Report Project 1 :
XML Schema Definition

Rémy Detobel & Nathan Liccardo

March 31, 2018

1 Introduction

This document aims at describing and explaining all the choices that we make for this project. As a reminder, the main goal has been to create an XSD file containing a specific XML Schema Definition. This report is structured into two parts : variables and schema definitions and hypothesis. As during course lectures, we decided to use trees to represent our XSD structure.

2 Variables

This project use many different kinds of variables. Some of them are complex (see below) but most of them are simple. Simple types are :

- *String* : title, publisher, abstract, edition, author, editor;
- *Integer* : volume, number, price, impact.

Concerning the complex types, they are used for defining the year, the genre and the ISBN. As we need to declare a year using a tag or an attribute, we defined it into two parts. Firstly, we declared the format (using a simple type) and then, secondly, we defined an attribute and an element (using the year format). Genre and ISBN types are based on a restriction of the simple string type. For the first one, we have been restricted possible input (thriller, horror, sci/fi, romance, literature) using an enumeration. Finally, ISBN uses many formats (see below) that we defined using a set of patterns.

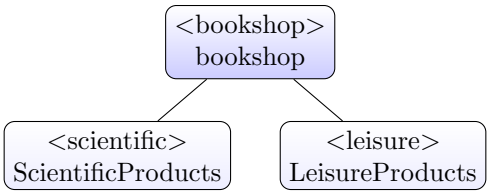
2.1 ISBN

ISO define two ISBN formats : ISBN-10 and ISBN-13 (10 or 13 characters). We decided to use ISBN-10 for this project. In this standard, the code is split into four parts : country (1 digit), editor (2, 3 or 5 digits), book number (6, 5 or 3 digits) and a verification code (1 digit). To check this property, we have used a pattern inside a restriction. Pattern allows us to define regular expressions :

- \d{1}-\d{5}-\d{3}-\d{1}
- \d{1}-\d{3}-\d{5}-\d{1}
- \d{1}-\d{2}-\d{6}-\d{1}

3 Structure of the schema

For this project, we were assigned to write an XSD file which define a book shop. This shop is separated into two parts : scientific products and leisure products. The following tree represent this first relation :

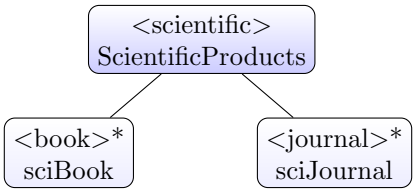


In this tree, each node contains the XML tag but also the type of the element. So here we can see that the "bookshop" type does have two children : `scientific` and `leisure`. These two types will be detailed in the following points. Note that they are not mandatory (hence their whiter color). The `bookshop` tag may be empty or contain only one of the two types.
Concretely in XML this gives the following structure :

```
<bookshop>
    <scientific>
        ...
    </scientific>
    <leisure>
        ...
    </leisure>
</bookshop>
```

3.1 Scientific products

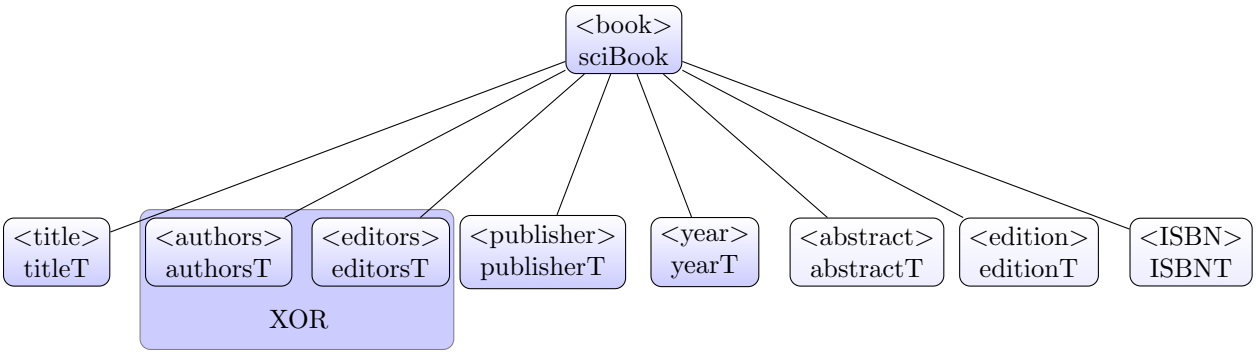
Scientific products are separated into two sub-categories. The first one define scientific books and the second one scientific journals. Those links can be represented as follow :



These two new type (sciBook and sciJournal) are a little more complex than what we have seen so far. As `bookshop`, `scientific` can be empty or contain only one of the two types. Note also that there can be several times the tag `books` or `journals` (hence the star (*) next to the tag).
Concretely in XML this gives the following structure :

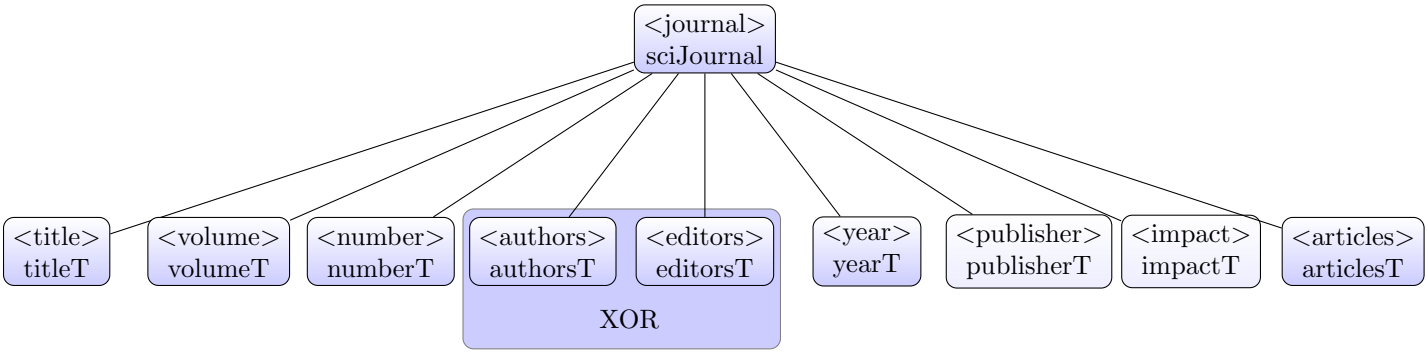
```
<bookshop>
    <scientific>
        <book>
            ...
        </book>
        <book>
            ...
        </book>
        <journal>
            ...
        </journal>
        ...
    </scientific>
    ...
</bookshop>
```

3.1.1 Scientific book



All types (except "authorsT" and "editorsT") have been defined previously (see point 2). The `<authors>` and `<editors>` tags still need to be defined. Both types contain lists of elements (respectively "authorT" and "editorT"). These two types cannot appear at the same time. Note that the last three elements are optional.

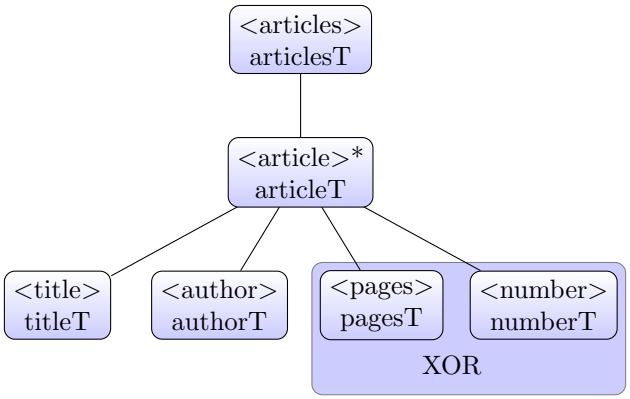
3.1.2 Scientific journal



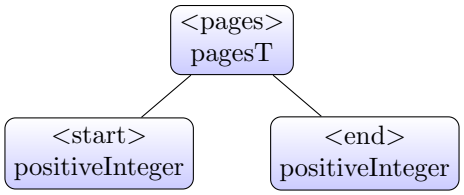
The structure is exactly the same as the previous point. All types (except "articlesT") have already been defined previously. Also as in the previous point, "authorsT" and "editorsT" cannot appear at the same time.

3.1.3 Articles

A scientific journal must have a list of the articles it contains. The tag `articles` therefore contains at least one tag `article` which itself contains attributes. All this can be represented by the following tree:



Like `authors` and `editors`, `pages` and `number` could not appear at the same time. The types "titleT", "authorT" and "numberT" have already be defined. For "pagesT" it's a bit different, we must to have a start page and an end page. Thus we have the following tree:



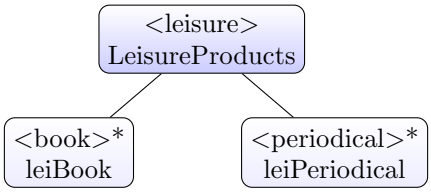
3.1.4 Impact

As indicated in section 2, the "impact" type is an integer calculated according to the number of citations, received in that year, of articles published in that journal during the two preceding years, divided by the total number of articles published in that journal during the two preceding years.
We can therefore see that the impact is different each year and that these two data are inseparable. The impact is characterized by an "year" attribute (which is "yearType").
Concretely in XML we can read this:

```
<impact year="XXXX"> ... </impact>
```

3.2 Leisure products

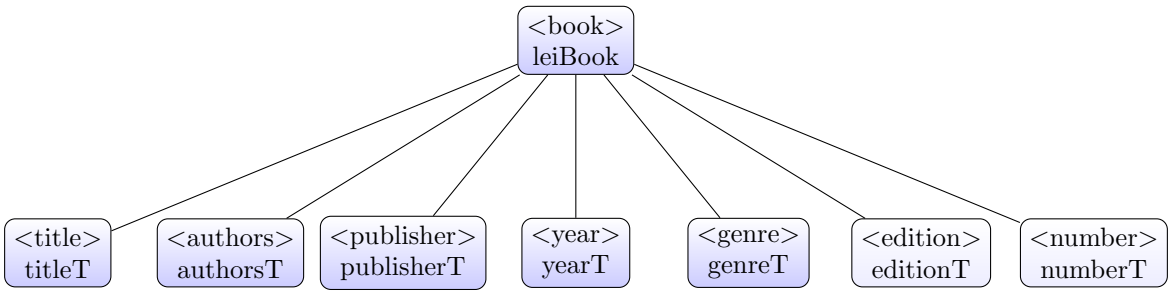
The second section of a book shop is dedicated to leisure products. As for scientific one, products are separated into two sub-categories. The first one is used to define leisure books and the second one leisure periodicals. Once again, those links can be illustrated by means of a tree :



Concretely in XML this gives the following structure :

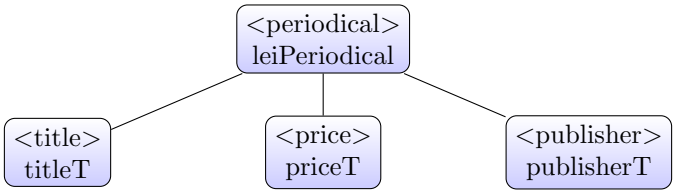
```
<bookshop>
    ...
    <leisure>
        <book>
            ...
        </book>
        <periodical>
            ...
        </periodical>
        <periodical>
            ...
        </periodical>
        ...
    </leisure>
</bookshop>
```

3.2.1 Leisure Book



All the types presented in the tree above have already been defined.

3.2.2 Periodical Leisure



Here too, everything has already been defined previously.

A Relation type

- bookshop → (scientific, ScientificProducts), (leisure, LeisureProducts)
- ScientificProducts → (book, sciBook)*, (journal, sciJournal)*
- LeisureProducts → (book, leiBook)*, (periodical, leiPeriodical)*
- sciBook → (title, titleT), (authors, authorsT)$^+$ | (editors, editorsT)$^+$, (publisher, publisherT), (year, yearT), (abstract, abstractT)?, (edition, editionT)?, (ISBN, ISBNT)?
- sciJournal → (title, titleT), (volume, volumeT), (number, numberT), (authors, authorsT)$^+$ | (editors, editorsT)$^+$, (year, yearT), (publisher, publisherT)?, (impact, impactT)?, (articles, articlesT)
- articlesT → (article, articleT)$^+$
- articleT → (title, titleT), (author, authorT), (pages, pagesT) | (number, numberT)
- pagesT → (start, positiveInteger), (end, positiveInteger)
- leiBook → (title, titleT), (authors, authorsT), (publisher, publisherT), (genre, genreT), (edition, editionT)?, (number, numberT)?
- leiPeriodical → (title, titleT), (price, priceT), (publisher, publisherT)