Databases and the Web:
From Semistructured Data to XML

Based on Dan Suciu's VLDB 99
tutorial

How the Web is Today

- HTML documents
- all intended for human consumption
- many generated automatically by applications

Easy to fetch any Web page, from any server, any platform

Paradigm Shift on the Web

- new Web standard XML:
  - XML generated by applications
  - XML consumed by applications
- data exchange
  - across platforms: enterprise interoperability
  - across enterprises

Web: from collection of documents to data and documents

Database Techniques Can Help

- query optimization, processing
- views, transformations
- data warehouses, data integration
- mediators, query rewriting
- secondary storage, indexes

But Needs a Paradigm Shift Too

- Web data differs from database data:
  - self-describing, schema-less
  - structure changes without notice
  - heterogeneous, deeply nested, irregular
  - documents and data mixed together
- designed by document, not db experts
- need Web data management

Outline

- what the database community has done
  - semistructured data model
  - query languages, schemas
- what the Web community has done:
  - data formats/models: XML, RDF
  - transformation language (XSL), schemas
- where they meet and where they differ

Outline

- Semistructured data and XML
- Query languages
- Schemas
- Systems issues

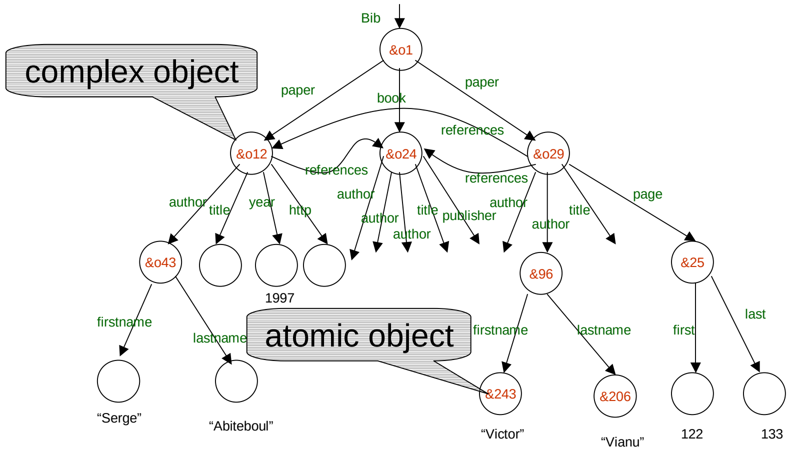
Part 1
Semistructured Data and XML

Semistructured Data

Origins:

- integration of heterogeneous sources
- data sources with non-rigid structure
- biological data
- *Web data*

The Semistructured Data Model



Object Exchange Model (OEM)

Syntax for Semistructured Data

```
Bib: &o1 { paper: &o12 { … },
           book:  &o24 { … },
           paper: &o29
                { author: &o52 “Abiteboul”,
                  author: &o96 { firstname: &243 “Victor”,
                                 lastname: &o206 “Vianu”},
                  title: &o93 “Regular path queries with constraints”,
                  references: &o12,
                  references: &o24,
                  pages: &o25 { first: &o64 122, last: &o92 133}
                }
         }
```

Syntax for Semistructured Data

May omit oid’s:

```
{ paper: { author: “Abiteboul”,
           author: { firstname: “Victor”,
                     lastname: “Vianu”},
           title: “Regular path queries …”,
           page: { first: 122, last: 133 }
         }
}
```

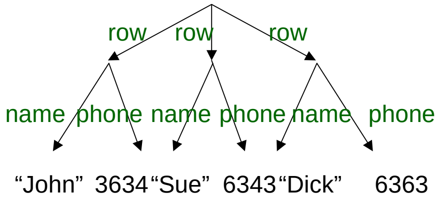
Characteristics of Semistructured Data

- missing or additional attributes
- multiple attributes
- different types in different objects
- heterogeneous collections

self-describing, irregular data, no a priori structure

Comparison with Relational Data

name	phone
John	3634
Sue	6343
Dick	6363

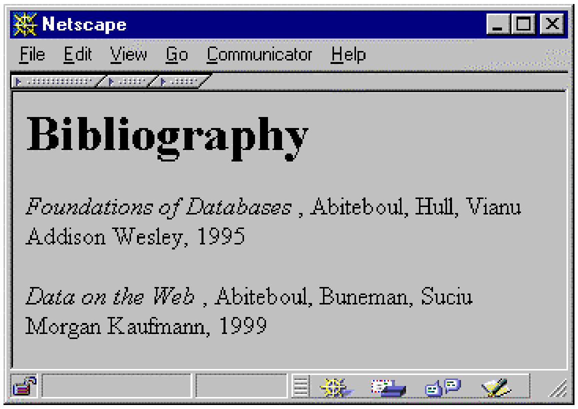


{ row: { name: "John", phone: 3634 },
row: { name: "Sue", phone: 6343 },
row: { name: "Dick", phone: 6363 }
}

XML

- a W3C standard to complement HTML
- origins: structured text SGML
- motivation:
  - HTML describes presentation
  - XML describes content
- HTML4.0 $\in$ XML $\subset$ SGML
- http://www.w3.org/TR/REC-xml (2/98)

From HTML to XML



Bibliography

Foundations of Databases , Abiteboul, Hull, Vianu
Addison Wesley, 1995

Data on the Web , Abiteboul, Buneman, Suciu
Morgan Kaufmann, 1999

HTML describes the presentation

HTML

```
<h1> Bibliography </h1>
<p> <i> Foundations of Databases </i>
        Abiteboul, Hull, Vianu
        <br> Addison Wesley, 1995
<p> <i> Data on the Web </i>
        Abiteoul, Buneman, Suciu
        <br> Morgan Kaufmann, 1999
```

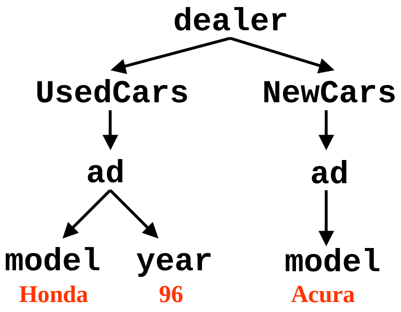
XML

```
<bibliography>
     <book>    <title> Foundations… </title>
               <author> Abiteboul </author>
               <author> Hull </author>
               <author> Vianu </author>
               <publisher> Addison Wesley </publisher>
               <year> 1995 </year>
     </book>
     …
</bibliography>
```

XML describes the content

Basic XML Document Structure

```
<dealer>
 <UsedCars>
  <ad>
   <model>Honda</model>
   <year>96</year>
  </ad>
 </UsedCars>
 <NewCars>
  <ad>
   <model>Acura</model>
  </ad>
 </NewCars>
</dealer>
```



XML Terminology

- tags: book, title, author, …
- start tag: <book>, end tag: </book>
- elements: <book>…<book>,<author>…</author>
- elements are nested
- empty element: <red></red> abbrv. <red/>
- an XML document: single *root element*

well formed XML document: if it has matching tags

More XML: Attributes

```
<book price = “55” currency = “USD”>
  <title> Foundations of Databases </title>
  <author> Abiteboul </author>
   …
  <year> 1995 </year>
</book>
```

attributes are alternative ways to represent data

More XML: Oids and References

```
<person id=“o555”>  <name> Jane </name> </person>

<person id=“o456”>  <name> Mary </name>
                    <children idref=“o123 o555”/>
</person>

<person id=“o123” mother=“o456”><name>John</name>
</person>
```

oids and references in XML are just syntax

XML v.s. Semistructured Data

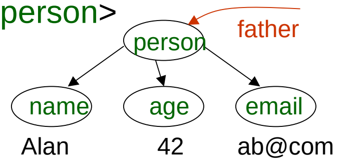
- both described best by a graph
- both are schema-less, self-describing

Similarities and Differences

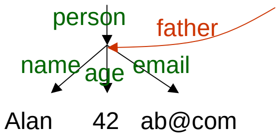
```
<person id="o123">
  <name> Alan </name>
  <age> 42 </age>
  <email> ab@com </email>
</person>
```

```
{ person: &o123
    { name: "Alan",
      age: 42,
      email: "ab@com" }
}
```

```
<person father="o123"> …
</person>
```



```
{ person: { father: &o123 …}
}
```



similar on trees, different on graphs

More Differences

- XML is ordered, ssd is not
- XML can mix text and elements:

```
<talk> Making Java easier to type and easier to type
        <speaker> Phil Wadler </speaker>
</talk>
```

- XML has lots of other stuff: entities, processing instructions, comments

---

Summary of Data Models

- semistructured data, XML
- data is self-describing, irregular
- schema embedded in the data

Part 2
Query Languages

- Semistructured data and XML
- Query languages
- Schemas
- Systems issues
- Conclusions

Query Languages: Outline

- for semistructured data:
  - Lorel
  - UnQL
  - StruQL
- for XML: XML-QL
- a different paradigm
  - structural recursion
  - XSL

Lorel

- part of the Lore system (Stanford)
- adapts OQL to semistructured data

example:

```
select X.title
from   Bib.paper X
where X.year > 1995
```

abbreviated to:

```
select Bib.paper.title
from   Bib.paper
where Bib.paper.year > 1995
```

Lorel v.s. OQL

- implicit coercions: 1995 to “1995”
- missing attributes
  - empty answer v.s. type error
- set-valued attributes
  - in X.year>1995, X may have several years
- regular path expressions (next)

Regular Path Expressions

```
select X.title
from    Bib.paper X, Bib.(paper|book) Y
where Y.author.lastname? = “Ullman”
    and Y.reference+ X
```

Useful for:

- syntactic substitute for inheritance: paper|book
- navigating partially known structures: lastname?
- transitive closure: reference+

UnQL

- Unstructured Query Language
- patterns, templates, structural recursion
- patterns:

```
select T
where Bib.paper: { title: T, year: Y, journal: “TODS”}
    and  Y > 1995
```

UnQL: Templates

```
select result: { fn: F, ln: L, pub: { title: T, year: Y }}
where Bib.paper: { title: T, year: Y, journal: “TODS”}
   and  Y > 1995
```

Result looks like:

```
 { result: { fn: “John”, ln: “Smith”,
               pub: { title: “P equals NP”, year: 2005}},
   result: { fn: “Joe”, ln: “Doe”,
               pub: { title: “Errata to P=NP”, year: 2006}}
     … }
```

Skolem Functions

- Maier, 1986
  - in OO systems
- Kifer et al, 1989
  - F-logic
- Hull and Yoshikawa, 1990
  - deductive db (ILOG)
- Papakonstantinou et al., 1996
  - semistructured db (MSL)
- illustrate with Strudel (next)

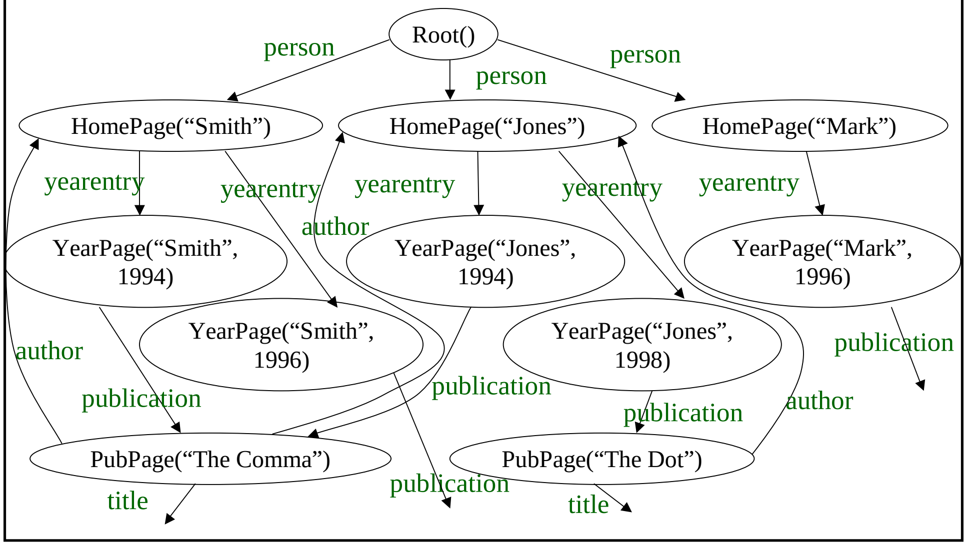
Skolem Functions in StruQL

- Strudel: a Web Site Management System
- StruQL: its query language

Example: Bibliography Data

```
{Bib: { paper: { author: “Jones”,
                   author: “Smith”,
                   title: “The Comma”,
                   year: 1994
                  }
        },
        { paper: ….. }
}
```

Example: A Complex Web Site



Example: Skolem Functions in StruQL

```
where Root -> "Bib" -> X, X -> "paper" -> P,
         P -> "author" -> A, P -> "title" -> T, P -> "year" -> Y

create  Root(), HomePage(A), YearPage(A,Y), PubPage(P)

link    Root() -> "person" -> HomePage(A),
         HomePage(A) -> "yearentry" -> YearPage(A,Y),
         YearPage(A,Y) -> "publication" -> PubPage(P),
         PubPage(P) -> "author" -> HomePage(A),
         PubPage(P) -> "title" -> T
```

XML-QL:
A Query Language for XML

- http://www.w3.org/TR/NOTE-xml-ql (8/98)
- features:
  - regular path expressions
  - patterns, templates
  - Skolem Functions
- based on OEM data model

Pattern Matching in XML-QL

```
where <book language=“french”>
              <publisher>
                     <name> Morgan Kaufmann </name>
              </publisher>
              <author> $a </author>
          </book> in “www.a.b.c/bib.xml”
construct $a
```

Simple Constructors in XML-QL

```
where <book language = $l>
                    <author> $a </>
          </> in “www.a.b.c/bib.xml”
construct <result> <author> $a </> <lang> $l </> </>
```

Note: </> abbreviates </book> or </result> or ...

```
<result> <author>Smith</author><lang>English</lang></result>
<result> <author>Smith</author><lang>Mandarin</lang></result>
<result> <author>Doe</author><lang>English</lang></result>
```

Skolem Functions in XML-QL

```
where <book language = $l>
             <author> $a </>
          </> in “www.a.b.c/bib.xml”
construct <result> <author id=F($a)> $a</>
                           <lang> $l </>
                </>
```

```
<result> <author>Smith</author>
           <lang>English</lang> <lang>Mandarin</lang>
</result>
<result> <author>Doe</author> <lang>English</lang> </result>
```

A Different Paradigm: Structural Recursion

Data as *sets* with a *union* operator:

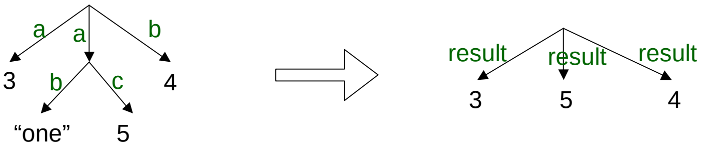
{a:3, a:{b:"one", c:5}, b:4} =

{a:3} U {a:{b:"one",c:5}} U {b:4}

Structural Recursion

Example: retrieve all integers in the data

```
f(T1 U T2) =  f(T1) U f(T2)
f({L: T})     =  f(T)
f({})            =  {}
f(V)             =  if isInt(V) then {result: V} else {}
```



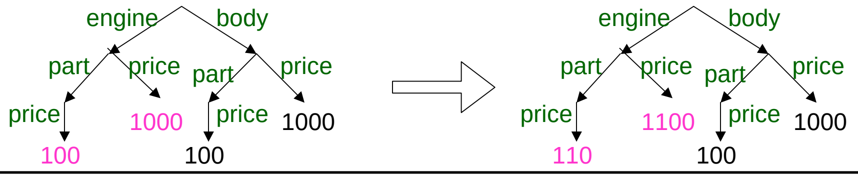
standard textbook programming on trees

Structural Recursion

Example: increase all engine prices by 10%

```
f(T1 U T2) =  f(T1) U f(T2)
f({L: T})     =  if L= engine
                            then {L: g(T)}
                     else {L: f(T)}
f({})          =  {}
f(V)           =  V
```

```
g(T1 U T2) =  g(T1) U g(T2)
g({L: T})     =  if L= price
                             then {L:1.1*T}
                     else {L: g(T)}
g({})          =  {}
g(V)           =   V
```



XSL

- two W3C drafts: XSLT and XPATH
  - http://www.w3.org/TR/xpath, 7/99
  - http://www.w3.org/TR/WD-xslt, 7/99
- in commercial products (e.g. IE5.0)
- purpose: stylesheet specification language:
  - stylesheet: XML -> HTML
  - in general: XML -> XML

```
<a> <e> <b> <c> 1 </c>
              <c> 2 </c>
         </b>
         <a> <c> 3 </c>
         </a>
    </e>
    <c> 4 </c>
</a>
```

⇒

```
<A> <B> <C> 1 </C>
          <C> 2 </C>
     </B>
     <A> <C> 3  </C>
     </A>
     <C> 4 </C>
</A>
```

XSL is Structural Recursion

Equivalent to:

```
f(T1 U T2) = f(T1) U f(T2)
f({L: T})     =  if L= c then {C: t}
                 else L= b then {B: f(t)}
                 else L= a then {A: f(t)}
                 else f(t)
f({})          =  {}
f(V)           =  V
```

Summary of Query Languages

- studied extensively in semistructured data
- some quite powerful features
- no standard for XML QL yet (WG soon)
- XSL available today (for stylesheets)
- XSL = structural recursion

Part 3
Schemas

- Semistructured data and XML
- Query languages
- Schemas
- Systems issues
- Conclusions

Schemas

- why ?
  - XML: to describe semantics
  - semistructured data: to improve processing
- what ?
  - semistructured data: foundational
  - XML: several concrete proposals

here lies our interest

Schemas

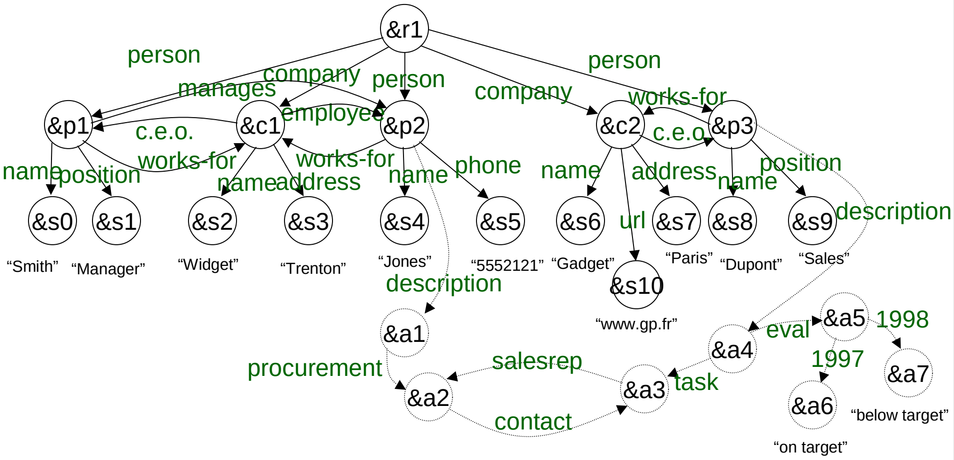
- when ?
  - semistructured data, XML: a posteriori
  - RDBMS: a priori, to interpret binary data
- how ?
  - semistructured data: schema is independent
  - XML: schema is hardwired with the data

Outline

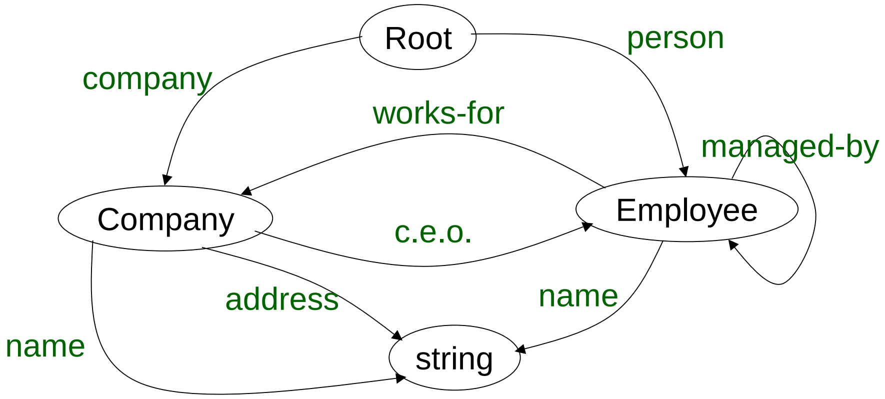
- schemas for semistructured data:
  - – foundations
  - – schema extraction
- schemas for XML:
  - – DTD
  - – XML-Schema

Schemas: An Example

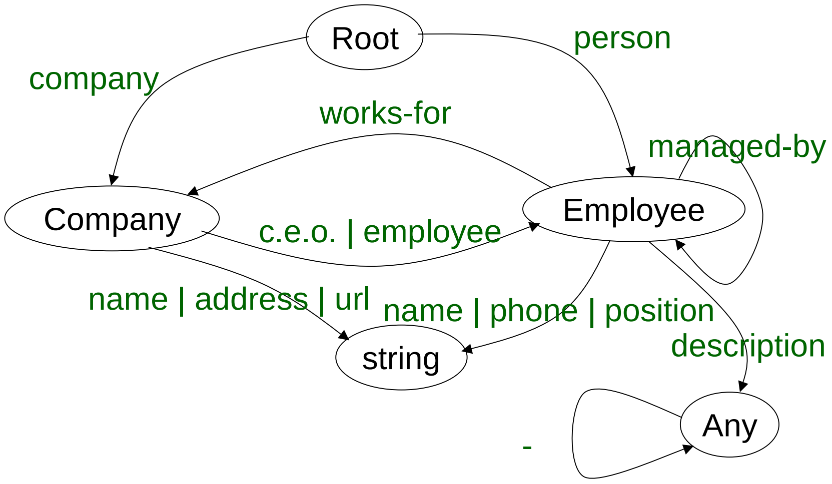
Some database:



Lower-Bound Schemas



Upper Bound Schemas



The Two Questions to Ask

Conformance: does that data conform to this schema ?

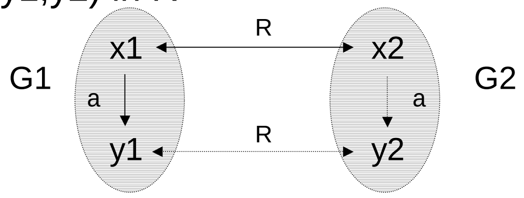
Classification: if so, then which objects belong to what classes ?

Graph Simulation

Definition Two edge-labeled graphs G1, G2

A *simulation* is a relation R between nodes:

- if (x1, x2) in R, and (x1,a,y1) in G1, then exists (x2,a,y2) in G2 (same label) s.t. (y1,y2) in R



Note: a simulation can be efficiently computed [Henzinger, et a. 1995]

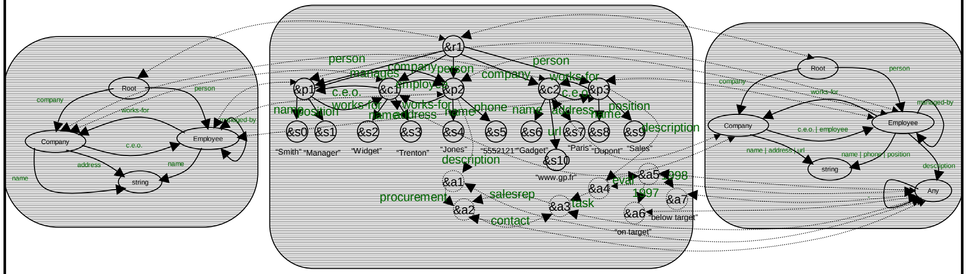
Using Simulation

Data graph D, schema S

- upper bound schema:
  - – conformance: find simulation R from D to S
  - – classification: check if (x,c) in R
- lower bound schema
  - – conformance: find simulation R from S to D
  - – classification: check if (c,x) in R

[Buneman et al 1997]

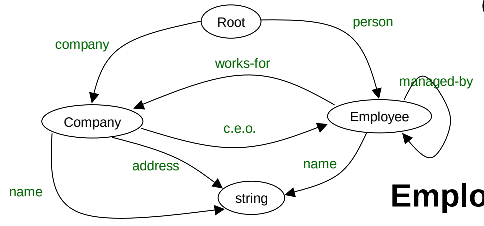
Example



Lower Bound Database Upper Bound

simulation: efficient technique for checking conformance to schema

Application 1:
Improve Secondary Storage



Lower-bound schema

Company

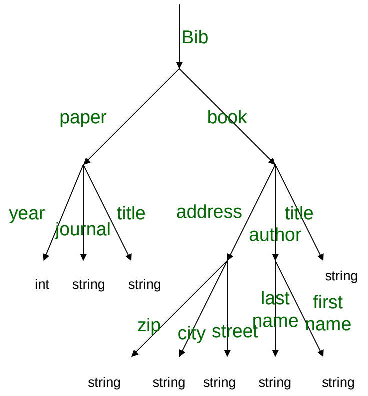
oid	name	address	c.e.o.
…	…	…	…
…	…	…	…

Employee

oid	name	managed-by	works-for
…	…	…	…
…	…	…	…

Store rest in overflow graph

Application 2:
Query Optimization



Upper-bound schema

select X.title
from Bib._ X
where X.*.zip = “12345”

select X.title
from Bib.book X
where X.address.zip = “12345”

[Fernandez, Suciu 1998]

Schema Extraction (From Data)

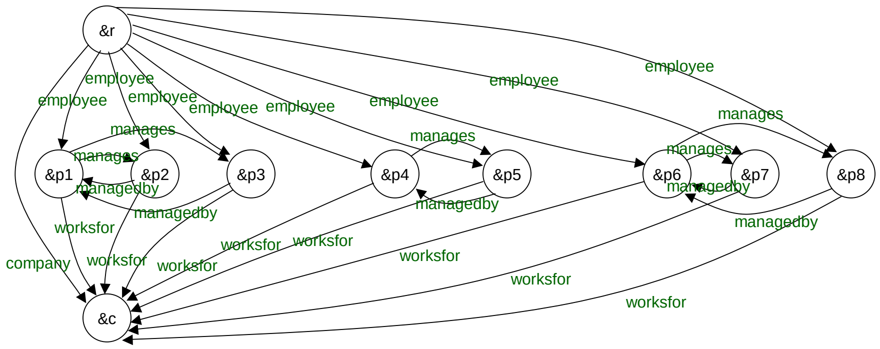
Problem statement

- given data instance D
- find the “most specific” schema S for D

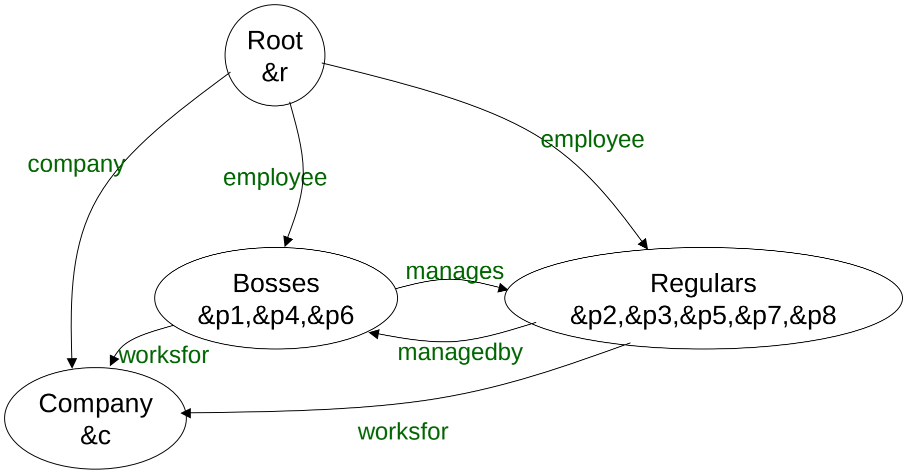
In practice: S too large, need to relax

[Nestorov et al. 1998]

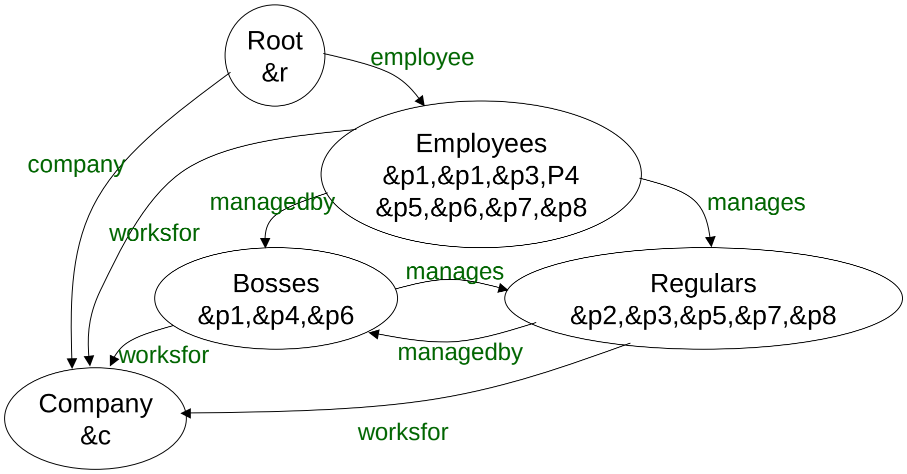
Schema Extraction: Sample Data



Lower Bound Schema Extraction



Upper Bound Schema Extraction: Data Guides



Schemas in XML

- Document Type Definition (DTD)
- XML Schema
- RDF Schema

Document Type Definition: DTD

- part of the original XML specification
- an XML document may have a DTD
- terminology for XML:
  - well-formed: if tags are correctly closed
  - valid: if it has a DTD and conforms to it
- validation is useful in data exchange

DTDs as Grammars

```
<!DOCTYPE paper [
  <!ELEMENT paper   (section*)>
  <!ELEMENT section ((title,section*) | text)>
  <!ELEMENT title     (#PCDATA)>
  <!ELEMENT text      (#PCDATA)>
]>
```

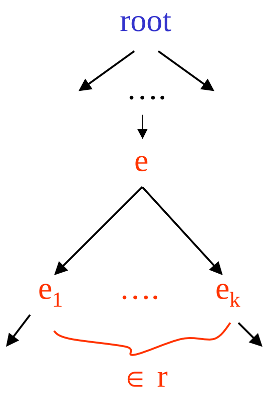
```
<paper> <section> <text> </text> </section>
        <section> <title> </title> <section> … </section>
                                   <section> … </section>
        </section>
</paper>
```

Σ: alphabet of element names, root ∈ Σ

set of rules:

$$\underset{\text{element name}}{e} \longrightarrow \underset{\text{regular expression over } \Sigma}{r}$$

Documents satisfying a DTD

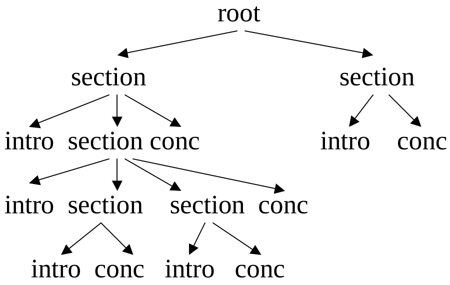


$e \longrightarrow r$

Example

A DTD and a document satisfying it:

```
root    → section*;
section → intro, section*,conclusions;
```

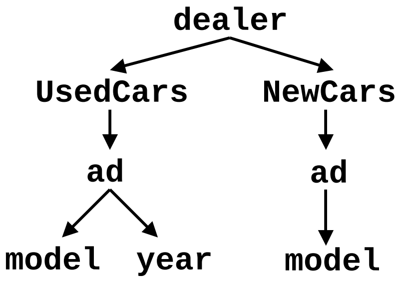


DTDs as Schemas

Shortcomings:

- impose unwanted constraints on order
  - <!ELEMENT person (name,phone)>
- references cannot be constrained
- can be too vague:
  - <!ELEMENT person ((name|phone|email)*)>

Another shortcoming of DTDs: context-free definition



Cannot write a DTD describing this single document:
ad has different structure in different contexts

XML Schemas

- very recent proposal
- unifies previous schema proposals
- generalizes DTDs
- uses XML syntax
- two documents: structure and datatypes
  - – http://www.w3.org/TR/xmlschema-1
  - – http://www.w3.org/TR/xmlschema-2

XML Schemas

```
<elementType name=“paper”>
    <sequence>
        <elementTypeRef name=“title”/>
        <elementTypeRef name=“author” minOccurs=“0”/>
        <elementTypeRef name=“year”/>
        <choice> <elementTypeRef name=“journal”/>
                 <elementTypeRef name=“conference”/>
        </choice>
    </sequence>
</elementType>
```

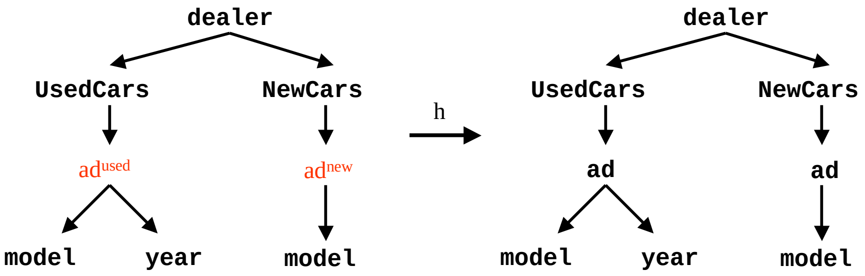
DTD: <!ELEMENT paper (title,author*,year, (journal|conference))>

Specialization (decoupled tags)

specialize ad: ad^{used} and ad^{new}

Dealers ⟶ UsedCars NewCars
UsedCars ⟶ ad^{used}
NewCars ⟶ ad^{new}
ad^{used} ⟶ model year
ad^{new} ⟶ model

$h(ad^{new}) = ad$
$h(ad^{used}) = ad$



Specialization (decoupled tags)

- Alphabet Σ
- Specialized alphabet Δ
- Specialization mapping h: Δ ⟶ Σ

  $h(a') = a$ “a’ is a specialization of a”
- DTD d over specialized alphabet Δ

A tree t with labels in Σ satisfies
the specialized DTD d iff $t \in h(T(d))$

- What sets of trees can DTDs + specialization define?

Exactly the regular tree languages
of unranked trees!
[Bruggemann-Klein+Murata+Wood]

- Consequences:

same closure properties (e.g. intersection, union, complement) and complexities of manipulations

---

Summary of Schemas

- in SS data:
  - graph theoretic
  - data and schema are decoupled
  - used in data processing
- in XML
  - from grammar to object-oriented
  - schema wired with the data
  - emphasis on semantics for exchange

Part 4
Systems Issues

- Semistructured data and XML
- Query languages
- Schemas
- Systems issues
- Conclusions

Systems Issues

- servers
- mediators

Servers for Semistructured Data / XML

- storage
- index
- query evaluation [McHugh, Widom 1999]

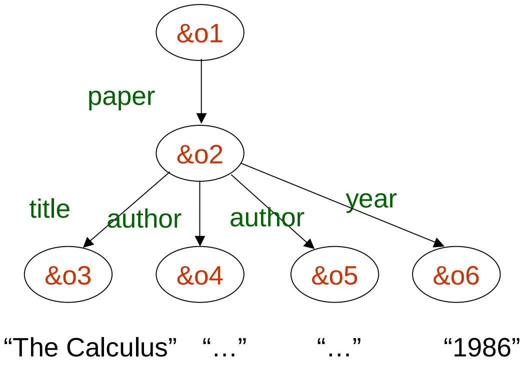
XML Storage

- text file (XML)
- store in ternary relation
- use DTD to derive schema
- mine data to derive schema
- build special purpose repository (Lore)

XML Storage: Text File

- advantages
  - simple
  - less space than one thinks
  - reasonable clustering
- disadvantage
  - no updates
  - require special purpose query processor

Store XML in Ternary Relation



Ref

Source	Label	Dest
&o1	paper	&o2
&o2	title	&o3
&o2	author	&o4
&o2	author	&o5
&o2	year	&o6

Val

Node	Value
&o3	The Calculus
&o4	…
&o5	…
&o6	1986

[Florescu, Kossman 1999]

Use DTD to derive Schema

- DTD:

```
<!ELEMENT employee (name, address, project*)>
<!ELEMENT address    (street, city, state, zip)>
```

- ODMG classes:

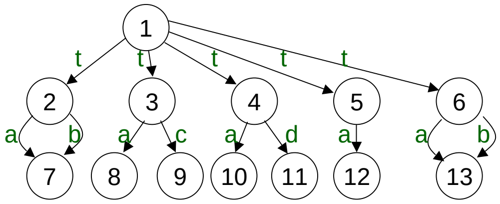
```
class Employee public type tuple
  (name:string, address:Address, project:List(Project))
class Address public type tuple (street:string, …)
```

- [Christophides et al. 1994 , Shanmugasundaram et al. 1999]

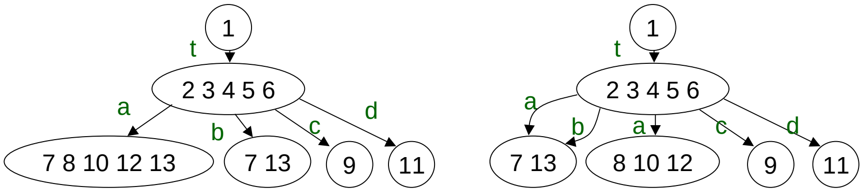
Indexing Semistructured Data

- regular path expressions
  - data guides [Goldman, Widom, 1997]
  - T-indexes   [Milo, Suciu, 1999]

Indexing All Paths in the Data



Semistructured Data



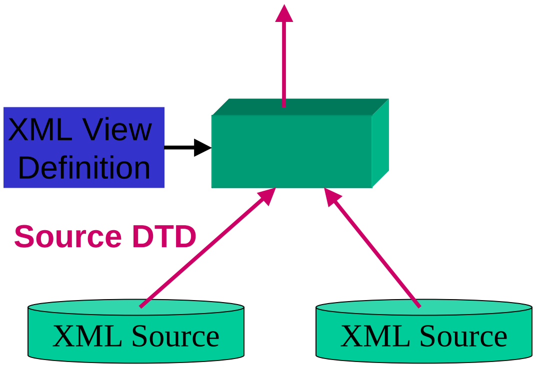
Data Guide

T-Index

Mediators for Semistructured Data / XML

- XML = virtual view of Relational/OO/OR sources
- mediator = translation, integration
- issues:
  - query composition and rewriting [Papakonstatinou, et al. 1996]
  - limited source capabilities [Yerneni, et al. 1999]
  - DTD inference, checking conformance to target DTD

Mediator of XML sources



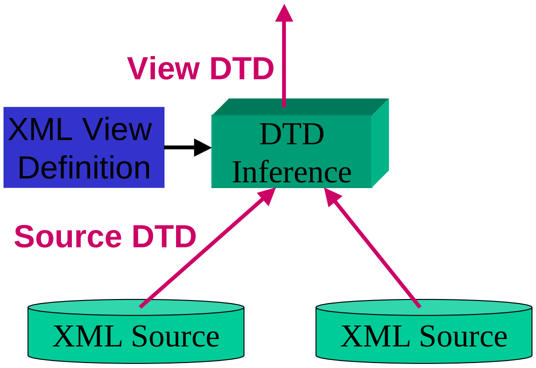
XML View Definition

Source DTD

XML Source

XML Source

Schema Inference or Conformance Checking



View DTD

XML View Definition

DTD Inference

Source DTD

XML Source

XML Source

Example: An XML Mediator

- relational database:

Store

sid	name
…	…
…	…

SB

sid	bid
…	…
…	…

Book

bid	title
…	…
…	…

- virtual XML view:

```
<store> <name> n1 </name>
            <book> ...  </book>
            <book> ...  </book>
            ...
 </store>
 <store> <name>n2 </name>
            <book> ...  </book>
            <book> ...  </book>
            …
</store>
```

Example: An XML Mediator

- specify mediator declaratively (a view):

```
from      Store,  SB,  Book
where     Store.sid=SB.sid and
          SB.bid=Book.bid
construct <store ID=f(Store.sid)>
              <name> Store.name </name>
               <book> Book.title </book>
          </store>
```

Example: An XML Mediator

- users ask XML-QL queries:
  - find stores who sell “The Calculus”

```
where      <store>  <name> $n </name>
                    <book>  The Calculus  </book>
           <store>
construct <result> $n </result>
```

Example: An XML Mediator

- system composes query with view:

```
from   Store, SB, Book
where Store.sid=SB.sid   and
       SB.bid=Book.bid  and
       Book.title=“The Calculus”
construct <result> Store.name </result>
```

Summary of Systems

- unclear today how XML will be used
  - materialized ? Need servers
  - virtual ? Need mediators
- most work is still ahead

Summary

- XML = what is out there
- semistructured data = what we can process
- paradigm shift, for both Web and db
- covered here:
  - data models, queries, schemas

Current and Future Technologies

- Web applications possible today:
  - – export relational data to XML (e.g. Oracle)
  - – import XML directly into applications
- Web applications in the future:
  - – mediator technology (XML view)
  - – store/process native XML data
  - – compress XML
  - – mine/analyze XML

Why This Is Cool for Database People

- put to work what you learned in CS101 !
  - – tree traversals (structural recursion, XSL)
  - – automata theory (DTD’s, path expressions)
  - – graph theory (simulation)
- adapt old DB tricks to new kind of data
- save the trees: from fax to XML

Further Readings

www. w3.org/XML

Abiteboul, Buneman, Suciu
Data on the Web: From Relational to Semistructured to XML
Morgan Kaufmann, 1999