# COMPSCI 732
# Software Tools and Techniques

Databases and Semistructured Data

## Who is this lecturer?

Gill Dobbie
Rm 303 475
Ph 09 373 7599 ext. 83949
Email gill@cs.auckland.ac.nz

- BTech, MTech at Massey Uni.
- PhD at the University of Melbourne (UoM).
"The foundations of deductive object oriented databases".
- Worked for a couple of years in industry.
- Lectured at Massey University, UoM, VUW.
- Collaborate with researchers at VUW and NUS.
- Current research in databases: building a database specialized to XML or semistructured data, algorithm correctness, access control, data modeling, data mining.

## Outline

- Why is data important?
- Why do we need novel database systems?
- What are Native XML databases (NXDs)?
- What are the advantages of storing semistructured data in NXDs rather than Relational Databases?
- How does querying work in NXDs?

## Why is data important?

- Organizational data is generally not valued as highly as other organizational assets such as cash or monetary resources, real estate, inventory and reputation.
- The data, then, is not well understood, often less well managed, and routinely underutilized.
- The wealth of information to be gleaned from stored data represents untapped resources that can be made available for corporate management to grow its business.

## Relational Databases

- Simple
- Deal with structured data
- Model constraints in schema
- Indexes to improve query performance
- Standard query language
- Have mathematical foundation that is used for modeling algorithms and reasoning about properties and correctness

## What is semistructured data?

- Data with no common structure
  E.g. an XML document that you create to record publications that you have written where the details that you store for conferences is different from the details of journals.
- Data with no predefined structure
  E.g. when you are building a website you do not want to have to define what the constraints on the data are because you don't want to be locked into a structure
- Compare this with data in a relational database

## Representation of semistructured data

- XML is the ideal way to represent semistructured data
- XML stands for eXtensible Markup Language
- XML is a markup language like HTML
- XML is designed to describe data and for the exchange of data
- XML is free and extensible, that is XML tags are not predefined, so you can invent your own tags
- XML is self-describing

**XML Joke [w3schools]**
Question: When should I use XML?
Answer: When you need a buzzword in your resume.

## Positive features of XML

- Do not have to think too hard about the structure of the data before building a document
- Do not have to define a schema
- The documents are very flexible i.e. can add new things and change things in an ad hoc manner

## Negative features of XML

- Do not have to think too hard about the structure of the data before building a document
- Do not have to define a schema, so you don't think about the constraints between bits of data
- The documents are very flexible i.e. can add new things and change things in an ad hoc manner and you may break constraints

## XML Documents

XML documents fall into two broad categories: *data-centric* and *document-centric*.

```
<?xml version='1.0'?>
 <!-- This file represents a fragment of a book store inventory database -->
<bookstore>
  <book genre="autobiography">
    <title>The Autobiography of Benjamin Franklin</title>
    <author>
      <first-name>Benjamin</first-name>
      <last-name>Franklin</last-name>
    </author>
    <price>(US)$8.99</price>
  </book>
  <book genre="novel">
    <title>The Confidence Man</title>
    <author> Herman Melville</author>
    <price>(NZ)$11.99</price>
  </book>
</bookstore>
```

**Data Centric**

**Document Centric**

```
<Product>
   <Name>Turkey Wrench</Name>
   <Developer>Full Fabrication Labs, Inc.</Developer>
   <Summary>Like a monkey wrench, but not as big.</Summary>
   <Description>
   <Para>The turkey wrench, which comes in <i>both right- and
   left-handed versions (skyhook optional)</i>, is made of the <b>finest stainless steel</b>.
   The Readi-grip rubberized handle quickly adapts to your hands, even in the greasiest situations.
   Adjustment is possible through a variety of custom dials.
   </Para>
   <Para>You can:</Para>
   <List>
   <Item><Link URL="Order.html">Order your own turkey wrench </Link> </Item>
   <Item><Link URL="Wrenches.htm">Read more about wrenches </Link> </Item>
   <Item><Link URL="Catalog.zip">Download the catalog</Link></Item>
   </List>
    <Para>The turkey wrench costs <b>just $19.99</b> </Para>
   </Description>
 </Product>
```

## Tools for the storage and management of XML documents

- I come to this area from a database perspective, as do many other researchers in this area.
- There are a lot of issues:
  - Removing redundancy
  - Modeling constraints on the data
  - Mapping XML to RDBs
  - Storing XML in native XML databases
  - Query performance in native XML databases
  - Indexes in native XML databases
  - Etc.

## Do we need XML databases?

Course web pages from the UoA

Software Engineering 2

SOFTENG 330 S1 C

2003

Contents:

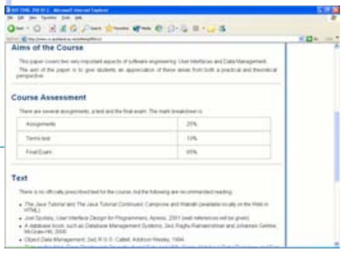
News

[illegible]

Aims of the Course

[illegible]

Characteristics:
-Currently stored in HTML in files
-Some content static, some changes often
-Not all pages have the same structure
-Updated by >1 person


Aims of the Course

[illegible]

Course Assessment

[illegible]

| | |
|---|---|
| Assignments | 25% |
| Terms test | 10% |
| Final Exam | 65% |

Text

[illegible]

Solution:
-Store information in database
-How should information be organized?
-Who should be able to access what information?

## What are XML databases?

- **XML-enabled databases** are databases with extensions for mapping data between XML documents and themselves, e.g., Access 2002, Informix, Oracle 9i.
- **Native XML databases** are databases that store XML in "native" form, generally as some variant of the DOM mapped to an underlying data store, e.g., eXist, Tamino, Xindice.

## What is a native XML database (NXD)?

- It's a database that stores and retrieves XML documents efficiently
- The view that users have should be that it stores and retrieves XML documents
- The query language should be based on XML, and the structure of XML
- The indexes should ensure the fast execution of queries against XML documents

## What is a native XML database (NXD)?

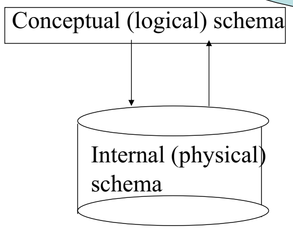
Definition from XML:DB Initiative (http://www.xmldb.org/)
a) Defines a (logical) model for an XML document – as opposed to the data in the document - and stores and retrieves documents according to that model. At a minimum, the model must include elements, attributes, PCDATA, and document order. Examples of such models are the XPath data model, the XML Infoset, and the models implied by the DOM and the events in SAX.
b) Has an XML document as its fundamental unit of (logical) storage, just as a relational database has a row in a table as its fundamental unit of (logical) storage.
c) Is not required to have any **particular** underlying physical storage model. For example, it can be built on a relational, hierarchical, or object-oriented database, or use a proprietary storage format such as indexed, compressed files.

## Background

**What is a logical model?**

It is the structure of the data as we understand it.

For relational databases the logical model consists of tables/rows/columns etc.



The physical model represents how the data is really stored.

## What are the implications of the definition?

1. A native XML database is specialized for storing XML documents and stores all components of the XML model intact.

2. XML documents go in and XML documents come out.

3. The underlying data storage format or the physical model is unimportant for the categorization of databases.

## Advantages of storing data in NXD

Semistructured data stored in a relational database will result in a large number of nulls or a large number of tables. ☑

☑ Retrieval of documents or parts of documents might be fast.

Retrieving a view over the data might be slower ☒

## More nulls or more tables

```
<list>
  <person>
    <name>bob</name>
    <age>15</age>
    <mother>mary</mother>
    <father>john</father>
  </person>
  <person>
    <name>john</name>
    <parent>jacob</parent>
  </person>
  <person>
    <name>mary</name>
  </person>
</list>
```

| name | age | mother | father | parent |
|---|---|---|---|---|
| bob | 15 | mary | john | null |
| john | null | null | null | jacob |
| mary | null | null | null | null |

| id | name |
|---|---|
| 1 | bob |
| 2 | john |
| 3 | mary |

| id | age |
|---|---|
| 1 | 15 |

| id | mother |
|---|---|
| 1 | mary |

| id | father |
|---|---|
| 1 | john |

| id | parent |
|---|---|
| 2 | jacob |

## Faster retrieval of documents

```
<list>
 <person>
   <name>bob</name>
   <age>15</age>
   <mother>mary</mother>
   <father>john</father>
 </person>
 <person>
   <name>john</name>
   <parent>jacob</parent>
 </person>
 <person>
   <name>mary</name>
 </person>
</list>
```

| id | name |
|---|---|
| 1 | bob |
| 2 | john |
| 3 | mary |

| id | age |
|---|---|
| 1 | 15 |

| id | mother |
|---|---|
| 1 | mary |

| id | parent |
|---|---|
| 2 | jacob |

| id | father |
|---|---|
| 1 | john |

## Slower retrieval of views

```
<list>
  <person>
    <name>bob</name>
    <age>15</age>
    <mother>mary</mother>
    <father>john</father>
  </person>
  <person>
    <name>john</name>
    <parent>jacob</parent>
  </person>
  <person>
    <name>mary</name>
  </person>
</list>
```

| id | name |
|---|---|
| 1 | bob |
| 2 | john |
| 3 | mary |

| id | age |
|---|---|
| 1 | 15 |

| id | mother |
|---|---|
| 1 | mary |

| id | parent |
|---|---|
| 2 | jacob |

| id | father |
|---|---|
| 1 | john |

Find the name of everyone who is 15

## Architectures for NXDs

Architectures for NXDs fall into two categories:

**Text based NXD**

**Model based NXD**

## Text based NXD

Stores XML as text e.g. in a file, in a relational database, in some other form.

Usually have indexes, allowing direct access within the XML document, improving access to documents or pieces of documents.

Problem when inverting the hierarchy or portions of it.

## Model based NXD

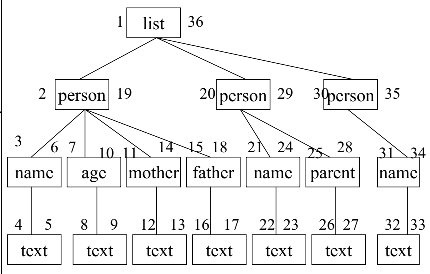
Rather than storing the XML document as text, build an internal object model from the document and store this model. How the model is stored depends on the underlying database. Some databases use a propriety storage format optimized for their model.

Tables can include doc, node, element_name, element_value, etc.

Model taken from http://www.informatics.bangor.ac.uk/~rich/research/papers/uwb_rge_IDEAL2000.pdf

## Model based NXD example

```
<list>
  <person>
    <name>bob</name>
    <age>15</age>
    <mother>mary</mother>
    <father>john</father>
  </person>
  <person>
    <name>john</name>
    <parent>jacob</parent>
  </person>
  <person>
    <name>mary</name>
  </person>
</list>
```



XPath queries:
/list/person[name="john"]/parent
/list/person[1]
/list/person[1]/*

## Model based NXD example

doc

| doc_id | root_node_id |
|---|---|
| 1 | 1 |

node

| node_id | owner_doc_id | depth | parent_node | prev_sibling | next_sibling | first_child |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | null | null | null | 2 |
| 2 | 1 | 2 | 1 | null | 20 | 3 |
| 3 | 1 | 3 | 2 | null | 7 | 4 |
| 4 | 1 | 4 | 3 | null | 8 | null |
| … | … | … | … | … | … | … |

element_name

| node_id | name |
|---|---|
| 1 | list |
| 2 | person |
| 3 | name |
| … | … |

element_value

| node_id | value |
|---|---|
| 4 | bob |
| 8 | 15 |
| … | … |

## Query languages

The most popular query languages to date are based on XPath.
Since the W3C has defined XQuery, XPath has become more popular.
Previously, many propriety query languages were supported.

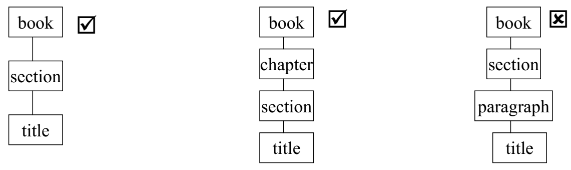
The query language must support different kinds of queries, including paths through the hierarchy, missing levels in the hierarchy, ordering, many documents, collections.

## XPath

XPath uses path expressions to navigate through the logical, hierarchical structure of an XML document. An XPath expression locates nodes within a tree.

book//section/title

Finds all "title" elements that are children of "section" elements which have an ancestor named "book".

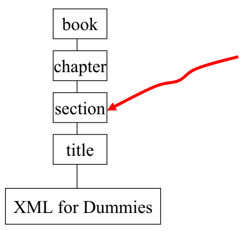


## XPath predicate expressions

book//section[contains(title, 'XML')]

Find all sections whose title contains the string "XML".

The result of a path expression is a sequence of distinct nodes in document order.



## XPath processing

book//figure/caption

Follow every path beginning at book to check for potential figure descendents and subsequently caption children, unless there is a way to determine the location of "figure" descendents in advance.

Index structures are needed to efficiently perform queries on large document collections.

The indexes must support both structural and value based selections. B+ trees (or similar) work well for value based selections. What about ancestor/descendant and parent/child relationships?

## Summary

- Data is an asset that is undervalued.
- Relational databases are a well developed technology for structured data.
- Relational databases are not suitable for unstructured or semistructured data.
- Native XML databases are suitable for semistructured data but they are less well developed.
- NXDs vary in the way they model and store data.
- There are two main architectures for NXDs: text based NXDs and model based NXDs.
- Text based NXDs are preferable if the user does not need to manipulate the structure of the data much.

## Source and further reading

- http://www.w3schools.com/
- http://www.w3.org/
- Ramakrishnan, Raghu and Gehrke, Johnannes. Database Management Systems.McGraw Hill.
- Elmasri, Ramez and Navathe, Shamkant. Fundamentals of Database Systems. Addison Wesley.
- Bourett, Ronald. XML and Databases.
  Available at : http://www.rpbourret.com/xml/XMLAndDatabases.htm
- Bourett, Ronald. XML Database Products.
  Available at : http://www.rpbourret.com/xml/XMLAndDatabases.htm
- Obasanjo, Dare. An exploration of XML in DBMS.
  Available at : http://www.25hoursaday.com/storingandqueryingxml.html
- Widom, Jennifer. Data Management for XML: Research Directions, IEEE Data Engineering 22(3)
- Abiteboul, Serge and Buneman, Peter and Suciu, Dan. Data on the Web:From Relations to Semistructured Data and XML, Morgan Kaufmann, 2000
- Chaudhri, Akmal and Rashid, Awais and Zicari, Roberto. XML Data Management: Native XML and XML-Enabled Databases, Addison Wesley Professional, 2003.

## Assorted Current Projects

- Proving properties of semistructured data and related algorithms
- Semantic query optimization in native XML databases
- Building a realtime ETL layer for data warehouses
- Building a tool to construct mashups

## Questions to ponder

- Are we replacing the steel wheel with a wooden wheel?
- Is the database community targeting the right problems or is our view too data centric?
- Is XML rich enough?
- Is this research limited to XML databases or does it apply to any systems that deal with XML documents?
- Should we take XML databases seriously or are they just a passing fad?
- Are XML databases just databases, as we know them?