Big Data – Exercises

Fall 2021 – Week 6 – ETH Zurich

Data Models

Reading:

- (Mandatory) Melnik, S., et al. (2011). Dremel: Interactive Analysis of Web-Scale Datasets. In CACM, 54(6). [DOI]
- (Recommended) M. Droettboom, Understanding JSON Schema [online]
- (Recommended) Harold, E. R., & Means, W. S. (2004). XML in a Nutshell. [Available in the ETH library] [online] (Chapter 17 on XML Schema, except 17.3 on namespaces)
- (Optional) White, T. (2015). Hadoop: The Definitive Guide (4th ed.). O'Reilly Media, Inc. [Available in the ETH library] [online] (Chapters 12 (Avro) and 13 (Parquet)

This exercise will consist of three main parts:

- XML data models
- XML schemas
- JSON Schemas
- JSound
- Dremel

1. XML Data Models – Information Sets

XML "Information Set" provides an abstract representation of an XML document—it can be thought of as a set of rules on how one would draw an XML document on a whiteboard.

An XML document has an information set if it is well-formed and satisfies the namespace constraints. There is no requirement for an XML document to be valid in order to have an information set. An information set can contain up to eleven different types of information items, e.g., the document information item (always present), element information items, attribute information item, etc.

Task 1.1

Draw the Information Set trees for the following XML documents. You can confine your trees to only have the following types of information items: *document information item, elements, character information items, and attributes.*

Document 1

```
<Burger>
    <Bun>
        <Pickles/>
        <Cheese origin="Switzerland" />
        <Patty/>
    </Bun>
</Burger>
```

Document 2

```
<catalog>
   <!-- A list of books -->
      <author>Gambardella, Matthew</author>
      <title>XML Developer's Guide</title>
      <genre>Computer</genre>
      <price>44.95</price>
      <publish_date version='hard' version2='soft'>2000-10-01</publish_date>
   </book>
</catalog>
```

Document 3

```
<eth date="11.11.2006">
   <date>16.11.2017</date>
   <president since="2020">Prof. Dr. Joël Mesot</president>
   <rector>Prof. Dr. Sarah M. Springman</rector>
</eth>
```

2. XML Schemas

In this task we will explore XML Schemas in detail. An XML Schema describes the structure of an XML document.

The purpose of an XML Schema is to define the legal building blocks of an XML document:

- the elements and attributes that can appear in a document
- the number of (and order of) child elements
- data types for elements and attributes
- default and fixed values for elements and attributes

When you open an XML Schema in oXygen, you can switch to its graphical representation, by choosing the "Design" mode at the bottom of the document pane; "Text" mode shows the XML Schema as an XML document.

Task 2.1

Match the following XML documents to XML Schemas that will validate them. Match them manually then validate with oXygen.

Document 1

```xml
<happiness xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation="Schema.xsd"/>
```

Document 2

```xml
<happiness xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation="Schema.xsd">
    <health/>
    <friends/>
    <family/>
</happiness>
```

Document 3

```xml
<happiness xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation="Schema.xsd">
    3.141562
</happiness>
```

Document 4

```xml
<happiness xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation="Schema.xsd">
    <health value="100"/>
    <friends/>
    <family/>
</happiness>
```

Document 5

```xml
<happiness xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation="Schema.xsd">
    <health/>
    <friends/>
    <family/>
    But perhaps everybody defines it differently...
</happiness>
```

Schema 1

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="happiness">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="health"/>
                <xs:element name="friends"/>
                <xs:element name="family"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Schema 2

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="happiness">
        <xs:complexType mixed="true">
            <xs:sequence>
                <xs:element name="health"/>
                <xs:element name="friends"/>
                <xs:element name="family"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Schema 3

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="happiness" type="xs:decimal"/>
</xs:schema>
```

Schema 4

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="happiness">
        <xs:complexType>
            <xs:sequence/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Schema 5

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="happiness">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="health">
                    <xs:complexType>
                        <xs:attribute name="value" type="xs:integer"
use="required"/>
                    </xs:complexType>
                </xs:element>
                <xs:element name="friends"/>
                <xs:element name="family"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Task 2.2

The Great Language Game is a game in which you are given a voice clip to listen, and you are asked to identify the language in which the person was speaking. It is a multiple-choice question–you make your choice out of several alternatives.

The following XML document presents a user's attempt at answering a single question in the game: it contains the identifier of the voice clip, the choices presented to the player, and the player's response.
Provide an XML Schema which will validate this document:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<attempt country="AU" date="2013-08-19">
    <voiceClip>48f9c924e0d98c959d8a6f1862b3ce9a</voiceClip>
    <choices>
        <choice>Maori</choice>
        <choice>Mandarin</choice>
        <choice>Norwegian</choice>
        <choice>Tongan</choice>
    </choices>
    <target>Norwegian</target>
    <guess>Norwegian</guess>
</attempt>

```

Task 2.3

Continuing the topic of the Great Language Game, provide an XML Schema which will validate the following document:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<attempts>
    <attempt country="AU" date="2013-08-19">
        <voiceClip>48f9c924e0d98c959d8a6f1862b3ce9a</voiceClip>
        <choices>
            <choice>Maori</choice>
            <choice>Mandarin</choice>
            <choice>Norwegian</choice>
            <choice>Tongan</choice>
        </choices>
        <target>Norwegian</target>
        <guess>Norwegian</guess>
    </attempt>
    <attempt country="US" date="2014-03-01">
        <voiceClip>5000be64c8cc8f61dda50fca8d77d307</voiceClip>
        <choices>
            <choice>Finnish</choice>
            <choice>Mandarin</choice>
            <choice>Scottish Gaelic</choice>
            <choice>Slovak</choice>
            <choice>Swedish</choice>
            <choice>Thai</choice>
        </choices>
        <target>Slovak</target>
        <guess>Slovak</guess>
    </attempt>
    <attempt country="US" date="2014-03-01">
        <voiceClip>923c0d6c9e593966e1b6354cc0d794de</voiceClip>
        <choices>
            <choice>Hungarian</choice>
            <choice>Sinhalese</choice>
            <choice>Swahili</choice>
        </choices>
        <target>Hungarian</target>
        <guess>Sinhalese</guess>
    </attempt>
</attempts>
```

Task 2.4

Let us now solve the reverse problem. Given the following XML Schema, provide a valid instance document:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="movies">
        <xs:complexType>
            <xs:sequence maxOccurs="unbounded" minOccurs="0">
                <xs:element name="Movie">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="title" type="xs:string"/>
                            <xs:element name="year" type="xs:gYear"/>
                            <xs:element name="_director">
                                <xs:complexType>
                                    <xs:sequence/>
                                    <xs:attribute name="name" type="xs:string"/>
                                </xs:complexType>
                            </xs:element>
                            <xs:choice minOccurs="1" maxOccurs="unbounded">
                                <xs:element name="comment">
                                    <xs:complexType>
                                        <xs:simpleContent>
                                            <xs:extension base="xs:string">
                                                <xs:attribute name="lang"
type="xs:string"/>
                                            </xs:extension>
                                        </xs:simpleContent>
                                    </xs:complexType>
                                </xs:element>
                                <xs:element name="newcomment" type="xs:string"/>
                            </xs:choice>
                        </xs:sequence>
                        <xs:attribute name="id" type="xs:ID"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

3. JSON Schemas

JSON Schema is a vocabulary that allows you to annotate and validate JSON documents. It is used to:

- Describe your existing data format(s).
- Provide clear human- and machine- readable documentation.
- Validate data, i.e., automated testing, ensuring quality of client submitted data.

Task 3.1

Provide an JSON Schema which will validate the following document.

```
{
  "firstName": "John",
  "lastName": "Doe",
  "age": 21
}
```

Provide an JSON Schema which will validate the following document.
The JSON Schema has to check for the following properties:

- The price of a product has to be strictly positive.
- Tags are describing the product and necessary for a proper product description. We need at least one tag per product and each tag should be unique.
- The "productId", "productName" and the "price" should always be contained in a valid JSON document.

```
{
  "productId": 1,
  "productName": "An ice sculpture",
  "price": 12.50,
  "tags": [ "cold", "ice" ],
  "dimensions": {
    "length": 7.0,
    "width": 12.0,
    "height": 9.5
  }
}
```

Task 3.3

Based on the given Json schema, can you give an instance of it?
HINT: We defined array of things.

```
{
  "$id": "https://example.com/arrays.schema.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "description": "A representation of a person, company, organization, or place",
  "type": "object",
  "properties": {
    "fruits": {
      "type": "array",
      "items": {
        "type": "string"
      }
    },
    "vegetables": {
      "type": "array",
      "items": { "$ref": "#/definitions/veggie" }
    }
  },
  "definitions": {
    "veggie": {
      "type": "object",
      "required": [ "veggieName", "veggieLike" ],
      "properties": {
        "veggieName": {
          "type": "string",
          "description": "The name of the vegetable."
        },
        "veggieLike": {
          "type": "boolean",
          "description": "Do I like this vegetable?"
        }
      }
    }
  }
}
```

4. JSound

JSound is a vocabulary that allows you to validate JSON documents. It employs a very simple and intuitive JSON-like synthax.

Task 4.1

Repeat the exercise in 3.1 but now produce a a JSound schema which will validate the following document.

```
{
  "firstName": "John",
  "lastName": "Doe",
  "age": 21
}
```

Task 4.2

Build a valid JSON document based on the following JSound schema.

```
{
    "id": "integer",
    "who": [{
        "name": "string",
        "type": "string",
        "preferred": "boolean"
    }],
    "year_of_birth": "integer",
    "living": "boolean"
}
```

5. Document Validation

XML, JSON, and other document formats have many different tools for validation.

Associate each document type with the type of schema that can be used to validate it:

```
Document Type:                                  Schema Type:

A. XML                                          1. XML Schema
B. JSON                                         2. DTD
C. Protocol buffers                             3. Schematron
D. XHTML                                        4. RelaxNG
                                                5. JSON Schema
                                                6. JSound
                                                7. Kwalify
                                                8. The XML Schema for XHTML
documents
                                                9. proto3
                                                10. No schema at all
```

6. Dremel

Dremel is a query system developed at Google for deriving data stored in a nested data format such as XML, JSON, or Google Protocol Buffers into column storage, where it can be analyzed faster.

For this exercise, you will apply the algorithm described in the Dremel paper to convert from a nested data format with an associated schema into column storage.

You are given the following Google Protocol Buffer schema:

```
message Catalog {
    repeated group Book {
        required string Title;
        repeated string Author;
        optional uint32 Year;
        repeated group Version {
            required string Type;
            optional uint32 Pages;
        }
    }
}
```

Use this to convert the following message to column storage:

```
Book
    Title: "Because Internet"
    Author: "McCulloch, Gretchen"
    Year: 2019
    Version
        Type: "soft"
Book
    Title: "Hitchhiker's Guide to the Galaxy"
    Author: "Adams, Douglas"
    Year: 1979
Book
    Title: "XML in a Nutshell"
    Author: "Harold, Elliotte Rusty"
    Author: "Means, Scott W."
    Year: 2005
    Version
        Type: "soft"
        Pages: 718
```

Fill in each of the following tables storing the data for each leaf field, where each row contains the value, the repetition levels, and the definitions levels for the given entry.

HINTS: the definitions of repetition levels and definitions levels are presented in the section 4.1 in the Dremel paper

Catalog.Book.Title

Catalog.Book.Author

Catalog.Book.Year

Catalog.Book.Version.Type

Catalog.Book.Version.Pages