

## Vorlesung Netzbasierte Informationssysteme

### Semantic Web I

**Prof. Dr. Adrian Paschke**

Arbeitsgruppe Corporate Semantic Web (AG-CSW)

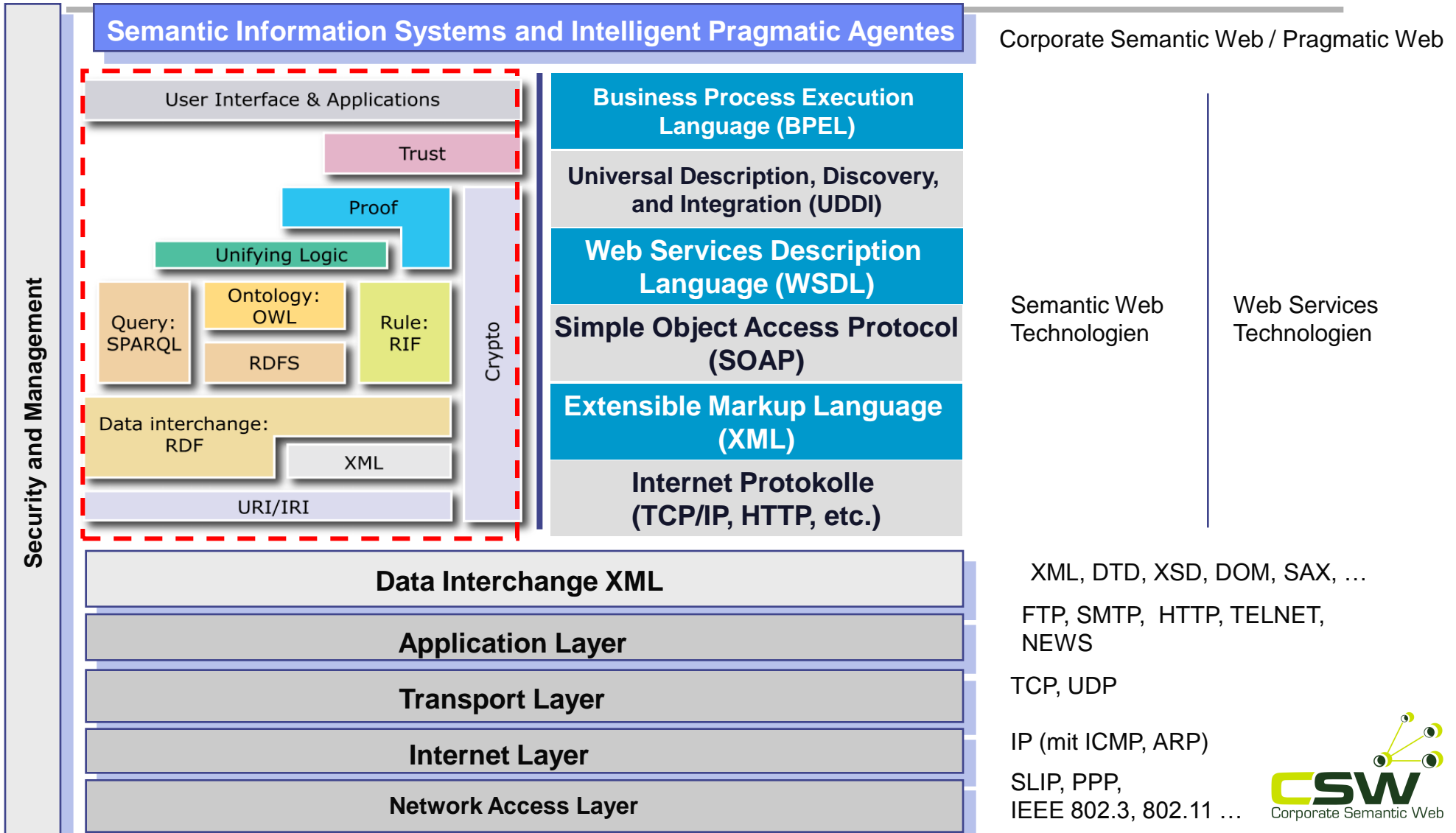
Institut für Informatik, Freie Universität Berlin

[paschke@inf.fu-berlin.de](mailto:paschke@inf.fu-berlin.de)

<http://www.inf.fu-berlin.de/groups/ag-csw/>



# Überblick



# Semantic Web

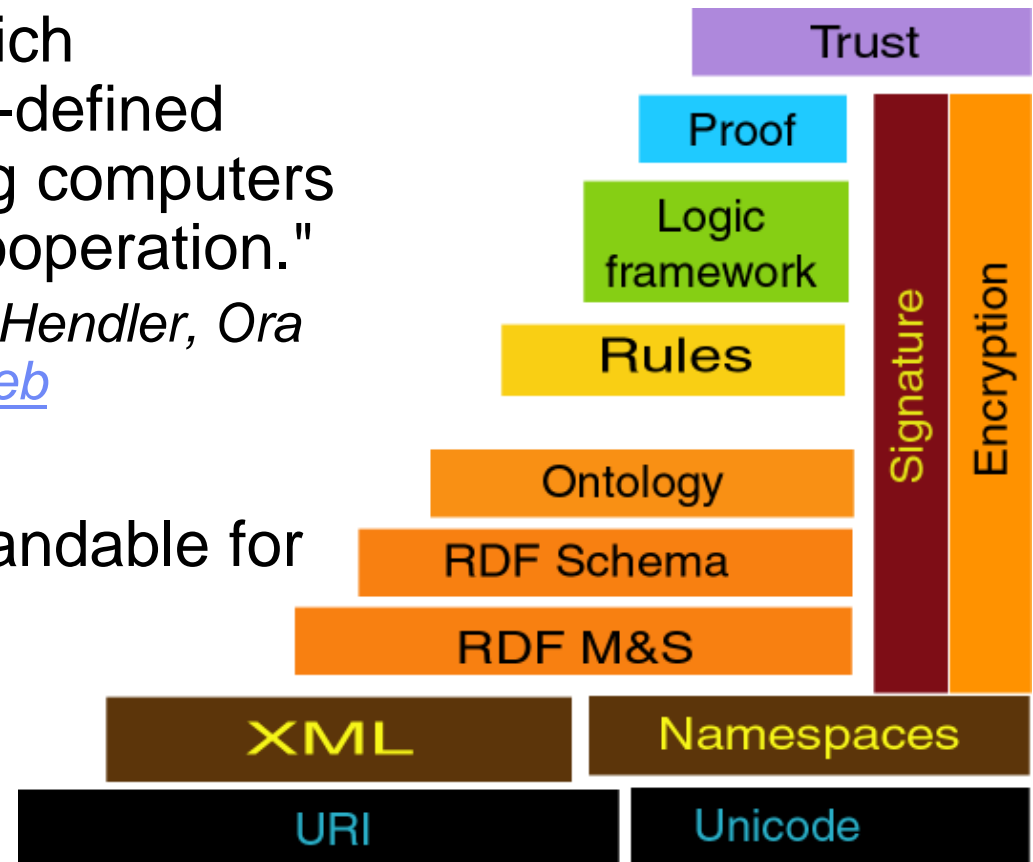
---

- Semantic Web – Ein Einführung (Teil 1)
  - Semantic Web Vision
  - RDF
  - RDFS
  - RDF Anfragesprachen / SPARQL
- Semantic Web – Ontologie (Teil 2)
  - Was ist eine Ontologie
  - W3C Web Ontology Language (OWL)
  - OWL 1.1 und OWL 2
- Semantic Web – Regeln und Erweiterte Konzepte (Teil 3)

# Die Semantic Web Vision

# Semantic Web – Eine Einführung

- "The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation."
  - *Tim Berners-Lee, James Hendler, Ora Lassila, [The Semantic Web](#)*
- „Make the Web understandable for machines“



# Das heutige Web

---

- Web Inhalte sind typischerweise für die Benutzung durch Menschen konzipiert
  - Automatisch generierte Webinhalte aus z.B. Datenbanken werden ohne Semantik und ihre ursprüngliche Struktur präsentiert
- Fehlende Unterstützung für automatische Bearbeitung durch Tools
  - Hauptsächlich nur Schlüsselwort-basierte Suchmaschinen

# Probleme der Schlüsselwort-basierten Suche

---

- Viele Ergebnisse (high recall), aber niedrige Genauigkeit (low precision)
- Oder niedrige bis keine Ergebnisse (low recall)
- Ergebnisse sind einzelne Web Seiten (URIs)
- Ergebnisse müssen vom Menschen interpretiert und kombiniert werden
- Ergebnisse der Suche sind nicht direkt verwertbar für Softwarewerkzeuge
  - Semantische Informationen über die Bedeutung von Webinhalten fehlen

# Nachteile von XML

---

- XML ist eine universelle Metasprache zur Definition von Auszeichnungen (Markup)
- Stelle eine uniforme Plattform dar, zum Austausch von Daten und Metadaten zwischen Anwendungen
- Aber, XML bietet keine Möglichkeiten, um über die Semantik (Bedeutung) von Daten zu sprechen
- z.B. keine explizite Bedeutung verknüpft mit der Verschachtelung von Tags
  - Die Anwendung muss die Verschachtelung interpretieren

# Verschachtelung von Tags in XML

---

*Adrian Paschke is a lecturer of Network Based Information Systems*

```
<course name="NBI">  
  <lecturer>Adrian Paschke</lecturer>  
</course>  
<lecturer name="Adrian Paschke">  
  <teaches>NBI</teaches>  
</lecturer>
```

***Gegenteilige Verschachtelung, aber selbe Information!***

# Der Semantik Web Ansatz

---

- Repräsentation von Web Inhalten in einer maschinen-verarbeitbaren Form
    - Mittels Auszeichnung durch **Metadaten** und **Ontologien**
  - Benutzung von intelligenten Schlussfolgerungstechniken (**Logik und Inferenz**) um Web-Inhalte automatisch zu verarbeiten und neues Wissen zu erschließen
  - Automatische Werkzeuge, z.B. **Rule-based Expert Systems, Web Services, Software Agenten**
- ➔ Das Semantik Web ist eine Ergänzung des existierenden WWW

# Bausteine des Semantic Webs (und darüber hinaus)

---

1. Explizite Metadaten im WWW
2. Ontologien
3. Logik und Inferenz
4. Software Agenten und Web Services

# 1. Explizite Metadaten im WWW

---

- Metadaten sind *Daten über Daten*.
- Metadaten im Web:
  - *Maschinell verarbeitbare Information über Information im Web*
  - Projekte im Web: PICS, Dublin Core (15 Kernattribute), RDF, IEEE LOM (Learning Objects Metadata), ...
- Problembereiche:
  - Syntax:
    - In welcher Form werden Metadaten ausgetauscht (Kandidat: XML)
  - Semantik:
    - Welche Metadaten können/dürfen für eine Ressource vergeben werden (Bestimmung eines Vokabulars, eines Schemas)
  - Assoziationsproblem:
    - Wie werden Metadaten mit Ressourcen verknüpft (wer vergibt Metadaten, werden Metadaten getrennt verwaltet, etc).

## 2. Ontologien

---

- Eine Ontologie ist eine explizite und formale Spezifikation einer Konzeptualisierung
- Ontologien beschreiben das gemeinsame Wissen einer Domäne (Semantik):
  - Semantische Interoperabilität zwischen (verknüpften) Vokabularen
- Typische Komponenten
  1. Klassen (Konzepte) der Domäne
  2. Eigenschaften (Rollen) der Klassen
  3. Einschränkungen (Constraints) der Eigenschaften
  4. Individuen (Ausprägungen) der Klassen → Wissensbasis

# 3. Logik und Inferenz

---

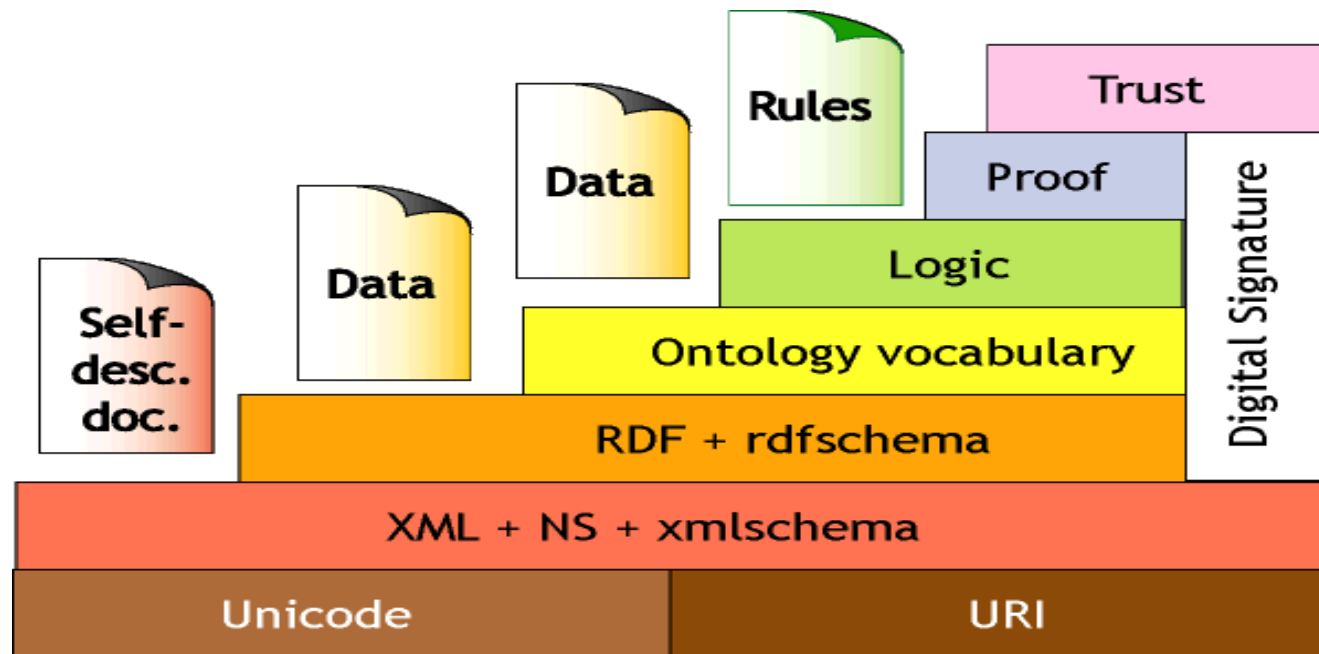
- Logik ist die Disziplin, die sich mit dem Prinzipien des Schlussfolgerns beschäftigt
- Formale Sprachen zur Repräsentation von Wissen mit klarer Semantik
  - Deklarative Wissensrepräsentation:  
*Beschreibung dessen **was** gilt, ohne sich um die Interpretation und Schlussfolgerungen zu kümmern, d.h. das **wie** wird an einen automatischen Interpreter/Reasoner übergeben*
- Automatische Reasoner, z.B. Regelmaschinen, können Schlussfolgerungen von gegebenen Wissen ableiten (Inferenz)

# 4. Softwareagenten und Web Services

---

- **Intelligente Softwareagenten** arbeiten autonom und proaktive
  - Sie besitzen eine eigenen Wissensbasis mit Entscheidungslogik (Regelbasiert, Expertenwissen)
  - Beispiele: Personeller Agent, Suchroboter
- „**Ein Web Service bzw. Webdienst** ist eine Software-Anwendung, die mit einem URI eindeutig identifizierbar ist und deren Schnittstellen als XML-Artefakte definiert, beschrieben und gefunden werden können. Ein Web Service unterstützt die direkte Interaktion mit anderen Softwareagenten unter Verwendung XML-basierter Nachrichten durch den Austausch über internetbasierte Protokolle.“ (Wikipedia)
- => keine klare Trennung zwischen Agent vs. Web Service
- => aber höhere Entscheidungsautonomie bei Agenten

# Schichtenarchitektur des Semantik Web



W3C Stack 2003

- Prinzipien (Semantic Web Stack 2003)
  - Die Entwicklung erfolgt in Schichten – jede Schicht baut auf der andern auf
  - Abwärtskompatibilität
  - Nach oben: Teilweises Verstehen
  - **Aber:** Neue Vorschläge zum Aufbau existieren, z.B. mit 2 Türmen (siehe Ausblick)

# Überblick Semantik Web Schichten (1)

---

- XML-Schicht
  - Syntaktische Basis
- RDF-Schicht
  - RDF als Datenmodell für Fakten und Metadaten
  - RDF Schema (RDFS) als einfache Ontologiesprache
- Ontologie-Schicht
  - Ausdrucksstarke Ontologiesprachen
  - Aktueller Standard: Web Ontology Language (OWL)

# Überblick Semantik Web Schichten (2)

---

- Logik-Schicht
  - Erweiterung der Ontologiesprachen, z.B. mit komplexen Regeln
- Beweis-Schicht (Proof Layer)
  - Beweisgenerierung, -austausch und –validierung
- Vertrauens-Schicht (Trust Layer)
  - Digitale Signaturen
  - Vorschläge (recommendations),  
Bewertungsagenturen (ratings)

# Resource Description Framework

# RDF

---

- WWW besteht aus vernetzten Ressourcen
- Resource Description Framework - ein Standard zur Beschreibung von Ressourcen
- Allgemeine Metadaten und deren Struktur können mit RDF abgebildet werden
- RDF kann andere Definitionsvorschläge für Meta-Information im Web abbilden (z.B. Dublin Core, P3P)
- RDF kann in XML repräsentiert werden (ist aber ansonsten unabhängig von XML)

# RDF Entstehungsgeschichte

---

- Beginn:
  - Erweiterung des PICS Systems (Platform for Internet Content Selection)
- 22.2. 1999: W3C Empfehlung „RDF Model & Syntax Specification“
- 27.3.2000: W3C Recommendation „RDF Schema Specification 1.0“
- 25.9.2001: W3C Arbeitspapier „RDF Model Theory“
- Heute: Vorschläge zur Erweiterung mit Negation

# RDF-Datenmodell

---

- RDF-Daten bestehen aus einem Tripel:
  - *Knoten* (resources – eine beliebige URI)
  - Attributen (properties)
  - Werten (values)
- Auch Werte können wiederum Knoten sein
- Die RDF-Information bildet einen gerichteten und beschrifteten Graphen
- Zum Datenaustausch von RDF-Modellen wird XML verwendet
- XML Namespaces können Namenskonflikte verhindern

# Fundamentale Konzepte von RDF

---

- Die fundamentalen Konzepte von RDF sind:
  - resources
  - properties
  - statements

# Resources

---

- Eine Resource ist ein Objekt, ein “Ding” (thing) über das man reden kann
  - z.B. Autoren, Bücher, Orte, Menschen
- Jede Resource hat einen **URI** (Universal Resource Identifier)
- Eine URI kann sein
  - eine URL (Web Adresse) oder
  - eine andere Art eines „Unique Identifiers“

# Properties

---

- Properties sind eine spezielle Form von Ressourcen
- Sie beschreiben die Relation zwischen Ressourcen
  - z.B. “written by”, “age”, “title”, etc.
- Properties werden auch durch URIs identifiziert

# Statements

---

- Statements (Aussagesätze) verbinden Ressourcen durch Properties
- Ein Statement ist ein Object-Attribute-Value Triple
  - Es besteht aus einer Resource, einer Property, und einem Wert
- Werte können sein Ressourcen or **Literale**
  - Literale sind atomare Werte (strings)

# Die drei Sichten auf ein Statement

---

- Ein Triple
- Ein Stück eines Graphen
- Ein Stück XML Code

Dementsprechend kann ein RDF Dokument gesehen werden als:

- Eine Menge von Triples
- Ein Graph (Semantisches Netz)
- Ein XML Dokument

# Statements als Triples

---

(“Adrian Paschke”,

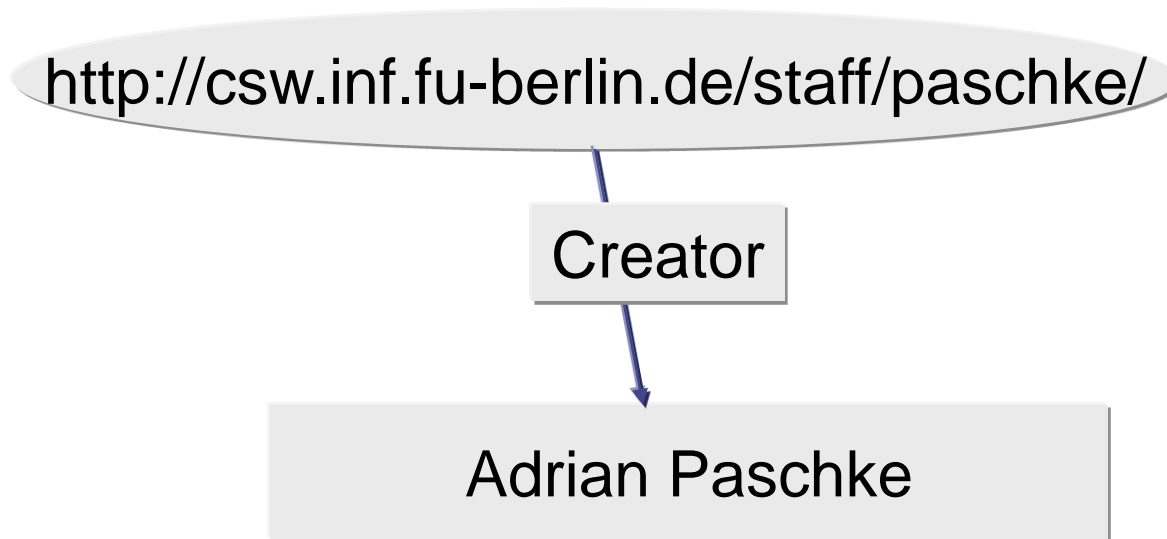
[http:// csw.inf.fu-berlin.de /site-owner](http://csw.inf.fu-berlin.de/site-owner),

[http:// csw.inf.fu-berlin.de](http://csw.inf.fu-berlin.de))

- Das Triple  $(x,P,y)$  kann als logische Formel  $P(x,y)$  gesehen werden
  - Das binäre Prädikat  $P$  verbindet Objekt  $x$  zu Objekt  $y$
  - RDF unterstützt nur **binäre Prädikate** (properties)

# Erstes RDF-Diagramm

---



Subjekt (= Ressource): `http://csw.inf.fu-berlin.de/staff/paschke/`

Prädikat (= Eigenschaft): `Creator`

Objekt (= Wert): `Adrian Paschke`

Leseweise: *<Ressource> hat <Eigenschaft> <Wert>*

# RDF/XML-Version

---

```
<?xml version="1.0"?>
```

```
<rdf:RDF
```

```
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:c="http://description.org/schema/">
```

```
<rdf:Description about="http://csw.inf.fu-berlin.de/staff/paschke/">
```

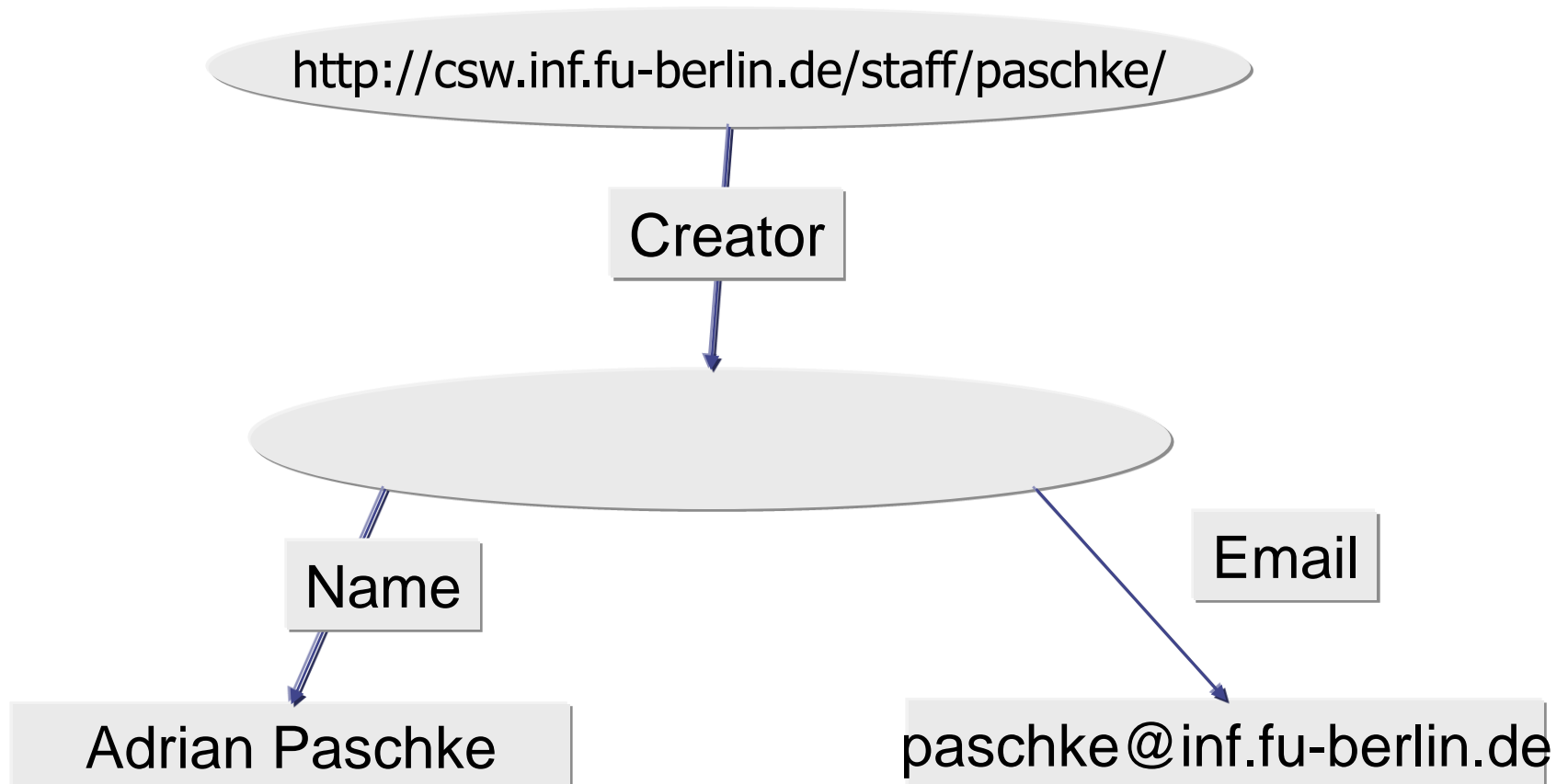
```
  <c:Creator>Adrian Paschke</c:Creator>
```

```
</rdf:Description>
```

```
</rdf:RDF>
```

# Erweitertes RDF-Diagramm

---



# RDF/XML-Version

---

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:c="http://description.org/schema/">
```

```
<rdf:Description about="http://csw.inf.fu-berlin.de/staff/paschke/">
```

```
<c:Creator>
```

```
<rdf:Description>
```

```
<c:Name>Adrian Paschke</c:Name>
```

```
<c:Email>paschke@inf.fu-berlin.de</c:Email>
```

```
</rdf:Description>
```

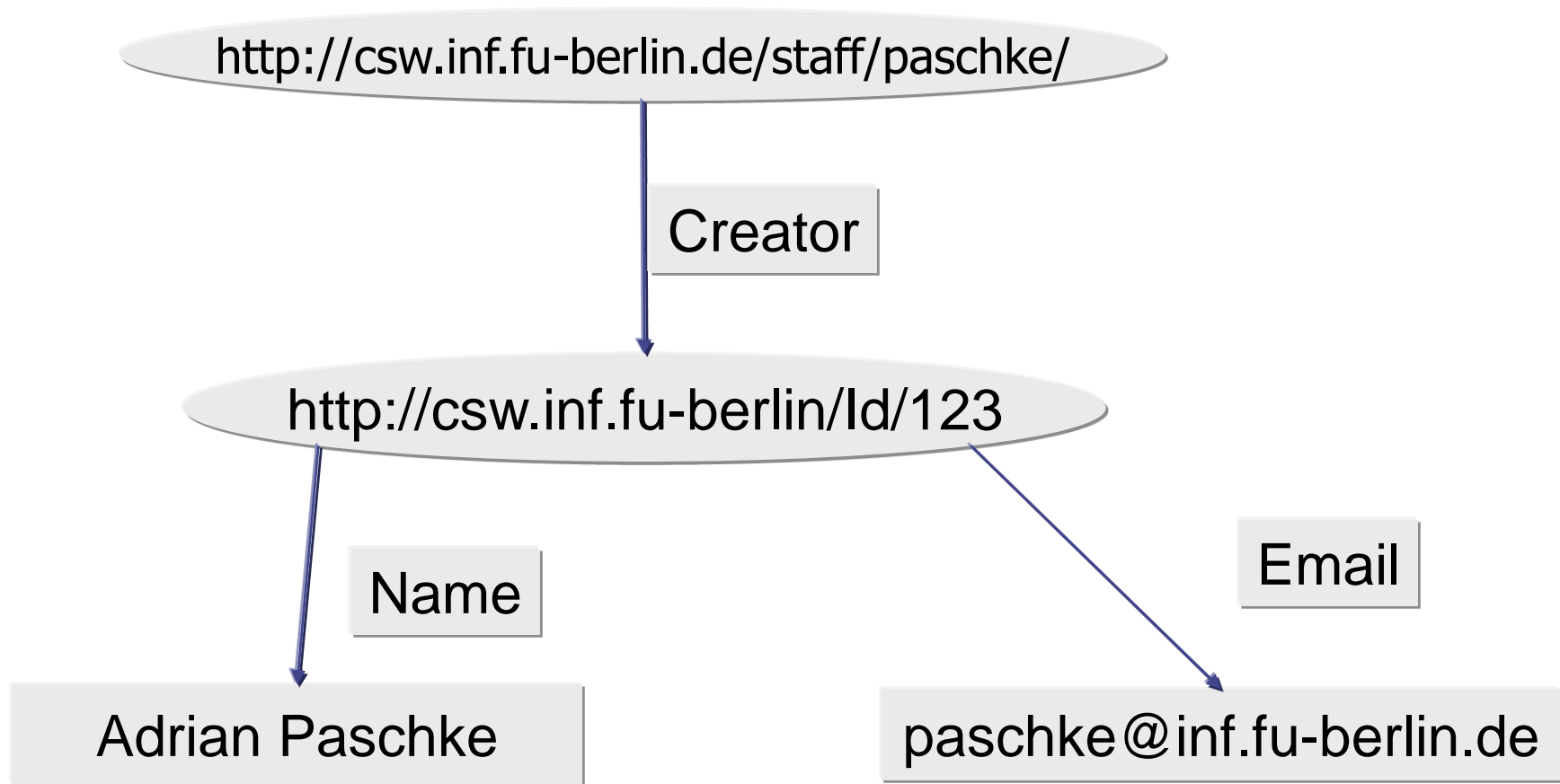
```
</c:Creator>
```

```
</rdf:Description>
```

```
</rdf:RDF>
```

# Weiteres RDF-Diagramm

---



# RDF/XML-Version

---

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:s="http://description.org/schema/">
```

```
  <rdf:Description about="http://csw.inf.fu-berlin/staff/bichler">
    <s:Creator rdf:resource="http://csw.inf.fu-berlin/Id/123" />
  </rdf:Description>
```

```
  <rdf:Description about="http://csw.inf.fu-berlin/Id/123">
    <s:Name>Adrian Paschke</s:Name>
    <s:Email>paschke@inf.fu-berlin.de</s:Email>
  </rdf:Description>
```

```
</rdf:RDF>
```

# Container in RDF

---

- Bag:
  - An unordered collection
- Sequence:
  - An ordered collection
- Alternative:
  - Unordered set of alternatives

# Beispiel Containertyp „Alternative“

---

```
<rdf:RDF>
  <rdf:Description about="http://x.org/packages/X11">
    <c:DistributionSite>
      <rdf:Alt>
        <rdf:li resource="ftp://ftp.x.org"/>
        <rdf:li resource="ftp://ftp.cs.purdue.edu"/>
        <rdf:li resource="ftp://ftp.eu.net"/>
      </rdf:Alt>
    </c:DistributionSite>
  </rdf:Description>
</rdf:RDF>
```

# RDF Collections

---

- Eine Einschränkung dieser Container ist, dass es keine Möglichkeit gibt diese explizit zu schließen
  - d.h. zu sagen “dies sind **alle** Mitglieder in der Collection”
- RDF bietet Unterstützung in Form von **RDF Collections** zur Beschreibung von Gruppen, welche **nur** die spezifizierten Elemente enthält.
  - **list** Struktur im RDF Graphen
  - Konstruiert durch Benutzung des Collection Vokabulars: **rdf:List**, **rdf:first**, **rdf:rest** und **rdf:nil**

# Reification - Reifizierung

---

- In RDF ist es möglich Aussagen über Aussagen zu machen
  - **Peter believes that Adrian is the creator of <http://www.csw.inf.fu-berlin.de/>**
- Solche Aussagen können benutzt werden, um Glauben oder Vertrauen in andere Aussagen zu beschreiben
- Jeder Aussage wird ein eindeutiger Identifizierer zugewiesen
  - Diese kann benutzt werden um die Aussage zu referenzieren

# Reification (2)

---

- Führe ein Hilfsobjekt ein (z.B. **belief1**)
- Setze es in Beziehung zu jedem der 3 Teile der Originalaussage durch **Subjekt**, **Prädikat** und **Objekt**
- Im Beispiel
  - **Subjekt** von **belief1** ist **Peter**
  - **Prädikat** von **belief1** ist **creator**
  - **Objekt** of **belief1** ist **<http://www.csw.inf.fu-berlin.de/>**

# Datentype

---

- Datentypen werden in Programmiersprachen zur Interpretation verwendet
- In RDF werden typisierte Literale verwendet, wenn notwendig

**(“Adrian”,**

**<http://www.mydomain.org/age>,**

**“31”^^<http://www.w3.org/2001/XMLSchema#integer>)**

# Datentypen (2)

---

- ^^-Notation bezeichnet den Typ des Literals
- Datentypen sind oft XML Schema Datentypen
  - Aber: Nutzung jedes extern definierten Datentypisierungsschemas ist in einem RDF Dokument möglich
- XML Schema predefiniert eine große Anzahl von Datentypen
  - z.B. Booleans, integers, floating-point numbers, times, dates, etc.

# Datentypen (3)

---

- Das Attribut **rdf:datatype="&xsd:integer"** wird benutzt um den Datentyp des Wertes der "age" Property zu definieren

```
<rdf:Description rdf:about="949318">  
  <uni:name>Adrian Paschke</uni:name>  
  <uni:title>Associate Professor</uni:title>  
  <uni:age  
    rdf:datatype="&xsd:integer">31</uni:age>  
</rdf:Description>
```

# Dublin Core – Metadaten in World Wide Web

---

- Dublin Core Metadata Element Set ist in verschiedenen Syntaxformaten darstellbar (z.B. RDF oder HTML)
  - 15 Dublin Core Kernelemente
- In HTML:

```
<HTML><HEAD>  
<TITLE>IBIS Homepage</TITLE>  
<META NAME=„DC.TITLE“ CONTENT=„Homepage of Internet-based  
Information Systems“>  
<META NAME=„DC.SUBJECT“ CONTENT=„decision support systems,  
supply chain management, multidimensional auctions“>  
<META NAME=„DC.CREATOR“ CONTENT=„M. Bichler“>  
</HEAD>
```

...

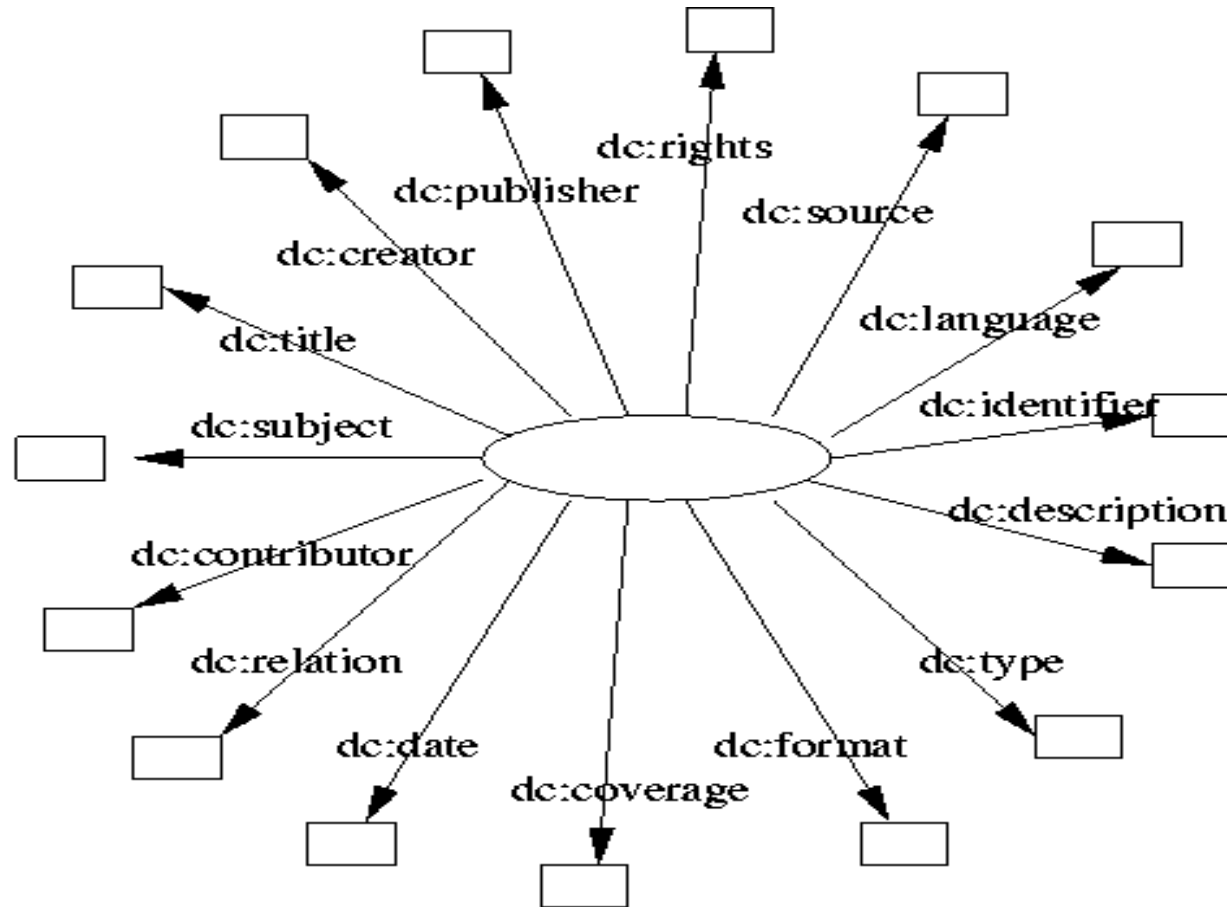
# Dublin Core in RDF

---

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:dc="http://purl.org/dublin_core/schema/">
  <rdf:Description rdf:about="http://inf.fu-berlin.de/en/groups/ag-csw">
    <dc:creator>Adrian Paschke</dc:creator>
    <dc:title>CSW Homepage</dc:title>
  </rdf:Description>
</rdf:RDF>
```

# Dublin Core Elemente

---



# Dublin Core in RDF

---

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:dc="http://purl.org/dublin_core/schema/">
  <rdf:Description rdf:about="responder.ruleml.org">
    <dc:creator>A. Paschke</dc:creator>
    <dc:title>Rule Responder</dc:title>
  </rdf:Description>
</rdf:RDF>
```

# FOAF 0.1

---

## FOAF Basics

- [Agent](#)
- [Person](#)
- [name](#)
- [nick](#)
- [title](#)
- [homepage](#)
- [mbox](#)
- [mbox\\_sha1sum](#)
- [img](#)
- [depiction](#) (depicts)
- [surname](#)
- [family\\_name](#)
- [givenname](#)
- [firstName](#)

## Personal Info

- [weblog](#)
- [knows](#)
- [interest](#)
- [currentProject](#)
- [pastProject](#)
- [plan](#)
- [based\\_near](#)
- [workplaceHomepage](#)
- [workInfoHomepage](#)
- [schoolHomepage](#)
- [topic\\_interest](#)
- [publications](#)
- [geekcode](#)
- [myersBriggs](#)
- [dnaChecksum](#)

## Online Accounts / IM

- [OnlineAccount](#)
- [OnlineChatAccount](#)
- [OnlineEcommerceAccount](#)
- [OnlineGamingAccount](#)
- [holdsAccount](#)
- [accountServiceHomepage](#)
- [accountName](#)
- [icqChatID](#)
- [msnChatID](#)
- [aimChatID](#)
- [jabberID](#)
- [yahooChatID](#)

## Projects and Groups

- [Project](#)
- [Organization](#)
- [Group](#)
- [member](#)
- [membershipClass](#)
- [fundedBy](#)
- [theme](#)

## Documents and Images

- [Document](#)
- [Image](#)
- [PersonalProfileDocument](#)
- [topic](#) (page)
- [primaryTopic](#)
- [tipjar](#)
- [sha1](#)
- [made](#) (maker)
- [thumbnail](#)
- [logo](#)

See details at  
<http://xmlns.com/foaf/0.1/>

# Ein Kritischer Blick auf RDF: Binäre Prädikate

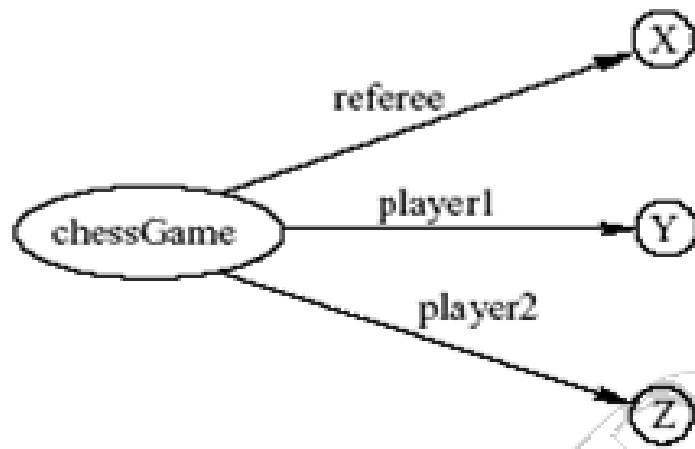
---

- RDF benutzt nur binäre Prädikate
  - Dies ist eine Einschränkung, da oft Prädikate mit mehr als 2 Argumenten benötigt werden
  - Aber man kann dies auch mit binären Prädikaten simulieren
- Beispiel: **referee(X,Y,Z)**
  - **X** ist ein Schiedsrichter in einem Schachspiel zwischen den Spielern **Y** und **Z**

# Ein Kritischer Blick auf RDF: Binäre Prädikate (2)

---

- Wir führen ein:
  - Eine neue Hilfsresource **chessGame**
  - Die binären Prädikate **ref**, **player1**, und **player2**
- Wir können **referee(X,Y,Z)** darstellen als:



# Ein Kritischer Blick auf RDF: Eigenschaften (Properties)

---

- Properties sind spezielle Arten von Ressourcen
  - Properties können als Objekt in einem Object-Attribute-Value Triple (statement) vorkommen
  - Sie sind unabhängig von der Resource definiert
- Diese Möglichkeit gibt viel Flexibilität
- Aber eher ungewöhnlich in Modellierungssprachen und OO Programmiersprachen
- Kann für Modellierer verwirrend sein

# Ein Kritischer Blick auf RDF: Reifizierung

---

- Der Reifizierungsalgorithmus ist sehr mächtig
- Er erscheint als zu mächtig in einer einfachen Sprache wie RDF
- Aussagen über Aussagen zu machen, führt zu einem hohen Maß an Komplexität, welches nicht in der Grundsicht des Semantic Webs sein sollte
- Besser wäre es Reifizierung in mächtigeren Schichten, mit mehr Ausdrucksmächtigkeit, einzubinden.

# Ein Kritischer Blick auf RDF: Zusammenfassung

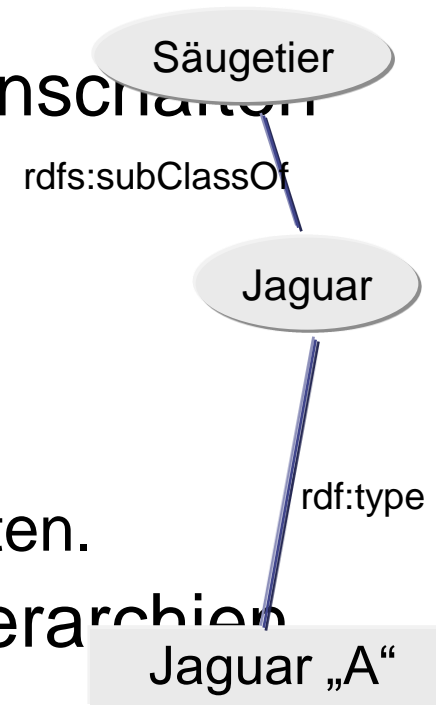
---

- RDF hat seine Eigenarten und ist keine optimale Modellierungssprache, **aber**
- Es ist bereits ein de facto Standard im Semantic Web und darüber hinaus
- Es hat genügende Ausdrucksmächtigkeit
  - Zumindest um weitere Schichte darauf aufzubauen
- Die Benutzung von RDF erlaubt Informationen eindeutig in ein Model einzubinden

# Resource Description Framework Schema

# RDF Schema

- Sprache zur Spezifikation von Schemata
  - Definiert Klassen und deren Eigenschaften, die in einer RDF Description mit Werten belegt werden können.
- Ein Schema legt die Klassen und Eigenschaften eines Anwendungsbereiches fest
- Definition von Beziehungen
  - zwischen Klassen,
  - zwischen Eigenschaften (*Properties*),
  - sowie zwischen Klassen und Eigenschaften.
- Ermöglicht den Aufbau von Konzepthierarchien
  - Jaguar erbt Eigenschaften von Säugetier



# Grundlegende Ideen von RDF Schema

---

- RDF ist eine universelle Sprache, welche es Nutzern erlaubt Ressourcen in ihre eigenen Vokabulare einzubinden
  - RDF nimmt keine Semantik an, noch definiert es eine spezielle für eine Domäne
- Der Nutzer kann das mit **RDF Schema** tun:
  - Klassen und Eigenschaften
  - Klassenhierarchien und Vererbung
  - Eigenschaftshierarchien

# Klassen und ihre Instanzen

---

- Wir müssen unterscheiden zwischen
  - Konkreten Dingen (“things” = individuelle Objekte) in der Domäne: lecture etc.
  - Mengen von Individuen mit gleichen Eigenschaften (properties) genannt Klassen (**classes**) : lecturers, students, courses etc.
- Individuelle Objekte, welche zu einer Klasse gehören, genannt Instanzen (**instances**) der Klasse
- Die Verbindungen zwischen Instanzen und Klassen in RDF durch **rdf:type**

# Warum sind Klassen nützlich

---

- Legen Einschränkungen anhand eines Schemas fest, was in RDF Dokumenten ausgedrückt werden kann
  - Wie in Programmiersprachen
  - z.B  $A+1$ , wobei A ein Array ist
  - Verboten die Formulierung von Nonsense

# Nicht-Sinnvolle Aussagen werden durch die Benutzung von Klassen ausgeschlossen

---

- **NBI is taught by Computer Science**
  - Kurse sollen natürlich nur von Lehrberechtigten unterrichtet werden sollen
  - Einschränkungen der Werte einer Eigenschaft “is taught by” (**range restriction**)
- **Room SS06 is taught by Adrian Paschke**
  - Nur Kurse können gelehrt werden
  - Diese wird durch Einschränkungen über Objekte festgelegt, auf die die Eigenschaft angewandt werden kann (**domain restriction**)

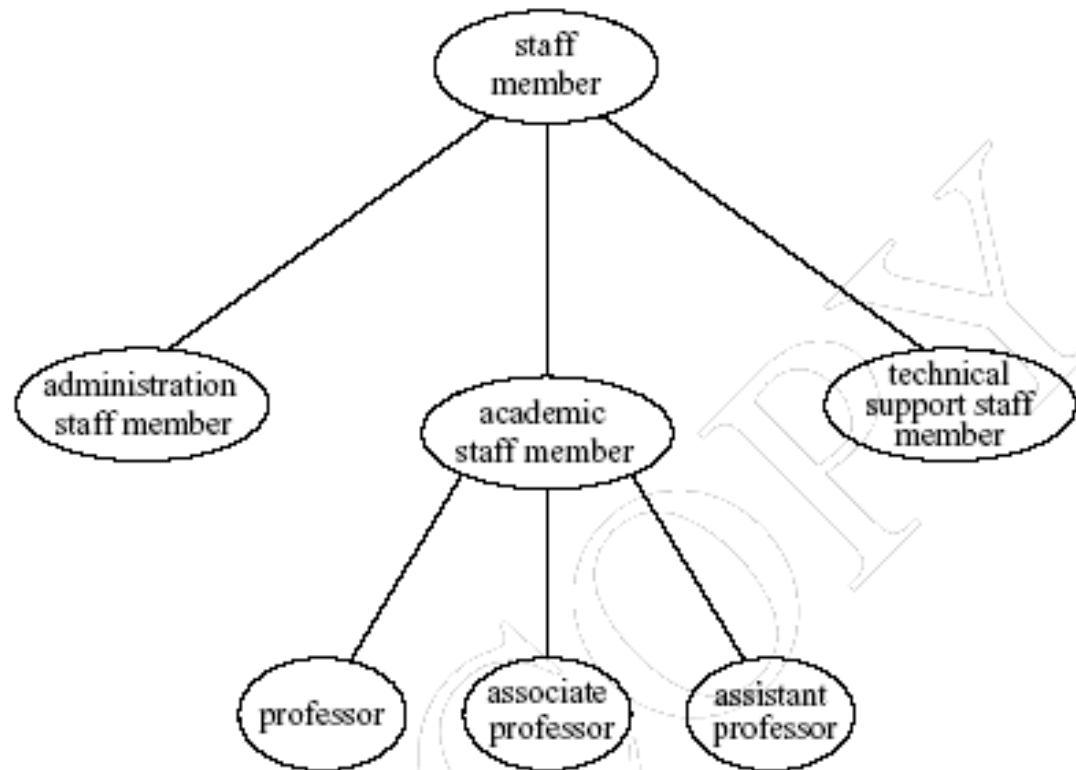
# Klassenhierarchien

---

- Klassen können in Hierarchien organisiert sein
  - A ist eine **Subklasse** von B, wenn A auch eine Instanz von B ist
  - Dann ist B eine **Superklasse** von A
- Ein Subklassengraph muss kein Baum sein
- Eine Klasse kann viele Superklassen haben

# Beispiel

---



# Vererbung in Klassenhierarchien

---

- Range restriction: Kurse müssen von akademischen Mitarbeitern unterrichtet werden
- Adrian Paschke is a professor
- Er bekommt die Erlaubnis zu Unterrichten von der Klasse “**Akademischer Mitarbeiter**” vererbt
- In RDF Schema wird das durch die Semantik von “is a subclass of” festgelegt
  - Es ist nicht Aufgabe der Anwendung (RDF Processing Software) “is a subclass of” zu interpretieren

# Eigenschaftshierarchien

---

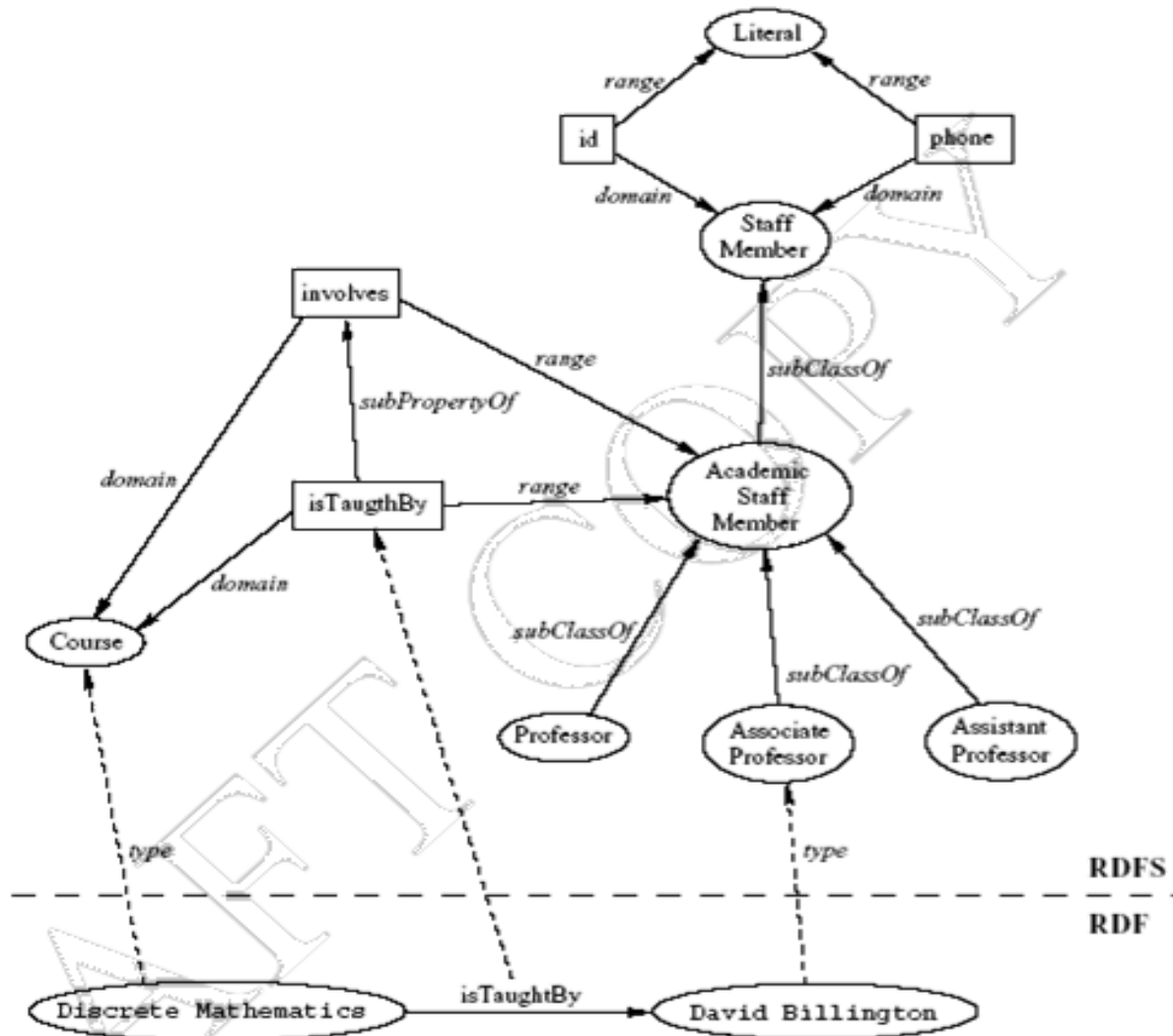
- Hierarchische Verbindungen von Eigenschaften
  - z.B., “is taught by” ist eine Subeigenschaft von “involves”
  - If a course C is taught by an academic staff member A, then C also involves A
- Das Gegenteil ist nicht notwendigerweise der Fall
  - Z.B., A kann Lehrer für Kurs C sein, oder
  - Ein Tutor welcher Studentenarbeiten bewertet, aber C nicht unterrichtet
- P ist eine Subeigenschaft (**subproperty**) von Q, wenn  $Q(x,y)$  wahr ist, immer dann wenn  $P(x,y)$  wahr ist

# RDF Schicht vs RDF Schema Schicht

---

- **NBI is taught by Adrian Paschke**
- Das Schema selbst ist in einer formalen Sprache geschrieben, nämlich RDF Schema, mit die Bestandteile beschrieben werden können:
  - subClassOf, Class, Property, subPropertyOf, Resource, etc.

# RDF Schicht vs RDF Schema Schicht (2)



# Kernklassen von RDF Schema

---

- **rdfs:Resource**  
Superklasse für alles was mit RDF beschrieben werden kann
- **rdf:Property**  
Repräsentiert eine Untermenge von Ressourcen, die Eigenschaften sind
- **rdfs:Class**  
Typ oder Kategorie, ähnlich dem Klassenkonzept in objektorientierten Programmiersprachen. Class ist in RDFS eine Unterklasse von Resource
- **rdfs:Literal**, Klasse aller Literale (strings)
- **rdf:Statement**, Klasse aller reifizierten Aussagen

# Eigenschaften

---

- **rdf:type**

Zeigt an, dass eine Ressource zur einer Klasse gehört, d.h. eine Ressource wird dadurch Instanz einer Klasse und weist die Eigenschaften der Klasse auf

- **rdfs:subClassOf**

Bestimmt Beziehungen zwischen Klassen. Ressourcen, die Instanzen einer Klasse sind, sind auch Instanzen der Superklasse

- **rdfs:subPropertyOf**

Spezialisierung einer Eigenschaft durch Vererbung

Beispiel: Wenn biologicalFather eine Spezialisierung von biologicalParent ist, und Max ist der biologicalFather von John, dann ist Max auch ein biologicalParent von John.

# Beziehung zwischen Kernklassen und Eigenschaften

---

- **rdfs:subClassOf** und **rdfs:subPropertyOf** sind transitiv, per Definition
- **rdfs:Class** ist eine Subklasse von **rdfs:Resource**
  - Da jede Klasse eine Resource ist
- **rdfs:Resource** ist eine Instanz von **rdfs:Class**
  - **rdfs:Resource** ist die Klasse aller Ressourcen, also eine Klasse
- Jede Klasse ist eine Instanz von **rdfs:Class**
  - S.O.

# Hilfseigenschaften

---

- **rdfs:seeAlso**
  - erweist auf eine andere Resource zur Beschreibung
- **rdfs:isDefinedBy**
  - ist eine Subeigenschaft von **rdfs:seeAlso**
- **rdfs:comment**
  - Kommentar
- **rdfs:label:**
  - Ein menschen-freundlicher Name für die Resource

# Reifizierung und Container

---

- **rdf:subject**, verbindet ein reifizierte Aussage zu seinem Subjekt
- **rdf:predicate**, verbindet ein reifizierte Aussage zu seiner Eigenschaft
- **rdf:object**, verbindet ein reifizierte Aussage zu seinem Objekt
- **rdf:Bag**, Klasse für Bags
- **rdf:Seq**, Sequenzen
- **rdf:Alt**, Alternativen
- **rdfs:Container**, Superklasse aller Containerklassen

# Constraints

---

- **rdfs:range**

Der Wert einer Eigenschaft muss von einer bestimmten Klasse sein

Beispiel: Die Eigenschaft Autor darf nur Werte der Klasse Person annehmen

- **rdfs:domain**

Eine Eigenschaft gilt nur für Ressourcen einer bestimmten Klasse

Beispiel: Die Eigenschaft „Autor“ darf nur für Ressourcen benannt werden, die Instanzen der Klasse Buch sind

# RDF Schema Beispiel

---

```
<rdf:RDF xml:lang="en"
  xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#">

  <rdfs:Class rdf:ID="Person">
    <rdfs:comment>The class of humans</rdfs:comment>
    <rdfs:subClassOf rdf:resource=http://www.w3.org/2000/03/example/classes#Animal />
  </rdfs:Class>

  <rdf:Property ID="age">
    <rdfs:range rdf:resource="http://www.w3.org/2000/03/example/classes#Integer" />
    <rdfs:domain rdf:resource="#Person" />
  </rdf:Property>

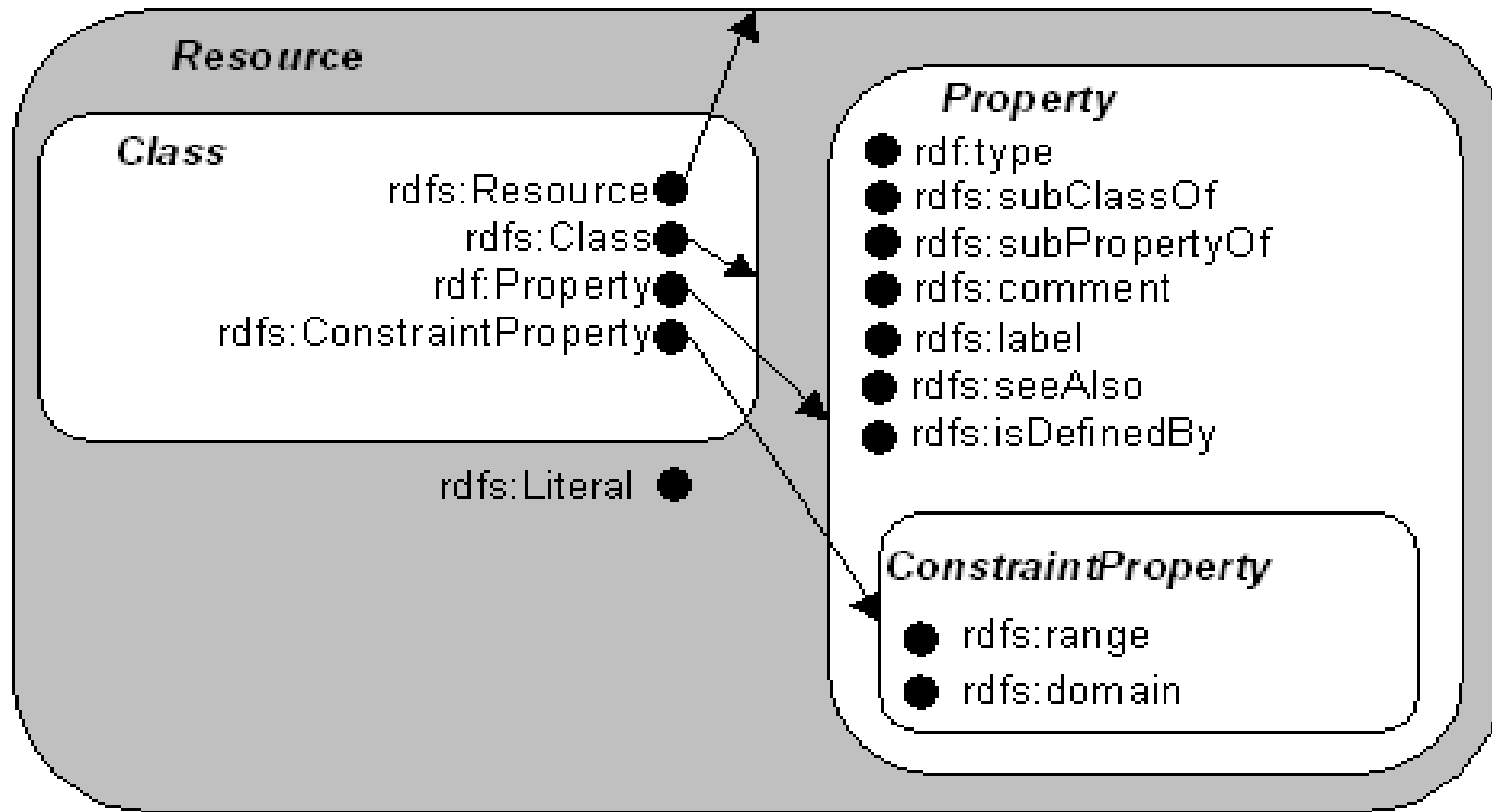
  <rdf:Property ID="maritalStatus">
    <rdfs:range rdf:resource = "#MaritalStatus"/>
    <rdfs:domain rdf:resource = "#Person" />
  </rdf:Property>

  ...

</rdf:RDF>
```

# Übersicht über RDFS

---



**SPARQL**

# SPARQL

---

- SPARQL (SPARQL Protocol and RDF Query Language)
  - “SQL für RDF”
  - W3C Standard Recommendation seit 15 January 2008
- Grundkonzept “Graph Pattern Matching”
- Einfaches Protokol und RDF Query Language
  - Basic Graph Patterns (Conjunctive queries)
  - UNIONS
  - GRAPH Patterns
  - OPTIONAL Patterns
  - FILTERs

# SPARQL Anfrage

---

- **SELECT**
  - Gibt alle, oder ein Subset, der in einem Anfragepattern gebundenen Variablen zurück. Das Format der Antwort kann in XML oder in RDF/XML sein
- **CONSTRUCT**
  - Gibt entweder einen RDF Graphen zurück, welcher Übereinstimmungen für alle Ergebnisse gibt, oder einen RDF Graph erzeugt durch die Substituierung der Variablen in einer Menge von Triple Patterns.
- **DESCRIBE**
  - Gibt einen RDF Graph zurück, der die gefundenen Ressourcen beschreibt
- **ASK**
  - Gibt an, ob ein Anfragepattern übereinstimmt oder nicht

# SPARQL SELECT

---

- **SELECT:**

```
SELECT Variables  
FROM Dataset  
WHERE Pattern
```

- **Beispiele:**

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>  
SELECT ?name  
WHERE ( ?x foaf:name ?name )
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>  
SELECT *  
WHERE ( ?x foaf:name ?name )
```

# SPARQL CONSTRUCT

---

## ■ CONSTRUCT:

```
PREFIX: vcard: <http://www.w3.org/2001/vcard-  
rdf/3.0#>
```

```
CONSTRUCT * WHERE ( ?x vcard:FN ?name )
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/> PREFIX vcard:  
<http://www.w3.org/2001/vcard-rdf/3.0#>
```

```
CONSTRUCT ( ?x foaf:name ?name ) WHERE ( ?x vcard:FN ?name )
```

# SPARQL DESCRIBE and ASK

---

- **DESCRIBE:**

```
DESCRIBE ?x
```

```
WHERE (?x ent:employeeId "1234")
```

- **ASK:**

```
PREFIX foaf:
```

```
<http://xmlns.com/foaf/0.1/> ASK (?x  
foaf:mbox_sha1sum "ABCD1234")
```

# Zusammenfassung RDF/RDFS

---

- RDF ist die Basis zur Darstellung und die Verarbeitung von Metadaten
- RDF hat ein Graphen-basiertes Datenmodell
- RDF hat eine XML Syntax um die syntaktische Interoperabilität zu gewährleisten.
  - XML und RDF ergänzen einander; RDF unterstützt die semantische Interoperabilität
- RDF hat eine dezentralisierte Philosophie und erlaubt das inkrementelle Aufbauen von Wissen, sowie die Wiederverwendung und gemeinsame Nutzung

# Zusammenfassung (2)

---

- RDF ist Domänen-unabhängig
- RDF Schema stellt ein Mechanismus zur Beschreibung spezifischer Domänen bereit
- RDF Schema ist eine primitive Ontologiesprache
  - Es bietet einfache Modellierungsprimitive mit festgelegter Bedeutung
- Schlüsselkonzepte von RDF Schema sind Klassen, Subklassenbeziehungen, Eigenschaften und Subeigenschaftsbeziehungen, sowie Domänen- und Wertebereichsbeschränkungen
- Es gibt Anfragesprachen für RDF und RDFS

# Zusammenfassung (3)

---

- RDF Schema ist eine sehr einfach als Modellierungssprache für das Web
- Viele wünschenswerte Modellierungsprimitive fehlen
- Daher benötigen wir eine Ontologieschicht die auf RDF aufbaut und mächtiger als RDF Schema ist.

# Web Ressourcen

---

- RDF Specification
  - <http://www.w3.org/RDF/>
- Ora Lassila; Introduction to RDF Metadata
  - <http://www.w3c.org/TR/NOTE-rdf-simple-intro-971113.html>
- RDF Schema
  - <http://www.w3.org/TR/rdf-schema/>
- Buch
  - Grigoris Antoniou, Frank van Harmelen: Semantic Web Primer