

Gruppe 20/22/48

Tutorübung zu Einsatz und Realisierung von Datenbanksystemen (SS 18)

Michael Schwarz

Institut für Informatik
Technische Universität München

26.06 / 28.06.2018

Folien von Alexander Beischl und Maximilian Bandle - Danke!



Kapitel 20

XML-Anfragesprachen

XML-Anfragesprachen

XML (eXtensible Markup Language)

Daten in Baumstruktur und Attributen gespeichert

Schema kann aber muss nicht definiert werden

Basis von HTML

```
<Student ID="M1337" MatrNr="M1337">  
  <Name>1337</Name>  
  <Semester>9</Semester>  
  <hoert Vorlesungen="V5043 V5052 V5259 V5216 V4630"/>  
</Student>
```

XML-Anfragesprachen

XML (eXtensible Markup Language)

Finde die Fehler

```
<Uni Name="Alexander Maximilian Universität" Kuerzel=AMU>
  <UniLeitung>
    <Rektor>Max</Rektor>
    <Senatsvorsitzender>Alex</Senat>
  </UniLeitung>
  <Studenten />
  <Student>
    <Name Peter Name>
    <MatrNr>03670815</MatrNr>
    <Vorlesungen>V1<V2<V3</Vorlesungen>
  </Studenten>
</Uni>
```

XML-Anfragesprachen

XML (eXtensible Markup Language)

Finde die Fehler

Attributwert nicht in Anführungszeichen

```
<Uni Name="Alexander Maximilian Universität" Kuerzel=AMU>
  <UniLeitung>
    <Rektor>Max</Rektor>
    <Senatsvorsitzender>Alex</Senat>
  </UniLeitung>
  <Studenten />
  <Student>
    <Name Peter Name>
    <MatrNr>03670815</MatrNr>
    <Vorlesungen>V1<V2<V3</Vorlesungen>
  </Studenten>
</Uni>
```

Öffnender und schließender Tag ungleich

Signalisiert leeren Tag, hat aber Inhalt

Schließender Tag fehlt

Einfach falsch

< und > dürfen nicht einfach im Text vorkommen

XML-Anfragesprachen

XML (eXtensible Markup Language)

Finde die Fehler

```
<Uni Name="Alexander Maximilian Universität" Kuerzel="AMU">
  <UniLeitung>
    <Rektor>Max</Rektor>
    <Senatsvorsitzender>Alex</Senatsvorsitzender>
  </UniLeitung>
  <Studenten>
    <Student>
      <Name>Peter</Name>
      <MatrNr>03670815</MatrNr>
      <Vorlesungen>V1, V2, V3</Vorlesungen>
    </Student>
  </Studenten>
</Uni>
```

XML-Anfragesprachen

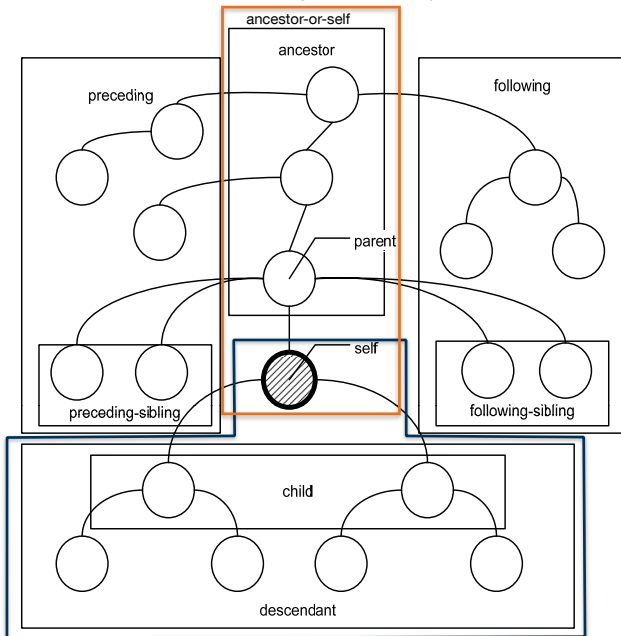
XPath (Finden von Knoten in XML)

Finden von bestimmten Knoten (und allen Nachfahren) im Dokument
Verschiedene Suchachsen zur Navigation durch den Baum
Ausgabe ist Text oder XML-Knoten

Syntax: /Achse::Knotentest[Prädikat]
beliebig oft hintereinander im Ausdruck nutzbar



XML Achsen



XML-Anfragesprachen

Vereinfachte Syntax der XPath-Achsen

<i>Achse</i>	<i>Beschreibung</i>	<i>Abkürzung</i>
<i>self</i>	der Kontextknoten selbst	. (Punkt)
<i>child</i>	direkt untergeordnete Knoten (Kinder)	/
<i>parent</i>	der direkt übergeordnete Elternknoten	.. (2x Punkt)
<i>descendant</i>	untergeordnete Knoten (Nachfahren)	//
<i>attribute</i>	Attributknoten	/@

XML-Anfragesprachen

XPath-Prädikate

Eine beliebige Anzahl kann hintereinander gestellt werden

XPath-Ausdrücke + Funktionen dürfen enthalten sein:

- Knotenindex [i] => i-ter Knoten (Zählung beginnt mit 1)
- Arithmetische Operationen (+, -, *, /, mod)
- Vergleiche (<, >, <=, >=, !=, =)
- logische Operationen (and, or)
- Aggregatsfunktionen (min, max, count)

```
doc('uni2')//ProfessorIn[count(../Vorlesung)=3]
```

XML-Anfragesprachen

XPath-Prädikate

```
<Uni Name="Alexander Maximilian Universität" Kuerzel="AMU">
  <UniLeitung>
    <Rektor>Max</Rektor>
    <Senatsvorsitzender>Alex</Senatsvorsitzender>
  </UniLeitung>
  <Studenten>
    <Student>
      <Name>Peter</Name>
      <MatrNr>03670815</MatrNr>
      <Vorlesungen>V1,V2,V3</Vorlesungen>
    </Student>
  </Studenten>
</Uni>
```

`doc('uniDoc')/Uni/Studenten/Student[Name = 'Peter']/MatrNr`

XML-Anfragesprachen

XPath-Prädikate

```
<Uni Name="Alexander Maximilian Universität" Kuerzel="AMU">
  <UniLeitung>
    <Rektor>Max</Rektor>
    <Senatsvorsitzender>Alex</Senatsvorsitzender>
  </UniLeitung>
  <Studenten>
    <Student>
      <Name>Peter</Name>
      <MatrNr>03670815</MatrNr>
      <Vorlesungen>V1,V2,V3</Vorlesungen>
    </Student>
  </Studenten>
</Uni>
```

doc('uniDoc')/Uni/Studenten/Student[Name = 'Peter']/MatrNr

Ausgabe: <MatrNr>03670815</MatrNr>

XML-Anfragesprachen

XPath-Prädikate

```
<Uni Name="Alexander Maximilian Universität" Kuerzel="AMU">  
  <UniLeitung>  
    <Rektor>Max</Rektor>  
    <Senatsvorsitzender>Alex</Senatsvorsitzender>  
  </UniLeitung>  
  <Studenten>  
    <Student>  
      <Name>Peter</Name>  
      <MatrNr>03670815</MatrNr>  
      <Vorlesungen>V1,V2,V3</Vorlesungen>  
    </Student>  
  </Studenten>  
</Uni>
```

Element

 `doc('uniDoc')//Student[Name = 'Peter']/MatrNr`

XML-Anfragesprachen

XPath-Prädikate

```
<Uni Name="Alexander Maximilian Universität" Kuerzel="AMU">
  <UniLeitung>
    <Rektor>Max</Rektor>
    <Senatsvorsitzender>Alex</Senatsvorsitzender>
  </UniLeitung>
  <Studenten>
    <Student>
      <Name>Peter</Name>
      <MatrNr>03670815</MatrNr>
      <Vorlesungen>V1,V2,V3</Vorlesungen>
    </Student>
  </Studenten>
</Uni>
```

Attribut

 doc('uniDoc')/Uni[@Name = 'Alexander Maximilian Universität']/UniLeitung/Rektor

```
<Rektor>Max</Rektor>
```

XML-Anfragesprachen

XPath-Prädikate

```
<Uni Name="Alexander Maximilian Universität" Kuerzel="AMU">
  <UniLeitung>
    <Rektor>Max</Rektor>
    <Senatsvorsitzender>Alex</Senatsvorsitzender>
  </UniLeitung>
  <Studenten>
    <Student>
      <Name>Peter</Name>
      <MatrNr>03670815</MatrNr>
      <Vorlesungen>V1,V2,V3</Vorlesungen>
    </Student>
  </Studenten>
</Uni>
```

`doc('uniDoc')//Student[Name = 'Peter']/../@Name`

`Name="Alexander Maximilian Universität"`

XML-Anfragesprachen

XQuery

Basiert auf XPath und kombiniert Ergebnisse der Anfragen

FLWOR-Syntax

For Schleifen

Let Variablen definieren

Where Selektieren

Order By Sortieren

Return Ergebnis als neues XML formatieren

XML-Anfragesprachen

XQuery

Es muss nicht die komplette FLWOR Syntax genutzt werden, aber immer wenn FLW oder O genutzt werden braucht man return

Variablen dürfen XML oder Unterabfragen (XPath oder XQuery) enthalten
Alle Variablen beginnen mit \$

Beim Einbetten von XQuery in XML müssen geschweifte Klammern benutzt werden (und auch nur dann)

```
<XML>{XQuery}</XML>
```

XML-Anfragesprachen

XQuery-Beispielsanfrage

```
<Professoren>
  {
    for $p in doc('uni2')//ProfessorIn
      let $v := $p/Vorlesungen/Vorlesung
      where count ($v) > 0
      order by sum ($v/SWS)
      return
        <ProfessorIn>
          {$p/Name}
          <Belastung>{sum($v/SWS)}</
Belastung>
        </ProfessorIn>
  }
</Professoren>
```