Report Project 1 :
XML Schema Definition

Rémy Detobel & Nathan Liccardo

April 23, 2018

1 Introduction

This document aims at describing and explaining all the choices that we made for this project. As a reminder, the main purpose has been to create an XSD file containing a specific XML Schema Definition. This report is divided into two parts : variable definitions and the XML Schema structure.

2 Variables definitions

Our implementation use two kinds of variables : simple and complex. Most of the time, complex variables are sequences of simple types. This section will be useful to describe variables that we will reuse in the XML schema structure section.

2.1 Simple variables

Simple variables are based on simple types (defined in the W3C standards). The project uses only two simple types to define every simple variables :

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

2.2 Complex variables

Year type Year type might be used with a tag or an attribute. For this reason, we decided to define a global year type which only restrict an integer type to the year format.

Genre type Genre is based on string restrictions. We restricted possible input strings (thriller, horror, sci/fi, romance, literature) using enumerations.

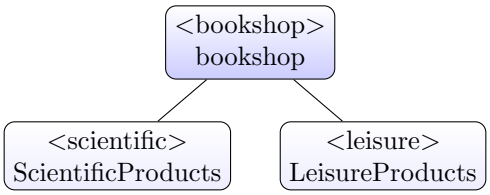
ISBN type ISO define two ISBN formats (10 or 13 characters). In this project, we have arbitrarily chosen to use ISBN-10 (10 characters). This standard split the code into four parts : a country (1 digit), an editor (2, 3 or 5 digits), a book number (6, 5 or 3 digits) and a verification code (1 digit). Has said before, we have been used patterns to check ISBN formats. Those patterns are defined as follows :

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

Which means that we can only use digits, in a structured forms and separated by dashes.

3 XML Schema structure

We were asked to write an XSD file defining a book shop. This shop had to be composed by scientific leisure products. This first relation is represented by the tree below. Lighter-coloured leafs represent non-mandatory elements.

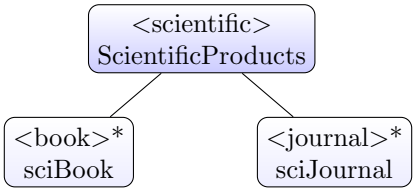


Each nodes contains a tag (XML) and a type name. Firstly, we can observe that `bookshop` have two children : `scientific` and `leisure` (see below). `Bookshop` may be empty, composed by one of the two element or by both. Finally, we obtain the below XML structure :

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

3.1 Scientific products

Scientific products are separated into two sub-categories : scientific books and scientific journals. For both of them, we can have zero or more occurrences. This property is represented using the asterisk in the following tree :

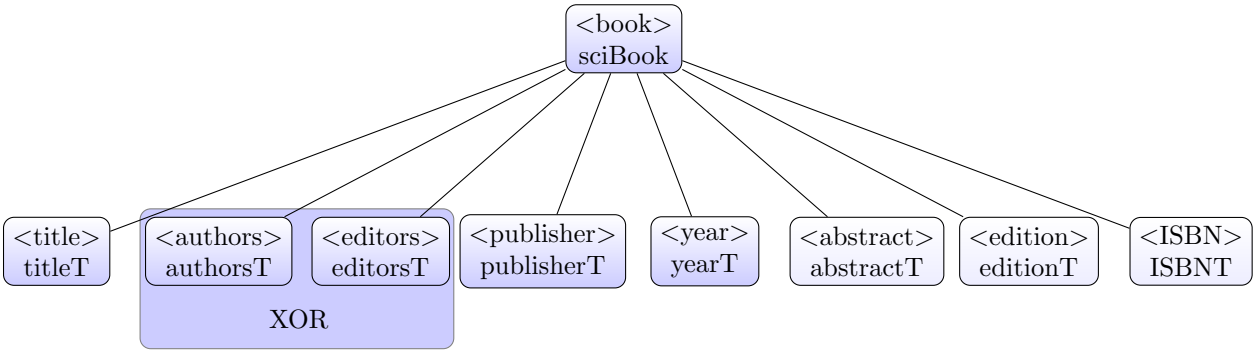


Which is represented using the below XML structure :

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

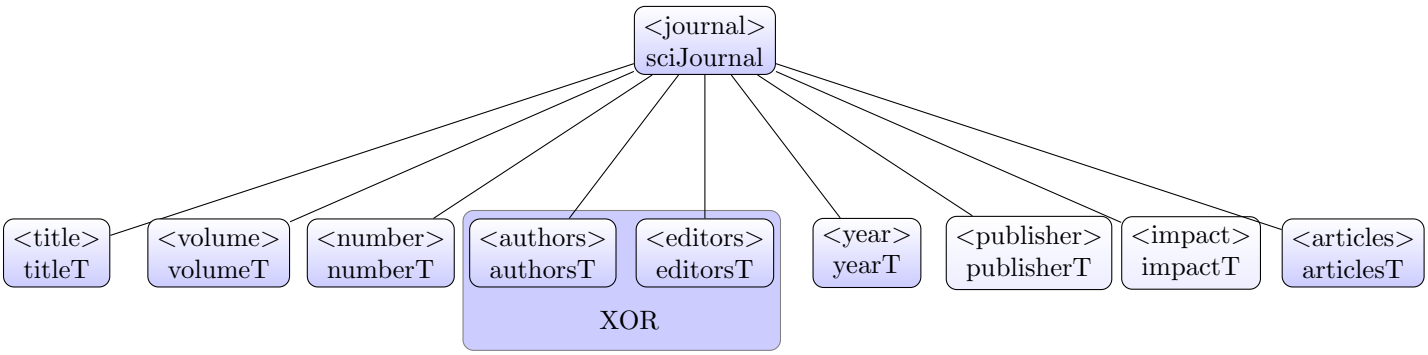
3.1.1 Scientific book

Scientific books are defined using simple types (previously defined) and two new complex types : `<authors>` and `<editors>`. Both contains list of elements (respectively “authorT” and “editorT”) and cannot appears at the same time. Please note also that the three last elements are optional.

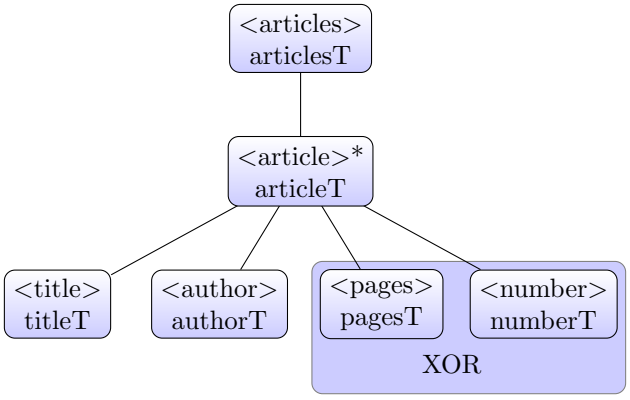


3.1.2 Scientific journal

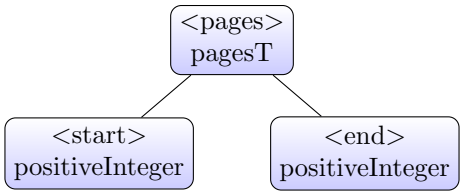
Scientific journals are structured as same as for scientific books. The difference lays in the used types (all are already defined excepted `articles`). Regarding to `author` and `editor` types, they are defined as for scientific books.



Articles A scientific journal must have a list of articles. Therefore, the `articles` tag contains at least one `article` tag which itself contains attributes. This can be represented as below :



Like `authors` and `editors`, `pages` and `number` could not appear at the same time. A `pages` element must have a start and end page (represented using *Integer* types). This relation is represented as follows :

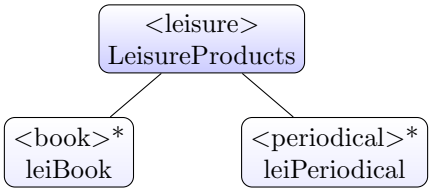


Impact As indicated previously (see section 2), the "impact" type is an integer. This integer is calculated according two factors : the number of citations (in one year) of articles published in that journal (within the two last years) and the total of articles published in that journal during the same two years. Finally, to obtain the "impact" value, we must divide the first factor by the second one. Therefore, impact value will be different each year. This implies that the impact is characterised by a "year" attribute. Below is the final XML structure :

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

3.2 Leisure products

Leisure products are separated into two parts : leisure books and leisure periodicals. The following relation has the same structure as for scientific products :

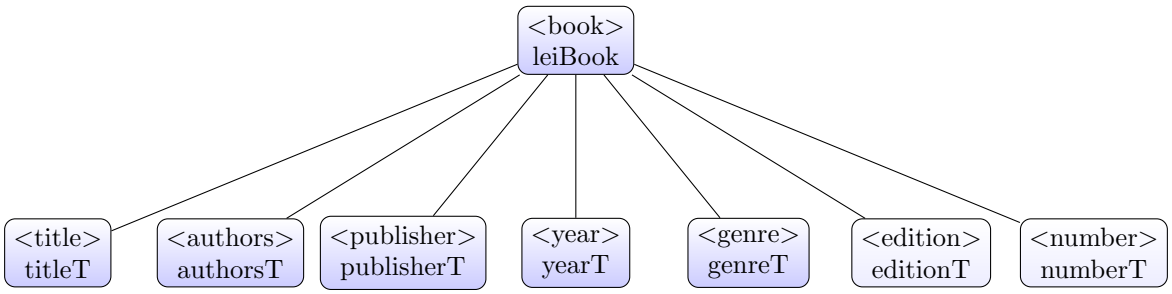


Once again, we can construct the XML code :

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

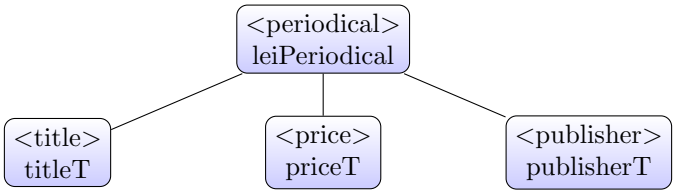
3.2.1 Leisure Book

Leisure books uses exactly the same structure as previous elements. In addition, all the used types have already been defined.



3.2.2 Leisure Periodicals

Leisure periodicals uses exactly the same structure as previous elements. Once again, every used types have already been defined.



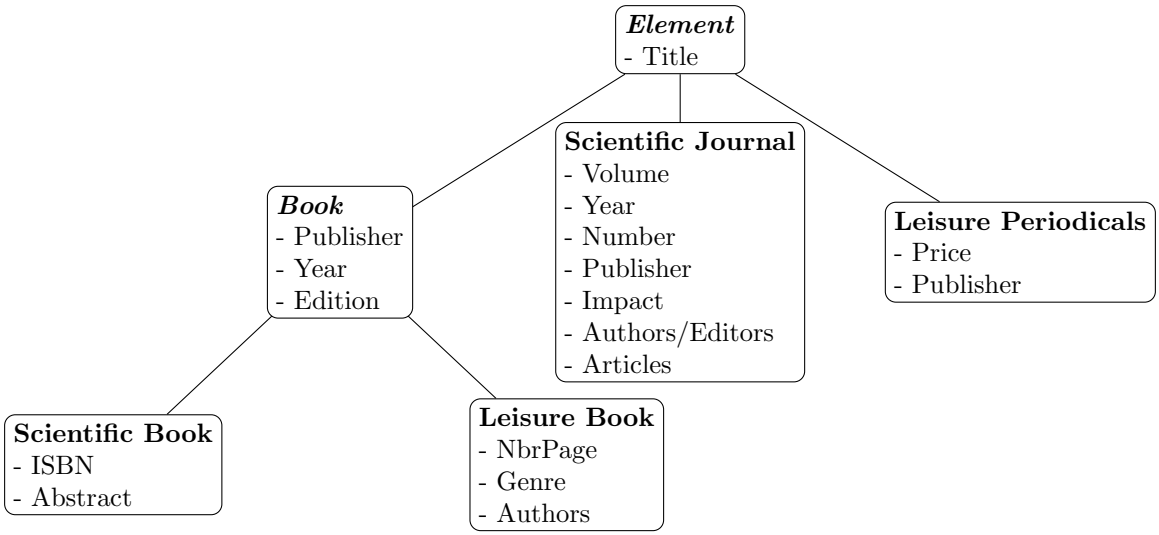
A Relation type

- 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)

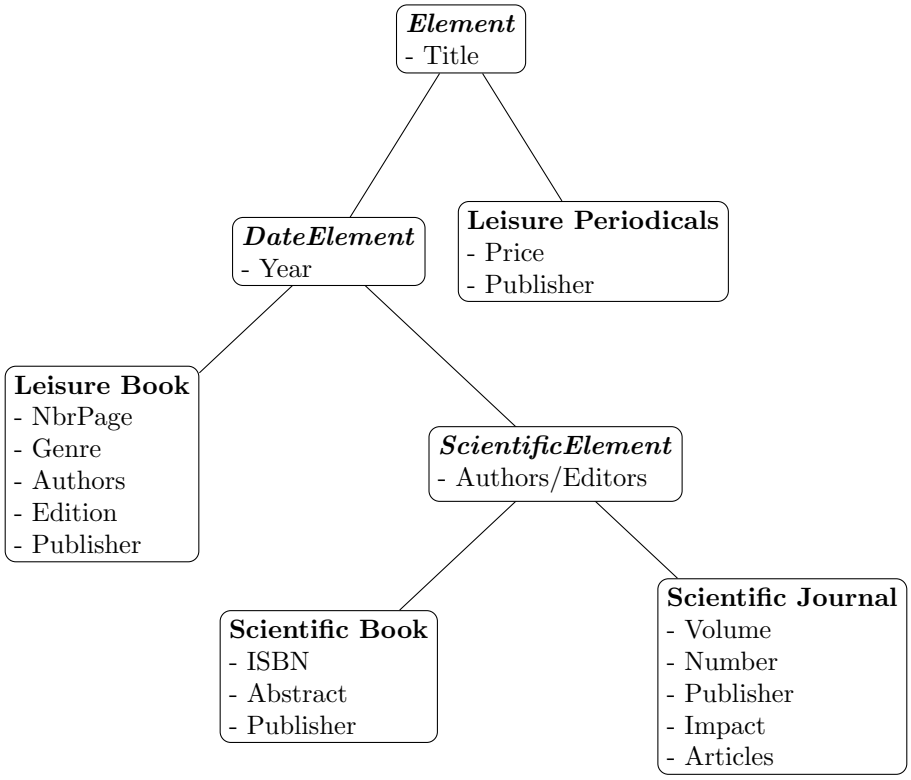
B Type specialisation

As in other programming languages, XML schema definition allows to define general and abstract types. Those types must then be specialised and specified before being used. For simplicity, we did not use those types inside our project. Nevertheless, we did note that some of these abstract types could be used. For this reason, we have been decided to add additional schemas and diagrams as appendix of this report. These diagrams describes different possible scenarios to implement - in an other way - our project. As an illustration, abstract property can be specified using a specific attribute as follow : `abstract="true"`. Italic texts are used to represent abstract types.

B.1 Book element



B.2 Date element



B.3 Published element