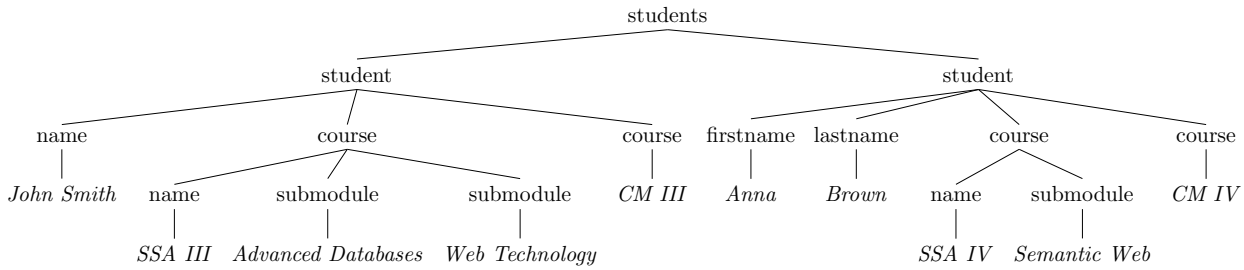
Advanced Databases

ffgt86

February 12, 2020

Part A

a) *Draw the directed tree structure of* `students.xml`.



b) *Write a DTD for* teachers.xml.

This DTD was validated using the `lxml` Python package. The provided file, `teachers.xml`, was modified slightly to facilitate this.

```
<?xml encoding="UTF-8"?>
<!ELEMENT teachers (teacher*)>
    <!ELEMENT teacher (name, course*)>
    <!ATTLIST teacher
        jobRole CDATA #REQUIRED
        joiningDate CDATA #REQUIRED>
    <!ELEMENT course (name,submodule+)>
        <!ELEMENT submodule (name,year+)>
            <!ELEMENT year (#PCDATA)>
            <!ELEMENT name (#PCDATA)>
```

However, with limited examples, and no formal specification, there are ambiguities:

- Should a teacher teach at least one course to be included in `teachers.xml`, i.e. `<!ELEMENT teacher (name, course+)>`?

- Should a course contain at least one submodule to be included in `teachers.xml`, i.e. `<!ELEMENT course (name, submodule+)>`?
- Which attributes (if any) are `#REQUIRED`, `#IMPLIED`, or `#FIXED`? Are there default values?
- Are there are limited number of values for `jobRole`? If there were only two roles available, for instance, `Professor` and `Researcher`, the attribute declaration should be `<!ATTLIST teacher jobRole(Professor | Researcher)>`.

c) *Is it possible to write a DTD for* `students.xml`?

It is not possible. The difficult construct is `<course>`. A student needs one or more courses (`course+`) but a course can either be `#PCDATA` or `(name, submodule*)`, and DTD does not allow mixed delcarations such as `<!ELEMENT course (#PCDATA | (name, submodule*)>`. This problem could be eliminated by forbidding `#PCDATA` in `<course>` and always using the `<name>` element, i.e. by changing:

```
<student enrolmentDate="2016">
        <firstname>Anna</firstname>
        <lastname>Brown</lastname>
        <course>
            <name>Software, Systems and Applications IV</name>
            <submodule>Semantic Web</submodule>
        </course>
        <course>Computing Methodologies IV</course>
    </student>
```

to:

```
<student enrolmentDate="2016">
        <firstname>Anna</firstname>
        <lastname>Brown</lastname>
        <course>
            <name>Software, Systems and Applications IV</name>
            <submodule>Semantic Web</submodule>
        </course>
        <course>
            <name>Computing Methodologies IV</name>
        </course>
    </student>
```

The problematic DTD element would then be `<!ELEMENT course (name, submodule*)>`

Part B

`XPath` queries were tested using the `lxml` Python package. The provided files, `teachers.xml` and `students.xml`, were modified slightly to facilitate this, and are included in the appendices. `XQuery` queries were tested using an online tool.[1]

a) *Find all students who study "Advanced Databases" this year.*

```
/students[@year='2019-2020']/descendant::submodule[text()='Advanced Databases']
/ancestor::student
```

1. Select the `<students>` node with a `year` attribute with value `'2019-2020'` (i.e. this year). There is no need to use the `year` attribute, as there can be only one root element in `students.xml`, but it is included here regardless.

   ```
   /students[@year='2019-2020']
   ```

2. Select all `<submodule>` descendants from step 1) with a text value of 'Advanced Databases':

   ```
   /descendant::submodule[text()='Advanced Databases']
   ```

3. Return the `<student>` ancestors from step 2):

   ```
   /ancestor::student
   ```

The output is:

```
<student enrolmentDate="2017">
        <name>John Smith</name>
        <course>
            <name>Software, Systems and Applications III</name>
            <submodule>Advanced Databases</submodule>
            <submodule>Web Technology</submodule>
        </course>
        <course>Computing Methodologies III</course>
    </student>
```

This approach is robust. Changes to the XML structure, short of removing the `students` or `submodule` elements, are handled by using `descendant` and `ancestor`, rather than hard-coding paths.

[1] http://videlibri.sourceforge.net/cgi-bin/xidelcgi

b) *Find all teachers who teach "Advanced Databases" this year.*

```
/teachers/descendant::submodule[name='Advanced Databases' and year='2019-2020']
/ancestor::teachers
```

1. Select all `<submodule>` descendants of `<teachers>` with nodes `<name>` and `<year>` with values 'Advanced Databases' and '2019-2020' (i.e. this year), respectively:

   ```
   /teachers/descendant::submodule[name='Advanced Databases' and year='2019-2020']
   ```

2. Return the `<teacher>` ancestors from step 1:

   ```
   /ancestor::teachers
   ```

The output is:

```
<teacher joiningDate="2018" jobRole="Professor">
        <name>Alexandra Cristea</name>
        <course>
            <name>Software, Systems and Applications III</name>
            <submodule>
                <name>Advanced Databases</name>
                <year>2018-2019</year>
                <year>2019-2020</year>
            </submodule>
        </course>
        <course>
            <name>Software, Systems and Applications IV</name>
            <submodule>
                <name>Semantic Web</name>
                <year>2018-2019</year>
                <year>2019-2020</year>
            </submodule>
        </course>
    </teacher>
```

This approach is similarly robust to part *a*).

c) *How many years has Professor Cristea been teaching "Advanced Databases" (at Durham)?*

```
xquery version "3.0";

for $t in /teachers/teacher
where $t/name='Alexandra Cristea'

for $s in $t/course/submodule
where $s/name='Advanced Databases'

return count($s/year)
```

1. Select teachers called 'Alexandra Cristea':

   ```
   for $t in /teachers/teacher
   where $t/name='Alexandra Cristea'
   ```

2. Select submodules called 'Advanced Databases' taught by `$t` (i.e. Alexandra Cristea):

   ```
   for $s in $t/course/submodule
   where $s/name='Advanced Databases'
   ```

3. Return the number of years the submodule `$s` has been taught:

   ```
   return count($s/year)
   ```

The output of this query is 2. If there exists more than one teacher with the name 'Alexandra Cristea', or more than one submodule with the name 'Advanced Databases', this query will sum results. In practice, a teacher or submodule would likely be selected by UID, rather than name, so this should not be an issue. An equivalent `XPath` query can be constructed:

```
count(/teachers/teacher[name='Alexandra Cristea']/course
/submodule[name='Advanced Databases']/year)
```

d) *Find all students in year 3 currently taught by Alexandra.*

```
xquery version "3.0";

for $s in /teachers/teacher[name='Alexandra Cristea']/course/submodule/name

for $k in /students/student[@enrolmentDate='2016']
where $s=$k/course/submodule

return $k
```

1. Select the names of all submodules `$s` taught by 'Alexandra Cristea':

   ```
   for $s in /teachers/teacher[name='Alexandra Cristea']/course/submodule/name
   ```

2. Return all students `$k` who enrolled in 2016 (and are thus in their third year) and join `$s` with `$k` on `$k/course/submodule`. This join only selects students who study submodules that 'Alexandra Cristea' teaches.

   ```
   for $k in /students/student[@enrolmentDate='2016']
   where $s=$k/course/submodule

   return $k
   ```

The outcome of this query is:

```
<student enrolmentDate="2016">
        <firstname>Anna</firstname>
        <lastname>Brown</lastname>
        <course>
            <name>Software, Systems and Applications IV</name>
            <submodule>Semantic Web</submodule>
        </course>
        <course>Computing Methodologies IV</course>
    </student>
```

The same issues as listed in part *c*) affect this query. Additionally, it is not clear whether the `<course>` elements in `<student>` contain courses studied *this year* or in previous years. If the latter is the case, the query becomes significantly more complex. No equivalent `XPath` query can be constructed, as there is no mechanism to 'join' documents, as is used here in step 2.

e) *How many teachers and how many students are kept in the databases where the last name is not known?*

This question is ambiguous:

1. Does it refer to teachers and students with no `<lastname>` element, those with only a single name in a `<name>` element, or both?. The former is assumed.
2. Does it require *two* results, one the number of teachers missing last names, and one the number of students missing last names, or their sum? The latter is assumed.

```
xquery version "3.0";

for $t in /teachers/teacher
where not(exists($t/lastname))

for $s in /students/student
    where not(exists($s/lastname))

return count($t) + count($s)
```

1. Select teachers `$t` without a `<lastname>` element:

   ```
   for $t in /teachers/teacher
   where not(exists($t/lastname))
   ```

2. Select students `$s` without a `<lastname>` element:

   ```
   for $t in /teachers/teacher
   where not(exists($t/lastname))
   ```

3. Return the sum of the number of elements in `$t` and `$s`:

   ```
   return count($t) + count($s)
   ```

No *single* equivalent XPath query can be constructed, as it is not possible to query two different documents in one query, but the results from two could easily be summed.

Appendices

The modified `students.xml`

```
<?xml version='1.0' standalone='yes'?>
<students year="2019-2020">
    <student enrolmentDate="2017">
        <name>John Smith</name>
        <course>
            <name>Software, Systems and Applications III</name>
            <submodule>Advanced Databases</submodule>
            <submodule>Web Technology</submodule>
        </course>
        <course>Computing Methodologies III</course>
    </student>
    <student enrolmentDate="2016">
        <firstname>Anna</firstname>
        <lastname>Brown</lastname>
        <course>
            <name>Software, Systems and Applications IV</name>
            <submodule>Semantic Web</submodule>
        </course>
        <course>Computing Methodologies IV</course>
    </student>
</students>
```

The modified `teachers.xml`

```
<?xml version="1.0" standalone="no"?>
<teachers>
    <teacher joiningDate="2018" jobRole="Professor">
        <name>Alexandra Cristea</name>
        <course>
            <name>Software, Systems and Applications III</name>
            <submodule>
                <name>Advanced Databases</name>
                <year>2018-2019</year>
                <year>2019-2020</year>
            </submodule>
        </course>
        <course>
            <name>Software, Systems and Applications IV</name>
            <submodule>
                <name>Semantic Web</name>
                <year>2018-2019</year>
                <year>2019-2020</year>
            </submodule>
        </course>
    </teacher>
</teachers>
```