EIM
ENTERPRISE INFORMATION MANAGEMENT CONFERENCE
TORONTO, CANADA • SEPTEMBER 21 - 23, 2010

# Structured Search

Version 4

Dan McCreary
President
Dan McCreary & Associates
dan@danmccreary.com
(952) 931-9198

# Presentation Description

Today most document search systems are based on simple keyword search. Their strategy is to extract a list of keywords from documents and use these keywords to match a user’s query.

But the keyword extraction process frequently discards one of the valuable pieces of document information: the **context** of the keyword. Context allows you to rank a keyword in a title higher than a keyword in the body of a text. This allows relevant documents to receive a higher ranking than other documents in a large document collection. Retaining document structure with keywords allows context to be preserved in the search and retrieval process and allows sub-documents to receive customized search ranking rules.

This presentation give a broad overview of structured search and describes how organizations are using document structure to create a better search experience for corporate search.

# After This Presentation Users Will Be Able To:

- Identify the differences between keyword-only and structured search
- Understand the benefits and costs of structured search
- Understand how structured search improves search precision and relevancy
- Describe typical business situations where structured search has benefits
- Describe how non-technical business users can create customized search solutions
- Understand how products and vendors are implementing structured search solutions

# Background for Dan McCreary

- Enterprise data architecture consultant
- Builds open source metadata registries using (ISO/IEC 11179) and US Federal XML standards (NIEM.gov)
- Interested in Semantic Web technologies
- Coined the term "XRX" (XForms-REST-XQuery)
- Was looking for a better search tool for metadata registries
  - "what you can't find gets duplicated…"

# How Many People…

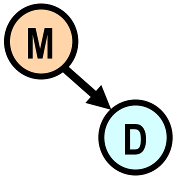
- Have heard of a company called "Google"?
- Think that "search" is one of the most important technologies in Enterprise Information Management (EIM)?
- Think that most corporations have a high-quality search on their Intranet?
- Would like to go to a **single** web search page and perform a search on both documents **and** data?
- Have ever heard the phrase "a little bit of semantics goes a long way"?

# Structured Search

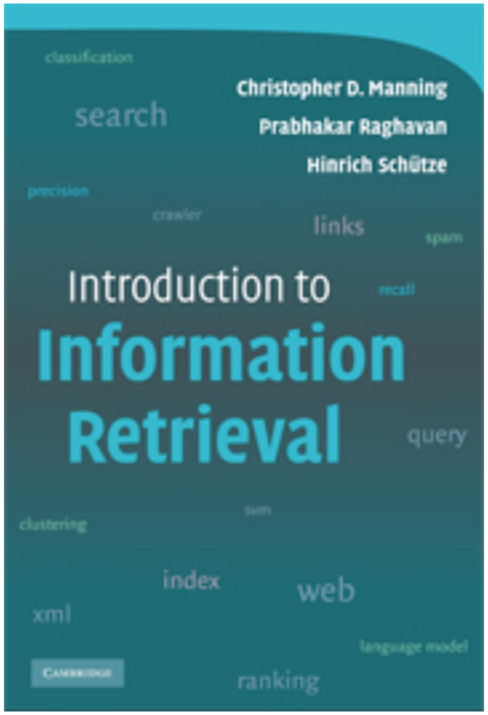
## Definition:

A method of using the **structure** of a document to help users find the right documents in a large collection of documents.

**But first a story……**



# Information Retrieval Textbook



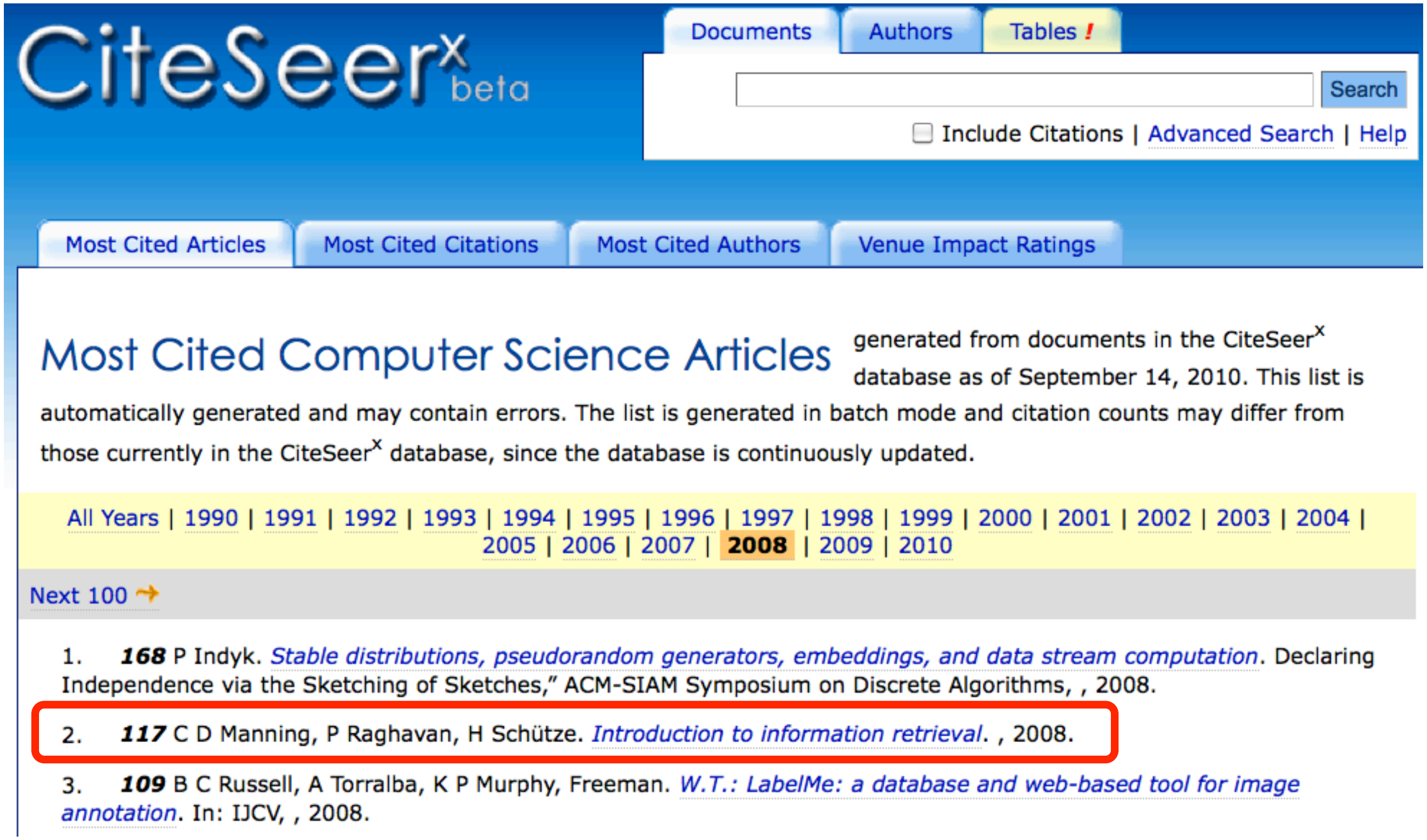
**Introduction to Information Retrieval**

by Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze

Cambridge University Press, 2008

http://nlp.stanford.edu/IR-book/information-retrieval-book.html

# 117 Citations in Computer Science



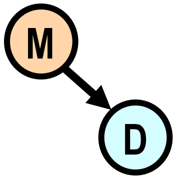
## Most Cited Computer Science Articles

generated from documents in the CiteSeer$^{x}$ database as of September 14, 2010. This list is automatically generated and may contain errors. The list is generated in batch mode and citation counts may differ from those currently in the CiteSeer$^{x}$ database, since the database is continuously updated.

All Years | 1990 | 1991 | 1992 | 1993 | 1994 | 1995 | 1996 | 1997 | 1998 | 1999 | 2000 | 2001 | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 | **2008** | 2009 | 2010

Next 100 →

1. ***168*** P Indyk. *Stable distributions, pseudorandom generators, embeddings, and data stream computation*. Declaring Independence via the Sketching of Sketches," ACM-SIAM Symposium on Discrete Algorithms, , 2008.
2. ***117*** C D Manning, P Raghavan, H Schütze. *Introduction to information retrieval*. , 2008.
3. ***109*** B C Russell, A Torralba, K P Murphy, Freeman. *W.T.: LabelMe: a database and web-based tool for image annotation*. In: IJCV, , 2008.



With 117 citations, the "Intro to IR" book is the second most cited Computer Science reference published in 2008.

# Table 10.1

| | RDB search | unstructured retrieval | structured retrieval |
|---|---|---|---|
| objects | records | unstructured documents | trees with text at leaves |
| model | relational model | vector space & others | ? |
| main data structure | table | inverted index | ? |
| queries | SQL | free text queries | ? |

**XML - Table 10.1 and structured information retrieval. SQLRDB (relational database) search, unstructured information retrieval**

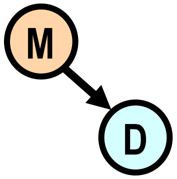
# Excerpt from IR Book…

*There is no consensus yet as to which methods work best for structured retrieval although many researchers believe that XQuery will become the standard for structured queries.*

- published 2008
- two years later…we have the answer...

# eXist Native XML Developers

eXist Meeting Prague March 12th, 2010



# Presentation Outline

- Background
  - Boolean (SQL) search
  - Keyword (Vector) search
  - Reverse indexes
- Structured search
  - How it works
  - Benefits and challenges
  - How to empower non-programmers
- How to setup up structured search pilot project
  - Resources for getting started
  - eXist case studies

# Table 10.1 - Revised

| | RDB search | unstructured retrieval | structured retrieval |
|---|---|---|---|
| objects | records | unstructured documents | trees with text at leaves |
| model | relational model | vector space & others | **XML hierarchy** |
| main data structure | table | inverted index | **trees with node-ids for document ids** |
| queries | SQL | free text queries | **XQuery fulltext** |

**XML - Table 10.1 and structured information retrieval. SQLRDB (relational database) search, unstructured information retrieval**

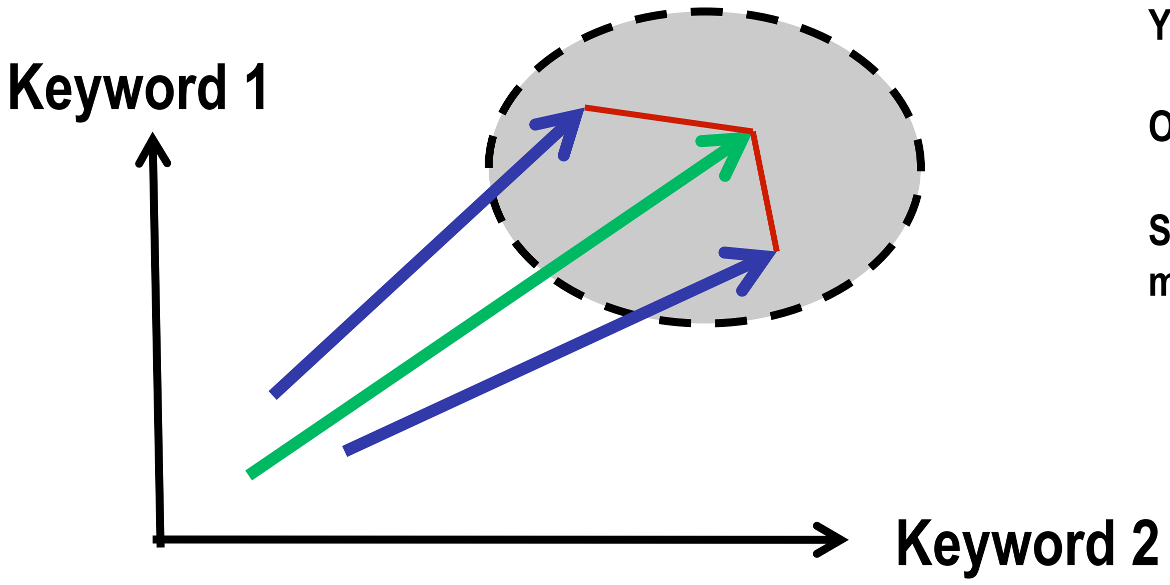
# Relational DB Boolean Search

- Exact match on tabular data
- No "score" associated with each "hit"
- Example from SQL:

```
SELECT * FROM PERSON
WHERE TITLE = 'manager'
ORDER BY SALARY
```

Note that the "order" is not the quality of a mach but another column in a table

# Vector Model



Your search keyword (green)

Other documents (blue)

Search score is distance measurement (red)

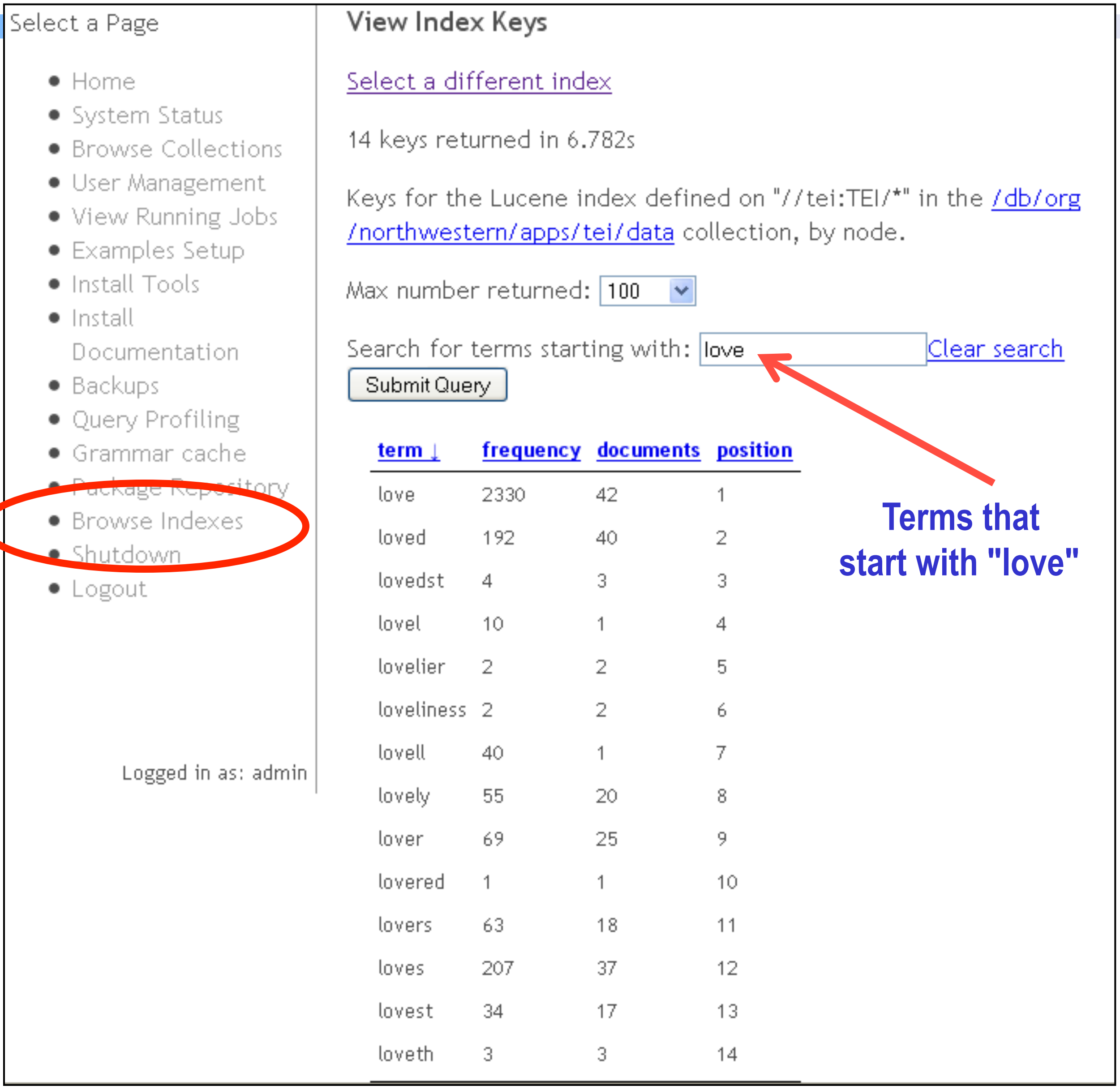
- Every search query is a "vector" in keyword space
- Distance from your query to documents are "scored"

# Reverse Index

| Word | Document IDs |
|---|---|
| **hate** | 12344, 34235, 43513, |
| **love** | 12344, 34235, 43513, 22345, 12313, 42345, 12313, 13124 |

**For each word, a reverse index tells you what documents contain that word.**

# Reverse Index in eXist 1.5



View Index Keys

Select a different index

14 keys returned in 6.782s

Keys for the Lucene index defined on "//tei:TEI/*" in the /db/org/northwestern/apps/tei/data collection, by node.

Max number returned: 100

Search for terms starting with: love Clear search

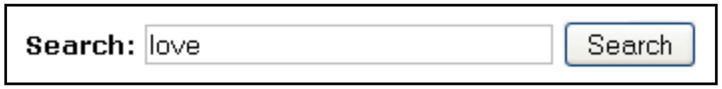
Submit Query

| term ↓ | frequency | documents | position |
|---|---|---|---|
| love | 2330 | 42 | 1 |
| loved | 192 | 40 | 2 |
| lovedst | 4 | 3 | 3 |
| lovel | 10 | 1 | 4 |
| lovelier | 2 | 2 | 5 |
| loveliness | 2 | 2 | 6 |
| lovell | 40 | 1 | 7 |
| lovely | 55 | 20 | 8 |
| lover | 69 | 25 | 9 |
| lovered | 1 | 1 | 10 |
| lovers | 63 | 18 | 11 |
| loves | 207 | 37 | 12 |
| lovest | 34 | 17 | 13 |
| loveth | 3 | 3 | 14 |

Terms that start with "love"

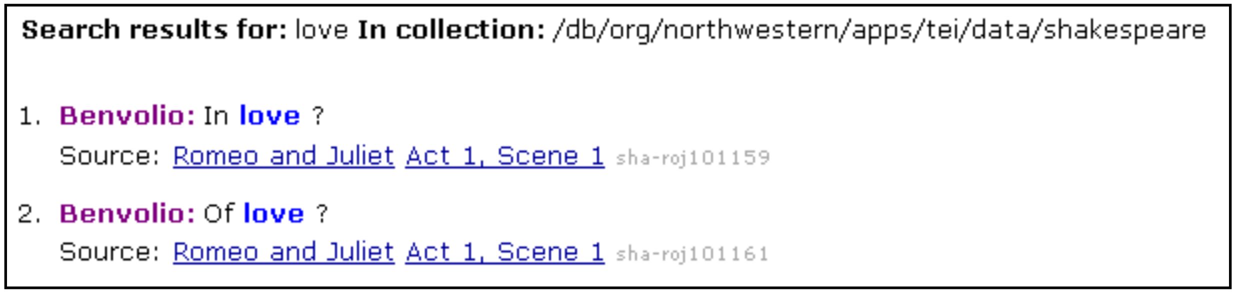
# Sample Keyword Search

## Keyword Search:



**Search:** love

## Resulting Hits:

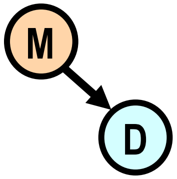


**Search results for:** love **In collection:** /db/org/northwestern/apps/tei/data/shakespeare

1. **Benvolio:** In **love** ?
   Source: Romeo and Juliet Act 1, Scene 1 sha-roj101159
2. **Benvolio:** Of **love** ?
   Source: Romeo and Juliet Act 1, Scene 1 sha-roj101161

## Code (XQuery):

```
$hits := collection($data-collection)/tei:TEI//tei:l[ft:query(., $q)]
 for $hit in $hits
     let $score := ft:score($hit)
     order by $score descending
         return
             <li class="hit-result">
```



# Calculating Score

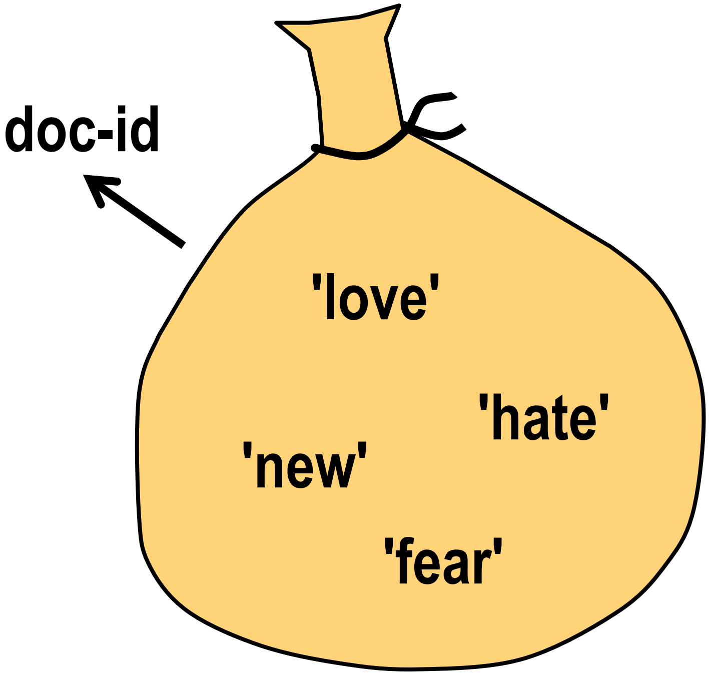
- Count the number of times the keywords appeared in each of the documents
- Calculate the size of the documents
- Calculate the "density" of the keywords in the documents
  - this prevents longer documents from getting higher scores
- Sort by the combined density numbers

# How is "Structured Search" Different?

- Most search engines today perform simple "keyword searches" and rank the scores based on keyword density
- In simple "keyword search" there is no consideration about where the keywords appear in the document
- There is no way to "boost" the search score if a keyword appears in a title, abstract or summary

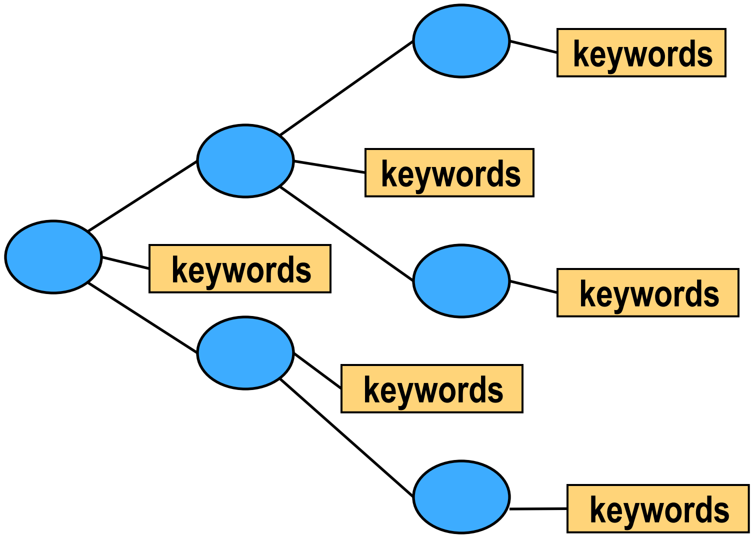
# Two Models

## "Bag of Words"



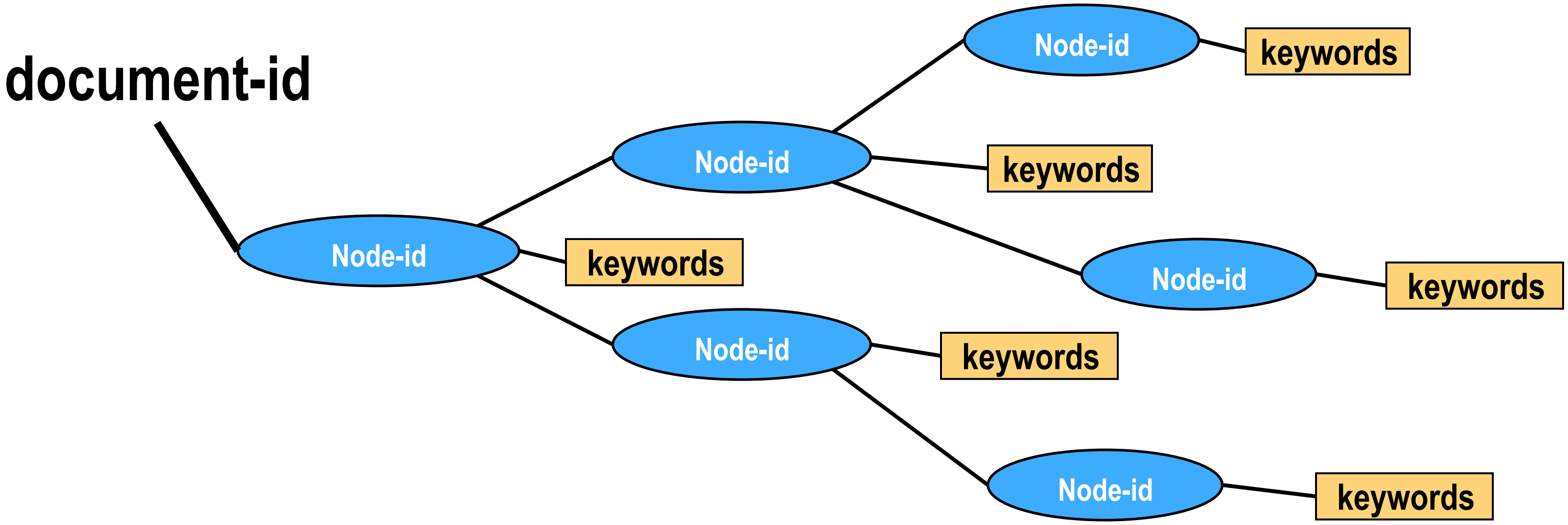
- All keywords in a single container
- Only count frequencies are stored with each word

## "Retained Structure"



- Keywords associated with each sub-document component

# Keywords and Node IDs

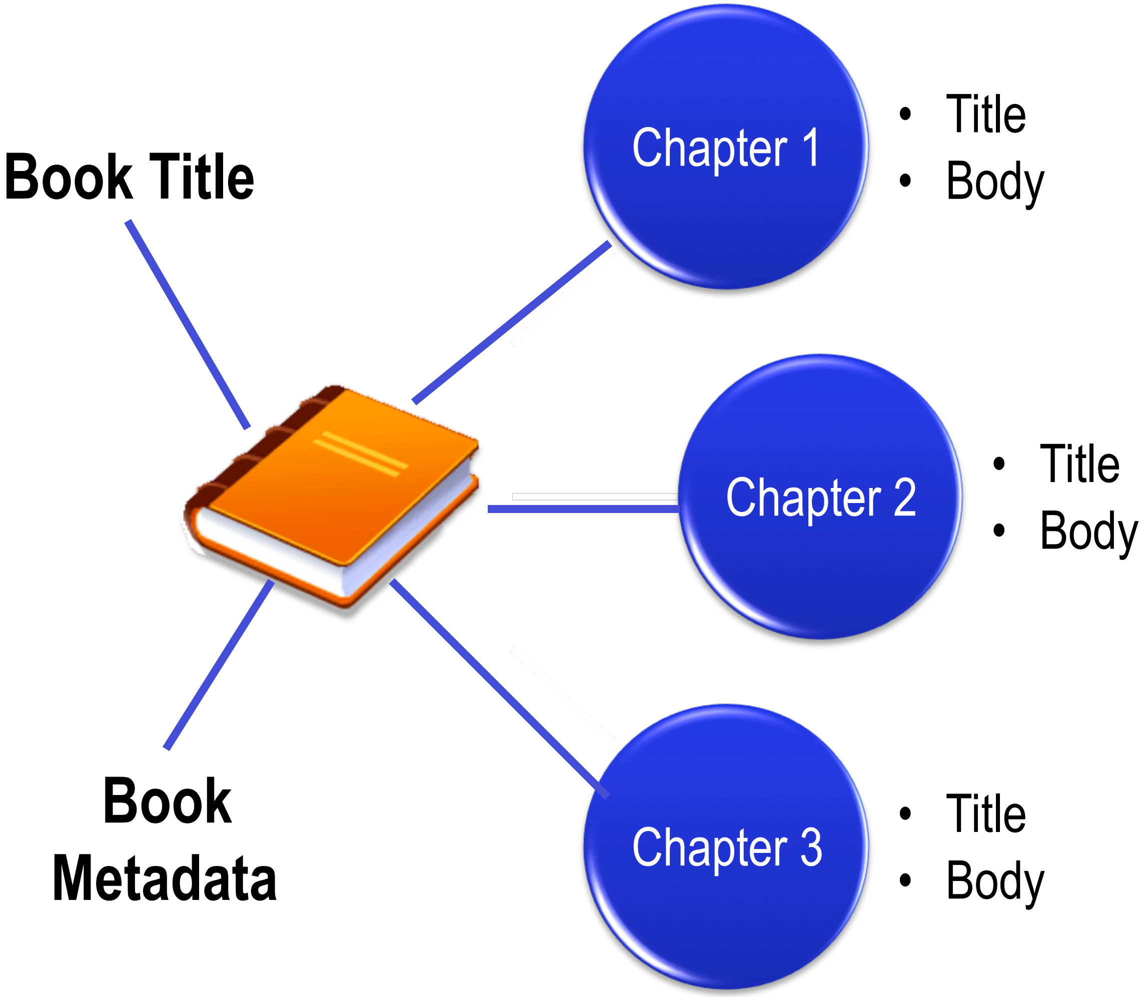


- Keywords in the reverse index are now associated with the **node-id** in every document

# Subdocuments

- Large documents are "decomposed" into many separate documents
- Each subdocument is given its own document identifier (node-id)
- Rules are set up to "score" hits on different components of the documents
- But do most documents have structure?

# Books Have Structure



Book Title

Book Metadata

Chapter 1

- Title
- Body

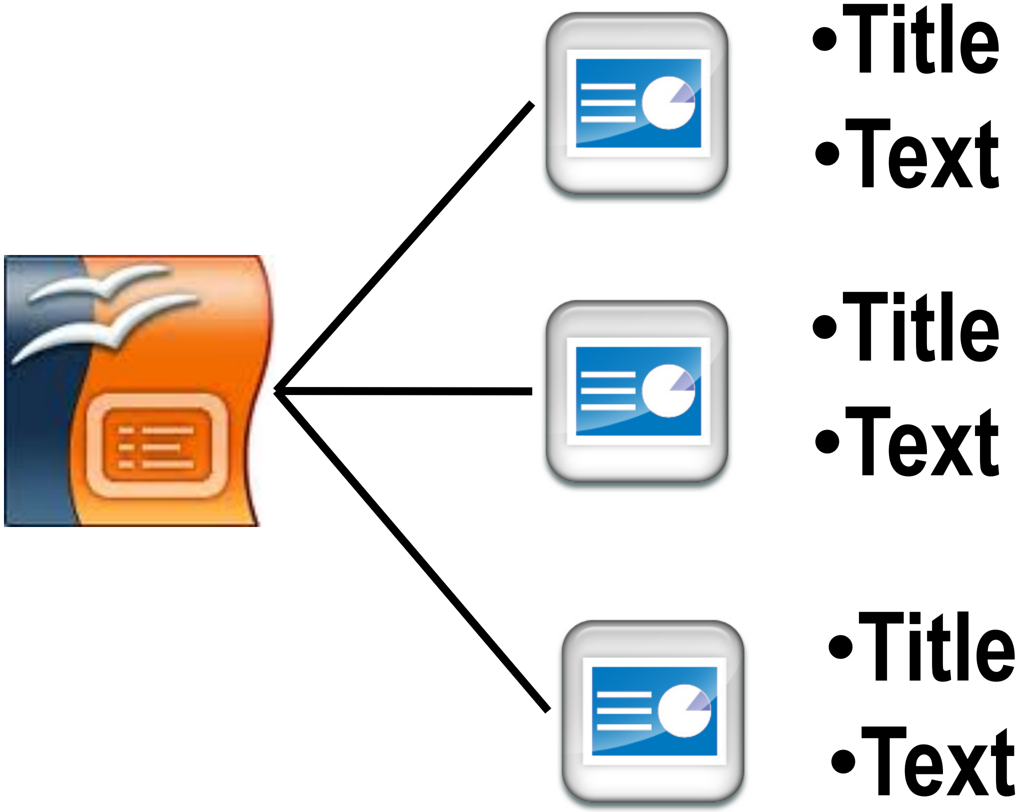
Chapter 2

- Title
- Body

Chapter 3

- Title
- Body

# Presentations Have Structure



- Title
- Text

- Title
- Text

- Title
- Text

Find all slides with the word "XML" in their title

# E-mail has structure

**From:** John Smith

**To:** All IT Staff

**Subject:** Findings of Search Survey

We just got the results back from our survey on intranet search effectiveness.  Although some users were satisfied most users said that it took them over 15 minutes to find a documents they were looking for. That means that our firm spends over $1,000,000 each year in lost productivity due to our lack of good search tools on our internal documents….

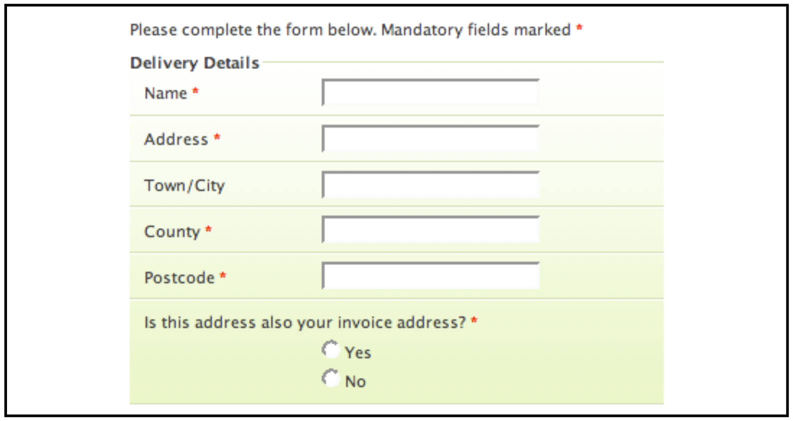
# Sample of XML

```
<PLAY>
        <TITLE>The Tragedy of Romeo and Juliet</TITLE>
        <PERSONAE>
           <TITLE>Dramatis Personae</TITLE>
           <PERSONA>ESCALUS, prince of Verona. </PERSONA>
           <PERSONA>PARIS, a young nobleman, kinsman to the prince.</PERSONA>
           <PGROUP>
              <PERSONA>MONTAGUE</PERSONA>
              <PERSONA>CAPULET</PERSONA>
              <GRPDESCR>heads of two houses at variance with each other.</GRPDESCR>
           </PGROUP>
           <PERSONA>An old man, cousin to Capulet. </PERSONA>
           <PERSONA>ROMEO, son to Montague.</PERSONA>
           <PERSONA>MERCUTIO, kinsman to the prince, and friend to Romeo.</PERSONA>
           <PERSONA>BENVOLIO, nephew to Montague, and friend to Romeo.</PERSONA>
           <PERSONA>TYBALT, nephew to Lady Capulet.</PERSONA>
```

# Many Objects Have Structure

Spreadsheets

Find all spreadsheets with a first row cell that contains the word "SSN"



Please complete the form below. Mandatory fields marked *

**Delivery Details**

Name *

Address *

Town/City

County *

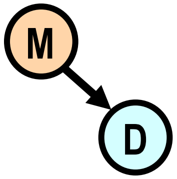
Postcode *

Is this address also your invoice address? *

○ Yes

○ No

Find all forms with a label of "Zipcode"



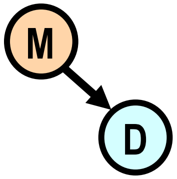
# But What About Microsoft Office?

*Structured search will only take off when our standardize office documents use XML…*

**ECMA-376**

Office Open XML (also informally known as OOXML or OpenXML) is a zipped, XML-based file format developed by Microsoft  for representing spreadsheets, charts, presentations  and word processing documents.

File extensions: .docx, .xlsx, .pptx are zipped folders that contain XML files



# Open Document XML Formats

.odt for word processing (text) documents

.ods for spreadsheets

.odp for presentations

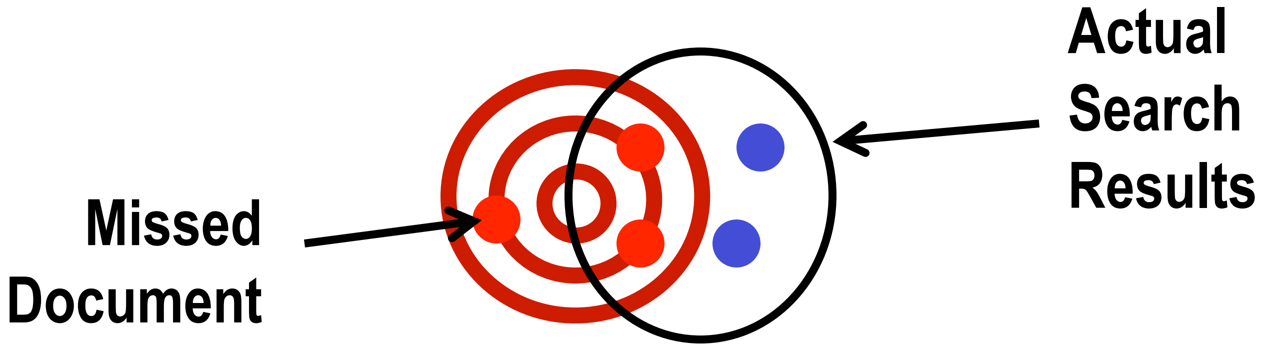
.odb for databases

.odg for graphics

.odf for formula, mathematical equations

# Benefits

- Better search **precision** and **recall**
  - Find the documents you are looking for faster
  - Avoid the inclusion of documents you are **not** interested in



Actual Search Results

Missed Document

● Target documents

● Other documents

# Results from Studies

"Bag-of-words" vs. "full structure"

- 63% improvement in top 5 hits
- 39% improvement in top 10 hits
- 5% improvement in top 20 hits
- .3% improvement in top 30 hits

Source: INEX 2003/2004

# Tibetan Buddhist Resource Center (TBRC)

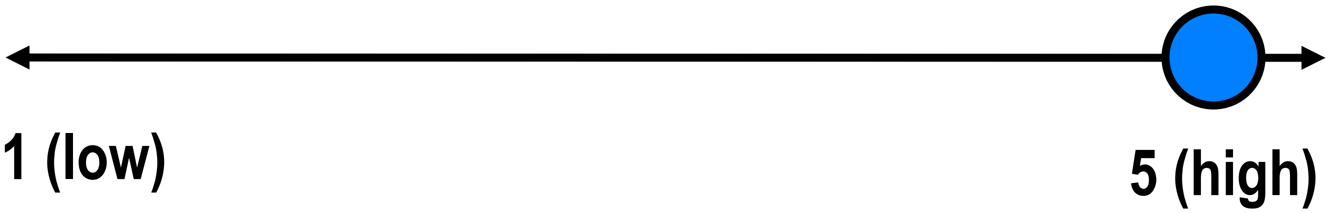
**Contacts**: Chris Tomlinson (Technical Lead) and
Jeffery Wallman (Executive Director)

**Project**: Mercury and Dharmacloud

**Description**:

- Multi-lingual (English, Tibetan and Chinese) web presence for TBRC: http://tbrc.org.
- Use of eXist and Lucene fulltext indexes
- Tibetan Buddhism documents about Works, Outlines, Persons, Places, Topics etc.
- Search metadata on seven million scanned pages of Tibetan texts
- Searching relies on both eXist range indexes and Lucene to search both Unicode and transliterated Tibetan content in a various fields such as titles, personal names, topics, colophons etc.
- Boost values placed on titles and names.
- Combination of indexes (keyword and N-gram) is critical

**Impact**: 5 of 5



# Woodruff Library, Emory University

**Contact:** Dr. Rebecca Sutton Koeser

**Project**: Emory University Finding Aids

**Description**:

- Tools to help researchers find manuscripts within the library
- Performs searches of Encoded Archival Description (EAD) XML documents or "Finding Aids" for archival and manuscript collections
- The most relevant results are now showing up in the first few hits rather than in the first few pages
- For documents like EAD, the structure and organization of the document is very significant and affects how the systems present and search the content
- Having the ability to combine the power of Lucene searching in with XQuery and eXist's native XML handling makes for a potent combination

**Impact**: 4 out of 5

# Challenges

- Difficult to convert some documents in XML format (e.g. PDF)
- What items should be returned in a search "hit"
- Search rules are not easy to set up if you do not have uniform structure
- Many search systems are not optimized for high-volume transactions (locking)

# Getting Data into XML

- Convert RDBMS data into XML using "data web services"
- Use "Apache POI filters" to convert documents to XML
- Begin with XML standards
  - Use existing XML Schemas
  - XForms
  - Native-XML databases

# What to Return in a Hit

- Best Practice: return the lowest level node in the document that contains a hit
- Example:

- **Series**
  - **Book**
    - **Chapter**
      - **Section**
        - **Subsection**

- **Conference**
  - **Presentation**
    - **Slide Group**
      - **Slide**
        - **Bullet Item**

Allow searchers to specify level with search options

# Steps in Testing Structured Search

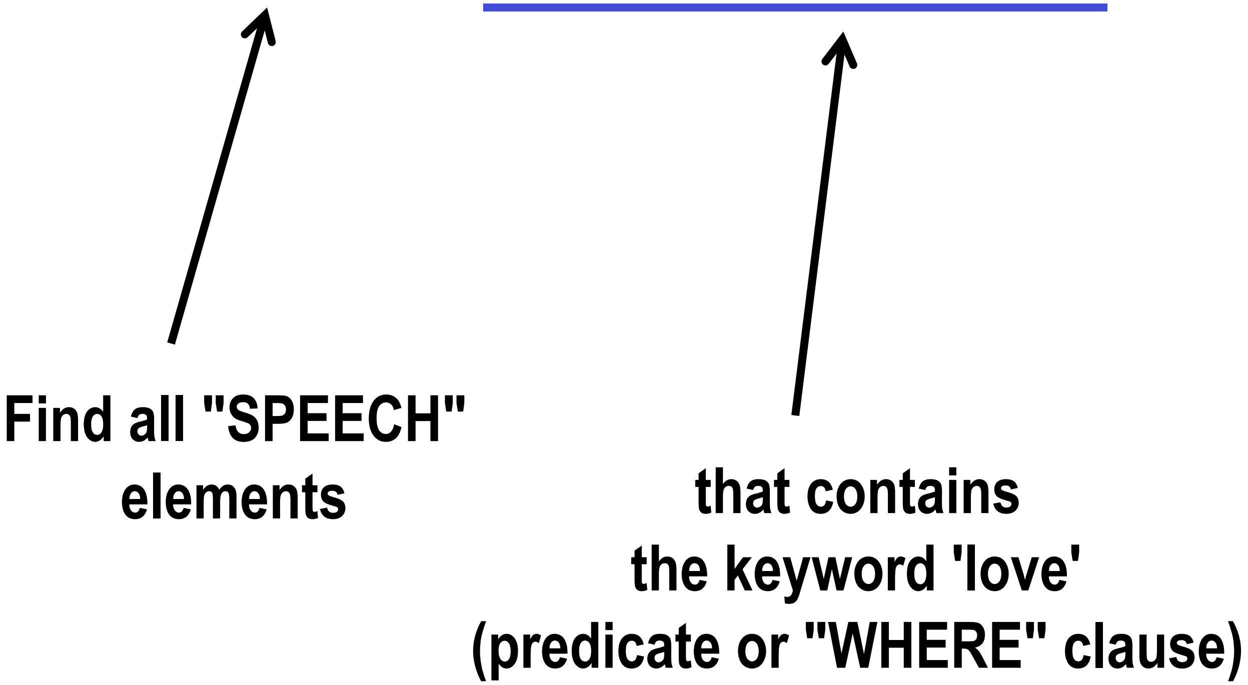
- Convert documents to a format that retains hierarchical structure (xml)
- Treat each "branch" as its own sub-document
- Give each branch a globally distinct type and a "node-id" or document ID
- Index keywords using re-index function
- Set up "weights" based on branch types
- Store the node ids in the reverse index

# Steps in Structured Search Project

1. ROI Analysis – costs of not finding information
2. Pilot Project Selection
3. Structure Extraction
4. Structure Storage
5. Document Consistency
6. Assignment of "boost" values
7. Tuning of boost values for precision and recall
8. ROI Analysis
9. Repeat

# Sample Queries

- //SPEECH[ft:query(., 'love')]



Find all "SPEECH" elements

that contains
the keyword 'love'
(predicate or "WHERE" clause)

# Near Operator

```
let $query :=
  <query>
    <near slop="20">
      <term>snake</term>
      <near>tongue dog</near>
    </near>
  </query>

  return //SPEECH[ft:query(., $query)]
```

# Skillsets Needed for Pilot Project

- Users must "know their data"
  - Subject-matter-experts
- Should be familiar with XML data types
- Need about a week of training with XQuery (depending on background)

# Predictions

- Structured search…
  - when combined with open source tools for extracting "entities" (e.g. Apache UIMA)
  - and machine learning (e.g. Weka)
  - and carefully managed taxonomies

  …will have a large impact on corporate search strategies in the next 10 years

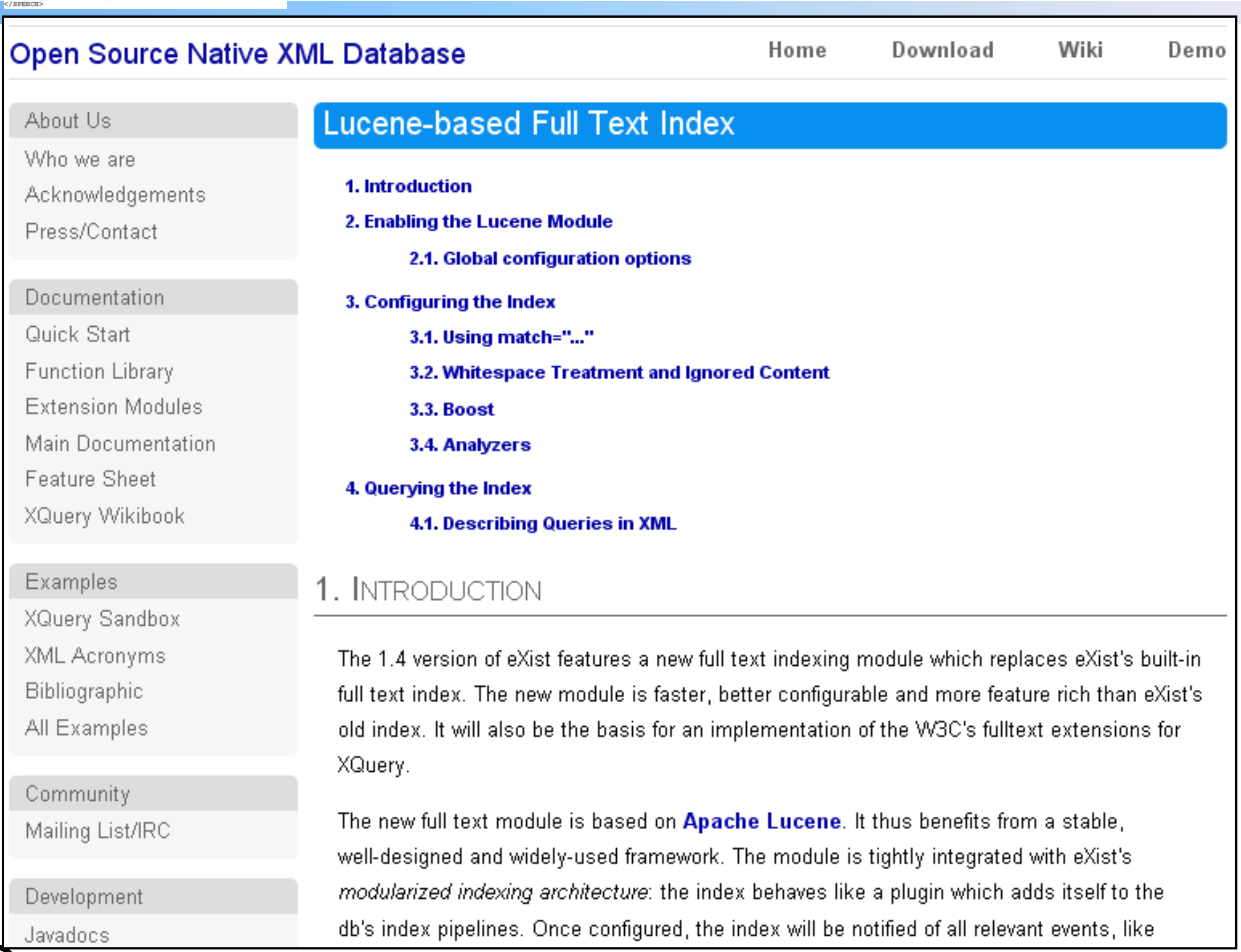
# Steps to Run Examples

- Download XML database (eXist)
- Load examples
- Use "Sandbox"
- Drag new XML documents into eXist WebDAV folder
- Configure index file and reindex
- Run XQuery

# Sample Configuration File

```
<collection xmlns="http://exist-db.org/collection-config/1.0">
        <index xmlns:tei="http://www.tei-c.org/ns/1.0">
            <lucene>
                <!-- Use the standard analyzer will ignore stopwords like
    'the', 'and' -->
                <analyzer
    class="org.apache.lucene.analysis.standard.StandardAnalyzer"/>
                <text match="//tei:TEI/*"/>
                <text match="//tei:title" boost="3.0"/>
            </lucene>
        </index>
    </collection>
```

Example of boost on title



# Lucene-based Full Text Index

## 1. Introduction

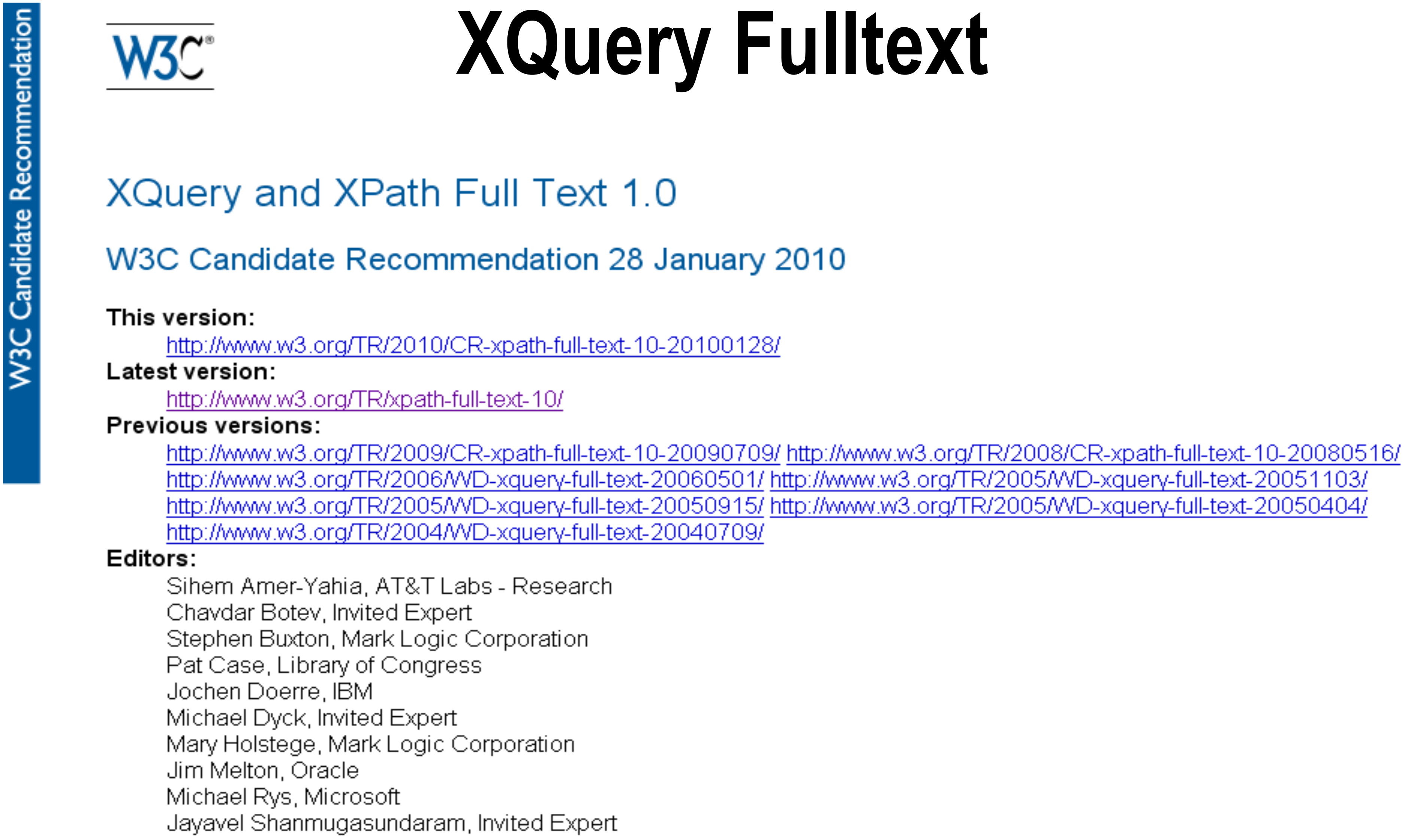
The 1.4 version of eXist features a new full text indexing module which replaces eXist's built-in full text index. The new module is faster, better configurable and more feature rich than eXist's old index. It will also be the basis for an implementation of the W3C's fulltext extensions for XQuery.

The new full text module is based on **Apache Lucene**. It thus benefits from a stable, well-designed and widely-used framework. The module is tightly integrated with eXist's *modularized indexing architecture*: the index behaves like a plugin which adds itself to the db's index pipelines. Once configured, the index will be notified of all relevant events, like

M

D

# XQuery Fulltext



## XQuery and XPath Full Text 1.0

### W3C Candidate Recommendation 28 January 2010

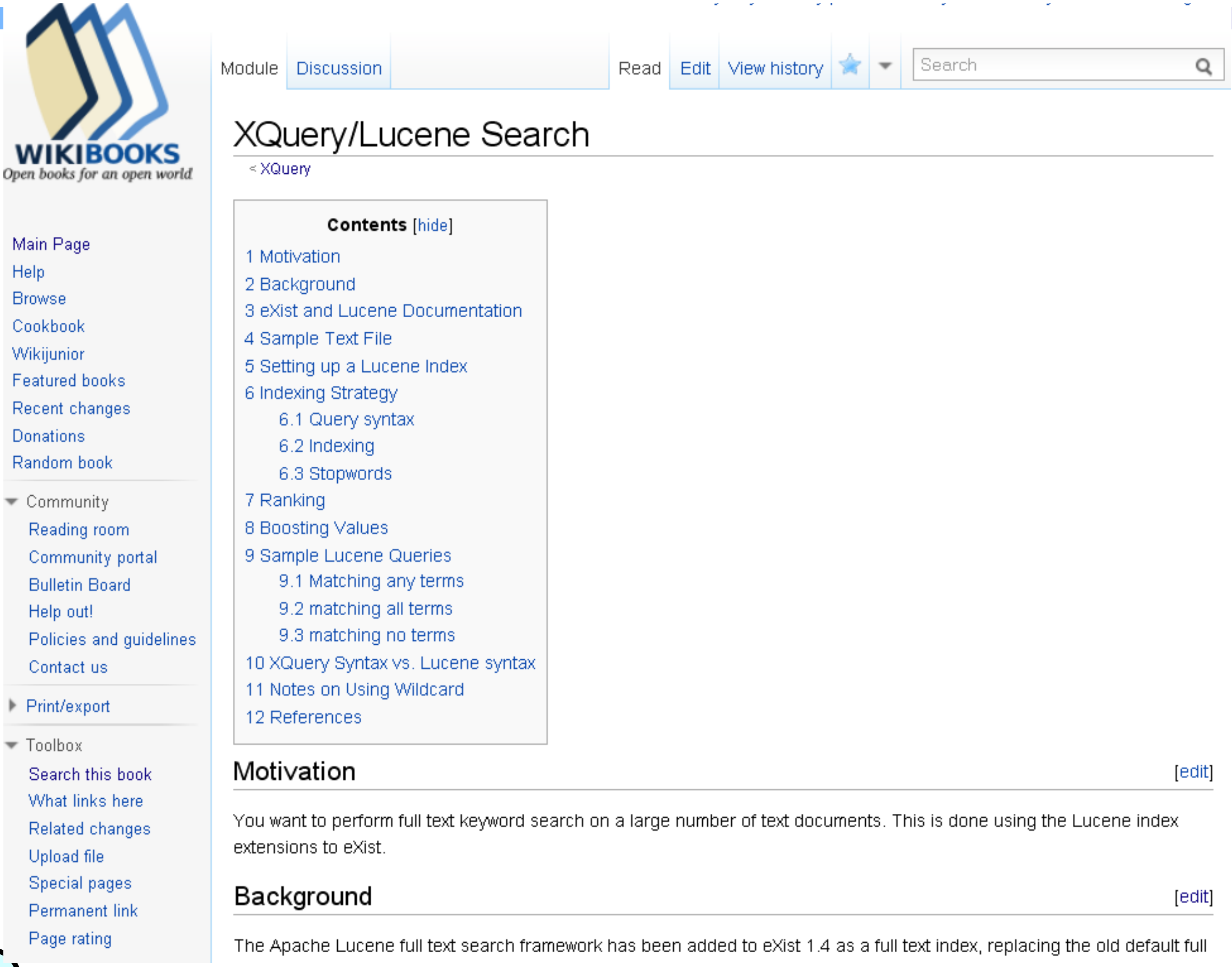
**This version:**
http://www.w3.org/TR/2010/CR-xpath-full-text-10-20100128/

**Latest version:**
http://www.w3.org/TR/xpath-full-text-10/

**Previous versions:**
http://www.w3.org/TR/2009/CR-xpath-full-text-10-20090709/ http://www.w3.org/TR/2008/CR-xpath-full-text-10-20080516/
http://www.w3.org/TR/2006/WD-xquery-full-text-20060501/ http://www.w3.org/TR/2005/WD-xquery-full-text-20051103/
http://www.w3.org/TR/2005/WD-xquery-full-text-20050915/ http://www.w3.org/TR/2005/WD-xquery-full-text-20050404/
http://www.w3.org/TR/2004/WD-xquery-full-text-20040709/

**Editors:**
Sihem Amer-Yahia, AT&T Labs - Research
Chavdar Botev, Invited Expert
Stephen Buxton, Mark Logic Corporation
Pat Case, Library of Congress
Jochen Doerre, IBM
Michael Dyck, Invited Expert
Mary Holstege, Mark Logic Corporation
Jim Melton, Oracle
Michael Rys, Microsoft
Jayavel Shanmugasundaram, Invited Expert

# XQuery/Lucene Search Wikibook



## XQuery/Lucene Search

< XQuery

**Contents** [hide]

### Motivation [edit]

You want to perform full text keyword search on a large number of text documents. This is done using the Lucene index extensions to eXist.

### Background [edit]

The Apache Lucene full text search framework has been added to eXist 1.4 as a full text index, replacing the old default full

# Acknowledgements

- Wolfgang Meier
  - Wrote Lucene implementation used in eXist 1.4
- Dr. Joe Wicentowski
  - US Department of State Office of the Historian
  - eXist index browser (eXist 1.5 release)
- Dr. Martin Mueller
  - Northwestern University
  - Supplied Shakespeare in TEI format
- Ron Van den Branden
  - Royal Academy of Dutch Language and Literature
  - Wrote Wikibook "how to" article for Lucene search
- eXist core development team

# References

- W3C XML standards
- Intro to Information Retrieval Textbook
- Lucene web site
- eXist web site
- XQuery Wikibook
- eXist Lucene implementation
- W3C XQuery
- W3C XQuery/XPath Fulltext extensions

Send e-mail to dan@danmccreary.com for extended list of "getting started" resources.

# Questions?

Dan McCreary
President
Dan McCreary & Associates
dan@danmccreary.com
(952) 931-9198