

1. Übungsblatt (SS 2019)

3.0 VU Semistrukturierte Daten

Informationen zum Übungsblatt

Allgemeines

Inhalt des ersten Übungsblattes ist die Entwicklung eines Schemata für ein Datenformat. Zuerst wird ein XML-Schema entwickelt und ein passendes XML Instanzdokument erstellt. Danach erfolgt die Umsetzung mittels Document Type Definition (DTD).

In dieser Übung geben Sie eine einzige ZIP Datei ab (max. 5MB). Diese ZIP Datei enthält ein XML Schema, eine DTD und zu Schema und DTD je ein passendes XML Instanzdokument, d.h. die Dateien:

- library.xsd
- library-xsd.xml
- library.dtd
- library-dtd.xml

Beachten Sie die Erklärungen bei den einzelnen Aufgaben. Das Übungsblatt enthält 5 Aufgaben, auf welche Sie insgesamt 10 Punkte erhalten können.

Deadlines

- bis **2.5. 12:00 Uhr** Upload der Abgabe über TUWEL
bis **5.5. 12:00 Uhr** Anmeldung zu einem Kontrollgespräch

Kontrollgespräch

Im Rahmen des Kontrollgespräches wird nicht nur die Korrektheit, sondern vor allem das Verständnis der Konzepte überprüft. Durch die Übung sollen sowohl Ihre praktische Problemlösungskompetenz als auch das theoretische Wissen über Semistrukturierte Daten gefördert werden. Sie müssen daher bei Ihrem Kontrollgespräch in der Lage sein, nicht nur Ihre Beispiele zu erklären, sondern ebenfalls zeigen, dass Sie die in der Vorlesung behandelte Theorie zu diesen Beispielen ausreichend verstanden haben. Dies soll Ihnen die Vorbereitung für die Prüfung erleichtern und so können Sie Ihr Wissen während der Kontrollgespräche selbst testen und gegebenenfalls vertiefen.

Die Bewertung Ihres Übungsblattes basiert zum überwiegenden Teil auf Ihrer Leistung beim Kontrollgespräch! Es ist daher im Extremfall durchaus möglich, dass eine korrekte Abgabe mit 0 Punkten bewertet wird. Insbesondere werden nicht selbstständig gelöste Abgaben immer mit 0 Punkten bewertet!

Erscheinen Sie in Ihrem eigenen Interesse bitte pünktlich zum Kontrollgespräch, da andernfalls nicht garantiert werden kann, dass Ihre gesamte Lösung in der verbleibenden Zeit beurteilt werden kann.

Bringen Sie bitte Ihren Studentenausweis zum Kontrollgespräch mit. Ein Kontrollgespräch ohne Ausweis ist nicht möglich.

Tutorensprechstunden (freiwillig)

Rund eine Woche vor der Abgabedeadline bieten die TutorInnen Sprechstunden an. Falls Sie Probleme mit oder Fragen zum Stoff des Übungsblattes haben, es Verständnisprobleme mit den Beispielen oder technische Fragen gibt, kommen Sie bitte einfach vorbei. Die TutorInnen beantworten Ihnen gerne Ihre Fragen zum Stoff, oder helfen Ihnen bei Problemen weiter.

Ziel der Sprechstunden ist es, Ihnen beim **Verständnis des Stoffs** zu helfen, nicht, das Übungsblatt für Sie zu lösen, oder die eigenen Lösungen vorab korrigiert zu bekommen.

Die Teilnahme ist vollkommen freiwillig — Termine und Orte der Tutorensprechstunden finden Sie in TUWEL.

Weitere Fragen – TUWEL Forum

Sie können darüber hinaus das TUWEL Forum verwenden, sollten Sie inhaltliche oder organisatorische Fragen haben.

Aufgaben: XML Schema

Sie werden zuerst ein XML Schema für eine Bibliothek erstellen. Speichern Sie dieses XML Schema in der Datei `library.xsd`. Zu diesem Schema wird in Aufgabe 3 ein XML Dokument `library-xsd.xml` erstellt.

Achtung: Stellen Sie sicher, dass Ihr Schema wohlgeformt und Ihr XML Dokument gültig ist! Sollte das nicht der Fall sein, werden Aufgaben 1 und 2 mit 0 Punkten bewertet! Falls Sie nur Teile der Aufgabe umsetzen können stellen Sie trotzdem sicher, dass sie ein wohlgeformtes Schema und ein zu diesem Schema gültiges XML Dokument abgeben!

Aufgabe 1 (Definieren von Elementen in `library.xsd`) [4 Punkte]

Das XML Schema soll XML Dokumente mit der folgenden Struktur validieren:

Element `library`. Das Wurzelement `library` speichert die gesamten Daten der “Bibliothek” und beinhaltet folgende Elemente in dieser Reihenfolge, wobei die Elemente `books` und `areas` *notwendig* sind während `members` *optional* ist:

- maximal ein `members` Element;
- genau ein `books` Element;
- beliebig viele `area` Elemente.

Element `members`. Das `members` Element hat keine Attribute und ist ein Kindelement von `library`. Dieser Teilbaum speichert alle Mitglieder unserer Bibliothek. Im `members` Element wird eine beliebige Anzahl von `member` Elementen gespeichert.

Element `member` (Kind von `members`). Ein `member` Element hat ein ganzzahliges Attribut `mid` und enthält als Inhalt den vollen Namen des Mitglieds (als String).

Element `books`. Das `books` Element hat keine Attribute und ist ein Kindelement von `library`. Dieser Teilbaum speichert alle in der Bibliothek vorhandenen Bücher. Im `books` Element werden *mindestens 2* `book` Elementen gespeichert (nach oben ist die Anzahl unbegrenzt).

Element book. Ein **book** Element hat ein ganzzahliges Attribut **id** und folgende Kindelemente (in genau dieser Reihenfolge):

- **title**: Speichert den Titel des Buchs als String.
- **isbn**: Speichert den ISBN code des Buchs als String. Dieses Element ist *optional*.
- Mindestens 1, maximal 2 **category** Elemente: Gibt Kategorien für das Buch als String an. Dabei sind nur genau die Werte “Mathematics”, “Logic”, “Computer Science” und “Cooking” erlaubt.
- Ein **location** Element.
- Ein **description** Element.
- Ein *optionales* **history** Element.

Element location. Ein **location** Element beschreibt den Ort an dem das Buch in der Bibliothek liegt. Es hat keine Attribute und enthält jedes der folgenden Kindelemente genau ein mal in *beliebiger* Reihenfolge.

- **area**: Speichert als String in welchem Bereich der Bibliothek das Buch liegt.
- **case**: Speichert die ganzzahlige Nummer des Bücherregals in dem das Buch liegt.
- **shelf**: Speichert die ganzzahlige Nummer der Reihe im Regal wo das Buch liegt.

Element description. Ein **description** Element enthält eine textuelle Beschreibung des Buches. Dabei soll es möglich sein beliebig viele Autoren und Jahreszahlen im Text mit **author** und **year** Elementen auszuzeichnen.

Element history. Ein **history** Element beschreibt die Entlehnungshistorie des Buchs. Ein **history** Element hat keine Attribute und enthält eine beliebige Anzahl von **borrowed** Elementen.

Element borrowed. **borrowed** Elemente haben ein *optionales* Attribut **until** das ein Datum enthält sowie ein *notwendiges* Attribut **by** das angibt welches Mitglied das Buch ausgeliehen hatte.

Element area (Kind von library). Ein **area** Element beschreibt einen Bereich der Bibliothek. Es hat immer ein **name** Attribut sowie zwei optionale Attribute **kind** und **floor**. Dabei sind **name** und **floor** Strings während **kind** nur die Werte **silent** und **free** annehmen kann. Falls kein **kind** Attribut vorhanden ist, wird **silent** als default Wert angenommen. In einem **area** Element wird genau ein **bookcases** Element als Kind gespeichert.

Element bookcases. Ein **bookcases** Element umfasst eine Menge an Bücherregalen. Konkret können beliebig viele **woodenCase**, **steelCase** und **lockedCase** Elemente, in beliebiger Reihenfolge und durchgemischt, in einem **bookcases** Element vorkommen.

Elemente `woodenCase`, `steelCase`, `lockedCase`. Jedes *woodenCase*, *steelCase* und *lockedCase* Element ist leer und hat zwei *notwendige ganzzahlige* Attribute, `id` und `shelves`.

Aufgabe 2 (Definieren von Keys und Referenzen in `library.xsd`) [2 Punkte]

Fügen Sie nun folgende Schlüssel zu ihrem Schema hinzu:

- Einen globalen Schlüssel `memberKeys` für das Attribut `mid` der `member` Kindelemente von `members`.
- Einen globalen Schlüssel `bookKeys` für das Attribut `id` der `book` Kindelemente von `books`.
- Einen globalen Schlüssel `areaKeys` für das Attribut `name` der `area` Elemente.
- Einen Schlüssel `caseKeys` für das `id` Attribut der Kindelemente eines `bookcases` Elements. Dabei muss der Schlüssel allerdings *nur im entsprechenden bookcases Element* einzigartig sein.

Fügen Sie folgende Schlüsselreferenzen zu ihrem Schema hinzu:

- Die `area` der `location` Elemente referenziert die `areaKeys`.
- Das `by` Attribut eines `borrowed` Elements referenziert die `memberKeys`.

Fügen Sie nun folgende Uniqueness-Constraint zu ihrem Schema hinzu:

- Stellen Sie sicher, dass alle Werte von `isbn` Elementen nur einmal vergeben werden.

Aufgabe 3 (Erstellen eines XML Dokuments `library-xsd.xml`) [1 Punkte]

Erstellen Sie ein XML Dokument `library-xsd.xml` für das Schema `library.xsd`. Das XML Dokument sollte folgende Bedingungen erfüllen:

- Erstellen Sie zumindest 6 `member` Elemente (im Element `members`).
- Erstellen Sie zumindest 4 `book` Elemente.
- Erstellen Sie zumindest 2 `area` Elemente (als Kinder von `library`).
- Ein `area` Element hat zumindest 3 Kinder in ihrem `bookcases` Element.
- Mindestens 2 `book` Elemente sollen `history` Kindelemente mit je mindestens 2 `borrowed` Elementen besitzen.
- Nutzen Sie jeden möglichen Werte für `category` jeweils in zumindest einem `book` Element.
- Erstellen Sie ein Element pro Schlüsselreferenz aus Aufgabe 2.
- Erstellen Sie zumindest 1 `author` und 1 `year` Element im Text eines `description` Elements.

Stellen Sie sicher, dass ihr XML Schema `library.xsd` ihr XML Dokument `library-xsd.xml` validiert. Dies können Sie mit folgendem Kommando testen (nachdem Sie `xmllint` installiert haben):

```
xmllint --schema library.xsd library-xsd.xml
```

Downloads und Installationsanweisungen zu `xmllint` können Sie im TUWEL finden.

Achtung: Falls ihr XML Dokument nicht wohlgeformt und gültig bezüglich des Schemas ist, wird diese Aufgabe mit 0 Punkten bewertet!

Aufgaben: Document Type Definition

Sie werden nun für obige Spezifikation eine DTD erstellen.

Aufgabe 4 (Erstellen einer DTD `library.dtd`) [2 Punkte]

Erstellen Sie eine Document Type Definition (DTD) `library.dtd`, welche zur obigen Beschreibung passt. Sollte eine Spezifikation nur sehr kompliziert oder gar nicht umsetzbar sein, treffen Sie entsprechende Annahmen.

Einige Spezifikationsmöglichkeiten von XML Schema lassen sich nur sehr umständlich oder gar nicht in einer DTD umsetzen. Welche Funktionalitäten sind dies? Sie müssen in jedem dieser Fälle beim Abgabegespräch begründen können, warum die Umsetzung in der DTD nicht oder nur teilweise möglich ist. **Wichtig:** Suchen Sie speziell auch nach Möglichkeiten um möglichst viele der Schlüssel und Schlüsselreferenzen umzusetzen.

Insbesondere ist es nicht notwendig, große Zahlenbereiche durch Aufzählungen umzusetzen (z.B. "Zahl zwischen 1 und 72" als Aufzählung von 72 Zahlen). Aufzählungen, die sich hingegen in einem kleinen Zahlenbereich bis inklusive 6 befinden (z.B. maximal fünf Levels, etc.) sind explizit umzusetzen!

Achtung: Falls sie eine DTD mit Syntaxfehlern abgeben, sie also nicht zum Validieren verwendet werden kann, wird diese Aufgabe mit 0 Punkten bewertet!

Aufgabe 5 (Erstellen eines XML Documents `library-dtd.xml`) [1 Punkte]

Sollte es nicht möglich sein, dass ihr zuvor erstelltes XML Dokument `library-xsd.xml` durch die DTD validiert werden kann, erstellen Sie ein zusätzliches XML Dokument `library-dtd.xml`, dass die selben Daten enthält und durch ihre DTD validiert wird.

Stellen Sie sicher, dass ihre DTD `library.dtd` ihr XML Dokument `library-dtd.xml` validiert. Dies können Sie mit folgendem Kommando testen (nachdem Sie `xmllint` installiert haben):

```
xmllint --dtdvalid library.dtd library-dtd.xml
```

Achtung: Falls ihr XML Dokument nicht wohlgeformt und gültig bezüglich der DTD ist, wird diese Aufgabe mit 0 Punkten bewertet!