Advanced databases and data models

Theme 1: Semi structured data, XML and RDF

Lecture 1: Modeling

Lena Strömbäck
What is the problem?

- The user’s effort is not enough for the task
- The data describes complex real world objects
- The data is not easily human interpretable
- There is a need for integration and comparison of data
In this course:

• What are the particular requirements for storing data on the web?
• Why are traditional databases not enough?
• Explore technologies for data management on the web.
• Four themes
  – Semi structured data, XML and RDF
  – Data base management systems
  – Semantic web: Ontologies and OWL
  – Data integration for the web
Personell and Course Information:

Available at: www.ida.liu.se/~TDDD43
Today´s lecture

Introduction to semi-structured data
Technologies
XML/RDF
Defining the data model
Data model vs. Data guides
Technologies
DTD/XML Schema/RDF Schema
Data modeling in XML
Other DB models
Semi-structured data

Data is not just text, but is not as well-structured as data in databases

Occurs often in web databanks

Occurs often in integration of databanks
Semi-structured data - properties

irregular structure

implicit structure

partial structure
Semi-structured data - model

network of nodes

object model (oid)
OEM (Object Exchange Model)

Graph
Nodes: objects
  oid
  atomic or complex
    - atoms: integer, string, gif, html, …
    - value of a complex object is a set of
      object references (label, oid)

Edges have labels
OEM is used by a number of systems (ex. Lorel)
Restaurant Guide

17 gourmet
13 Chef Chu
14 street
66 Vietnamese
23 Saigon
18 city
25 Mountain View
44 El Camino Real
15 Palo Alto
16 92310
44 15 16
18 23 25
66 55 79
54 92310
19
35
12 Guide
19 restaurant
35 restaurant
cafe
17
gourmet

nearby

El Camino Real restaurant near Palo Alto
nearby

nearby

Gourmet Chef Chu near El Camino Real in Palo Alto.
Exercise 1

Represent the relations below using the OEM data model.
Technologies: XML and RDF

Why not relational databases?

Technologies:
XML
RDF

Definition of datamodel:
DTD
XMLschema
RDFSschema

Semantic models: Ontoligies and OWL later in the course.
Example

Muscle cell
**Relational representation**

<table>
<thead>
<tr>
<th>Compartment</th>
<th>Reaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>Name</td>
</tr>
<tr>
<td>Blood</td>
<td>Inblood</td>
</tr>
<tr>
<td>Cell</td>
<td>Musclecell</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Species</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
</tr>
<tr>
<td>Sug1</td>
</tr>
<tr>
<td>Ins</td>
</tr>
<tr>
<td>Sug2</td>
</tr>
<tr>
<td>En</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reactant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reaction</td>
</tr>
<tr>
<td>ToCell</td>
</tr>
<tr>
<td>ToCell</td>
</tr>
<tr>
<td>Move</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reaction</td>
</tr>
<tr>
<td>ToCell</td>
</tr>
<tr>
<td>Move</td>
</tr>
</tbody>
</table>
Relational model - drawbacks

• Far from semi-structured proposal
  • Not suitable for describing tree structure
  • Too general or many tables
• Static – all attributes typed
• All data entries atomic – in principle
XML representation

- **Ordered tree**
  Similar to semi-structured proposal

- **Element vs. Attribute**

- **Extensible**
  New kinds of data can be integrated

- **Flexible**
  Easy to mix different kinds of data

```xml
<?xml version="1.0" encoding="UTF-8"?>
<minimodel name="sugartransport"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="minimodel.xsd">
  <listOfCompartments>
    <compartment id="blood" name="inblood"/>
    <compartment id="cell" name="musclecell"/>
  </listOfCompartments>
  <listOfSpecies>
    <species id="sug1" name="sugar" compartment="blood"/>
    <species id="ins" name="insulin" compartment="blood"/>
    <species id="sug2" name="sugar" compartment="cell"/>
    <species id="en" name="energy" compartment="cell"/>
  </listOfSpecies>
  <listOfReactions>
    <reaction id="tocell" name="sugartocell">
      <listOfReactants>
        <speciesReference species="sug1"/>
        <speciesReference species="ins"/>
      </listOfReactants>
      <listOfProducts>
        <speciesReference species="sug2"/>
      </listOfProducts>
    </reaction>
    <reaction id="move" name="makemovement">
      <listOfReactants>
        <speciesReference species="sug2"/>
      </listOfReactants>
      <listOfProducts>
        <speciesReference species="en"/>
      </listOfProducts>
    </reaction>
  </listOfReactions>
</minimodel>
```
RDF: Resource Description Framework

Framework for describing resources on the web
Designed to be read and understood by computers
Not designed for being displayed to people

Written in XML
RDF is a W3C Recommendation
<?xml version="1.0" encoding="UTF-8"?>

<species metaid="_506372" id="E1" name="MAPKKK activator" compartment="compartment"
initialConcentration="3e-05">
  <annotation>
    <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:bqbiol="http://biomodels.net/biology-qualifiers/"
    xmlns:bqmodel="http://biomodels.net/model-qualifiers/">
      <rdf:Description rdf:about="#_506372">
        <bqbiol:isVersionOf>
          <rdf:Bag>
            <rdf:li rdf:resource="http://www.ebi.ac.uk/interpro/#IPR003577"/>
          </rdf:Bag>
        </bqbiol:isVersionOf>
      </rdf:Description>
    </rdf:RDF>
  </annotation>
</species>
RDF Data model: Triples

A **Resource** is anything that can have a URI, such as our molecule "_506372 "

A **Property** is a Resource that has a name, such as “isVersionof"

A **Property value** is the value of a Property, such as " IPR003577 "

(note that a property value can be another resource)

Suitable for semi-structured data.
Part of our example model as RDF triples

1 blood #name "in blood"
3 sug1 #name "sugar in blood"
4 sug1 #compartment blood
11 st #name "sugartransport"
12 genid:A71987 #type Bag
13 st #reactants genid:A71987
14 genid:A71987 1 sug1
15 genid:A71987 2 ins
16 genid:A71988 #type #Bag
17 st #products genid:A7 1988
18 genid:A71988 1 sug2
Semi-structured data - properties

Data model/guide changes commonly

Object can change type/class

The distinction between data and schema is blurred
Semi-structured data – data models vs data guides

a posteriori ’data guide’ versus a priori schema

Data model/data guide could be supportive or a hinder while querying

Definition of data model for XML – DTD or XML Schema

Data model for RDF – RDF schema
Data Guides

A structural summary over a databank that is used as a dynamic schema

Is used in query formulation and optimization

Is often created a posteriori

Properties:
  concise
  accurate
  convenient
Defining the XML model: DTD

A Document Type Definition (DTD) defines the legal building blocks of an XML document.

It defines the document structure with a list of legal elements and attributes.

In the DTD all XML documents are one of:

- Elements
- Attributes
- Entities
- PCDATA
- CDATA
Defining the XML model: XML Schema

The XML Schema defines the legal building blocks of an XML document.

An XML Schema:
- defines elements
- defines attributes
- defines which elements are child elements
- defines the order of child elements
- defines the number of child elements
- defines data types for elements and attributes
- defines default and fixed values for elements and attributes
XML Schema vs. DTD

XML Schemas are extensible to future additions
extend element definitions

XML Schemas are richer and more powerful than DTDs

XML Schemas are written in XML

XML Schemas support data types

XML Schemas support namespaces
RDF Schema – define relations between objects

<?xml version="1.0" encoding="UTF-8"?>

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

    <rdf:Description rdf:ID="species">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    </rdf:Description>

    <rdf:Description rdf:ID="protein">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
        <rdfs:subClassOf rdf:resource="#species"/>
    </rdf:Description>

</rdf:RDF>
Data modelling with XML

- Elements vs. Attributes
- Keys
- Many to many relations
Lab exercises:

Construct a data model in relational model and XML.

Answer questions, compare and write report.

Tools: Oxygen XML and MS Server
XML in relational databases

create table person(
  id integer primary key,
  namn varchar(15),
  description XML);
Neo

Neo4j is a **graph database**. It is an embedded, disk-based, fully transactional Java persistence engine that stores data structured in graphs rather than in tables.

Linköping related company.

Interesting for semi-structured data.
The Neo Persistence Engine

Primitives:
- nodes
- relationships
- properties

Features
- ACID transaction
- Durable persistence
- Transaction recovery

Implementation
- Java
Representing XML in Neo: Basic solution

```xml
<rock>
  ...
  <group name="Pearl Jam">
    <member>
      <firstName>Eddie</firstName>
      <lastName>Vedder</lastName>
      <instrument type="vocal"/>
    </member>
    ...
  <album year="1991">Ten</album>
  <album year="1994">Vitalogy</album>
  ...
  </group>
  ...
  <singer name="Bob Dylan" songwriter="yes">
    <album year="1965">Highway 65 Revisited</album>
    <album year="1966">Blonde on Blonde</album>
    ...
  </singer>
  ...
</rock>
```
Representing XML in Neo: Customizations