School of Engineering and Computer Science

# SWEN 432
# Advanced Database Design and Implementation

## Assignment 2

Due date: Friday 18 April at 23:59

The objective of this assignment is to test your understanding of:

- XML Schema,
- Identity constraints of XML Schema,
- XML functional dependencies according to the approach of Arenas and Libkin, and
- XML Normal Form.

The Assignment is worth 5.0% of your final grade. The Assignment is marked out of 100.

Submit your answers to assignment questions via the school electronic submission system and a printed version in the hand-in box in the second floor of the Cotton Building.

When doing the assignment you will need to use the `xmllint` XML parser to check whether your XML documents are valid with regard to an XML Schema schema.

You can use `xmllint` from UNIX shell. To do so, type

```
>xmllint
```

To get information how to use `xmllint`, type

```
>man xmllint
```

Include your commands and `xmllint` messages from the shell prompt in your answers.

The `xmllint` parser does not recognize the syntax of new elements (like `xsd:assert`) introduced in XSD 1.1. Pavle downloaded Xerces2 Java free XSD 1.1 parser and validator from:

https://jeszysblog.wordpress.com/2012/09/27/free-and-open-source-xsd-1-1-validation-tool/

and it does the job.

You may find the validator here:

`/vol/courses/swen432/xsd11-validator.jar`

You may run the validator using the command

```
%java -jar xsd11validator.jar -sf <xsd_schema_file> -if <xml_file>
```

You may also have realized that `xmllint` does not perform `IDREF` checking when validates a document against an XML Schema. Use Xerces2 Java instead, if you need to validate `ID/IDREF` constraints using XML Schema.

## Question 1. Mapping the `Boat_Hire_A2.dtd` schema into an XML Schema schema [15 marks]

There are the `Boat_Hire_A2_14.dtd` schema and the `Boat_Hire_A2_14.xml` document given at the Assignments course page. The document `Boat_Hire_A2_14.xml` is an instance of the `Boat_Hire_A2_14.dtd` schema.

- **[10 marks]** Map the `Boat_Hire_A2_14.dtd` schema into an XML Schema schema with no loss of any structuring information or constraints. Call it `Boat_Hire_A2_a_14.xsd`. The `Boat_Hire_A2_14.xml` should validate with `Boat_Hire_A2_a_14.xsd`. Use constraints of `ID/IDREF` type in your `Boat_Hire_A2_a_14.xsd`.
- **[5 marks]** According to the `Boat_Hire_A2_14.xml`, the reservation of the boat b313 made by the sailor s111 starts at 6:00 pm and finishes at 4:30 pm on the same date (17.03.2014.). Make a constraint that will prevent such errors to happen. Call the new schema `Boat_Hire_A2_b_14.xsd`.
  **Note:** `xmllint` does not recognize the XML Schema 1.1 syntax. To perform validation of constraint above you need to use an appropriate XML Schema 1.1 validator.

## Question 2. `Boat_Hire_A2_b.xsd` Identity Constraints [25 marks]

Use the `Boat_Hire_A2_a_14.xsd` schema you have produced in the question 1 above to define the identity constraints. Change the type of all `ID/IDREF` attributes to `xsd:string`. Call the new schema `Boat_Hire_A2_IC_14.xsd`.

Define:

**a) [2 marks]** The attribute `SailorId` as a key of `Sailor` within the scope of all `Sailor` elements.

**b) [2 marks]** The attribute `GradeId` of `Grade` as a key of `Grade` within the scope of all `Grade` elements.

**c) [2 marks]** The attribute `ref_grade` of `Sailor` as a constraint referencing `Grade`.

**d) [2 marks]** The attribute `ref_sail` of `Res_Sailor` as a constraint referencing `Sailor`.

**e) [2 marks]** The attribute `MarinaId` of `Marina` as a key of `Marina` within the scope of all `Marina` elements.

**f) [2 marks]** The attribute `Number` of `Boat` as a key of `Boat` within the scope of all `Boat` elements.

**g) [4 marks]** A constraint that will enforce the rule that each boat within a marina has to have a unique name.

**h) [4 marks]** A constraint that will enforce the rule that each sailor (as a `Res_Sailor`) is not allowed reserving the same boat on the same date more than once.

**i) [5 marks]** All `Boat` elements have a unique (and not null) `@Number` attribute. All `Reserves` elements of a boat have unique (and not null) `@date` attribute. Can you make a constraint that will act as a key of `Reserves` within the scope of all `Reserves` elements using `@Number` and `@date` attributes? If you think you can, show how. If you think you can not, explain why not.

## Question 3. Comparing `ID/IDREF` and Identity Constraints [20 marks]

In this question, you are asked to analyze, and discuss differences in checking abilities of `ID/IDREF` and Identity Constraint mechanisms. To answer the question, you will use the following files:

- `Boat_Hire_A2_14.dtd`,
- `Boat_Hire_A2_IC_14.xsd` schema (you have produced in the question 2 above), and
- `Boat_Hire_A2_WRONG_a_14.xml` (given on the course Assignments page).

You will also need to make a new XML Schema. Call the new XML Schema `Boat_Hire_A2_Q3_14.xsd`. Copy your `Boat_Hire_A2_IC_14.xsd` into `Boat_Hire_A2_Q3_14.xsd`.

The XML document `Boat_Hire_A2_WRONG_a_14.xml` validates against the schema `Boat_Hire_ A2_14.dtd` and `xmllint` fails to report any errors, although the document `Boat_Hire_A2_WRONG_a_14.xml` contains a number of them.

**a) [8 marks]** Use XML Schema schema `Boat_Hire_A2_IC_14.xsd` to validate the document `Boat_Hire_A2_WRONG_a_14.xml`:

- **i.** [6 marks] In your answer, show the errors detected by schema identity constraints and reported by `xmllint`.
- **ii.** [2 marks] If you think the errors can be detected using `ID/IDREF` mechanisms, show how. If you think the errors can't be detected using `ID/IDREF` mechanisms, explain why not. Comment each error separately.

**b) [6 marks]** For safety reasons, marine authorities require that a sailor making a boat reservation has to have a grade S (skipper). The requirement is encoded in the `Boat_Hire_A2_14.dtd` using a `#FIXED` attribute. But this way, the opportunity to check whether a sailor is really a skipper has not been obtained.

**i.** [4 marks] Extend your `Boat_Hire_A2_Q3.xsd` by adding identity constraints that will detect errors of the considered type.

**ii.** [2 marks] If you think the errors of the considered type can be detected using `ID/IDREF` mechanisms, show how. If you think the errors can't be detected using `ID/IDREF` mechanisms, explain why not.

**c) [6 marks]** Assume you want to define a new simple type for the `SailorId` attribute that will restrict its values to a pattern consisting of two lower case "sa" letters followed by three digits (e.g. "sa007").

**i.** [2 marks] Can you make this extension within the `Boat_Hire_A2_a_14.xsd` schema and still use the `SailorId` as an `ID` type attribute? Justify your answer.

**ii.** [4 marks] Extend your `Boat_Hire_A2_Q3_14.xsd` schema by defining a simple type that will restrict the `SailorId` attribute values to a pattern consisting of two lower case letters "sa" followed by three digits. Note, lower case letters should be exactly "s" and "a".

## Question 4. XML Functional Dependencies [20 marks]

In this question you are asked to demonstrate your understanding of the formal approach to XML defined by Arenas and Libkin.

There are an XML document `Train_TimeTable_A2_14.xml` and `Train_TimeTable_A2_14.dtd` given on the course Assignments page. Note that comments next to each element in the `Train_TimeTable_A2_14.xml` document contain node identifiers. Use these identifiers when answering the following questions. The document is valid with regard to the DTD.
There is a graphical representation of a part of the `Train_TimeTable_A2_14.xml` document given on the Course Assignemtnts page. The Table 1 (given on the next page) contains the set of `Train_TimeTable_A2_14.dtd` paths and five tree tuples of the `TrainTimeTable_A2_14.xml` document. You may find useful to consider the graphical representation and the Table 1 when answering the following questions.

**The Set of Paths of the `Train_TimeTable_A2_14.dtd` schema and Maximal Tree Tuples of the `Train_TimeTable_A2_14.xml` Document**

| Path | $t_1$ | $t_2$ | $t_3$ | $t_4$ | $t_5$ |
|---|---|---|---|---|---|
| TrainTimeTable | &1 | &1 | &1 | &1 | &1 |
| TrainTimeTable.Line | &2 | &2 | &2 | &50 | &50 |
| TrainTimeTable.Line.@name | Hutt Valey | Hutt Valey | Hutt Valey | Waikanae | Waikanae |
| TrainTimeTable.Line.Direction | &3 | &3 | &3 | &51 | &51 |
| TrainTimeTable.Line.Direction@name | North | North | North | Nort-West | Nort-West |
| TrainTimeTable.Line.Direction.Service | &4 | &4 | &26 | &52 | &52 |
| TrainTimeTable.Line.Direction.Service.@no | 1 | 1 | 2 | 1 | 1 |
| TrainTimeTable.Line.Direction.Service.@from | Wellington | Wellington | Wellington | Wellington | Wellington |
| TrainTimeTable.Line.Direction.Service.@to | Upper Hutt | Upper Hutt | Taita | Waikanae | Waikanae |
| TrainTimeTable.Line.Direction.Service.Station | &5 | &23 | &30 | &53 | &56 |
| TrainTimeTable.Line.Direction.Service. Station.@name | Wellington | Upper Hutt | Petone | Wellington | Paekakariki |
| TrainTimeTable.Line.Direction.Service. Station.@arr_time | ⊥ | 06:10 am | 07:11 am | ⊥ | 06:10 am |
| TrainTimeTable.Line.Direction.Service. Station.@dept_time | 05:25 am | ⊥ | 07:12 am | 05:25 am | 06:12 am |
| TrainTimeTable.Line.Direction.Service.Station. Station_Facilities | &6 | &24 | &31 | &54 | &57 |
| TrainTimeTable.Line.Direction.Service.Statio. Station_Facilities.Text | &7 | &25 | &32 | &55 | &58 |
| TrainTimeTable.Line.Direction.Service.Station. Station_Facilities.Text.S | Lengthy description | Lengthy description | Lengthy description | Lengthy description | Lengthy description |
| TrainTimeTable.Line.Direction.Service.Station. Station_Facilities.Photo | ⊥ | ⊥ | ⊥ | ⊥ | ⊥ |
| TrainTimeTable.Line.Direction.Service.Station. Station_Facilities.Photo.@url | ⊥ | ⊥ | ⊥ | ⊥ | ⊥ |

**a)** **[4 marks]** How many maximal tree tuples can be inferred from the XML document `Train_TimeTable_A2_14.xml`? Justify your answer.

**b)** **[4 marks]** Define the tree tuple $t_6$ you will need to show that the `Train_TimeTable_A2_14.xml` document does not satisfy the following functional dependency

{TrainTimeTable.Line.@name, TrainTimeTable.Line.Direction.Service.@no} → TrainTimeTable.Line.Direction.@name

ANSWER

| Path | $t_6$ |
|---|---|
| TrainTimeTable | |
| TrainTimeTable.Line | |
| TrainTimeTable.Line.@name | |
| TrainTimeTable.Line.Direction | |
| TrainTimeTable.Line.Direction@name | |
| TrainTimeTable.Line.Direction.Service | |
| TrainTimeTable.Line.Direction.Service.@no | |
| TrainTimeTable.Line.Direction.Service.@from | |
| TrainTimeTable.Line.Direction.Service.@to | |
| TrainTimeTable.Line.Direction.Service.Stop | |
| TrainTimeTable.Line.Direction.Service.Stop.@name | |
| TrainTimeTable.Line.Direction.Service.Stop.@arr_time | |
| TrainTimeTable.Line.Direction.Service.Stop.@dept_time | |
| TrainTimeTable.Line.Direction.Service.Stop.Facilities | |
| TrainTimeTable.Line.Direction.Service.Stop.Facilities.S | |

**c)** **[4 marks]** Justify the claim that the `TrainTimeTable.xml` document does not satisfy the functional dependency

{TrainTimeTable.Line.@name, TrainTimeTable.Line.Direction.Service.@no} → TrainTimeTable.Line.Direction.@name

Use a tuple from the Table 1 and the tuple you defined in the question above.

**d)** **[4 marks]** Use functional dependencies of Arenas&Libkin to define a key of the `Service` element. Note, a separate space of ordinal numbers is assigned to `@no` values of each `Service` within a `Direction` of a `Line`.

**e)** **[4 marks]** Define an XML functional dependency that will enforce the following rule in all `Train_TimeTable.dtd` instances:

"Two (or more) trains from the same direction are not allowed to arrive simultaneously at the same stop."

## Question 5. XML Normal Form [10 marks]

This question builds on Question 4. The `Train_TimeTable.dtd` schema is not in XML normal form.

**a) [5 marks]** List all non transitive anomalous XML functional dependencies of `Train_TimeTable.dtd` that are satisfied by the `TrainTimeTable.xml` document. Take an anomalous functional dependency and justify your claim that it is anomalous.

**b) [5 marks]** Transform the XML Schema `Train_TimeTable_A2_14.dtd` into a DTD whose instances will be in XML Normal Form. Indicate clearly which of anomalous functional dependencies each of your changes intends to address.

## Question 6. XML Normal Form of (*D*, Σ) [10 marks]

Consider the following DTD schema *D*:

```
<!ELEMENT Department (Course+)>
<!ATTLIST Department did ID #REQUIRED name CDATA #IMPLIED>
<!ELEMENT Course (Title, Lecturer+)>
<!ATTLIST Course cid ID #REQUIRED>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Lecturer (Name, Position, Address)>
<!ATTLIST Lecturer lid ID #REQUIRED>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Position (#PCDATA)>
<!ELEMENT Address (#PCDATA)>
```

and the following set of XML functional dependencies:

$\Sigma$ = {Department.Course.Lecturer.@lid $\rightarrow$ Department.Course.Lecturer}

**a) [2 marks]** What is the meaning of the sole functional dependency in $\Sigma$?

**b) [8 marks]** Let us designate by *Inst*(*D*, $\Sigma$) the set of all XML documents that are valid with regard to *D* and satisfy the set of XML functional dependencies $\Sigma$.

Define the set of all functional dependencies *AFD* that may be anomalous in a document that belongs to *Inst*(*D*, $\Sigma$). Justify your answer.

## Submission Instruction:

Submit all your files electronically (including the files you used for testing), so that they can be tested using `xmllint` or `Xerces2 Java`. Submit a printed version of your answers in the SWEN432 hand-in box on the second floor of the Cotton Building. Whenever appropriate, include the `xmllint` output produced by using `–noout` option instead of a whole document.