Modern Databases

Database Systems

Michael Pound

This Lecture

- Row-Orientated Vs Column-Orientated Databases
- Other Types of Database
  - OODBMSs and ORDBMSs
  - Distributed DBMSs
    - Example: BigTable
  - Semi-structured Data
  - Multimedia Databases
- Further reading
  - The Manga Guide To Databases, Chapter 6
  - Database Systems, Chapters 24, 27 and 29
  - BigTable:http://labs.google.com/papers/bigtable.html

Row-orientated DBMSs

- Until now, we have focused on RDBMS, which are row-orientated
  - Data is arranged in rows (tuples)
  - This is reflected on the disks in pages

1001001	Smith	Andrew	5 Arnold Close	2	
1001002	Brooks	James	7 Holly Avenue	2	Page 1
1001003	Anderson	Max	15 Main Street	3	
1001004	Evans	Sarah	Flat 1a, High Street	2	
1001005	Harrison	Sam	Newark Hall	1	Page 2
1001006	Jones	Ben	Southwell Hall	1	

Column-orientated DBMSs

- Large scale databases often store data in a column-orientated way
  - Data is arranged by column
  - This is also reflected on the disks in pages

1001001	1001002	1001003	1001004	1001005	100100	Page 1
Smith	Brooks	Anderson	Evans	Harrison	Jones	Page 17
Andrew	James	Max	Sarah	Sam	Ben	Page 34
5 Arnold Close	7 Holly Avenue	15 Main Street	Flat 1a, High			Page 70

2	2	3	2	1	1	1	2	3	2	2	3	3	2	2	1	1	Page 101

Rows vs Columns

- Database speed is usually limited by disk reads. You want to use the optimum storage strategy for the demands on the database
  - Row-orientated is extremely good when queries often require many columns in the same row
  - Row-orientated provides fast inserts of complete rows
  - Column-orientated is more suited to fast analysis over single columns, e.g. Aggregate functions
  - Column-orientated provides fast inserts when many rows are inserted at once

Weaknesses of RDBMSs

- Conversion to and from standard procedural languages often requires a lot of code
- Support for anything beyond standard types is poor. For example, images and video
- Complex objects and relationships aren't well handled. Often involving a great deal of normalisation, and eventually a lot of JOINs to recombine data

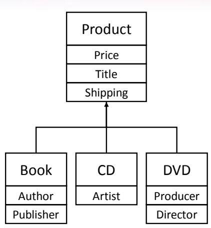
Object Oriented Databases

- An object oriented database (OODB) is a collection of persistent objects
  - Objects - instances of a defined class
  - Persistent - objects exist independently of any program
- An object oriented DBMS
  - Manages a collection of objects
  - Allows objects to be made persistent
  - Permits queries to be made of the objects
  - Does all the normal DBMS things as well

OODB Example

- Consider a store with a variety of products
  - Books
  - CDs
  - DVDs
- This lead to missing data among the various types
- OODB solution
  - We make an abstract Product class
  - Book, CD, and DVD are each a concrete subclass of Product
  - The database is a persistent collection of Products

OODB Example



- Product is abstract
  - You cannot make a Product directly
  - You can, however, make a Book, CD, or DVD, and these are Products

Object Oriented Databases

- Advantages
  - Good integration with Java, C++, etc
  - Can store complex information
  - Fast to recover whole objects
  - Has the advantages of the (familiar) object paradigm
- Disadvantages
  - There is not underlying data model that everyone agrees on (unlike say, the relational model)
  - Can be more complex and less efficient
  - OODB queries tend to be procedural, unlike SQL

Object-Relational Databases

- Extend a RDBMS with object concepts
  - Data values can be objects of arbitrary complexity
  - These objects have inheritance etc.
  - You can query the objects as well as the tables
- An object relational database
  - Retains most of the structure of the relational model
  - Needs extensions to query languages (SQL or relational algebra)
  - Most DBMSs already implement this as part of SQL3 onwards

ORDBMS Example

- In Oracle, PostgreSQL etc, you can define your own types (exact syntax varies):

```
CREATE TYPE Address AS OBJECT (
 aNumber INT,
 aRoad VARCHAR(64),
 aPostcode VARCHAR (8));

CREATE TYPE Student AS OBJECT (
 sID INT,
 sName VARCHAR(64),
 sAddress Address,
 sYear
 MAP MEMBER FUNCTION
 age RETURN INT);
```

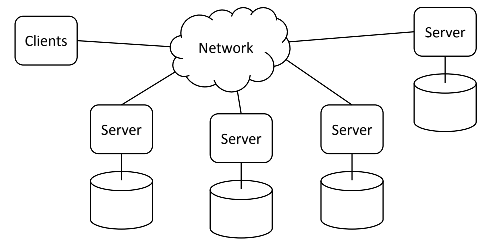
```
CREATE TYPE HomeStudent
  UNDER Student (
 sHomeLEA INT);

CREATE TYPE ExchangeStudent
  UNDER Student (
 sVisaInfo VisaInfo);
```

Distributed Databases

- A distributed DB system consists of several sites
  - Sites are connected by a network
  - Each site can hold data and process it
  - It shouldn't matter where the data is - the system is a single entity
- Distributed database management system (DDBMS)
  - A DBMS (or set of them) to control the databases
  - Communication software to handle interaction between sites

Distributed Databases

- Distributed databases often make use of a client/server architecture

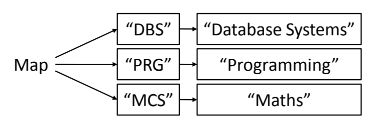


Example: Google BigTable

- Google BigTable is a distributed database (more or less) that contains sparse, distributed, multi-dimensional and sorted maps of data
  - Sparse: Much of the data has no value
  - Distributed: Tables are stored over many locations, and duplicated
  - Multi-dimensional: Cells are accessed by more than just a row and column
- The aim of BigTable is massive scalability. It's used on Google Earth, Google Maps, Youtube etc.

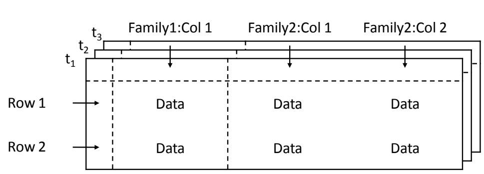
Maps

- A map is a data structure that holds <key,value> pairs. Much like the associative arrays we saw in PHP
- Values are accessed by using the required key



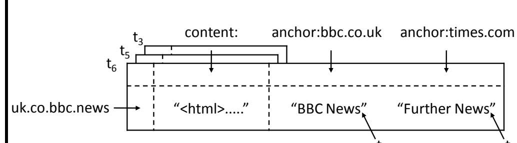
BigTable Maps

- In BigTable, maps are multi-dimensional, with string keys (row, column, time) referencing a single string
- Exact use of rows and columns, much like any database, depends on the data being stored
- Time is used to store multiple versions of data
- Columns are further split into families



BigTable Example

- Google use an example of a table holding information on webpages
  - Rows are reversed URLs for the pages
  - The 'content' column family holds the HTML
  - 'anchor' columns hold data on websites that link to this one
  - 'content' has a time dimension to hold older version of each page



BigTable Data Organisation

- The way BigTable is stored on disk, and the algorithms for querying and sorting the data are fairly complex. Some notable points are:
  - Data is sorted by rows, and groups of rows are split into tablets. Tablets are the unit of distribution and load balancing
  - A scheduler will keep note of current queries and rearrange if certain tablets or servers are under too much load
  - Timestamps can be used to only keep data for a certain amount of time etc.
  - Locking is performed on column families

Semistructured data

- Semistructured Data : A new data model designed to cope with problems of information integration. That is, where data from multiple sources could theoretically be used together, but varies wildly in structure
- XML : A standard language for describing semistructured data schemas and representing data.

XML

- XML = Extensible Markup Language.
- While HTML uses tags for formatting (e.g., “italic”), XML uses tags for semantics (e.g., “this is an address”).
- Key idea: create tag sets for a domain (e.g., bars), and translate all data into properly tagged XML documents.
- Well formed XML - XML which is syntactically correct; tags and their nesting totally arbitrary.
- Valid XML - XML which has DTD (document type definition); imposes some structure on the tags, but much more flexible than relational database schema.

XML and Semistructured Data

- Well-Formed XML with nested tags is exactly the same idea as trees of semistructured data.
- XML also enables non-tree structures (with references to IDs of nodes), as does the semistructured data model.

XPATH and XQUERY

- XPATH is a language for describing paths in XML documents.
  - Really think of the semistructured data graph and its paths.
  - Why do we need path description language: can’t get at the data using just Relation.Attribute expressions.
- XQUERY is a full query language for XML documents with power similar to OQL (Object Query Language, query language for object-oriented databases).
  - XQUERY is now seeing use in some modern DBMSs where you can opt to use XML for a storage structure

Multimedia Databases

- Multimedia DBs can store complex information
  - Images
  - Music and audio
  - Video and animation
  - Full texts of books
  - Web pages
- They can be used in a wide range of application areas
  - Entertainment
  - Marketing
  - Medical imaging
  - Digital publishing
  - Geographic Information Systems

Querying Multimedia DBs

- Metadata searches
  - Information about the multimedia data (metadata) is stored
  - This can be kept in a standard relational database and queried normally
  - Limited by the amount of metadata available
- Content searches
  - The multimedia data is searched directly
  - Potential for much more flexible search
  - Depends on the type of data being used
  - Often difficult to determine what the 'correct' results are

Metadata Searches

- Example - indexing films we might store
  - Title
  - Year
  - Genre(s)
  - Actor(s)
  - Director(s)
  - Producer(s)
- We can then search for things like
  - Films starring Kevin Spacey
  - Films directed by Peter Jackson
  - Dramas produced in 2000
- We don't actually search the films

Metadata Searches

- Advantages
  - Metadata can be structured in a traditional DBMS
  - Metadata is generally concise and so efficient to store
  - Metadata enriches the content
- Disadvantages
  - Metadata can't always be found automatically, and so requires data entry
  - It restricts the sorts of queries that can be made

Content Searches

- An alternative to metadata is to search the content directly
  - Multimedia is less structured than metadata
  - It is a richer source of information but harder to process
- Example of content based retrieval
  - Find images similar to a given sample
  - Hum a tune and find out what it is
  - Search for features, such as cuts or transitions in films

Revision Lectures

- There are two revision lectures that will take place in the first week back after the holiday
- I will include information on what sort of things you can expect in the exam, and the general structure of the exam
- There is plenty of time to go over more challenging topics from the module, email me if you are unsure about a subject and I will add it to the revision lectures