTO PRESSE_JOURNALISTES(node)
    ↪  VALUES (15);
110 INSERT INTO PRESSE_JOURNALISTES_JOURNALISTE(node)
    ↪  VALUES (16);
111 INSERT INTO PRESSE_JOURNALISTES_JOURNALISTE_NOM(node,txtval)
    ↪  VALUES (17,'Vuillard');
```

```
112 INSERT INTO PRESSE_JOURNALISTES_JOURNALISTE_PRENOM(node,txtval)
    ↪  VALUES (18,'Eric');
113 INSERT INTO PRESSE_JOURNALISTES_JOURNALISTE_IDJ(node,txtval)
    ↪  VALUES (19, 'j1');
114 INSERT INTO PRESSE_JOURNALISTES_JOURNALISTE(node)
    ↪  VALUES (20);
115 INSERT INTO PRESSE_JOURNALISTES_JOURNALISTE_NOM(node,txtval)
    ↪  VALUES (21,'Dupont');
116 INSERT INTO PRESSE_JOURNALISTES_JOURNALISTE_PRENOM(node,txtval)
    ↪  VALUES (22,'Jean');
117 INSERT INTO PRESSE_JOURNALISTES_JOURNALISTE_IDJ(node,txtval)
    ↪  VALUES (23, 'j2');
118 INSERT INTO
    ↪  PRESSE_JOURNALISTES_JOURNALISTE_ANONYMISATION(node,txtval)
    ↪  VALUES (24, 'oui');
```

1.3 laide du langage SQL, exprimer cinq requtes XPath de votre choix sur chaque schma

1.3.1 Le nom du journal

```
1 /presse/journal/nom
```

Vertical-Edge

```
1 SELECT N.txtval nom_journal
2 FROM PRESSE  P,
3      JOURNAL J,
4      NOM     N
5 WHERE
6        P.Target = J.Source
7 AND    J.Target = N.Source;
```

Monet

```
1 SELECT txtval nom_journal
2 FROM   PRESSE_JOURNAL_NOM;
```

1.3.2 Le nom du directeur

```
1 /presse/journal/directeur/nom
```

Vertical-Edge

```
1 SELECT N.txtval nom_directeur
2 FROM PRESSE    P,
```

```
      JOURNAL   J,
      DIRECTEUR D,
      NOM       N
WHERE
       P.Target = J.Source
AND    J.Target = D.Source
AND    D.Target = N.Source;
```

Monet

```
SELECT txtval nom_directeur
FROM PRESSE_JOURNAL_DIRECTEUR_NOM;
```

1.3.3 Le titre de tout les articles

```
/presse/journal/article/@titre
```

Vertical-Edge

```
SELECT T.txtval titre_article
FROM PRESSE  P,
     JOURNAL J,
     ARTICLE A,
     TITRE   T
WHERE P.Target = J.Source
AND   J.Target = A.Source
AND   A.Target = T.Source;
```

Monet

```
SELECT txtval titre_article
FROM PRESSE_JOURNAL_ARTICLE_TITRE;
```

1.3.4 Le nombre d article

```
/presse/journal/count(article)
```

Vertical-Edge

```
SELECT COUNT(A.Target) nombre_d_article
FROM PRESSE  P,
     JOURNAL J,
     ARTICLE A
WHERE P.Target = J.Source
AND   J.Target = A.Source;
```

Monet

```
SELECT COUNT(A.node) nombre_d_article
FROM PRESSE_JOURNAL_ARTICLE A;
```

1.3.5 Les identifiants des journalistes

```
/presse/journal/journalistes/journaliste/@idJ
```

Vertical-Edge

```
SELECT J3.txtval identifiant_de_journaliste
FROM PRESSE        P,
     JOURNALISTES  J1,
     JOURNALISTE   J2,
     JOURNALISTEID J3
WHERE P.Target = J1.Source
AND   J1.Target = J2.Source
AND   J2.Target = J3.Source;
```

Monet

```
SELECT txtval identifiant_de_journaliste
FROM PRESSE_JOURNALISTES_JOURNALISTE_IDJ;
```

Chapitre 2

Stockage schema-aware : Verical-Edge vs Monet

2.1 A partir de la DTD pour les batiments presentee dans lenonce du TD1, definir un schema de stockagerelationnel suivant la methode presentee en cours.

2.1.1 DTD batiment

```
<!DOCTYPE batiment [
<!ELEMENT batiment (etage)+ >
<!ELEMENT etage (description,(bureau+|salle+)) >
<!ELEMENT description (#PCDATA) >
<!ELEMENT bureau (code, personne*) >
<!ELEMENT code (#PCDATA) >
<!ELEMENT personne (#PCDATA) >
<!ELEMENT salle (nombrePlaces) >
<!ELEMENT nombrePlaces (#PCDATA) >]>
```

2.1.2 Suppression des symboles +

```
<!DOCTYPE batiment [
<!ELEMENT batiment (etage*,etage) >
<!ELEMENT etage (description,((bureau*,bureau)|(salle*,salle)) >
<!ELEMENT description (#PCDATA) >
<!ELEMENT bureau (code, personne*) >
<!ELEMENT code (#PCDATA) >
<!ELEMENT personne (#PCDATA) >
<!ELEMENT salle (nombrePlaces) >
```

```
<!ELEMENT nombrePlaces (#PCDATA) >]>
```

2.1.3 Suppression de l'ordre et des correlations

```
<!DOCTYPE batiment [
<!ELEMENT batiment (etage* | etage) >
<!ELEMENT etage (description | bureau* | bureau | salle* |
↪  salle)) >
<!ELEMENT description (#PCDATA) >
<!ELEMENT bureau (code | personne*) >
<!ELEMENT code (#PCDATA) >
<!ELEMENT personne (#PCDATA) >
<!ELEMENT salle (nombrePlaces) >
<!ELEMENT nombrePlaces (#PCDATA) >]>
```

2.1.4 Simplifications

r |r* est equivalent a r*

```
<!DOCTYPE batiment [
<!ELEMENT batiment (etage*) >
<!ELEMENT etage (description | bureau* | salle*) >
<!ELEMENT description (#PCDATA) >
<!ELEMENT bureau (code | personne*) >
<!ELEMENT code (#PCDATA) >
<!ELEMENT personne (#PCDATA) >
<!ELEMENT salle (nombrePlaces) >
<!ELEMENT nombrePlaces (#PCDATA) >]>
```

2.1.5 Representation sous forme de graphe

Voir graphe en annexe

2.1.6 Relations

- batiment(batimentID : integer, flagRoot : integer)
- etage(etageID : integer, batimentID : integer, description : string)
- bureau(bureauID : integer, etageID : integer, code : string)
- personne(personneID : integer, bureauID : integer)
- salle(salleID : integer, etageID : integer, nombreDePlace : integer)

2.1.7 Creation des tables

```
CREATE TABLE BATIMENT (
       batimentID int NOT NULL PRIMARY KEY,
       flagRoot int NOT NULL
);
```

```
5
6  CREATE TABLE ETAGE (
7          etageID int NOT NULL PRIMARY KEY,
8          batimentID int NOT NULL,
9          description varchar(30),
10         CONSTRAINT fk_batimentID
11         FOREIGN KEY (batimentID)
12         REFERENCES BATIMENT (batimentID)
13 );
14
15 CREATE TABLE BUREAU (
16         bureauID int NOT NULL PRIMARY KEY,
17         etageID int NOT NULL,
18         code varchar(10),
19         CONSTRAINT fk_bureau_etageID
20         FOREIGN KEY (etageID)
21         REFERENCES ETAGE (etageID)
22 );
23
24 CREATE TABLE PERSONNE (
25         personneID int NOT NULL PRIMARY KEY,
26         bureauID int NOT NULL,
27         CONSTRAINT fk_bureauID
28         FOREIGN KEY (bureauID)
29         REFERENCES BUREAU (bureauID)
30 );
31
32 CREATE TABLE SALLE (
33         salleID int NOT NULL PRIMARY KEY,
34         etageID int NOT NULL,
35         nombreDePlace int,
36         CONSTRAINT fk_salle_etageID
37         FOREIGN KEY (etageID)
38         REFERENCES ETAGE (etageID)
39 );
```

2.2 Peupler les tables avec des lignes correspondants au document XML que vous propose en reponse a laquestion 1 du TD1

```
1 INSERT INTO BATIMENT (batimentID, flagRoot)          VALUES (0,
  ↪  1);
2 INSERT INTO ETAGE (etageID, batimentID, description) VALUES (0,
  ↪  0, 'descriptionETAGE1');
```

```
INSERT INTO ETAGE (etageID, batimentID, description) VALUES (1,
↪  0, 'descriptionETAGE2');
INSERT INTO SALLE (salleID, etageID, nombreDePlace)  VALUES (0,
↪  0, 25);
INSERT INTO SALLE (salleID, etageID, nombreDePlace)  VALUES (1,
↪  0, 50);
INSERT INTO BUREAU (bureauID, etageID, code)         VALUES (0,
↪  1, 'B02');
INSERT INTO PERSONNE (personneID, bureauID)          VALUES (0,
↪  0);
INSERT INTO BUREAU (bureauID, etageID, code)         VALUES (1,
↪  1, 'B03');
INSERT INTO PERSONNE (personneID, bureauID)          VALUES (1,
↪  1);
INSERT INTO PERSONNE (personneID, bureauID)          VALUES (2,
↪  1);
```

2.3 A laide du langage SQL, exprimer cinq requtes XPath de votre choix

2.3.1 Les descriptions d'etage du batiment d'identifiant 0

```
//batiment[@id=0]/description/text()
```

Requte SQL

```
SELECT DESCRIPTION
FROM ETAGE
WHERE batimentID = 0;
```

2.3.2 Le nombre de place des salles de l etage d identifiant 0 du batiment d identifiant 0

```
//batiment[@id=0]/etage[@id=0]/salle/nombredeplace
```

Requte SQL

```
SELECT S.NOMBREDEPLACE
FROM SALLE S JOIN ETAGE E ON S.etageID = E.etageID
WHERE E.batimentID = 0 AND E.etageID = 0;
```

2.3.3 Le nombre de personne travaillant dans des bureaux

```
count(//batiment/etage/bureau/personne)
```

Requte SQL

```
SELECT count(*)
FROM personne;
```

2.3.4 Le code des bureaux de l etage 1 du batiment 0

```
//batiment[@id=0]/etage[@id=1]/bureau/@code
```

Requte SQL

```
SELECT B.code
FROM BUREAU B JOIN ETAGE E ON B.etageID = E.etageID
WHERE E.batimentID = 0 AND E.etageID = 1;
```

2.3.5 Le nombre d'etage du batiment 0

```
count(//batiment[@id=0]/etage)
```

Requte SQL

```
SELECT COUNT(*)
FROM ETAGE
WHERE batimentID = 0;
```

2

Chapitre 3

Interval-encoding avec SAX

3.1 Illustrer lencodage (1) begin/end et (2) Dewey de lXML du document propos pour les btiments

3.1.1 BeginEnd

Voir arbre et graphe begin/end en annexe

3.1.2 Dewey

Voir arbre Dewey en annexe

3.2 laide du langage SQL, crer le schma de stockage pour les intervalles et ensuite peupler la tableNODE avec les lignes correspondantes au document propos pour les btiments

```
1 CREATE TABLE NODE (
2         begin int NOT NULL,
3         end int NOT NULL,
4         par int,
5         tag varchar(20),
6         type varchar(20) NOT NULL
7 );
8
9 INSERT INTO NODE (begin, end, tag, type) VALUES (1, 50,
  ↪  'batiment', 'ELT');
```

```
10 INSERT INTO NODE                          VALUES (2, 19, 1,
   ↪  'etage', 'ELT');
11 INSERT INTO NODE                          VALUES (3, 6, 2,
   ↪  'description', 'ELT');
12 INSERT INTO NODE (begin, end, par, type) VALUES (4, 5, 3,
   ↪  'TEXT');
13 INSERT INTO NODE                          VALUES (7, 12, 2,
   ↪  'salle', 'ELT');
14 INSERT INTO NODE                          VALUES (8, 11, 7,
   ↪  'nombrePlace', 'ELT');
15 INSERT INTO NODE (begin, end, par, type) VALUES (9, 10, 8,
   ↪  'TEXT');
16 INSERT INTO NODE                          VALUES (13, 18, 2,
   ↪  'salle', 'ELT');
17 INSERT INTO NODE                          VALUES (14, 17, 13,
   ↪  'nombrePlace', 'ELT');
18 INSERT INTO NODE (begin, end, par, type) VALUES (9, 10, 8,
   ↪  'TEXT');
19 INSERT INTO NODE                          VALUES (20, 49, 1,
   ↪  'etage', 'ELT');
20 INSERT INTO NODE                          VALUES (21, 24, 20,
   ↪  'description', 'ELT');
21 INSERT INTO NODE (begin, end, par, type) VALUES (22, 23, 21,
   ↪  'TEXT');
22 INSERT INTO NODE                          VALUES (25, 34, 20,
   ↪  'bureau', 'ELT');
23 INSERT INTO NODE                          VALUES (26, 29, 25,
   ↪  'code', 'ELT');
24 INSERT INTO NODE (begin, end, par, type) VALUES (27, 28, 26,
   ↪  'TEXT');
25 INSERT INTO NODE                          VALUES (30, 33, 25,
   ↪  'personne', 'ELT');
26 INSERT INTO NODE (begin, end, par, type) VALUES (31, 32, 30,
   ↪  'TEXT');
27 INSERT INTO NODE                          VALUES (35, 48, 20,
   ↪  'bureau', 'ELT');
28 INSERT INTO NODE                          VALUES (36, 39, 35,
   ↪  'code', 'ELT');
29 INSERT INTO NODE (begin, end, par, type) VALUES (37, 38, 36,
   ↪  'TEXT');
30 INSERT INTO NODE                          VALUES (40, 43, 35,
   ↪  'personne', 'ELT');
31 INSERT INTO NODE (begin, end, par, type) VALUES (41, 42, 40,
   ↪  'TEXT');
32 INSERT INTO NODE                          VALUES (44, 47, 35,
   ↪  'personne', 'ELT');
```

```
INSERT INTO NODE (begin, end, par, type) VALUES (45, 46, 44,
↪  'TEXT');
```

3.3 laide de la classeSaxParser(.java sur Moodle), programmer lencodage par intervalles begin/end

```
public class SaxParser extends DefaultHandler {
  /** Constants used for JAXP 1.2 */
  static final String JAXP_SCHEMA_LANGUAGE =
      "http://java.sun.com/xml/jaxp/properties/schemaLanguage";
  static final String W3C_XML_SCHEMA =
      "http://www.w3.org/2001/XMLSchema";
  static final String JAXP_SCHEMA_SOURCE =
      "http://java.sun.com/xml/jaxp/properties/schemaSource";

  private Deque<Integer> stackBegin;
  private Integer counter;

  // Parser calls this once at the beginning of a document
  public void startDocument() throws SAXException {
      stackBegin = new ArrayDeque<>();
      counter = 0;
  }

  // Parser calls this for each opening of an element in a
  ↪  document
  public void startElement(String namespaceURI, String localName,
                           String qName, Attributes atts)
      throws SAXException
  {
      stackBegin.push(++counter);
  }

  // Parser calls this for each end of an element in a document
  public void endElement(String namespaceURI, String localName,
                           String qName)
      throws SAXException
  {
      int begin = stackBegin.pop();
      int end = ++counter;
      Integer parent = stackBegin.peek();

      String parentStr;
```

```
37          if (parent == null) {
38              parentStr = "NULL";
39          } else {
40              parentStr = String.valueOf(parent);
41          }
42
43          String output = "INSERT INTO NODE (begin, end, parent, tag,
            ↪  nodetype) VALUES(";
44          output += begin + ", " + end + ", " + parentStr + ", '" +
            ↪  qName + "', 'element');";
45
46          System.out.println(output);
47      }
48
49      // Parser calls this once after parsing a document
50      public void endDocument() throws SAXException {
51      }
52
53      // Parser calls after parsing a text node
54      public void characters(char[] ch, int start, int length) throws
        ↪  SAXException
55      {
56          String str = new String(ch, start, length);
57          str = str.replace(" ", "").replace("\n", "");
58          // System.out.println(str.length());
59
60          if (str.length() != 0) {
61              int begin = ++counter;
62              int end = ++counter;
63              String output = "INSERT INTO NODE (begin, end, parent,
                ↪  tag, nodetype) VALUES(";
64              output += begin + ", " + end + ", " + stackBegin.peek()
                ↪  + ", NULL, 'text');";
65
66              System.out.println(output);
67          }
68      }
69  }
```

Annexe A

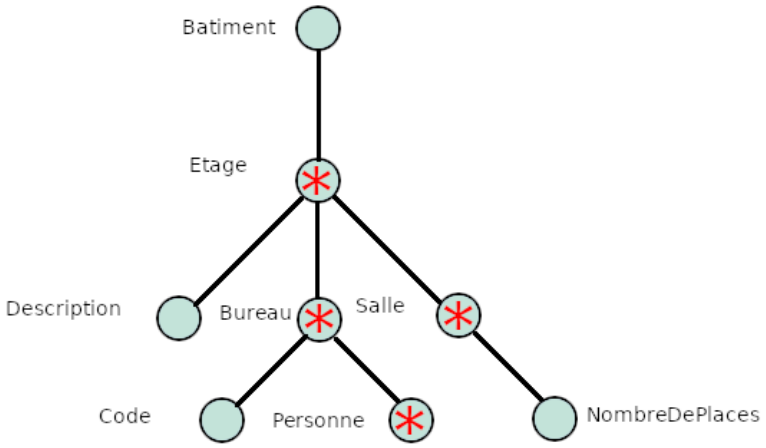
XML - Presse

```xml
<presse>
  <journal>
    <nom>
      Cnews
    </nom>
    <directeur>
      <nom>
        Pepega
      </nom>
      <prenom>
        Kekw
      </prenom>
    </directeur>
    <article titre="fake" auteur="j1">
      <corps>
        Des fake news
      </corps>
    </article>
    <article titre="news" auteur="j1">
      <corps>
        Encore des fake news
      </corps>
    </article>
  </journal>
  <journalistes>
    <journaliste idJ="j1">
      <nom>
        vuillard
      </nom>
      <prenom>
        eric
```

```
32        </prenom>
33      </journaliste>
34      <journaliste idJ="j2" anonymisation="oui">
35        <nom>
36          Dupont
37        </nom>
38        <prenom>
39          Jean
40        </prenom>
41      </journaliste>
42    </journalistes>
43  </presse>
```

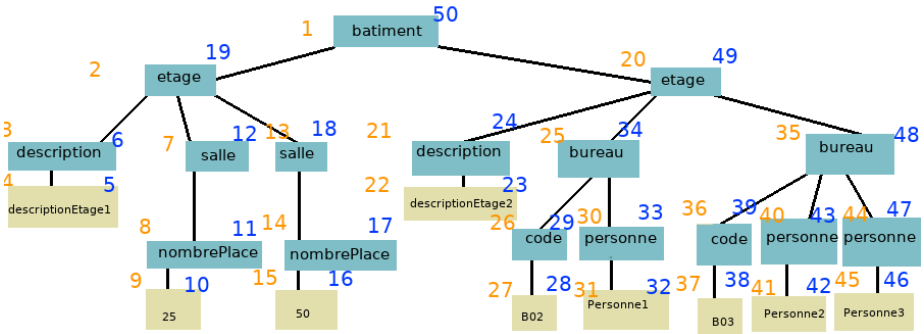
Annexe B

Representation sous forme de graphe


Batiment
Etage
Description
Bureau
Salle
Code
Personne
NombreDePlaces

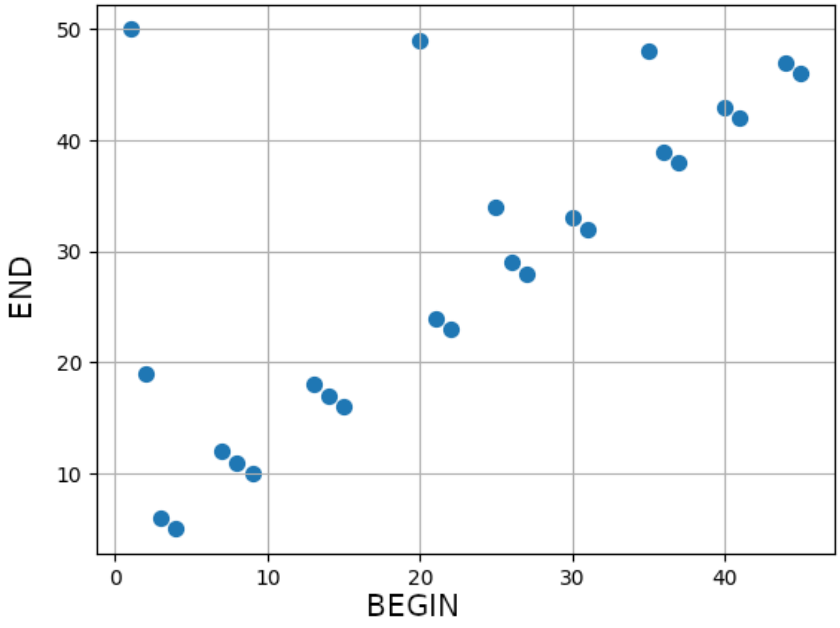
Annexe C

Arbre begin/end pour le document batiment


batiment
etage
etage
description
salle
salle
description
bureau
bureau
descriptionEtage1
nombrePlace
nombrePlace
descriptionEtage2
code
personne
code
personne
personne
25
50
B02
Personne1
B03
Personne2
Personne3

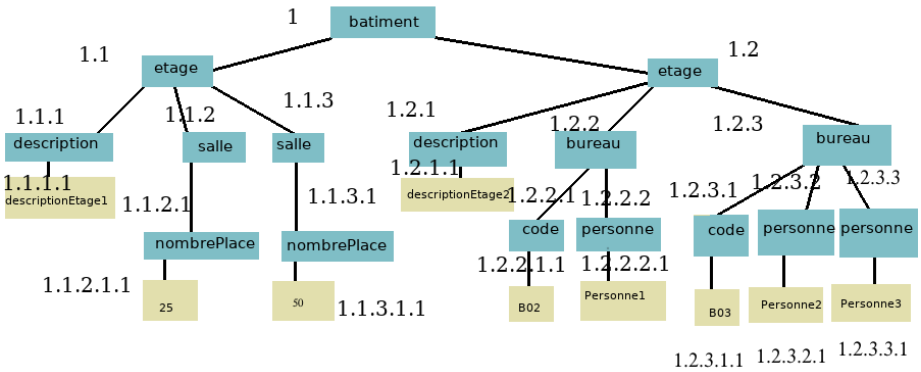
Annexe D

Graphe begin/end pour le document batiment


END
BEGIN
50
40
30
20
10
0
10
20
30
40

Annexe E

Arbre Dewey pour le document batiment


1
batiment
1.1
etage
1.2
etage
1.1.1
description
1.1.2
salle
1.1.3
salle
1.2.1
description
1.2.2
bureau
1.2.3
bureau
1.1.1.1
descriptionEtage1
1.1.2.1
nombrePlace
1.1.3.1
nombrePlace
1.2.1.1
descriptionEtage2
1.2.2.1
code
1.2.2.2
personne
1.2.3.1
code
1.2.3.2
personne
1.2.3.3
personne
1.1.2.1.1
25
1.1.3.1.1
50
1.2.2.1.1
B02
1.2.2.2.1
Personne1
1.2.3.1.1
B03
1.2.3.2.1
Personne2
1.2.3.3.1
Personne3