Exercises: Assignment

Basi di dati 2

Disheng Qiu
disheng.qiu@gmail.com

Luca Rossi
luca.rossi.917@gmail.com

Exercises:

2x From instance to DTD

2x From instance to XSD

3x From specs to XPath

2x From specs to XQuery

Document 1: Student

```
<?xml version="1.0" encoding="UTF-8"?>
<student>
        <firstName>Luca</firstName>
        <lastName>Rossi</lastName>
        <id>281283</id>
        <plan>
                <courses year="3">
                        <course>
                                <name> Programmazione Orientata agli Oggetti </name>
                                <shortName>POO</shortName>
                                <record>
                                        <grade>30</grade>
                                        <date>13/06/11</date>
                                </record>
                        </course>
                        <course>
                                <name>Analisi e progettazione del software</name>
                                <shortName>APS</shortName>
                        </course>
                </courses>
        </plan>
</student>
```

Document 2: Email

```
<?xml version="1.0" encoding="UTF-8"?>
<email>
        <from> luca.rossi.917@gmail.com </from>
        <to> atzeni@dia.uniroma3.it </to>
        <content>
                Dear <person> Paolo </person>,
                here are some very hard exercises for the upcoming assignment of <course> Basi di Dati 2 </course>:
                <exercises>
                        <exercise>
                                <topic> DTD </topic>
                                <description> From Instance to DTD </description>
                        </exercise>
                        <exercise>
                                <topic> XPath </topic>
                                <description> Find students with average grade better than 26 </description>
                        </exercise>
                </exercises>
                Best Regards,
                <person> Luca </person>
        </content>
</email>
```

Exercise 1

Define a **DTD** that validates **Document 1**

Exercise 1: Solution

```
<!DOCTYPE student [
   <!ELEMENT student (firstName, lastName, id, plan)>
   <!ELEMENT firstName (#PCDATA)>
   <!ELEMENT lastName (#PCDATA)>
   <!ELEMENT id (#PCDATA)>
   <!ELEMENT plan (courses*)>
   <!ELEMENT courses (course*)>
   <!ATTLIST courses year CDATA #REQUIRED>
   <!ELEMENT course (name, shortName, record?)>
   <!ELEMENT record (grade, date)>
   <!ELEMENT name (#PCDATA)>
   <!ELEMENT shortName (#PCDATA)>
   <!ELEMENT grade (#PCDATA)>
   <!ELEMENT date (#PCDATA)>
]>
```

Exercise 2

Define a **DTD** that validates **Document 2**

Exercise 2: Solution

```
<!DOCTYPE email [
   <!ELEMENT email (from, to, content)>
   <!ELEMENT from (#PCDATA)>
   <!ELEMENT to (#PCDATA)>
   <!ELEMENT content (#PCDATA|person|exercises|course)*>
   <!ELEMENT exercises (exercise*)>
   <!ELEMENT exercise (topic, description)>
   <!ELEMENT topic (#PCDATA)>
   <!ELEMENT description (#PCDATA)>
   <!ELEMENT person (#PCDATA)>
   <!ELEMENT course (#PCDATA)>
]>
```

Exercise 3

Define an **XSD** that validates **Document 1**

Exercise 3: Solution

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:element name="student">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="firstName" type="xsd:string"/>
            <xsd:element name="lastName" type="xsd:string"/>
            <xsd:element name="id" type="xsd:ID"/>
            <xsd:element ref="plan"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
   …
```

Exercise 3: Solution

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:element name="student">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="firstName" type="xsd:string"/>
            <xsd:element name="lastName" type="xsd:string"/>
            <xsd:element name="id" type="xsd:ID"/>
            <xsd:element ref="plan"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
   …
```

Exercise 3: Solution

```
<xsd:element name="record">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="grade" type="xsd:string"/>
            <xsd:element name="date" type="xsd:date"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="course">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="name" type="xsd:string"/>
            <xsd:element name="shortName" type="xsd:string"/>
            <xsd:element ref="record" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

Exercise 3: Solution

```
<xsd:element name="courses">
   <xsd:complexType>
      <xsd:sequence minOccurs="0" maxOccurs="unbounded">
         <xsd:element ref="course"/>
      </xsd:sequence>
      <xsd:attribute name="year" type="xsd:string"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="plan">
   <xsd:complexType>
      <xsd:sequence minOccurs="0" maxOccurs="unbounded">
         <xsd:element ref="courses"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:unique name="coursesYear">
      <xsd:selector xpath="courses"/>
      <xsd:field xpath="@year"/>
   </xsd:unique>
</xsd:element>
```

Exercise 3: Solution

```
<xsd:element name="courses">
   <xsd:complexType>
      <xsd:sequence minOccurs="0" maxOccurs="unbounded">
         <xsd:element ref="course"/>
      </xsd:sequence>
      <xsd:attribute name="year" type="xsd:string"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="plan">
   <xsd:complexType>
      <xsd:sequence minOccurs="0" maxOccurs="unbounded">
         <xsd:element ref="courses"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:unique name="coursesYear">
      <xsd:selector xpath="courses"/>
      <xsd:field xpath="@year"/>
   </xsd:unique>
</xsd:element>
```

Exercise 4

Define an **XSD** that validates **Document 2**

Exercise 4: Solution

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:element name="email">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="from" type="xsd:string" />
            <xsd:element name="to" type="xsd:string" />
            <xsd:element name="content">
               ....
            </xsd:element>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

Exercise 4: Solution

```
<xsd:complexType mixed="true">
  <xsd:sequence>
    <xsd:element name="person" type="xsd:string"/>
    <xsd:element name="course" type="xsd:string"/>
    <xsd:element name="exercises">
      <xsd:complexType>
        <xsd:sequence minOccurs="0" maxOccurs="unbounded">
          <xsd:element name="exercise">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="topic" type="xsd:string"/>
                <xsd:element name="description" type="xsd:string"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="person" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

Exercise 4

From specs to **XPath**

Exercise 4: Solution

1) **Find students who have passed POO with grade > 24**
2) Find students with average grade > 26
3) Find students who have passed more exams in the 2nd year than in the 3rd

Exercise 4: Solution

1) **Find students who have passed POO with grade > 24**

```
//student[.//course[shortName="POO" and record>24]]
```

2) Find students with average grade > 26
3) Find students who have passed more exams in the 2nd year than in the 3rd

Exercise 4: Solution

1) Find students who have passed POO with grade > 24
2) **Find students with average grade > 26**
3) Find students who have passed more exams in the 2nd year than in the 3rd

Exercise 4: Solution

1) Find students who have passed POO with grade > 24
2) **Find students with average grade > 26**

   //student[avg(.//grade)>26]
3) Find students who have passed more exams in the 2nd year than in the 3rd

Exercise 4: Solution

1) Find students who have passed POO with grade > 24
2) Find students with average grade > 26
3) **Find students who have passed more exams in the 2nd year than in the 3rd**

Exercise 4: Solution

1) Find students who have passed POO with grade > 24
2) Find students with average grade > 26
3) **Find students who have passed more exams in the 2nd year than in the 3rd**

```
//student[count(.//courses[@year="3"]/course) >
count(.//courses[@year="2"]/course)]
```

Exercise 5

From specs to **XQuery**

Exercise 5: Solution

1) **For each student, return distinct couples of exams passed with the same grade**
2) Return students who improved their average grade in the 3rd year with respect to the 2nd.

Exercise 5: Solution

1) For each student, return distinct couples of exams passed with the same grade

```
xquery version "1.0";
for $s1 in fn:doc("student.xml")//student
for $c1 in $s1//course
for $c2 in $s1//course
where $c2 >> $c1 and $c1//grade = $c2//grade
return
<coppiaCorsi>
   <corso>
      <name>{$c1/name}</name>
      <grade>{$c1//grade}</grade>
   </corso>
   <corso>
      <name>{$c2/name}</name>
      <grade>{$c2//grade}</grade>
   </corso>
</coppiaCorsi>
```

Exercise 5: Solution

1) For each student, return distinct couples of exams passed with the same grade
2) **Return students who improved their average grade in the 3rd year with respect to the 2nd.**

Exercise 5: Solution

1) For each student, return distinct couples of exams passed with the same grade
2) **Return students who improved their average grade in the 3rd year with respect to the 2nd.**

```
xquery version "1.0";
for $s in fn:doc("student.xml")//student
let $avg3 := fn:avg($s//courses[@year="3"]//grade)
let $avg2 := fn:avg($s//courses[@year="2"]//grade)
where $avg3 > $avg2
return
<students>
   <student>
      {$s/id}
      <avg year="3">{$avg3}</avg>
      <avg year="2">{$avg2}</avg>
   </student>
</students>
```