Modeling and Querying Web Data

Dr P Sreenivasa Kumar
Professor
Department of CS&E
I I T - Madras

Data - Metadata

- Relational model
  - Data : values in the relational tuples
  - Metadata : the schema information
    - The relation names
    - The attribute names and their types
    - The constraints – referential integrity, key etc
- Object data model
  - Data : objects or class instances
  - Metadata:
    - Object or class definitions, relationship definitions etc

Semi-structured Data

Neither raw data nor strictly typed data

Raw data : images / sound data

Typed/structured data:

- Relational Data - table-oriented
- Object Data - ODMG model

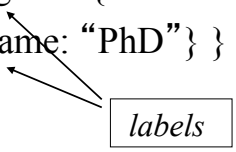
Typical sources of semi-structured data:

- Web data
- Data integrated from many heterogeneous data sources

Syntax for Data

- A simple but yet versatile syntax for data

  {name: "mathematics", office: "HSB231", phone: 8452,
  hod: "Prof Mathews", program: {name: "MSc",
  strength: 25}, program: {name: "PhD"} }



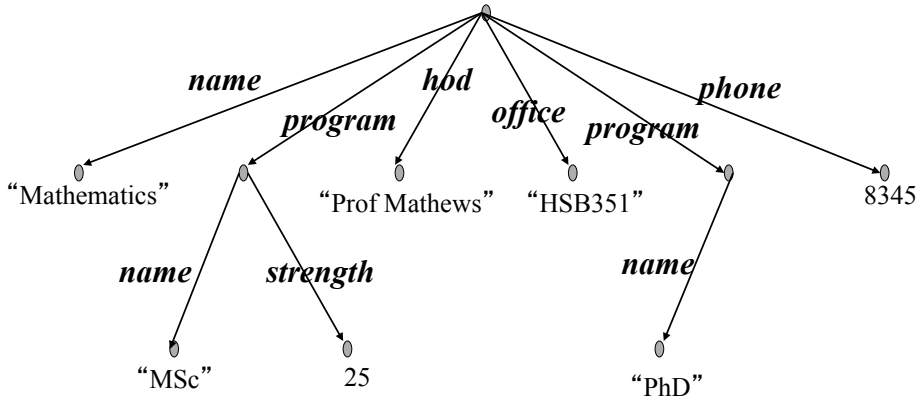
- General Syntax:

  $t_0 :: = \{l_1: t_1, l_2 : t_2 , \ldots , l_n: t_n \} \mid a$

  l_i - label , t_i - trees

  a - atomic value - integer/real/string/gif/jpeg etc

Tree Representation



Typical Data

{institute:
  {dept: { name: "maths", address: "HSB321",
    phone: "8510", head: "Choudam"}}
  {dept: { name: "cs", address: {bldg: "BSB", room: 301},
    phone: 8331, phone: 8330, head: "Raman"}}
  {unit: { name: "icsr", address: "#1,Alumni Ave",
    phone:{reception: 8401, sec-dean: 8402, dean: 8400},
    head: "Murthy" }}
  {center:{ name: "cce", phone: 8702, head: "Rao"}
}

xml

Data Integration Context

- Sources
  - Autonomous, independent entities
  - Domain of operation – same / related
- Semi-structured data model
  - Appropriate option
  - Prior agreement on schema/structure
    - Difficult to achieve / sometimes not even possible
- Dream
  - Two independent Info Systems
    - Able to exchange data without any prior arrangement
  - Ontologies

Properties of SSD

- Irregular structure
  - Heterogeneous elements, different types for same info,
  - Some elements are incomplete
- Indicative rather than constraining schema
  - Conventional model - strict typing - schema
  - Unenforceable in web-like applications
- A-priori schema -- a-posteriori data guide
  - Conventional - schema defined first - data loaded next
  - Semi-structured - data is self describing - infer schema

More Properties

- Fixed schema versus Evolving schema
  - conventional - schema changes - infrequent - costly
  - semi-structured - structure evolves rapidly
- Small schema versus Large schema
  - semi-structured data
    - large - heterogeneous origin
- Schema and data - blurred distinction
  - {person: {name: "rama", gender: "male", age: 36},
    person: {name: "rama", male: 0, female: 1, age: 30}}

SSD - Relational Data

- Student(name,rollNo,gender,address,phone,branch)
- {student:

  {row: {name: Suresh, rollNo: CS02023, gender: M,
  address: "12, I Main, Gandhi Nagar",
  phone: 441 4345, branch: CS },
  {row: {name: …
  },
  …
  }

Models for Semi-structured Data

- Object Exchange Model (OEM)
  - developed at the Stanford University DB Group
  - used in LORE (*l*ight-weight *o*bject *re*pository)
    - First database for s-s data
  - simple data exchange format - data integration
- Extensible Markup Language (XML)
  - developed by the document community
  - addresses the inadequacies in HTML for data exchange
  - data can be made self-describing
  - user defined tag sets versus fixed set of tags in HTML

Object Exchange Model

- OEM data
  - graph with objects as nodes and labels on edges
- Objects
  - entities are represented by objects
  - each object has a unique object-identifier (oid)
  - atomic objects : indivisible, of basic types
    - integer, real, string, Boolean, gif, html, jpeg etc
  - complex objects
    - set of object references - label, oid pairs

Example OEM data

Guide &11
- hotel &12
  - name &13 "janapriya"
  - category &14 "andhra style"
  - address &16 "25, Sardar Patel Road"
  - pin &17 600020
  - nearby &21
- hotel &21
  - name &22 "treat"
  - price &24 "costly"
  - category &23 "general"
  - address &25
    - street&26 "20, 4rth cross"
    - locality&27 "Gandhi Nagar"
    - pincode &28 "600 020"
  - nearby &12

Queries

Example OEM data (contd.)

- hotel &31
  - name &32 "saravana bhavan"
  - category &33 "general"
  - price &34 "inexpensive"
  - address &35 "25 Usman Road, T Nagar"
  - address &36 "33 Pondy Bazar"
- hotel &41
  - name &42 "eden"
  - category &43 "continental"
  - price &44 "very costly"
  - address &45
    - street &46 "19, Beach Road"
    - locality &47 "Besent Nagar"

Queries

Querying Semi-structured Data

- Lorel language
  - companion to the Lore - light-weight object repository
  - simple language for flexible querying of s-s data
- Features
  - type coercion
    - query should not fail due to type mismatches
    - comparisons - converted into existential checks
  - powerful path expressions
    - regular expressions on the alphabet of labels
    - regular expressions on label strings

Path Expressions

- Simple Path Expressions
  - sequence of labels starting with the root

    Guide.hotel.address.street

    set of objects reachable using the path

    {“20, 4rth cross”, “19, Beach Road”}
- General Path Expressions
  - use regular expressions on labels and label completion
  - wild card characters
    - % - a sequence of one or more characters in a label / label
    - # - matches a path of length zero or more labels

General Path Expressions

- Simplified Grammar

path-expr ::= var {gpe-component}+
var ::= identifier
gpe-component ::= "•"label_expr |
gpe-component "|" gpe-component |
gpe-component gpe-component |
"(" gpe-component ")" ["*" | "+" | "?"]
label_expr ::= "#" | [A-Za-z0-9%_]+

Example Queries

- select Guide.hotel.name
  from Guide.hotel
  where Guide.hotel.#.pin% = 600020

  data

  - pin code directly under hotel or component of address
- select Guide.hotel.name
  from Guide.hotel
  where Guide.#.(locality|address) = "%Nagar"
  - get the names of hotels in the locality of some Nagar

What is XML?

- XML - Extensible Markup Language
- XML - not a single, predefined markup language
  - A framework for creating mark-up languages
- Domain specific tag names can be introduced
  - extensibility
  - interoperability
  - flexibility
- Data can be made self-describing
  - XML data - data marked up with tags needed for that data conveying what the data means
  - Separation of data and its presentation

Origins of XML

- Standardized General Markup Language (SGML)
  - creating and exchanging complex documents
    - aircraft maintenance, programming language design
  - standardized in 1986 (ISO8879)
  - rich, complex and powerful language
  - difficult to use and implement
- Hypertext Markup Language (HTML)
  - publishing on the WEB - 1989 - Tim Berners Lee
  - derivative of SGML - simple, fixed set of tags
- XML - simplified version of SGML - 1997
  - content and presentation separated

HTML Data

<!--HTML-->
<h2> Address List </h2>
 Kumar, Bangalore
Mohan, Chennai
Geetha, Hyderabad
Kishore, Bangalore

No way to convey
1st component is Name, 2nd component is City
XML: <name>, <city> can be introduced
Produce a list of Bangalore based friends
-- not possible with HTML, possible with XML

XML Data

```
<institute>
  <dept><name>mathematics</name> <address> HSB321</address>
        <phone>8510</phone> <head>Choudam</head>
  </dept>
  <dept><name>cs</name>
        <address><bldg>BSB</bldg>
                 <room>321</room>
         </address>
         <phone>8330</phone>   <phone>8331</phone>
         <head>Raman</head>
  </dept>      …
</institute>
```

ss-data

HTML versus XML

HTML
- designed to display data and to focus on how data looks
- fixed set of tags for providing information to browsers
- difficult to use by programs

XML
- designed to describe data and to focus on what data is
- complementary to HTML
- easy to use for programs

Well-formed XML Documents

- XML document - hierarchical collection of elements
- All XML documents must be well-formed
- Rules for "Well-formed"-ness:
  - There is a unique root element
  - Each element - opens and closes with same tag
  - Elements may not overlap
    - should be properly nested
    - <e1> …<e2> … <\e2> … <\e1> --- ok
    - <e1> …<e2> … <\e1> … <\e2> --- not allowed
  - Attribute values are quoted

Validity

A *valid* XML document:

- Well-formed
- Must have a Document Type Definition (DTD)
- Must comply with the constraints specified in the DTD

Document Type Definition

- a grammar specifying the structure of the document
- what are the elements that are allowed in the document
- what are the attributes of the elements
- how do elements relate to each other
  - containment and sequencing
- DTD's are optional

A Sample XML document

```
<? Xml version = "1.0"?>
<!DOCTYPE emails[
<!ELEMENT emails (email)*>
<!ELEMENT email (from, to, body)>
<!ATTLIST email date CDATA>
<!ELEMENT from (#PCDATA)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT body(#PCDATA)>]>
<emails>
    <email date = "15-12-2000">
        <from>xyz@first.com</from>
        <to>abc@last.com</to>
        <body>Hello, How are you? </body>
    </email>
</emails>
```

DTD

Instance

Info-card DTD

```
<!DOCTYPE AddressInformationCards[
<!ELEMENT infoCards (card*)>
<!ELEMENT card ((person+|company), address+>
<!ELEMENT person(name, emailAddress?, mobilePhone?)>
<!ELEMENT name ( fName, mName?, lName)>
<!ELEMENT address ( streetAddress, locality?, city, pin, phone*)>
<!ELEMENT company(name, contactPerson)
<!ELEMENT contactPerson(name, emailAddress?, mobilePhone?)>
<!ELEMENT emailAddress ( personal+, official+)>
   . . .
```


OEM versus XML

- OEM
  - order among the components of a complex object unimportant
  - data - unordered labeled tree/graph
- XML
  - order among the sub elements of an element conveys important information
  - can also model semi-structured data

Querying XML Data

- XML-QL, QUILT, XQL, etc – several query languages.

  An example using XML-QL: collect all mails to manager

```
<mailsToManager>
    where <email date=$d>
                <from> $s </from>
                <to> manager@tcs.chennai.co.in </to>
                <body> $b </body>
          </email> in "mailsdb.xml"
    construct
          <mail>
                <sentBy> $s </sentBy><date>$d</date>
                <message> $b </message>
          </mail>
</mailsToManager>
```

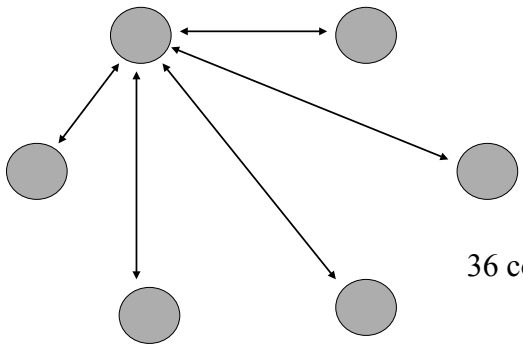

Interesting Issues

- Storage of XML data on the disk
  - Text files
  - Layout on disk blocks
- Index structures for XML data
  - Efficient execution of path expressions
  - Fast evaluation of queries
- New applications
  - Exploit the features of s-s data model
  - Corporate knowledge management

Applications of XML

- Application-independent data exchange
- E-commerce
  - Data exchange / WEB- services
  - Catalogue publishing for agents / personalization
  - Supplying data to integration systems
- Creation of new mark-up languages
  - Molecular dynamics markup language(MoDL),
  - Biopolymer markup language (BIOML),
  - Gene expression markup language GEML),
  - Chemical markup language (CML) , MathML etc

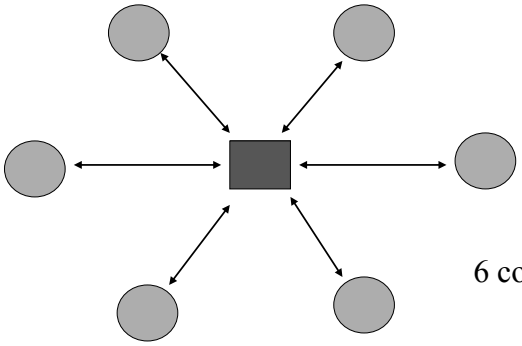
Data Exchange



36 conversions

6 different programs dealing with financial data
using proprietary data formats

Data Exchange Simplified



6 conversions

6 different programs dealing with financial data using common XML data model

References - Text Books

- S Abiteboul, Querying semi-structured data, Intl. Conference on Database Theory (ICDT), 1997.
- S Abiteboul, P Buneman and D Suciu, *Data on the web,(From relations to semistructured data and XML),* Morgan Kaufmann, 2000.
- S Abiteboul et al, The Lorel query Language for semi-strucured data, Intl. Jl. of Digital Libraries, 1(1):68-88, 1997.
- The WWW Consortium(W3C) XML web page, http:// www.w3c.org/XML/.
- Charles F Goldfarb, Paul Prescod, *The XML Handbook*, Addison-Wesley, 1999.