

# XML

Maciej Janik

Semantic Web  
2009-07-01



- eXtensible Markup Language
- Derived from structured text ( $\text{HTML4.0} \in \text{XML} \subset \text{SGML}$ )
- Web-Standard (W3C) for exchanging data:
  - ◆ XML describes inputs and outputs of many applications (in most cases called: services)
  - ◆ Industry created and supported xml standards for applications, communication protocols, service descriptions, etc. (e.g. [www.oasis-open.org](http://www.oasis-open.org) or [www.xml.org](http://www.xml.org) )
- Complementary to HTML:
  - ◆ HTML is only one of the applications for XML
  - ◆ HTML describes presentation layer
  - ◆ XML describes content structure
- **Extensible**, unlike HTML
  - ◆ Users can add new tags, and separately specify how the tag should be handled for display
- Data modeling: XML is a data model for semi-structure data

- XML innovations
  - ◆ Specify new tags
  - ◆ Create nested tag structures – hierarchical approach
  - ◆ Enable to exchange (annotated) **data**, not only documents
  - ◆ Tags create content independent of visualization (vs. HTML)
- Tags make data (relatively) self-documenting

```
<bank>
  <account>
    <account_number> A-101 </account_number>
    <branch_name> Downtown </branch_name>
    <balance> 500 </balance>
  </account>
  <depositor>
    <account_number> A-101 </account_number>
    <customer_name> Johnson </customer_name>
  </depositor>
</bank>
```

- Data interchange is critical in today's networked world
  - ◆ Examples:
    - Banking: funds transfer
    - Order processing (especially inter-company orders)
    - Scientific data
      - Chemistry: ChemML, ...
      - Genetics: BSML (Bio-Sequence Markup Language), ...
  - ◆ Paper flow of information between organizations is being replaced by electronic flow of information
- Each application area has its own set of standards for representing information
- XML has become the basis for all new generation data interchange formats

- Earlier generation formats were based on plain text with line headers indicating the meaning of fields
  - ◆ Similar in concept to email headers
  - ◆ Does not allow for nested structures, no standard “type” language
  - ◆ Tied too closely to low level document structure (lines, spaces, etc)
- Each XML based standard defines what are valid elements, using
  - ◆ XML type specification languages to specify the syntax
    - DTD (Document Type Descriptors)
    - XML Schema
  - ◆ Plus textual descriptions of the semantics
- XML allows new tags to be defined as required
  - ◆ However, this may be constrained by DTDs
- A wide variety of tools is available for parsing, browsing and querying XML documents/data

- Nesting of data
  - ◆ Useful in data transfer
  - ◆ Create hierarchical data structures
  - ◆ Represent subelements of a larger entity
    - E.g.: elements representing *title* and *professor* are nested within an *Lecture* element

```
<Event id="o1">
  <Lecture LNr="5001">
    <title>Introduction to CS</title>
    <prof>
      <pnr>2137</pnr>
      <name>Kant</name>
      <rank>C4</rank>...
    </prof>
  </Lecture>
  ...
</Event>
```

- Nesting is not supported, or discouraged, in relational databases
  - ◆ With multiple orders, customer name and address are stored redundantly
  - ◆ normalization replaces nested structures in each order by foreign key into table storing customer name and address information
- Nesting is supported in object-relational databases
- But nesting is appropriate when transferring data
  - ◆ External application does not have direct access to data referenced by a foreign key

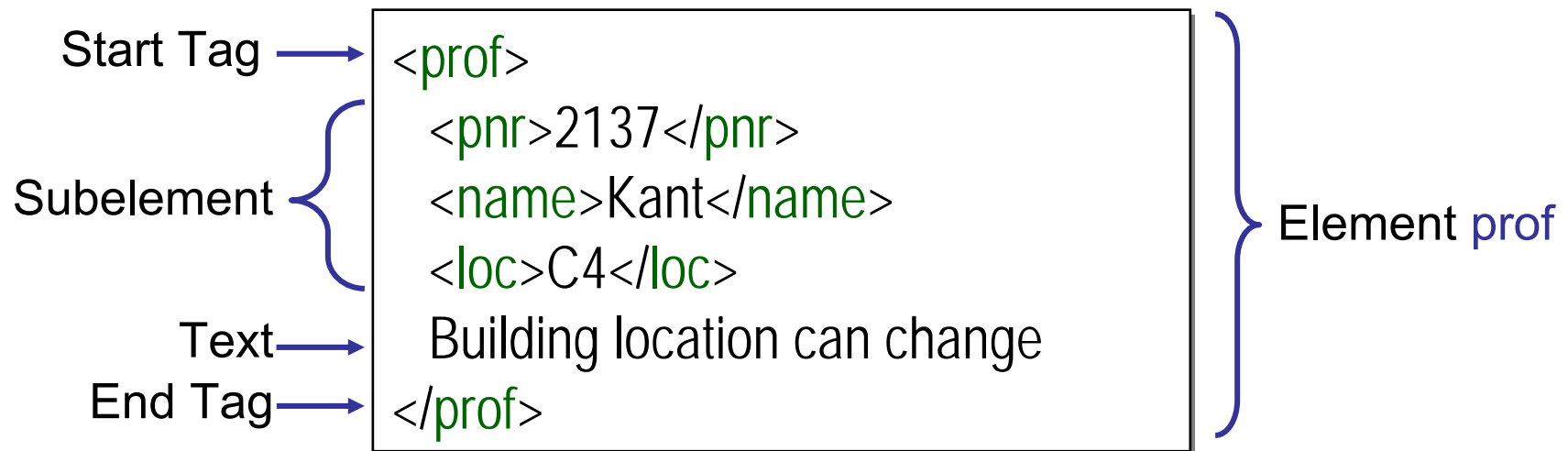
- **HTML**: fixed **Tags und Semantics** (presentation layer)
- **XML**: variable **Tag Set** specific for given application or standard (meta-grammar)
- $XML \subset SGML$  (Standard Generalized Markup Language)

```
<h1> Event </h1>
  <p>
    <i> Introduction to CS </i>
    Kant
    <br > Tuesday, 16.00
  <p>
    ...
```

**HTML**

```
<Event id="o1">
  <Lecture LNr="5001">
    <title> Introduction to CS</title>
    <prof>
      <pnr>2137</pnr>
      <name>Kant</name>
      <rank>C4</rank>...
    </prof>
  </Lecture>
  ...
</Event>
```

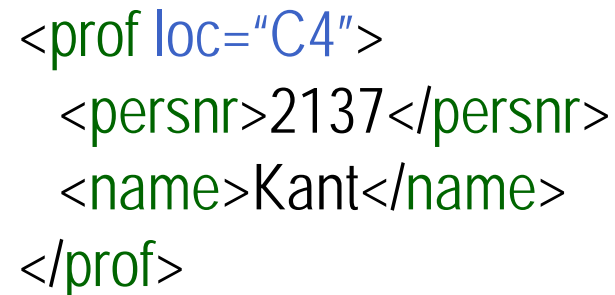
- XML element
  - Object is defined by a pair of corresponding tags, like <prof> (opening tag) and </prof> (closing tag)
  - Content of the element: text and other elements (subelements) included between tags
  - Elements can be nested (no depth restrictions)
  - Empty elements: <year></year> can be shortened: <year/>



- Elements must be properly nested
  - ◆ *Proper nesting*
    - `<account> ... <balance> .... </balance> </account>`
  - ◆ *Improper nesting*
    - `<account> ... <balance> .... </account> </balance>`
  - ◆ Every start tag must have a matching end tag on the same level (same parent element).
  
- Improper nesting in HTML may not be harmful ...
  - In <i>HTML <b>improper</i> nesting</b> may work
  - Still produces output
  - In *HTML improper nesting* may work

- XML Attribute:
  - ◆ Name-value pair inside starting tag of element
  - ◆ Tied to a specific xml element
  - ◆ Alternative notation to nested tags
  - ◆ Element can have multiple attributes, but each occurs only once

Attribute *loc*



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

Another notation of the same data:

```
<prof pnr="2137" name="Kant" loc="C4"/>
```

- Document view
  - ◆ subelement contents are part of document contents
  - ◆ attributes are part of markup
- Data representation view
  - ◆ ... unclear and confusing ...
  - ◆ Same information can be represented in two ways
    - `<prof name="Kant"> ... </prof>`
    - `<prof>`  
    `<name>Kant</name>`  
    `</prof>`
- Tip: use subelements for content (objects ...) and attributes as identifiers of elements

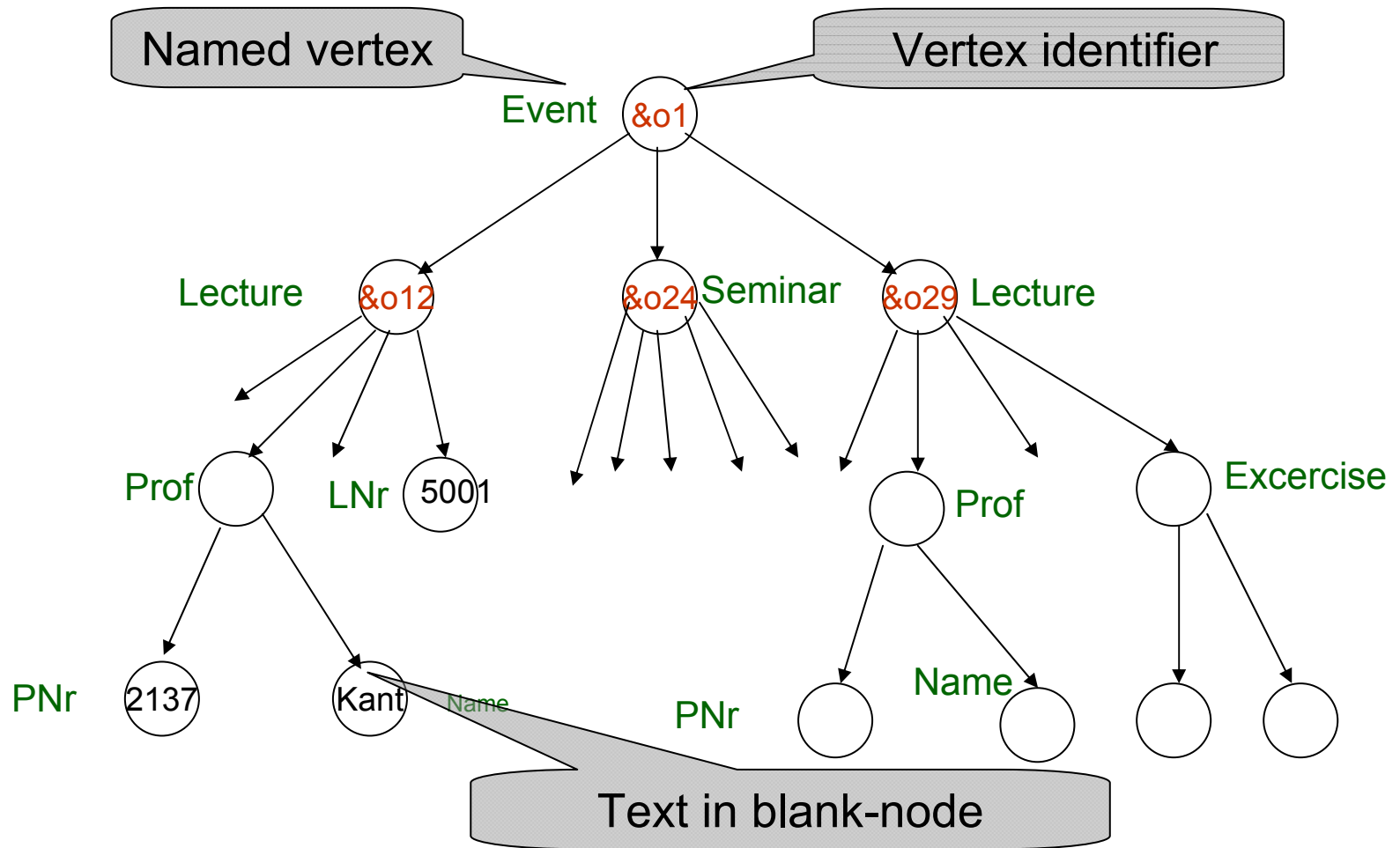
- XML can be exchanged between organizations
- Problem
  - ◆ Same tag + Different organizations → Different meaning
- Specifying a unique string as an element name avoids confusion
- Better solution: use **unique-name:element-name**
- Avoid long unique names by using XML Namespaces

```
<Lectures xmlns:uni='http://www.university.edu/'>
```

```
  ...  
  <uni:lecture>  
    <uni:title> Introduction to CS </uni:title>  
    <uni:location> F112 </uni:location>  
  </uni:lecture>
```

```
  ...  
</Lectures >
```

## XML can be represented as directed graph

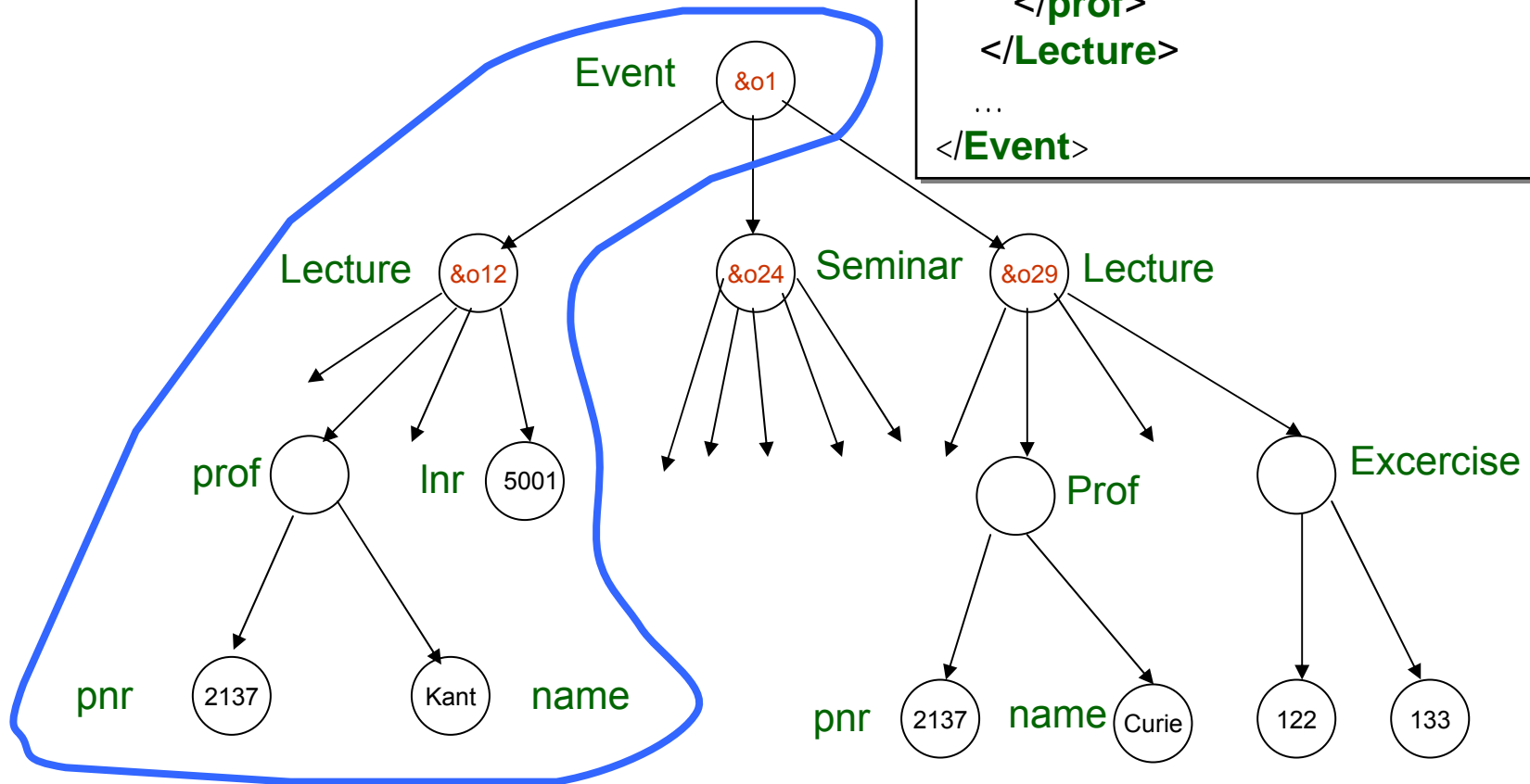


# XML Model

```

<Event id="o1">
  <Lecture>
    <title>Introduction to CS</title>
    <Inr>5001</Inr>
    <prof>
      <pnr>2137</pnr>
      <name>Kant</name>
      <loc>C4</loc>...
    </prof>
  </Lecture>
  ...
</Event>

```



- **XML Document:**
  - ◆ Text Document with XML descriptions
  - ◆ Database: document is a semi-structured database
    - Includes specific schema
- **Well-formed XML document**
  - ◆ All Elements are correctly nested with matching Start and End Tags
  - ◆ Document has one root element
  - ◆ It still can contain unstructured text
  - ◆ Specific characters in XML have to be represented in special way
- **Valid XML Document:**
  - ◆ Well-formed XML Document, that corresponds to a specific defined Schema
  - ◆ Schema is used to validate document
  - ◆ Appropriate for data used in Web Portal

- Schemas are very important for XML data exchange
  - ◆ Otherwise, a site cannot automatically interpret data received from another site
  
- Two mechanisms for specifying XML schema
  - ◆ **Document Type Definition (DTD)**
    - Widely used
  - ◆ **XML Schema**
    - Newer, more powerful but more complicated

- DTD specifies type and structure of XML document
- DTD constraints structure of XML data
  - ◆ What elements can occur
  - ◆ What attributes can/must an element have
  - ◆ What subelements can/must occur inside each element, and how many times.
- DTD does not constrain data types
  - ◆ All values represented as strings in XML
- DTD syntax
  - ◆ `<!ELEMENT element (subelements-specification) >`
  - ◆ `<!ATTLIST element (attributes) >`

- Subelements are specified as
  - ◆ names of elements, or
  - ◆ #PCDATA (parsed character data), i.e., character strings
  - ◆ EMPTY (no subelements) or ANY (anything can be a subelement)
- Subelement specification may have regular expressions
  - Notation:
    - “|” - alternatives
    - “+” - 1 or more occurrences
    - “\*” - 0 or more occurrences

- Example

```
<!DOCTYPE bank [  
  <!ELEMENT bank ( ( account | customer | depositor)+)>  
  <!ELEMENT account (account_number branch_name balance)>  
  <!ELEMENT balance(#PCDATA)>  
  <!ELEMENT customer_name(#PCDATA)>  
  ...  
>
```

- XML Schema
  - ◆ Much more expressive than DTD,
  - ◆ Significantly more complicated than DTD.
- XML Schema supports
  - ◆ Typing of values
    - Integer, string, etc
    - Constraints on min/max values
  - ◆ Complex types (user-defined),
  - ◆ Many more features, including
    - uniqueness and foreign key constraints, inheritance
- XML Schema is
  - ◆ Specified in XML syntax,
  - ◆ More-standard representation (but verbose),
  - ◆ Already integrated with namespaces.

# Example of XML Schema

```
<xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema>
```

```
<xs:element name="bank" type="BankType"/>
```

```
<xs:element name="account">
```

```
  <xs:complexType>
```

```
    <xs:sequence>
```

```
      <xs:element name="account_number" type="xs:string"/>
```

```
      <xs:element name="branch_name" type="xs:string"/>
```

```
      <xs:element name="balance" type="xs:decimal"/>
```

```
    </xs:sequence>
```

```
  </xs:complexType>
```

```
</xs:element>
```

..... definitions of customer .....

```
<xs:complexType name="BankType">
```

```
  <xs:sequence>
```

```
    <xs:element ref="account" minOccurs="0" maxOccurs="unbounded"/>
```

```
    <xs:element ref="customer" minOccurs="0" maxOccurs="unbounded"/>
```

```
  </xs:sequence>
```

```
</xs:complexType>
```

```
</xs:schema>
```

- **XQuery**

- **FLOWR Expression**

- FOR** - Loop with variables
- LET** - Variable declaration and intermediate results
- ORDER** - Sorting
- WHERE** - Pattern matching
- RETURN** - Construction of results

```
<ISWebLecture>
  where
    <Event>
      <Lecture>
        <Prof><PNr>$P</PNr><name>$N</name></Prof>
        <LNr>$V</LNr>
        <title>$T</title>
      </Lecture>
    </Event> in "www.uni-koblenz.de/event_directory",
    <memberNr>$P</memberNr> in "isweb.uni-koblenz.de/team"
  construct
    <Lecture>
      <title>$T</title>
      <name>$N</name>
    </Lecture>
  </ISWebLecture>
```

Assuming Kant is a member of ISWeb:

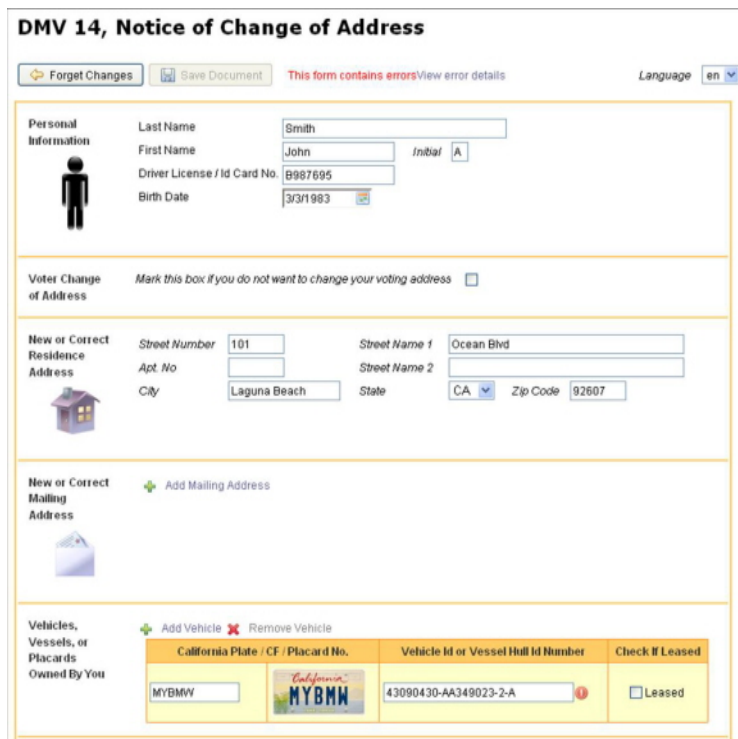
```
<ISWebLecture><Lecture>
  <title>Introduction to CS</title>
  <name>Kant</name>
</Lecture></ISWebLecture>
```

- All big database companies support XML
  - ◆ Oracle, DB1, MS SQL Server,...
  
- There are native XML databases available on the market, but without big success
  - ◆ Tamino, InfoNyte, ...
  
- Exchanging and defining data using XML became industrial standard
  - ◆ Web Services are completely based on XML
  - ◆ They have specific communication protocol in XML
  
- XML applications – examples on next slides

- XHTML is an XML'ized version of HTML
- Use of CSS to format XHTML documents
- XHTML + CSS = GUI (presentation to user)
- Example: web portal



- XForms - collect user input and automatically generate an XML document composed of it



**DMV 14, Notice of Change of Address**

Forget Changes Save Document This form contains errors View error details Language en

**Personal Information**

Last Name: Smith  
First Name: John Initial: A  
Driver License / Id Card No.: B987695  
Birth Date: 3/31/1983

**Voter Change of Address**

Mark this box if you do not want to change your voting address


**New or Correct Residence Address**

Street Number: 101 Street Name 1: Ocean Blvd  
Apt. No.: Street Name 2:  
City: Laguna Beach State: CA Zip Code: 92607

**New or Correct Mailing Address**

+ Add Mailing Address

**Vehicles, Vessels, or Placards Owned By You**

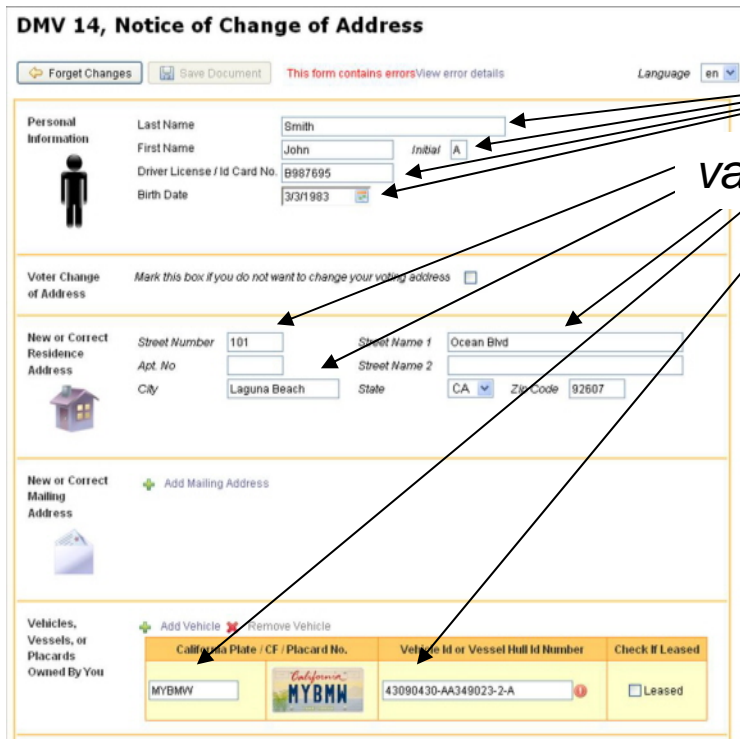
California Plate / CF / Placard No.	Vehicle Id or Vessel Hull Id Number	Check If Leased
MYBMW 	43090430-AA349023-2-A	<input type="checkbox"/> Leased

```
<DMV>
  <personal-information>
    <name>
      <first-name>John</first-name>
      <initial>A</initial>
      <last-name>Smith</last-name>
    </name>
    <driver-license-number>B987695</driver-license-number>
    <birth-date>1983-03-03</birth-date>
  </personal-information>
  ...
</DMV>
```

XML

Web Server

- XML Schema - validate input as it's entered by the user



**DMV 14, Notice of Change of Address**

Forget Changes Save Document This form contains errors View error details Language en

**Personal Information**

Last Name: Smith  
First Name: John Initial: A  
Driver License / Id Card No.: B987695  
Birth Date: 3/31/1983

**Voter Change of Address**

Mark this box if you do not want to change your voting address

**New or Correct Residence Address**

Street Number: 101 Street Name 1: Ocean Blvd  
Apt. No.: Street Name 2:  
City: Laguna Beach State: CA Zip Code: 92607

**New or Correct Mailing Address**

Add Mailing Address

**Vehicles, Vessels, or Placards Owned By You**

Add Vehicle Remove Vehicle

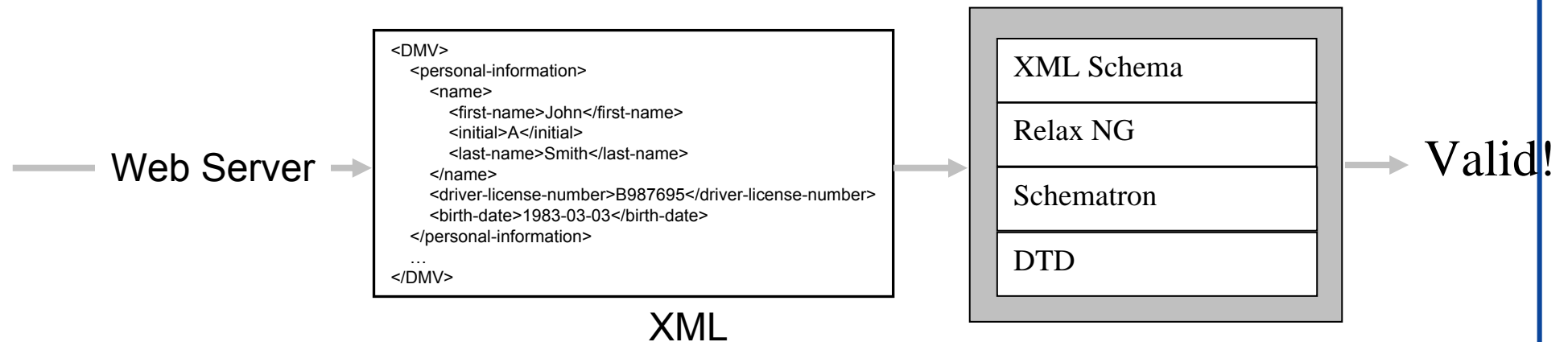
California Plate / CF / Placard No.	Vehicle Id or Vessel Hull Id Number	Check If Leased
MYBMW	43090430-AA349023-2-A	<input type="checkbox"/> Leased

validate

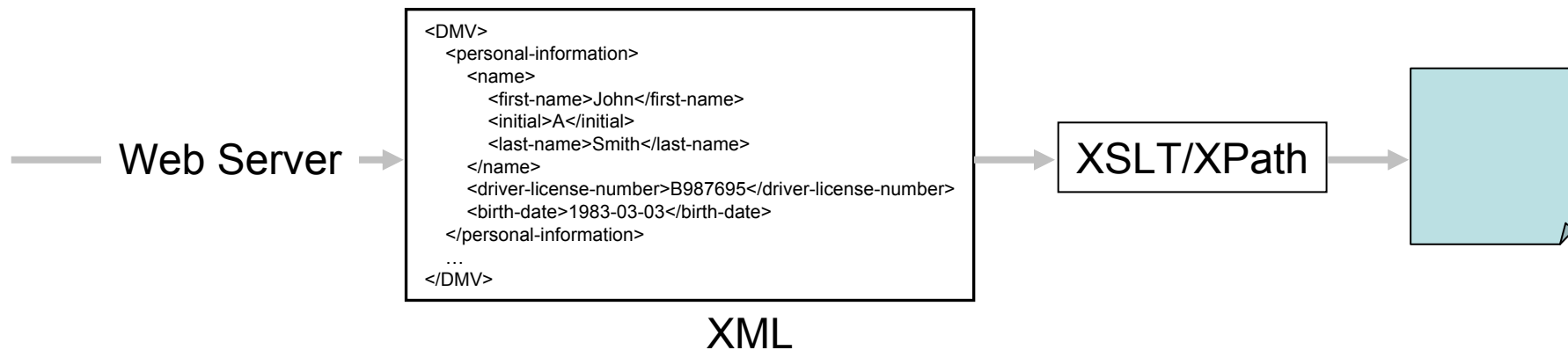
```
<xs:schema targetNamespace="http://orbeon.org/oxf/examples/dmv"
  xmlns:dmv="http://orbeon.org/oxf/examples/dmv"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xs:element name="DMV">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="personal-information">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="name">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="first-name" type="dmv:first-name"/>
                    <xs:element name="initial" type="dmv:initial"/>
                    <xs:element name="last-name" type="dmv:last-name"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="driver-license-number" type="dmv:driver-license-number"/>
              <xs:element name="birth-date" type="xs:date"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        ...
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

XML Schema

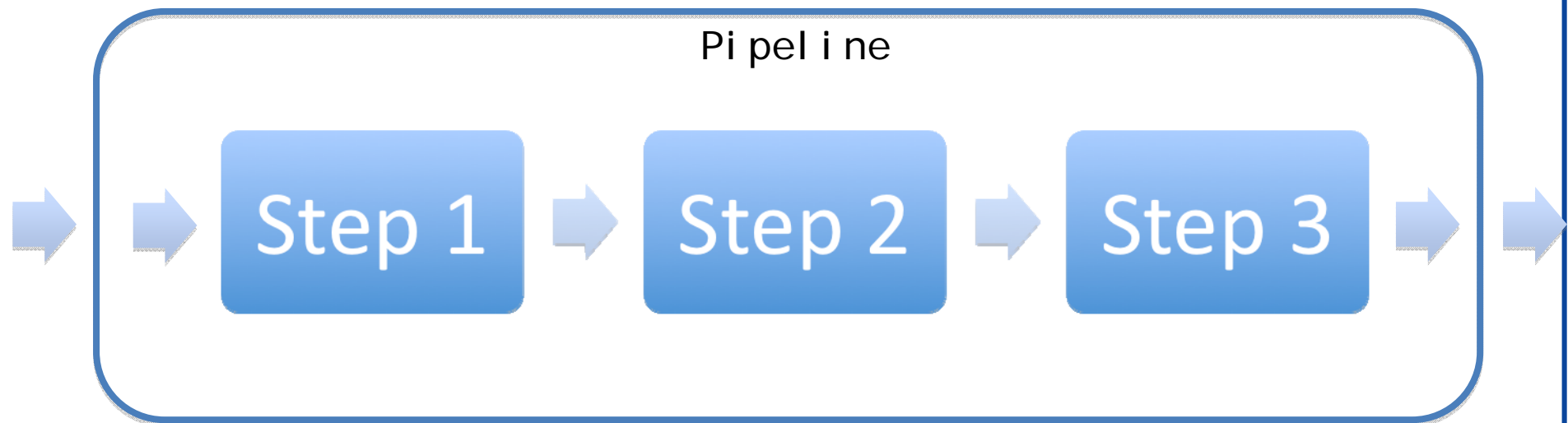
- XML Schema, Schematron, Relax NG, DTD, NVDL
  - ◆ validation languages on server-side



- Process, transform, apply functions, merge, sort ... the XML document to get desirable output (e.g. HTML)
  - ◆ XSL – eXtensible Stylesheet Language
  - ◆ XSLT – XSL Transformation



- XProc – the XML Pipeline language. Specify the series of actions (steps) to be applied to the XML document
  - ♦ e.g. validate then process then store in DB then query then ...



- Direct use of XML in the Simple Object Access Protocol (SOAP) standard:
  - ◆ Invocation of procedures across sites and applications with distinct databases
  - ◆ XML used to represent procedure input and output
  
- Web service
  - ◆ Site providing services as SOAP procedures
  - ◆ Service is described using WSDL (Web Services Description Language)
  - ◆ UDDI (Universal Description, Discovery, and Integration) – standard for defining directories of web services

## Relational and object model

### Pros:

- Clear consistency properties
- Partially:  
simple and clean formal model

### Cons:

- Only pre-defined data structures
- Designed for fully-defined data
- Not interchangeable
- Not easy to read

## XML

### Pros:

- Easy to read (relatively)
- Incomplete or not fully defined data is not a problem
- Serializable
- Easily interchangeable

### Cons:

- No simple and nice model
- Document-centered: not data or object centric model
  - ◆ Document can be serialized in multiple different ways

- What is the relation between „Professor“ and „Employee“
- Is „Professor“ node in „Lecture“ the same as „Professor“ in „Seminar“, and does „Professor“ has to be in the list or Professors?
- How can I refer to “Kant” on the Web?
- How do I link to external or internal data in/from the document?  
Internal and external references.
- Is possible to order information and what meaning it has?

⇒ despite all these ...

XML remains successful as document and data exchange format