Datenmodelle

1. Relationen
2. Objekte
3. XML
4. RDF

(5. Später: Logik)

Datenmodelle: Relationen

UNIVERSITÄT KOBLENZ · LANDAU

Datenmodell: Relationen

<is web>

- Mathematische Relationen
- Schema:
  Professoren {[PersNr: Int, Name: Varchar(30), Rang: Char(2), Raum: Char(4)]}
- Ausprägung, vgl. Beispieldaten

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

Studenten

MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
26830	Aristoxenos	8
27550	Schopenhau	6
28106	Carnap	3
29120	Theophrasto	2
29555	Feuerbach	2

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	Wissenschaftstheor	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

voraussetzen

Vorgänger	Nachfolger
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

hören

MatrNr	VorlNr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022
25403	5022

Assistenten

PersInr	Name	Fachgebiet	Boss
3002	Platon	Ideenlehre	2125
3003	Aristoteles	Syllogistik	2125
3004	Wittgenstein	Sprachtheorie	2126
3005	Rhetikus	Planetenbewegung	2127
3006	Newton	Keplersche Gesetze	2127
3007	Spinoza	Gott und Natur	2126

prüfen

MatrNr	VorlNr	PersNr	Note
28106	5001	2126	1
25403	5041	2125	2
27550	4630	2137	2

Anfragen auf Relationen

- Formale Grundlagen
  - Relationale Algebra
  - Beispiel:

    $\sigma_{(name=„Curie“)}$ (Professoren ⋈ Vorlesung)

  - Relationenkalkül
  - Beispiel:

    {t | t2 Vorlesung ^
    9 p2 Personen
    (t.gelesenVon=p.PersNr ^
    p.Name=„Curie“)}

- Anfragesprache SQL:

  SELECT titel
  FROM Vorlesung, Professoren
  WHERE name=`Curie‘ and
  Vorlesungen.gelesenVon=Professoren.PersNr

Anwendung

- Relationale Datenbanken
- De-facto Industriestandard für Datenmodell und Anfragen seit 15-20 Jahren
- Flexibel, mächtig und
  - vergleichsweise – abgeschottet von der Welt!
  - Dumps möglich, aber nicht interoperabel!

Information Systems & Semantic Web
University of Koblenz ▪ Landau, Germany

Datenmodelle: Objekte

UNIVERSITÄT KOBLENZ · LANDAU

Datenmodell: Objekte

- Abstraktion
- Vererbung
- Kapselung von
  - Daten
  - Methoden

Deklaration

```
class Professor extends Angestellter{
public int PersNr;
public string Name;
...
public static int AnzahlProf();}

class Vorlesung {
public int VorlNr;
public string Titel;
public Professor gelesenVon;
...}
```

Daten

```
Professor o1 = new Professor(2137, `Aristoteles‘)
Vorlesung v1 = new Vorlesung(5001, `Grundzüge‘,
o1)
```

Anfragen auf Objekten

<is web>

- Pfadnavigation
  v1->o1.name
- Selektion
  . . .
  where v1.titel=„Grundzüge“
  . . .
- Joins

Kein so unmittelbares, formales Modell für die Datenaspekte!

Anwendung

<is web>

- Programmierung: Java, C#, C++, Smalltalk, Python,…
- Objektrelationale Datenbanken:
  jede populäre kommerzielle relationale Datenbank hat objektorientierte Erweiterungen:
  IBM DB1, Oracle, MS SQL Server,…
- Aber:
  kein echter Austausch über die Grenzen der Datenbank hinaus

<is web>

Information Systems & Semantic Web
University of Koblenz ▪ Landau, Germany

Datenmodelle: Dokumentstrukturen

UNIVERSITÄT KOBLENZ · LANDAU

Dokumentmodell: XML

<is web>

- eXtensible Markup Language
- Abstammung: strukturierter Text (HTML4.0 ∈ XML ⊂ SGML)
- Web-Standard (W3C) für Datenaustausch:
  - Eingabe und Ausgabedaten vieler Anwendungen werden über XML beschrieben
  - Für eine Anwendung einigt sich die Industrie auf Standardbeschreibungen (vgl. http://Oasis.org)
- „Komplementär“ zu HTML:
  - HTML ist eine Anwendung von XML
  - HTML beschreibt Präsentation von Inhalten
  - XML beschreibt Struktur von Inhalten
- Datenmodellierung: XML als Datenmodell für semi-strukturierte Daten

XML vs HTML

- HTML: Fixierte Tags und Semantik (Präsentation von Text)
- XML: Variables Tag Set um anwendungsspezifische Syntax zu beschreiben (Meta Grammatik)
- XML⊂SGML

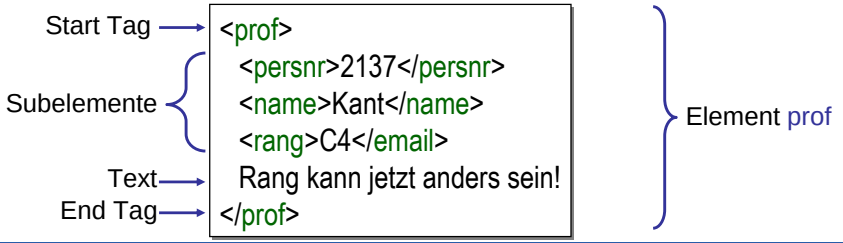
```
<h1> Veranstaltungsverzeichnis </h1>
  <p>
    <i> Grundzüge </i>
     Kant
    < br > Donnerstag, 16.00
  <p>
    ...
```

HTML

```
<Veranstaltung id="o1">
  <Vorlesung vorlnr=„5001“>
    <titel> Grundzüge<titel>
    <prof>
        <persnr>2137</persnr>
        <name>Kant</name>
        <rang>C4</email>…
    </prof>
  </Vorlesung>
  ...
</Veranstaltung>
```


XML Syntax (1) – XML Element

- XML element
  - Beschreibung von Objekten durch korrespondierende Tags wie <prof> und </prof>
  - Inhalt eines Elements: Text und/oder weitere Elemente (Subelemente)
  - Elemente können beliebig tief geschachtelt sein
  - Leere Elemente: <year></year> abgekürzt: <year/>



XML Syntax (2) – XML Attribute

- XML Attribute:
  - Name-value-Paare
  - An Element gebunden
  - Alternative zur Datenbeschreibung

Attribut rang

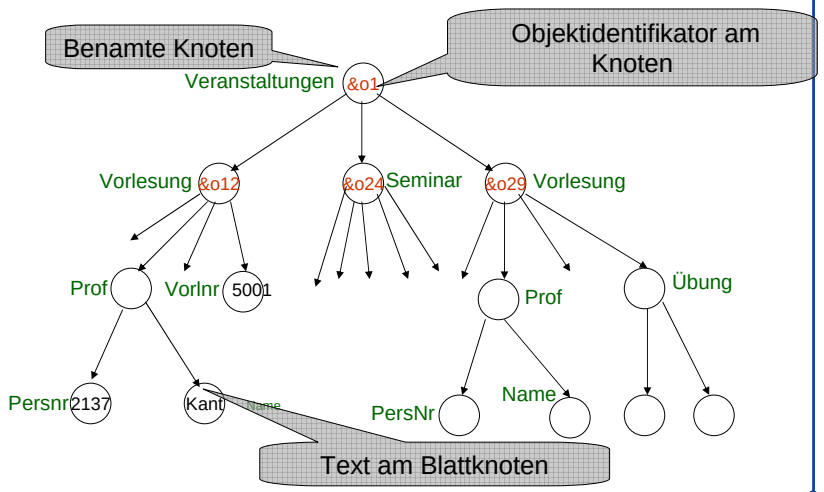
```
<prof rang="C4">
  <persnr>2137</persnr>
  <name>Kant</lastname>
</prof>
```

Andere syntaktische Version für die gleichen Daten:

```
<prof persnr="2137" name="Kant" rang="C4"/>
```

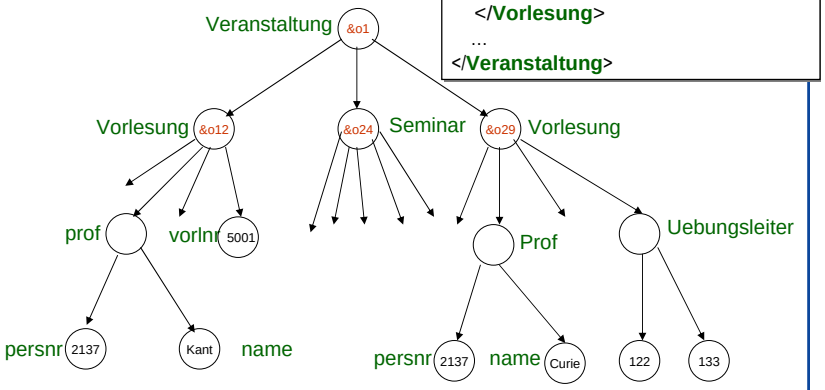

XML Modell

Abbildung von Objekten als gerichtete Graphen



XML Modell

```
<Veranstaltung id="o1">
  <Vorlesung>
    <titel>Grundzüge</titel>
    <vorlnr>5001</vorlnr>
    <prof>
        <persnr>2137</persnr>
        <name>Kant</name>
        <rang>C4</email>…
    </prof>
  </Vorlesung>
  …
</Veranstaltung>
```



Schemata

- **XML Dokument**:
  - Text Dokument mit XML Beschreibungen
  - Datenbanksicht: Dokument ist eine Datenbasis
- **Well-formed XML document** (wohlgeformte XML Dokumente)
  - Alle Elements sind korrekt geschachtelt mit passenden Start und End Tags
  - Dokument hat ein Wurzelelement
  - Wohlgeformte XML-Dokumente können immer noch unstrukturierten Text enthalten
  - XML Sonderzeichen müssen gesondert dargestellt werden!
- **Gültiges (valid) XML Dokument**:
  - Wohlgeformtes XML Dokument, das einem verbundenen Schema entspricht
  - Schema wird benutzt, um das Dokument zu validieren
  - Vernünftig für Datenaustausch oder Web Portal

Anfragen auf XML: XQuery

- **XQuery**
- **FLOWR Ausdrücke**

FOR	- Laufvariable
LET	- Zwischenergebnis
ORDER	- Sortieren
WHERE	- Pattern Matching
RETURN	- Resultatskonstruktion

Beispielanfrage

```
<ISWebLecture>
  where
   <Veranstaltung>
       <Vorlesung>
              <Prof><PersNr>$P</PersNr><name>$N</name></Prof>
              <VorlNr>$V</VorlNr>
              <titel>$T</titel>
       </Vorlesung>
   </Veranstaltung>  in "www.uni-koblenz.de/veranstaltungsverzeichnis",
   <memberNr>$P</memberNr> in "isweb.uni-koblenz.de/team"
  construct
   <Lecture>
        <title>$T</title>
        <name>$N</name>
   </Lecture>
</ISWebLecture>
```

Falls Kant ISWeb Mitglied ist:

```
<ISWebLecture><Lecture>
        <title>Grundzüge</title>
        <name>Kant</name>
</Lecture></ISWebLecture>
```

Anwendungen

- Alle großen kommerziellen Datenbanken unterstützen XML: Oracle, DB1, MS SQL Server,…
- Native XML Datenbanken haben sich nicht durchgesetzt: Tamino, InfoNyte, etc.
- Datenaustausch mit Hilfe von XML ist Standard
  → Web Services basieren vollständig auf XML!

Semi-strukturierte Daten

Relationenmodelle oder Objektmodelle

Vorteile:
- Klare Konsistenzeigenschaften
- Teilweise: einfache, saubere, formale Modelle

Nachteile:
- Nur vollständige Daten
- Nicht austauschbar
- Nicht gut lesbar

XML

Vorteile:
- Vergleichsweise gut lesbar
- Unvollständige Daten kein Problem
- Leicht serialisierbar,
- Gut austauschbar

Nachteile:
- Kein einfaches, schönes Modell
- Dokumentenzentriert, nicht daten- oder objektzentriert ⇒Viele Möglichkeiten Objekte zu serialisieren

Offene Fragen bei XML

- Wie verhält sich „Prof“ zu „Angestellter“
- Bedeutet „Prof“ unter „Vorlesung“ das gleiche wie „Prof“ unter „Seminar“ und wie „Prof“ unter der Liste der Professoren?
- Wie verweise ich auf Kant im Web?
- Wie verlinke ich Daten über Dokumentgrenzen hinweg?
- Bedeutet die Reihenfolge etwas?

⇒ XML ist ein Dokumentenformat geblieben

Information Systems & Semantic Web
University of Koblenz ▪ Landau, Germany

Semantic Web - RDF

UNIVERSITÄT KOBLENZ · LANDAU

RDF Datenmodell

- **Resources**
  - Eine Resource ist ein referenzierbares Ding (Klasse, Objekt, …)
  - Resourcen haben
    - URIs – Uniform Resource Identifiers oder
    - IRIs - Internationalized Resource Identifiers

Benamung und Addressierung

- URIs und IRIs
  - http://www.w3.org/Addressing/
  - http://www.ietf.org/rfc/rfc3987.txt
- Namen mit
  - Pfadnormalisierung: http://a.b.c/d/../d = http://a.b.c/d
  - Umlauten: http://häuser.und.bäume.de
  - Rechts- nach Linksschreibung: http://a.b.c/inuredna Leserichtung <-

URIs: Best Practices

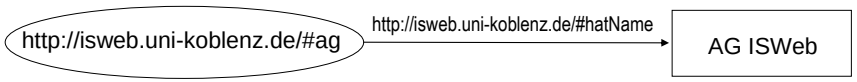
- Verwende URIs, die bereits bekannt sind
  - Suchmaschinen: Swoogle, Okkam
- Stelle für jede URI ein Dokument bereit, das die URI erklärt
  - Gut: http://isweb.uni-koblenz.de/#ag für ISWeb
  - Schlecht: http://ThisSiteDoesNotExist/#ag für ISWeb
- Benutze etablierte Konventionen zur Bildung von URIs:
  - Für Telefonnummern, ISSN, etc.
- Verwende keine URL als URI fuer eine Person oder Organisation, z.B. http://isweb.uni-koblenz.de ist eine Webseite und nicht die Arbeitsgruppe ISWeb
  - Schlecht: http://isweb.uni-koblenz.de für die Arbeitsgruppe ISWeb
  - Besser: http://isweb.uni-koblenz.de/#ag für die Arbeitsgruppe
- Leite neue URIs aus Webseite ab, über die man die Kontrolle hat:
  - Gut: http://isweb.uni-koblenz.de/#neu für mich
  - Schlecht: http://isweb.uni-koblenz.de/#neu für Sie

RDF Datenmodell

- **Resource**
  - Eine Resource ist ein referenzierbares Ding (Klasse, Objekt, …)
  - Resourcen haben
    - URIs – Uniform Resource Identifiers oder
    - IRIs - Internationalized Resource Identifiers
- **Property**
  - In etwa (nicht ganz!) vergleichbar mit UML Assoziationen
  - Verbindungen von Resourcen zu anderen Resourcen oder Literalen
  - URI
- **Literal**
  - Einfacher atomarer Datentyp (fast wie String; später mehr)
- **Statements**
  - "Resource has Property with Value"
  - Rollen: Subjekt – Prädikat – Objekt
  - Values können Resourcen oder Literale sein

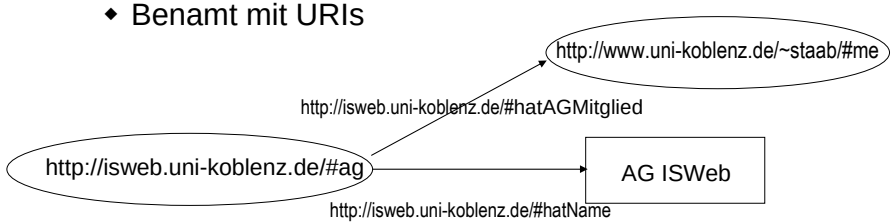
Einfaches Beispiel

- **Statement**
  - "Die Resource http://isweb.uni-koblenz.de/#ag hat den Namen AG ISWeb"
- **Struktur**
  - Resource (subject) http://isweb.uni-koblenz.de/#ag
  - Property (predicate) http://isweb.uni-koblenz.de/#hatName
  - Value (object) "AG ISWeb"
- **Gerichteter Graph**



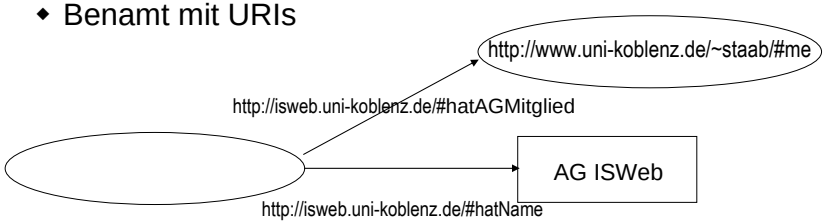
Graphen

- **Knoten:**
  - **Resourcen, benamt mit URIs**
  - Unbenamte Resourcen (Blank Nodes)
  - Literale, benamt mit Strings
- **Gerichtete Kanten:**
  - Benamt mit URIs



Graphen: auch zulässig

- **Knoten:**
  - Resourcen, benamt mit URIs
  - Unbenamte Resourcen (**Blank Nodes**)
  - Literale, benamt mit Strings
- **Gerichtete Kanten:**
  - Benamt mit URIs



Schreibweisen in Turtle

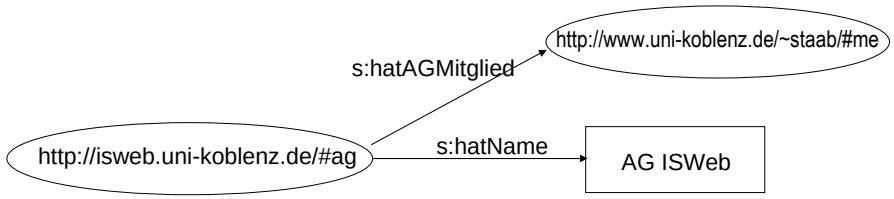
Turtle

```
<http://isweb.uni-koblenz.de/#ag>
  <http://isweb.uni-koblenz.de/preds/hatAGMitglied>
    <http://www.uni-koblenz.de/~staab/#me>
```

Turtle mit Namespaces

```
@prefix s  <http://isweb.uni-koblenz.de/preds>

<http://isweb.uni-koblenz.de/#ag>
  s:hatAGMitglied <http://www.uni-koblenz.de/~staab/#me>
```



Schreibweisen in Turtle

<is web>

```
@prefix s  <http://isweb.uni-koblenz.de/preds/>

<http://isweb.uni-koblenz.de/#ag> s:hatAGMitglied <http://www.uni-koblenz.de/~staab/#me> .
<http://isweb.uni-koblenz.de/#ag> s:hatAGMitglied <http://www.uni-koblenz.de/~sizov/#me> .
<http://isweb.uni-koblenz.de/#ag> s:hatAGMitglied <http://www.uni-koblenz.de/~saathoff/#me> .
<http://isweb.uni-koblenz.de/#ag> s:hatName “AG ISWeb”
```

Kürzer

```
@prefix s  <http://isweb.uni-koblenz.de/preds>
@prefix u <http://www.uni-koblenz.de/>
<http://isweb.uni-koblenz.de/#ag>     s:hatAGMitglied u:~staab/#me;
                                      s:hatAGMitglied u:~sizov/#me;
                                      s:hatAGMitglied u:saathoff/#me;
                                      s:hatName “AG ISWeb”.
```

Noch Kürzer

```
@prefix s  <http://isweb.uni-koblenz.de/preds>
@Prefix u <http://www.uni-koblenz.de/#ag>
<http://isweb.uni-koblenz.de/>
                s:hatAGMitglied u:~staab/#me, u:~sizov/#me, u:saathoff/#me;
                s:hatName “ISWeb”.
```