

## 3 XML Data Management and Bioinformatics applications

- 3.1 Introduction to XML
- 3.2 XML syntax
- 3.3 Document Type Definitions
- 3.4 Namespaces, schemas and more
- 3.5 Usage: Logical – physical Layout
- 3.6 XML in Bioinformatics: some examples
- 3.7 Querying XML documents: XPATH
- 3.8 XML Data Management: mapping documents to relations
- 3.9 Note on Information Integration using XML
- 3.10 Ontologies

using material from  
- Alan Robinson, (recommended! [http://industry.ebi.ac.uk/~alan/XML\\_Workshop](http://industry.ebi.ac.uk/~alan/XML_Workshop))  
- Silverschatz and M. Sapossnek .

## 3.6 Bioinformatic DTDs (from Robinson, 2000)

- Some DTD's have been proposed publicly as XML formats for biological data
  - GAME Drosophila Genome Project/Celera
  - BIOML ProteoMetrics
  - XML Schema EMBL sequence records
  - BSML VisualGenomics
  - CML OMF
  - DAS CSHL
  - BSA OMG-LSR
  - BLAST Various
  - INSDSeq Sequences

HS /Bio DBS05-XML2 2

## Small part of Game DTD

```
<!ELEMENT game ( seq+, map_position, annotation*,
computational_analysis* ) >
<!ATTLIST game version NMTOKEN #REQUIRED >

<!ELEMENT seq ( name, description?, residues?, dbxref*, organism?
) >
<!ATTLIST seq version CDATA #REQUIRED >

<!ATTLIST seq type (AA | DNA | RNA) #IMPLIED >
<!ATTLIST seq length NMTOKEN #REQUIRED >
<!ATTLIST seq md5checksum CDATA #IMPLIED >
<!-- ** Flag for discussion - unique id? Same as name? -->
<!ATTLIST seq id IDREF #REQUIRED >

<!ELEMENT name ( #PCDATA ) >
<!ELEMENT description ( #PCDATA ) >

<!ELEMENT residues ( #PCDATA ) >
```

See local copy of DTD

HS /Bio DBS05-XML2 3

## Bioinformatic Sequence Markup Language

- BSML
  - "...aims to provide a single document interface to integrate all project information, complete with protocols for network data retrieval."
  - "... is an extensible language specification and container for bioinformatic data."
  - Two kinds of information:
    - The Definitions section encodes the **bioinformatic data** (sequences, sets, sequence features, analytical outputs, relationships, annotations)
    - The **optional Display section** encodes information for graphic representation of the bioinformatic data."

Cf. A Robinson

HS /Bio DBS05-XML2 4

## So what?

- Current DTD's biased towards **sequence data**
  - gene expression DTD GML21 exists, proprietary
  - BLAST output DTD
- Probably more XML DTD's will be produced as people "roll their own": BAD
- Without co-ordination - likely to be all subtly different and incompatible in syntax and semantics
- Needed: common namespace, xml schema for different applications (sequence description, expression, ...)
- Next step: use XML DB instead of RDB ?!

HS /Bio DBS05-XML2 5

## 3.7 Why Query Languages for XML?

- Data Extraction and Filtering
  - Transformation with XSLT requires pattern sophisticated extraction language -> XPath
  - Extraction from large "documents"  
"Blast output which produced more than 3 hits"
- Construction of XML documents  
"Output an XML document of all hits..."
- Data conversion
  - Restructuring of documents , e.g. for data exchange
- Data integration
  - Combine different documents in a new one

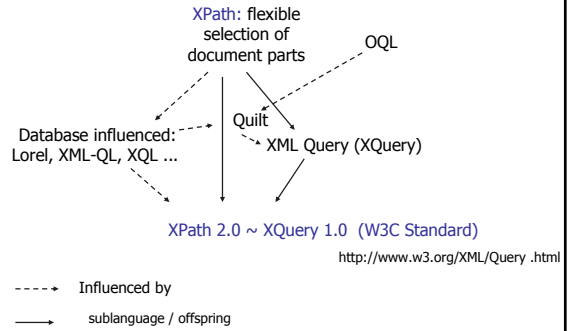
HS /Bio DBS05-XML2 6

## Why not SQL / OQL ?

- Main Argument: Different data models
  - 'Tree' not 'table' or 'Object'
- Optional parts in XML document  
"price < 100 if <price> element exists "
- Text retrieval
  - Not addressed in SQL / OQL
  - ... except user defined type "text"
- Construction: how to represent SQL query results as XML?
  - Only addressed in new SQL extension SQLX (or SQL XML)

HS /Bio DBS05-XML2 7

## Language development



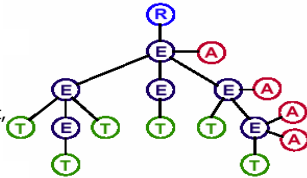
HS /Bio DBS05-XML2 8

## XPATH

- Language for navigating in XML-Documents
- W3C-recommendation 11/99:  
<http://www.w3.org/TR/xpath>
- Used for selecting parts of a document in a declarative way

Basic document model:

Tree with node types  
root, element, attribute,  
text, namespace, comment,  
processing instruction

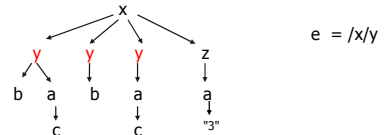


XPath expression: mapping from a node (the context node) to a set of nodes

HS /Bio DBS05-XML2 9

## XPath expressions

- Expression  $e$  specifies paths in an XML document
- Value of  $e$ : Set of nodes which match the path specified by the expression, boolean, number or string



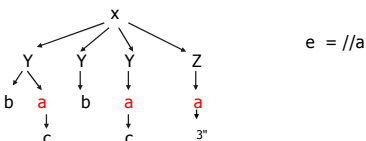
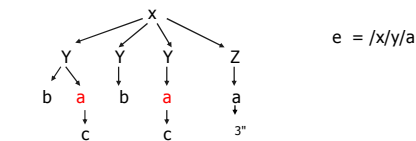
- Context defines where to start (above root)
- Application (e.g. XSLT) defines context by position in tree
- Absolute location path\* starts at root /, relative from context node

\*The only type of expression we discuss here

HS /Bio DBS05-XML2 10

## XPath expressions

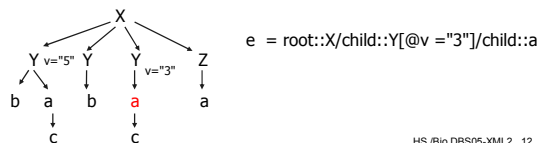
Examples



HS /Bio DBS05-XML2 11

## XPath: syntax (long)

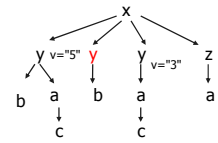
- Basic syntax of a location step:  
**axis::nodetest[predicate]**
- Axis: direction from context node (e.g. child, following, parent, ancestor, descendant..)
- Nodetest: basically the <element> to select,
- Predicate: predicate on elements and attributes which filters nodes on the path



HS /Bio DBS05-XML2 12

## XPath: abbreviated syntax

root:: / ix  
 Parent:: .. //c ../..!@v  
 Self:: . /a  
 Attribute:: @ attribute of self  
 // descendants of self, any depth //  
 \* all elements /\*/a  
 [n] n-th element /x/y[2] built-in functions:



e.g.  
 count() = number of nodes in context node set;  
 true(), false(), ....

HS /Bio DBS05-XML2 13

## Examples

Find BBB Elements which have name attribute

```

<AAA>
  <BBB id = "b1"/>
  <BBB id = "b2"/> //BBB[@name]
  <BBB name = "bbb"/>
</BBB/>
</AAA>
  
```

BBB elements without attributes

```

<AAA>
  <BBB id = "b1"/>
  <BBB id = "b2"/> //BBB[not (@*)]
  <BBB name = "bbb"/>
</BBB/>
</AAA>
  
```

Try: <http://www.zvon.org/xxl/XPathTutorial/General/examples.html>

HS /Bio DBS05-XML2 14

## Examples

BBB elements and EEE elements which are children of root AAA

```

<AAA>
  <BBB/> //AAA/EEE | //BBB
  <CCC/>
  <DDD>
    <CCC>
      <BBB/>
    </CCC>
  </DDD>
  <EEE/>
</AAA>
  
```

All /AAA/BBB descendants

```

/AAA/BBB/descendant::*
  
```

HS /Bio DBS05-XML2 15

## XPath examples

```

<addressBook>
  <address>
    <firstName>John</firstName>
    <surname>Smith</surname>
    <email>smithj@world.org</email>
    <tel type = "work">234-123-222</tel>
  </address>
  <address>
    <firstName>Alice</firstName>
    <surname>Brown</surname>
    <email>Alice.Brown@europe.com</email>
    <tel type = "home">22-33-444</tel>
    <tel type = "work">11-43-222</tel>
  </address>
  <address>
    <firstName>George</firstName>
    <surname>White</surname>
    <email>gw@rock.com</email>
  </address>
</addressBook>
  
```

**Queries**  
 q = /surname  
 result: -  
 q = // surname  
 result  
 q = // address  
 [firstName="Alice"]  
 /email  
 result

HS /Bio DBS05-XML2 16

## XPath examples

/address/tel[2]

```

<addressBook>
  <address ID=1 roomMate="3">
    <firstName >John</firstName>
    <surname>Smith</surname>
    <email>smithj@world.org</email>
    <tel type = "work">234-123-222</tel>
  </address>
  <address ID = 2 >
    <firstName>Alice</firstName>
    <surname>Brown</surname>
    <email>Alice.Brown@europe.com</email>
    <tel type = "home">22-33-444</tel>
    <tel type = "work">11-43-222</tel>
  </address>
  <address ID = 3 roomMate = "1">
    <firstName>George</firstName>
    <surname>White</surname>
    <email>gw@rock.com</email>
  </address>
</addressBook>
  
```

HS /Bio DBS05-XML2 17

## XPath examples

//address[tel="22-33-444"]/tel/@type

```

<addressBook>
  <address>
    <firstName>John</firstName>
    <surname>Smith</surname>
    <email>smithj@world.org</email>
    <tel type = "work">234-123-222</tel>
  </address>
  <address>
    <firstName>Alice</firstName>
    <surname>Brown</surname>
    <email>Alice.Brown@europe.com</email>
    <tel type = "home">22-33-444</tel>
    <tel type = "work">11-43-222</tel>
  </address>
  <address>
    <firstName>George</firstName>
    <surname>White</surname>
    <email>gw@rock.com</email>
  </address>
</addressBook>
  
```

HS /Bio DBS05-XML2 18

## XPath examples

Find type ("home" or "work") for tel# 22-33-444

```
//tel[self::tel="22-33-444"]/@type
```

```
<addressBook>
  <address>
    <firstName>John</firstName>
    <surName>Smith</surName>
    <email>smithj@world.org</email>
    <tel type = "work">234-123-222</tel>
  </address>
  <address>
    <firstName>Alice</firstName>
    <surName>Brown</surName>
    <email>Alice.Brown@europa.com</email>
    <tel type = "home">22-33-444</tel>
    <tel type = "work">11-43-222</tel>
  </address>
  <address>
    ....
  </address>
</addressBook>
```

HS /Bio DBS05-XML2 19

## XQuery = XPath 2.0

Basic structure of expressions

- **Pattern clause**
  - Matching of substructures (-> XPath)
  - Binding of variables (Syntax: \$x)
- **Filter clause**
  - Predicate on variables , comparison with constants etc
- **Construction clause**
  - Construct result as an XML document

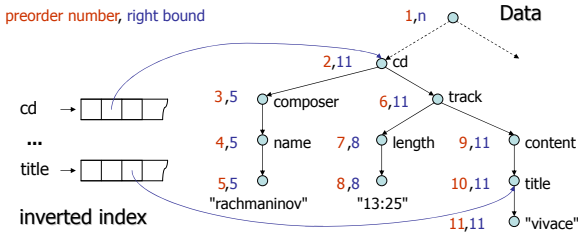
```
for $a in document("addressbook.xml") //address
let $t := $a/tel
where count($t) > 1
return <manyPhones>
  { $a/firstName, $a/surName, <num> count($t) </num> }
</manyPhones>
```

FLWR ("Flower")-expression

HS /Bio DBS05-XML2 20

## Implementation issues

Encoding trees goal: efficient tree manipulation e.g.: "Is x ancestor of y?"



$$\text{ancestor}(x, y) = \text{pre}(x) < \text{pre}(y) \wedge \text{rb}(x) \geq \text{pre}(y)$$

$$\text{ancestor}(\text{cd}, \text{title}) = 2 < 10 \wedge 11 \geq 10$$

Tools for managing XML documents ⇔ DBS 21

HS /Bio DBS05-XML2 21

## 3.8 XML Data Management

- Approaches
  - Use Relational DBMS and map documents to relations
    - Generic mapping
    - DTD oriented mapping
    - Use SQL for queries, transform result set into XML document
  - Use XML (native or enhanced ) data management system
    - Examples: Tamino
    - XLM enhancement of most RDBS
    - Support XPath and XQuery (some)
- Mapping documents to relations
  - Tree model of XML documents does not fit to table model of RDB

HS /Bio DBS05-XML2 22

## The mapping problem

- Variability of structures makes uniform mapping difficult

```
<text> .....This is a very long text.....
</text>
```

- "data centric"

```
<orders>
  <order ID= 4330>
    <itemList>
      <item ID>
        <price>
        <quantity>
        <description> ..A not so long txt
      </description>
      ...
    </itemList>
  </order>
</orders>
```

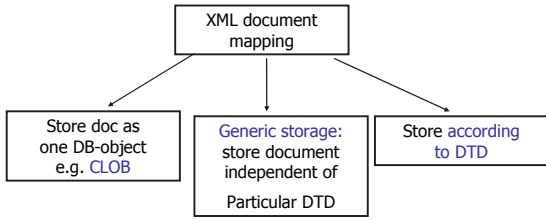
HS /Bio DBS05-XML2 23

## Why mapping to a DB?

- Non functional characteristics of DBS
  - Fault tolerance
  - Concurrent access
  - Stability
- Searching and transforming data (to some extent) are at the heart of every DBS
- Performance
  - Searching in large text strings?
  - Tree traverse ?
  - Joins?
- ....depends on types of operations and data

HS /Bio DBS05-XML2 24

# Classification of mappings



References:  
 M. Klette, H. Meyer: Speicherung von XML-Dokumenten – eine Klassifikation, Datenbank-Spektrum 5/2003  
 D. Florescu, D. Kossmann: A Performance Evaluation of Alternative Mapping Schemes for Storing XML Data in a Relational Database, Inria TR, 1999  
 A. Chaudri et al (eds.): XML Data Management, Addison-Wesley 2003

# Example document

```

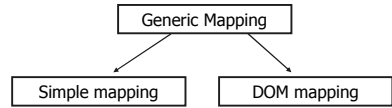
<Movie title="American Beauty" RunningTime="121"
imdbCode="0169547" Rating="R" >
  <Director>
    <First> Sam </First>
    <Last> Mendes </Last>
  </Director>
  <Award From="Oscar" Category="Best Director"> </Award>
  <PlotSummary> Lester Burnham is in a mid mid-life crisis...
</PlotSummary>
  <Cast>
    <Actor Role="Lester Burnham">
      <First> Kevin Kevin</First>
      <Last> Spacey Spacey</Last>
      <Award From="Oscar" Category="Best Actor"> </Award>
      <Award From="BAFTA" Category="Best Actor"> </Award>
    </Actor>
    <Actress Role="Carolyn Burnham">
      <First> Annette Annette</First>
      <Last> Bening Bening</Last>
      <Award From="BAFTA" Category="Best Actress"> </Award>
    </Actress>
  </Cast>
  <Award From="Oscar" Category="Best Film"> </Award>
  <Award From="BAFTA" Category="Best Film"> </Award>
</Movie>
  
```

```

<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT Actor (First,Last,Award+)>
<!ATTLIST Actor Role CDATA #REQUIRED>
<!ELEMENT Actress (First,Last,Award)>
<!ATTLIST Actress Role CDATA #REQUIRED>
<ELEMENT Award EMPTY>
  <!ATTLIST Award
    Category CDATA #REQUIRED
    From NMTOKEN #REQUIRED
  >
<ELEMENT Cast (Actor,Actress)>
<ELEMENT Director (First,Last,Award)>
<ELEMENT First (#PCDATA)>
<ELEMENT Last (#PCDATA)>
<ELEMENT Movie (Director,PlotSummary,Cast,Award+)>
  <!ATTLIST Movie
    Rating NMTOKEN #REQUIRED
    RunningTime NMTOKEN #REQUIRED
    Title CDATA #REQUIRED
    imdbCode NMTOKEN #REQUIRED
  >
<ELEMENT PlotSummary (#PCDATA)>
  
```

DTD for Movie DB

# Generic Mapping



**Simple relational:**  
 element- attribute- and edges relations  
 preserve order of elements

**DOM oriented:**  
 map class structure of DOM model  
 onto relations or classes (oo or OR DBS)

# Simple

- Element table
  - An element row for each element in the document
    - A generated document id
    - A generated element id
    - Predecessor of element node
    - Order of childs
    - Value, if element has a value
- Attribute table
  - Row for each (attribute , value) pair in document
    - Element id of this attribute
    - Attribute name
    - Attribute value
    - Order (if needed)

DocId	element	ID	predec	order	value
m002	movie	200	-	1	-
m002	director	201	200	1	
m002	first	202	201	1	Sam
m002	last	203	201	2	Mendes
m002	award	204	201	1	
m002	cast	205	200	1	
m002	actor	206	205	1	
m002	first	207	206	1	Kevin
m002	last	208	206	2	Spacey
...					

elemId	DocId	attribute	val	order
200	m002	title	American Beauty	1
200	m002	RunningTime	121	2
204	m002	from	Oscar	1
...	...			

## Simple generic mapping

- Advantage
  - Independent of document structure
  - Simple database structure
- Disadvantage
  - MANY joins to reconstruct the document
  - Attribute types
    - No problem if only "string" type represented
    - Cast to other types: inlining with one value column per type or value table for each type

element id	...	valString	valInt
m002 last	203 201 2	Mendes	null
m002 age	204 201 1	null	53

- ....
- At least n recursive joins when traversing on an n-element path from root down the tree

HS /Bio DBS05-XML2 31

## DOM-oriented generic mapping

- Node model of DOM mapped to relations
    - Node table
    - Node : element | attribute
    - Value table
- TreeStructTable

node_id	node_type	doc_id	parent	prev_sibling	next_sibling
---------	-----------	--------	--------	--------------	--------------

NodeValueTab

node_id	tag	value
---------	-----	-------

HS /Bio DBS05-XML2 32

## Alternative representation of order

- Coding of node order with parent-successor relation
  - preorder
    - does not represent parent child relationship
  - ( preorder# , bound) where
    - bound = max {preorder#(y) | y successor of this}
  - y successor of x  $\Leftrightarrow$  pre(x) < pre (y) and bound(x) >= pre(y)
    - very easy successor test

HS /Bio DBS05-XML2 33

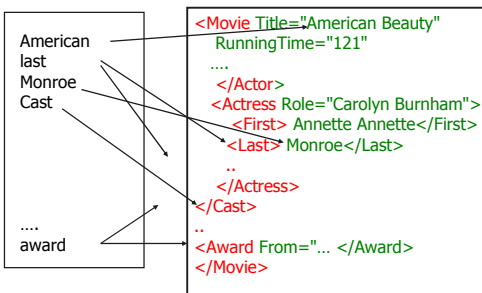
## XML document as one DB object

- BLOB or CLOB
  - => no reconstruction effort, no query support
  - useful for document centric objects ?
- (User) defined type "text"
  - => no reconstruction effort, indexing support queries like " find all occurrences of 'actor' " (in a DB storing movies) , no separation of tags and data (!),
  - More sophisticated queries like:
    - "find occurrences of 'Monroe' where 'actress' occurs in a window of n words before/after"
    - locates "actress" elements with (partial) value 'Monroe'

HS /Bio DBS05-XML2 34

## XML document as one DB object

- Indexed 'text' attribute



HS /Bio DBS05-XML2 35

## XML document as one DB object

- Querying
    - only boolean queries on keywords
    - no difference between structure and content
    - only useful in document centric applications
  - Search predicates (Oracle)
    - CREATE Table MovieTab AS (id INTEGER, txt CLOB)
    - SELECT id, txt, SCORE(1)
      - From MovieTab
      - WHERE CONTAINS (txt, 'Monroe',1)>0 ;
    - Score: relevance measure
    - Value of CONTAINS: 0..100
- Place holder for relevance

HS /Bio DBS05-XML2 36

## XML document as one DB object

- Improvement "XML type"
    - Difference between markup and strings
    - Allows to search between <tag> and </tag>
- ```
CREATE Table MovieTab AS (id INTEGER, doc XMLType)
SELECT id, doc, SCORE(1)
  From MovieTab
 WHERE CONTAINS (txt, 'Monroe WITHIN Last',1)>0 ;

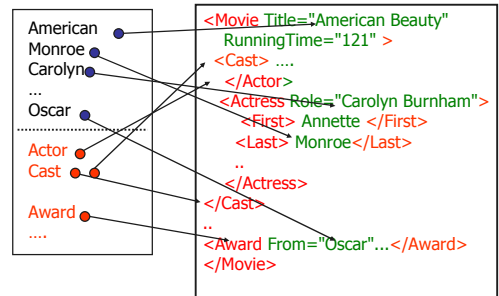
SELECT id
  FROM MovieTab
 WHERE CONTAINS (txt, 'Monroe INPATH(//Actress/Last'...);

.... CONTAINS (txt, 'HASPETH(//Actress/Last="Monroe"'.)..;
// exact match required
```
- For performance reasons separation of structure index and data index

HS /Bio DBS05-XML2 37

## Sophisticated indexing

- Relational attribute of XML type



HS /Bio DBS05-XML2 38

## DTD-Oriented mapping

- Automatic approach
  - Map each element to a relation
  - Object relational / object oriented: to a class
  - Attribute of element -> attribute of a relation / class
  - Tree structure by some ordering scheme
  - Easier to map into an object oriented / object relational data model
- User defined Mapping approach
  - Define a mapping between DTD and database schema
  - More flexible, more effort

See also R. Bourret: mapping DTDs to Databases

HS /Bio DBS05-XML2 39

## DTD-oriented approach

- Automatic:
  - Simple elements (not nested) -> attributes
  - Nested elements -> object types (classes)

```
DTD                                     Classes
<!ELEMENT A (B, C)>                     class A {
<!ELEMENT B (#PCDATA)>                   String b;
<!ATTLIST A                               C c;
F CDATA #REQUIRED>                       String f; }
<!ELEMENT C (D, E)>                       class C {
<!ELEMENT D (#PCDATA)>                   String d;
<!ELEMENT E (#PCDATA)>                   String e; }
```

- Sibling order is lost
- Could be retained by artificial order attributes

HS /Bio DBS05-XML2 40

## DTD-oriented approach

- Mapping guidelines

|           | XML         | Database Def        |
|-----------|-------------|---------------------|
| element   | root        | Relation            |
|           | simple      | attribute           |
|           | sequence    | attributes          |
|           | alternative | attributes          |
|           | elem with ? | attribute, NULL     |
|           | elem +, *   | SET, LIST           |
|           | complex     | Class (Object type) |
| attribute | XML attr    | Attribute of Rel    |
|           | #IMPLIED    | ^                   |
|           | #REQUIRED   | NOT NULL            |
|           | default     | default             |

HS /Bio DBS05-XML2 41

## DTD-oriented approach

- Some issues
  - Data types
    - DTD: manual adaption of schema necessary
    - XML schema: adapt data types to database types (automatically)
  - Mixed content
    - <desc> The values are
      - <simple> like 'X'</simple> or
      - <compound> like '(x,y)' </compound>
    - </desc>
      - Map onto different attributes and define an order
      - Tab (order, desc, simple, compound)
  - Order

HS /Bio DBS05-XML2 42

## DTD-oriented approach: order

```
<!ELEMENT A (#PCDATA | B | C)*>
<!ELEMENT B (#PCDATA)>
<!ELEMENT C (#PCDATA)>
```

```
class A {
    String[] pcdData;
    int[] pcdDataOrder;
    String[] b;
    int[] bOrder;
    String[] c;
    int[] cOrder; }
```

- Not supported by most systems
- +, \* can also be mapped to SET or LIST – if supported by data model

HS /Bio DBS05-XML2 43

## DTD-oriented approach

- Advantages
  - Object oriented / Object relational allow fine-granular, "natural" mapping of object structure
  - DTD is needed (of course) – compared to generic mapping
  - SQL and XQuery / XPath as query languages as opposed to 'XMLtype': basically operations on text
- Disadvantages
  - No document reconstruction
  - possible in principle, but expensive

HS /Bio DBS05-XML2 44

## User defined mapping

- Define your own mapping called Data access Definition in DB2

```
<ClassMap>
<ElementType name = "Movies"/>
<ToClassTable>
<Table Name="MovieTable"/>
</ToClassTable>
<PropertyMap>
<Attribute Name = "Title"/>
<ToColumn>
<Column Name = "Titel"/>
```

- Very flexible
- Makes sense if XML docs are mapped to an existing DB schema

HS /Bio DBS05-XML2 45

## 3.9 Information Integration

- Goal
  - access to all kinds of data of an application domain independent of local, data format, query language
  - Online access to sources ("Portals") instead of locally copied data
  - "Semantic linking" of databases / database queries
- Architectures
  - Language and infrastructure for linking DBs
  - Integrated architecture using wrappers and "mediator"

HS /Bio DBS05-XML2 46

## Information integration: SRS

- Sequence Retrieval Service (SRS)
  - Link operators ">" and "<" allow to link databases

Examples:

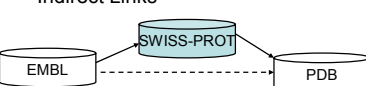
SWISS-PROT > PDB

finds all protein structures in PDB which are referenced by or linked to entries in SWISS-PROT

[swissport-def:kinase] > PDB

1. Kinase sequences in SWISS-PROT
  2. Link to PDB entries
- ⇒ Known tertiary structures of kinases

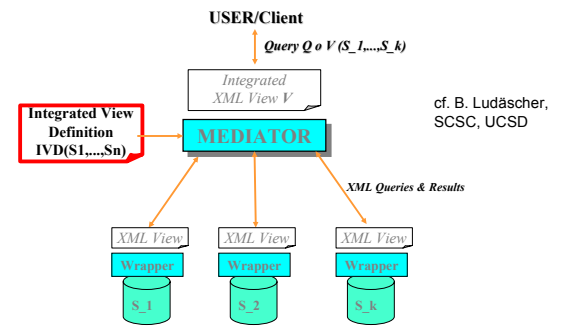
Basis: database of metadata



SRS finds "the best intermediate DB" to support a link

HS /Bio DBS05-XML2 47

## XML-Based Mediator Architecture

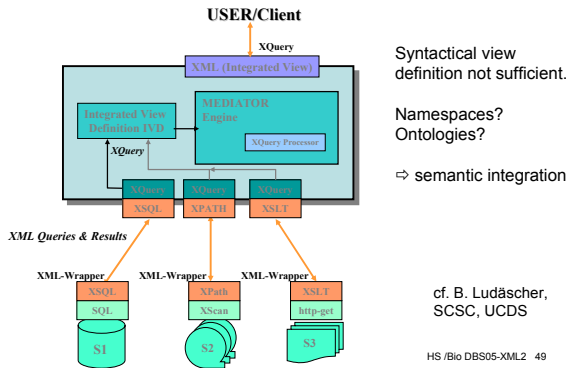


cf. B. Ludäscher, SCSC, UCSD

HS /Bio DBS05-XML2 48



## A Concrete XML-Based Mediator System



## 3.10 Ontologies

Def: **ontology** (from the [Greek](#) *ov* = *being* and *λόγος* = *word/speech*) is the most fundamental branch of [metaphysics](#). It studies [being](#) or [existence](#) as well as the [basic categories](#) thereof—trying to find out what entities and what types of entities exist.

Def-2: In [information science](#), an **ontology** is the product of an attempt to formulate an exhaustive and rigorous [conceptual schema](#) about a domain. An ontology is typically a [hierarchical data structure](#) containing all the relevant [entities](#) and their [relationships](#) and rules within that domain (e.g., a **domain ontology**).

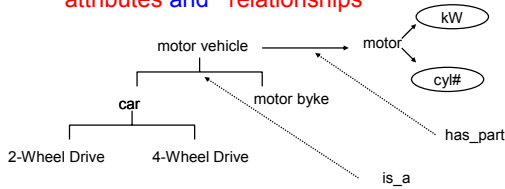
cited from Wiki encyclopedia

HS /Bio DBS05-XML2 50

## Ontologies

Ontology:

A formal description of concepts using attributes and relationships



Directed Graph with node and edge semantics

HS /Bio DBS05-XML2 51

## Ontologies

Languages for ontologies

- purely [descriptive](#)  
example: RDF triples (source, predicate, destination)  
(`person_url`, `is_author`, `doc_url`)

– Logic language

example: CYC

- individuals	<code>#\$Berlin</code>
- sets	<code>#\$cities</code>
- predicates	<code>#\$is_elem</code> <code>#\$Berlin</code> <code>#\$cities</code>
- functions	<code>#\$hasPopulation</code>

(`#$relationAllExists` `#$biologicalMother` `#$ChordataPhylum` `#$FemaleAnimal`)

for every instance of the collection `#$ChordataPhylum`, there exists an instance of `#$FemaleAnimal` which is its mother (described by the predicate `#$biologicalMother`).

HS /Bio DBS05-XML2 52

## Ontologies in Bioinformatics

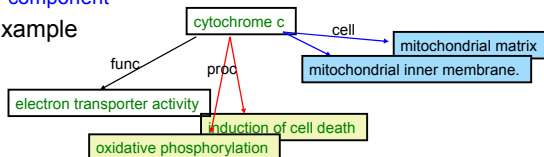
Goal:

consistent descriptions of gene products in different databases

Description of an entity by

molecular function, biological process and cellular component

Example

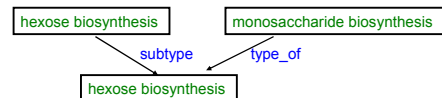


expl from [Gene Ontology](#).

HS /Bio DBS05-XML2 53

## Ontologies in Bioinformatics

Example: Generic relationships



Which relationships and other primitives for building an ontology?

Define set of relations, attributes, processes, functions.....in XML

HS /Bio DBS05-XML2 54

## Onotologies in Bioinformatics

### XML DTD for Gene Ontology exchange format

```
<!ELEMENT term (id|name|namespace|def?[is_a*|alt_id*|subset*|comment*?|is_anonymous*?|is_obsolete*?|is_root*?|xref_analog*|xref_unknown*|synonym*|relationship*|intersection_of*|union_of*|lexical_category*?)+>
```

<!-- TERM ELEMENTS -->

```
<!ELEMENT id (#PCDATA)>
```

```
<!ELEMENT name (#PCDATA)>
```

```
<!ELEMENT namespace (#PCDATA)>
```

```
<!ELEMENT def (defstr|dbxref)*>
```

<!-- DBXREF ELEMENTS -->

```
<!ELEMENT xref_analog (acc|dbname|name)+>
```

```
<!ELEMENT xref_unknown (acc|dbname|name)+>
```

```
<!ELEMENT is_a (#PCDATA)>
```

```
<!ELEMENT relationship (type|to)+>
```

```
<!ELEMENT type (#PCDATA)>
```

```
<!ELEMENT to (#PCDATA)>
```

```
<!ELEMENT alt_id (#PCDATA)>
```

```
<!ELEMENT subset (#PCDATA)>
```

```
<!ELEMENT comment (#PCDATA) .....>
```

HS /Bio DBS05-XML2 55

```
</term>
```

```
<id>GO:0001868</id>
```

```
<name>regulation of complement activation, lectin pathway</name>
```

```
<namespace>biological_process</namespace>
```

```
<def>
```

```
<defstr>Any process that modulates the frequency, rate or extent of the lectin pathway of complement activation.</defstr>
```

```
<dbxref>
```

```
<acc>0781735149</acc>
```

```
<dbname>ISBN</dbname>
```

```
<name>Fundamental Immunology</name>
```

```
</dbxref>
```

```
<dbxref>
```

```
<acc>add</acc>
```

```
<dbname>MGI</dbname>
```

```
<name></name>
```

```
</dbxref>
```

```
</def>
```

```
<is_a>GO:0030449</is_a>
```

```
<relationship>
```

```
<type>part_of</type>
```

```
<to>GO:0001867</to>
```

```
</relationship>
```

```
</term>
```

### A term definition from the Gene Ontology

... used by human readers ("what is the definition of...") and by programs:

- interoperability
- integration
- query processing

## Summary

- Strength and weaknesses of XML
  - flexible data structuring meta language for non-regular data
  - standardized
  - human and machine readable
  - Data management of XML documents advanced
  - XML took off already
  - ideal for syntactic integration and of course: data exchange
- Weaknesses
  - no inheritance
  - not types -> solved by XML schema
  - concept of "relationship" not well developed (IDREF)
  - Only "exact" query(XPath), structural similarity?

HS /Bio DBS05-XML2 57