The Continued Saga of DB-IR Integration

Ricardo Baeza-Yates
rbaeza@dcc.uchile.cl
www.baeza.cl
Center for Web Research
Dept. of Computer Science
University of Chile

Mariano P. Consens
consens@mie.utoronto.ca
www.cs.toronto.edu/~consens
Information Engineering, MIE
& Dept. of Computer Science
University of Toronto

Very Large Data Bases

Agenda

1. Motivation
2. An Introduction to IR
3. Requirements for DB-IR
4. Semi-structured Data
5. Industrial DB-IR Examples: Oracle, Verity
6. DB Approaches
7. IR & Hybrid Approaches
8. Open Problems
9. Bibliography

Very Large Data Bases

Disclaimer

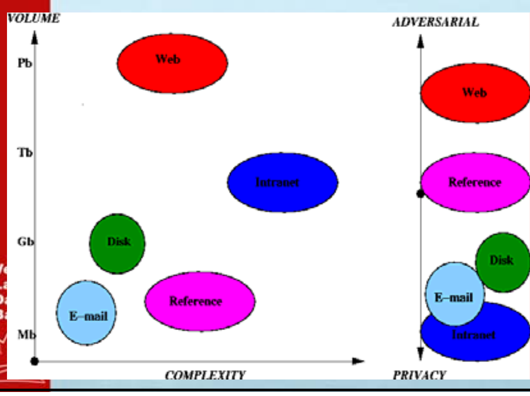
- This tutorial reflects the personal biases, preferences and limitations of the presenters ☺
- It does not cover everything done in DB+IR
- It does not cover related areas such as other XML problems, multimedia, spatial databases, etc.

Very Large Data Bases

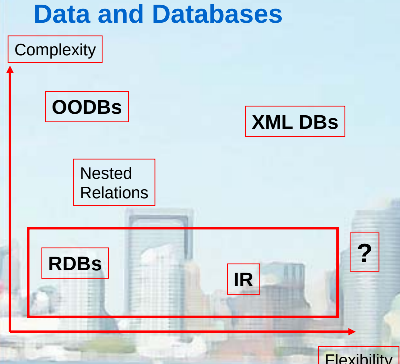
1. Motivation

- Types of data
- DB & IR Views
- Possible Solutions
- Applications
- Search Problems

Different Views on Data



Data and Databases



RDB vs. IR

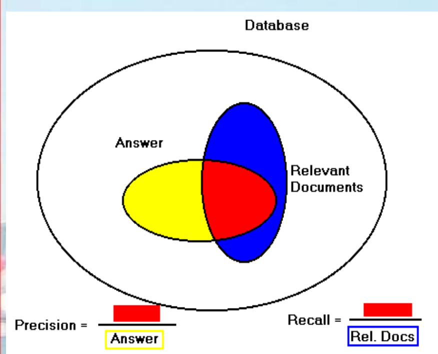
- DBs allow structured querying
- Queries and results (tuples) are different objects
- Soundness & completeness expected
- All results are equally good
- User is expected to know the structure

- IR only supports unstructured querying
- Queries and results are both documents
- Results are usually imprecise & incomplete
- Some results are more relevant than others
- User is expected to be dumb

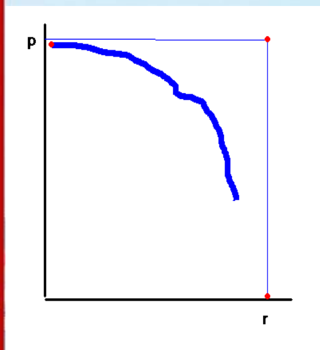
The Notion of Relevance

- Data retrieval: semantics tied to syntax
- Information retrieval: ambiguous semantics
- Relevance:
  - Depends on the user
  - Depends on the context (task, time, etc)
  - Corollary: The Perfect IR System does not exist

Evaluation: First Quality, next Efficiency



Evaluation: Comparing Systems



p-r normalized graph

TREC:

Collection
+
Queries
+
Answers

Possible Architectures

- IR on top of RDBs
- IR supported via functions in an RDB
- IR on top of a relational *storage* engine

- Middleware layer on top of RDB & IR systems
- RDB functionality on top of an IR system
- Integration via an XML database & query language

Problems of the IR view

- Very simple query language
  - It is natural language the solution?
- No query optimization
- Does not handle the complete answer

Problems of the DB view

- The syndrome of the formal model
  - Model is possible because of structure
- The syndrome of “search then rank”
  - Large answers
  - Optimization is useless
  - Quality vs. Speed
  - E.g. XQuery
- What is a Data Base?
- Are RDBs really a special case of IR systems?

Applications for Integrated Systems

- E-commerce search
- Intranets & enterprise data
- Customer support (e.g. CRM)
- News archives, bulletin boards, etc.
- Personal information (e.g. My Life Bits)
- P2P Web Search

Challenges posed by the Web

- Integration of autonomous data sources
  - Data/information integration
- Supporting heterogeneous data
  - How to do effective querying in the presence of structured and text data
  - How to support IR-style querying on DBs
    - Because now users seem to know IR/keyword style querying more, even though structure is good because it supports structured querying!
  - How to support imprecise queries

Enterprise Search is Different

- Sophisticated systems run by librarians are morphing into simple self-service web-based search
  - Must be scalable, reliable, highly available
- Data is different
  - Heterogeneous in format & structure (documents, DBs, etc)
  - Less volume & better quality
- Searching is also different
  - Less & better queries, different tasks
  - Focus in recall rather than precision
- Other issues: security, able to search but not to see

Very Large Data Bases

What is a Bad Interface/Result?

- No search box
- Inability to judge user intent
  - No spell checking
  - No context disambiguation (cricket: game or bug?)
  - No recommendation system, no user feedback
- Too many hits: answer overload
  - Return 10,000 hits when the average user looks only at the top-20
- The most relevant item is not at the top of the list
- Too many similar documents
  - Poor duplicate detection, poor clustering/categorization
- Inability to understand why a document has been returned
  - No KWIC
- Lack of Meta information
  - Size, format, date, etc.

Very Large Data Bases

Cost of a Bad Search

- Information is useless if no one can find it
  - ROI for employee productivity
  - ROI for customer satisfaction
  - Cost of people using out-of-date information
  - Cost of people using wrong information
  - Cost of recreating information which cannot be found
  - Cost of opportunity for not finding the information

Very Large Data Bases

Some Examples - I

Where is the search box?

Very Large Data Bases

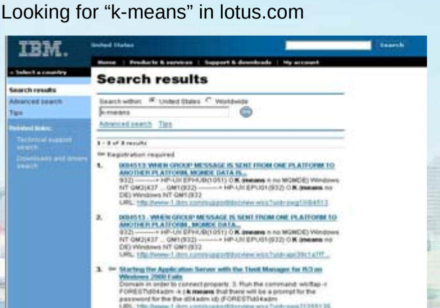
Some Examples – II

“ultra seek” or “ultraseek”?

Very Large Data Bases

Some Examples - III

Looking for “k-means” in lotus.com



Very Large Data Bases

Agenda

1. Motivation
2. An Introduction to IR
3. Requirements for DB-IR
4. Semi-structured Data
5. Industrial DB-IR Examples: Oracle, Verity
6. DB Approaches
7. IR & Hybrid Approaches
8. Open Problems
9. Bibliography

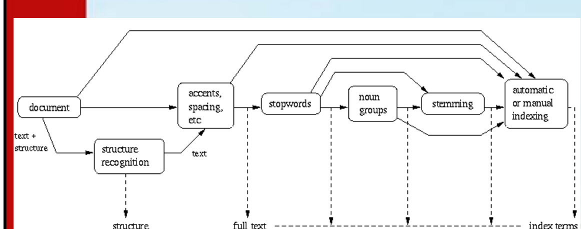
Very Large Data Bases

2. Introduction to IR through Web Retrieval

- IR challenges posed by the Web
- Logical view of text
- Similarity models
- IR system architecture
- IR query languages & interfaces

Very Large Data Bases

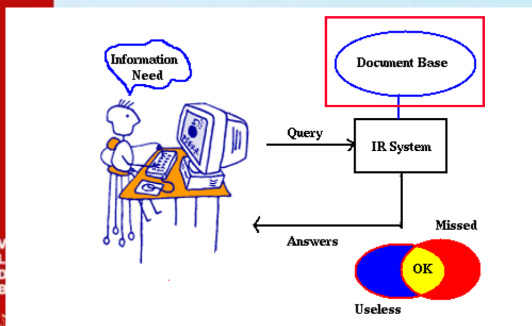
Bag-of-Words Representation



Very Large Data Bases

Full-text continuum:
ambiguity vs. completeness trade-off

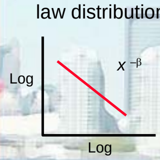
Challenges in Current IR Systems



Document Base: Web

- Largest public repository of *data*
  (more than 6 billion static pages?)

- Today, there are more than 60 million Web servers
- Well connected graph with out-link and in-link power law distributions

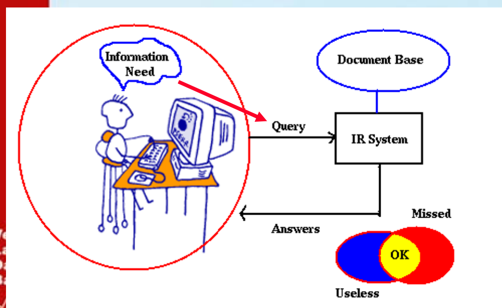


Self-similar &
Self-organizing

Web Retrieval

- Problems:
  - volume
  - fast rate of change and growth
  - dynamic content
  - redundancy
  - organization and data quality
  - diversity
  - …..
- Deal with data overload

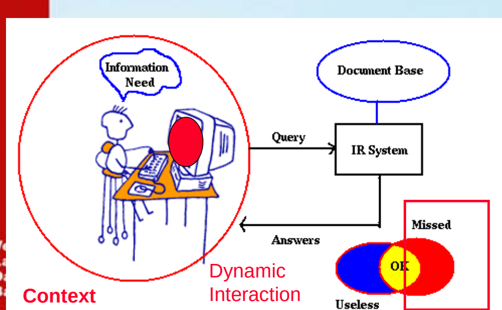
Challenges in Current IR Systems



Web Users

- Cultural and educational diversity
- Short queries
  - Inherent to users or due to the query language?
- Different goals:
  - Information need
  - Navigational need
  - Transactional need
- Short patience
  - few queries posed & few answers seen
- Other problems: concurrency, scale, ...

Challenges in Current IR Systems

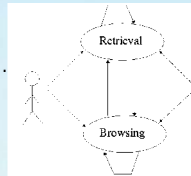


Interaction

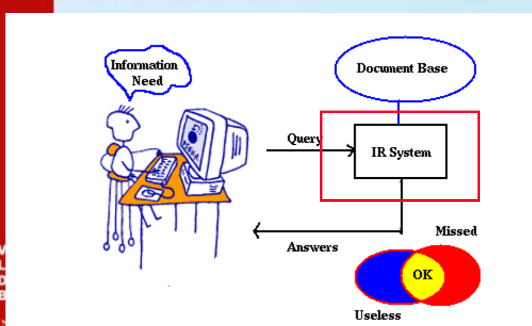
- Inexperienced users
- Dynamic information needs
- Varying task: querying, browsing,...

- No content overview
- Poor query language, no help

- Poor preview, no visualization
- Missing answers: partial Web coverage, invisible Web, different words or media, ...
- Useless answers

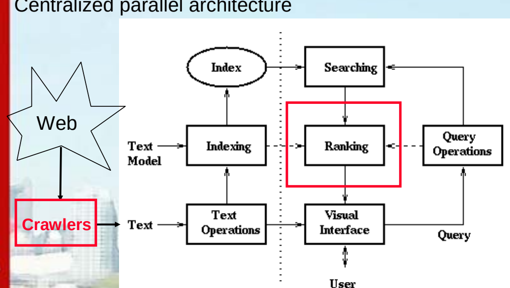


Challenges in Current IR Systems



Web Retrieval Architecture

Centralized parallel architecture



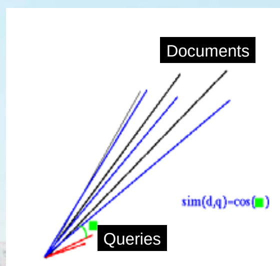
Algorithmic Challenges

- Crawling:
  - Quantity
  - Freshness
  - Quality
  - Politeness vs. Usage of Resources
- Ranking
  - Words, links, usage logs, … , metadata
  - Spamming of all kinds of data
  - Good precision, unknown recall Adversarial IR

Very Large Data Bases

Text Similarity Models

Vector model:
• words are dimensions
• *tf-idf* is used for weights

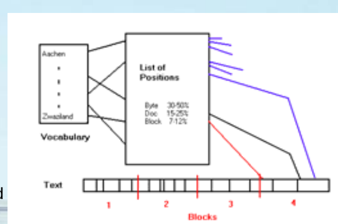


- Set Models:
  - Boolean, Fuzzy sets, ...
- Algebraic Models:
  - Vector, LSI, etc.
- Probabilistic Models:
  - Probabilistic, Inference & belief networks

Very Large Data Bases

Index

- Inverted index
- Lists sorted by weight
  - global (e.g. Pagerank)
  - local (e.g. word weights)
- Hashing + set operations
- Compressed
- Incremental updates



Very Large Data Bases

Parallel Case

- Collection is divided per server
- Local indexes are used
  - Document partitioning
- Brokers distribute queries and merge results
- Simpler to build and update
- Good load balance, low concurrency

- In theory a global partitioned index achieves higher concurrency but has lower load balance and more difficult to build & maintain

Non-word based Applications

- Suffix trees
- Linear building time
- Linear space (but larger than data)
- Suffix arrays
- Linear building time, less space
- Powerful search:
  - any substring
  - approximate search
  - regular expressions
- Applications: biology, music, linguistic, etc.

Link Ranking

- Incoming links count (Li, 1997)
- HITS (Kleinberg, 1998)
  - Authorities: good pages
  - Hubs: good links
- PageRank (Page & Brin, 1998)
  - Random walk + random jumps if “bored”
- Many variations of these ideas
- Good to find communities, spam, etc.
- Application to other problems (e.g. ranking relations)

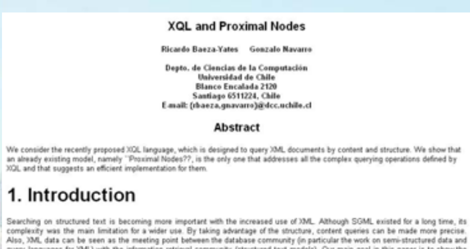
Agenda

1. Motivation
2. An Introduction to IR
3. Requirements for DB-IR
4. Semi-structured Data
5. Industrial DB-IR Examples: Oracle, Verity
6. DB Approaches
7. IR & Hybrid Approaches
8. Open Problems
9. Bibliography

3. Requirements for DB-IR

- Motivating Applications
- Data and Query Requirements
- Sample Use Cases

Sample Paper on the Web



XQL and Proximal Nodes

Ricardo Baeza-Yates   Gonzalo Navarro

Depto. de Ciencias de la Computación
Universidad de Chile
Blanco Encalada 2120
Santiago 6511224, Chile
E-mail: {rbaeza,gnavarro}@dcc.uchile.cl

Abstract

We consider the recently proposed XQL language, which is designed to query XML documents by content and structure. We show that an already existing model, namely ``Proximal Nodes??, is the only one that addresses all the complex querying operations defined by XQL and that suggests an efficient implementation for them.

1. Introduction

Searching on structured text is becoming more important with the increased use of XML. Although SGML existed for a long time, its complexity was the main limitation for a wider use. By taking advantage of the structure, content queries can be made more precise. Also, XML data can be seen as the meeting point between the database community (in particular the work on semi-structured data and query languages for XML) with the information retrieval community (structured text models). Our main goal in this paper is to show the

Bibliography Entry

```
<proceedings>
  <inproceedings>
    <author>Ricardo Baeza-Yates</author>
    <author>Gonzalo Navarro</author>
    <title>XQL and Proximal Nodes</title>
    ...
  </inproceedings>
</proceedings>
```

- Describes metadata for the workshop article
- The XML data conforms to the DBLP schema (DTD)

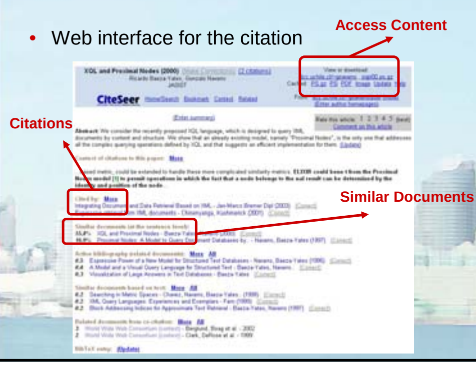
Paper Content in XML

```
<workshop date="28 July 2000">
  <title> XML and Information Retrieval: A SIGIR 2000 Workshop
  </title>
  <editors> David Carmel, Yoelle Maarek, Aya Soffer
  </editors>
  <proceedings>
  <paper id="1">
    <title> XQL and Proximal Nodes </title>
    <author> Ricardo Baeza-Yates </author>
    <author> Gonzalo Navarro </author>
    <abstract> We consider the recently proposed language …
    </abstract>
    <section name="Introduction">
      Searching on structured text is becoming more important with XML …
    </section>
    …
    <cite xmlns:xlink="http://www.acm.org/sigir/.../paper/xmlql"> … </cite>
  </paper>
  …
</workshop>
```

- The XML data conforms to the publisher's DTD

A Digital Library Application

- Web interface for the citation



Applications Areas

- Scientific, Technical and Medical Reference Books, Journals, Publications
- Case Law and Litigation Materials
- Regulatory and Business Filings
- Maintenance, Repairs and Operations Manuals
- Product Documentation
  - Design
  - Procurement (SRM)
  - Customer Service (CRM)
- Collaboration, Portals
- Web, Intranet, Group & Personal Repositories
- Represents "80% of enterprise data"

Data Requirements

Objects

- Text, Documents, Images, Application Files, Multimedia Content
- Structured Data
  - Relations: Refers (From, To)
  - Hierarchies: proceedings/paper/section
- Semi-structured Data
  - Editorial comments on the paper

Nested

- Assumption: XML provides a reasonably way to capture the requirements above

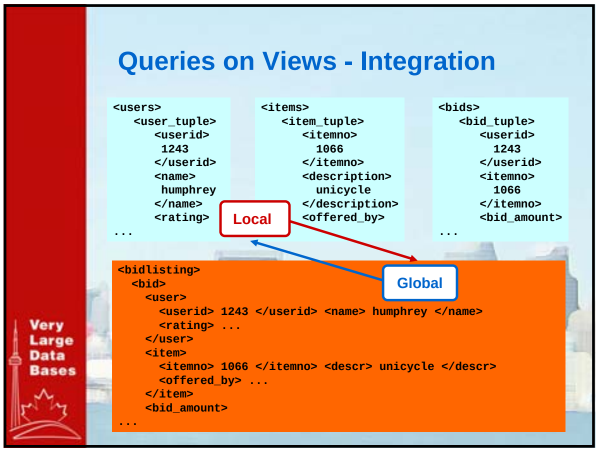
Publishing Relational Data

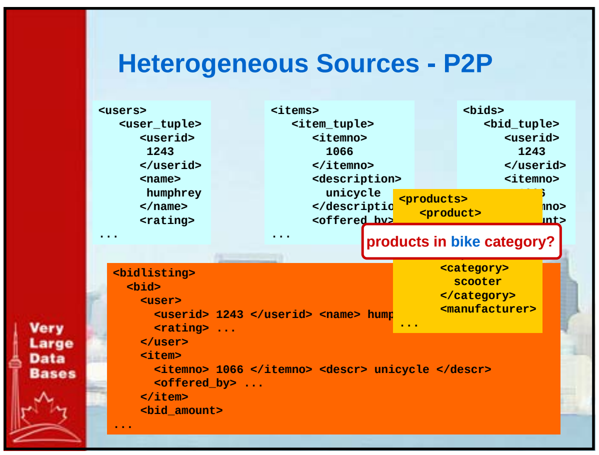
USERS: USERID NAME RATING

ITEMS: ITEMNO DESCRIPTION OFFERED_BY RESERVE_PRICE

BIDS: USERID ITEMNO BID_AMOUNT BID_DATE

```
<bidlisting>
  <bid>
    <user>
      <userid> 1243 </userid> <name> humphrey </name>
      <rating> ...
    </user>
    <item>
      <itemno> 1066 </itemno> <descr> unicycle </descr>
      <offered_by> ...
    </item>
    <bid_amount>
...
```

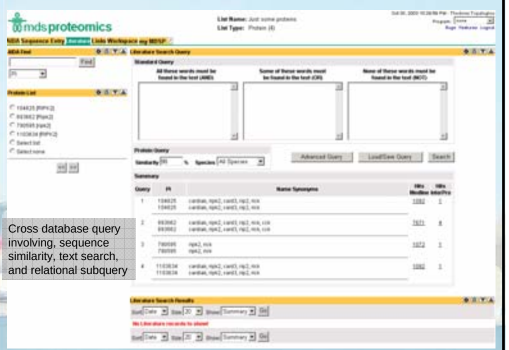

Queries on Views - Integration
<users>
<user_tuple>
<userid>
1243
</userid>
<name>
humphrey
</name>
<rating>
...
<items>
<item_tuple>
<itemno>
1066
</itemno>
<description>
unicycle
</description>
<offered_by>
<bids>
<bid_tuple>
<userid>
1243
</userid>
<itemno>
1066
</itemno>
<bid_amount>
...
Local
Global
<bidlisting>
<bid>
<user>
<userid> 1243 </userid> <name> humphrey </name>
<rating> ...
</user>
<item>
<itemno> 1066 </itemno> <descr> unicycle </descr>
<offered_by> ...
</item>
<bid_amount>
...
Very Large Data Bases


Heterogeneous Sources - P2P
<users>
<user_tuple>
<userid>
1243
</userid>
<name>
humphrey
</name>
<rating>
...
<items>
<item_tuple>
<itemno>
1066
</itemno>
<description>
unicycle
...
<bids>
<bid_tuple>
<userid>
1243
</userid>
<itemno>
<products>
<product>
products in bike category?
<category>
scooter
</category>
<manufacturer>
...
<bidlisting>
<bid>
<user>
<rating> ...
</user>
<item>
<itemno> 1066 </itemno> <descr> unicycle </descr>
<offered_by> ...
</item>
<bid_amount>
...
Very Large Data Bases


Query Requirements Overview
• Developing the web application
Content-only
Structure-only
Content and Structure
Relevance, Similarity
Score
Top-k
CiteSeer
Very Large Data Bases

Proteomics Portal (courtesy T. Topaloglou, Protana)

- Map the proteins seen in a experiments to the scientific literature



Cross database query involving, sequence similarity, text search, and relational subquery

Very Large Data Bases

DB-IR Query Requirements

- Express arbitrary Full-Text (FT) searches
- Select the substructures where the FT condition applies (*search context*)
- Select the substructures to be returned (*return context*)
- Choose how to determine relevance for results and (weighted) queries
- Access and combine the relevance scores
- Limit answer to top-k
- Support approximate structural searches
  S. Amer-Yahia, N. Koudas, D. Srivastava, ICDE 2003 Tutorial
- Full composition of FT and structural queries

Very Large Data Bases

Additional DB-IR Requirements

- Efficient and scalable query evaluation, supported by
  - Indexes (FT and structural)
  - Optimizer (plans and operators)
- Rich functionality for presenting answers
  - Visual interfaces
  - Highlight the FT terms *in context*
- Support queries on integrated views
- Query heterogeneous structure
  - Within a single collection
  - In data repository crawled from web sources
  - Across peer sources

Very Large Data Bases

Sample Use Cases

- Quick overview of the range of possible DB-IR requirements
  - Identify search and return contexts
  - Motivate relevance
  - Illustrate composition
- Extension of use cases from Full-text XQuery (//www.w3.org/TR/xmlquery-full-text-use-cases)

Finding Text in Elements

- Find all book titles containing the word "usability“
- Find all books with the phrase "usability tests" in book or chapter titles
  - Multiple search contexts, different return
- Find all books with the phrase "usability tests" (even across elements)
- Find all book titles for books with abstracts mentioning software developers (interpreted as having broad terms “software” near “developer”)
  - Proximity
  - Thesaurus (developer, programmer)

Finding Text in Structure

- Find the first two sections mentioning “task” in chapters on "conducting usability tests“ with the book abstract not mentioning “software”
  - Structured search contexts
    - book/chapter//section
    - book/chapter
    - book/abstract
- Do the above ignoring footnotes in chapters but not in abstracts
  - Modifies the search contexts
  - Match the contexts approximately

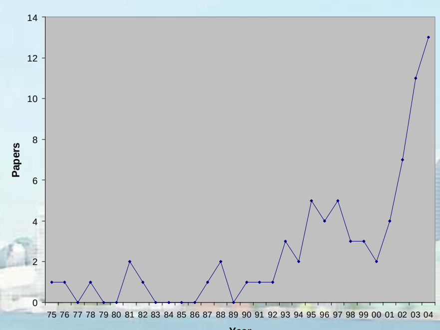
Ranking

- Find how relevant to "usability" are the books
- Find the best two books on "usability tests“
  - Take into account reviewers comments
- Return all books with only the sections highly relevant to "usability”
- Rank on both approximate structure and content matching the sections mentioning “task” in chapters on "conducting usability tests“ with the book abstract not mentioning “software”

Composing Queries

- For books with "usability" in the title create a flat list of all titles and the authors
- Find the 10 most relevant books about conducting "usability tests" which have more than one author and are published after "2000“
- Find all books published after "2001" which share a subject with the 10 most relevant books on “usability" that have titles mentioning “software“ and “developer”

The (VLDB-only) DB-IR Saga



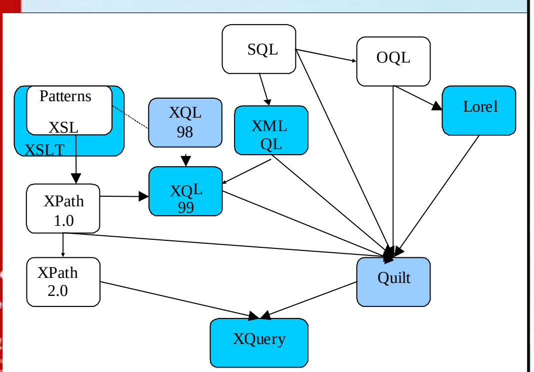
Agenda

1. Motivation
2. An Introduction to IR
3. Requirements for DB-IR
4. Semi-structured Data
5. Industrial DB-IR Examples: Oracle, Verity
6. DB Approaches
7. IR & Hybrid Approaches
8. Open Problems
9. Bibliography

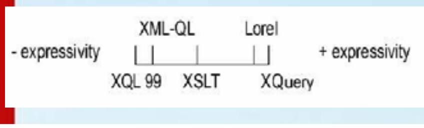
4. Semistructured Data

- XQuery
- XQuery & Full-text
- Structured Text Models
  - Proximal Nodes

XQuery History



XML Query Language Comparison



	Lorel	XSLT	XML-QL	XQL 99	XQuery
Main functions	Queries of semi-structured data	Transformation of documents	Data queries, transformations, integration of XML data from different sources	Queries within a document and queries on collections of documents	Queries on heterogeneous data sources
Data model	Graph / Tree	Tree (such as XPath 1.0)	Graph	Tree (DOM of XML)	Ordered sequence of nodes (such as XPath 2.0)
Input source & format	XML Documents	XML Document/s + StyleSheet	XML Documents from different sources	XML Document/s	XML Document, XML Fragments, Collections of XML documents
Output information	XML Document (Ordered list of identifiers of the resulting elements)	XML Document (Transformed XML tree), Collections of XML documents (*xsl:document*)	XML Document (XML Fragments)	XML Document (XML Fragments, List of resulting elements)	XML Document, XML Fragment, Collections of XML documents

XML Query Language Comparison

		Lorel	XSLT	XML-QL	XQL 99	XQuery
Selection Operation	**Pattern/ Filter/ Constructor**	**select** *constructor* **from** *pattern* **where** *filter*	**<xsl:for-each select=** pattern **> <xsl:if match=**filter**> <copy-of /> </xsl:if> </xsl:for-each>**	**WHERE** *pattern* **IN** *source, filter* **CONSTRUCT** *constructor*	pattern [filter]	**FOR** *patterns* **LET** *bindings* **WHERE** *filter* **RETURN** *constructor*
	Relational Operators	>, >=, <, <=, =, <>, ==	>, >=, <, <=, =, !=	>, >=, <, <=, =, !=	>, >=, <, <=, =, !=	>, >=, <, <=, =, != For nodes: ==, !==
	Boolean Operators	*and, or, not*	*and, or*	No	*and, or*	*AND, OR*
	Nesting queries	Yes	Yes	Yes	Yes	Yes
	Creation of new elements	Yes	Yes	Yes	No	Yes
Filtering of elements preserving hierarchy		No	Yes (using templates)	No	Yes	Yes (*filter*)
Reduction		No	Yes	No	Yes	No
Restructuring operations	**Grouping of results**	Yes (*group by*)	No	No	Only by structure, not by value	Yes
	Skolem Functions	Yes	No	Yes	No	Yes
	Sorting of results	Yes (*order by*)	Partial (xsl:sort[a])	Yes (*ORDER-BY*)	No	Yes (*SORTBY*)
Inter-document links (join), Intra-documents links (semi-join)		Join, Semi-join	Semi-join	Join, semi-join	Semi-join, join	Join, semi-join

XML Query Language Comparison

		Lorel	XSLT	XML-QL	XQL 99	XQuery
Use of tag variables		Yes	Yes	Yes	No	Yes
Path expressions		Regular expression operators: *, |, +, ? Qualifiers: >, @	XPath Expressions	Regular expression operators *, |, +, .	Wild card: * Path Operators: /, //	XPath Expressions
Dereferencing of IDREF(S) attributes		Yes (As a subelement using the point notation)	Yes (*id()*)	Yes (By means of a join)	Yes (*id()*)	Yes (Dereference Operator =>)
Set Functions		*min, max, count, sum, avg*	*sum, count*	*min, max, count, sum, avg*	*sum, count*	*min, max, count, sum, avg*
Quantifiers	**Existential**	Yes (*exists*)	Yes (implicit)	Yes (implicit)	Yes (implicit)	Yes (*SOME*)
	Universal	Yes (*for all*)	No	No	Yes (*all*)	Yes (*EVERY*)
Handling of datatypes (XML Schema)		Partial	No (under study)	No	No	Yes
Insertion, delete and update		Yes	Yes	No	No	No

XML Query Language Comparison

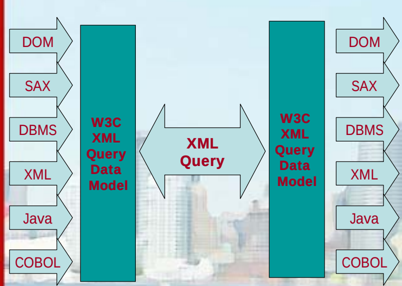
		Lorel	XSLT	XML-QL	XQL 99	XQuery
Keywords	A word inside free text	By means of path expressions	By means of path expressions	By means of path expressions	By means of path expressions	By means of path expressions
	Similarity	No	No	No	No	No
	Context	No	No	No	No	No
	Boolean Operators	Yes	Yes	No	Yes	Yes
Pattern matching		operators: *like, grep, soundex*	String operators and functions	*Like* operator	String operators and functions	String operators and functions
Structural Queries	Structural Inclusion	By means of path expressions	By means of path expressions	By means of path expressions	By means of path expressions	By means of path expressions
	Positional Inclusion	Yes	Yes	Yes	Yes	Yes
	Structural proximity	No	No	No	Yes (immediately precedes ";")	Context node
	Structural Order	By means of comparison of positional indexes	Yes (*preceding, preceding-siblings, following, following-siblings*)	By means of comparison of positional indexes	Yes (*before, after*)	Yes (*BEFORE, AFTER*)
Assignation of weighting to the terms of the query		No	No	No	No	No
RDF support		No	No	No	No	No
XLink and Xpointer support		No	No	No	Partial	No (In study)
Operations over sets		Intersection, union, difference	Union, difference	Intersection, union	Intersection, union	Intersection, union, difference

XML Query Data Model

- Joint with XPath 2.0, XSL 2.0
  - Last version of Feb 2004
- Ordered, labeled forest
- Based on XML Information Set, PSVI
- Has node identity
- DTDs (from SGML, IR style)
- XML Scheme (DB style)
  - Provide data types

Very Large Data Bases

XQuery and the Data Model



Very Large Data Bases

XML Query Formal Semantics

- XQuery is a functional language
  - A query is an expression
  - Expressions can be nested with full generality.
  - A pure functional language with impure syntax
- Static Semantics
  - Type inference rules
  - Structural subsumption
- Dynamic Semantics
  - Value inference rules
  - Define the meaning of XQuery expressions in terms of the XML Query Data Model

Very Large Data Bases

XQuery Expressions

- Element constructors
- Path expressions
- Restructuring
  - FLWOR expressions
  - Conditional expressions
  - Quantified expressions
- Operators and functions
- List constructors
- Expressions that test or modify data types

Very Large Data Bases

Path Expressions

```
<bib>
  <book year="1994">
    <title>TCP/
    <author>
      <last>Stev
      <first>W.<
    </author>
    <publisher>
    <price> 65.
  </book>
```

```
{-- XQuery uses the abbreviated syntax
    of XPath for path expressions     --}
document("bib.xml")
/bib/book/author
/bib/book//*
//author[last="Stevens" and first="W."]
document("bib.xml")//author
```

Very Large Data Bases

FLWOR Expressions

- FOR - LET - WHERE - ORDER BY - RETURN
- Similar to SQL's SELECT - FROM - WHERE

```
for $book in document("bib.xml")//book
where $book/publisher = "Addison-Wesley"
return
    <book>
     {
       $book/title,
      $book/author
     }
    </book>
```

Very Large Data Bases

SQL vs. XQuery

"Find item numbers of books"

- SQL:

```
SELECT itemno
FROM items AS i
WHERE description LIKE 'Book'
ORDER BY itemno;
```

- XQuery:

```
FOR $i IN //item_tuple
WHERE contains($i/description, "Books")
RETURN $i/itemno ORDERBY(.)
```

Very Large Data Bases

Inner Join

"List names of users and descriptions of the items they offer"

- SQL:

```
SELECT u.name, i.description
FROM users AS u, items AS i
WHERE u.userid = i.offered_by
ORDER BY name, description;
```

- XQuery:

```
FOR $u IN //user_tuple, $i IN //item_tuple
WHERE $u/userid = $i/offered_by
RETURN
   <offering> {
     $u/name,
     $i/description
   } </offering> ORDERBY(name, description)
```

Very Large Data Bases

Text Search

<section><title>Procedure</title>
The patient was taken to the operating room where she was placed in a supine po
<anesthesia>i
</anesthesia>
<prep> <actio
bladder</ac
and the abd
</prep>
<incision>A c
<geography>
</geography
and the subcutaneous tissue was divided
<instrument>using electrocautery.</instrument>
</incision>

Conditions on Text

Equality:

//section[title="Procedure"]

Full-text:

//section[contains(title, "Procedure")]

Full-text Requirements - I

- Full-text predicates and SCORE functions are independent
- Full-text predicates use a language subset of SCORE functions
- Allow the user to return and sort-by SCORE (0..1)
- SCORE must not require explicit global corpus statistics
- SCORE algorithm should be provided and can be disabled
- Problems:
  - Not clear how to rank without global measures
  - Many/no answers problems
  - Search then rank is not practical
  - How to integrate other SCORE functions?

Full-text Requirements - II

- Minimal operations:
  - Single-word and phrase search with stopwords
  - Suffix, prefix, infix
  - Proximity searching (with order)
  - Boolean operations
  - Word normalization, diacritics
  - Ranking relevance (SCORE)
- Search over everything, including attributes
- Proximity across markup elements
- Extensible

XQuery Implementations

- Software AG's Tamino XML Query
- Microsoft, Oracle,
- Lucent Galax
- GMD-IPSI\item X-Hive
- XML Global
- SourceForge XQuench, Saxon, eXist, XQuery Lite
- Fatdog
- Qexo (GNU Kawa) - compiles to Java byte code
- Openlink, CL-XML (Common Lisp), Kweelt,...
- Soda3, DB4XML and about 15 more

Very Large Data Bases

Why XQuery?

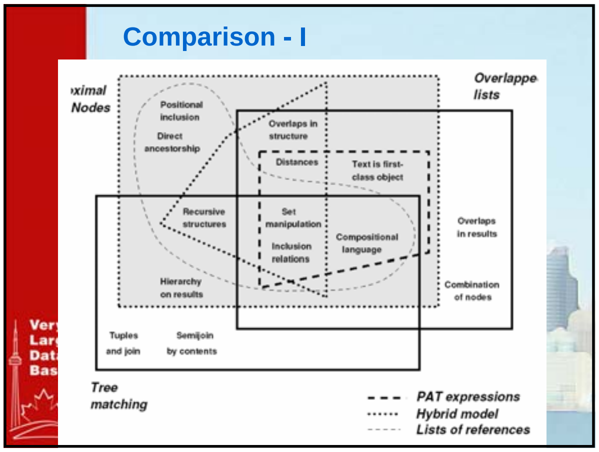
- Expressive power
- Easy to learn (?)
- Easy to implement (?)
- Optimizable in many environments
- Related to concepts people already know
- Several current implementations
- The accepted W3C XML Query Language

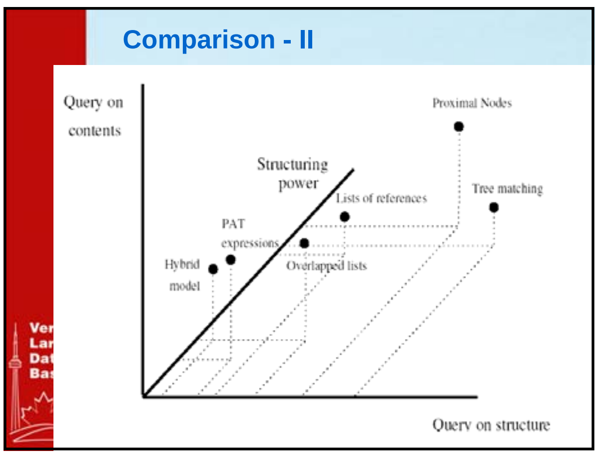
Very Large Data Bases

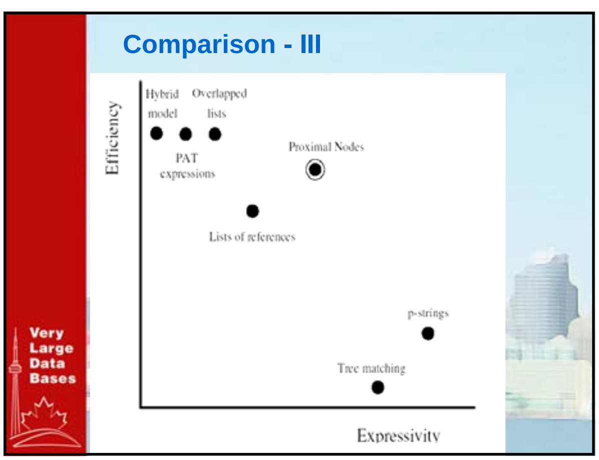
Structured Text Models

- Trade-off: expressiveness vs. efficiency
- Models (1989-1995)
  - Hybrid model (flat fields)
  - PAT expressions
  - Overlapped lists
  - Reference lists
  - Proximal nodes
  - Region algebra
    - Proposed as Algebra for XML-IR-DB Sandwich
  - p-strings
  - Tree matching

Very Large Data Bases


Comparison - I
ximal
Nodes
Overlappe
lists
Positional
inclusion
Direct
ancestorship
Overlaps in
structure
Distances
Text is first-
class object
Recursive
structures
Set
manipulation
Inclusion
relations
Compositional
language
Overlaps
in results
Hierarchy
on results
Combination
of nodes
Tuples
and join
Semijoin
by contents
Tree
matching
PAT expressions
Hybrid model
Lists of references


Comparison - II
Query on
contents
Proximal Nodes
Structuring
power
Lists of references
Tree matching
PAT
expressions
Hybrid
model
Overlapped lists
Query on structure


Comparison - III
Efficiency
Hybrid
model
Overlapped
lists
PAT
expressions
Proximal Nodes
Lists of references
p-strings
Tree matching
Expressivity
Very
Large
Data
Bases

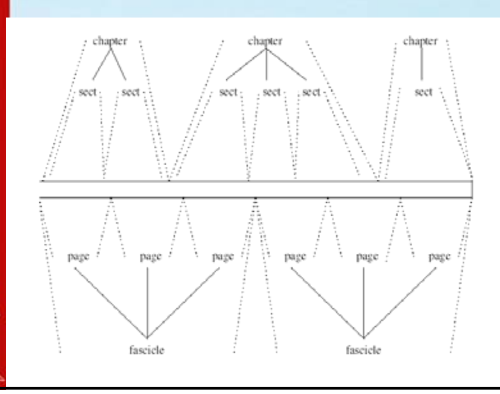
Example: Proximal Nodes (Navarro & Baeza-Yates, 1995)

- Hierarchical structure
- Set-oriented language
- Avoid traversing the whole database
- Bottom-up strategy
- Solve leaves with indexes
- Operators work with near-by nodes
- Operators cannot use the text contents
- Most XPath and XQuery expressions can be solved using this model

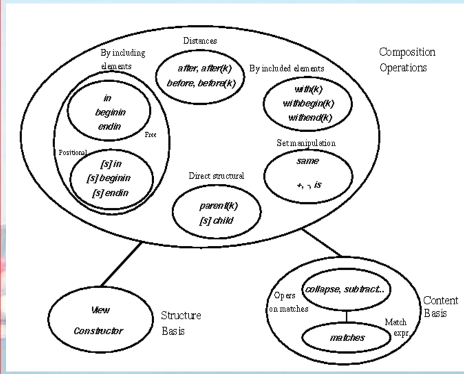
Proximal Nodes: Data Model

- Text = sequence of symbols (filtered)
- Structure = set of independent and disjoint hierarchies or "views"
- Node = Constructor + Segment
- Segment of node $\supseteq$ segment of children
- Text view, to modelize pattern-matching queries
- Query result = subset of some view

Proximal Nodes: Hierarchies

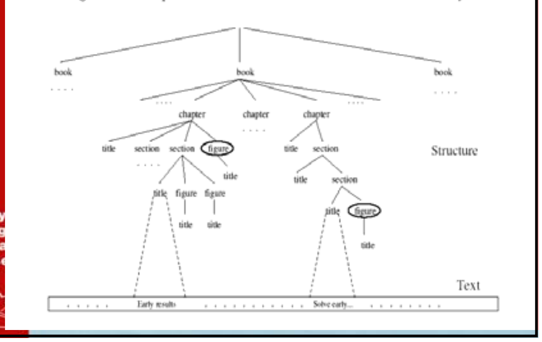


Proximal Nodes: Operations

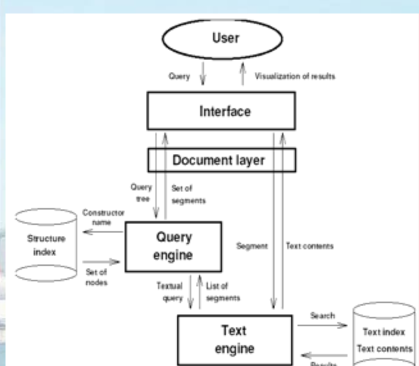


Proximal Nodes: Query Example

```
[last] figure in (chapter with (section with (title with "early")))
```



Proximal Nodes: Architecture



Agenda

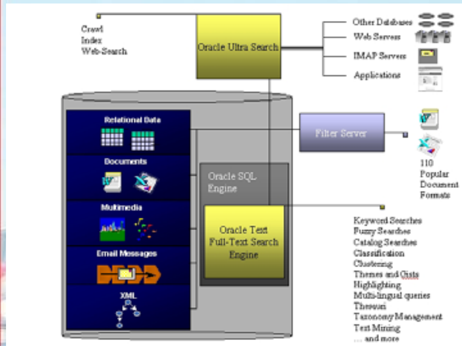
5. Industrial DB-IR Examples: Oracle, Verity

- DB View: Oracle
- IR View: Verity
- Provided by them!
- Thanks to
  - Omar Alonso (Oracle)
  - Prabakhar Raghavan (Verity)

A DB Example: Oracle

- Oracle Text
  - Complete API for building any type of search application
  - Features range from basic keyword searching to advanced techniques like classification and information visualization
- Oracle Ultra Search
  - Out-of-the-box solution that requires no coding
  - Can search across OCS components, websites, databases, files, email, and Portal
  - Built on top of Oracle Text
- Included free with the standard system

Oracle Text Search Architecture



Very Large Data Bases

Common Myths about Oracle Search (according to Oracle)

- Database-Integrated Search Technology is slow
- Oracle's Search Technology is less functional than specialized search-only engines
- Major sites must run specialized search engines
- Oracle is expensive
- Oracle is complex
- Oracle's search technology will not scale out
- You can only search database-resident content with Oracle

Very Large Data Bases

Oracle Text Search Functionality

- Fully integrated with the database
- Premier text search quality (TREC-8 win)
- Advanced linguistics: built-in extensible thesaurus, themes, gists, fuzzy, internationalization features for multilingual applications, etc.
- Document services: multilingual highlighting, themes, navigation …
- XML support
- Classification (TREC-10 win)
- Statistical Text Processing: Clustering
- Integrated with JDeveloper Java IDE
- Filters for 100+ document formats
- Specialized indexes for catalogs, classification, XPath searches
- Visualization
- Integrated web-crawler and out-of-the-box-GUI with Ultra Search

Very Large Data Bases

Quality

- Link awareness
  - Popular pages and hubs
  - Website structure
  - Page structure
- Duplicate elimination
  - Remove URLs with duplicate or near duplicate content
- Spelling correction
  - Component that uses a dictionary and data from query logs
  - *Did you mean …?*
- KWIC (Key Word In Context)
  - Highlights relevant parts of the document
  - No need to open the URL if it doesn't look relevant

Very Large Data Bases

Performance

- Oracle Text integrates with and benefits from features like
  - Data partitioning
  - RAC
  - Query optimization
- Common and rare queries
  - Small index on URL and title for common queries
  - Large index on document content for rare queries
- Query Relaxation
  - Enables you to execute most restrictive query first
  - Then relaxing the search

Very Large Data Bases

Ease of Use

- Users want a simple and easy to use search interface
- Hide all the complexity and expose simple interface
- Ultra Search
- Two search modes
  - Basic: simple search box where search results are sorted by relevance
  - Advanced: interface with more options where user has more control over the collection

Very Large Data Bases

Personalization

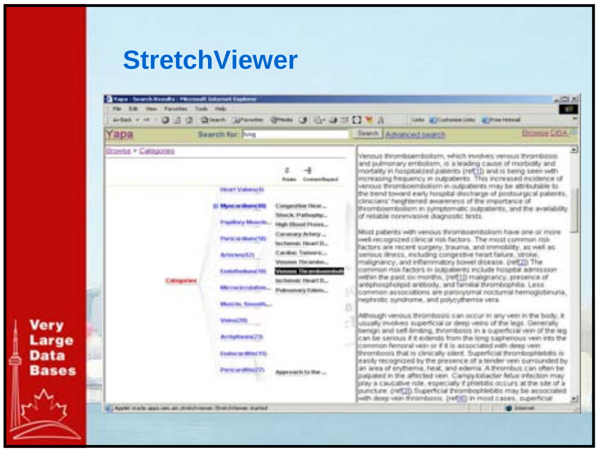
- Know user search patterns
  - What do they search?
  - When do they search?
- Search query log analysis
  - Which queries were made?
  - Which queries were successful?
  - How many times was each query made?

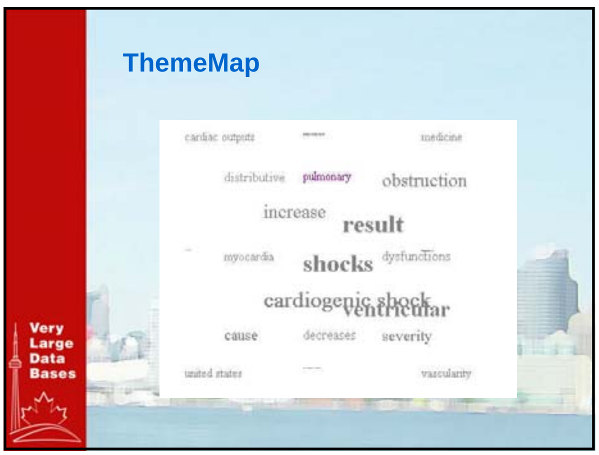
Advanced Features

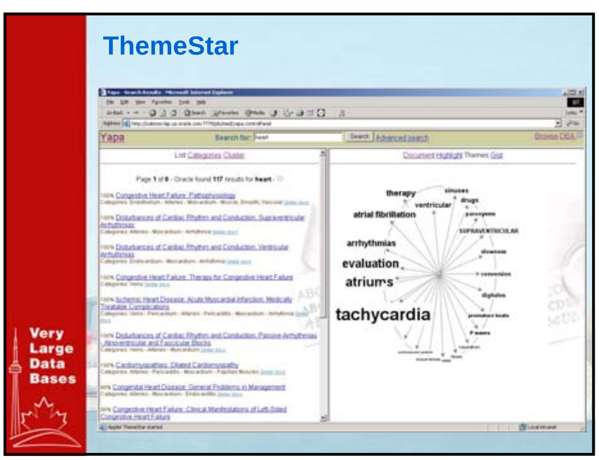
- Classification
  - Supervised classification of content
  - Two ways: rules or training sets
  - You can group a number of categories into a taxonomy
  - Very useful for defining a common vocabulary in an enterprise
- Clustering
  - Unsupervised classification of patterns into groups
  - The engine analyzes the document collection and outputs a set of clusters with documents on it
  - Very useful for *discovering* patterns or nuggets in collections
  - Could be used as a starting point when there is no taxonomy present

Information Visualization

- Present searched information in ways other than hit-lists
- Shows relationship across items in addition to satisfying query results
- Better IR using visual metaphors
- Very useful for
  - Navigation through large data sets
  - Discover relationships and associations between items
  - Focus + context tasks
- Number of visualizations available
  - StretchViewer
  - Interactive Viewer (ThemeMap, Cluster visualization)
  - Integration with 3rd party vendors


StretchViewer
Very
Large
Data
Bases


ThemeMap
cardiac outputs
medicine
distributive
pulmonary
obstruction
increase
result
myocardia
shocks
dysfunctions
cardiogenic shock
ventricular
cause
decreases
severity
united states
vascularity
Very
Large
Data
Bases


ThemeStar
therapy
ventricular
atrial fibrillation
arrhythmias
evaluation
atriums
tachycardia
Very
Large
Data
Bases

Is Oracle's Text Search Complex?

- Easy to Develop
  - Simple SQL and PL/SQL interface
    - Can be used by any developer that knows SQL
    - Can be called by any tool that knows SQL
    - Using any language: Java, JSP, PL/SQL, C, etc.
  - Choice of datastores
    - Stored in the database
    - Stored in the file system
    - Stored on the web (URL)
    - User-defined datastore
- Easy to Deploy
- Easy to Maintain

Very Large Data Bases

Oracle Text API

- Three index types
  - `context`: classic text searching
  - `ctxcat`: catalog searching
  - `ctxrule`: classification/routing applications
- Extensions to SQL
  - `select title from my_table where contains(text,'Java')>0;`
  - `select title from my_categories where matches(myquery, mydoc) > 0;`

Very Large Data Bases

Oracle Text API – II

- Operators: Boolean expressions, phrases, proximity, fuzzy, stemming, wildcards, accumulate scores, term weighting, XPath, etc.
- Packages
  - CTX_DOC: document services
  - CTX_QUERY: query feedback
  - CTX_REPORT: index information
  - CTX_OUTPUT: logging
  - CTX_THES: thesaurus features
  - CTX_CLS: training set
  - CTX_ADM: administration
  - CTX_DDL: create/manages index preferences, sections, stop lists

Very Large Data Bases

An IR Example: Verity Structured data

- Indexing databases
  - Used to import data from ODBC databases into Verity indexes ("collections")
  - Similar to Verity gateways to other backend repositories e.g., Lotus Notes, Exchange, Documentum, Filenet, etc.
- Parametric selection for search
  - Intersect full-text search with range queries/selection
  - When a field is a taxonomy (e.g., Continent/Country/City/Street), you have relational taxonomies = Cartesian product of taxonomies

Database indexing – 2 choices

- "Export" to XML or Bulk Insert File
- ODBC Gateway
- The common theme to either approach is to preserve the database structure in the index, such that you can query/display/sort on fields of integer, float, date, string, "attachment" data types.

"Export" to XML or BIF - Overview

- Many applications use a database as a storage component.
- Verity may not have an official gateway to that system because the APIs may not exist and/or a simpler solution exists.
- Sample list of applications that may be indexed using this approach
  - MatrixOne, Siebel, Interwoven, Fatwire, Virage, many others
- The general concept is to temporarily export the database row/field structure in a Verity compatible format.
- A variety of integration languages have been used – including, but not limited to ASP, Java/JSP/JDBC, Perl/ODBC, etc.

Verity Gateways

- Pre-built Gateways provide access to the most common enterprise repositories
- Gateway developer's kit enables you to build custom gateways to virtually any application
- K2 Enterprise enforces existing security models
  - Including native security of applications accessed by Verity Gateways
  - Ensures end-users can only view the information that they are authorized to access

Very Large Data Bases

Verity Gateways

Pre-built Verity Gateways

- Available for the following repositories:
  - Documentum
  - File Systems (NFTS and UNIX)
  - HTTP
  - Lotus Notes
  - Microsoft Exchange
  - ODBC databases

Verity Gateway Development Kit

- Quickly and easily build secure custom gateways to additional repositories

Very Large Data Bases

ODBC gateway

- Verity product that uses ODBC (Data-Direct drivers) to stream records from database into Verity collections.
- A graphical tool (MMC plug-in) is used to build the text-based configurations that control the desired mapping behavior.

Very Large Data Bases

ODBC GW - Certified Platforms

- Windows (with access to Oracle, DB2, Microsoft SQL Server)
- Solaris (with access to Oracle and DB2)
- AIX (with access to Oracle and DB2)
- HP-UX (with access to Oracle and DB2)
- Linux (with access to Oracle and DB2)

- Other databases such as Informix, Sybase, MySQL and others are supported
  - Gateway uses ODBC 3.5 API calls to insure compatibility

Very Large Data Bases

Feature Highlights

- SQL statements that select fields from one or more tables (gateway join)
- Full Data Type support
  - Blobs, unsigned/signed integers, floats, dates
  - Filebyname – treat field as file system path and automatically follow and index
- Multi-row records
- Compound primary keys
- Efficient spidering
  - Event-driven updates – use database triggers
  - Where clauses can be used for crawling limit

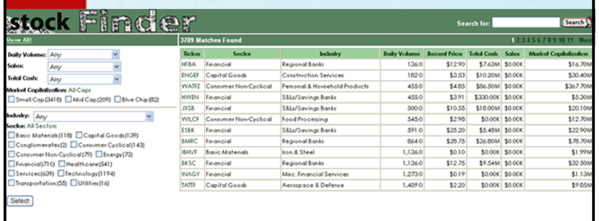
Very Large Data Bases

Verity K2 Enterprise Search - Parametric Selection

- Intuitive interface enables users to easily sort and filter information by selecting pre-set parameters and searching through filtered text fields and document content for specific text

Very Large Data Bases

Verity K2 Enterprise Search - Parametric Selection Example

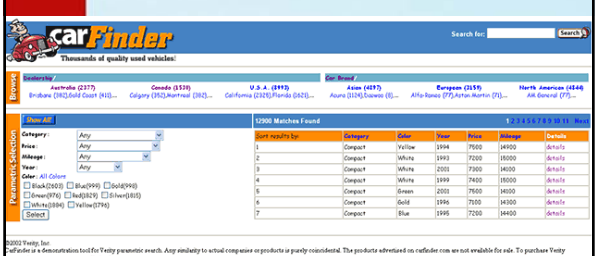


Verity K2 Enterprise Search - Relational Taxonomies

- Allows users to quickly narrow down information in the way that makes the most sense to them
  - Users take alternate paths through the same topics or categories to quickly and easily narrow down on the information they need
  - Users can navigate to information using two or more taxonomies at once
- Dramatically improve the finding experience for data with attributes

Very Large Data Bases

Verity K2 Enterprise Search - Relational Taxonomies Example



Agenda

1. Motivation
2. An Introduction to IR
3. Requirements for DB-IR
4. Semi-structured Data
5. Industrial DB-IR Examples: Oracle, Verity
6. DB Approaches
7. IR & Hybrid Approaches
8. Open Problems
9. Bibliography

Very Large Data Bases

6. DB Approaches

- IR on Relational Data
  - Keyword search
- IR on XML
  - Keyword search
  - Full QL + IR extension
  - Algebras and Evaluation

Very Large Data Bases

6-1. IR on Relational: Keywords

- BANKS
  - Gaurav Bhalotia, Arvind Hulgeri, Charuta Nakhe, Soumen Chakrabarti, S. Sudarshan, *Keyword Searching and Browsing in Databases using BANKS*, ICDE 2002
- DBXplorer
  - Sanjay Agrawal, Surajit Chaudhuri, Gautam Das, *DBXplorer: A System for Keyword-Based Search over Relational Databases*, ICDE 2002
- DISCOVER
  - Vagelis Hristidis, Yannis Papakonstantinou: DISCOVER, *Keyword Search in Relational Databases*, VLDB 2002

Very Large Data Bases

Keyword Search

- Keywords could be:
  - In the same tuple
  - In the same relation
  - In the Data or the Metadata
  - Connected through primary-foreign key relationships
- Results can be scored based on:
  - Distance of keywords within a tuple
  - Distance between keywords in # edges
  - IR-style ranking
  - Random walk probability (PageRank style)
  - Some combination of the above

Example Query [V. Hristidis]

Keywords: Smith Miller

ORDERS

	ORDERKEY	CUSTKEY	TOTALPRICE	CLERK	...
o_1	1000105	12312	$5,000	John Smith	
o_2	1000111	12312	$3,000	Mike Miller	
o_3	1000125	10001	$7,000	Mike Miller	
o_4	1000110	10002	$8,000	Keith Brown	

CUSTOMER

	CUSTKEY	NAME	NATIONKEY	...
c_1	12312	Brad Lou	01	
c_2	10001	George Walters	01	
c_3	10013	John Roberts	01	

NATION

	NATIONKEY	NAME	REGIONKEY
n_1	01	USA	N.America

Results:

Size	Result
2	$o_1 \leftarrow c_1 \rightarrow o_2$
4	$o_1 \leftarrow c_1 \leftarrow n_1 \rightarrow c_2 \rightarrow o_3$

Smaller sizes usually denote tighter association between keywords

6-2. IR on XML: Keywords

- XKeyword
  - V. Hristidis, Y. Papakonstantinou, A. Balmin, *Keyword proximity search on XML graphs*, ICDE 2003
  - A. Balmin, V. Hristidis, N. Koudas, Y. Papakonstantinou, D. Srivastava, T. Wang, *A System for Keyword Search on XML Databases,* VLDB 2003
- XSearch
  - S. Cohen, J. Mamou, Y. Kanza, Y. Sagiv, *XSearch: a semantic search engine for XML*, VLDB 2003
- XRANK
  - L. Guo, F. Shao, C. Botev, J. Shanmugasundaram, *XRANK: Ranked keyword search over XML documents*, SIGMOD 2003

XSearch Example

```
<proceedings>
  <inproceedings>
    <author>Moshe Y. Vardi</author>
    <title>Querying Logical Databases</title>
  </inproceedings>
  <inproceedings>
    <author>Victor Vianu</author>
    <title>A Web Odyssey: From Codd to
  XML</title>
  </inproceedings>
</proceedings>
```

The Content-Only Approach

Find papers by Vianu on the topic of "logical databases"

Search: Vianu logical databases

- Each document in the corpus is treated as a unit.
- A document containing some of the three query terms is considered as a result

The document contains the three query terms. Hence, it is returned by a standard search engine. BUT

```
<proceedings>
  <inproceedings>
    <author>Moshe Y. Vardi</author>
    <title>Querying Logical Databases</title>
  </inproceedings>
  <inproceedings>
    <author>Victor Vianu</author>
    <title>A Web Odyssey: From Codd to
          XML</title>
  </inproceedings>
</proceedings>
```

XQuery+FT Query Language

```
FOR $i IN document("bib.xml")//inproceedings
WHERE $i/author contains 'Vianu'
   AND $i/title contains 'Logical'
   AND $i/title contains 'Databases'
RETURN   <result>
             <author> $i/author </author>
            <title> $i/title </title>
          </result>
```

This does work, BUT

- Much more complicated query expression than search box
- Extensive knowledge of the document structure is required to write the query
- Still need to choose a mechanism for ranking the results

Requirements from the Search Tool

- A simple syntax that can be used by naive users
- Search results should include XML fragments and not necessarily full documents
- The XML fragments in an answer, should be semantically related
  - For example, a paper and an author should be in an answer only if the paper was written by this author
- Search results should be ranked
- Search results should be returned in "reasonable" time

XSEarch Query Syntax

- A *query* is a list of *query terms*
- A query term can be a
  - *Keyword*, e.g., `database`
  - *Tag*, e.g., `inproceedings:`
  - *Tag-keyword* combination, e.g., `author:Vianu`
- Optionally preceded by a '+'

The Example Revisited

- Find papers by Vianu on the topic of "logical databases"

logical +database inproceedings: author:Vianu

The keyword database must appear in the fragment

of Vianu under the
n the fragment,
increas increases the rank of this fragment

XSEarch: author:Vianu title:

```
<proceedings>
  <inproceedings>
    <author>Moshe Y. Vardi</author>
    <title>Querying Logical Databases</title>
  </inproceedings>
  <inproceedings>
    <author>Victor Vianu</author>
    <title>A Web Odyssey: From Codd to XML</title>
  </inproceedings>
</proceedings>
```

Good Result!

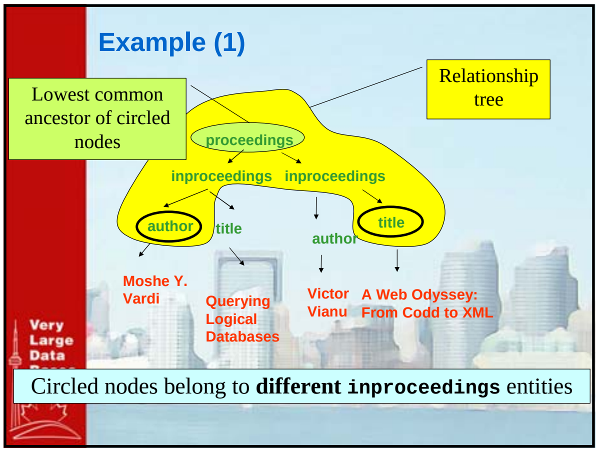
title and author elements ARE semantically related

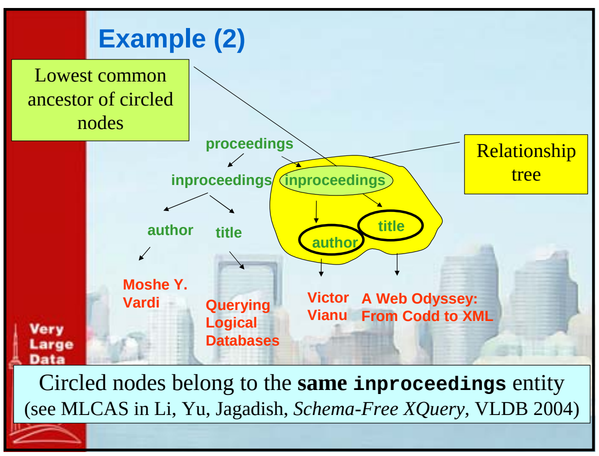
XSEarch: author:Vianu title:

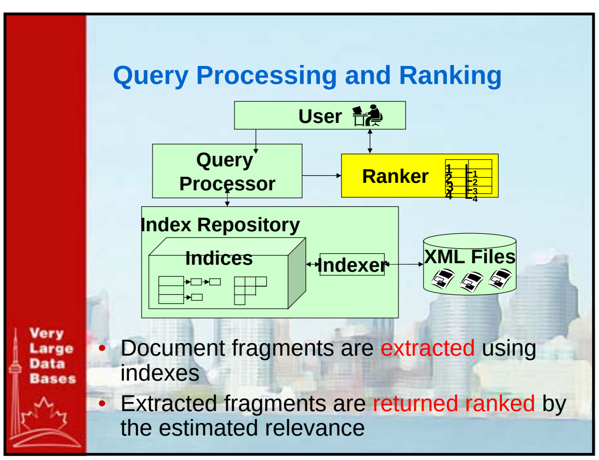
```
<proceedings>
  <inproceedings>
    <author>Moshe Y. Vardi</author>
    <title>Querying Logical Databases</title>
  </inproceedings>
  <inproceedings>
    <author>Victor Vianu</author>
    <title>A Web Odyssey: From Codd to XML</title>
  </inproceedings>
</proceedings>
```

Bad Result!

title and author elements ARE NOT semantically related


Example (1)
Lowest common ancestor of circled nodes
Relationship tree
proceedings
inproceedings
inproceedings
author
title
author
title
Moshe Y. Vardi
Querying Logical Databases
Victor Vianu
A Web Odyssey: From Codd to XML
Very Large Data
Circled nodes belong to **different inproceedings** entities


Example (2)
Lowest common ancestor of circled nodes
Relationship tree
proceedings
inproceedings
inproceedings
author
title
author
title
Moshe Y. Vardi
Querying Logical Databases
Victor Vianu
A Web Odyssey: From Codd to XML
Very Large Data
Circled nodes belong to the **same inproceedings** entity
(see MLCAS in Li, Yu, Jagadish, *Schema-Free XQuery*, VLDB 2004)


Query Processing and Ranking
User
Query Processor
Ranker
Index Repository
Indices
Indexer
XML Files
Very Large Data Bases
• Document fragments are extracted using indexes
• Extracted fragments are returned ranked by the estimated relevance

Result Ranking

Several factors increase the rank of a result

- Similarity between query and result
- Weight of labels appearing in the result
- Characteristics of result tree

TF-ILF

- – Extension of TF-IDF, classical in IR
- – Term Frequency: number of occurrences of a query term in a fragment
- – Inverse Leaf Frequency: number of leaves containing a query term divided by number of leaves in the corpus

Very Large Data Bases

TF-ILF

- Term frequency of keyword k in a leaf node n_l

$$tf(k, n_l) := \frac{occ(k, n_l)}{\max\{occ(k', n_l) \mid k' \in words(n_l)\}}$$

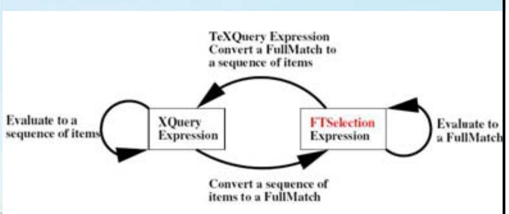
- Inverse leaf frequency

$$ilf(k) := \log\left(1 + \frac{|N|}{|\{n' \in N \mid k \in words(n')\}|}\right)$$

TF-ILF is the product between *tf* and *ilf*

Very Large Data Bases

6-2. IR on XML: TeXQuery



- Composability: conversion back and forth from FullMatch to XQuery data model (within TexQuery expression)

Very Large Data Bases

TeXQuery Expressions

- Contains

FTContainsExpr::= ContextExpr "ftcontains" FTSelection
returns true if a node in ContextExpr satisfies FTSelection

```
//book[
 .//section ftcontains ("usability" && "software")
 ]/title
```

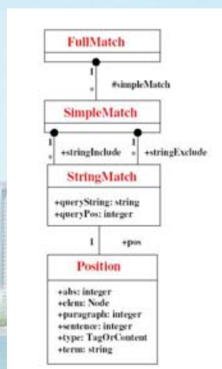
- Score

FTScoreExpr::= ContextExpr "ftscore" FTWeightedSelection
returns a sequence of scores (for ranking and top-k)

```
//book ftscore ("usability" weight 0.8
                && $i/topic weight 0.2)
```

Very Large Data Bases

TeXQuery Full-Text Model



Very Large Data Bases

QL-IR Design Choices

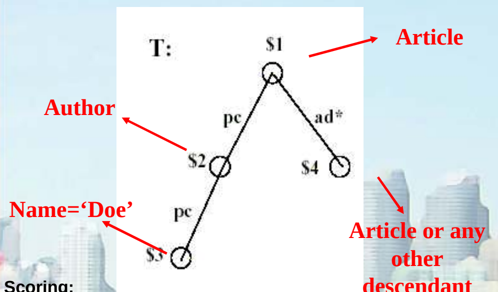
- SQL/MM structured text proposal
  - L. Brown, M. Consens, I. Davis, C. Palmer, F. Tompa, *A Structured Text ADT for Object-Relational Databases*, Theory and Practice of Object-Systems 1998
  - Functions have IR sublanguage as an argument, so the expression string can be constructed as a query
  - Explicit mark_subtexts() function supports highlighting matches
- TeXQuery
  - IR sublanguage grammar exposed and fully composable with XQuery
  - Implementation defined positions and scores

Very Large Data Bases

6-2. IR on XML: TIX Algebra

- TIX is an extension of the bulk XML algebra TAX that manipulates collections of *scored trees* with matching defined via *scored pattern trees*
  S. Al-Khalifa, C. Yu, H. Jagadish, *Querying structure text in an XML database*, SIGMOD 2003
- Find document components in articles that
  - Are part of an article written by an author with last name "Doe" and are about "search engine"
  - Relevance to "internet" and "information retrieval" is desirable (but not necessary)

Example Scored Pattern

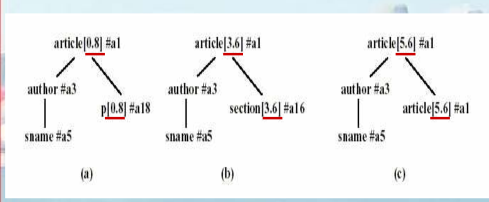


Scoring:
$4.score = ScoringFunction(
{"search engine"}, {"internet", "information retrieval"})
$1.score = $4.score

Scored Selection

Scored pattern tree (*p*)

Scored data trees *C* ⇨ $\sigma'_{\rho}(C)$ ⇨ **Scored data trees ~ *p***



Scored Projection

Scored data trees C ⇨ $\Pi'_{p,\,PL}(C)$ ⇨ Scored data trees ~ C

⇧ A scored pattern tree (p)

⇧ A projection list (PL)

- Combine multiple scores (from multiple pattern matches) by keeping the maximum

Scored Joins

- Find relevant document components in articles as before
- For articles containing such components, find the reviews with similar titles

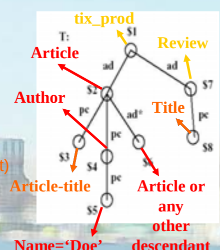
Scoring:

$6.score = ScoringFunction({"search engine"}, {"internet", "information retrieval"})

$2.score = $6.score

$joinScore = ScoreSim($3.content, $8.content)

$1.score = ScoreBar($joinScore, $6.score)



IR-style Operations

- Threshold
  - Projection that retains input trees where at least one node has a top-k score, or a score higher than a threshold
- Pick
  - Projection that uses a condition with functions that can traverse the tree to remove redundant answers
- Operations implemented using stack-based algorithms on regions

Query Evaluation with Relevance

R. Fagin, A. Lotem, M. Naor, *Optimal aggregation algorithms for middleware*, JCSS 2003 (Garlic System 1995)

- Threshold Algorithm
  - Given m sorted lists with object rankings
  - Aggregate the rankings from each list for each object
  - Return the top k ranked objects
  - Instance Optimal Solution: do sorted access (and the corresponding random access) until you know you have seen the top k answers
- IR Application: objects are document (fragments) and each list has the relevance of each document for a given keyword

Agenda

1. Motivation
2. An Introduction to IR
3. Requirements for DB-IR
4. Semi-structured Data
5. Industrial DB-IR Examples: Oracle, Verity
6. DB Approaches
7. IR & Hybrid Approaches
8. Open Problems
9. Bibliography

7. Hybrid & IR Approaches

- Overview of Approaches
- Retrieval Models
- Indexing
- INEX
- Ranking XML

Overview of Approaches

- RBD + IR: Two different APIs
- RDB + IR Hybrid: QUIQ, MOA, HySpirit, ...
- RBD “text search” accelerator
  - Text content is transformed to flat XML
  - XML is searched using an IR API
  - Results can be later combined with SQL
- IR System with SQL support
  - Special indexes for atomic data types
- XML Databases
  - Atomic data types as attributes (metadata)
  - Implementation on top of structured text models?

QUIQ (Kabra *et al*, 2003)

- Tuple: <tag-name, tag-type, tag-value>
- Query: *match-filter-quality*
  - Result: AND of *match* & *filter*
  - *Match* are approximate constraints
  - *Filter* are exact constraints
  - Relevance is adjusted by *quality*
- Indexing: built on top of a RDBMS
  - Non-text data is mapped to pseudo-words
  - Unified index & common TF-IDF model
  - Deferred update operations
- Evaluation: 60% faster than a RDBMS text extension

Retrieval Models

- Relational Model: DB2XML, XML-QL, TSIMMIS, LOREL
- Object-oriented Model: SOX, StruQL, …
- Extended Vector Model
- Weighted Boolean Model: XQL, …
- Probabilistic Model: XIRQL, ELIXIR, JuruXML, ...

Indexing

- Flat File: add information, SQL accelerators,...
- Semi-structured:
  - Field based: no overlapping, Hybrid model,..
  - Segment based: Overlapped list, List of references, p-strings
  - Tree based: Proximal Nodes, XRS, ...
- Structured:
  - IR/DB, Path-based, Position-based, Multidimensional
- Indexes:
  - Structure + Value index (XML on top of RDBs):
    - Toxin, Dataguides, T-indexes, Index Fabric, etc.
  - Integrated Full-text and Structure index:
    - Proximal Nodes, Region Algebra, String Indexing, ...

XPath over Proximal Nodes (Navarro & Ortega, 2003)

- A fast implementation of XPath subset
- Maps XPath expressions into Proximal Nodes algebra
- Format translation of Axes
- Node + Text index
- Lazy evaluation

Query	IXPN	Xind	eXist	Grep	Saxon	MS	Toxin
/tstmt/bookcoll/book/chapter	1.8	20.5	8.8	3.4	4.0	3.3	2.5
/tstmt/coverpg/coverpg[titlel]	0.5	2.8	2.2	0.7	3.3	1.3	-
/tstmt[//chapter	1.8	58.9	8.8	3.8	4.1	3.2	2.5
/tstmt[//chapter]	0.9	22.7	8.8	3.7	4.0	4.2	-
v[.=~"love"]	0.4	9.9	9.8	0.7	3.4	1.8	3.7
/tstmt[/coverpg/title /following-silbling::subtitle	0.5	2.6	9.8	0.7	3.3	1.3	-

INEX

- Initiative for the Evaluation of XML
- Three types of tasks:
  - Content only search
  - Content & Structure Search
  - Clustering
- Started in 2002
- Cooperative relevance assesment
- About 40 groups per year

Ranking XML

- Content only:
  - exploit hierarchical structure
  - exploit importance of tags
- Content & structure:
  - Query languages with uncertainty & vagueness
  - Data types with vague predicates
  - Strict & fuzzy structural conditions
  - Dynamic *tf* × *idf*

Very Large Data Bases

Integrated IR (Bremer & Gertz)

- Extension to XQuery
- Based on XML fragments
- Schemas are extended DataGuides
  - Enumeration of all rooted label paths
- Ancestor relationships from structural joins
- RANKBY operator
  - based on local & dynamic *tf-idf*
- New node enumeration encoding
- Path & term-index
  - Other smaller indexes (in total less than 60%)
- More than 10 times faster than other XQuery prototypes

Very Large Data Bases

Agenda

1. Motivation
2. An Introduction to IR
3. Requirements for DB-IR
4. Semi-structured Data
5. Industrial DB-IR Examples: Oracle, Verity
6. DB Approaches
7. IR & Hybrid Approaches
8. Open Problems
9. Bibliography

Very Large Data Bases

8. Open Problems

- Heterogenous data
- Ranking tuples & XML
- New retrieval models
- DB issues for documents
- Simple/succinct vs. complex/verbose QL
  - Define an XQuery core?
- Optimization and algebras
- Efficient algorithms
- Indexing & searching
- Quality evaluation (Web, XML)

Very Large Data Bases

Thank You

1. Motivation
2. An Introduction to IR
3. Requirements for DB-IR
4. Semi-structured Data
5. Industrial DB-IR Examples: Oracle, Verity
6. DB Approaches
7. IR & Hybrid Approaches
8. Open Problems
9. Bibliography

Come to SIGIR 2005, Salvador, Bahia, Brazil (August)

Very Large Data Bases

9. Bibliography – 1

- Baeza-Yates & Ribeiro-Neto, Modern Information Retrieval, Addison-Wesley, 1999.
- Baeza-Yates & Navarro, Integrating contents and structure in text retrieval, SIGMOD 25 (1996), 67-79.
- Baeza-Yates and Navarro, XQL and Proximal Nodes, JASIST 53, 504--514, 2002.
- Baeza-Yates, Carmel, Maarek, and Sofer, editors. Special issue on XML Retrieval, JASIST, 53, 2002.
- Baeza-Yates, Fuhr, and Maarek, editors. Proceedings of the SIGIR 2002 Workshop on XML and Information Retrieval.
- Bremer & Gertz, Integrating Document & Data Retrieval Based on XML, to appear.
- Chinenyanga and Kushmerik, Expressive retrieval from XML documents, Proc. of the 24th SIGIR, 163-171, New York, 2001.
- Delgado & Baeza-Yates, A Comparison of XML Query Languages, Upgrade 3, 12-25, 2002.

Very Large Data Bases

Bibliography - 2

- Fuhr and Grossjohann , XIRQL: An XML query language based on IR concepts. ACM TOIS 22, 313--356, 2004.
- Fuhr, Govert, Kazai, and Lalmas, editors. INitiative for the Evaluation of XML Retrieval. Proceedings of the First INEX Workshop. Dagstuhl, Germany, Dec., 8--11, 2002
- Fuhr, Lalmas, and Malik, editors. Proc. of the Second INEX Workshop. Dagstuhl, Germany, Dec. 15--17, 2003, 2004.
- Grabs and Schek, Flexible information retrieval from XML with PowerDB-XML, In INEX 2003, 141-148.
- Kabra, Ramakrishnan, Ercegovac, The QUIQ Engine: A Hybrid IR DB System, ICDE 2003.
- Luk, Leong, Dillon,Chan, Croft & Allan, A Survey on Indexing and Searching XML, "Special Issue on XML and IR", *JASIST*, 2002.
- Mass, Mandelbrod, Amitay, and Soffer, JuruXML - an XML retrieval system at INEX 2002. In INEX 2003, 73-90.

Very Large Data Bases

Bibliography - 3

- Mihajlovic, Hiemstra, Block & Apers, An XML-IR-DB Sandwich: Is it better with an Algebra in between?, I Workshop on DB-IR integration at SIGIR, 2004.
- Navarro and Baeza-Yates, Proximal Nodes, SIGIR 1995 (journal version in ACM TOIS, 1997).
- Navarro and Ortega, IXPN: An index-based XPath implementation, Technical Report, U. de Chile, 2003.
- Piwowarski, Vu, and Gallinari. Bayesian networks and INEX 2003. In INEX 2004.
- Sayyadian, Shakery, Doan & Zhai, Toward Entity Retrieval over Structured and Text Data, I Workshop on DB-IR integration at SIGIR, 2004.

Very Large Data Bases

Bibliography - 4

- S. Amer Yahia, M. Fernandez, D. Srivastava, Y.Xu, Phrase Matching in XML, VLDB 2003
- D. Florescu, D. Kossmann, I. Manolescu, Integrating Keyword Search into XML Query Processing, WWW 2000
- R. Goldman, N. Shivakumar, S. Venkatasubramanian, H. Garcia-Molina, Proximity Search in Databases, VLDB 1998
- A. Theobald, G. Weikum, The index-based XXL search engine for querying XML data with relevance ranking, EDBT 2002
- S. Al-Khalifa, C. Yu, H. Jagadish, *Querying structure text in an XML database*, SIGMOD 2003

Very Large Data Bases

Bibliography - 5

- K. Böhm , K. Aberer , E. Neuhold , X. Yang, Structured document storage and refined declarative and navigational access mechanisms in HyperStorM, The VLDB Journal 1997
- E. Brown, Fast evaluation of structured queries for information retrieval, SIGIR 1995
- S. Amer-Yahia, S. Cho, D. Srivastava, Tree pattern relaxation, EDBT 2002
- S. Amer-Yahia, L. Lakshmanan, S. Pandit, FleXPath: flexible structure and full-text querying for XML, SIGMOD 2004
- S. Amer-Yahia, N. Koudas, D. Srivastava, Approximate Matching in XML, ICDE 2003 Tutorial

Very Large Data Bases

Bibliography - 6

- G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, S. Sudarshan, *Keyword Searching and Browsing in Databases using BANKS*, ICDE 2002
- S. Agrawal, S. Chaudhuri, G. Das, *DBXplorer: A System for Keyword-Based Search over Relational Databases*, ICDE 2002
- V. Hristidis, Y. Papakonstantinou: DISCOVER, *Keyword Search in Relational Databases*, VLDB 2002
- V. Hristidis, Y. Papakonstantinou, A. Balmin, *Keyword proximity search on XML graphs*, ICDE 2003
- A. Balmin, V. Hristidis, N. Koudas, Y. Papakonstantinou, D. Srivastava, T. Wang, *A System for Keyword Search on XML Databases,* VLDB 2003
- S. Cohen, J. Mamou, Y. Kanza, Y. Sagiv, *XSearch: a semantic search engine for XML*, VLDB 2003
- L. Guo, F. Shao, C. Botev, J. Shanmugasundaram, *XRANK: Ranked keyword search over XML documents*, SIGMOD 2003

Very Large Data Bases