Data - Metadata

- **Data**
  - **Johnny, Pasta, Lund, 2001-02-12, true, 677**
- **Metadata**
  - **name, name, city, start date, sent, weight**
- **Types of metadata**
  - **Structural**
  - **Semantic**
  - **Catalog (classification)**
  - **Integration (mapping)**

Structure

- **Modeling**
  - TechTarget: Data modeling is the analysis of data objects that are used in a business or other context and the identification of the relationships among these data objects.
- **Database solutions**
  - **Relational model**
    - » **Tables, columns, domains, keys, integrity constraints**
  - **Object-oriented, Object databases**
    - » **Classes, attributes, references, rules**
  - **XML**
    - » **Elements, attributes, rules**
  - **Other**
    - » **?**

Semantics

- The meaning of the data and metadata
  - Metadata
    - » name
    - » price
    - » weight
    - » sent
  - Semantics
    - » The thing that identifies each product type uniquely
    - » The number of SEK the customer pays including VAT for one piece
    - » Specifies the weight of the product including packaging in grams
    - » True if the order has been sent from our storage, otherwise false

Semi-structured data

- No structure (schemaless)
- Implicit structure (self-describing)
  - metadata built-in to the data
    - » no data → no metadata
- SSD

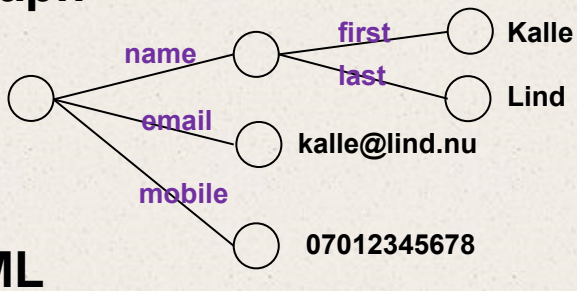
```
{name:{first:"Kalle", last:"Lind"},
 email:"kalle@lind.nu",
 mobile:"07012345678"}

{name:"Lisa",
 phone:"0709999999"}
```

Representations

- **SSD**

```
{name:{first:"Kalle", last:"Lind"},
 email:"kalle@lind.nu",
 mobile:"07012345678"}
```

- **Graph**



- **XML**

```
<Root>
        <name first="Kalle" last="Lind" />
        <email>kalle@lind.nu</email>
        <mobile>07012345678</mobile>
</Root>
```

- **JSON**

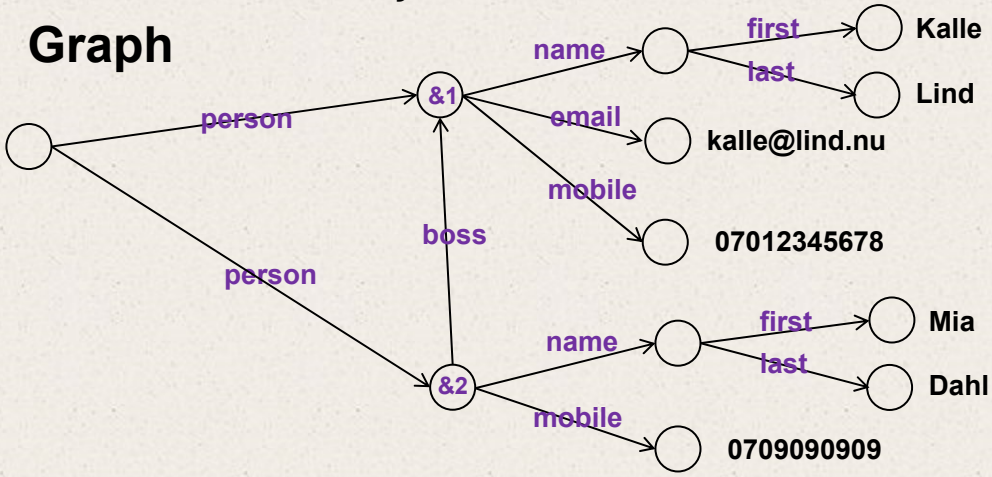
```
{"name" : {"first" : "Kalle",
           "last" : "Lind"},
 "email" : "kalle@lind.nu",
 "mobile" : "07012345678"}
```


Tree vs. Network

- **SSD**

```
{person: &1{name:{first:"Kalle", last:"Lind"},
            email:"kalle@lind.nu",
            mobile:"07012345678"},
 person: &2{name:{first:"Mia", last:"Dahl"},
            mobile:"0709090909",
            boss: &1}
```

- **Graph**



XML

- Stands for Extensible Markup Language
- A language for defining document structures
- XML provides a textual representation of data
- Is used within different areas:
  - Data storage
  - Web pages (XHTML)
  - Configuration files
  - Transport format (integrations, conversions)
- Rules can be specified through
  - DTD (Document Type Definition)
  - XML Schema
- Case sensitive

XML - Syntax

- Element

```
<Person>Kalle</Person>
```

- Attribute

```
<Person name="Kalle"></Person>
```

- Nested elements

```
<Person id="59">
    <Fname>Kalle</Fname>
    <Lname>Lind</Lname>
    <Address>
        <Street>Kungsgatan 53</Street>
        <PostalCode>12332</PostalCode>
        <City>Stockholm</City>
    </Address>
</Person>
```

- Empty element

```
<Person name="Kalle"></Person>
<Person name="Kalle" />
```

XML Document

- XML declaration

  <?xml version="1.1" encoding="UTF-8" ?>
- DOCTYPE – reference to rules

  <!DOCTYPE Person SYSTEM "Person.dtd">
- Namespaces
  - qualification of element and attribute names

    <sdxml:Person sdxml:name="Kalle"></sdxml:Person>
  - default and other namespaces

    <Root xmlns="default ns URI" xmlns:sdxml="sdxml ns URI">

    …

    </Root>

XML - References

- ID

  <Person name="Kalle" id="39"></Person>
- IDREF

  <Organization name="IBM" boss="39"></Organization>

DTD (Document Type Definition)

- Defines the XML structure (elements and attributes)

```
<!ELEMENT db (Person*)>
<!ELEMENT Person (Address)>
<!ELEMENT Address EMPTY>
<!ATTLIST Person
        name CDATA #REQUIRED
        id ID #REQUIRED
        birthdate CDATA #IMPLIED
        father IDREF #IMPLIED>
<!ATTLIST Address
        street CDATA #REQUIRED
        code CDATA #REQUIRED
        city CDATA #REQUIRED>
```


XML Schema

- Stronger than DTD
  - More flexible structures
  - data types
- XML syntax

```
<element name="db" type="dbType"/>
<complexType name="dbType">
    <sequence>
        <element name="Person" type="PersonType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
</complexType>
<complexType name="PersonType">
    <sequence>
        <element name="Address" type="AddressType" />
    </sequence>
    <attribute name="name" type="string" use="required"/>
    <attribute name="id" type="id" use="required"/>
    <attribute name="birthdate" type="date" use="optional"/>
    <attribute name="father" type="idref" use="optional"/>
</complexType>
<complexType name="AddressType">
…
```

Well-formed & Valid

- Well-formed XML
  - Syntactically correct
  - Starts with an XML declaration
  - Contains only one root element
  - Matching opening and closing tags
- Valid XML
  - Is well-formed
  - Follows the rules of the associated DTD or XML Schema

XML-based languages

- Definition of structure
- Definition of semantics

- XML basic rules
  - Alphabet, vocabulary
- XML Schema (or DTD)
  - Grammar, syntax
- XML Schema explanation (for humans)
  - Semantics, meaning

XML - Representations

- Textual representation (serialized XML document)
- Abstract node structure representation
  - XML Infoset
  - PSVI (Post-schema-validation Infoset)
  - XPath 1.0 model
  - XQuery 1.0 model
    - » XQuery 3.0 model
    - » XQuery 3.1 model

XML Infoset

- Representation of the significant parts of the content of an XML document
  - Some syntactical details are ignored
  - Does not care about XML Schema or data types
- 11 information items, among them
  - Document Information Item ("the root")
  - Element Information Item
  - Attribute Information Item
  - Comment Information Item
  - Processing Instruction Information Item
  - Document Type Declaration Information Item
  - Character Information Item
  - Namespace Information Item

PSVI

- Post-Schema-Validation Infoset
- Extends Infoset with support for XML Schema information
  - data types
  - validation status

XPath 1.0 model

- Tree representation of XML documents
- 7 node types
  - root
  - element
  - attribute
  - text
  - namespace
  - comment
  - processing instruction
- Every node has a value
  - The concatenation of all contained text nodes
- Node sets

1999

XQuery 1.0 model (XPath 2.0)

- Can represent
  - XML documents (tree structure)
  - nodes
  - values
  - sequences of nodes and/or values
- 7 types of nodes
  - document
  - element
  - attribute
  - text
  - comment
  - processing instruction
  - namespace

2007

http://www.w3.org/TR/xpath-datamodel/all

XPath/XQuery 3.0 model

- Extends the previous version with
  - functions

2014

https://www.w3.org/TR/xpath-datamodel-30/

XPath/XQuery 3.1 model

- Extends the previous version with
  - maps
  - arrays

2017

https://www.w3.org/TR/xpath-datamodel-31/

XQuery model - node properties

- Element node
  - children (element nodes, PI nodes, comment nodes, text nodes)
  - parent (element node or document node)
  - attributes (attribute nodes)
  - namespaces (namespace nodes)
  - string-value, typed-value
  - Namespaces and attributes are not children
- Attribute node
  - parent (element node) (called owner in Infoset)
  - string-value, typed-value
- Document node
  - children
  - string-value, typed-value

XQuery model - node properties

- Text node
  - string-value
  - typed-value
  - parent (element node)
- Comment node
  - string-value
  - parent (element node or document node)
- PI node
  - string-value
  - parent (element node or document node)
- Namespace node
  - string-value
  - parent (element node)

What to do next

- Quiz about XML (Quiz 1)