Prof. Dr. Stefan Brass, Dr. Henning Thielemann
Institut für Informatik
Martin-Luther-Universität Halle-Wittenberg

6. September 2012

Vorlesung „XML und Datenbanken“
— Klausur —

Name: _____

Studiengang: _____

Matrikelnummer: _____

(Diese Daten werden zur Ausstellung des Leistungsnachweises benötigt.)

Aufgabe	Punkte	Max. Punkte	Zeit
1. Allgemeines		10	20 min
2. XML-Schema		20	30 min
3. XPath/XQuery		15	20 min
4. XML-Schema-Überprüfung		5	15 min
Summe		50	85 min

☐ Ich fühle mich gesundheitlich in der Lage, diese Prüfung abzulegen. (Bitte sprechen Sie mit dem Aufsichtspersonal, falls Sie sich krank fühlen.)

☐ Für den Fall, daß ich nicht korrekt zu dieser Prüfung angemeldet sein sollte, erkläre ich mich sowohl damit einverstanden, daß ich rückwirkend angemeldet werde, als auch damit, daß diese Prüfung nicht zählt (Entscheidung des Dozenten).

Unterschrift: _____

Hinweise:

- Bearbeitungsdauer: 90 Minuten
- Skript, Bücher, Notizen sind erlaubt.
  Notebooks, PDAs, Handys etc. dürfen nicht verwendet werden.
- Die Klausur hat 14 Seiten. Bitte prüfen Sie die Vollständigkeit.
- Verwenden Sie weder Rot- noch Bleistift zum Bearbeiten der Aufgaben.
- Bitte benutzen Sie den vorgegebenen Platz. Wenn Sie auf die Rückseite ausweichen müssen, markieren Sie klar, dass es eine Fortsetzung gibt.
- Tauschen Sie keinesfalls irgendwelche Dinge mit den Nachbarn aus. Notfalls rufen Sie eine Aufsichtsperson zur Kontrolle.
- Bei Aufgabe 1 zum Ankreuzen sollten Sie wenigstens raten, wenn Sie die richtige Lösung nicht wissen.
- Fragen Sie, wenn Ihnen eine Aufgabe nicht klar ist!

Aufgabe 1 (Allgemeines) 10 Punkte

Bitte kreuzen Sie jeweils die korrekte Antwort an. Es ist genau eine Antwort pro Teilaufgabe richtig. (Mehr als ein Kreuz pro Teilaufgabe wird mit 0 Punkten bewertet.)

a) Welches der folgenden Dinge ist **keine** XML-Spezifikationssprache?

- ☐ DTD
- ☐ XML-Schema
- ☐ XSLT
- ☐ Relax NG

b) Welches der folgenden Dinge ist eine XML-Anfragesprache, also eine Alternative zu XPath?

- ☐ XDM
- ☐ XQuery
- ☐ Relax NG

c) Was gibt es bei der Groß- und Kleinschreibung in XML zu beachten?

- ☐ Namensraum-Kürzel wie `xs`, `xsi` müssen immer kleingeschrieben werden.
- ☐ Alle XML-Bezeichner, also sowohl Namensraum-Kürzel als auch Element- und Attribut-Namen, müssen klein geschrieben sein.
- ☐ Bei Namensraum-Kürzeln wird Groß- und Kleinschreibung unterschieden, bei Element- und Attribut-Namen hingegen nicht.
- ☐ Bei allen XML-Bezeichnern wird Groß- und Kleinschreibung unterschieden und es sind große und kleine Buchstaben erlaubt.

d) Welche der folgenden Aussagen über Typen in Document Type Definitions trifft zu?

- ☐ Eine DOCTYPE-Definition kennt gar keine Typen.
- ☐ Eine DOCTYPE-Definition kann nur Attributen Typen verleihen, Element-Inhalten hingegen nicht.
- ☐ Eine DOCTYPE-Definition kann sowohl Element-Inhalten als auch Attributen Typen zuweisen.

e) Betrachten Sie die beiden Definitionen von erlaubten Unterelementen in einer Document Type Definition:

1. `(A,B)*`
2. `(A*,B*)`

Sind die beiden Ausdrücke gleichwertig?

- ☐ Beide Ausdrücke erlauben die gleichen Folgen mit den Elementen `A` und `B`.
- ☐ Im allgemeinen sind die Ausdrücke verschieden, aber wenn `A` als `EMPTY` definiert ist, dann sind beide Ausdrücke gleichwertig.
- ☐ Unabhängig von der Definition von `A` sind die Ausdrücke verschieden.

f) Bedeuten `<abc></abc>` und `<abc/>` immer dasselbe?

- ☐ Die Schreibweise `<abc></abc>` steht für ein Element mit leerem Text, wohingegen `<abc/>` für ein Nil (vergleichbar dem NULL-Wert in relationalen Datenbanken) steht.
- ☐ Ja, so ist es in der W3C-Recommendation festgelegt. Allerdings wird für XHTML zur Unterstützung älterer Browser empfohlen, für Nicht-EMPTY-Elemente nur die nicht-minimierte Schreibweise zu verwenden (`<div></div>`) und für EMPTY-Elemente nur die minimierte Schreibweise (`<br/>` oder `<br />`).
- ☐ Ist das Element `abc` in der DTD als `EMPTY` definiert, so darf nur `<abc/>` geschrieben werden. Ist das Element `abc` in der DTD nicht als `EMPTY` definiert, so darf nur `<abc></abc>` geschrieben werden.

g) Betrachten Sie eine Schlüsseldefinition in einem XML-Schema. Welcher Bestandteil von XPath-Ausdrücken ist in der Definition eines Schlüsselattributs (also dort, wo in `<xs:field xpath="???"/>` die Fragezeichen stehen) verboten?

- ☐ `..`
- ☐ `/`
- ☐ `:`
- ☐ `@`

h) Was ergibt der XPath-Ausdruck `((1,2,3),(4,5,6),(7,8,9))[3][2]`?

- ☐ `8`, weil das im dritten Tripel als zweites Element steht.
- ☐ `6`, weil das im zweiten Tripel als drittes Element steht.
- ☐ Es ist `()`, weil der Operator `[3]` eine Folge der Länge eins erzeugt und damit `[2]` auf ein Element außerhalb einer einelementigen Folge zugreift, was als `()` behandelt wird.
- ☐ Es ist undefiniert, weil XPath keine geschachtelten Folgen kennt und der Operator `[3]` ein einzelnes Element als Ergebnis hat, man `[2]` aber nur auf Folgen anwenden kann.

i) Beim Entwurf eines XML-Schemas ist die Entscheidung darüber, ob man Informationen in einem Attribut oder einem Element verwaltet, häufig eine Geschmacksfrage. Welcher der folgenden Schlüsse ist allerdings zwingend?

- ☐ Wenn ich einen Wert mit einfachem Typ speichern will, muss ich dafür ein Attribut verwenden.
- ☐ Wenn ich einen Wert mit komplexem Typ speichern will, muss ich dafür ein Element verwenden.
- ☐ Schlüsselattribute müssen in XML-Attributen gespeichert sein.
- ☐ Attribute eines Fremdschlüssels müssen in XML-Attributen gespeichert sein.
- ☐ Texte, die länger als 256 Zeichen sind, dürfen nicht in einem Attribut stehen, und müssen daher in einem Element untergebracht werden.

j) Muss in einem Dokument der Inhalt eines Elementes genau den Typ haben, der dem Element im XML-Schema zugewiesen wurde?

- ☐ Ja, der Elementinhalt in einem Dokument muss immer genau zur Typdefinition im XML-Schema passen.
- ☐ Nein, der Typ eines Elementes kann im Dokument jederzeit durch ein `xsi:type`-Attribut spezialisiert werden.
- ☐ Der Standardfall ist, dass der Typ eines Elementes im Dokument durch ein `xsi:type`-Attribut spezialisiert werden kann. Durch die Verwendung von `block` im XML-Schema, kann die Verwendung von `xsi:type` jedoch für bestimmte Typen unterbunden werden.

Aufgabe 2 (XML-Schema) 20 Punkte

Ihr Auftrag ist der Entwurf einer XML-Struktur zur Verwaltung einer Agentur zur Statistikschönung und Gängelung von Erwerbslosen. Ihr XML-Entwurf ist als Übergangslösung nötig geworden, nachdem die offizielle Software auch nach Jahren des Einsatzes grobe Fehler enthält oder aber von den umgeschulten und befristet angestellten Telekom-Mitarbeitern nicht bedient werden kann. Ihr Entwurf soll folgende Dinge erfassen:

- Bittsteller,
- Fallmanager, also Agenturmitarbeiter, die den sozialen Fall der Bittsteller organisieren,
- Maßnahmen, mit denen Bittsteller aus der Statistik und der Schwarzarbeit ferngehalten werden.

Eine Beispieldatei ist auf der folgenden Seite abgedruckt. Es gelten folgende Beziehungen:

- Bittsteller und Fallmanager haben einen Namen, über den sie identifiziert werden.
- Optional kann ein Bittsteller einen Namenszusatz bekommen, der durch eine bundesweit vertriebene Tageszeitung verliehen wird.
- Jeder Bittsteller ist genau einem Fallmanager unterstellt. Einem Fallmanager können kein, ein oder mehrere Bittsteller untergeordnet sein.
- In jede Maßnahme können mehrere Bittsteller gedrängt werden. Eine Maßnahme kann aber auch ohne Teilnehmer bleiben.
- In einer Maßnahme kann jeder Bittsteller nur einmal auftauchen.

Beispieldokument für Agenturverwaltung

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<agentur xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<manager name="Fred Hülsensack">
<bittsteller name="Arno Dübel"
             zusatz="Deutschlands frechster Arbeitsloser" />
<bittsteller name="Rolf J." zusatz="Florida-Rolf" />
</manager>

<manager name="Frau Werner">
<bittsteller name="Verena Storm" />
</manager>

<manager name="Kurt Beck">
<bittsteller name="Henrico Frank" />
</manager>

<massnahme name="Puzzle zusammensetzen">
<bittsteller name="Arno Dübel" />
<bittsteller name="Rolf J." />
<bittsteller name="Henrico Frank" />
</massnahme>

<massnahme name="Sittenwidriges Arbeitsangebot">
<bittsteller name="Verena Storm" />
</massnahme>

<massnahme name="Waschen und Rasieren">
<bittsteller name="Henrico Frank" />
</massnahme>

</agentur>
```

Entwickeln Sie ein XML-Schema zu der gegebenen XML-Datei. Schöpfen Sie alle Möglichkeiten von XML-Schema aus, um alle beschriebenen Integritätsbedingungen zu sichern. Nennen Sie die Bedingungen, die Ihr Schema nicht sicherstellen kann.

Aufgabe 3 (XPath/XQuery) 15 Punkte

In dieser Aufgabe geht es um Anfragen an die Agentur-Verwaltungsdatenbank aus der vorigen Aufgabe, mit dem Beispieldokument auf Seite 7.

a) Was berechnet die folgende Anfrage?

```
xquery version "1.0" encoding "ISO-8859-1";
<liste>
{
for $b in agentur/manager/bittsteller
return
  element bittsteller {
    attribute name {$b/@name},
    attribute massnahmen
      {count(agentur/massnahme/bittsteller[@name=$b/@name])}
  }
}
</liste>
```

.....

.....

b) Schreiben Sie für folgende Aufgaben jeweils eine Anfrage. Entscheiden Sie selbst, welche Anfrage sich besser mit XPath oder mit XQuery formulieren lässt. Die Anfrage soll für jedes Dokument die passende Antwort finden, nicht nur für das Beispieldokument.

- Ermitteln Sie die Namen aller Bittsteller, für die ein Namenszusatz gespeichert ist.

- Geben Sie für jeden Bittsteller den zugehörigen Fallmanager aus. Etwa so `Henrico Frank, Fallmanager: Kurt Beck`.

- Geben Sie für jede Maßnahme die Anzahl der Teilnehmer aus.

- Geben Sie den Namen des Bittstellers aus, der mit den meisten Maßnahmen beschäftigt wird.

Aufgabe 4 (XML-Schema-Überprüfung) 5 Punkte

Im folgenden sind ein XML-Schema und ein XML-Dokument für einen Bücherkatalog aufgelistet. Neben Autor und Titel sind zu jedem Buch Angebote von gebrauchten Exemplaren verzeichnet. Das XML-Schema ist fehlerfrei, jedoch enthält das XML-Dokument 6 Fehler. Es gibt sowohl XML-Syntax-Fehler als auch Inkonsistenzen zwischen XML-Schema und XML-Dokument. Geben Sie 5 dieser Fehler an! Es werden nur die ersten 5 Fehler gewertet, falls Sie mehr als 5 Fehler angeben. Bitte tragen Sie Ihre Lösung in die folgenden Felder ein. Nutzen Sie dabei die angegebenen Zeilennummern für die Beschreibung des Fehlerortes.

.....

.....

.....

.....

.....

Es folgen das XML-Schema und das XML-Dokument.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:buch="http://www.buecherwurm.de/"
        targetNamespace="http://www.buecherwurm.de/"
        elementFormDefault="qualified"
        attributeFormDefault="unqualified">
  <element name="katalog" type="buch:katalogtyp"/>
  <complexType name="katalogtyp">
    <sequence>
      <element name="haendler" type="buch:haendlertyp"/>
      <element name="liste"    type="buch:listentyp">
        <unique name="isbnschlüssel">
          <selector xpath="buch:artikel"/>
          <field xpath="@isbn"/>
        </unique>
      </element>
    </sequence>
  </complexType>
  <complexType name="haendlertyp">
    <sequence>
      <element name="haendlername" type="string"/>
      <element name="haendlerort" type="string"/>
    </sequence>
  </complexType>
  <complexType name="listentyp">
    <sequence>
      <element name="artikel" minOccurs="1" maxOccurs="unbounded">
        <complexType>
          <sequence>
            <element name="titel" type="string"/>
            <element name="autor" type="string"/>
            <element name="datum" type="buch:datum"/>
            <element name="angebot" maxOccurs="unbounded">
              <complexType>
                <attribute name="zustand" type="buch:zustandstyp"/>
                <attribute name="preis" type="buch:eurobetrag"/>
              </complexType>
            </element>
          </sequence>
          <attribute name="isbn" type="buch:isbn"/>
        </complexType>
      </element>
    </sequence>
  </complexType>
  <simpleType name="isbn">
    <restriction base="string">
```

```
47        <pattern value="(\d{3})?\d{9}[0-9X]"/>
48      </restriction>
49    </simpleType>
50    <simpleType name="datum">
51      <restriction base="string">
52        <pattern value="\d{4}-(0[1-9]|1[0-2])-(0[1-9]|[1-2][0-9]|3[01])"/>
53      </restriction>
54    </simpleType>
55    <simpleType name="eurobetrag">
56      <restriction base="decimal">
57        <fractionDigits value="2"/>
58      </restriction>
59    </simpleType>
60    <simpleType name="zustandstyp">
61      <restriction base="string">
62        <enumeration value="neu" />
63        <enumeration value="wie neu" />
64        <enumeration value="sehr gut" />
65        <enumeration value="gut" />
66        <enumeration value="akzeptabel" />
67      </restriction>
68    </simpleType>
69  </schema>
```

```
<?XML version="1.0" encoding="ISO-8859-1"?>
<buch:katalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:buch="http://www.buecherwurm.de/">
  <buch:haendler>
    <buch:haendlername>Bücherwurm</buch:haendlername>
    <buch:haendlerort>Leserattenland</buch:haendlerort>
  <buch:haendler/>
  <buch:liste>
    <buch:artikel isbn="349962107X">
      <buch:titel>Wir sind besser, als wir glauben</buch:titel>
      <buch:autor>Peter Bofinger</buch:autor>
      <buch:datum>2006-31-03</buch:datum>
      <buch:angebot zustand="neu" preis="20.00"/>
      <buch:angebot zustand="zerfleddert" preis="12.00"/>
    </buch:artikel>
    <buch:artikel isbn="9783834818478">
      <buch:titel>Numerik gewöhnlicher Differentialgleichungen</buch:titel>
      <buch:datum>2012-04-20</buch:datum>
      <buch:autor>Strehmel, Weiner, Podhaisky</buch:autor>
      <buch:angebot zustand="neu" preis="59.95"/>
    </buch:artikel>
    <buch:artikel isbn="349962107X">
      <buch:titel>Pearls of Functional Algorithm Design</buch:titel>
      <buch:autor>Richard Bird</buch:autor>
      <buch:datum>2010-11-01</buch:datum>
      <buch:angebot zustand="sehr gut" preis="49.00"/>
    </buch:artikel>
  </buch:liste>
</buch:katalog>
```