Querying the Web

Daniela Florescu
INRIA

Juliana Freire
Database Systems Research
Bell Labs - Lucent Technologies
The Web - some history

- 1989 First Web browser
- 1993 Mosaic is released, there are 50 Web sites
- 1994 First search engines (WWW, WebCrawler)
- 1996 US$1 billion spent in Internet shopping, users in 150 countries
- 1997 1 million Web sites
- 1998 300 thousand servers world-wide
Data on the Web

- What is the Web today?
  - HTML documents intended for human consumption
  - easy to fetch any Web page, from any server and platform
- Everything is on the Web - over 1 billion pages
  - news, hotel information, airfares, groceries, movie times, location of famous people’s burial places, ...
- How to find the information you need?
  - browsing: time consuming, easy to get lost
  - query!
Querying the Web: The Evolution

- **Search engines** were the 1st tools available: indexes of Web pages are built that allow for "fast" keyword search
  - limited ad hoc querying - limited query interface

- **Web languages** allow more “structured” queries
  - integrate navigation and querying - improve results by providing more info to guide the search;
  - document query languages (e.g., XSLT, XMLQL)

- **Webbases**: the Web as a database
  - a unified view of information from diverse sources (e.g., comparison shopping, portals, mediators)
Tutorial Goals

- Expose attendees to key concepts and technologies for finding, querying and integrating information on the Web
- Identify technical challenges in designing systems that support Web queries
- Survey recent literature, and discuss interesting research directions
Tutorial Outline

- Introduction
- Search engines
- Web languages
  - from WebSQL to XML query languages
- Integrating Web information: querying the Web as a database
- Concluding remarks and future directions
Roadmap

- Search engines: brief overview
- Web languages
- Webbases
Search engines

- Manually built indexes (e.g., yahoo.com)
  - browse through site hierarchy

- Automated search engines (e.g., google.com)
  - Web robots traverse the Web hypertext structure and retrieves all documents that are referenced
  - retrieved documents are parsed and indexed (inverted-list)
  - keyword-based search

  *find all documents that contain string "XML"*
Search Engines: Evolution

- **1994** WWW - the World Wide Web Worm indexed 110k pages
- **1997** top engines claimed to index 100 million pages
- **2000** over 500 million pages, and over 300 search engines!

GG=Google, WT=WebTop.com, AV=AltaVista, FAST=FAST, NL=Northern Light, EX=Excite, INK=Inktomi, Go=Go (Infoseek).
Size and Coverage

**Millions of Web Pages Indexed**

<table>
<thead>
<tr>
<th>Provider</th>
<th>Pages Indexed</th>
</tr>
</thead>
<tbody>
<tr>
<td>GG</td>
<td>1060</td>
</tr>
<tr>
<td>WT</td>
<td>500</td>
</tr>
<tr>
<td>AV</td>
<td>350</td>
</tr>
<tr>
<td>FAST</td>
<td>340</td>
</tr>
<tr>
<td>NL</td>
<td>265</td>
</tr>
<tr>
<td>EX</td>
<td>250</td>
</tr>
<tr>
<td>INK</td>
<td>110</td>
</tr>
<tr>
<td>Go</td>
<td>50</td>
</tr>
</tbody>
</table>

**% Of Web Indexed**

<table>
<thead>
<tr>
<th>Provider</th>
<th>Indexed %</th>
</tr>
</thead>
<tbody>
<tr>
<td>GG</td>
<td>56%</td>
</tr>
<tr>
<td>WT</td>
<td>50%</td>
</tr>
<tr>
<td>AV</td>
<td>50%</td>
</tr>
<tr>
<td>FAST</td>
<td>35%</td>
</tr>
<tr>
<td>NL</td>
<td>34%</td>
</tr>
<tr>
<td>INK</td>
<td>27%</td>
</tr>
<tr>
<td>EX</td>
<td>25%</td>
</tr>
<tr>
<td>Go</td>
<td>5%</td>
</tr>
</tbody>
</table>

GG=Google, WT=WebTop.com, AV=AltaVista, FAST=FAST, NL=Northern Light, EX=Excite, INK=Inktomi, Go=Go (Infoseek).
Research Issues

- Fast crawling needed to gather information and keep it up to date
- Efficient use of storage to store indexes and the documents themselves
- Efficient index system capable of processing hundreds of gigabytes of data
- Efficiently handle hundreds to thousands of queries per second search
- Quality of results: too many irrelevant answers
Improving Quality of Results

- Google: page ranking based on backlinks [Brin-WWW7]
- Duplicate removal [SIFT, Shivakumar-WebDB'98]
- Meta-search: combine and processing of results of various search (e.g., to filter out bad answers) [search.com]
- Collaborative engines (e.g., www.hotbot.com)
- Directories, webrings, domain specific, etc
- Focused crawling [Chakrabarti et al-WWW8]
Meta-Search: Search.com

- Findwhat
- Direct Hit
- goto.com
- WebCrawler

Web Pages
- MobileNow - Discount Laptop Batteries - Discount prices on laptop batteries for 1000's of models! Come visit our showroom for batteries, carry cases, DFC charging cords, desktop changers and much more.. http://www.mobilenow.com
- YearEnd on the Toshiba Satellite 1100CS Laptop - Toshiba's http://www.yearend.com
- AOCNEWS.com - Overstock Orders Doctrine Windows Defend Day - Feb. 3. Sherry with a simple moral. Always read the fine print. In February 1998 an Australian systems manager named Geoffrey Bennett, after buying a Toshiba laptop with Windows 95 installed, noted an interesting clause in Microsoft's standard End...
- Toshiba/Microsoft Page - Hope that this web page will prove useful to those people who want to purchase a laptop without Microsoft Windows. The short summary is this: it is near impossible to buy a laptop without Windows http://www.toshiba.com/specs/toshiba.html
- Find Toshiba Laptops Here - True Data Products is your best online computer wholesaler for all the most popular laptops, notebooks and desktop computers! http://www.true-data.com
- Receive Laptop Computer Info Via Email - You can receive all the latest info on laptop computers, and other related topics, by visiting...
HotBot

Plug and Play.
Find great prices on video games, electronics, home theater and portable audio. Bid Now!

Results for: cook
Look for: the person

PEOPLE WHO DID THIS SEARCH ALSO SEARCHED FOR
Cooking Chef Recipes Bake Cookbook Glen Cook Ingredients Roger zelazny

SBBD 2000 - Querying the Web  D. Florescu, J. Freire
Semantic Web

- Add metadata about properties and relationships of items on the Web.
- RDF (Resource Description Format)
  - Declarative language that provides a standard way for using XML to represent metadata in the form of statements about properties and relationships of items on the Web.
- OIL (Ontology Interchange Language)
  - Standard for specifying and exchanging ontologies.
Roadmap

- **Search engines**: brief overview
- **Web languages**
  - First Web languages: WebSQL, W3QL, WebLog
  - XML
  - Some XML query languages
- **Webbases**
Web Languages

- Search engines have a very limited query interface
  - only keyword-based queries are allowed
  - flat queries over content of individual documents
  - not expressive enough
- Need more “expressive” queries
  - besides content, query Web topology and document structure
First Web Languages

- Use topology of the Web in queries to control navigation and get better answers: combine browsing and searching
  - WebSQL, W3QL

- Query the document contents taking structure into account, and build new documents
  - WebLog
WebSQL

- Exploit the structure and topology of the document network
- Clear semantics based on a virtual graph model
- Relational view of the Web:
  \[ \text{Document}(\text{url}, \text{title}, \text{text}, \text{type}, \text{length}, \text{modif}) \]
- A query: *find all HTML documents about XML*
  
  ```sql
  SELECT d.url, d.title
  FROM Document d
  SUCH THAT d MENTIONS "XML"
  WHERE d.type="text,html"
  ```
WebSQL (cont.)

- Introduced the notion of cost (local vs remote)
- Another query: *starting from the W3C home page, find all documents with XML in the title that linked through paths of length 2 or less containing only local links*

```sql
SELECT d.url, d.title
FROM Document d SUCH THAT
    "http://w3c.org"=-->|-->.-.->d
WHERE d.title CONTAINS "XML"
```
Other languages

- **W3QL**
  - similar to WebSQL, but the focus is on interoperability - use together with other tools

- **WebLog**
  - model contents of document based on HTML structure (assumes knowledge about structure)
  - logic-based language for querying and restructuring information
Some WebLog queries

% Find all DBLP pages that refer to XML

dblp_pages(http://www.informatik..../~ley/db).

dblp_pages(U) :- dblp_pages(V), V[hlink->L],
    href(L,U),

xml_pages(U) :- dblp_pages(U), U[occurs->I],syn(I,'XML').

% Collect all links to HTML documents, and the corresponding document title

ans.html[title->'all citations', hlink->>L, occurs->>>T] :-
    dblp_pages[hlkink->>>L], href(L,U), U[title->T].
**HTML vs. XML**

- HTML is only for presentation
  
  ```html
  <H2> The Advanced Html Companion</H2>
  US$35.96
  <UL>
    <LI> Keith Schengili-Roberts
    <LI> Kim Silk-Copeland
  </UL>
  ```

- XML can describe the information content

  ```xml
  <Book>
    <Title>The Advanced Html Companion</Title>
    <Author> Keith Schengili-Roberts </Author>
    <Author> Kim Silk-Copeland</Author>
    <Price> 35.96</Price>
  </Book>
  ```
The secret of HTML success

◆ Everybody can write it:
  > HTML is simple
  > HTML is textual: it is human readable, you can use any editor, ...

◆ Everybody can read it
  > HTML is Portable on any platform
  > The browser is the universal application

◆ It connects pieces of information together
  > Through hypertext links
But new applications = new needs

- **Infomediaries:**
  - Search engines
  - Web portals
  - Digital libraries
  - Virtual enterprises

- **Electronic commerce:**
  - On-line catalogs and procurement
  - Comparison shoppers
  - Market places

- **Scientific applications**
- **Manufacturing engineering**
  etc.

More than HTML: data on the Web
More than the browser: applications on the Web
The secret of XML popularity

It looks like HTML...
- Simple, familiar, easy to learn, human-readable
- Universal and portable
- Supported by the W3C: trusted and quickly adopted by the industry

...but it’s more than HTML!
- flexible: you can represent any information
- extensible: you can represent it the way you want!
But XML is just the beginning...

- We now want to build applications
  - There is an urgent need for XML tools

- Designing XML tools is a data management problem:
  - XML 1.0 to describe structured documents
    = Syntax for trees
  - XML data models to describe the information content
    = Data model for trees
  - XML schemas to describe the structure of information
    = Data definition language for trees
  - XML languages to describe information processing
    = Data manipulation language for trees
XML 1.0: Well formed documents

- An XML Document is composed of:
  - markup: element, attributes
  - text: `#PCDATA, CDATA`

- Well-formed document:
  - verifies XML lexical conventions
  - contains properly nested elements with a single root element
  - can contain empty elements, mixed text and elements

```xml
<book year="1967">
  <title>The politics of experience</title>
  <author>R.D. Laing</author>
  <ref isbn="1341-1444-555"/>
  <section>
    The great and true Amphibian, whose nature is disposed to.....
    <title>Persons and experience</title> Even facts become...
  </section> ...
</book>
```
XML 1.0: Valid documents

- A valid XML document verifies a Document Type Definition (DTD):
  - grammar for the document
  - constraints on the structure of elements, attributes, entities, notations...
  - a DTD is optional

```xml
<?xml version="1.0"?>
<DOCTYPE book [  
  <!ELEMENT book (title, author*, publisher?, section+)>
  <!ATTLIST book year CDATA #IMPLIED>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT author (#PCDATA)>
  <!ELEMENT section (#PCDATA | title | section)>
]> ...
```
<?xml version="1.0"?>
<bib>

<book ISBN="10" year="1967">
  <title>The politics of experience</title>
  <author><firstname>R.D.</firstname><lastname>Laing</lastname></author>
  <publisher>Pantheon Books</publisher>
  <section id="1">
    <title>Persons and experience</title>
    The great and true Amphibian,
    <!-- This is a comment about Amphibians -->
    whose nature is disposed to.....
    <section id="1.1">
      Exploitation must not be seen....
    </section>
  </section>
</book>

</bib>
In search of a query language...

- What do we call a query language?

The language used to describe, in a declarative fashion, the mapping between an input instance of the data model to an output instance of the data model.

What data model for XML?
XML vs. graph-based models

- XML document content could be modeled as a graph
  - entities (e.g. elements, attributes) organized in an hierarchical structure

- ...but XML is more complicated than that
  - several distinct types of nodes
    » text, elements, attributes, comments, processing instructions, etc.
  - some parts are ordered (e.g. children of an element) and some other parts not ordered (e.g. attributes)
  - in the absence of a DTD or schema, the document is a tree; otherwise it could be a graph
Some relevant query languages

- **Research** query languages for XML
  - e.g. XML-QL, Lorel, XML-GL, Quilt, Xduce
- **Industry** query languages for XML
  - e.g. XQL, OQL extensions to query SGML documents
- **Standard** processing languages for XML (W3C standards)
  - e.g. Xpath, XSLT
### Th big picture

<table>
<thead>
<tr>
<th>Data model</th>
<th>Simple graphs</th>
<th>Idealized XML data model</th>
<th>Real XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navigation &amp; selection</td>
<td>SPJ + RegExp</td>
<td>SPJ + RegExp + grouping.</td>
<td>Xpath(5)</td>
</tr>
<tr>
<td>SPJ + RegExp + grouping.</td>
<td>XML-QL (2)</td>
<td>Lorel (3)</td>
<td>XSLT (7)</td>
</tr>
<tr>
<td>OQL + RegExp</td>
<td>YATL (4)</td>
<td></td>
<td>Quilt (6)</td>
</tr>
<tr>
<td>OQL + conditional + full recursion</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SBBD 2000 - Querying the Web**

D. Florescu, J. Freire
XML-QL (Deutsch et al)

- Data model:
  - node and edge labeled graph (elements & attributes)
  - a (totally) ordered or a (totally) unordered graph

- Language description:
  - WHERE clause to bind variables and to test predicates
  - CONSTRUCT clause to create new XML structures

- Features:
  - XML patterns for both the WHERE clause and the CONSTRUCT clause
  - regular expressions for navigation
  - joins on multiple input sources
  - Skolem functions to create nested structures
XML patterns

Query1: “Retrieve the titles of the books written by Laing before 1967”

WHERE

```xml
  <bib>  <book year= $y ISBN= $isbn>
      <title>$t</title>
      <author> <lastname>Laing</lastlname> </author>
  </book>
</bib> in “bib.xml”,
$y<1967
```

CONSTRUCT

```xml
  <resultBook ISBN= $isbn >
      <resultTitle>$t</resultTitle>
  </resultBook>
</resultBook>
```

<table>
<thead>
<tr>
<th>$y</th>
<th>$isbn</th>
<th>$t</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Joins in XML-QL

Query2: “Retrieve all the reviews about books written by Laing”.

WHERE

\[
\begin{align*}
&\langle \text{bib}\rangle \langle \text{book} \rangle \text{ ISBN} = \$i \\
&\quad \langle \text{author} \\
&\quad \quad \langle \text{lastName} \rangle \text{Laing} \langle /lastname \rangle \\
&\quad \langle /author \rangle \\
&\quad \langle /book \rangle \langle /bib \rangle \text{ in “www-caravel.inria.fr:bib.xml”,} \\
&\langle \text{reviews} \\
&\quad \langle \text{review} \text{ ISBN} = \$i \rangle \langle /review \rangle \text{ ELEMENT_AS } \$e \\
&\quad \langle /reviews \rangle \text{ in “www.crossgain.com:reviews.xml”} \\
&\text{CONSTRUCT} \\
&\quad \$e
\end{align*}
\]
Meta-data queries

Query3: “Which kind of elements can be found in the content of the element corresponding to the book with isbn=10?"

WHERE

```
<bib>
  <book ISBN="10"> <$tagName> </> </book>
</bib> in “bib.xml”,
```

CONSTRUCT

```
<result>$tagName <result>
```
Fusion using Skolem functions

- Query4: “Retrieve the titles of the all the books, grouped first by year and then by publisher”.

WHERE

```xml
  <bib><book year=$y>
    <title> $t </title>
    <publisher>$p/publisher>
  </book><bib>
```

CONSTRUCT

```xml
  <bookPerYear id=F1($y)>
    <bookPerYear&Publisher id=F2($y,$p)>
      <bookTitle> $t </bookTitle>
    </bookPerYear&Publisher>
  </bookPerYear>
```

Automatic fusion of all the bookPerYear elements with the same id attribute

$y | $p | $t
Lorel (Abiteboul et al)

- Semi-structured data (OEM), reconverted to XML
- Lorel is an extension of OQL for OEM:
  - functional language
  - applies type coercion (relaxes the strong typing constraint of OQL)
  - uses path expressions with full regular expressions
  - adds an XML element creation operator
  - adds Skolem functions for grouping
- Query1: “Retrieve the books written by Laing before 1967.”
  ```sql
  SELECT xml(result: $b )
  FROM $b in bib.book
  WHERE $b.author.lastname?=“Laing” and $b.@year<1967
  ```
- Different syntax, but equivalent to XML-QL in expressive power.
**YATL**

- **Initial goal:** data conversion and integration
- **Data model:** ordered trees, references, node-labeled
- **Language description:**
  - like OQL & Lorel: functional language
  - like others: database iterator (make...match...where)
  - like others: Skolem functions to manipulate references
  - pattern matching with horizontal regular expressions
  - local functions with full recursive functions for conversions
- **Implementation:** v1 INRIA in 1998 & v2 Bell Labs in 2000
Tree patterns in YatL

Query1: “Retrieve the titles of the books published in 1967 by ‘Pantheon Books’.

MAKE result [ $t ]
MATCH  « bib.xml »  WITH  book[ @year[$y],
title[$t],
publisher[$p] ]
WHERE $p = “Pantheon Books” and $y=1967
Research query languages for graph-based data and XML

- **New features:**
  - node fusion via Skolem functions
  - type coercion
  - vertical regular expression
  - horizontal regular expression
  - meta-data queries
  - recursive functions

- **Data model:** “idealization” of a real XML document
  - do not model some XML features such as comments, processing instructions, namespaces, etc
  - schema agnostic

- **Not complete for “real XML”**
## Overview

### Data model

<table>
<thead>
<tr>
<th>Real XML</th>
<th>Idealized XML data model</th>
<th>Simple graphs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xpath(5)</td>
<td>XML-QL (2)</td>
<td>UnQL (1)</td>
</tr>
<tr>
<td></td>
<td>Lorel (3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>YATL (4)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XSLT (7)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Quilt (6)</td>
<td></td>
</tr>
</tbody>
</table>

### Expressive power

- SPJ + RegExp
- SPJ + RegExpr + grouping.
- OQL + RegExpr
- OQL + conditional + full recursion

---

**SBBD 2000 - Querying the Web**

D. Florescu, J. Freire
XPath(1)

- Syntax for XML document navigation and node selection
- Papers:
  - “XML Path Language (Xpath)”, W3C recommendation
- Building block for other W3C standards:
  - XSL Transformations (XSLT): transform XML→XML
  - XML Link (XLink): constructs that support describing the links between addressed information
  - XML Pointer (XPointer): constructs that support addressing into the internal structural of XML
**XPath(2)**

- A query is an expression (Location Path)
  - describes a single navigation path in an XML document
- A query simply selects a list of nodes from the input document
- A Location Path consists of:
  - a context node
  - a series of Location Steps separated by /
- A verbose Location Step consists of:
  - an axis, a node test, a list of predicates

```
  descendant::section / [position()=1]
```
XPath in action (3)

- **Location step:**
  - an axis, a node test, a list of predicates

- **13 Axes:**
  - ancestor, ancestor-or-self, attribute, child, descendant, descendant-or-self, following, following-sibling, namespace, parent, preceding, preceding-sibling, self

- **Node Test:**
  - name test (e.g. section, *, myNs:myTag)
  - type test (e.g. text(), comment(), node())

```xml
```
XPath abbreviated syntax

<table>
<thead>
<tr>
<th>XPath Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>book</td>
<td>CN/child::book</td>
</tr>
<tr>
<td>section[1]</td>
<td>CN/child::section[position()=1]</td>
</tr>
<tr>
<td>.</td>
<td>CN</td>
</tr>
<tr>
<td>..</td>
<td>CN/parent::*</td>
</tr>
<tr>
<td>../../../text()</td>
<td>CN/parent::*/child::text()</td>
</tr>
<tr>
<td>//section</td>
<td>ROOT/descendent-or-self::section</td>
</tr>
<tr>
<td>/section</td>
<td>ROOT/child::section</td>
</tr>
<tr>
<td>//</td>
<td>ROOT/descendent-or-self::*</td>
</tr>
<tr>
<td>//section[last()]</td>
<td>ROOT/descendent-or-self::section[position()=last()]</td>
</tr>
<tr>
<td>//section[5] [title=&quot;introduction&quot;]</td>
<td></td>
</tr>
<tr>
<td>//section [title=&quot;introduction&quot;] [5]</td>
<td></td>
</tr>
</tbody>
</table>
Quilt (Chamberlin et al)

- Borrows features from OQL, XML-QL, LoreL, XQL, ML
- Design goals:
  - learn from previous experience
  - keep it simple
  - make sure it is useful
  - make sure it is semantically clean :)
- Limitations and problems:
  - no fusion operation
  - no support for full regular expressions
  - semantic problems with Xpath
  - many other open issues…
Quilt implementations

- Implementations:
  - Agora (INRIA)
  - Univ. Washington
  - Univ. Pennsylvania
  - Niagara project (Univ. Wisconsin)
XSLT(1)

- Paper:
  - “XSL Transformations (XSLT)”, W3C recommendation
- XML to XML rule based transformation language
- An XSLT program is an XML document itself
An XSLT program is a valid XML document containing:
- elements in the `<xsl:>` namespace (i.e. the XSLT statements)
- elements in other namespaces (i.e. the user-defined data)

The result of the evaluation of an XSLT program on an input XML document := the XSLT document where each `<xsl:` element has been replaced with the result of its "evaluation"

- Uses XPath as a sublanguage
- Used mostly as a stylesheet language
XML Query Language Working Group at the W3C

- Technical documents:
  - XML Query Requirements
  - XML Query Data Model

- Several use cases:
  - SQL-like queries, queries that use references, queries that exploit the hierarchy and the sequence, SGML-like queries, queries that need text operations and namespace handling, metadata querying, recursive queries, etc.

- Current work: XML algebra

- Promised release date: November 2000
XML query language requirements

1. **Select** portions of a document
   - all

2. **Copy** portions of a document while preserving the hierarchy and the order of the nodes
   - YaTL, Quilt, XSLT

3. **Combine** (join) two documents
   - all except Xpath and XSLT (only intra-document joins)

4. **Construct** new documents
   - all except Xpath

5. **Navigate** irregular or unknown documents
   - full (vertical) regular expressions: Lorel, XML-QL
   - simple (vertical) navigation: Xpath, XSLT, Quilt
   - horizontal regular expressions: YaTL
XML query language requirements (2)

6. Formulate predicates on the tag names and attribute names
   - all

7. Query and preserve the nodes global topological order
   - Quilt

8. Apply aggregation and sorting functions
   - all except Xpath, XML-QL

9. Apply existential and universal quantifiers
   - Lorel, Quilt, XSLT (not explicitly!)

10. Apply full-text predicates and text operations
    - none satisfactorily
Open questions

◆ General questions:
  - Which programming style? (most powerful and most natural)
  - Which type of syntax? (beauty contest...)

◆ Unclear technical issues:
  - Should the query semantics depend on the presence of the schema?
  - Automatic type coercion?
  - Type inference and type checking?
  - Recursive functions?
  - Full regular expressions?
  - Fusion operator?
  - Copy semantics or identity-based semantics?
  - Copy by reacheability?
  - Dereferencing operator semantics?
  - Extensibility: protocol and exception handling?

Your opinion IS IMPORTANT!
Roadmap

- **Search engines**: brief overview
- **Web languages**
- **Webbases**
  - querying the Web as a database
  - combine/relate information from multiple Web sources
The Web is a great source of information

But...

- The are too many sites
- Information is spread and disorganized
- Sites are becoming increasingly complex to navigate (frames, multiple forms, etc)
- It is hard to relate information from multiple sites
Querying the Web

- Find documents that contain keyword “XML”
- Starting from http://w3c.org, and within this domain, find a document with keyword “XML”
- Find all authors that published papers with “XML” in the title, that have also published a book which costs more than US$50

<table>
<thead>
<tr>
<th>Data model</th>
<th>Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search engines</td>
<td>URL, keywords</td>
</tr>
<tr>
<td>Web languages</td>
<td>URL, doc title/length, keywords</td>
</tr>
<tr>
<td>Webbases</td>
<td>Site schemas descriptions</td>
</tr>
</tbody>
</table>
The Web is not a Database

query

? No schema!

Web

HTML documents

Text documents

Form interfaces

XML documents
Webbases: Organizing the Web

- An easy-to-use place to find and organize the various types of unconnected data scattered across the Web, e.g., autoweb.com, carpoint.com, pricescan.com, mynetscape.com
Complex Web queries

Make a list of used BMWs 3-series advertised in the tri-state area such that each car is 1996 or later model, whose price is less than its Blue Book value.

Make a list with prices of Toshiba laptops with at least 64MB of RAM, and processor with at least 600Mhz, and whose price is less than US$2000.
Webbases: Applications

- Portals
- Comparison shopping
- E-commerce
- Virtual enterprises
  
  ...and many more!
A look at PriceScan.com

Available at buydig.com

Computer > Hardware > Notebook Systems > PC Compatible

Manufacturer: Toshiba
CPU Type: No preference
CPU Speed: at least 600
RAM: at least 64
Storage Capacity: at least No preference
Modem: No preference
Display Type: No preference
Display Size: at least No preference
Maximum Price: 2000

Search Clear
## Laptops at PriceScan.com

<table>
<thead>
<tr>
<th>Product Description</th>
<th>Best Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toshiba Satellite 2250XCDS C/600/64/6.0GB/13.0/CD/98</td>
<td>$1,161.00</td>
</tr>
<tr>
<td>Toshiba Satellite 1695CDT C/600/64/6.0GB/12.1/CD/98</td>
<td>$1,399.00</td>
</tr>
<tr>
<td>Toshiba Satellite Pro PS432U-QL1560 P3/600/12.1TFT/98</td>
<td>$1,681.00</td>
</tr>
<tr>
<td>Toshiba Satellite 2755DVD P3/600/64/6.0GB/12.1/DVD/96</td>
<td>$1,699.00</td>
</tr>
<tr>
<td>Toshiba Satellite Pro PS432U-QL1560 P3/600/12.1TFT/2K</td>
<td>$1,775.00</td>
</tr>
<tr>
<td>Toshiba Satellite 2775XVDVD P3/650/64/12.0GB/14.1/DVD/98</td>
<td>$1,970.00</td>
</tr>
</tbody>
</table>

### Toshiba Satellite 2250XCDS C/600/64/6.0GB/13.0/CD/98
- Mfg Part No.: PS225U-M91J08
- Celeron, 600MHz, 64MB RAM, 6.0GB HD, CD-ROM Drive, Modem, 13.0 Active Matrix Display

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Source</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dartek</td>
<td>Direct from Vendor 09/10/00</td>
<td>$1,160.58</td>
</tr>
<tr>
<td>Firstsource</td>
<td>Direct from Vendor 09/10/00</td>
<td>$1,187.43</td>
</tr>
<tr>
<td>PC Zone</td>
<td>Direct from Vendor 09/18/00</td>
<td>$1,189.98</td>
</tr>
<tr>
<td><a href="#">Computers4Sure</a></td>
<td>Direct from Vendor 09/18/00</td>
<td>$1,199.00</td>
</tr>
<tr>
<td>PC Mall</td>
<td>Direct from Vendor 09/18/00</td>
<td>$1,199.00</td>
</tr>
</tbody>
</table>

---

SBBD 2000 - Querying the Web  
D. Florescu, J. Freire  
65
Webbases

Database systems designed for casual Web users to query integrated Web content.
- Provide easy/convenient access to information in the Web
- Allow complex queries over a multitude of Web sources
Challenges in WebBase Design

- Requirements from a user's perspective:
  - Logical independence: naive users should be able to easily formulate ad-hoc queries
  - Site independence: users should not be required to locate related Web sites and resolve vocabulary and presentation differences in the sites
  - Navigation independence: navigational details of querying and retrieving dynamic content from a Web site must be completely transparent to the user

- From a designers perspective:
  - Sites are autonomous
Webbases vs Mediators

- Much larger scale, and higher degree of autonomy
- Web is dynamic - Web sites change constantly, and new sites are added often
- Little metadata about characteristics of sources
- Users come from a wide variety of backgrounds - need flexible query interfaces
- Standard protocol and languages
WebBase vs Databases

**Traditional DB**

- **External Schema**
  - SQL, QBE, ...
- **Logical Schema**
  - Relational algebra
- **Physical Schema**
  - Low-level access methods

**Physical DB**

**WebBase**

- **External Schema**
- **Logical Schema**
- **Virtual Physical Schema**
  - High-level algebraic access methods
  - Navigation, data extraction

**Raw Web**

**Logical independence**

**Site independence**

**Navigation independence**
Building a WebBase

- Find information sources
- Wrap information sources
  - extract “schema”
  - access + extraction
  - Web models
- Integrate sources
  - schema + semantic integration
- Query processing
Wrapping information sources

- Access
- Extract information
- Describe information

How do you build wrappers?
Accessing information

- Web uses standard protocol (HTTP)
- Hidden Web: 80% of all data on the Web is published through forms-based interfaces (e.g., product catalogs and searchable classified ads)
Example: Accessing Travelocity

Steps to retrieve airfare info:
- Go to travelocity.com
- Choose Find/Book a Flight
- Enter login information
- Choose the 9 Best Itineraries
- Specify details of Itineraries
- View the results
Automating Web Navigation

- Write programs: Java, WebL (H. Marais, WWW7), etc.
  - need programming expertise, and maintenance
  - not very scalable

- WebVCR (Freire et al, WWW9)

- Mapping by example (Davulcu et al, SIGMOD’99)
  - semi-automatic discovery of “site maps” - topology + contents and capabilities of Web sites, no programming required
  - access scripts are automatically generated
The WebVCR

- Transparently track user actions:
  - ease-of-use: targeted to casual users
  - mimic browsing experience
- Record navigation sequences in *smart bookmarks*
  - links followed
  - forms filled out, information entered
  - buttons clicked
- Playback
  - using recorded information, follow links, fill forms, ... to get to desired content
  - *correctly* replaying a smart bookmark is challenging as pages may change between record and replay time
Recording: Step-by-Step

Start applet and press “Record”

Press “Stop”, save to a file
A navigation map is a labeled directed graph, where nodes represent Web pages and edges represent navigation (browsing) actions.
Extracting information

Web is (mostly) unstructured data: HTML

2. **The Advanced Html Companion**  
   Our Price: $35.96  
   You Save: $8.99 (20%)  
   Usually ships in 24 hours  
   Average Customer Review: ★★★★★

3. **Applied XML Solutions (Sams Professional Publishing)**  
   by Benoit Marchal. Paperback (August 29, 2000)  
   Our Price: $35.99  
   You Save: $9.00 (20%)  
   Usually ships in 24 hours  
   Average Customer Review: ★★★★★

4. **Applied XML: A Toolkit for Programmers**  
   by Alex Cepenkus, Faraz Hoodbhoy. Paperback (July 1, 1999)  
   Our Price: $39.99  
   You Save: $10.00 (20%)  
   Usually ships in 24 hours  
   Average Customer Review: ★★★★★

Title: The Advanced Html Companion  
Authors: Keith Schengili-Roberts, Kim Silk-Copeland  
Price: 35.96
Extracting structure from Web pages

- Write specialized parsers: hard to create and maintain - not scalable

- Parsing by example:
  - NoDoSe (B. Adelberg - SIGMOD’98) - interactive tool for semi-automatically determining the structure of text documents
  - Ariadne (C. Knoblock et al - AAAI’97): demonstration oriented user interface
Extracting structure from Web pages (cont.)

- **W4F** (Sahuguet & Azavant - VLDB'99)
  - wizard-based wrapper generation toolkit for HTML documents
  - HEL is a language to express extraction rules (DOM + flow + regular expressions)
  - Users must write HEL rules

- **XML**
  - self-describing data - simplifies extraction
  - extraction can be expressed as a query
XML: Parsing is easier...

2. **The Advanced Html Companion**  
   Our Price: $35.96  
   You Save: $8.99 (20%)  
   Usually ships in 24 hours  
   Average Customer Review: ★★★★★

3. **Applied XML Solutions (Sams Professional Publishing)**  
   by Bencit Marchal. Paperback (August 29, 2000)  
   Our Price: $35.99  
   You Save: $9.00 (20%)  
   Usually ships in 24 hours  
   Average Customer Review: ★★★★★

4. **Applied XML: A Toolkit for Programmers**  
   by Bencit Marchal. Paperback (August 29, 2000)  
   Our Price: $35.99  
   You Save: $9.00 (20%)  
   Usually ships in 24 hours  
   Average Customer Review: ★★★★★

```xml
<Book>
    <Title>The Advanced Html Companion</Title>
    <Author>Keith Schengili-Roberts</Author>
    <Author>Kim Silk-Copeland</Author>
    <Price>35.96</Price>
</Book>
```
Issues

- Automation and simplicity are key
- What if the documents change?
  - Need to make wrappers robust
  - W4F relies on wrapper creator
  - WebVCR uses heuristics for robust access
  - Resilient extraction expressions (Davulcu et al - PODS 2000)
  - Robust locations (Phelps and Wilenski - WWW9)
- Automatic generation of extraction expressions
Modeling the Web

- How to describe Web information sources?

- Information:
  - contents (for example, movies)
  - attributes found in the source (genre, cast)
  - constraints on contents (only Brazilian movies)
  - completeness and reliability

- Access
  - query processing capabilities
  - how to navigate
Different Models

- Graph-based data models to represent semistructured data (e.g., Lorel)
- Information Manifold (Levy et al):
  - source description language: users may describe complex constraints on contents, source capabilities
  - e.g., Source 1: Japanese cars beginning 1990,
    \[ v(model, year, price, owner) :- \\
    Car(model, year, price, owner), \\
    JapaneseCar(model), year \geq 1990 \]
Different Models (cont.)

- ADM - Araneus data model (Mecca et al):
  - ODMG-like model for describing Web hypertexts
  - defines a page-based “scheme”: Web is a collection of unstructured pages connected by links
  - a page is a URL-identified nested object, whose attributes that correspond to the relevant pieces of information in the page
  - similar pages are grouped into page-schemes
Different Models (cont.)

- Navigation maps (Davulcu et al)
  - models the structure of a Web site and how to retrieve the information in the site
  - created by example: capture designer's actions while he/she browses, and extracts useful information from visited Web pages
  - meta-data: obligatory vs optional attributes, domains of certain attributes
  - navigation expressions automatically generated
  - hidden (dynamic) Web
Navigation Maps and Web Objects

Web structures in a navigation map can be represented as F-logic objects:

\[
\text{web\_page[}
\begin{align*}
\text{address} & \rightarrow \text{url;} \\
\text{title} & \rightarrow \text{string;} \\
\text{contents} & \rightarrow \text{string;} \\
\text{actions} & \rightarrow \{\text{action}\}
\end{align*}
\]

\textit{Declaration of Web page class}

\textit{URL of page}

\textit{Title of the page}

\textit{HTML contents of the page}

\textit{Actions in a page}

\[
\text{action[}
\begin{align*}
\text{object} & \rightarrow \{\text{link,form}\}; \\
\text{link} & \\
\text{source} & \rightarrow \text{url;} \\
\text{targets} & \rightarrow \text{web\_page;} \\
\text{doit@attrValPair} & \rightarrow \text{web\_page}
\end{align*}
\]

\textit{Declaration of action class}

\textit{Action can apply to a form or link}

\textit{Page where the action belongs}

\textit{Pages action may lead to}

\textit{Method to execute the action}
Web Objects (cont.)

link[
  name → string;
  address → url];

Declaration of link class
Name of link
URL of link

form[
  cgi → url;
  method → meth;
  mandatory → attribute;
  optional → attribute;
  state → attrValPair]

Declaration of form class
Script associated with form
CGI invocation method
Mandatory attributes
Optional attributes
Set of attribute-value pairs
An action object

submit_form:action[object->form1[
  cgi → " /cgi-bin/nclassyNDD.x/"
  method → "post"
  mandatory → {make};
  optional → {}]
  source → www.newsday.com;
  targets → →{CarFeatures, CarData};
  doit@attrValPair → web_page]
A used-car classified ads site

- Newsday
  - link(11)
  - link(auto)
  - link(12)
  - link(13)

- Dealer Cars
- Used Cars
- Collectible Cars
- Sport Utility

- Car features
  - form f1(make)
  - form f1(make)
  - form f2(model, year)

- Car Data
  - extract_table(t1)

- Cars(make, model, year, price)
Modeling Navigation Processes

- Navigation processes
  - algebraic expressions
  - map dynamic Web data onto virtual relations

\[
\text{newsday\_used\_cars}(\text{Make}, \text{Model}, \text{Price}, \text{Contact}) \leftarrow \\
\text{newsdayPg}.\text{actions}:\text{follow\_link}[@\text{object} \rightarrow \text{link(auto)}; \text{doit@()} \rightarrow \\
\quad \text{UsedCarPg}] \otimes \\
\quad \text{UsedCarPg}.\text{actions}:\text{submit\_form}@[@\text{object} \rightarrow \text{form(f1)}; \text{doit@}(\text{Make}) \\
\quad \quad \rightarrow \text{CarPg1}] \otimes \\
\quad (\text{CarPg1}:\text{data\_page}[@\text{extract} \rightarrow \text{tuple(\text{Contact},\text{Price})}] \lor \\
\quad (\text{CarPg1}.\text{actions}:\text{submit\_form}@[@\text{object} \rightarrow \text{form(f2)}; \\
\quad \quad \quad \text{doit@}(\text{Model}) \rightarrow \text{CarPg2}] \otimes \text{CarPg2}:\text{data\_page}[@\text{extract} \rightarrow \text{tuple(\text{Contact},\text{Price})}]))
\]
Mapping Web Content onto Virtual Relations

- Web content is mapped onto virtual relations through access handles
- Each access handle $H$ for a relation $R$ contains a binding pattern and a navigation process

$$H = \{\text{mandatory-attrs, selection-attrs, } R, \text{navigation-expression}\}$$

e.g.,

$$H = \{\{\text{make}\}, \{\text{model, year}\}, R, \text{newsday_used_cars}\}$$
After wrapping: integrate!

WebBase

External Schema

Ad-hoc query interface

Logical Schema

High-level algebraic access methods

Virtual Physical Schema

Navigation, data extraction

Raw Web

Site independence

Navigation independence

logical relations

virtual relations
Integrate sources

- Similar to mediators
- How to integrate:
  - warehousing vs virtual
  - local vs global as view
  - relational vs XML
  - integration languages (YATL, MSL, Relational Algebra, Datalog)
  - site capabilities and binding propagation
- Semantic integration
Query Reformulation

- User does not pose queries directly in the schema in which data is “stored”
- A user queries must be reformulated into a query over the schemas of the data sources
Local as View

- Define a global schema
- Users pose queries over the global schema
- For every information source $S$, write a query over the global schema that describe the tuples in $S$
- Simple to add and delete sources
- Query reformulation is complex
- Strategy used by the Information Manifold
Global as View

- Build logical relations from virtual physical relations exported by the wrappers
- Users pose queries over logical relations
- Query reformulation is simple: just unfold!
- Addition and deletion of sources may require modifications on definitions of logical relations
- Strategy used by TSIMMIS and Web Integrator
Local vs Global as View

- Local as view

**source1**: Luxury cars

\[
\text{luxury}(\text{make}, \text{model}, \text{year}, \text{price}) :\neg \text{car}(\text{make}, \text{model}, \text{year}, \text{price}), \text{price} > 20000
\]

**source2**: Old car reviews

\[
\text{reviews}(\text{product}, \text{year}, \text{review}) :\neg \text{review}(\text{product}, \text{year}, \text{review}), \text{car}(\text{product}), \text{year} < 1989
\]

- Global as view

**logical relation 1**: Luxury cars

\[
\text{luxury}(\text{make}, \text{model}, \text{year}, \text{price}) :\neg \text{source1}(\text{make}, \text{model}, \text{year}, \text{price}) \cup \text{source2}(\text{make}, \text{model}, \text{year}, \text{price})
\]

**logical relation 2**: Old car reviews

\[
\text{reviews}(\text{product}, \text{year}, \text{review}) :\neg \text{source1}(\text{car}, \text{year}), \text{year} < 1989, \text{source2}(\text{car}, \text{year}, \text{review})
\]
Many Web information integration systems are based on the relational model (e.g., Information Manifold, Web Integrator). Recently, XML integration systems have been proposed (e.g., YAT) – Flexible data model – Fast wrapping tools – Fast integration based on declarative XML languages
Site capabilities

- Which queries are supported by the Web sites
- For example, a classified ads site requires users to specify *make, model and year* of cars in order to output the car listings
- Logical relations must adhere to the binding requirements of the virtual relations
Mapping Virtual Relations onto Logical Schema

Virtual relations:

- newsday (Make, Model, Year, Price, Contact)
- nytimes (Make, Model, Year, Price, Contact)

- If logical relation classifieds is mapped by:
  classifieds ≡ π Make, Model, Year, Price, Cont, Feat (newsday)
  ∪ π Make, Model, Year, Price, Contact, Feat (nytimes)

- Then Make, Model and Year are the pre-computed binding pattern for classifieds
Semantic Integration

- Mapping of the semantics of terms from different information sources
- Different terms represented using the same syntactic structure (e.g., car - used or new)
- Different syntactic structures to represent the same term (e.g., Volkswagen vs VW)
- Very hard to automate
  - approximate matchings (WHIRL - Cohen)
- Use ontologies
After integrating: query!

Logical independence

External Schema

Ad-hoc query interface

Logical Schema

High-level algebraic access methods

Virtual Physical Schema

Navigation, data extraction

Raw Web

Site independence

WebBase

external views

logical views
Mapping Logical onto External Schema

- Naive users should be able to easily formulate *ad-hoc queries*
- Casual users query the WebBase through the external schema
- We propose **structured universal relation** as a query model
  - provides **logical independence** by allowing users to formulate complex queries *interactively* without specifying joins.
- To guide the user, **structured universal relation**:
  - Maps related logical relations onto *domain concepts*, and
  - structures related domain concepts into a *concept hierarchy*
- **Semantic compatibility rules** are provided for reinforcing *correct joins*, and preventing *incorrect joins*
Structured UR: Example

Concept hierarchy for used cars:

Dealers → Classifieds → Lease → Loan → Full Coverage → Liability → Retail Value → TradeInValue

UsedCar(Car, Price, Contact) → Interest(Car, Rate) → Insurance(Car, Cost)

UsedCarUR(Car, Price, BBPrice, Rate, Contact, Cost) → BlueBook(Car, BBPrice)

Compatibility constraints define meaningful and meaningless joins:

Classifieds → ¬Lease  Can’t lease a “used” car
Lease → FullCoverage  Leased cars must be fully insured
UsedCar → ¬TradeInValue  Tradeins are not applicable to used cars
Structured UR: Example query

Make a list of used BMWs advertised in New York City area sites such that each car's monthly payments are less than 1,000 dollars, and its selling price is less than its Blue Book price

\[
\text{UsedCarUR}(\text{bmw,Model, Year, Price, BBPrice, Rate, InsCost}), \
\quad \text{Price} < \text{BBPrice}, (\text{Price} \cdot \text{Rate} + \text{InsCost})/12 < 1000
\]

Out of 16 candidate maximal objects the following satisfy constraints:

- Dealers, Lease, Full, RetailVal
- \cup \text{Dealers, Loan, Full, RetailVal}
- \cup \text{Dealers, Loan, Liability, RetailVal}
- \cup \text{Classifieds, Loan, Liability, RetailVal}
- \cup \text{Classifieds, Loan, Full, RetailVal}
Web Integration Systems

- TSIMMIS and Information Manifold
  - integrate data from heterogeneous sources, including the Web
- Araneus and Web Integrator
  - WebBases: basis of Web-based information
- YAT
  - XML integration system
Information Manifold

- Data model: relations
- Integration language: conjunctive formulas
- Wrappers
  - implementors write source descriptions - which relations are found in the data sources
- Focus:
  - site independent queries
  - model fine-grained differences between sources
  - query planning: which information sources and their combinations
TSIMMIS

- Data model: OEM (object exchange model) - describe contents
- Integration language: MSL - a rule-based language
- Wrappers
  - implementors write a set of templates/rules that describe queries accepted by the wrapper and objects it returns
Araneus

- Data model: relational
- Web model: ADM - graph whose nodes correspond to pages (pages are complex objects)
- Language: ULIXES - SQL-like language over ADM objects
  - allows end users and application builders to query Web sources
- Wrappers
  - implementors manually create ADM for each site, and write a set of ULIXES queries
Web Integrator

- Web model: navigation map - action-oriented graph whose nodes correspond to pages, and edges to actions (e.g., fill form, follow link)
- Language: Transaction-logic
- Wrappers
  - navigation maps are created semi-automatically
  - queries are automatically generated from navigation maps
  - WebBase designer need not write programs
YAT

- Data model: XML
- Language: YATL - a rule-based integration/conversion language
- Wrappers
  - designers write YATL rules
- XML algebra and efficient query processing techniques
Query Processing

- Optimization is hard
  - no information about data cardinality, distribution, indexes
  - because of limited access patterns, optimizer cannot fall back on full scan [Florescu et al - SIGMOD’99]

- Overlap and redundancy among sources
  - hard to quantify - need detailed description of contents of each source

- Data availability is unpredictable

- Data arrival rates may vary dramatically
  - adaptive query execution [Urhan et al - SIGMOD’98, Ives et al - SIGMOD’99]
  - query feedback to predict response times [Gruser at al, VLDB Journal, 2000]
Summary and Conclusions

- Presented key concepts and technologies for finding, querying and integrating information on the Web
- Keyword search and directory browsing are still essential services, but tools are needed for ad-hoc search and querying
- Lots of work have been put into helping people find the information they want - but a lot more is needed
- We need “components” to build search engines, and web information integration systems
XML is not a panacea...

- It does help, but a lot needs to be done...
- XML presents many challenges for the database community
- Large quantities of a new type of data
  - textual, irregular, self-organizing, distributed, replicated, etc.
- Many orders of magnitude larger:
  - the volume of XML data
  - the number of XML data repositories
- The need for such a technology is here
- The solutions are not here!
- Myriad of standards and products issued from industry
Future directions (1)

- XML data management
  - Update languages
  - Storing XML data in object-relational DBMSs, and native storage
  - Indexing
- XML views of object-relational databases
- Query processing algorithms for XML data
- Mixing structured search with full-text search
- XML benchmarks
Future directions (2)

- Continuous queries for notification services
- Querying streams of data (e.g., call information)
- Accessing the Web from wireless Internet devices
  - different requirements for query processing
  - flexible re-structuring of sites
- How to describe Web content/services
  - service composition
- Online semantic integration
- Alternative query interfaces for WebBases
Future directions (cont.)

- Search engines
  - How to scale more
    » incremental crawling - how much does the Web change
  - Scaling up the operation may be feasible, but it useful?
  - RDF/Ontologies/XML
  - Limited to static Web - can’t search through dynamically generated pages