Data Models and Query Languages

Prof. Dr. Peter Fischer

Organisational (1)

- Lecture:
  - Monday 16:15 – 18:00
  - Thursday 10:15 – 11:00
- Exercises
  - Thursday 11:15 – 12:00
- Exercises mandatory, 50 % successful hand-ins necessary for exam admittance

Organisational (2)

- No lecture April 30th
  (business trip + “Brückentag”)
- No exercise on April 26th
- Instead, 2 hours of lecture on April 26th
- First exercise handed out end of this week

Structure of the lecture

1. Data Models/Query Languages beyond relational
   - Semantic Web: RDF/SPARQL
   - Semistructured Data: XML/XQuery
   - Deductive Databases: Datalog
   - If interest: temporal, objectrelational, “NoSQL”, …
2. Formal Foundations of Query Languages
   - Precise Semantics
   - Complexity
   - Expressiveness
   - Optimizations

Motivation (1)

- All of you know the relational model (hopefully)
- Industry expects knowledge of SQL
- So this is it?
- Relational model not always best solution
  - Complex data difficult to express in “flat tables”
  - Rigidity of schema
  - Core language restricted (SQL 92)
  - Complete language a sprawling mess (SQL 2011)

Motivation (2)

- Typically,
  - we look at the “surface” of a data model and query language
  - Understand at the level of “gut feeling”
  - Sufficient for “normal” use
- Go beyond, and understand
  - precise behavior
  - tradeoff between *expressiveness* and *complexity*
  - *connections* and *communalities* between *different* models and languages

What make Databases special?

1. A well-defined data model
2. A declarative query language

$\Rightarrow$ Users describe what they want to retrieve, not how

$\Rightarrow$ Physical data independence: abstract away details like storage, work purely on logical model

$\Rightarrow$This lecture is all about this logical level

Data Model / Query Language Stack

- Data Model Definition
- Metadata description model
- Data Definition Language
- Query Language
- Data Manipulation Language

- Cross-cutting: Formal Semantics

Possible classification of Models

- Means of structure:
  - Relational
  - Tree/Semistructured
  - Graph-Oriented
- N.B. Not completely orthogonal
- Other dimensions of classification
  - Interpretation: Syntax, Model, Knowledge
  - …

Relational Model

- Relation: Set of data items (tuples)
- Each tuple of a relation has same attributes
- Express complex data by decomposition
- Extensive theory on
  - Data Model
  - Expressiveness
  - Normalization
  - …

Title	ECTS	Prereq
DBIntro	6	NULL
DMQL	6	DBIntro

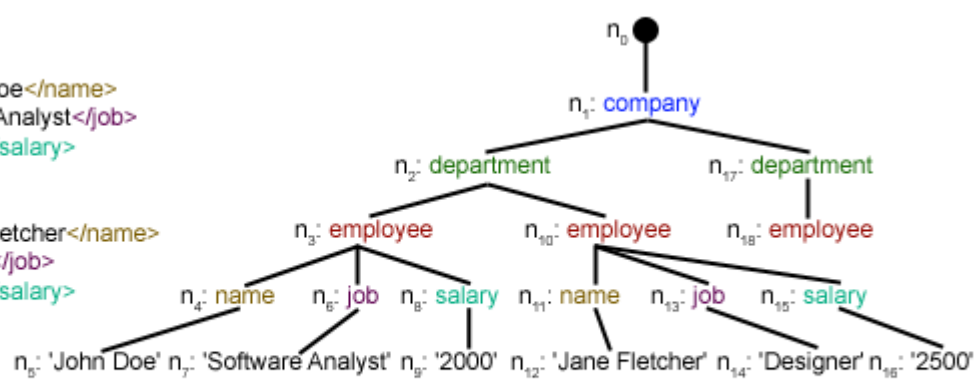
MatNr	Title
1234	DMQL
4711	DBIntro

MatNr	Name
1234	A.B.
4711	X.Y.

Semistructured Model

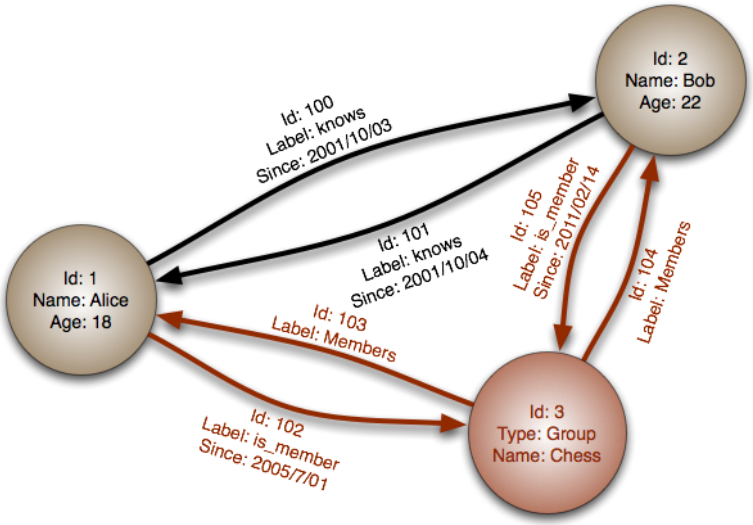
- Structural flexibility, extensability
- Self-Describing
- Often Tree-Like structure
- Often Markup languages
- Examples:
  - XML
  - JSON

```
<company>
 <department>
  <employee>
   <name>John Doe</name>
   <job>Software Analyst</job>
   <salary>2000</salary>
  </employee>
  <employee>
   <name>Jane Fletcher</name>
   <job>Designer</job>
   <salary>2500</salary>
  </employee>
 </department>
 <department>
  <employee>
  </employee>
 </department>
</company>
```



Graph Model

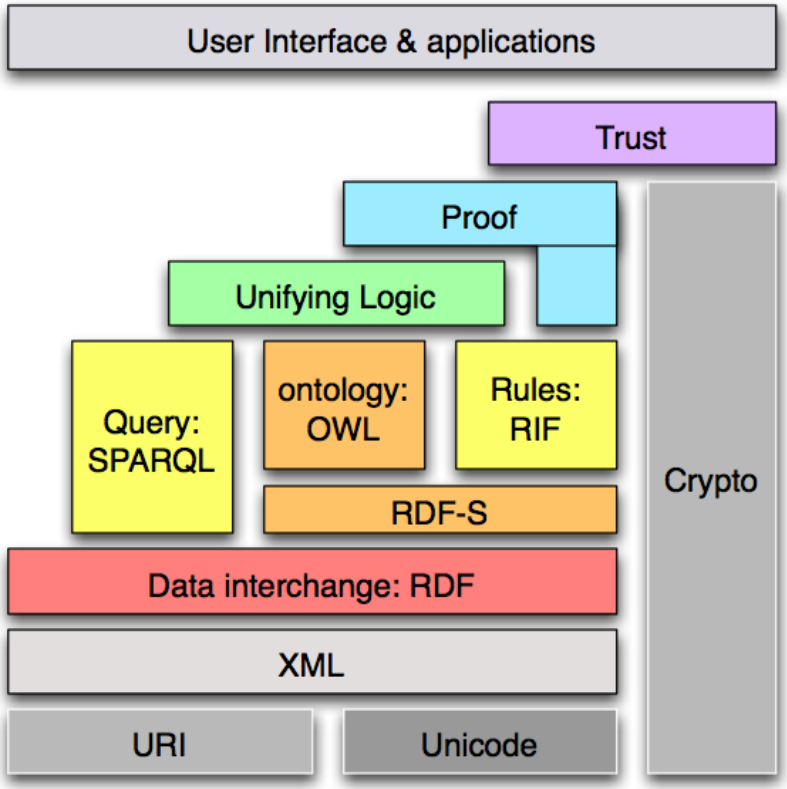
- Allow arbitrary connections
- High flexibility in connectedness
- High flexibility in structure
- Many novel applications:
  - (Social) networks
  - Scienctific data
  - Knowledge Management
- Examples:
  - RDF



Recap of “Web Stack” Basics

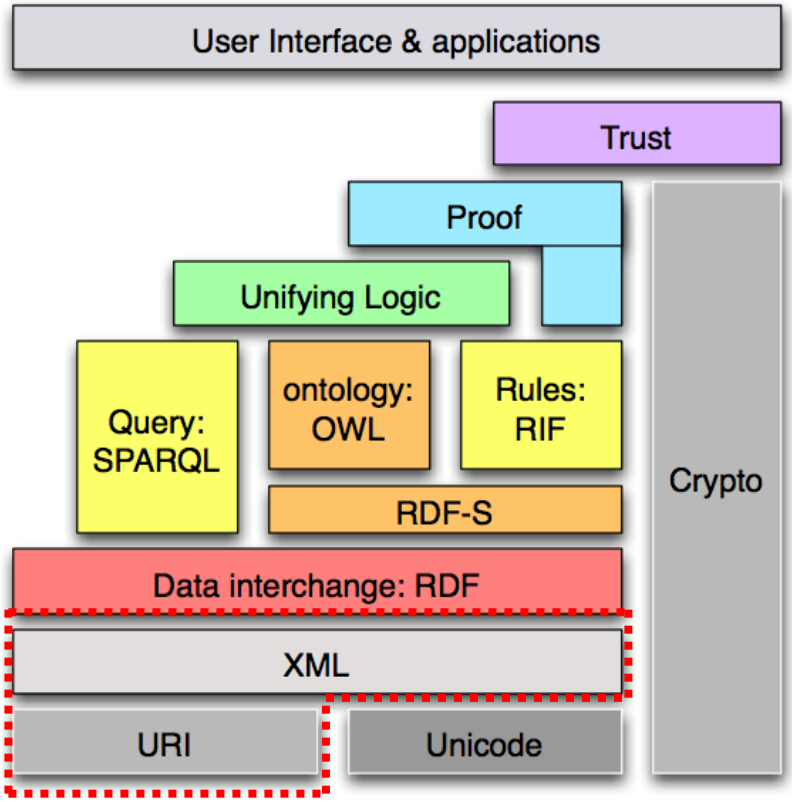
- Semantic Web/RDF/SPARQL as well as XML/XQuery are part of W3C “stack”
- Abstract principles
  - Autonomy
  - Openness
  - Compatibility (upward+downward)
  - Platform- and vendor independance

Semantic Web Stack



- Part of complete W3C stack,
- Upper parts varies (e.g. Web Services stack)

Common Foundations



- URI: Resource Identification
- XML Data Syntax

Universal Resource Identifiers

- URL (Universal Resource Locators): deferenceable identifier on the Web
  - The target of an URL pointer is an file (virtual or materialized)
- URIs (Unique Resource Identifier): general purpose key to resources on the Web
  - Uniquely identifies a resource
  - Target is not an HTML file, can be anything (schema, table, file, entity, object, tuple, person, physical item, etc)
  - Lifetime and scope of this “key” is user dependent
- IRI (Internationalized Resource Identifiers)
  - Allow non Latin characters (Chinese, Arabic, Japanese, etc)
- URL, URI, IRIs
  - All strings
  - Very LONG strings

URI Syntax

- scheme:[//authority]path[?query][#fragment]
- scheme: type of URI, e.g. http, ftp, mailto, file, irc
- authority: typically a domain name
- path: e.g. /etc/passwd/
- query: optional; provides non-hierarchical information. Usually for parameters, e.g. for a web service
- fragment: optional; often used to address part of a retrieved resource, e.g. section of a HTML file.
- not all characters are allowed in URIs.

Defining URIs+Examples

- Many common URI pools known
  - Web Standards, Industry Standards
  - Linked Open Data, e.g. DBPedia
- Use your own domain for your URIs
- Some samples

  http://dbpedia.org/resource/Freiburg_im_Breisgau
  urn:issn:1535-3613
  http://musicbrainz.org/artist/20ff3303-4fe2-4a47-a1b6-291e26aa3438.html

What is XML?

- Lots of <>? (tag soup)
- “The Extensible Markup Language (XML) is the **universal format** for **structured documents** and **data** on the Web.”
- A **syntax** to **serialize** data
- **Basis for family** of **standards**:
  Schema, Web Services, Processing, Semantic Web, …
- Base specifications:
  - XML 1.0, W3C Recommendation Feb '98
  - Namespaces, W3C Recommendation Jan '99

Expressing information

- Key for all data processing
- Different approaches for different data models

- Now looking at semistructured data
- Information about a book, natural text:

  "Politics of experience by Ronald Laing, published in 1967"

Information about a book in HTML


```
<td><h1 class=”Books">Politics of experience by Ronald Laing, published in
    1967</h1></td><td align="right" nowrap> Item
    number:320070381076</td><td align="right" valign="top"><img
    src="http://pics.booksstatic.com/aw/pics/globalAssets/rtCurve.gif"
    width="8" height="8"></td></tr><tr><td colspan="6" valign="middle"
    bgcolor="#5F66EE"><img src="http://pics.booksstatic.com/aw/pics/s.gif"
    width="1" height="4"></td></tr></table><table width="100%"
    border="0" cellpadding="0" cellspacing="0"><tr><td
    bgcolor="#CCCCFF"><img src="http://pics.booksstatic.com/aw/pics/s.gif"
    width="1" height="1"></td><td bgcolor="#EEEEFF"><div
    id="FastVIPBIBO"><table border="0" cellpadding="0" cellspacing="0"
    width="100%">
```

Same Data as XML

```
<book year="1967">
   <title>The politics of experience
   </title>
    <author>
          <firstname>Ronald</firstname>
          <lastname>Laing</lastname>
    </author>
</book>
```

- Information is (1) decoupled from presentation, then (2) chopped into smaller pieces, and then (3) marked with semantic meaning
- It can be processed by machines
- Like HTML, only syntax, not logical abstract data model

XML Components

- Elements
- Attributes
- Text
- Comments
- Processing Instructions
- Namespace Declarations
- Document = Prolog + Root Element (+ Text)
  + Comments + Processing Instructions
- All inherited from SGML, then HTML

Elements

- Enclosed in Tags
  - Begin Tag: e.g., <bibliography>
  - End Tag: e.g., </bibliography>
  - Element without content: e.g., <bibliography />
- Elements can be nested
  <bib> <book> Wilde Wutz </book> </bib>
- Subelements can implement multisets
  <bib> <book> ... </book> <book> ... </book> </bib>
- Documents must be well-formed
  <a> is forbidden!
  <a>   is forbidden!

Attributes

- Attributes are associated to Elements

```
<book  price = "55" currency = "USD" >
   <title> ... </title>
   <author> ... </author>
</book>
```

- Element that only has attributes

```
<person name = "Wutz" age = "33"/>
```

- What is the difference between a nested element and an attribute? Are attributes useful?
- Attribute names must be unique! (No Multisets)

  `<person name = "Wilde" name = "Wutz"/>` is illegal!

Text and Mixed Content

- Text appears in element content
  - <title>The politics of experience</title>
- Can be mixed with other subelements
  - <title>The politics of experience</title>
- Mixed Content
  - For “document – oriented“ XML – very useful
  - The need does not arise in „data“ processing, only entities and relationships
  - People speak in sentences, not E/R. XML allows to preserve the structure of natural language, while adding semantic markup that can be interpreted by machines.

Spectrum: Natural language, semi-structured data, and structured data

1. Dana said that the book entitled „The politics of experience“ is really excellent !
2. <citation author=„Dana“> The book entitled „The politics of experience“ is really excellent ! </citation>
3. <citation author=„Dana“> The book entitled <title> The politics of experience</title> is really excellent ! </citation>
4. <citation>
   <author>Dana</author>
   <aboutTitle>The politics of experience</aboutTitle>
   <rating> excellent</rating>
   </citation>

Comments, PIs, Prolog

- Comment: Syntax as in HTML
  <!-- This is just a comment -->
- Processing Instructions
  - Contain no data - interpretation by processor
  - Syntax: <?pause 10 secs ?>
  - Pause is "Target"; 10secs is "Content"
  - xml is a reserved target for prolog (all case variants)
- Prolog
  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
  <?xml-stylesheet href="demo.css" type="text/css"?>
  - Standalone defines whether there is a DTD
  - Encoding is usually Unicode.

Whitespace

- Whitespace = Text of Space, Tabs and Returns
- Special Attribute xml:space to control use
- Human-readible XML (with Whitespace)

```
<book xml:space="preserve" >
   <title>Die wilde Wutz</title>
   <author>D.A.K.</author>
</book>
```

- (Efficient) machine-readible XML (no WS)

```
<book xml:space="default" ><title>Die wilde
Wutz</title><author>D.A.K.</author></book>
```

- Performance improvement: ca. Factor 2.

CDATA sections

- Sometimes we would like to preserve the original characters, and not interpret them as markup
- CDATA sections
  - Not parsed as XML

```
<message>
   <greeting>Hello,world!</greeting>
</message>
<message> <![CDATA[<greeting>Hello, world!</greeting>]]>
</message>
```

Language declaration

Express that certain elements are in a formal/natural language

```
<p xml:lang="en">The quick brown fox jumps over the lazy dog.</p>
<p xml:lang="en-GB">What colour is it?</p>
<p xml:lang="en-US">What color is it?</p>
```

Namespaces

- Integration of Data from diverse data sources
- Integration of different XML Vocabularies (aka Namespaces)
- Each „vocabulary“ has a unique key, identified by a URI/IRI
- Same local name, from different vocabularies can have
  - Different meaning
  - Different structure associated with it

QNames

- Qualified Names (QName) to attach a „name“ to its „vocabulary“
  - for all nodes in an XML document that has names (Attributes, Elements, Pis
- QName ::= triple (URI [prefix:] localname)
  - Binding (prefix, URI) is introduced in elements start tag
  - Later only the prefix is used, not the long URIs
  - Prefix is optional, default namespaces
  - Prefix and localname a separated by „:“

Namespace Definitions

- Namespace definitions look like Attributes
  - Identified by „xmlns:prefix“ or „xmlns“ (default namespace)
  - Bind the Prefix to the URI
- Scope is the entire element in which it is declared
  - Includes the element itself, its attributes and its subtrees
  - Scopes can be nested
  - Redeclaration possible (same prefix to different URI, different prefixes to same URI)
  - „Undeclare“ only with newer versions of XML+Namespace
- Example

```
<ns:a xmlns:ns=„someURI“ ns:b=„foo“>
    <ns:b>content</ns:b>
</ns:a>
```

Example: Namespaces

- DQ1 defines dish for tableware
  - Diameter, Volume, Decor, ...
- DQ2 defines dish for satellite
  - Diameter, Frequency
- How many "dishes" are there?
- Better ask for:
  - "How many dishes are there?"

    or
  - "How many dishes are there?"

Example: Namespaces

```
<gs:dish  xmlns:gs = "http://tableware.com" >
   <gs:dm gs:unit = "cm">20</gs:dm>
   <gs:vol gs:unit = "l">5</gs:vol>
   <gs:decor>Gelsenkirchener Barock</gs:decor>
</gs:dish>
<sat:dish xmlns:sat = "http://television.com" >
   <sat:dm>200</sat:dm>
   <sat:freq>20-2000MHz</sat:freq>
</sat:dish>
```

Mixing Several Namespaces

```
<gs:dish  xmlns:gs = "http://tableware.com"
                xmlns:uom = "http://units.com">
  <gs:dm uom:unit = "cm">20</gs:dm>
  <gs:decor>Gelsenkirchener Barock</gs:decor>
  <gs:vol  uom:unit = "l">5</gs:vol>
  <comment>I am unqualified</comment>
</gs:dish>
```

Default Namespaces

tableware.com if not specified otherwise.

```
<dish  xmlns = "http:// tableware.com"
                xmlns:uom = "http://units.com">
  <dm uom:unit = "cm">20</dm>
  <decor>Gelsenkirchener Barock</decor>
  <vol  uom:unit = "l">5</vol>
  <comment>I am qualified</comment>
</dish>
```