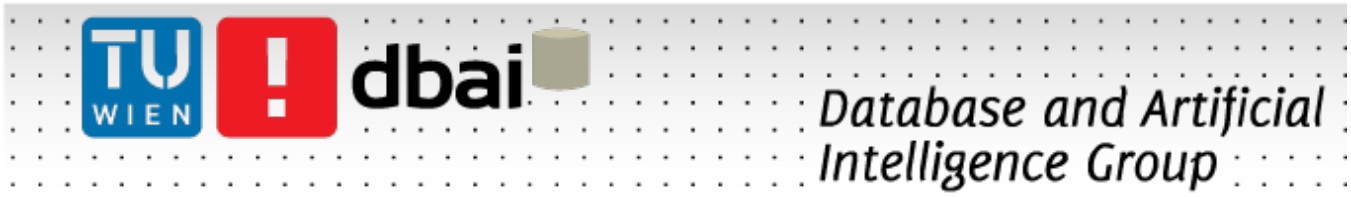

Exercise 2

(10 points)

The goal in this year's exercise is to develop an XML-based format to represent data for the quiz game "Jeopardy!". In Exercise 2 you will create an XML Schema and XML document and in Exercise 3 you will be querying and transforming this document with XML-related technologies.

In Jeopardy two players compete against each other. The game displays questions of different value in different categories. Each player chooses alternately one of these questions. Then, the question will be displayed and the players can choose one or more answers. At least one answer is correct. If the answer is correct the players account is increased by the value of the question. The player with the highest account balance wins.

Remark: The following data format can be seen as a savegame for Jeopardy.

In this exercise we will develop the XML Schema and an XML Document, that validates against the XML Schema. Afterwards, we will translate this XML Schema into a Document Type Definition (DTD).

XML Schema `jeopardy.xsd`

The first part of this exercise consists of writing an XML Schema document `jeopardy.xsd`. The XML Schema should allow for XML Documents according to the following definitions:

Element `quiz`

The root element `jeopardy` stores all the relevant information for Jeopardy.

It contains the following 4 child elements in exactly this order, but each element is **optional**:

- `overview`
- `categories`
- `users`
- `games`

Element `overview`

The element `overview` is used to give a game description. It can contain elements `p` and `a`, and addtionally, also text (in any order, with any multiplicity). The element `a` contains just text and has an attribute `href`, which contains a link to some other page. The element `p` can contain again an elements `p` and `a`, and additionally, also text (again in any order, with any multiplicity).

For example

```
<overview><p>This is an <a href="example.htm">example</a> <p>text </p>!</p>
</overview>
```

is a valid overview element.

Element `categories`

The element `categories` has no attributes and is a child element of `jeopardy`. The subtree rooted

at `categories` saves the categories and the questions, which belong to this category. It may contain an unbounded number of `category` elements.

Element `category`
Each `category` element has a `name` attribute, which contains a unique category name and may have an unbounded number of `question` elements as children.

Element `question`
The element `question` stores a question with its answers. It contains in this order:

1. exactly one `text` element, which stores the question as string; and
2. at least two `answer` elements.

The `question` element has two attributes:

- an `id`, which is a nonnegative number and should globally identify this question;
- and a `value`, which gives the value of the question and can take one of the following values: "100", "200", "500", "750" and "1000".

Element `answer`
The element `answer` describes an answer of some question. It contains a string and has a `correct` attribute. The attribute `correct` can either take the value "yes" or "no". If the attribute `correct` is missing, it should take the value "no".

Element `users`
The element `users` has no attributes and is a child element of `jeopardy`. The subtree rooted at `users` saves the users. It may contain an unbounded number of `user` elements.

Element `user`
The element `user` describes a possible Jeopardy! player. It has the attribute `username`, which should be unique. Additionally, it may have the optional attribute `gender` which can only take the values "male" or "female". The element `user` has the following child elements in exactly this order:

1. Exactly one element `password` that contains a string.
2. - Either the element `fullname` that contains a string, or
   - the element `name` which contains the elements `firstname` and `lastname` (both contain a string).
3. The element `birthdate` that contains a date in the format: YYYY-MM-DD.
4. An unbounded number of `email` elements, all of which contain a string.

Element `games`
The element `games` has no attributes, but may contain an unbounded number of `game` elements.

Element `game`
The element `game` describes a Jeopardy! game. It has an attribute `session` which uniquely identifies the game. The `session` attribute is a string. It has the following child elements in exactly this order:

- Exactly two `player` elements; and
- At most ten `asked` elements.

Element `player`
The `player` element identifies one player of a game. The element has no content. Its only attribute

`ref` references a `username` attribute of a `user` element.

Element `asked`
The element `asked` saves the information on the already asked questions and the given answers. The asked question is stored in the `question` attribute, which must refer to an `id` attribute of a `question` element. It may have an unbounded number of `givenanswer` elements as children.

Element `givenanswer`
The element `givenanswer` contains a string, i.e. the answer, and has a required `player` attribute which references the `username` attribute of a `user` element.

Remark: You don't need to ensure that the text of the given answer corresponds to one of the answers in the corresponding question element. Additionally, you also don't need to check if the player attribute is one of the players given by the `player` elements. Are these checks possible with XML Schema? (think about it for the assignment discussion)

Keys
Add the following keys to your document:

- `userKeys` for the usernames.
- `questionKeys` for the questions.

The keys are referenced in the following fields:

- `userKeys` is referenced in the `ref` attribute of the `player` element and in the `player` attribute of the `givenanswer` element.
- `questionKeys` is referenced in the `question` attribute of the `asked` element.

General Remarks
Please pay attention to the following remarks:

- If nothing else is mentioned all elements and attributes are required. The word "may" hints you to optional elements or attributes.
- All numbers are integers.

Summary

- **Files**: `jeopardy.xsd`
- **Maximum number of points**: 5

XML Document `jeopardy.xml`

Create an XML Document `jeopardy.xml` for the XML Schema `jeopardy.xsd`. The XML Document should satisfy the following criteria:

- Create at least three `category` elements.
- Create at least six `question` elements with at least 3 answers each.
- Create four users. Please make sure that you have male and female users.
- Create at least one `game` element.
- The game should have at least two `asked` elements.
- Make sure that each `asked` element has at least two `givenanswer` elements.

Make sure that your XML Document `jeopardy.xml` validates against your XML Schema

`jeopardy.xsd`. This can be done with the following command (after you have installed `xmllint`:

```
xmllint --schema jeopardy.xsd jeopardy.xml
```

Downloads and user instructions to `xmllint` can be found on our exercise page.

Summary

- **Files**: `jeopardy.xml`
- **Maximum number of points**: 3

Document Type Definition (DTD) `jeopardy.dtd`

Create a Document Type Definition (DTD) `jeopardy.dtd`, for which the most XML Documents are valid, which are also valid for the XML Schema `jeopardy.xsd`.

There are some specifications in your XML Schema file which have a very complicated or no equivalent at all in DTDs. Which functionalities are these? (think about it for the assignment discussion)

Especially you don't need to create large number ranges as enumerations (e.g. "Numbers between 1 and 72 as an enumeration of 72 numbers).

Bemerkungen
Make sure that your XML Document `jeopardy.xml` validates against your DTD `jeopardy.dtd`. This can be done with the following command (after you have installed `xmllint`:

```
xmllint --dtdvalid jeopardy.dtd jeopardy.xml
```

Summary

- **Files**: `jeopardy.dtd`
- **Maximum number of points**: 2

Submission

In total you have to upload the following files:

- `jeopardy.xml`
- `jeopardy.xsd`
- `jeopardy.dtd`

These files should be zipped and the resulting ZIP-file `exercise2.zip` has to be uploaded until 07.05.2015 23:59 in our CourseManager . We will grade the last uploaded solution.

Assignment discussion
You can receive at most 10 points for Exercise 2. During the assignment discussion we will not only check your solution for correctness, but will also ask some questions regarding the used technologies. Please be prepared to answer the questions mentioned in this document as well.

To obtain the full number of points your solution has to be solved correctly and you have to be able to explain it. Copied solutions are awarded 0 points!

In your own interest come **ontime** to your assignment discussion or else we don't guarantee that your solution is completely checked in the remaining time of the reserved slot.