

# Klausur “Semistructured Data and XML”

Summer Term 2020

Prof. Dr. Wolfgang May

21. August 2020, 10-12 Uhr

**Working Time:** planned 90 Minutes, then extended to 150 Min  
(carried out as a computer-based ILIAS exam)

Vorname:

Nachname:

Matrikelnummer:

**Setting:** on the E-exam-computers, ILIAS was running, Notepad++ as editor, and also saxon (with the aliases saxonValid, saxonXQ, and saxonXSL defined as in the course) were installed.

In this exam basically all paper-based aids are allowed. For the exam, it was recommended to have the printed slides, and a condensed self-prepared “cheat sheet” (preparation of a cheat sheet is a very effective way to work through the materials). Mobile phones must be turned off.

Answers might be given in English or German (most answers are program code anyway). In the text, the german translation is sometimes given in parentheses.

Give *all* answers via the ILIAS system.

Like in a “paper exam”, also solutions that do maybe not work (or do not work completely) can be delivered and will be graded with appropriately partial points.

For **passing** the exam, **50** points are sufficient.

	Max. Punkte	Erreichte Punkte
Aufgabe 1 (XML)	20	
Aufgabe 2 (DTD)	15	
Aufgabe 3 (XPath (a) )	2	
Aufgabe 4 (Appl.-semantic validation of query result)	2	
Aufgabe 5 (XPath/XQuery (b) )	4	
Aufgabe 6 (XPath/XQuery (c) )	6	
Aufgabe 7 (XPath/XQuery (d) )	5	
Aufgabe 8 (XPath/XQuery (e) )	5	
Aufgabe 9 (XPath/XQuery (f) )	10	
Aufgabe 10 (XPath/XQuery (g) )	10	
Aufgabe 11 (XSLT )	15	
Aufgabe 12 (Miscellaneous (a) )	2	
Aufgabe 13 (Miscellaneous (b) )	4	
Summe	100	

**Note:**

---

## Project: Housing Cooperative (german: Wohnungsgenossenschaft)

All exercises are based on a common “project”: the database of a housing cooperative that owns houses with rental apartments in cities all over Germany.

1. Assume that cities are uniquely identified by their name. Also, every street name exists at most once in every city.
2. For each building that is owned by the cooperative, the address (city, street, house number) is stored. It is also stored how many apartments are in the building (assume that always all apartments in the building are owned by the cooperative).
  - The building *Hauptstraße 100* in *Göttingen* consists of 80 apartments.
  - The building *Rheinstraße 53* in *Köln* consists of 100 apartments.
3. Every apartment has a (unique) number in its building. For each apartment, the number of rooms, the size (in square meters) and the current monthly rent (=Kaltmiete) is stored.
  - The apartment with the number 42 of the building *Hauptstraße 100* in *Göttingen* has three rooms, 82 m<sup>2</sup>, for 850 EUR per month.
  - The apartment with the number 43 of the same building has two rooms, 50 m<sup>2</sup>, for 550 EUR per month.
  - The apartment with the number 17 of the building *Rheinstraße 53* in *Köln* has three rooms, 95 m<sup>2</sup>, for 1300 EUR per month.
4. For being entitled to rent an apartment in a cooperative, persons must buy a share and become *members* of the cooperative. For every person, the name, the (unique) membership code (recall that XML IDs must start with a letter), the birthdate, and the birth place (city name, country) are stored.
  - John Doe is born on February 28th, 1995 in Boston, USA.
  - Markus Meier is born on July 10th, 1980 in Göttingen, Germany.
  - Liese Müller is born on October 1st, 1950 in Köln, Germany.
  - Jan Jansen is born on August 10th, 1985 in Stockholm, Sweden.
5. For every individual rental agreement (=Mietvertrag) the name of the tenant (=Mietter) is stored (assume that the agreement is always made with a single person), together with the starting date, and the contact address where the tenant lived when the contract was signed.

Contracts always begin with the first day of a month, and end with the last day of a month.

Additionally, it is stored whether the contract is terminated or announced to be terminated, and for which date (=gekündigt; in Germany, a rental contract must be announced to be terminated three months in advance – e.g. today you could announce to terminate the contract for your apartment for November 30th, 2020).

- *John Doe* lives in the apartment nr. 42 of the building *Hauptstrasse 100* in *Göttingen* since April 1st, 2018. The contract is not announced to be terminated.
  - *Markus Meier* lives in the apartment nr. 43 of the building *Hauptstrasse 100* in *Göttingen* since August 1st, 2011. He already announced to terminate the contract for October 31st, 2020.
  - *Liese Müller* lives in the apartment nr. 17 of the building *Rheinstrasse 53* in *Köln* since March 1st, 1990. She announced to terminate the contract for August 31st, 2020.
6. When a new contract is signed, the (at this point current) address of the tenant and the starting date of the contract is stored.
- The above-mentioned *Markus Meier* has signed a contract for the apartment nr. 17 of the building *Rheinstrasse 53* in *Köln*, beginning October 1st, 2020.
  - *Jan Jansen*, who currently lives at the address *Hafenstrasse 10, Hamburg* (this apartment is not owned by the cooperative), signed a contract for the apartment nr. 43 of the building *Hauptstrasse 100* in *Göttingen* from December 1st, 2020 on.
7. If a tenant moves somewhere else (not necessarily to an apartment of the cooperative), the new address is stored (e.g., to send the final cost bills).
- The future address of *Liese Müller* is *Mühlengasse 24, Mühlhausen* in a building that does not belong to the cooperative.
8. Data about older contracts is kept in the database, it is *not* deleted.

### Exercise 1 (XML [20 Points])

Design an XML structure (use the frame given in file `exam.xml`) and fill it with some sample data (e.g. with some of the example data given in the text).

(Information about handling dates in XML can be found in the course's slides around Slide 300.)

Copy-and-paste the XML from the file `exam.xml` afterwards (at the end of the exam, because it will be extended in later exercises) here:

### Exercise 2 (DTD [15 Points])

Develop the DTD for your document developed in Exercise 1, use the file `exam.dtd`.

Use the saxon call

```
saxonValid.bat -s:exam.xml
```

for validating it.

Copy-and-paste the DTD from the file `exam.dtd` afterwards here:

### Exercise 3 (XPath (a) [2 Points])

Use your `exam.xml` XML file as a basis for solving this and the following exercises.

None of the results should contain duplicates.

Give an XPath query or an XQuery query that returns the names of all cities where the cooperative owns at least one apartment that consists of *exactly three* rooms.

Write the XPath query string in the file `query1.xq` and call it with

```
saxonXQ.bat -s:exam.xml query1.xq
```

Copy-and-paste the query from `query1.xq` afterwards here:

### Exercise 4 (Appl.-semantic validation of query result [2 Points])

Add additional sample data to your XML designed in Exercise 1 to test that your result does not return duplicates in case that there are multiple such apartments in a city.

Paste your *result* of evaluating the query from Exercise 3 (XPath (a)) here:

### Exercise 5 (XPath/XQuery (b) [4 Points])

Give an XPath query or an XQuery query that returns the names of all persons who have *currently* rented an apartment that consists of exactly three rooms and costs less than 900 EUR.

(note: the current date can be obtained by the XPath function `current-date(.)`.)

Copy-and-paste the query from `query2.xq` afterwards here:

### Exercise 6 (XPath/XQuery (c) [6 Points])

Give an XPath query or an XQuery query which, for every city, yields the total earnings of the cooperative in October 2020. The results should be of the format

```
<city name="..." total="..."/>
```

Copy-and-paste the query from `query3.xq` afterwards here:

**Exercise 7 (XPath/XQuery (d) [5 Points])**

Give an XPath query or an XQuery query that yields the names of all persons who ever rented an apartment of the cooperative, but never rented an apartment of the cooperative in Köln.

Copy-and-paste the query from query4.xq afterwards here:

**Exercise 8 (XPath/XQuery (e) [5 Points])**

Give an XQuery query or an XPath query that returns the names of all cities such that in *every* building owned by the cooperative in this city there is at least one apartment that has at least 100 square meters.

Copy-and-paste the query from query5.xq afterwards here:

**Exercise 9 (XPath/XQuery (f) [10 Points])**

Give an XQuery query that returns information about all persons who moved (or will move) from one apartment *directly* to another (both owned by the cooperative), and the second apartment is more expensive than the one before (use the *current rent* for time-independent comparison of “expensiveness” of apartments).

The results should be of the format

```
<person name="..." rent1="..." rent2="..." />
```

Copy-and-paste the query from query6.xq afterwards here:

**Exercise 10 (XPath/XQuery (g) [10 Points])**

Give an XQuery query or an XSLT stylesheet that, for each city “A”, gives a list of all cities “B” such that *some* person moved (or will move) *directly* from an apartment in city “A” to an apartment in “B”, and both of the apartments belong to the cooperative.

for each such city “A”, the result should be of the form

```
<result city="name-of-city-A">
  name-of-city-B1 ... name-of-city-Bn
</result>
```

(no duplicate B’s, in any order, with arbitrary whitespaces)

Copy-and-paste the query from query7.xq afterwards here:

**Exercise 11 (XSLT [15 Points])**

Extend the given XSL stylesheet frame exam.xsl to an XSLT stylesheet that for each person returns the name in an h3-element, followed by an HTML list that chronologically lists all apartments (by address) that the person had rented from the cooperative.

(an HTML list consists of an ul-element with nested li-elements; omit the surrounding HTML stuff, don’t care too much for formatting)

Use the following call to execute it:

```
saxonXSL.bat -s:exam.xml exam.xsl
```

Copy-and-paste the XSLT stylesheet from exam.xsl afterwards here:

**Exercise 12 (Miscellaneous (a) [2 Points])**

Describe (short and concisely) what ID and IDREF in XML, and primary keys and foreign keys in relational databases both are used for:

### Exercise 13 (Miscellaneous (b) [4 Points])

Describe **two** aspects in which the ID/IDREF mechanism in the XML world differs from the primary key/foreign key concept in relational databases:

---

The following frames can be used:

- Frame for XML file exam.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE put-name-here SYSTEM "exam.dtd">
  to be extended here
```

- Frame for XML stylesheet exam.xsl:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="2.0">
  to be extended here
</xsl:stylesheet>
```