# Exercise Sheet 1 (SS 2021)

**3.0 VU Semistructured Data**

## Information on the Exercise Sheet

### General

In the first exercise sheet you will develop a schema for a file format. First, you will create an XML-Schema (XSD) and write a matching XML document. In the second part, you will convert the XML-Schema into a Document Type Definition (DTD).

Submit your solution as a single ZIP file (max. 5MB). The ZIP file should contain an XML-Schema, a DTD as well as fitting XML documents for the XML-Schema and the DTD. Thus, you should submit the following files:

- `vaccination-plan.xsd`
- `vaccination-plan-xsd.xml`
- `vaccination-plan.dtd`
- `vaccination-plan-dtd.xml`

The exercise sheet is made up of 5 tasks (described below). You can earn a total of 10 points.

### Deadline

| | | |
|---|---|---|
| **at the latest April 29th** | **12:00** | Upload your submission on TUWEL |
| **Please do not forget!** | $\Longrightarrow$ | Register for an exercise interview in TUWEL |

### Exercise Interviews

During the solution discussion, the correctness of your solution as well as your understanding of the underlying concepts will be assessed.

The scoring of your submission is primarily based on your performance at the solution discussion. Therefore, it is possible (in extreme cases) to receive 0 points although the submitted solution was technically correct.

Please, be punctual to your solution discussion. Otherwise, we cannot guarantee that your full solution can be graded in your assigned time slot. Remember to bring your student id to the solution discussion. It is not possible to assess your solution without an id.

### Changes due to COVID-19

Due to the ongoing situation with COVID-19 we will not offer in-person office hours for the exercise sheets. If you have technical issues, trouble understanding the tasks on this sheet, or other questions please use the TUWEL forum. We also recommend that you get involved in the forum and actively discuss with your colleagues on the forum. From experience we believe that this helps all parties in the discussion greatly to improve their understanding of the material.

### Further Questions – TUWEL Forum

If you have any further questions, regarding the organization or material, you can use the TUWEL forum.

## Exercises: XML Schema

First, you will write an XML schema for managing the COVID-19 vaccination plan. Save the created XML-Schema in the `vaccination-plan.xsd` file. In task 3, you will then be asked to create an XML document `vaccination-plan-xsd.xml` that matches the schema.

**Important:** Make sure that your schema file is well-formed and that your XML document is valid! If this is not the case you will receive 0 points for the associated tasks! If you have trouble implementing all aspects of the schema you still have to make sure that your schema is well-formed and the document is valid.

### Aufgabe 1 (Defining the Elements of `vaccination-plan.xsd`) *[4 Punkte]*

The XML-Schema shall validate XML documents with the following structure:

**Element `vaccination-plan`.** The root element `vaccination-plan` stores all the information of the vaccination plan and contains the following elements in *any order exactly once*:

- A `vaccine-types` element;
- a `vaccines` element;
- a `patients` element.

**Element `vaccine-types`.** The `vaccine-types` element has no attributes. This subtree stores all the information of various COVID-19 vaccine types. To do so, the `vaccine-types` element contains an arbitrary number of `vaccine` elements.

**Element `vaccine` (Child of `vaccine-types`).** A `vaccine` element contains the following required elements in any order: a `name` and a `type` stored as a string, as well as an `authorized` element stored as a boolean.

**Element `vaccines`.** This subtree stores all the information of ordered COVID-19 vaccines. To do so, the `vaccines` element contains an arbitrary number of `vaccine` elements.

**Element `vaccine` (Child of `vaccines`).** A `vaccine` element has a string attribute `type_ref`. Furthermore, the `vaccine` element contains *at least* one pair of the following elements in the specified order: a *necessary* `batch` element and an *optional* `info` element that holds information about the purchase of the corresponding batch if available. For instance, a valid example of the content of the `vaccine` element would be: "`batch`, `info`, `batch`, `batch`, `info`", where information about the order of the first and the last batch is available, while information about the order of the second batch is absent.

**Element `batch`.** The `batch` element contains a textual description of the batch. Additionally, the `batch` element has an attribute `id` that adheres to the following format:

It starts with one upper case letter, followed by two lower case letters and another upper case letter. Next, the `id` continues with a dash ("-"), followed by four digits (0 to 9) and followed by another dash. Finally, the `id` ends with a digit (0 to 9), followed by an upper case letter and another digit (0 to 9). For example: "AstZ-5587-8B5".

**Element `info`.** An `info` element has an attribute `date` that stores the date of its last modification. Furthermore, the `info` element stores information about the details of the purchased batch. To model this behaviour it contains a textual description mixed with the following elements in the specified order:

- one *necessary* `size` element that contains the number of ordered doses of the batch;
- one *necessary* `order-date` element that states the date when the batch has been ordered;
- and followingly an *arbitrary number* of pairs of `size` and `delivery-date` elements in this order, where the `delivery-date` may occur 1 to 3 times and states when the doses are going to be delivered.

An example for valid content of the `info` element is stated followingly:

"An order for `<size>160</size>` doses has been placed on `<order-date>2021-01-10</order-date>`. `<size>55</size>` doses will arrive on `<delivery-date>2021-02-01</delivery-date>`. Additionally, `<size>105</size>` doses will arrive on `<delivery-date>2021-02-20</delivery-date>` and `<delivery-date>2021-02-21</delivery-date>`."

**Element `patients`.** The `patients` element stores all necessary information of patients who shall be vaccinated. Therefore, it consists of an arbitrary number of `patient` elements.

**Element `patient`.** A `patient` element stores all important information about a patient and their vaccination status. Therefore, it contains the following elements in the specified order: A `risk-group` element, whose content can be exclusively one of the strings "High", "Medium" or "Low". Followed by, at a minimum 0 and at a maximum 2 pairs of `vaccine` and `vaccination-date` elements in this order. Finally, the content of this element ends with one `residences` element.

Furthermore, the `patient` element has the following attributes:

- A *required* `name`, describing the name of the patient.
- an *optional* `birth_year`, describing the year in which the patient has been born.
- a *required* `pid` that identifies the patient. It adheres to the following format: A `pid` starts with the letter "P" followed by at least one digit. For example: "P152847".

**Element `vaccine` (Child of `patient`).** A `vaccine` element is empty and has one string attribute `ref_batch`.

**Element `vaccination-date`.** A `vaccination-date` element is empty and has one attribute `date` that describes when the vaccine has been injected.

**Element `residences`.** A `residences` element contains an arbitrary amount of `second` elements and exactly one `main` element in any order (e.g. `second`, `second`, `main`, `second`). Both the `main` and `second` element contain solely a string that describes the address of the main or second residence of the patient respectively.

## Aufgabe 2 (Define keys and references `vaccination-plan.xsd`) *[2 Punkte]*

Add the following keys to your schema:

- A global key `vaccineTypeKey` over the `name` element of `vaccine` (child of `vaccine-types`) elements.
- A global key `batchKey` for the attribute `id` of `batch` elements.

- A global key `patientKeys` over `pid` of `patient` elements .

Now add the following key references to your schema:

- The `type_ref` attribute of `vaccine` (child of `vaccines`) elements references the `vaccineTypeKey`.
- The `ref_batch` attribute of `vaccine` elements (child of `patient`) references the `batchKey`.

Finally, add the following uniqueness constraint to your schema:

- Remember that each `patient` element contains a `residences` element, which contains arbitrary many `second` elements and exactly one `main` element. Make sure that *locally* in each such `residences` node, there are no duplicate entries in its child `main` and `second` elements. For instance, the following example would violate the constraint:

  ```
  <residences>
   <second>Vienna, AT</second>
   <main>Basel, CH</main>
   <second>Basel, CH</second>
  </residences>
  ```

  Additionally, also the second example below violates the described constraint:

  ```
  <residences>
   <main>Vienna, AT</main>
   <second>Basel, CH</second>
   <second>Basel, CH</second>
  </residences>
  ```

  But it is allowed that “Basel, CH” occurs as the child of different `residences` elements.

**Aufgabe 3 (Creation of the XML document** `vaccination-plan-xsd.xml`**)** *[1 Punkte]*

Create the XML document `vaccination-plan-xsd.xml` for the schema `vaccination-plan.xsd`. The XML document shall satisfy the following conditions:

- Create at least 4 `vaccine` elements (child of `vaccine-types`), employing all possible values of the `authorized` child element (i.e. `authorized` should contain at least once true and at least once false)
- Create at least 3 `vaccine` elements (child of `vaccines`).
- Each `vaccine` element (child of `vaccines`) shall have at least 2 `batch` elements. Furthermore, for each of these `vaccine` elements, at least one `batch` shall be followed by a valid `info` element.
- Furthermore, at least one `info` element shall contain 2 `delivery-date` elements in its textual content.
- Create at least 3 `patients`.
- Use each possible value of the `risk-group` element at least once.
- Additional, model that at least one `patient` element that belongs to the *high* `risk-group` has already received 2 vaccinations (using the `vaccine` (child of `patient`) and `vaccination-date` elements). Moreover, model that at least one `patient` element belonging to the *medium* `risk-group` has already received exactly one vaccination.

- Create in total at least 5 `second` elements (child of `residences`).
- Create one element per key reference of task 2.

Make sure that your XML-Schema `vaccination-plan.xsd` validates your `vaccination-plan-xsd.xml` document. You can check this via the following command (after installing `xmllint`):

```
xmllint --schema vaccination-plan.xsd vaccination-plan-xsd.xml
```

Instructions for downloading and installing `xmllint` can be found in TUWEL.

**Important:** If your XML document is not well formed and valid with regards to the schema, you will receive 0 points for this task!

## Document Type Definition

Now create a DTD for the above specification.

**Aufgabe 4 (Creating a DTD `vaccination-plan.dtd`)** *[2 Punkte]*

Create a Document Type Definition (DTD) `vaccination-plan.dtd`, which realizes the specification form Exercise 1 and 2 above. It is possible that parts of the specification are very complicated or impossible to implement in DTD. In that case make reasonable assumptions and adaptions to implement them as close as possible in the DTD.

In your exercise interview you have to be able to explain which parts are not fully realizable in a DTD (and why). **Important:** It is particularly important that you try to implement the keys and key references. On the other hand it is not necessary to explicitly implement large number ranges through explicit enumeration of all numbers (e.g., you do not have to create an enumeration of the numbers 1 through 72 to implement a "number between 1 and 72"). Enumerations that have a small range of up to 6 values should however be implemented fully in the DTD.

**Important:** If you submit a DTD with syntax errors, meaning it cannot be used for validation, you will receive 0 points for the task.

**Aufgabe 5 (Creating the XML document `vaccination-plan-dtd.xml`)** *[1 Punkte]*

If your previous XML document `vaccination-plan-xsd.xml` does not validate with your DTD, you will have to create an additional XML document `vaccination-plan-dtd.xml`, which contains the same data but validates for your DTD.

Make sure that your DTD `vaccination-plan.dtd` validates your XML document `vaccination-plan-dtd.xml`. You can check this using the following command (after installing `xmllint`):

```
  xmllint --dtdvalid vaccination-plan.dtd vaccination-plan-dtd.xml
```

**Important:** If your XML document is not well-formed or valid with regards to your DTD, you will receive 0 points for this task.