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 :

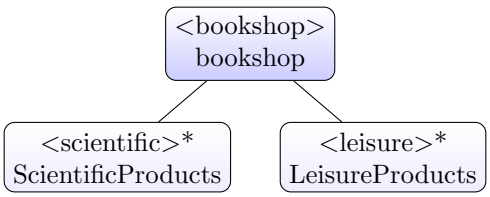
- *String* : title, publisher, abstract, edition, author, editor;
- *Integer* : volume, number, price, impact.

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

3 Structure of the schema

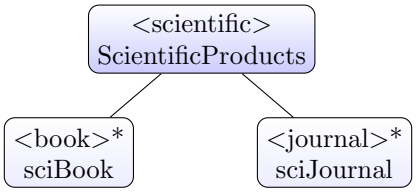
For this project, we were assigned to write an XSD file which define a book shop. This shop is separated into two parts : scientific products and leisure products. The following tree represent this first relation :



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. Note also that there can be several times the tag `scientific` or `leisure` (hence the star (*) next to the tag).

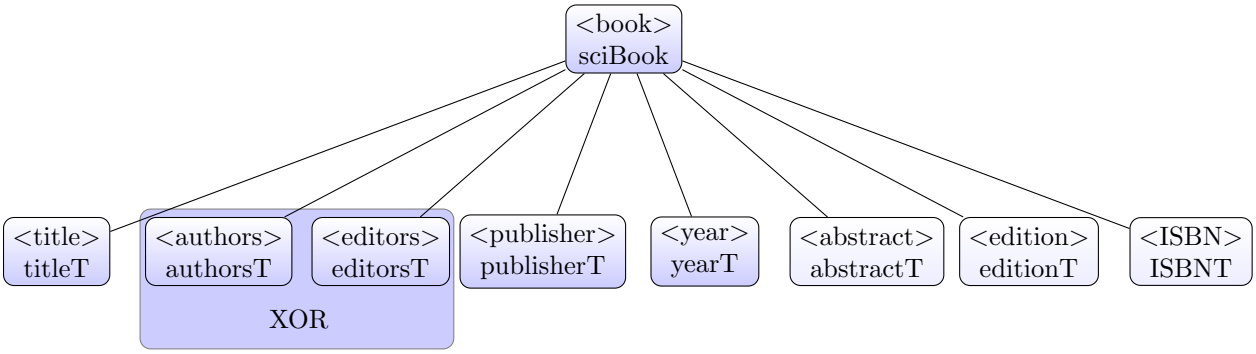
3.1 Scientific products

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 :



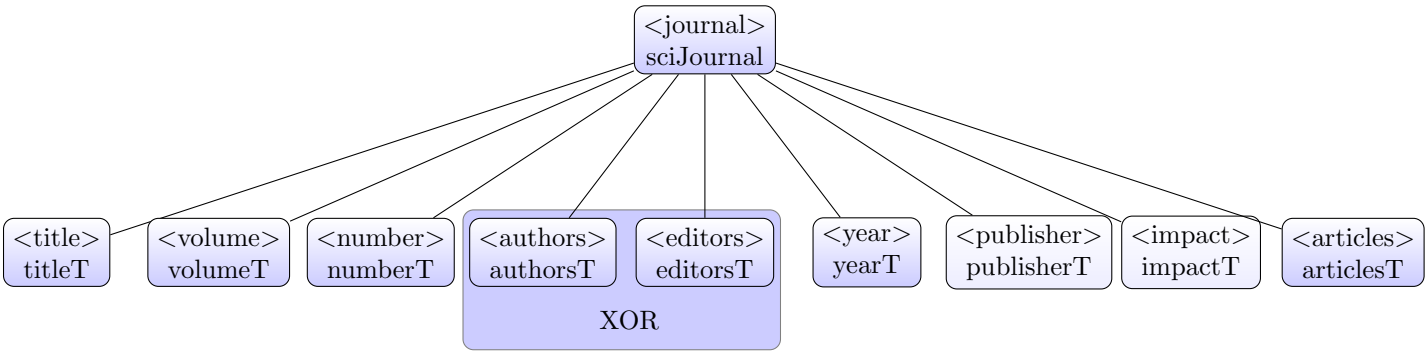
These two new type (sciBook and sciJournal) are a little more complex than what we have seen so far. As `bookshop`, `scientific` can be empty or contain only one of the two types and here too there may be several `books` or `journals`.

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.

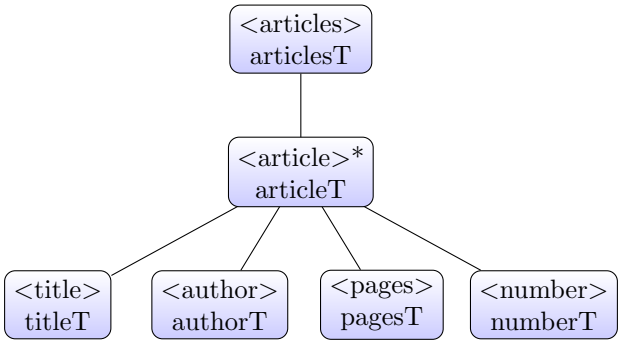
3.1.2 Scientific journal



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

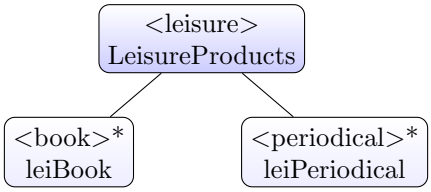
A scientific journal must have a list of the articles it contains. The tag `articles` therefore contains at least one tag `article` which itself contains attributes. All this can be represented by the following tree:



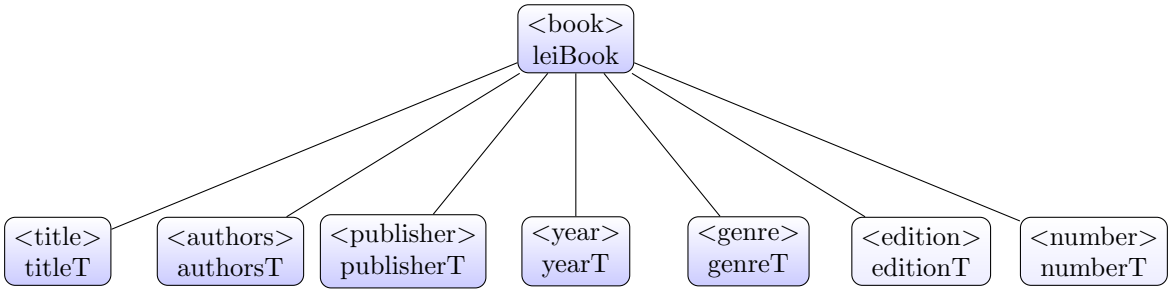
3.1.4 Impact

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 :

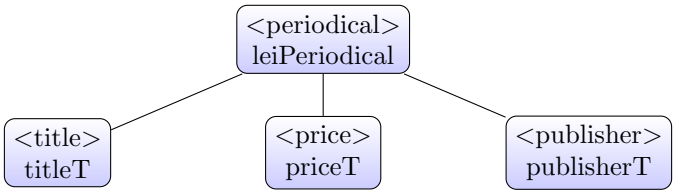


3.2.1 Leisure Book



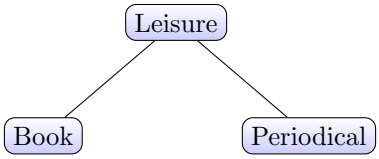
All the types presented in the tree above have already been defined.

3.2.2 Periodical Leisure



Here too, everything has already been defined previously.

4 Old part



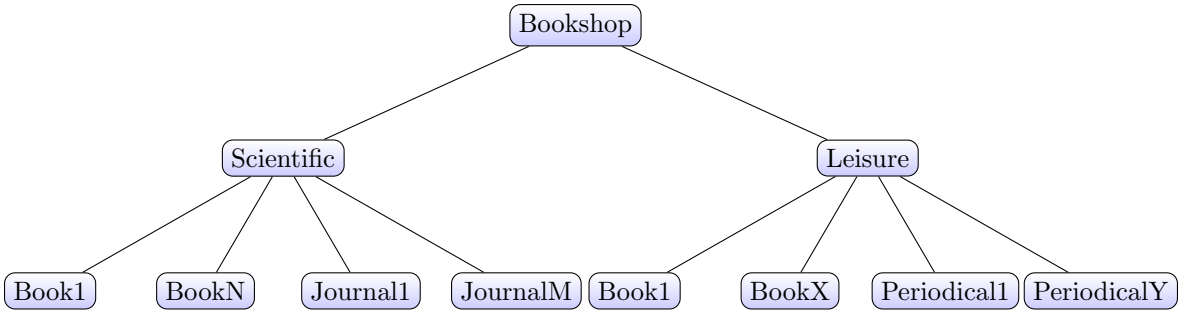
Following this tree, we can obtain two relations. These last are used to define an abstract bookshop (bookshopType) and a final leisure (leisureType). Finally, we obtain the next definitions :

- bookshopType → (leisure, leisureType)
- leisureType → (book, booktype2), (periodical, periodicalType)

As said in the previous sub-section, bookshopType is actually not completely defined. For this reason, there exist some incompatibilities between the scientific and leisure part. Notice that leisure books and scientific books are not using same types. In fact we will see in the next section that they have two different definitions.

4.1 Complete schema

The two previous parts were focused on the scientific and leisure definitions. We have now to merge those definitions to obtain the final tree. In fact, a bookshop is a combination of scientific and leisure products. Notice that for each element (book, journal, ...) we can also have multiple instances of it. Finally, we obtain the following tree :



Previously, we defined the bookshopType as an abstract one. In fact, this type was a sequence of scientificType and leisureType and could not be defined in one time. As we changed the definition of a bookshopType but also of a scientificType and a leisureType, we obtain three new relations :

- bookshopType → (scientific, scientificType),(leisure, leisureType)
- scientificType → (book, booktype1)*, (journal, journalType)*
- leisureType → (book, booktype2)*, (periodical, periodicalType)*

For the moment, we only defined links between elements. All those are created by defining complex types with some elements inside. Each of these elements are themselves referring to other complex type excepted for final types (Book, Journal, ...). The second part of this report will be focused on explaining and illustrating all the choices that we made during the implementation. For this reason, we will also detail the structure of each final types.

5 Hypotheses and choices

All along the previous section, we only described the global structure of the schema. We will now focus on what we called previously "final types". There exist four different kinds of this specific element : bookType1,

journalType, bookType2 and periodicalType. As said in the previous section, bookType1 and bookType2 are both identified by the `<book>` tag but are not referring to the same type. This can be achieved due to the scope definition. In fact, bookType1 is defined inside the scientific scope unlike bookType2 which is defined inside the leisure scope. Regarding to the two last types, journalType is identified by the `<journal>` tag (scientific scope) and periodicalType by the `<periodical>` tag (leisure scope). This section will be separated into two sub-sections : the XSD's structure and the final types structures.

5.1 XSD's structure

Our file has been split into five distinct parts. Each part is used to represent one level of the XML Schema. Indeed, the order of declarations is made from bottom to top level. Here is the used structure :

1. Simple types : This first section is used to define each atomic elements. This is mainly useful to reduce duplicate definitions and code. For example, we define the title element, the publisher element, etc.
2. Scientific part : Define all complex types which belong to the scientific product tree. We can see that scientificBook and scientificJournal are defined at the end of this section. Detail about this part will be given in the next section.
3. Leisure part : As the scientific part, leisure part is used to define all complex types which belong to the leisure tree. Once again, details about this part will be given later.
4. Products part : Define two complex types, scientific product and leisure product. Both can be seen as root elements from the two previous parts.
5. Bookshop part : Is used to merge both products elements. This section also define the root element of the xml file named `<b:bookshop>`.

5.2 Final types

This second section will be focused on the structure of each final types. As said before, there exist four different final types : bookType1, journalType, bookType2 and periodicalType.

5.2.1 BookType1

According to the assignment, a scientific book must be composed of : a title, a list of authors xor a list of editors, a publisher and a year of publication. Optionally, a book can have : an abstract, an edition and an ISBN. Formally, we obtain the following definition :

- bookType1 $\rightarrow$ (title, titleT), (authors, authorsT)$^+$ | (editors, editorsT)$^+$, (publisher, publisherT), (year, yearT), (abstract, abstractT)?, (edition, editionT)?, (ISBN, ISBNT)?

All the types, excepted authorsT and editorsT, used in the previous section are defined in the simple types section. Regarding to authorsT and editorsT, their definitions are more complex. Indeed, they are themselves complex types definitions. We made the arbitrary choice that a list of authors (or editors) must be structured as follow :

```
<authors>
    <author> ... </author>
    ....
    <author> ... </author>
</author>
```

Which implies that author and editor tags are defined by authorT and editorT. These new types are finally defined in the simple type section (as string elements).

5.2.2 JournalType

A scientific journal as also two parts : a mandatory one and an optional one. For the mandatory part, we must have : a title, a volume, a number, a list of authors xor a list of editors and a year. Concerning to the optional part, a journal type can have a publisher and an impact. In addition, a journal must also have a list of articles. Finally, we obtain this relation :

- journalType → (title, titleT), (volume, volumeT), (number, numberT), (authors, authorsT)$^+$ | (editors, editorsT)$^+$, (year, yearT), (publisher, publisherT)?, (impact, impactT)?, (articles, articlesT)

Were we have two new complex types : impact and article. All the other one are defined in the simple type section excepted editors and authors which were defined previously. An impact is defined with two elements namely a year and an impact factor. For this reason, we decided to structure this complex type as follow :

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

Notice that another solution was to create a complex type (impact) composed by a year tag and a factor tag. Regarding to the articles type, we had to define a list of articles. To do that, we defined a complex type composed itself by a list of article tags. Here is the XML logic :

```
<articles>
    <article> ... </article>
    ...
    <article> ... </article>
</articles>
```

Inside each article, we must find those elements : a title, an author, an interval of pages xor a number corresponding to the article. This can be formalised by the following relation :

- articlesT → (article, articleT)$^+$
- articleT → (title, titleT), (author, authorT), (pages, pagesT) | (number, numberT)

Note that this type contains only one author but also that we use the complex type pagesT and the simple type numberT (define in the first section). PagesT is a very simple complex type composed of two integers corresponding to the start and end page.

5.2.3 BookType2

The second book type is defined inside the leisure scope. For this reason, the composition of this new book type will differ from bookType1. Inside this new book type we must find : a title, a list of authors, a publisher, a year and a genre. Optionally, we can add an edition and a number. Formally, we obtain the following relation :

- bookType2 → (title, titleT), (authors, authorsT), (publisher, publisherT), (genre, genreT), (edition, editionT)?, (number, numberT)?

We were asked to restrict the value of a genre. To do that we used a restriction tag with multiple enumerations inside. All the other types were defined previously or are defined in the simple type section.

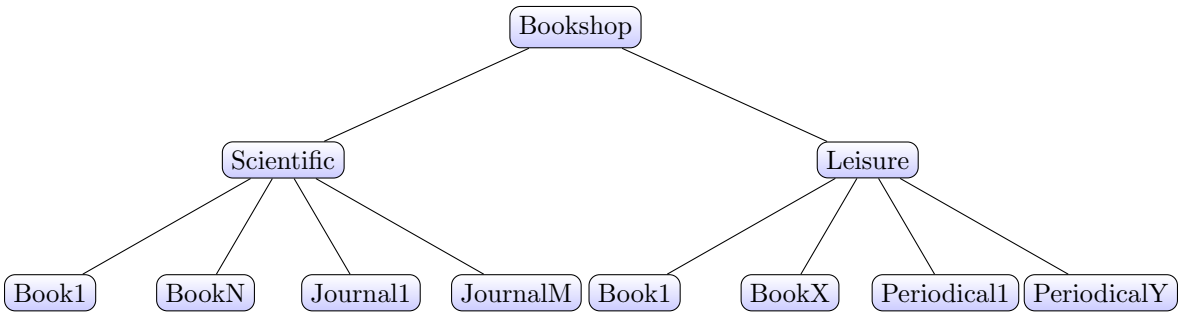
5.2.4 PeriodicalType

This last complex type is only composed by a sequence of three simple type elements. As these elements are very simple, we will not give any more informations about them. Here is the formal definition of a periodicalType :

- periodicalType → (title, titleT), (price, priceT), (publisher, publisherT)

5.3 Complete schema and relations

To complete this report, we can now show all the relations corresponding to the final tree. Note that those relations are now correctly named (as same as in the XSD file) and that we order relations from top to bottom (reverse file order). Due to their trivial aspects, we didn't add simple type (texts, numbers, ...)



- bookshopType → (scientific, scientificType),(leisure, leisureType)
- scientificType → (book, booktype1)*, (journal, journalType)*
- leisureType → (book, booktype2)*, (periodical, periodicalType)*
- bookType1 → (title, titleT), (authors, authorsT)$^+$ | (editors, editorsT)$^+$, (publisher, publisherT), (year, yearT), (abstract, abstractT)?, (edition, editionT)?, (ISBN, ISBNT)?
- journalType → (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)
- bookType2 → (title, titleT), (authors, authorsT), (publisher, publisherT), (genre, genreT), (edition, editionT)?, (number, numberT)?
- periodicalType → (title, titleT), (price, priceT), (publisher, publisherT)