XML-Datenmodellierung und Web-Services

XML
- **Datenmodell**
- **Schemabeschreibungssprachen**
- **Anfragesprachen: XPath und XQuery**

Web-Services
- **Überblick**
- **WSDL**
- **UDDI**
- **SOAP**

XML: Extensible Markup Language

Datenmodell

Schemabeschreibung

Anfragesprachen

HTML-Datenmodell

```
<UL>
    <LI> Curie
    <LI> Sokrates
</UL>

<UL>
    <LI> Mäeutik
    <LI> Bioethik
</UL>
```

Kein Schema

Nur Insider können die beiden Listen interpretieren
- Oben: Professoren
- Unten: Vorlesungen

Wenig geeignet als Datenaustauschformat
- Man muß irgendwie dann auch mitschicken, was damit gemeint ist

Verarbeitung von HTM-Daten
- „Screen-scraping“
- Wrapper

Relationales Datenmodell

Professoren			
PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Vorlesungen			
VorlNr	Titel	SWS	Gelesen Von
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

- Schema ist vorgegeben und man kann nur schema-konforme Daten einfügen (Problem Ausnahmen → null-Werte)
- Bedeutung der Daten wird durch das Schema definiert
- Kein Datenaustauschformat

XML-Datenmodell

Liegt irgendwo dazwischen

- HTML
  - Schema-los
  - Beliebige Daten, solange Syntax stimmt
- Relationen
  - Schema
  - Keine Abweichungen

Semi-strukturierte Daten

- Teilweise schematisch
- Aber Ausnahmen
- Wenn Schema, dann muss es eingehalten werden

Unsere Beispiel-Daten in XML ...

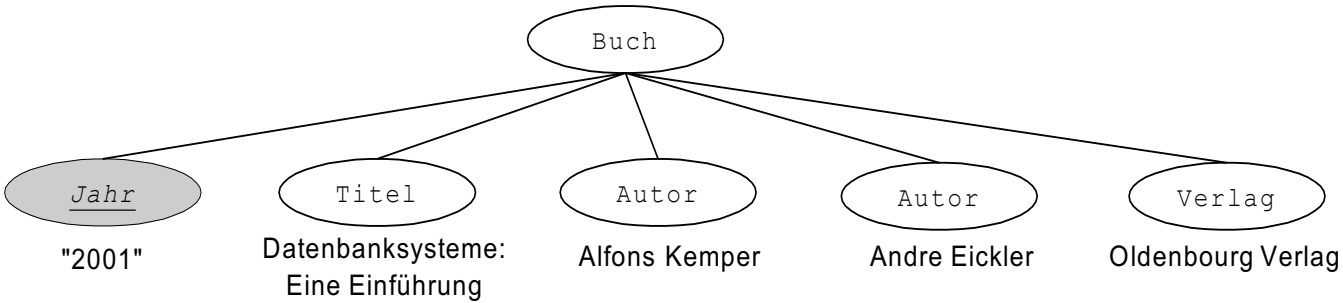
```
<Professoren>
        <ProfessorIn>Curie</ProfessorIn>
        <ProfessorIn>Sokrates</ProfessorIn>
</Professoren>

<Vorlesungen>
        <Vorlesung>Mäeutik</Vorlesung>
        <Vorlesung>Bioethik</Vorlesung>
</Vorlesungen>
```

Semantische Tags (Marken): Geben die Bedeutung der Elemente an, immer paarweise <...> Element </...>

XML-Daten mit Schema (DTD)

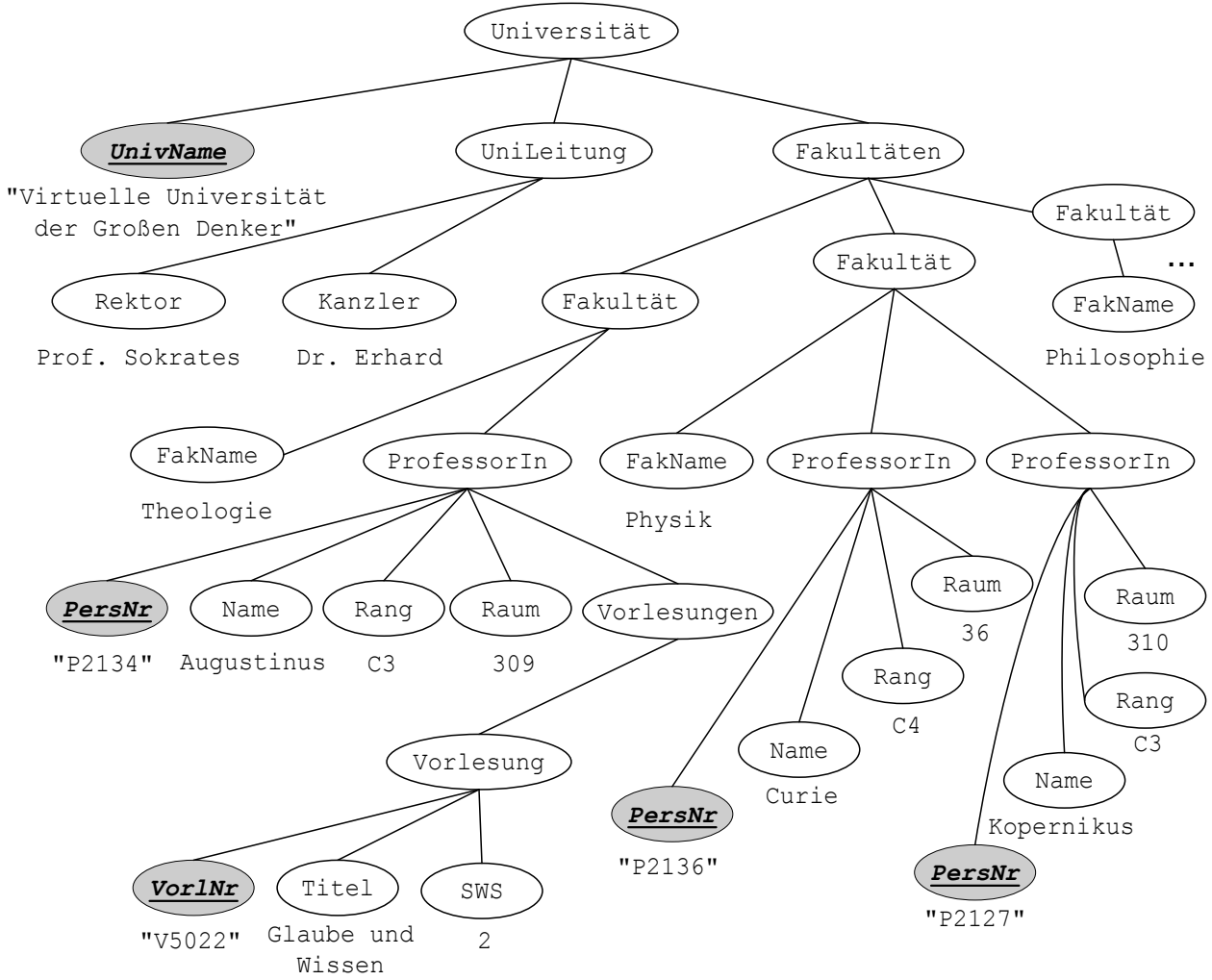
```
<?xml version="1.0" encoding='ISO-8859-1'?>
<!-- obige Zeile ist der Prolog, diese Zeile ist ein Kommentar -->

<!-- Schema als DTD -->
<!DOCTYPE Buch[
   <!ELEMENT Buch (Titel, Autor*, Verlag)>
   <!ATTLIST Buch Jahr CDATA #REQUIRED>
   <!ELEMENT Titel (#PCDATA)>
   <!ELEMENT Autor (#PCDATA)>
   <!ELEMENT Verlag (#PCDATA)>
]>

<!-- Wurzelelement-->
<Buch Jahr="2001">
   <Titel>Datenbanksysteme: Eine Einführung</Titel>
   <Autor>Alfons Kemper</Autor>
   <Autor>Andre Eickler</Autor>
   <Verlag>Oldenbourg Verlag</Verlag>
</Buch>
```

Die hierarchische Struktur im Bild



Rekursive Strukturen

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

<!-- Schema als DTD -->
<!DOCTYPE Bauteil[
   <!ELEMENT Bauteil (Beschreibung, Bauteil*)>
   <!ATTLIST Bauteil Preis CDATA #REQUIRED>
   <!ELEMENT Beschreibung (#PCDATA)>
]>

<!-- Wurzelelement-->
<Bauteil Preis="350000">
   <Beschreibung>Maybach 620 Limousine</Beschreibung>
   <Bauteil Preis="50000">
      <Beschreibung>V12-Biturbo Motor mit 620 PS</Beschreibung>
      <Bauteil Preis="2000">
         <Beschreibung>Nockenwelle</Beschreibung>
      </Bauteil>
    </Bauteil>
    <Bauteil Preis="7000">
      <Beschreibung>Kühlschrank für Champagner</Beschreibung>
    </Bauteil>
</Bauteil>
```


Universität
UnivName
"Virtuelle Universität der Großen Denker"
UniLeitung
Fakultäten
Rektor
Prof. Sokrates
Kanzler
Dr. Erhard
Fakultät
Fakultät
Fakultät
...
FakName
Philosophie
FakName
Theologie
ProfessorIn
FakName
Physik
ProfessorIn
ProfessorIn
PersNr
"P2134"
Name
Augustinus
Rang
C3
Raum
309
Vorlesungen
Raum
36
Raum
310
Rang
C4
Rang
C3
Vorlesung
PersNr
"P2136"
Name
Curie
Name
Kopernikus
VorlNr
"V5022"
Titel
Glaube und Wissen
SWS
2
PersNr
"P2127"

```
<?xml version="1.0" encoding='ISO-8859-1'?>
<Universität UnivName="Virtuelle Universität der Großen Denker">
<UniLeitung>
  <Rektor>Prof. Sokrates</Rektor>
  <Kanzler>Dr. Erhard</Kanzler>
</UniLeitung>
<Fakultäten>
  <Fakultät>
    <FakName>Theologie</FakName>
    <ProfessorIn PersNr="2134">
      <Name>Augustinus</Name>
      <Rang>C3</Rang>
      <Raum>309</Raum>
      <Vorlesungen>
        <Vorlesung VorlNr="5022">
        <Titel>Glaube und Wissen</Titel>
        <SWS>2</SWS>
        </Vorlesung>
      </Vorlesungen>
    </ProfessorIn>
  </Fakultät>
```

XML-Dokument der Universität

```
<Fakultät>
    <FakName>Physik</FakName>
    <ProfessorIn PersNr="2136">
      <Name>Curie</Name>
      <Rang>C4</Rang>
      <Raum>36</Raum>
    </ProfessorIn>
    <ProfessorIn PersNr="2127">
      <Name>Kopernikus</Name>
      <Rang>C3</Rang>
      <Raum>310</Raum>
    </ProfessorIn>
  </Fakultät>

  <Fakultät>
    <FakName>Philosophie</FakName>
    ...
    ...
  </Fakultät>
</Fakultäten>
</Universität>
```

XML Namensräume

```
...
<Universität xmlns="http://www.db.uni-passau.de/Universitaet"
     UnivName="Virtuelle Universität der Großen Denker">
   <UniLeitung>
...
```

XML Namensräume

```
...
<Universität xmlns="http://www.db.uni-passau.de/Universitaet"
    xmlns:lit="http://www.db.uni-passau.de/Literatur"
    UnivName="Virtuelle Universität der Großen Denker">
  <UniLeitung>
...
      <Vorlesung>
        <Titel> Informationssysteme </Titel>
        ...
        <lit:Buch lit:Jahr="2004">
          <lit:Titel>Datenbanksysteme: Eine Einführung</lit:Titel>
          <lit:Autor>Alfons Kemper</lit:Autor>
          <lit:Autor>Andre Eickler</lit:Autor>
          <lit:Verlag>Oldenbourg Verlag</lit:Verlag>
        </lit:Buch>
      </Vorlesung>
...
```

XML-Schema: mächtiger als DTDs

```
<?xml version="1.0" encoding='ISO-8859-1'?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.db.uni-passau.de/Universitaet">

 <xsd:element name="Universität" type="UniInfoTyp"/>

 <xsd:complexType name="UniInfoTyp">
  <xsd:sequence>
   <xsd:element name="UniLeitung">
    <xsd:complexType>
     <xsd:sequence>
      <xsd:element name="Rektor" type="xsd:string"/>
      <xsd:element name="Kanzler" type="xsd:string"/>
     </xsd:sequence>
    </xsd:complexType>
   </xsd:element>
  <xsd:element name="Fakultäten">
```

XML-Schema: mächtiger als DTDs

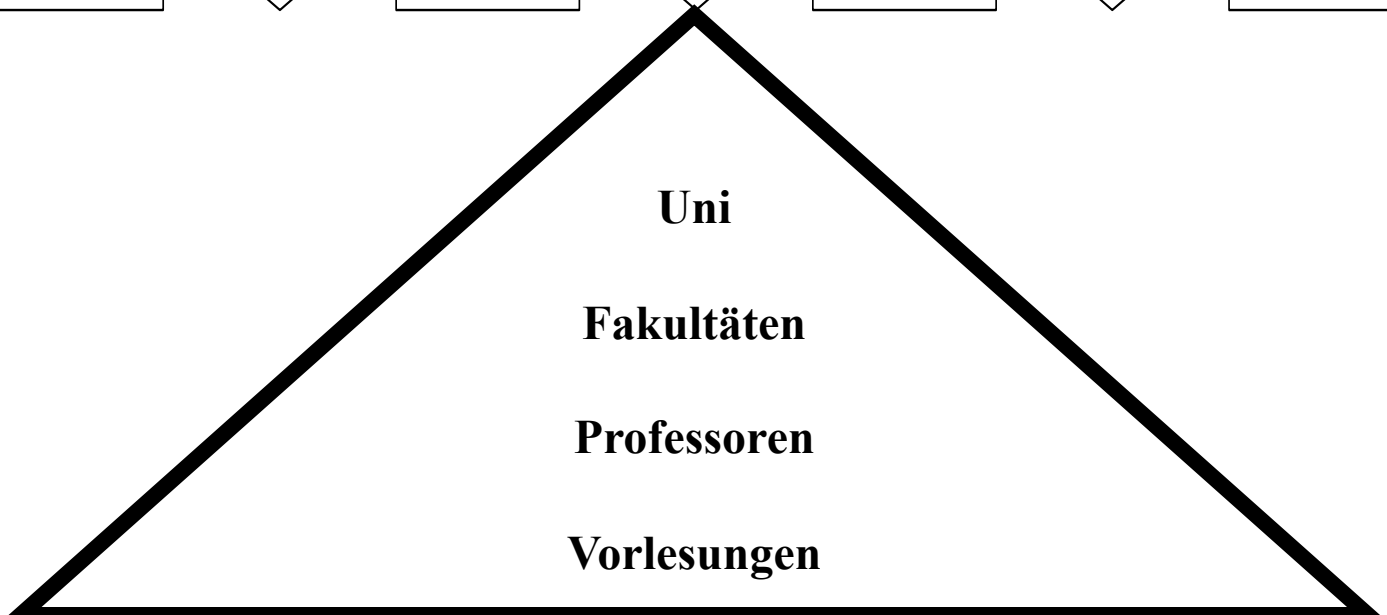
```
<xsd:element name="Fakultäten">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Fakultät"
                   minOccurs="0"
                   maxOccurs="unbounded"
                   type="Fakultätentyp"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
</xsd:sequence>
  <xsd:attribute name="UnivName" type="xsd:string"/>
 </xsd:complexType>
```

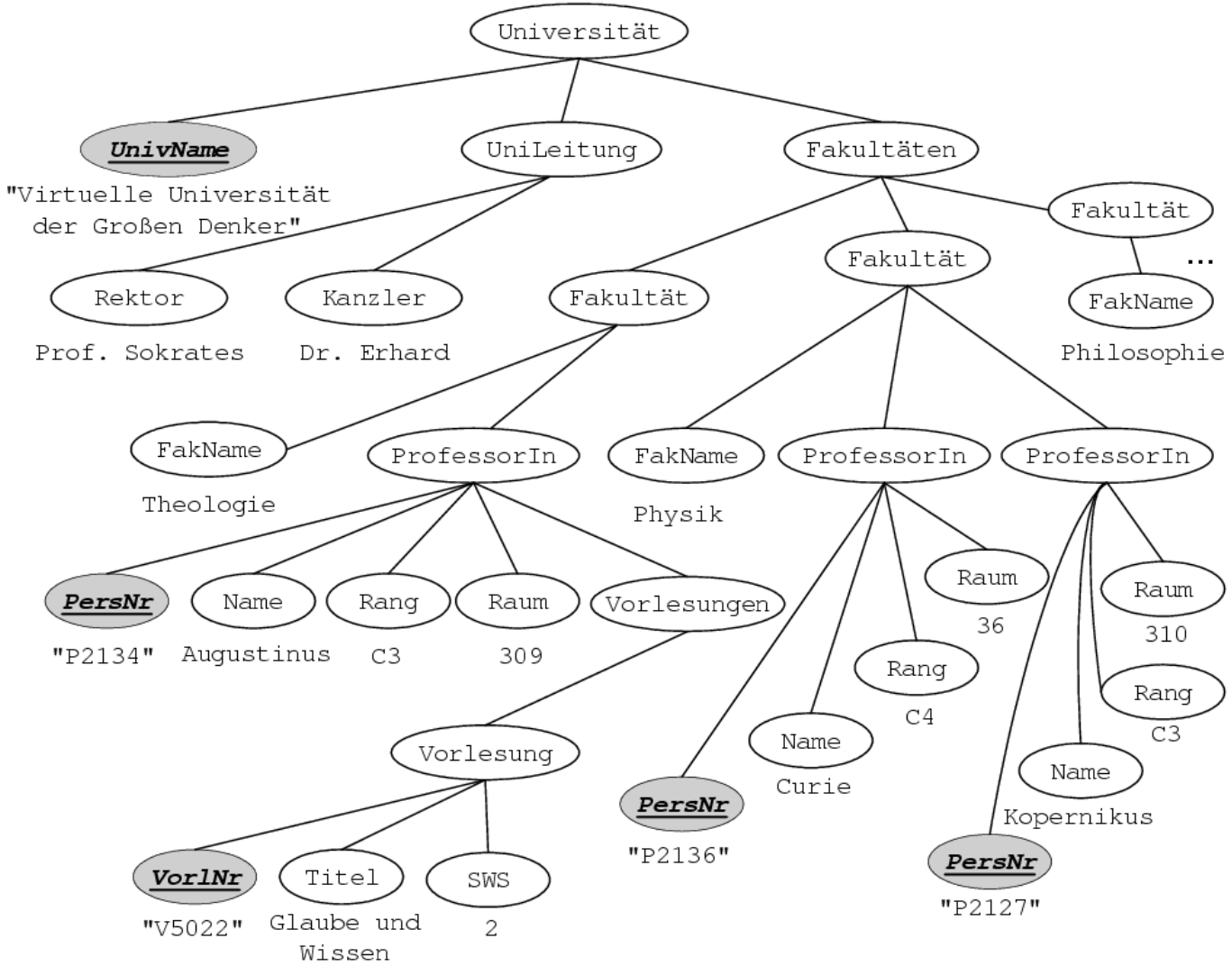
```
<xsd:complexType name="FakultätenTyp">   <xsd:sequence>
    <xsd:element name="FakName" type="xsd:string"/>
    <xsd:element name="ProfessorIn" minOccurs="0" maxOccurs="unbounded">
     <xsd:complexType>
      <xsd:sequence>
       <xsd:element name="Name" type="xsd:string"/>
       <xsd:element name="Rang" type="xsd:string"/>
       <xsd:element name="Raum" type="xsd:integer"/>
       <xsd:element name="Vorlesungen" minOccurs="0" type="VorlInfo"/>
      </xsd:sequence>
      <xsd:attribute name="PersNr" type="xsd:ID"/>
     </xsd:complexType>
    </xsd:element>
 </xsd:sequence>   </xsd:complexType>
```

```
<xsd:complexType name="VorlInfo">   <xsd:sequence>
  <xsd:element name="Vorlesung" minOccurs="1" maxOccurs="unbounded">
   <xsd:complexType>
    <xsd:sequence>
     <xsd:element name="Titel" type="xsd:string"/>
     <xsd:element name="SWS" type="xsd:integer"/>
    </xsd:sequence>
    <xsd:attribute name="VorlNr" type="xsd:ID"/>
    <xsd:attribute name="Voraussetzungen" type="xsd:IDREFS"/>
   </xsd:complexType>
  </xsd:element>
</xsd:sequence>   </xsd:complexType>

</xsd:schema>
```

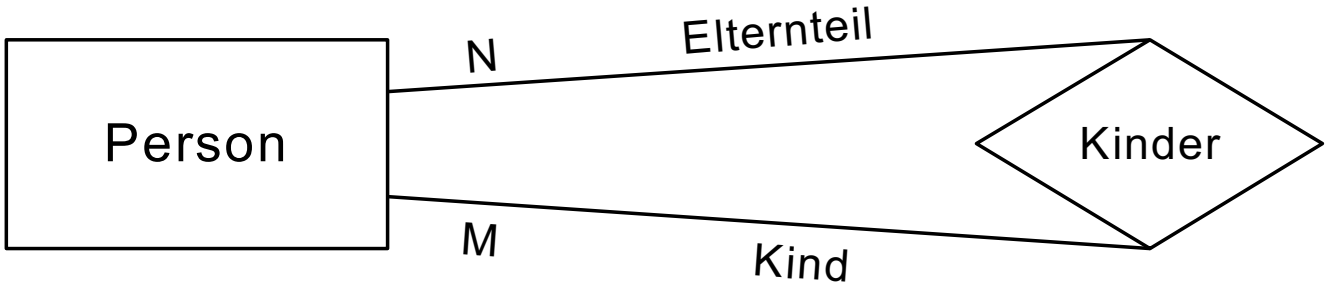
Verweise in XML-Dokumenten

- XML ist „super“ für die Modellierung von Hierarchien
- Entsprechen den geschachtelten Elementen
- Genau das hatten wir in dem Uni-Beispiel




Universität
UnivName
"Virtuelle Universität der Großen Denker"
UniLeitung
Fakultäten
Rektor
Prof. Sokrates
Kanzler
Dr. Erhard
Fakultät
Fakultät
Fakultät
...
FakName
Philosophie
FakName
Theologie
ProfessorIn
FakName
Physik
ProfessorIn
ProfessorIn
PersNr
"P2134"
Name
Augustinus
Rang
C3
Raum
309
Vorlesungen
Raum
36
Raum
310
Rang
C4
Rang
C3
Vorlesung
Name
Curie
Name
Kopernikus
PersNr
"P2136"
PersNr
"P2127"
VorlNr
"V5022"
Titel
Glaube und Wissen
SWS
2

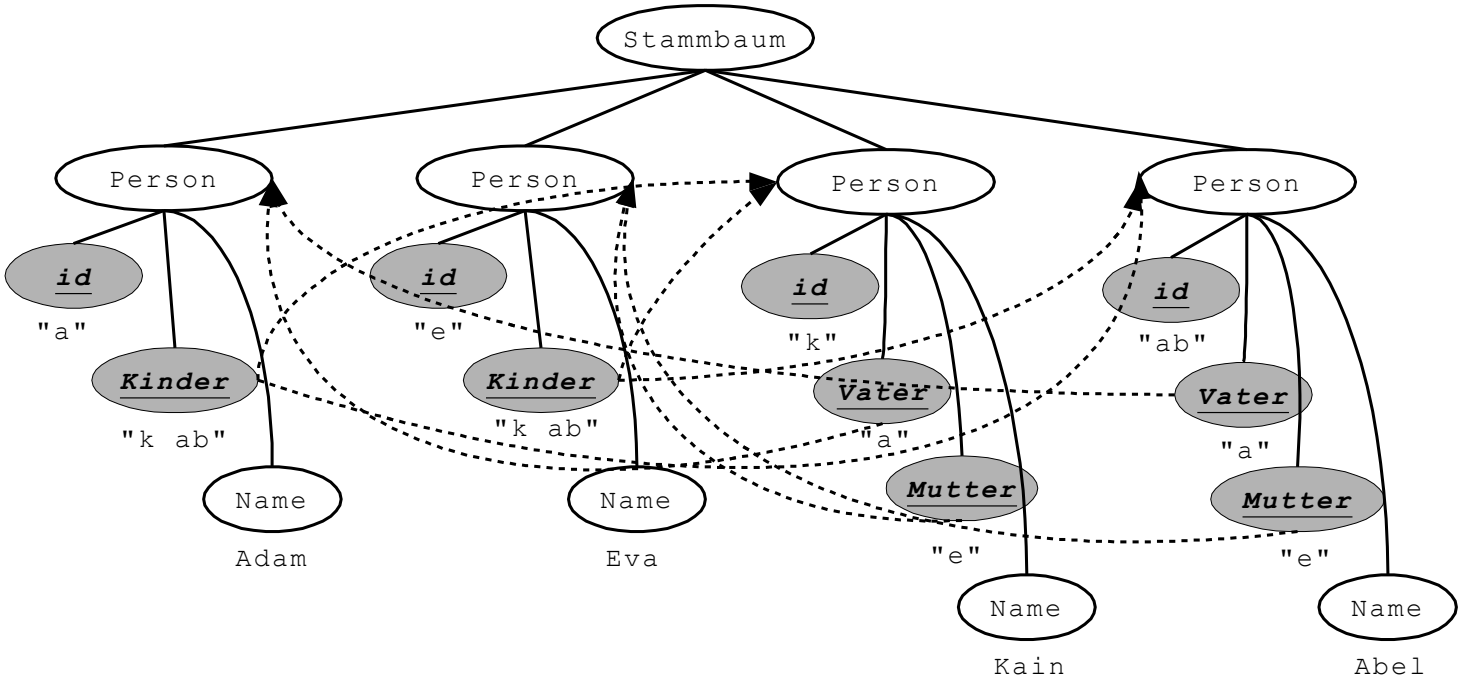
Modellierung des Stammbaums



Kinder haben zwei Elternteile

- Also kann man ein Kind nicht mehr als Sub-Element einer Person modellieren
- Wem soll man es zuordnen (Vater oder Mutter)
- Also muss man mit Verweisen (IDREF und IDREFS) „arbeiten“

Graphische Darstellung des XML-Dokuments



Familie.xml

```
<!DOCTYPE Stammbaum [
   <!ELEMENT Stammbaum (Person*)>
   <!ELEMENT Person (Name)>
   <!ELEMENT Name (#PCDATA)>
        <!ATTLIST Person id      ID     #REQUIRED
                         Mutter  IDREF  #IMPLIED
                         Vater   IDREF  #IMPLIED
                         Kinder  IDREFS #IMPLIED> ]>

<Stammbaum>
  <Person id="a" Kinder="k ab">
       <Name>Adam</Name> </Person>
  <Person id="e" Kinder="k ab">
      <Name>Eva</Name> </Person>
  <Person id="k" Mutter="e" Vater="a">
       <Name>Kain</Name> </Person>
  <Person id="ab" Mutter="e" Vater="a">
        <Name>Abel</Name> </Person>
</Stammbaum>
```

XML-Anfragesprache XQuery

Basiert auf Xpath, einer Sprache für Pfadausdrücke
Ein Lokalisierungspfad besteht aus einzelnen Lokalisierungsschritten
Jeder Lokalisierungsschritt besteht aus bis zu drei Teilen:

- Achse::Knotentest[Prädikat]

Folgende Achsen gibt es:

self: Hierbei handelt es sich um den Referenzknoten.

attribute: Hierunter fallen alle Attribute des Referenzknotens -- falls er überhaupt welche besitzt.

child: Entlang dieser Achse werden alle direkten Unterelemente bestimmt.

descendant: Hierunter fallen alle direkten und indirekten Unterelemente, also die Kinder und deren Kinder u.s.w.

descendant-or-self: Wie oben, außer dass der Referenzknoten hier auch dazu gehört.

XPath-Achsen

self: Hierbei handelt es sich um den Referenzknoten.

attribute: Hierunter fallen alle Attribute des Referenzknotens -- falls er überhaupt welche besitzt.

child: Entlang dieser Achse werden alle direkten Unterelemente bestimmt.

descendant: Hierunter fallen alle direkten und indirekten Unterelemente, also die Kinder und deren Kinder u.s.w.

descendant-or-self: Wie oben, außer dass der Referenzknoten hier auch dazu gehört.

parent: Der Vaterknoten des Referenzknotens wird über diese Achse ermittelt.

ancestor: Hierzu zählen alle Knoten auf dem Pfad vom Referenzknoten zur Wurzel des XML-Baums.

Achsen … cont‘d

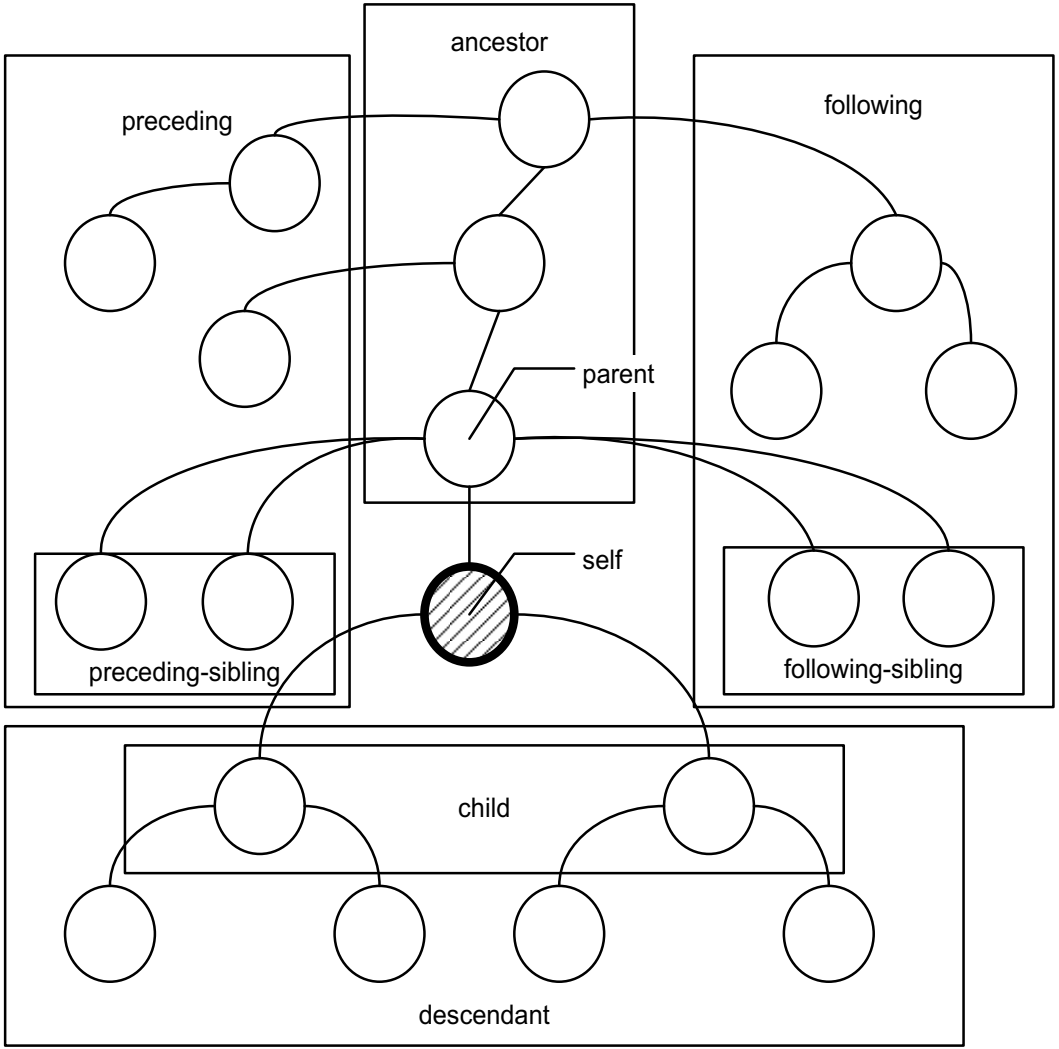
ancestor-or-self: Wie oben, außer dass der Referenzknoten auch mit eingeschlossen wird.

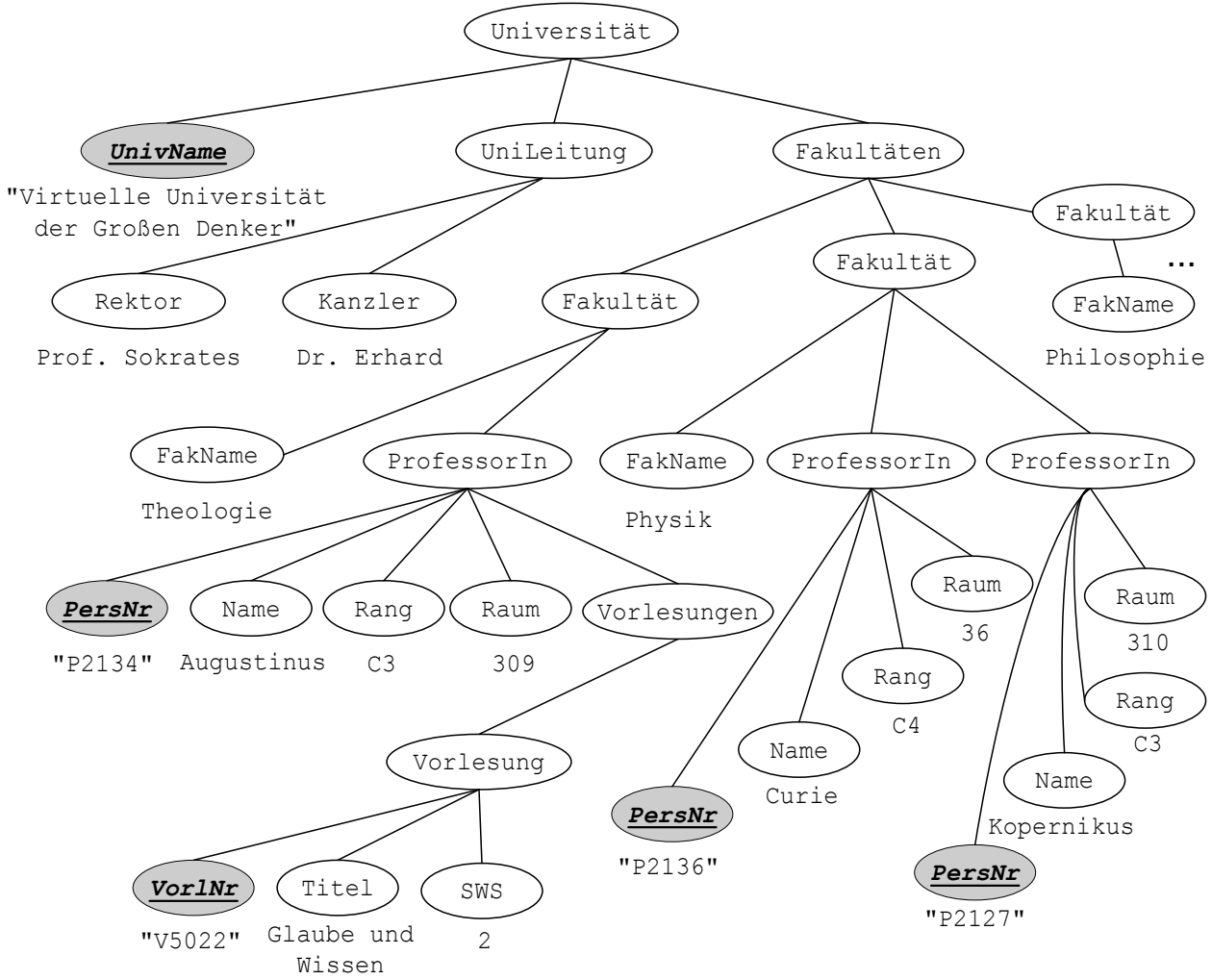
following-sibling: Dies sind die in Dokumentreihenfolge nachfolgenden Kinder des Elternknotens von self.

preceding-sibling: Hierbei handelt es sich um die in Dokumentreihenfolge vorangehenden Kinder des Elternknotens von self.

following: Alle Knoten, die in Dokumentreihenfolge nach dem Refernzknoten aufgeführt sind. Die Nachkommen (descendant) des Referenzknotens gehören aber nicht dazu.

preceding: Alle Knoten, die im Dokument vor dem Referenzknoten vorkommen -- allerdings ohne die Vorfahren (ancestor).


ancestor
preceding
following
parent
self
preceding-sibling
following-sibling
child
descendant


Universität
UnivName
"Virtuelle Universität der Großen Denker"
UniLeitung
Fakultäten
Rektor
Prof. Sokrates
Kanzler
Dr. Erhard
Fakultät
Fakultät
Fakultät
...
FakName
Philosophie
FakName
Theologie
ProfessorIn
FakName
Physik
ProfessorIn
ProfessorIn
PersNr
"P2134"
Name
Augustinus
Rang
C3
Raum
309
Vorlesungen
Raum
36
Raum
310
Rang
C4
Rang
C3
Vorlesung
PersNr
"P2136"
Name
Curie
Name
Kopernikus
VorlNr
"V5022"
Titel
Glaube und Wissen
SWS
2
PersNr
"P2127"

XPath-Ausdrücke

doc("Uni.xml")/child::Universität[self::*/attribute::UnivName=
"Virtuelle Universität der Großen Denker"]
Als Ergebnis bekommt man den gesamten Baum unseres Beispieldokuments

doc("Uni.xml")/child::Universität/child::Fakultäten/child::Fakultät/
child::FakName

<FakName>Theologie</FakName>
<FakName>Physik</FakName>
<FakName>Philosophie</FakName>

Äquivalent für unser Beispiel:
- doc("Uni.xml")/descendant-or-self::FakName

doc("Uni.xml")/child::Universität/attribute::UnivName

UnivName="Virtuelle Universität der Großen Denker„

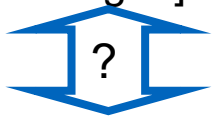
```
doc("Uni.xml")/child::Universität/child::Fakultäten/
        child::Fakultät[self::*/child::FakName="Theologie"]/
            descendant-or-self::Vorlesung/child::Titel
```

```
doc("Uni.xml")/child::Universität/child::Fakultäten/
      child::Fakultät[child::FakName="Philosophie"]/
           child::ProfessorIn[child::Rang="C4"]/child::Vorlesungen/
                child::Vorlesung/child::Titel
```

```
<Titel>Ethik</Titel><Titel>Mäeutik</Titel><Titel>Logik</Titel>
<Titel>Erkenntnistheorie</Titel><Titel>Wissenschaftstheorie</Titel>
<Titel>Bioethik</Titel><Titel>Grundzüge</Titel><Titel>Die 3 Kritiken</Titel>
```

```
doc("Uni.xml")/child::Universität/child::Fakultäten/
        child::Fakultät/child::FakName[parent::Fakultät/
                child::ProfessorIn/child::Vorlesungen]

<FakName>Theologie</FakName> <FakName>Philosophie</FakName>
```

```
doc("Uni.xml")/child::Universität/child::Fakultäten/
        child::Fakultät[/descendant::Vorlesungen]/child::FakName
```




```
doc("Uni.xml")/child::Universität/child::Fakultäten/
          child::Fakultät[descendant::Vorlesungen]/child::FakName
```

```
doc("Uni.xml")/child::Universität/child::Fakultäten/
                        child::Fakultät[position()=2]
```

wird also die zweite Fakultät ausgegeben:

```
<Fakultät>
   <FakName>Physik</FakName>
   <ProfessorIn PersNr="P2136">
      <Name>Curie</Name>
      <Rang>C4</Rang>
      <Raum>36</Raum>
   </ProfessorIn>
   <ProfessorIn PersNr="P2127">
      <Name>Kopernikus</Name>
      <Rang>C3</Rang>
      <Raum>310</Raum>
   </ProfessorIn>
</Fakultät>
```

```
doc("Uni.xml")/child::Universität/child::Fakultäten/
      child::Fakultät[child::ProfessorIn/child::Vorlesungen/
            child::Vorlesung/child::Titel="Mäeutik"]/child::FakName
```

```
<FakName>Philosophie</FakName>
```

Verkürzte Syntax

. Aktueller Referenzknoten

.. Vaterknoten

/ Abgrenzung einzelner Schritte oder Wurzel

ElemName1/ElemName2/ElemName3

// descendant-or-self::node()

@AttrName Attributzugriff

```
doc("Uni.xml")/Universität/Fakultäten/
                Fakultät[FakName="Physik"]//Vorlesung

doc("Uni.xml")/Universität/Fakultäten/
                Fakultät[position()=2]//Vorlesung

doc("Uni.xml")/Universität/Fakultäten/Fakultät[2]//Vorlesung


doc("Uni.xml")/Universität/Fakultäten/Fakultät[FakName="Physik"]/
                        ProfessorIn/Vorlesungen/Vorlesung
```

Beispiel-Pfadausdrücke

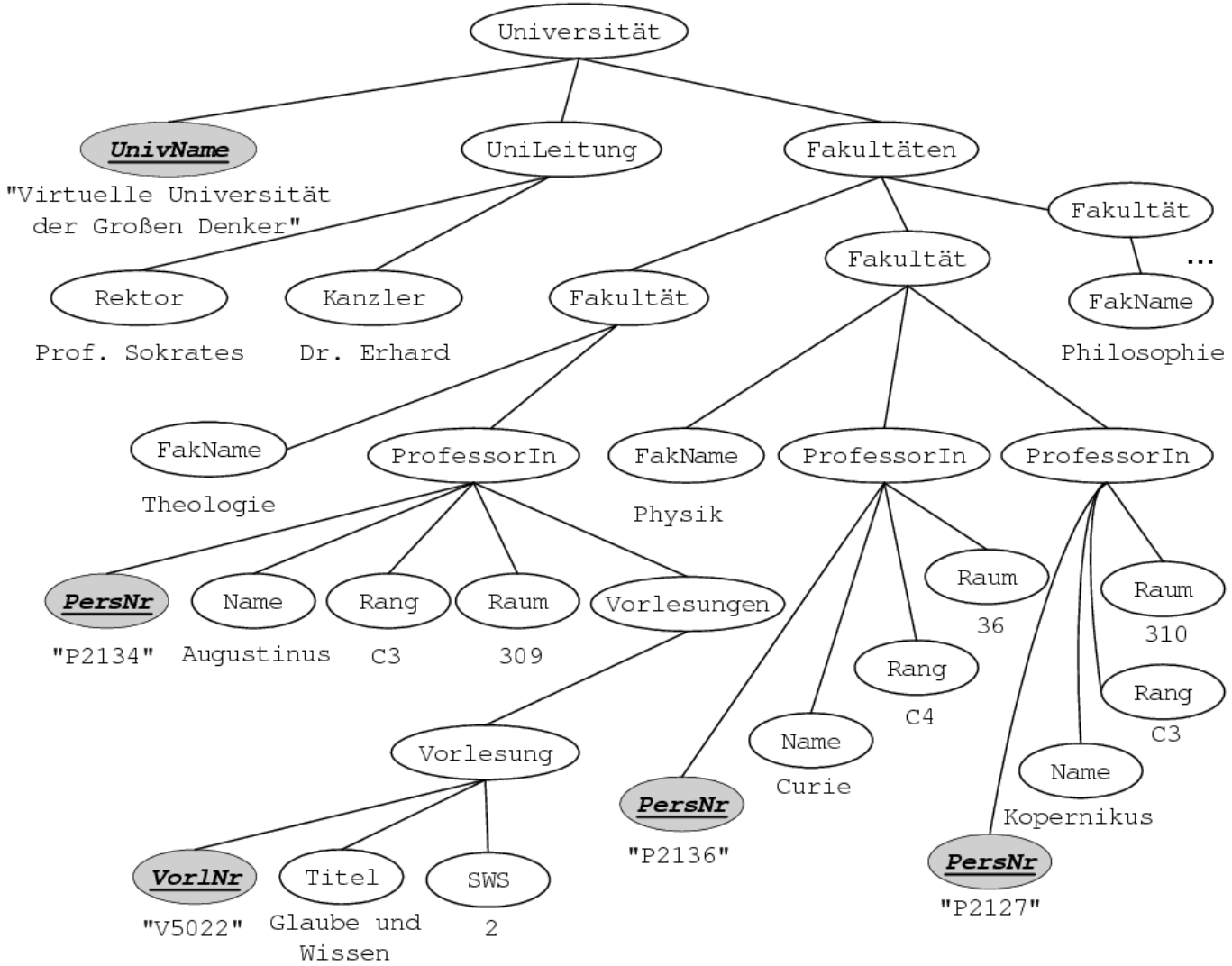
document("uni.xml")/Fakultäten/Fakultät[FakName="Physik"]
//Vorlesung

document("uni.xml")/Fakultäten/Fakultät[2]//Vorlesung

document("uni.xml")/Fakultäten/Fakultät[FakName="Physik"]/
ProfessorIn/Vorlesungen/Vorlesung

document("Stammbaum.xml")/Person[Name="Kain"]
/@Vater->/Name

document("uni.xml")//Vorlesung[Titel="Mäeutik"]/
@Voraussetzungen->/Titel


Universität
UnivName
"Virtuelle Universität der Großen Denker"
UniLeitung
Fakultäten
Rektor
Prof. Sokrates
Kanzler
Dr. Erhard
Fakultät
Fakultät
Fakultät
...
FakName
Philosophie
FakName
Theologie
ProfessorIn
FakName
Physik
ProfessorIn
ProfessorIn
PersNr
"P2134"
Name
Augustinus
Rang
C3
Raum
309
Vorlesungen
Raum
36
Raum
310
Rang
C4
Rang
C3
Vorlesung
Name
Curie
Name
Kopernikus
PersNr
"P2136"
PersNr
"P2127"
VorlNr
"V5022"
Titel
Glaube und Wissen
SWS
2

XQuery-Anfragsyntax

FLOWOR-Ausdrücke

- For ..
- Let …
- Where …
- Order by …
- Return …

XML-Beispielanfrage

```
<Vorlesungsverzeichnis>
    {for $v in doc("Uni.xml")//Vorlesung
     return
         $v}
</Vorlesungsverzeichnis>
```

```
<VorlesungsVerzeichnis>
    <Vorlesung VorlNr=„V5022">
        <Titel>Glaube und Wissen</Titel>
        <SWS>2</SWS>
    </Vorlesung>
    ...
</VorlesungsVerzeichnis>
```

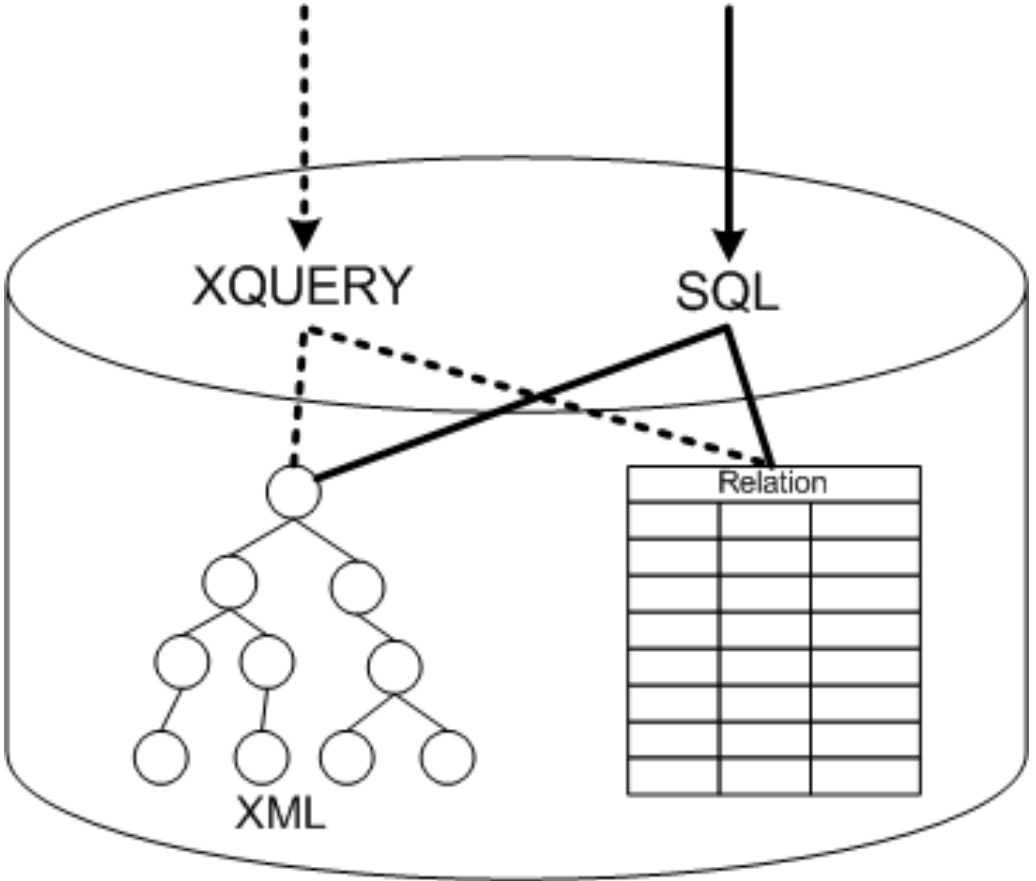
Xquery in DB2

```
select xmlquery(
‘
for $f in $d//Fakultäten/Fakultät
let $v:=$f//Vorlesung
where count($v) > 1
return <PhysikProfessoren>
                {$f//ProfessorIn}
</PhysikProfessoren>
'
passing u.doc as "d")
from Unis u
where u.Name = 'VirtU'
```

Unis

Name	doc
VirtU	<Universität> <UniLeitung> … </UniLeitung> <Fakultäten> <Fakultät> … </Fakultät> …. </Universität>
TUM	<Universität> … </Universität>

XML/Relationen-Koexistenz in DB2



Natives XQUERY

```
xquery db2-fn:xmlcolumn('UNI.UNIS.DOC')
```

Relationale Sicht aus XML

```
create view UniProfsVorls (Name, ProfName, VorlTitel) as
        select u.Name, t.Name, t.Titel
        from UNI.UNIS u,
        xmltable('$d//ProfessorIn' passing u.DOC as "d"
        columns Name varchar(20) path 'Name',
          Titel varchar(20) path 'Vorlesungen/Vorlesung[1]/Titel') as t
```

VirtU	Augustinus	Glaube und Wissen
VirtU	Curie	
VirtU	Kopernikus	
VirtU	Sokrates	Ethik
VirtU	Russel	Erkenntnistheorie
VirtU	Popper	Der Wiener Kreis
VirtU	Kant	Grundzüge

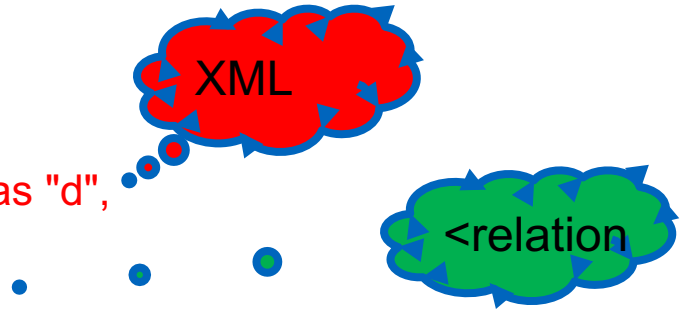
Sicht: Alle Vorlesungen

```
create view UniVorls (Name, ProfName, VorlTitel) as
            select u.Name, t.Name, t.Titel
            from UNI.UNIS u,
            xmltable('$d//ProfessorIn/Vorlesungen/Vorlesung' passing u.DOC as "d"
            columns Name varchar(20) path './../../Name',
              Titel varchar(20) path 'Titel') as t;
```

```
VirtU Augustinus Glaube und Wissen
VirtU Sokrates Ethik
VirtU Sokrates Mäeutik
VirtU Sokrates Logik
VirtU Russel Erkenntnistheorie
VirtU Russel Wissenschaftstheorie
VirtU Russel Bioethik
VirtU Popper Der Wiener Kreis
VirtU Kant Grundzüge
VirtU Kant Die 3 Kritiken
```

Join zwischen Relationen und XML

```
select  xmlquery('for $p in $d//ProfessorIn
                                    where $p/Name = $profN
                                    return $p' passing u.doc as "d",
                                  prof.Name as "profN")
from UNI.UNIS u, UNI.Professoren prof, Uni.prüfen ex
where prof.PersNr = ex.PersNr  and ex.Note < 3.0
```



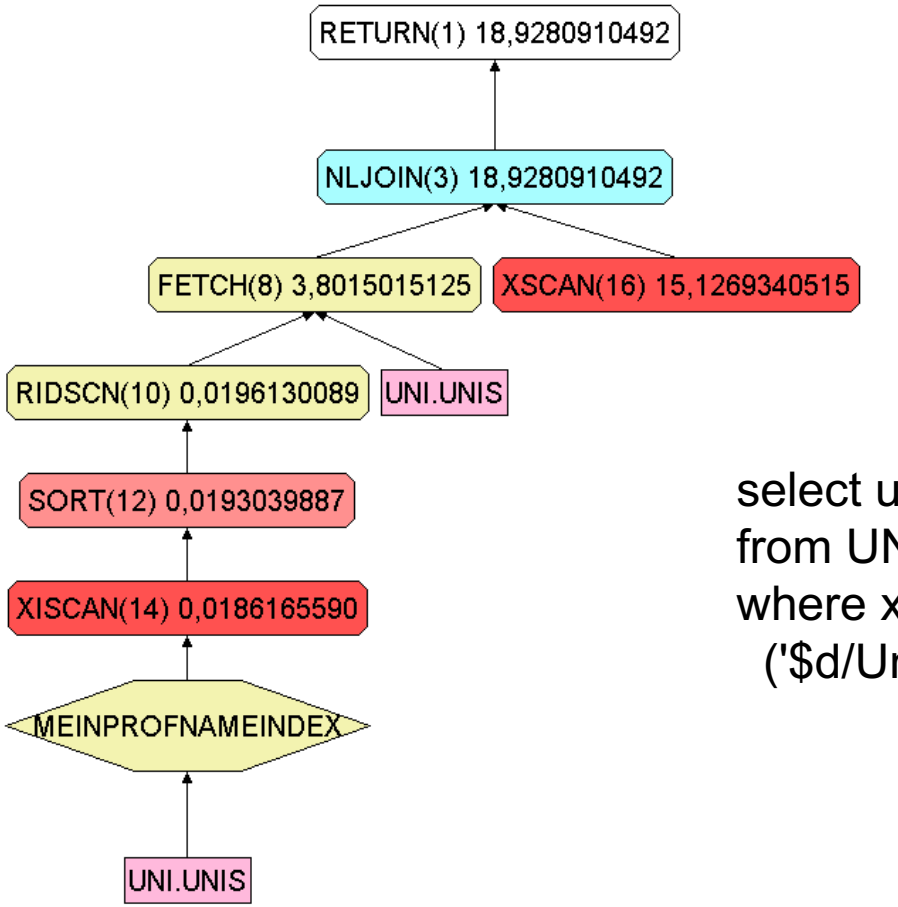
In welcher Uni arbeitet Sokrates: **xmlexists**()

```
select u.Name
from UNI.UNIS u
where xmlexists('$d//ProfessorIn[Name="Sokrates"]'
                passing u.DOC as “d")
```

Index auf XML-Elemente

```
create index meinProfNameIndex on UNI.UNIS(DOC)
        generate key using xmlpattern
    '/Universität/Fakultäten/Fakultät/ProfessorIn/Name'
        as sql varchar(20)
```

Nutzung des Index‘




```
select u.Name
from UNI.UNIS u
where xmlexists
  ('$d/Universität/Fakultäten/Fakultät/
                ProfessorIn[Name="Sokrates"]'
           passing u.DOC as "d")
```

if … then … else

```
xquery
<ProfessorenListe>
{for $p in db2-fn:xmlcolumn('UNI.UNIS.DOC')//ProfessorIn
return (
        if ($p/Vorlesungen/Vorlesung[2]) then
                <LehrProfessorIn>
                        {$p/Name/text()}
                </LehrProfessorIn>
        else
                <ForschungsProfessorIn>
                        {$p/Name/text()}
                </ForschungsProfessorIn>
        )
}
</ProfessorenListe>
```

Ergebnis

```
<ProfessorenListe>
<ForschungsProfessorIn>
        Augustinus
</ForschungsProfessorIn>
<ForschungsProfessorIn>
        Curie
</ForschungsProfessorIn>
<ForschungsProfessorIn>
        Kopernikus
</ForschungsProfessorIn>
<LehrProfessorIn>
        Sokrates
</LehrProfessorIn>
<LehrProfessorIn>
        Russel
</LehrProfessorIn>
<ForschungsProfessorIn>
        Popper
</ForschungsProfessorIn>
<LehrProfessorIn>
        Kant
</LehrProfessorIn>
</ProfessorenListe>
```

```
<Vorlesungsverzeichnis>
   <Vorlesung VorlNr="V5022">
      <Titel>Glaube und Wissen</Titel>
      <SWS>2</SWS>
   </Vorlesung>
   <Vorlesung Voraussetzungen="V5001" VorlNr="V5041">
      <Titel>Ethik</Titel>
      <SWS>4</SWS>
   </Vorlesung>
   <Vorlesung Voraussetzungen="V5001" VorlNr="V5049">
      <Titel>Mäeutik</Titel>
      <SWS>2</SWS>
   </Vorlesung>
   <Vorlesung VorlNr="V4052">
      <Titel>Logik</Titel>
      <SWS>4</SWS>
   </Vorlesung>
   <Vorlesung Voraussetzungen="V5001" VorlNr="V5043">
      <Titel>Erkenntnistheorie</Titel>
      <SWS>3</SWS>
   </Vorlesung>
   <Vorlesung Voraussetzungen="V5043 V5041" VorlNr="V5052">
      <Titel>Wissenschaftstheorie</Titel>
      <SWS>3</SWS>
   </Vorlesung>
   <Vorlesung Voraussetzungen="V5041" VorlNr="V5216">
      <Titel>Bioethik</Titel>
      <SWS>2</SWS>
   </Vorlesung>
   <Vorlesung Voraussetzungen="V5052" VorlNr="V5259">
      <Titel>Der Wiener Kreis</Titel>
      <SWS>2</SWS>
   </Vorlesung>
   <Vorlesung VorlNr="V5001">
      <Titel>Grundzüge</Titel>
      <SWS>4</SWS>
   </Vorlesung>
   <Vorlesung VorlNr="V4630">
      <Titel>Die 3 Kritiken</Titel>
      <SWS>4</SWS>
   </Vorlesung>
</Vorlesungsverzeichnis>
```

XML-Beispielanfrage

```
<Vorlesungsverzeichnis>
    {for $v in doc("Uni.xml")//Vorlesung[SWS=4]
     return
         $v}
</Vorlesungsverzeichnis>
<Vorlesungsverzeichnis>
    {for $v in doc("Uni.xml")//Vorlesung
     where $v/SWS = 4
     return
         $v}
</Vorlesungsverzeichnis>
```

XML-Beispielanfrage

```
<VorlesungsVerzeichnisNachFakultät>
    {for $f in doc("Uni.xml")/Universität/Fakultäten/Fakultät
    return
     <Fakultät>
        <FakultätsName>{$f/FakName/text()}</FakultätsName>
        {for $v in $f/ProfessorIn/Vorlesungen/Vorlesung
         return $v}
     </Fakultät>}
</VorlesungsVerzeichnisNachFakultät>
```

Joins in XQuery

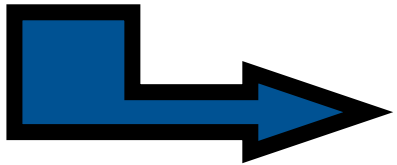
```
<MäeutikVoraussetzungen>
    {for $m in doc("Uni.xml")//Vorlesung[Titel="Mäeutik"],
         $v in doc("Uni.xml")//Vorlesung
     where contains($m/@Voraussetzungen,$v/@VorlNr)
     return $v/Titel}
</MäeutikVoraussetzungen>

<MäeutikVoraussetzungen>
    <Titel>Grundzüge</Titel>
</MäeutikVoraussetzungen>
```

XML-Beispielanfrage

```
<ProfessorenStammbaum>
    {for $p in doc("Uni.xml")//ProfessorIn,
        $k in doc("Stammbaum.xml")//Person,
        $km in doc("Stammbaum.xml")//Person,
        $kv in doc("Stammbaum.xml")//Person
    where $p/Name = $k/Name and $km/@id = $k/@Mutter and
          $kv/@id = $k/@Vater
    return
     <ProfMutterVater>
       <ProfName>{$p/Name/text()}</ProfName>
       <MutterName>{$km/Name/text()}</MutterName>
       <VaterName>{$kv/Name/text()}</VaterName>
     </ProfMutterVater> }
</ProfessorenStammbaum>
```

XML-Beispielanfrage

```
<GefährdetePersonen>
    {for $p in doc("Stammbaum.xml")//Person[Name = "Kain"],
        $g in doc("Stammbaum.xml")//Person[
                @Vater = $p/@Vater and @Mutter = $p/@Mutter]
    return $g/Name }
</GefährdetePersonen>
```



```
<GefährdetePersonen>
    <Name>Kain</Name>
    <Name>Abel</Name>
</GefährdetePersonen>
```

Das LET-Konstrukt

for $x in (1,2) return <zahl> {$x} </zahl>
liefert als Ergebnis:
<zahl>1</zahl> <zahl>2</zahl>
Andererseits liefert
let $x := (1,2) return <zahl> {$x} </zahl>
das Ergebnis

<zahl>12</zahl>

Welche Fakultäten bieten so viele Vorlesungen an wie die Theologie

```
let $TheolVorls := doc("Uni.xml")//Fakultät
                                [FakName ="Theologie"]//Vorlesung
for $f in doc("Uni.xml")//Fakultät
let $fVorls := $f//Vorlesung
where count($fVorls) >= count($TheolVorls)
return $f/FakName
```

```
<FakName>Theologie</FakName>
<FakName>Philosophie</FakName>
```

XML-Beispielanfrage

```
<Professoren>
   {for $p in doc("Uni.xml")//ProfessorIn
    let $v := $p/Vorlesungen/Vorlesung
    where count($v) > 1
    order by sum($v/SWS)
    return
     <ProfessorIn>
        {$p/Name}
        <Lehrbelastung>{sum($v/SWS)}</Lehrbelastung>
     </ProfessorIn>
    }
</Professoren>
```

Ergebnis der XML-Beispielanfrage

```
<Professoren>
   <ProfessorIn>
      <Name>Russel</Name>
      <Lehrbelastung>8.0</Lehrbelastung>
   </ProfessorIn>
   <ProfessorIn>
      <Name>Kant</Name>
      <Lehrbelastung>8.0</Lehrbelastung>
   </ProfessorIn>
   <ProfessorIn>
      <Name>Sokrates</Name>
      <Lehrbelastung>10.0</Lehrbelastung>
   </ProfessorIn>
</Professoren>
```

Dereferenzierung durch wert-basierten Join

```
<VorlesungsBaum>
{for $p in doc("Uni.xml")//Vorlesung
return
   <Vorlesung Titel="{ $p/Titel/text() }">
      {for $s in doc("Uni.xml")//Vorlesung
        where contains($p/@Voraussetzungen,$s/@VorlNr)
        return <Vorlesung Titel="{ $s/Titel/text() }"> </Vorlesung> }
   </Vorlesung> }
</VorlesungsBaum>
```

Dereferenzierung durch id()-Funktion

```
<VorlesungsBaum>
{for $p in doc("Uni.xml")//Vorlesung
return
   <Vorlesung Titel="{ $p/Titel/text() }">
      {for $s in id($p/@Voraussetzungen)
       return <Vorlesung Titel="{ $s/Titel/text() }"> </Vorlesung> }
   </Vorlesung> }
</VorlesungsBaum>
```

Ergebnis

```
<VorlesungsBaum>
   <Vorlesung Titel="Glaube und Wissen"/>
   <Vorlesung Titel="Ethik">
      <Vorlesung Titel="Grundzüge"/>
   </Vorlesung>
   <Vorlesung Titel="Mäeutik">
      <Vorlesung Titel="Grundzüge"/>
   </Vorlesung>
   <Vorlesung Titel="Logik"/>
   <Vorlesung Titel="Erkenntnistheorie">
      <Vorlesung Titel="Grundzüge"/>
   </Vorlesung>
   <Vorlesung Titel="Wissenschaftstheorie">
      <Vorlesung Titel="Ethik"/>
      <Vorlesung Titel="Erkenntnistheorie"/>
   </Vorlesung>
   <Vorlesung Titel="Bioethik">
      <Vorlesung Titel="Ethik"/>
   </Vorlesung>
   <Vorlesung Titel="Der Wiener Kreis">
      <Vorlesung Titel="Wissenschaftstheorie"/>
   </Vorlesung>
   <Vorlesung Titel="Grundzüge"/>
   <Vorlesung Titel="Die 3 Kritiken"/>
</VorlesungsBaum>
```

Rekursion … einfach

```
for $m in doc("Bauteile.xml")/Bauteil
                  [Beschreibung="Maybach 620 Limousine"]
let $teile := $m//Bauteil
return
  <Kosten>
   <Verkaufspreis> {$m/@Preis} </Verkaufspreis>
   <PreisDerEinzelteile> {sum($teile/@Preis)} </PreisDerEinzelteile>
  </Kosten>
```

```
<Kosten>
    <Verkaufspreis Preis="350000"/>
    <PreisDerEinzelteile>59000.0</PreisDerEinzelteile>
</Kosten>
```

Rekursive Strukturen

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

<!-- Schema als DTD -->
<!DOCTYPE Bauteil[
   <!ELEMENT Bauteil (Beschreibung, Bauteil*)>
   <!ATTLIST Bauteil Preis CDATA #REQUIRED>
   <!ELEMENT Beschreibung (#PCDATA)>
]>

<!-- Wurzelelement-->
<Bauteil Preis="350000">
   <Beschreibung>Maybach 620 Limousine</Beschreibung>
   <Bauteil Preis="50000">
      <Beschreibung>V12-Biturbo Motor mit 620 PS</Beschreibung>
      <Bauteil Preis="2000">
         <Beschreibung>Nockenwelle</Beschreibung>
      </Bauteil>
    </Bauteil>
    <Bauteil Preis="7000">
      <Beschreibung>Kühlschrank für Champagner</Beschreibung>
    </Bauteil>
</Bauteil>
```

Rekursion … schwieriger

```
<!DOCTYPE VorlesungsBaum [
   <!ELEMENT VorlesungsBaum (Vorlesung *)>
   <!ELEMENT Vorlesung (Vorlesung *)>
   <!ATTLIST Vorlesung
        Titel  CDATA  #REQUIRED>
]>
```

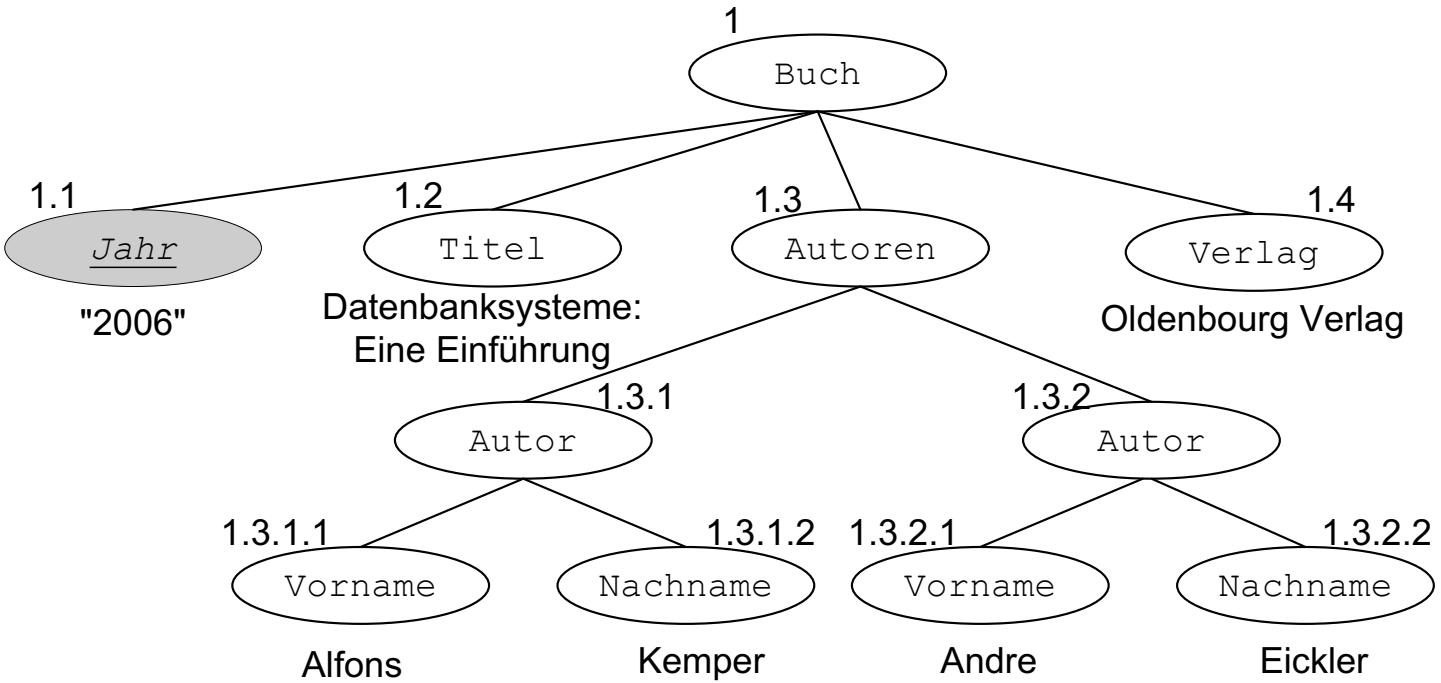
Rekursion … schwieriger

```
declare function local:eineEbene($p as element()) as element()
{
    <Vorlesung Titel="{ $p/Titel/text() }">
        {
        for $s in doc("Uni.xml")//Vorlesung
        where contains($p/@Voraussetzungen,$s/@VorlNr)
        return local:eineEbene($s)
        }
    </Vorlesung>
};

<VorlesungsBaum>
  {
    for $p in doc("Uni.xml")//Vorlesung
    return local:eineEbene($p)
  }
</VorlesungsBaum>
```

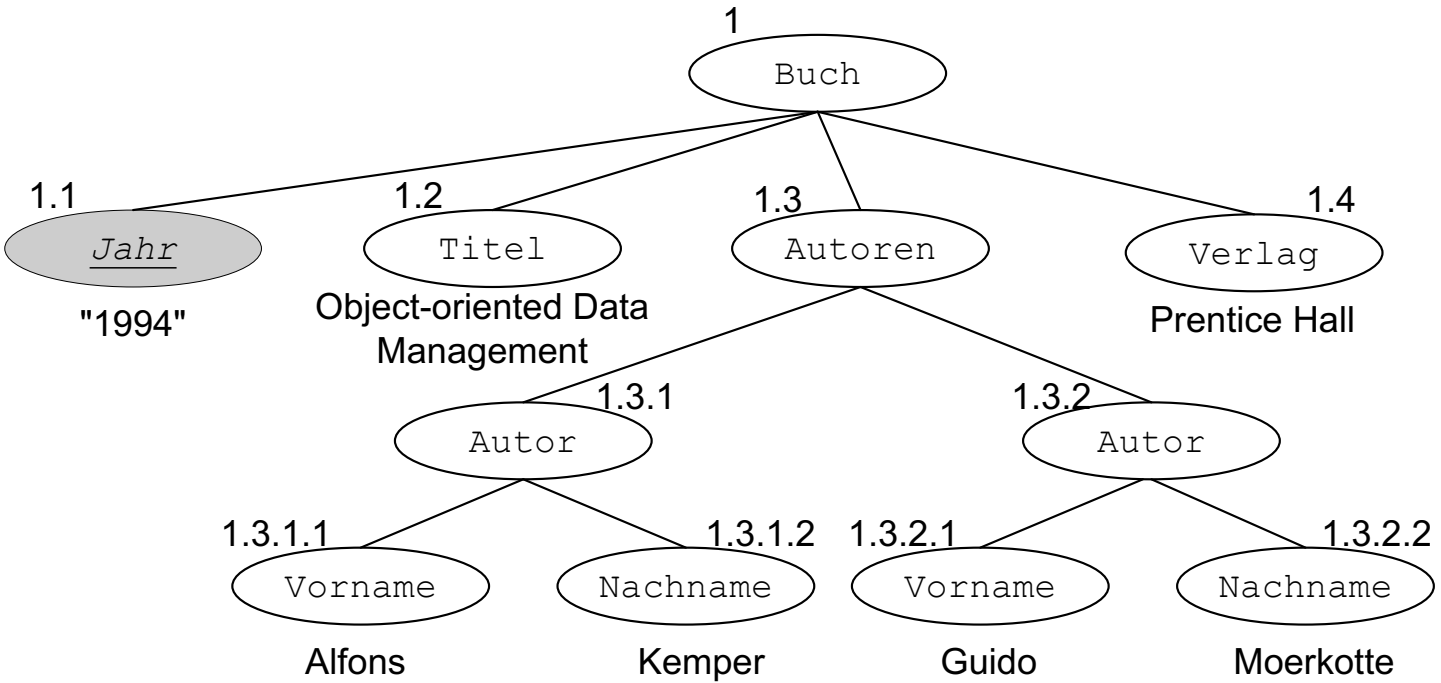
Ergebnis

```
<VorlesungsBaum>
    <Vorlesung Titel="Glaube und Wissen"/>
    <Vorlesung Titel="Ethik">
        <Vorlesung Titel="Grundzüge"/>
    </Vorlesung>
    <Vorlesung Titel="Mäeutik">
        <Vorlesung Titel="Grundzüge"/>
    </Vorlesung>
    <Vorlesung Titel="Logik"/>
    <Vorlesung Titel="Erkenntnistheorie">
        <Vorlesung Titel="Grundzüge"/>
    </Vorlesung>
    <Vorlesung Titel="Wissenschaftstheorie">
        <Vorlesung Titel="Ethik">
            <Vorlesung Titel="Grundzüge"/>
        </Vorlesung>
        <Vorlesung Titel="Erkenntnistheorie">
            <Vorlesung Titel="Grundzüge"/>
        </Vorlesung>
    </Vorlesung>
    <Vorlesung Titel="Der Wiener Kreis">
        <Vorlesung Titel="Wissenschaftstheorie">
            <Vorlesung Titel="Ethik">
                <Vorlesung Titel="Grundzüge"/>
            </Vorlesung>
            <Vorlesung Titel="Erkenntnistheorie">
                <Vorlesung Titel="Grundzüge"/>
            </Vorlesung>
        </Vorlesung>
    </Vorlesung>
    <Vorlesung Titel="Grundzüge"/>
    <Vorlesung Titel="Bioethik">
        <Vorlesung Titel="Ethik">
            <Vorlesung Titel="Grundzüge"/>
        </Vorlesung>
    </Vorlesung>
    <Vorlesung Titel="Die 3 Kritiken"/>
</VorlesungsBaum>
```

Bücher als XML-Dokument



Weiteres Buch …



Speicherung von XML Dokumenten in Relationen

Einfach als BLOB (binary large object)
- Keine Anfragemöglichkeit
- Keine Strukturierung
- Nur Archivierung

Shreddern ~ Zerlegen des XML Dokuments
- Speicherung aller Kanten des XML-Baums
- [von, nach, Tag/Marke, …]
- Einfache Relationale Darstellung
  - Eine Relation für ALLE Dokumente

Objektrelationale Speichermodelle
- Ausnutzung von Schemainformation

ge-shredder‘ed

InfoTab

DocID	ORDpfad	Tag	KnotenTyp	Wert	Pfad
4711	1	Buch	Element	–	#Buch
4711	1.1	Jahr	Attribut	2006	#Buch#@Jahr
4711	1.2	Titel	Element	Datenbank...	#Buch#Titel
4711	1.3	Autoren	Element	–	#Buch#Autoren
4711	1.3.1	Autor	Element	–	#Buch#Autoren#Autor
4711	1.3.1.1	Vorname	Element	Alfons	#Buch#Autoren#Autor#Vorname
4711	1.3.1.2	Nachname	Element	Kemper	#Buch#Autoren#Autor#Nachname
4711	1.3.2	Autor	Element	–	#Buch#Autoren#Autor
4711	1.3.2.1	Vorname	Element	Andre	#Buch#Autoren#Autor#Vorname
4711	1.3.2.2	Nachname	Element	Eickler	#Buch#Autoren#Autor#Nachname
4711	1.4	Verlag	Element	Oldenbourg	#Buch#Titel
5813	1	Buch	Element	–	#Buch
5813	1.1	Jahr	Attribut	1994	#Buch#@Jahr
5813	1.2	Titel	Element	Object...	#Buch#Titel
5813	1.3	Autoren	Element	–	#Buch#Autoren
5813	1.3.1	Autor	Element	–	#Buch#Autoren#Autor
5813	1.3.1.1	Vorname	Element	Alfons	#Buch#Autoren#Autor#Vorname
5813	1.3.1.2	Nachname	Element	Kemper	#Buch#Autoren#Autor#Nachname
5813	1.3.2	Autor	Element	–	#Buch#Autoren#Autor
5813	1.3.2.1	Vorname	Element	Guido	#Buch#Autoren#Autor#Vorname
5813	1.3.2.2	Nachname	Element	Moerkotte	#Buch#Autoren#Autor#Nachname
5813	1.4	Verlag	Element	Prentice Hall	#Buch#Titel
8769	1	Universität	Element	–	#Universität
8769	1.1	UnivName	Attribut	Virtuelle Uni...	#Universität#@UnivName
8769	1.2	UniLeitung	Element	–	#Universität#UniLeitung
8769	1.2.1	Rektor	Element	Prof. Sokrates	#Universität#UniLeitung#Rektor
8769	1.2.2	Kanzler	Element	Dr. Erhard	#Universität#UniLeitung#Kanzler
8769	1.3	Fakultäten	Element	–	#Universität#Fakultäten
...	...	...	...	...	...

Auswertung von Pfadausdrücken

/Buch/Autoren/Autor/Nachname

In der relationalen InfoTab-Darstellung kann man diese Information mit folgender SQL-Anfrage aus der Relation *InfoTab* extrahieren:

```
select n.Wert
from InfoTab n
where n.Pfad = '#Buch#Autoren#Autor#Nachname'
```

Ohne Nutzung des Pfad-Attributs –

oh je → Self-Joins „ohne Ende“

```
select n.Wert
from InfoTab b, InfoTab as, InfoTab a, InfoTab n
where b.Tag = 'Buch' and as.Tag = 'Autoren' and
      a.Tag = 'Autor' and n.Tag = Nachname and
      b.KnotenTyp = 'Element' and
      as.KnotenTyp = 'Element' and a.KnotenTyp = 'Element'
      and n.KnotenTyp = 'Element' and
      PARENT(as.ORDpfad) = b.ORDpfad and as.DOCid =b.DOCid
      and PARENT(a.ORDpfad) = as.ORDpfad and
      a.DOCid = as.DOCid and
      PARENT(n.ORDpfad) = a.ORDpfad and n.DOCid = a.DOCid
```

Pfade mit Prädikat

/Buch[Titel='Datenbanksystme']/Autoren/Autor/Nachname

Die korrespondierende SQL-Anfrage ist jetzt deutlich komplexer, da sie mehrere Self-Joins enthält.

```
select n.Wert
from InfoTab b, InfoTab t, InfoTab n
where b.Pfad = '#Buch' and
      t.Pfad = '#Buch#Titel'
      n.Pfad = '#Buch#Autoren#Autor#Nachname' and
      t.Wert = 'Datenbanksysteme' and
      PARENT(t.ORDpfad) = b.ORDpfad and
      t.DOCid = b.DOCid  and
      PREFIX(b.ORDpfad,n.ORDpfad) and b.DOCid = n.DOCid
```

Pfade mit descendant-or-self-Achse

//Autor/Nachname

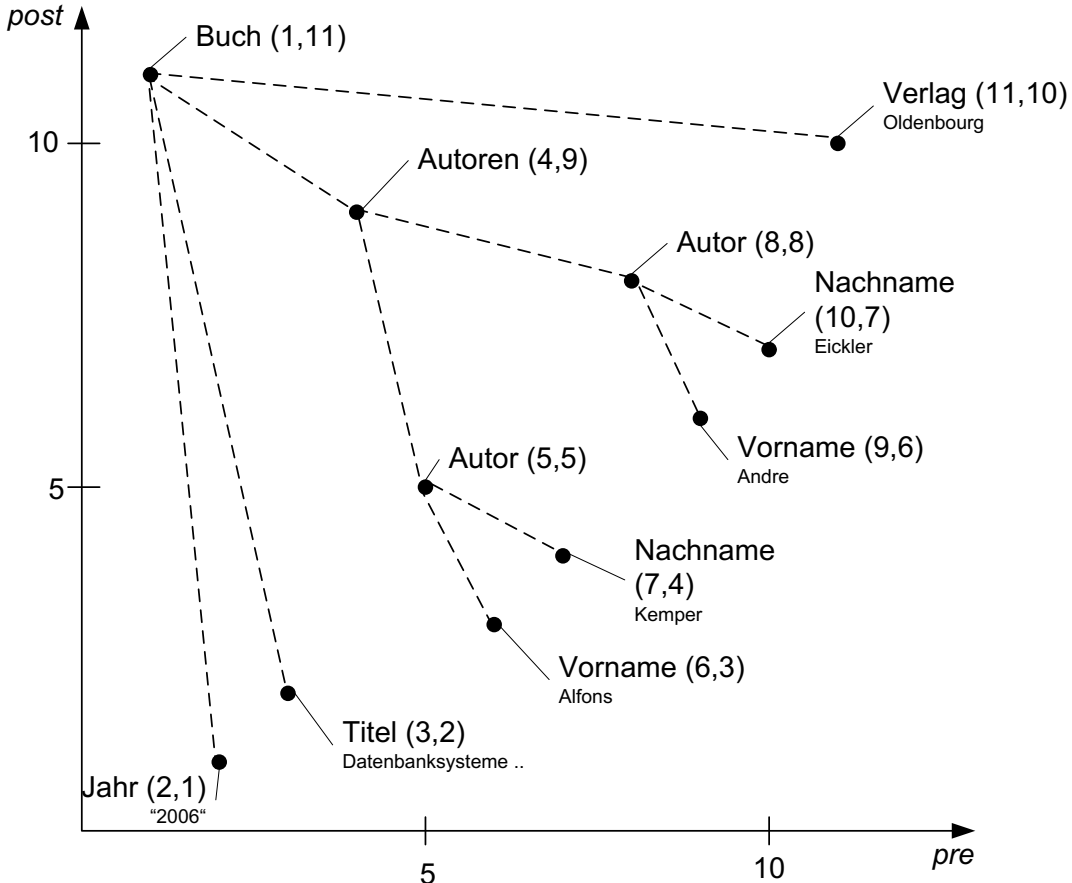
```
select n.Wert
from InfoTab n
where n.Pfad like '%#Autor#Nachname'
```

Rekursion im Prädikat

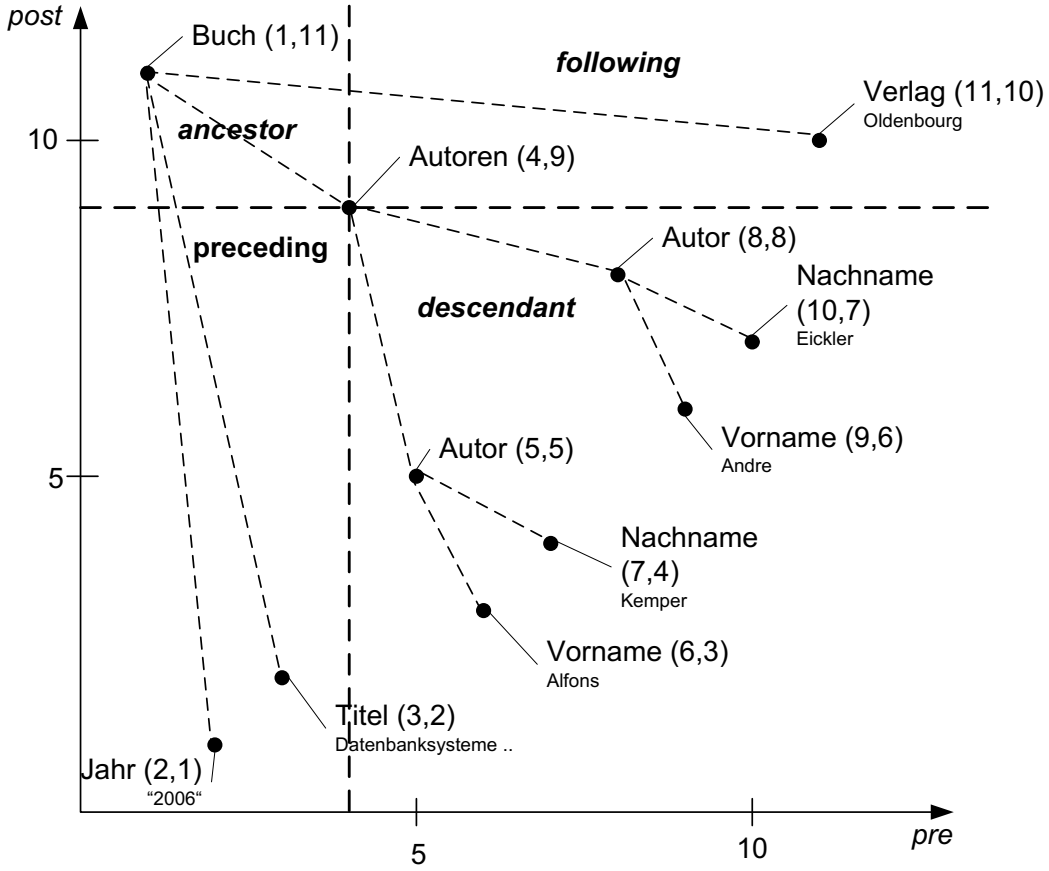
/Buch[.//Nachname = 'Kemper']/Titel

```
select t.Wert
from InfoTab b, InfoTab n, InfoTab t
where b.Pfad = '#Buch' and
      t.Pfad = '#Buch#Titel'
      n.Pfad like '%#Nachname' and
      t.Wert = 'Datenbanksysteme' and
      PARENT(t.ORDpfad) = b.ORDpfad and
      t.DOCid = b.DOCid and
      PREFIX(b.ORDpfad,n.ORDpfad) and b.DOCid = n.DOCid
```

Indexierungsschema von Grust



Indexierungsschema von Grust



Neuer Datentyp in rel DB: xml

create table Bücher (ISBN varchar(20), Beschreibung xml)

```
insert into Bücher values('3486273922',
'<Buch Jahr="2004">
        <Titel> Datenbanksysteme </Titel>
        <Autoren>
                <Autor>
                        <Vorname> Alfons </Vorname>
                        <Nachname> Kemper </Nachname>
                </Autor>
                <Autor>
                        <Vorname> Andre </Vorname>
                        <Nachname> Eickler </Nachname>
                </Autor>
        </Autoren>
        <Verlag> Oldenbourg </Verlag>
</Buch> ' )
```

Noch ein Buch speichern …

```
insert into Bücher values('0136292399',
'<Buch Jahr="1994">
  <Titel> Object-oriented Data Management </Titel>
  <Autoren>
    <Autor>
      <Vorname> Alfons </Vorname>
      <Nachname> Kemper </Nachname>
    </Autor>
    <Autor>
      <Vorname> Alfons </Vorname>
      <Nachname> Moerkotte </Nachname>
    </Autor>
  </Autoren>
  <Verlag> Prentice Hall </Verlag>
</Buch>' )
```

SQL mit XQuery-Anteilen

```
select Beschreibung.query('for $b in Buch[@Jahr=2004]
                           return $b/Autoren')
from Bücher
```

```
<Autoren>
  <Autor>
    <Vorname> Alfons </Vorname>
    <Nachname> Kemper </Nachname>
  </Autor>
  <Autor>
    <Vorname> Andre </Vorname>
    <Nachname> Eickler </Nachname>
  </Autor>
</Autoren>
```

Speicherung der Uni-Beschreibung

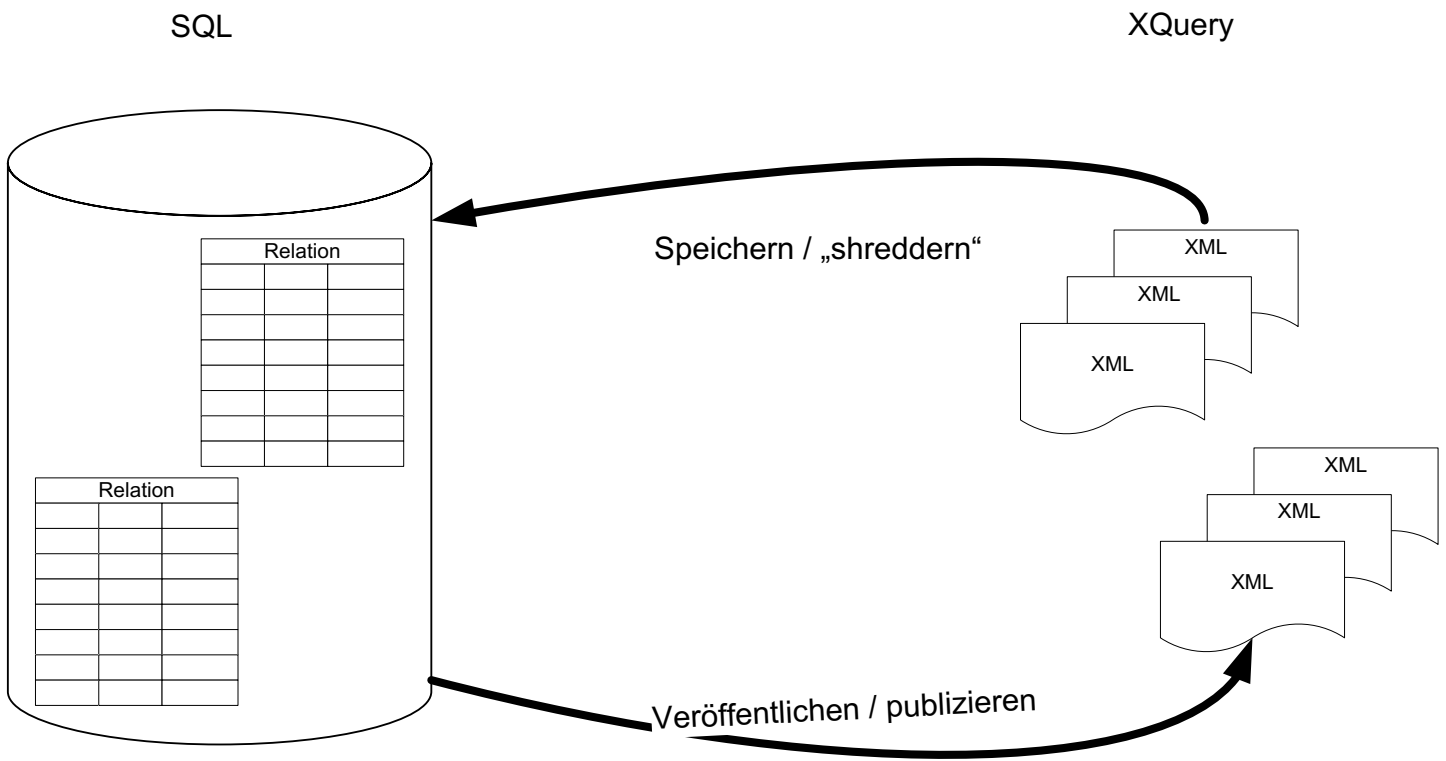
```
CREATE TABLE [dbo].[Unis](
        [Name] [varchar](30),
        [Beschreibung] [xml]
)

select Name, Beschreibung.query('for $d in //ProfessorIn
            where $d/Vorlesungen/Vorlesung
            return $d/Name') as xml
from Unis

Name            |  xml
=============================================
Virtuelle Uni  |  <Name>Augustinus</Name>
                  <Name>Sokrates</Name>
                  <Name>Russel</Name>
                  <Name>Popper</Name>
                  <Name>Kant</Name>
```

Anfrage auf der Uni-Relation

Uni: {[Name varchar, Beschreibung xml]}

```
select Name, Beschreibung.query('for $d in //ProfessorIn
                where $d/Vorlesungen/Vorlesung[SWS=4]
                return $d/Name') as xml
from Uni

==========

Virtuelle Uni  |  <Name>Sokrates</Name>
                  <Name>Kant</Name>
```

Zusammenspiel: relationale DB und XML



Publizieren: Relationen → XML

```
select *
from Professoren
for xml auto
```

```
<Professoren PersNr="2125" Name="Sokrates" Rang="C4" Raum="226" />
<Professoren PersNr="2126" Name="Russel" Rang="C4" Raum="232" />
<Professoren PersNr="2127" Name="Kopernikus" Rang="C3" Raum="310" />
<Professoren PersNr="2133" Name="Popper" Rang="C3" Raum="52" />
<Professoren PersNr="2134" Name="Augustinus" Rang="C3" Raum="309" />
<Professoren PersNr="2136" Name="Curie" Rang="C4" Raum="36" />
<Professoren PersNr="2137" Name="Kant" Rang="C4" Raum="7" />
```

Publizieren: raw Format →Tupel-weise

```
select *
from Professoren
for xml raw
```

```
<row PersNr="2125" Name="Sokrates" Rang="C4" Raum="226" />
<row PersNr="2126" Name="Russel" Rang="C4" Raum="232" />
<row PersNr="2127" Name="Kopernikus" Rang="C3" Raum="310" />
<row PersNr="2133" Name="Popper" Rang="C3" Raum="52" />
<row PersNr="2134" Name="Augustinus" Rang="C3" Raum="309" />
<row PersNr="2136" Name="Curie" Rang="C4" Raum="36" />
<row PersNr="2137" Name="Kant" Rang="C4" Raum="7" />
```

Publizieren: explizite XML-Restrukturierung

```
SELECT Name, Rang,
       ( SELECT Titel, SWS
         FROM  Vorlesungen
         WHERE gelesenVon = PersNr
         FOR XML AUTO, type)
FROM Professoren
FOR XML AUTO, type
```

```
<Professoren Name="Sokrates" Rang="C4">
  <Vorlesungen Titel="Logik" SWS="4" />
  <Vorlesungen Titel="Ethik" SWS="4" />
  <Vorlesungen Titel="Maeeutik" SWS="2" />
</Professoren>
<Professoren Name="Russel" Rang="C4">
  <Vorlesungen Titel="Erkenntnistheorie" SWS="3" />
  <Vorlesungen Titel="Wissenschaftstheorie" SWS="3" />
  <Vorlesungen Titel="Bioethik" SWS="2" />
</Professoren>  ….
```

Publizieren: Restrukturierung mit Aggregation

```
SELECT Name, Rang,
       ( SELECT sum(SWS) as Gesamt
         FROM  Vorlesungen as Lehrleistung
         WHERE gelesenVon = PersNr
         FOR XML AUTO, type)
FROM Professoren
FOR XML AUTO, type
```

```
<Professoren Name="Sokrates" Rang="C4">
  <Lehrleistung Gesamt="10" />
</Professoren>
<Professoren Name="Russel" Rang="C4">
  <Lehrleistung Gesamt="8" />
</Professoren>
<Professoren Name="Kopernikus" Rang="C3">
  <Lehrleistung />
</Professoren> …
```

Standardisierte Syntax: XMLELEMENT

```
SELECT XMLELEMENT (
        Name "Professoren",
        XMLATTRIBUTES (p.Name,p.Rang),
        XMLELEMENT (
          Name "Lehrleistung",
          (SELECT sum(v.SWS)
           FROM Vorlesungen v
           WHERE v.gelesenVon = p.PersNr )
        )
      )
FROM Professoren p
```

```
<Professoren NAME="Sokrates"
RANG="C4"><Lehrleistung>10</Lehrleistung></Professoren>
<Professoren NAME="Russel"
RANG="C4"><Lehrleistung>8</Lehrleistung></Professoren>
<Professoren NAME="Kopernikus"
RANG="C3"><Lehrleistung></Lehrleistung></Professoren>
```

Aggregation/Schachtelung

```
select xmlelement( Name "ProfessorIn",
                   xmlattributes(p.Name),
                   xmlagg( xmlelement( Name "Titel", v.Titel)))
from Professoren p, Vorlesungen v
where p.PersNr = v.gelesenVon
group by p.PersNr,p.Name;
```

```
<ProfessorIn
NAME="Sokrates"><Titel>Ethik</Titel><Titel>Maeeutik</Titel><Titel>L...
<ProfessorIn
NAME="Russel"><Titel>Erkenntnistheorie</Titel><Titel>Bioethik</Tite...
<ProfessorIn NAME="Popper"><Titel>Der Wiener Kreis</Titel></ProfessorIn>
<ProfessorIn NAME="Augustinus"><Titel>Glaube und
Wissen</Titel></ProfessorIn>
<ProfessorIn NAME="Kant"><Titel>Grundzuege</Titel><Titel>Die 3
Kritiken</Titel><...
```

XML-Elemente → Attribut-Werte

```
select isbn,
     Beschreibung.value('(/Buch/@Jahr)[1]','varchar(20)') as Jahr,
     Beschreibung.value('(/Buch/Autoren/Autor/Nachname)[1]',
                                                  'varchar(20)') as Erstautor
from Bücher
```

3486273922	2004	Kemper
0136292399	1994	Kemper

```
update Bücher
set Beschreibung.modify('insert <Vorname> Heinrich </Vorname>
                  as first into (/Buch/Autoren/Autor)[1]')
where isbn = '3486273922'

select isbn, Beschreibung from Bücher
where isbn = '3486273922'
```

```
<Buch Jahr="2004">
  <Titel> Datenbanksysteme </Titel>
  <Autoren>
    <Autor>
      <Vorname> Heinrich </Vorname>
      <Vorname> Alfons </Vorname>
      <Nachname> Kemper </Nachname>
    </Autor>
    <Autor>
      <Vorname> Andre </Vorname>
      <Nachname> Eickler </Nachname>
    </Autor>
  </Autoren>
  <Verlag> Oldenbourg </Verlag>
</Buch>
```

```
UPDATE Bücher
SET Beschreibung.modify('delete
                  /Buch/Autoren/Autor/Vorname[1]')
where isbn = '3486273922'

<Buch Jahr="2004">
  <Titel> Datenbanksysteme </Titel>
  <Autoren>
    <Autor>
      <Vorname> Alfons </Vorname>
      <Nachname> Kemper </Nachname>
    </Autor>
    <Autor>
      <Nachname> Eickler </Nachname>
    </Autor>
  </Autoren>
  <Verlag> Oldenbourg </Verlag>
</Buch>
```

Web-Services

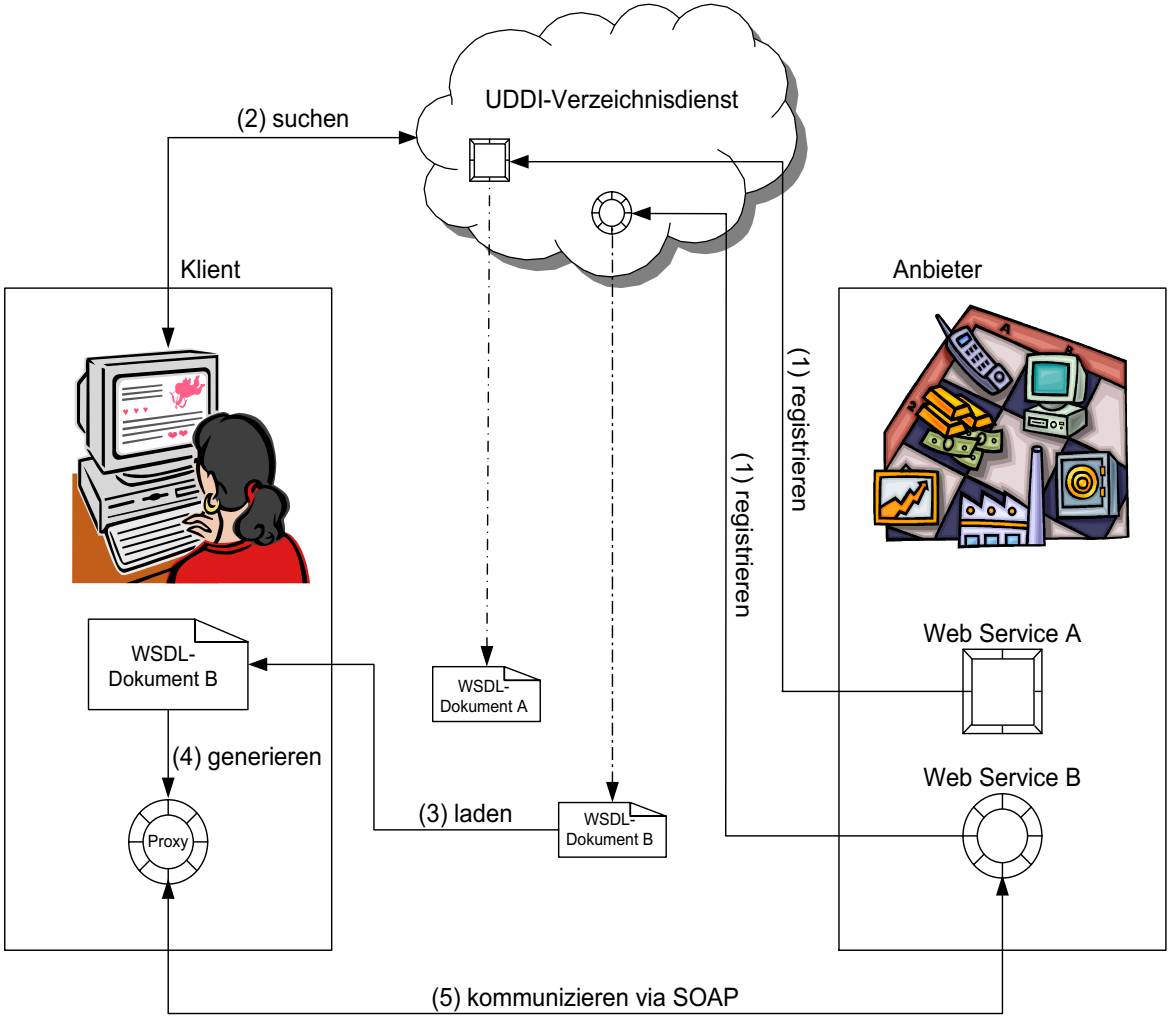
XML wird die lingua franca des Internets
Wird jetzt auch für die Kommunikation zwischen Programmen benutzt
SOAP: Simple Object Access Protocol

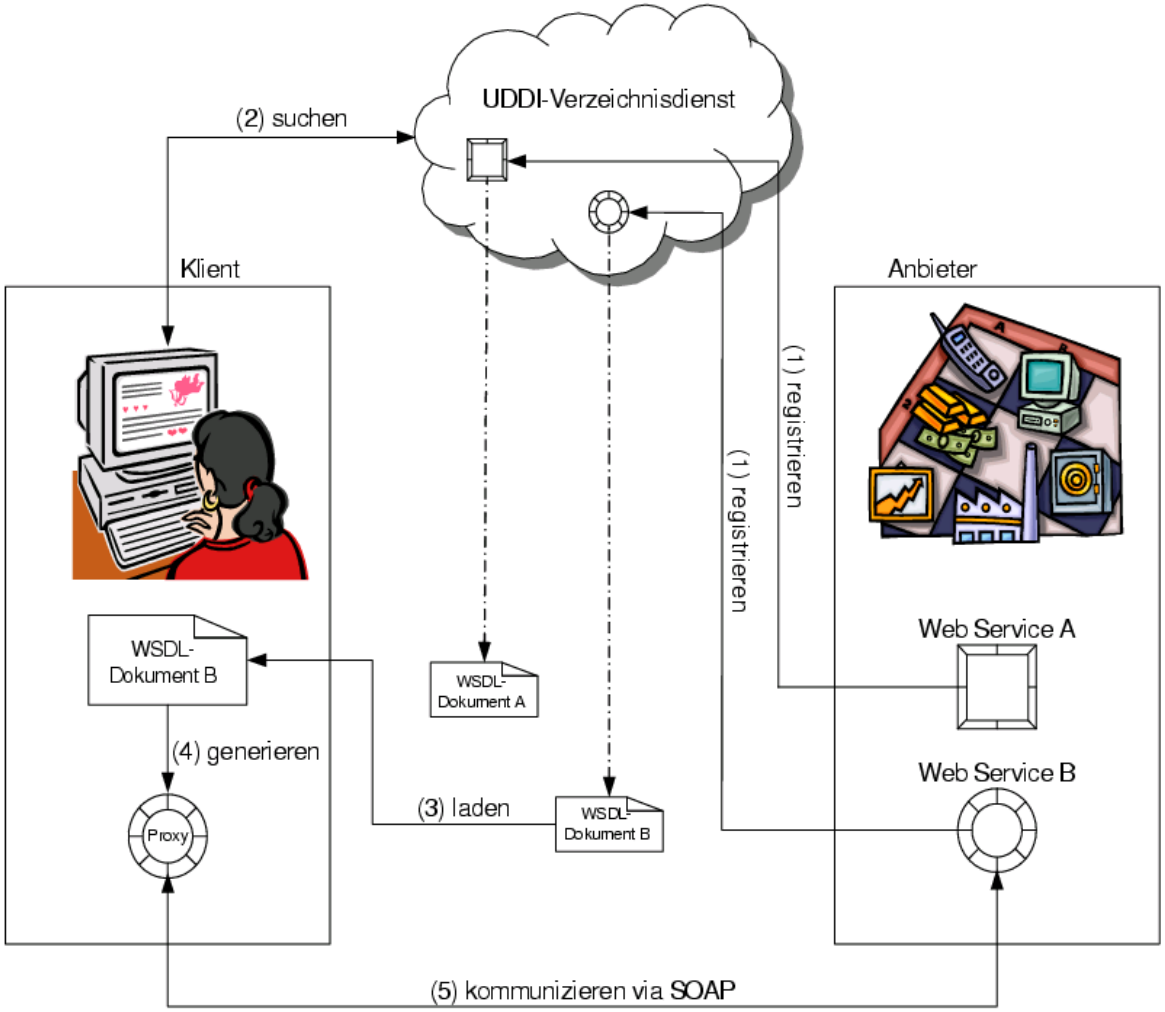
- Basiert auf XML
- Ermöglicht i.w. entfernte Prozeduraufrufe

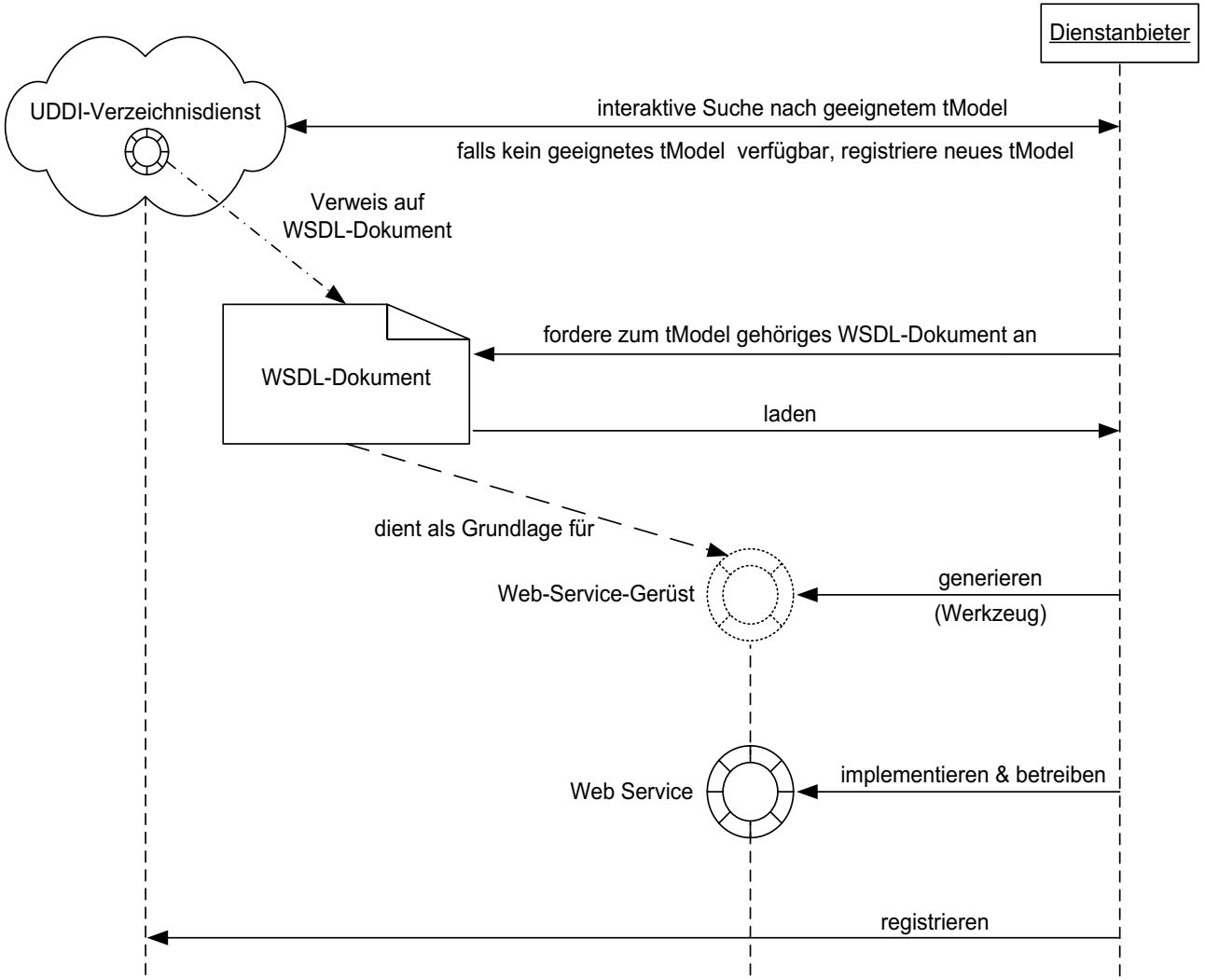
Standards: die Wichtigsten

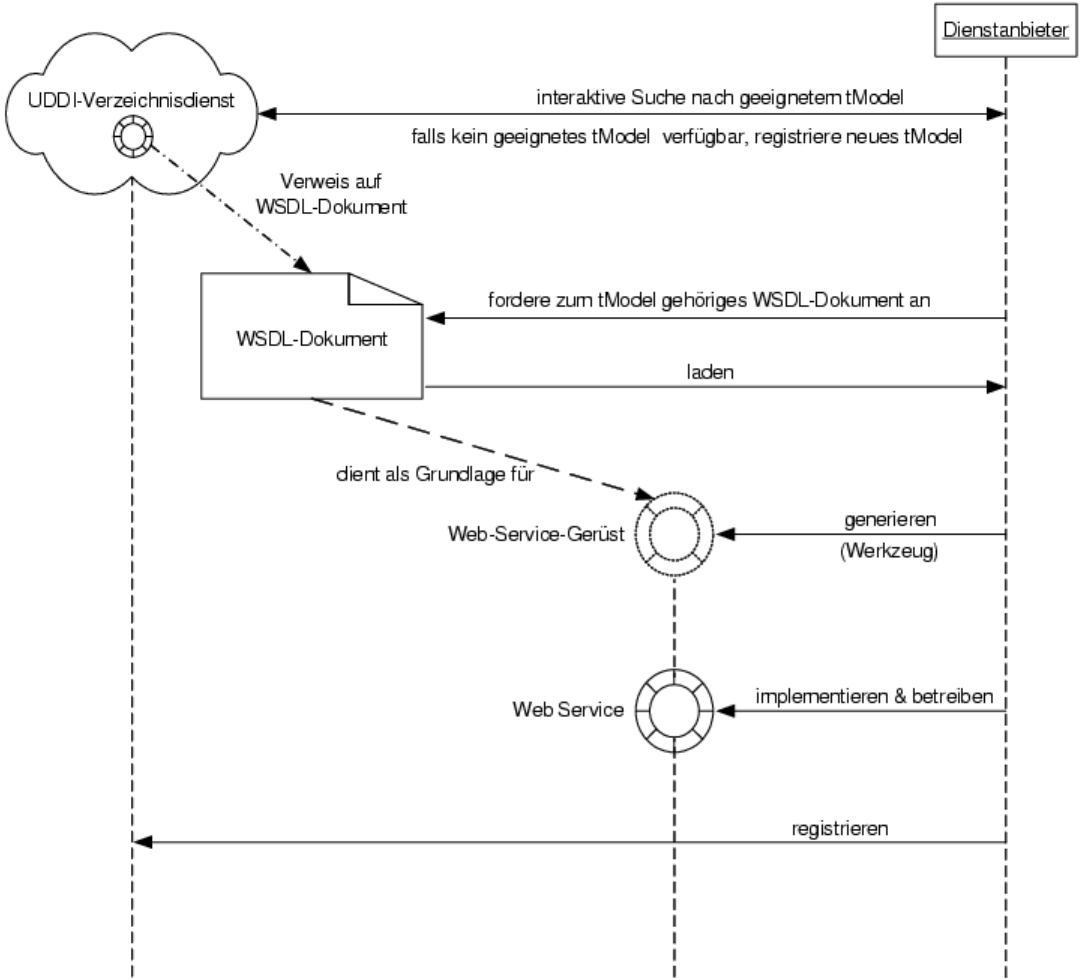
- SOAP (Simple Object Access Protocol von IBM, Microsoft, u.a.),
- UDDI (Universal Description, Discovery and Integration von HP, IBM, Intel, Microsoft, SAP, Software AG, Sun, u.a.),
- WSDL (Web-Services Description Language von Ariba, IBM und Microsoft),
- WSFL (Web-Services Flow Language von IBM),
- XLANG (Microsoft) und
- WS-Inspection (Web-Service Inspection Language von IBM und Microsoft).

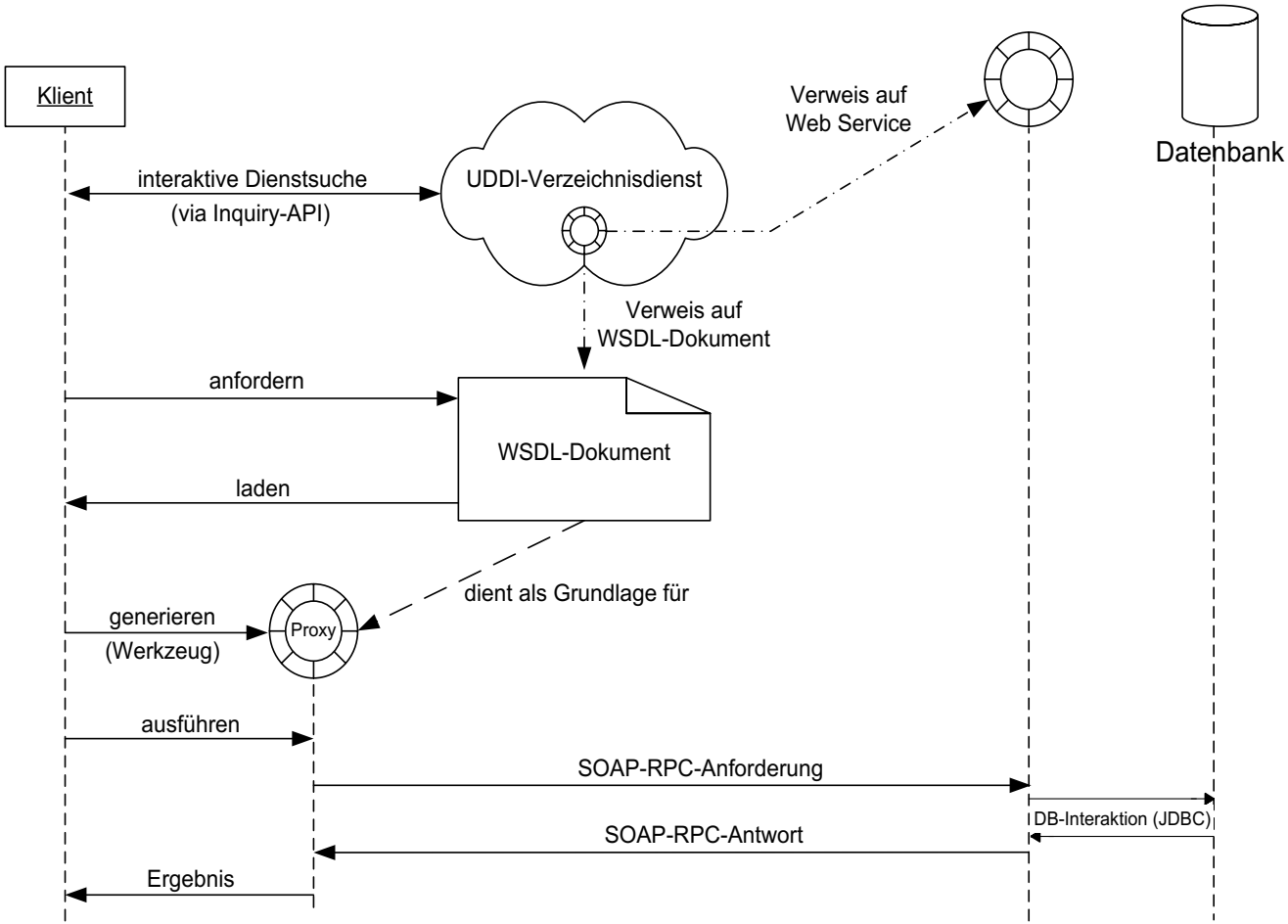
Übersicht

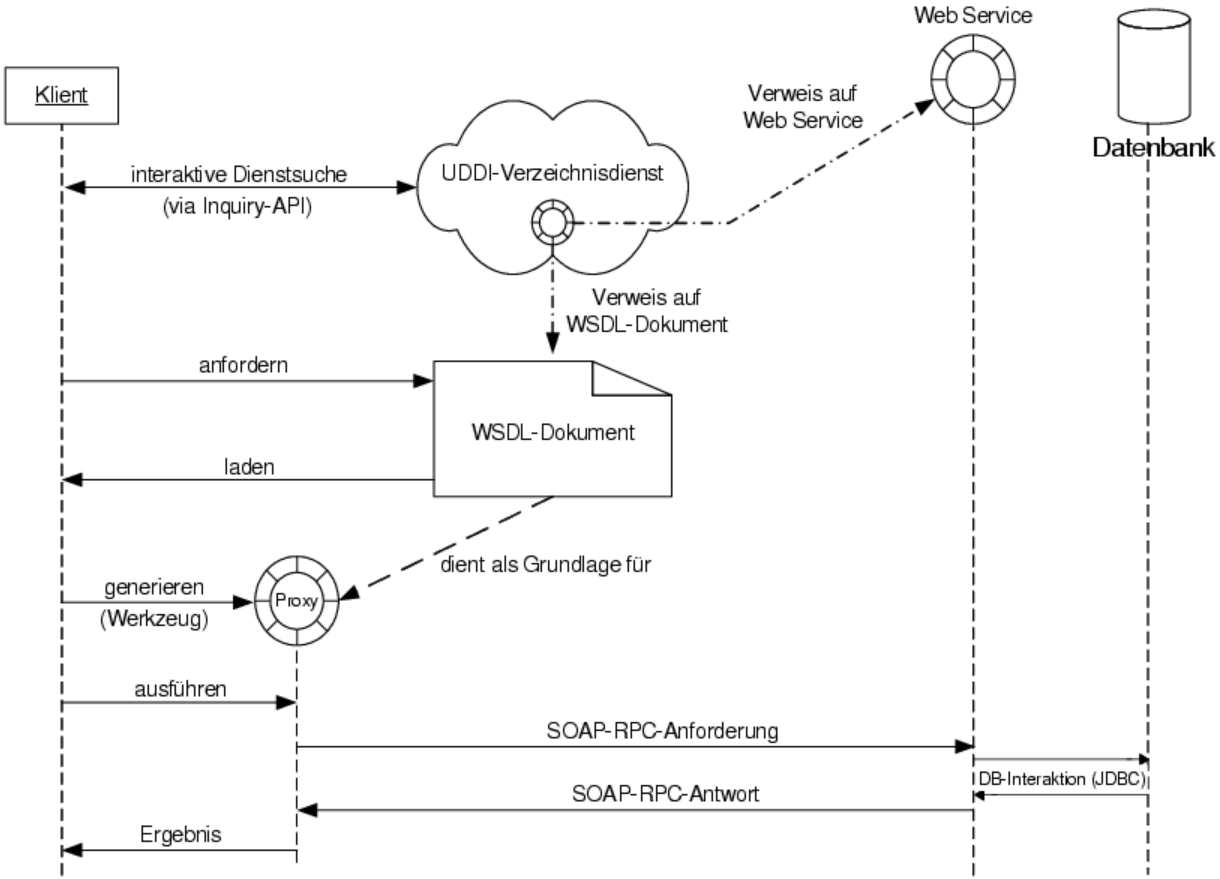

UDDI-Verzeichnisdienst
(2) suchen
Klient
Anbieter
(1) registrieren
(1) registrieren
WSDL-
Dokument B
WSDL-
Dokument A
Web Service A
(4) generieren
Web Service B
Proxy
(3) laden
WSDL-
Dokument B
(5) kommunizieren via SOAP


UDDI-Verzeichnisdienst
(2) suchen
Klient
Anbieter
(1) registrieren
(1) registrieren
WSDL-
Dokument B
WSDL-
Dokument A
Web Service A
(4) generieren
Web Service B
Proxy
(3) laden
WSDL-
Dokument B
(5) kommunizieren via SOAP


Dienstanbieter
UDDI-Verzeichnisdienst
interaktive Suche nach geeignetem tModel
falls kein geeignetes tModel verfügbar, registriere neues tModel
Verweis auf
WSDL-Dokument
WSDL-Dokument
fordere zum tModel gehöriges WSDL-Dokument an
laden
dient als Grundlage für
Web-Service-Gerüst
generieren
(Werkzeug)
Web Service
implementieren & betreiben
registrieren


Dienstanbieter
UDDI-Verzeichnisdienst
interaktive Suche nach geeignetem tModel
falls kein geeignetes tModel verfügbar, registriere neues tModel
Verweis auf
WSDL-Dokument
WSDL-Dokument
fordere zum tModel gehöriges WSDL-Dokument an
laden
dient als Grundlage für
Web-Service-Gerüst
generieren
(Werkzeug)
Web Service
implementieren & betreiben
registrieren


Klient
interaktive Dienstsuche
(via Inquiry-API)
UDDI-Verzeichnisdienst
Verweis auf
Web Service
Datenbank
Verweis auf
WSDL-Dokument
anfordern
WSDL-Dokument
laden
dient als Grundlage für
generieren
(Werkzeug)
Proxy
ausführen
SOAP-RPC-Anforderung
DB-Interaktion (JDBC)
SOAP-RPC-Antwort
Ergebnis


Web Service
Klient
Verweis auf
Web Service
Datenbank
interaktive Dienstsuche
(via Inquiry-API)
UDDI-Verzeichnisdienst
Verweis auf
WSDL-Dokument
anfordern
WSDL-Dokument
laden
dient als Grundlage für
generieren
(Werkzeug)
Proxy
ausführen
SOAP-RPC-Anforderung
DB-Interaktion (JDBC)
SOAP-RPC-Antwort
Ergebnis

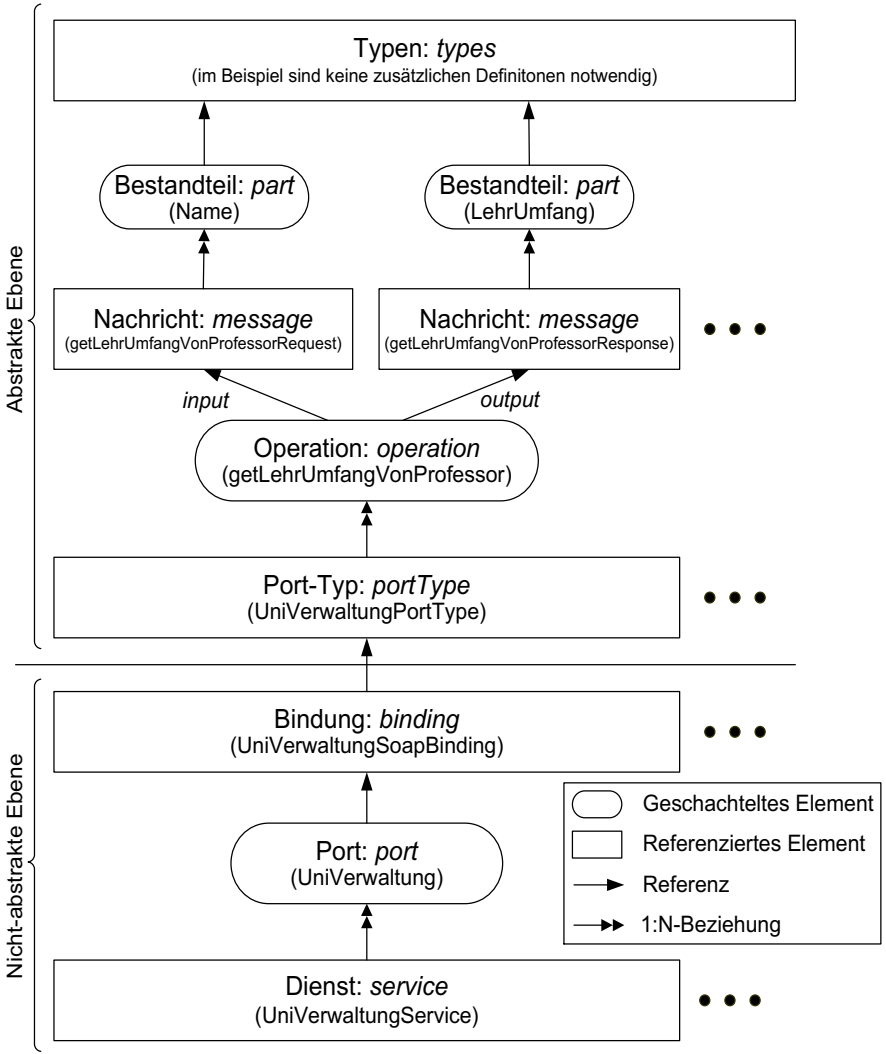
SOAP-Kommunikation

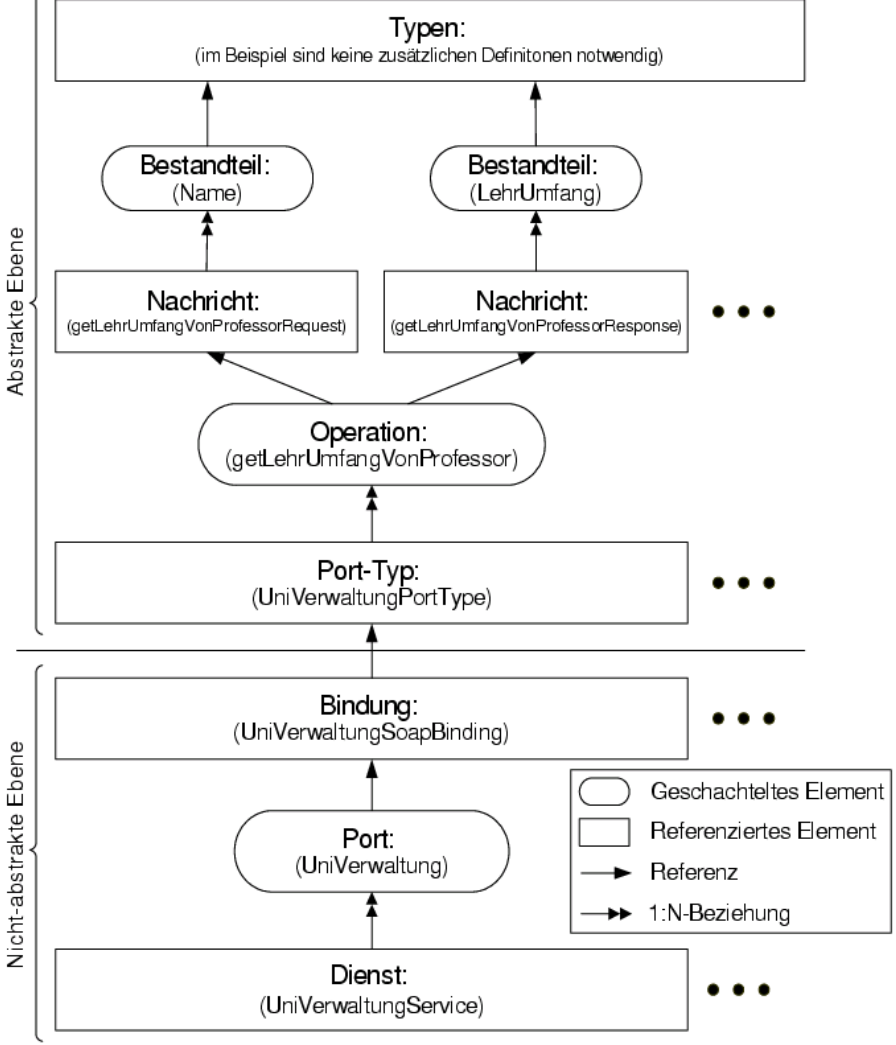
```
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    soap:encodingStyle= "http://schemas.xmlsoap.org/soap/encoding/">

    <soap:Body>
        <ns1:getLehrUmfangVonProfessor
            xmlns:ns1="http::/www.db.fmi.uni-passau.de/UniVerwaltung.wsdl">
                <ProfName xsi:type="xsd:string">Sokrates</ProfName>
        </ns1:getLehrUmfangVonProfessor>
    </soap:Body>
</soap:Envelope>

<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    soap:encodingStyle= "http://schemas.xmlsoap.org/soap/encoding/">

    <soap:Body>
        <ns1:getLehrUmfangVonProfessorResponse
            xmlns:ns1="http::/www.db.fmi.uni-passau.de/UniVerwaltung.wsdl">
                <LehrUmfang xsi:type="xsd:int">10</LehrUmfang>
        </ns1:getLehrUmfangVonProfessorResponse>
    </soap:Body>
</soap:Envelope>
```

WSDL: Web-Service Description Language

```
<?xml version="1.0" ?>
<definitions name="UniVerwaltung"
      targetNamespace="http://www.db.fmi.uni-passau.de/UniVerwaltung.wsdl"
      xmlns:tns="http://www.db.fmi.uni-passau.de/UniVerwaltung.wsdl"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
      xmlns="http://schemas.xmlsoap.org/wsdl/">

  <message name="GetLehrUmfangVonProfessorRequest">
    <part name="ProfName" type="xsd:string"/>
  </message>
  <message name="GetLehrUmfangVonProfessorResponse">
    <part name="LehrUmfang" type="xsd:int"/>
  </message>

  <portType name="UniVerwaltungPortType">
    <operation name="getLehrUmfangVonProfessor">
      <input message="tns:GetLehrUmfangVonProfessorRequest"/>
      <output message="tns:GetLehrUmfangVonProfessorResponse"/>
    </operation>
  </portType>

    …
```

```
  <binding name="UniVerwaltungSOAPBinding" type="tns:UniVerwaltungPortType">
    <soap:binding style="rpc"
                  transport="http://schemas.xmlsoap.org/soap/http"/>

    <operation name="getLehrUmfangVonProfessor">
      <soap:operation soapAction=""/>
      <input>
        <soap:body use="encoded"  namespace="UniVerwaltung"
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </input>
      <output>
        <soap:body use="encoded"  namespace="UniVerwaltung"
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </output>
    </operation>
  </binding>

  <service name="UniVerwaltungService">
    <port name="UniVerwaltung" binding="tns:UniVerwaltungSOAPBinding">
      <soap:address location=
        "http://www.db.fmi.uni-passau.de/axis/services/UniVerwaltung"/>
    </port>
  </service>

</definitions>
```


Typen: types
(im Beispiel sind keine zusätzlichen Definitonen notwendig)
Bestandteil: part
(Name)
Bestandteil: part
(LehrUmfang)
Nachricht: message
(getLehrUmfangVonProfessorRequest)
Nachricht: message
(getLehrUmfangVonProfessorResponse)
input
output
Operation: operation
(getLehrUmfangVonProfessor)
Port-Typ: portType
(UniVerwaltungPortType)
Abstrakte Ebene
Bindung: binding
(UniVerwaltungSoapBinding)
Port: port
(UniVerwaltung)
Dienst: service
(UniVerwaltungService)
Nicht-abstrakte Ebene
Geschachteltes Element
Referenziertes Element
Referenz
1:N-Beziehung


Typen:
(im Beispiel sind keine zusätzlichen Definitonen notwendig)
Bestandteil:
(Name)
Bestandteil:
(LehrUmfang)
Abstrakte Ebene
Nachricht:
(getLehrUmfangVonProfessorRequest)
Nachricht:
(getLehrUmfangVonProfessorResponse)
Operation:
(getLehrUmfangVonProfessor)
Port-Typ:
(UniVerwaltungPortType)
Bindung:
(UniVerwaltungSoapBinding)
Nicht-abstrakte Ebene
Port:
(UniVerwaltung)
Dienst:
(UniVerwaltungService)
Geschachteltes Element
Referenziertes Element
Referenz
1:N-Beziehung

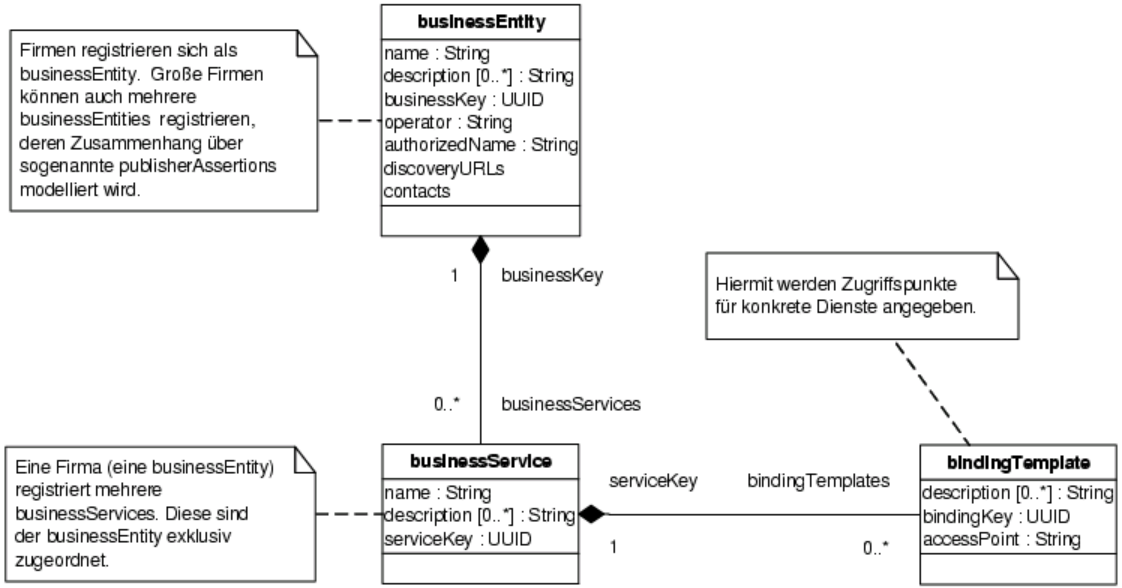
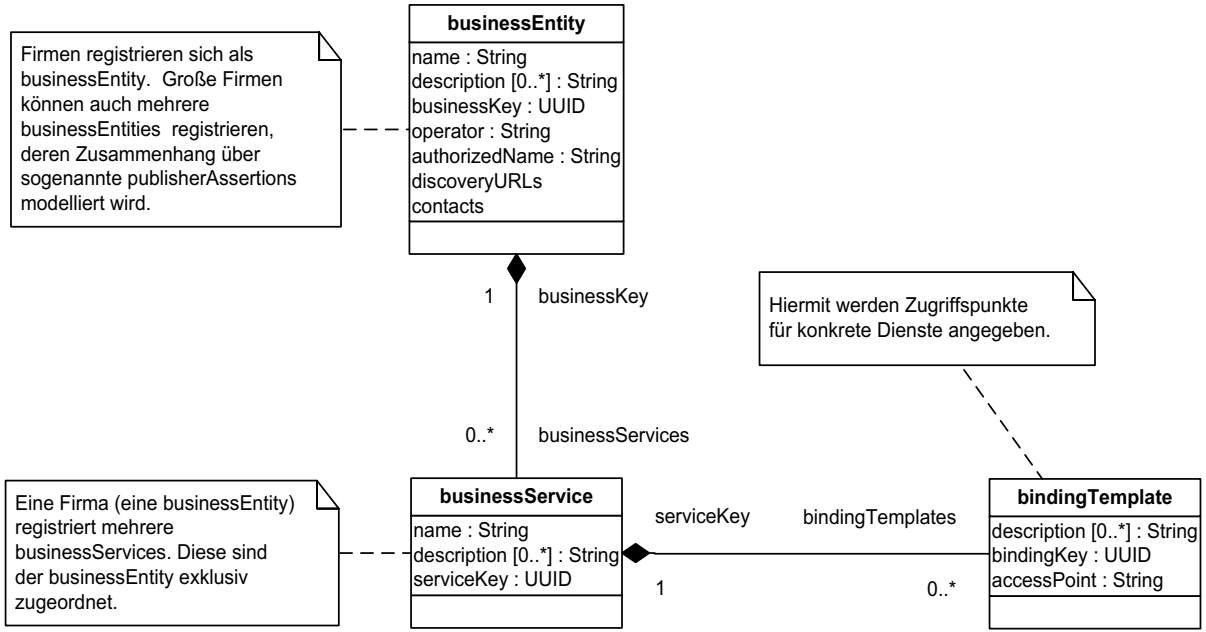


Abb. 19.13: Zusammenhang der UDDI-Komponenten

Das Ziel der UDDI-Initiative ist die Festlegung eines Standards für Verzeichnisdienste von Web-Services. Aus konzeptueller Sicht definiert UDDI ein verteiltes Datenbanksystem zur Speicherung von Dienst-Metadaten basierend auf offenen Standards und Protokollen. Wesentliche Eigenschaften eines solchen Systems


businessEntity
name : String
description [0..*] : String
businessKey : UUID
operator : String
authorizedName : String
discoveryURLs
contacts
Firmen registrieren sich als businessEntity. Große Firmen können auch mehrere businessEntities registrieren, deren Zusammenhang über sogenannte publisherAssertions modelliert wird.
1
businessKey
0..*
businessServices
Hiermit werden Zugriffspunkte für konkrete Dienste angegeben.
businessService
name : String
description [0..*] : String
serviceKey : UUID
Eine Firma (eine businessEntity) registriert mehrere businessServices. Diese sind der businessEntity exklusiv zugeordnet.
serviceKey
bindingTemplates
1
0..*
bindingTemplate
description [0..*] : String
bindingKey : UUID
accessPoint : String

Implementierung des Web-Services

```
public class UniVerwaltungSOAPBindingImpl
   implements UniVerwaltung.UniVerwaltungPortType {
   public int getLehrUmfangVonProfessor(java.lang.String profName)
      throws java.rmi.RemoteException {
         return InquireDB.getLehrUmfangVonProfessor(profName); } }

import java.sql.*;
class InquireDB {
   public static int getLehrUmfangVonProfessor(String profName) {
      int LehrUmfang = 0;
      try {         // connect to database:
         Class.forName("oracle.jdbc.driver.OracleDriver");
         Connection conn =   DriverManager.getConnection(
           "jdbc:oracle:thin:@devilray:1522:lsintern","WSUSER","Passwort");
         Statement stmt = conn.createStatement();
         ResultSet rset = stmt.executeQuery(
           "select sum(v.SWS) as LehrUmfang "
         + "from Vorlesungen v, Professoren p "
         + "where v.gelesenVon = p.PersNr and p.Name = '" + profName +"'");
         rset.next();
         LehrUmfang=java.lang.Integer.parseInt(rset.getString("LehrUmfang"));
         // disconnect
         rset.close(); stmt.close(); conn.close();
      } catch (Exception e) {}
      return LehrUmfang; } }
```

Aufruf des Web-Services (Klient)

```
package UniVerwaltung;
import java.net.URL;

public class Klient {
  public static void main(String[] args) throws Exception {
    UniVerwaltungService uvws = new UniVerwaltungServiceLocator();
    UniVerwaltungPortType uv = uvws.getUniVerwaltung(new URL
      ("http://www.db.fmi.uni-passau.de/axis/services/UniVerwaltung"));
    System.out.println("Lehrumfang von Professor/in " +
      "Sokrates"  +": " +
      uv.getLehrUmfangVonProfessor("Sokrates")); //Dienstinvokation
   }
}
```

Handgestrickter Klient

```
import java.io.*; import java.net.*;

public class ClientUniVerwaltung {
   private static final int BUFF_SIZE = 100;

   public static void main(String[] argv) throws Exception {
      String request =
      "<?xml version='1.0' encoding='UTF-8'?>"+
        "<soap:Envelope " +
         "xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/' " +
         "xmlns:xsd='http://www.w3.org/2001/XMLSchema' " +
         "xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance' " +
         "soap:encodingStyle= " +
               "'http://schemas.xmlsoap.org/soap/encoding/'> " +
       "<soap:Body> "+
         "<ns1:getLehrUmfangVonProfessor " +
            "xmlns:ns1='http::/www.db.fmi.uni-passau.de/" +
                 "UniVerwaltung.wsdl'> " +
           "<ProfName xsi:type='xsd:string'>Sokrates</ProfName>" +
           "</ns1:getLehrUmfangVonProfessor>" +
        "</soap:Body>"+
       "</soap:Envelope>";
```

Handgestrickter Klient … cont‘d

```
    URL url = new URL(
        "http://www.db.fmi.uni-passau.de/axis/services/UniVerwaltung");
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();

    conn.setDoOutput(true); conn.setUseCaches(false);
    conn.setRequestProperty("Accept", "text/xml");
    conn.setRequestProperty("Connection", "keep-alive");
    conn.setRequestProperty("Content-Type", "text/xml");
    conn.setRequestProperty(
      "Content-length",
      Integer.toString(request.length()));
    conn.setRequestProperty("SOAPAction", "\" \"");

    OutputStream out = conn.getOutputStream();
    out.write(request.getBytes()); out.flush();

    StringBuffer response = new StringBuffer(BUFF_SIZE);
    InputStreamReader in =
      new InputStreamReader(conn.getInputStream(), "UTF-8");
    char buff[] = new char[BUFF_SIZE];  int n;
    while ((n = in.read(buff, 0, BUFF_SIZE - 1)) > 0) {
      response.append(buff, 0, n);
    }
    out.close(); in.close();
    System.out.println( response.toString() );
  }
}
```