SPARK
THE UNIVERSITY OF AUCKLAND ENTREPRENEURSHIP CHALLENGE

- Grand Launch for 2005
  - Thursday 10th March, 6pm, @ Fale Pasifika
  - Featured Speakers:
    - Vice Chancellor - Professor Stuart McCutcheon
    - Entrepreneur & Founder of 42 Below Vodka - Geoff Ross
  - Hear about the additional new Spark Aspire Social Entrepreneurship Challenge
  - Snacks and Refreshments provided
  - RSVP online by March 7th to win prizes valued at $3000.
- Vision to Business Seminar Series
  - Learn how to turn your idea into a successful business
  - FREE, Starts 15th March
- Visit www.spark.auckland.ac.nz for full details

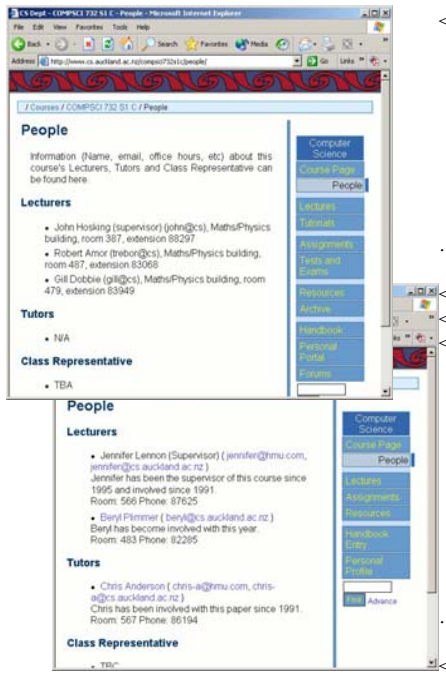
COMPSCI 732 S1 C

Software Tools and Techniques

Native XML databases (NXDs)

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 that the fast execution of queries against XML documents




```
<course code = "compsci732s1c">
  <people>
    <lecturer>
          <name>John Hosking</name>
          <email>john@cs</email>
          <building>Maths/Physics</building>
          <room>387</room>
          <phone>88297</phone>
    </lecturer>
  …….
  </people>
</course>
<course code = "compsci708s1c">
<people>
    <lecturer>
          <name>Jennifer Lennon</name>
          <email>jennifer@hmu.com</email>
          <email>jennifer@cs.auckland.ac.nz</email>
          <note>Jennifer has been the supervisor
          of this course since 1995 and involved
          since 1991 </note>
          <room>506</room>
          <phone>87625</phone>
    </lecturer>
  ……..
  </people>
</course>
```

What is a native XML database (NXD)?

Definition from XML:DB Initiative (http://www.xmldb.org/)

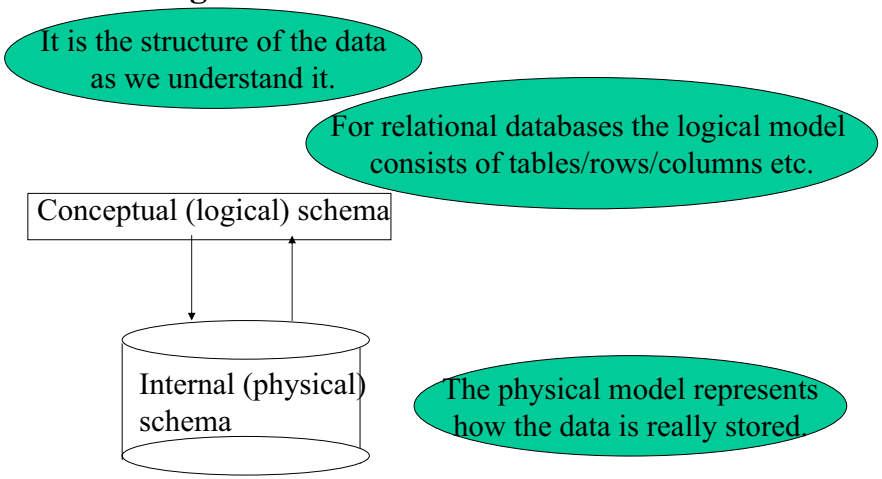
a) Defines a (logical) model for an XML document – as opposed to the data in the document - and stores and retrieves documents according to that model. At a minimum, the model must include elements, attributes, PCDATA, and document order. Examples of such models are the XPath data model, the XML Infoset, 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?



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.

What are the suggested logical models?

Infoset – XML Information Set is a definition proposed by the W3C that provides a consistent set of definitions to refer to the definitions in an XML document, with terms such as Information Set and Information Item. See http://www.w3.org/TR/xml-infoset/

DOM – Document Object Model is a platform- and language-neutral interface that allows programs to dynamically access and update the content, structure and style of document. It views a document as a hierarchy of nodes. See http://www.w3.org/DOM/

XPath data model – based on Infoset with extra features:
-representation for collection of documents and complex types
-support for typed atomic values (XML Schema)
-support for ordered heterogeneous sets
See http://www.w3.org/TR/xpath-datamodel/

SAX – simple API for XML. See http://www.saxproject.org/

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

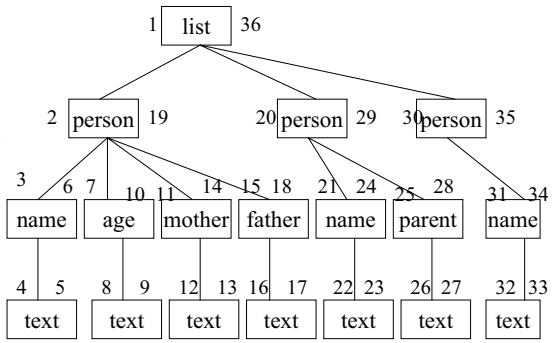
Rather than storing the XML document as text, build an internal object model from the document and store this model. How the model is stored depends on the underlying database. Some databases use a propriety storage format optimized for their model.

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
…	…

Summary

- NXDs vary in the way they model and store data. (This is due in part to how new the database is.)
- 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.