Outline

• An outside look: DB Application
• An inside look: Anatomy of DBMS
• Project ideas: DB Application
• Project ideas: DBMS Internals
Database Management System (DBMS): a software package designed to store and manage a large amount of data.
An Outside Look: DB Application

High-level, declarative interface

- Persistent storage
- Performance
- Concurrency
- Automatic recovery
- Security…

Data-Intensive Application
Case Study: The Internet Shop*

- **DBDudes Inc.:** a well-known database consulting firm
- **Barns and Nobble (B&N):** a large bookstore specializing in books on horse racing
- **B&N** decides to go online, asks DBDudes to help with the database design and implementation
- **Step 0:** DBDudes makes B&N agree to
  - pay steep fees and
  - schedule a lunch meeting for requirements analysis

* The example and all related material was taken from “Database Management Systems” Edition 3.

Yanlei Diao, University of Massachusetts Amherst 2/13/2006
Step 1: Requirements Analysis

• “I’d like my customers to be able to browse my catalog of books and place orders online.”
  
  – **Books:**
    • For each book, B&N’s catalog contains its ISBN number, title, author, price, year of publication, …
  
  – **Customers:**
    • Most customers are regulars with names and addresses registered with B&N.
    • New customers must first call and establish an account.
  
  – **On the new website:**
    • Customers identify themselves before browsing and ordering.
    • Each order contains the ISBN of a book and a quantity.

  – **Shipping:**
    • For each order, B&N ships all copies of a book together once they become available.
Step 2: Conceptual Design

• A high level description of the data in terms of the Entity-Relationship (ER) model.

• Design review:
  – What if a customer places two orders of the same book in one day?
  – Modification: add “ordernum” to Orders.
Step 3: Logical Design

- Mapping the ER diagram to the relational model

CREATE TABLE Books
(isbn CHAR(10),
title CHAR(80),
author CHAR(80),
qty_in_stock INTEGER,
price REAL,
year INTEGER,
PRIMARY KEY(isbn))

CREATE TABLE Customers
(cid INTEGER,
cname CHAR(80),
address CHAR(200),
PRIMARY KEY(cid))

CREATE TABLE Orders
(ordernum INTEGER,
isbn CHAR(10),
cid INTEGER,
cardnum CHAR(16),
qty INTEGER,
order_date DATE,
ship_date DATE,
FOREIGN KEY (isbn) REFERENCES Books,
FOREIGN KEY (cid) REFERENCES Customers)

FOREIGN KEY (isbn) REFERENCES Customers)
Step 3: Logical Design

- Mapping the ER diagram to the relational model

**CREATE TABLE Books**
(isbn CHAR(10),
title CHAR(80),
author CHAR(80),
qty_in_stock INTEGER,
price REAL,
year INTEGER,
PRIMARY KEY(isbn))

**CREATE TABLE Customers**
(cid INTEGER,
cname CHAR(80),
address CHAR(200),
PRIMARY KEY(cid))

**CREATE TABLE Orders**
isbn CHAR(10),
cid INTEGER,
qty INTEGER,
order_date DATE,
ship_date DATE,
FOREIGN KEY (isbn) REFERENCES Books,
FOREIGN KEY (cid) REFERENCES Customers)

**CREATE VIEW OrderInfo**
(isbn, cid, qty, order_date, ship_date)
AS SELECT O.isbn, O.cid, O.qty,
O.order_date, O.ship_date
FROM Orders O

- Access control: use views to restrict the access of certain employees to customer sensitive information
Step 4: Schema Refinement

Orders

<table>
<thead>
<tr>
<th>ordernum</th>
<th>isbn</th>
<th>cid</th>
<th>cardnum</th>
<th>qty</th>
<th>order_date</th>
<th>ship_date</th>
</tr>
</thead>
<tbody>
<tr>
<td>120</td>
<td>0-07-11</td>
<td>123</td>
<td>40241160</td>
<td>2</td>
<td>Jan 3, 2006</td>
<td>Jan 6, 2006</td>
</tr>
<tr>
<td>120</td>
<td>1-12-23</td>
<td>123</td>
<td>40241160</td>
<td>1</td>
<td>Jan 3, 2006</td>
<td>Jan 11, 2006</td>
</tr>
<tr>
<td>120</td>
<td>0-07-24</td>
<td>123</td>
<td>40241160</td>
<td>3</td>
<td>Jan 3, 2006</td>
<td>Jan 26, 2006</td>
</tr>
</tbody>
</table>

Redundant Storage!

Orderlists

<table>
<thead>
<tr>
<th>ordernum</th>
<th>isbn</th>
<th>qty</th>
<th>ship_date</th>
</tr>
</thead>
<tbody>
<tr>
<td>120</td>
<td>0-07-11</td>
<td>2</td>
<td>Jan 6, 2006</td>
</tr>
<tr>
<td>120</td>
<td>1-12-23</td>
<td>1</td>
<td>Jan 11, 2006</td>
</tr>
<tr>
<td>120</td>
<td>0-07-24</td>
<td>3</td>
<td>Jan 26, 2006</td>
</tr>
</tbody>
</table>
Step 5: Internet Application Development

Presentation tier
- Interface to the user
- Adapt to display devices

Application logic tier
- Business logic (actions, state between steps)
- Access multiple sources

Data management tier
- One/multiple DBMS(s)

Client Program (Web Browser)  HTML, Javascript, Cookies

HTTP

Application Server (Apache Tomcat…)

JDBC

Database System (DB2, MySQL…)

JSP, Servlets, XSLT

XML, stored procedures

B&N Client:
- User input
- Session state

B&N Business logic:
- Home page
- Login page
- Search page
- Cart page
- Confirm page

B&N Data:
- Books
- Customers (User login)
- Orders
- Orderlists

Yanlei Diao, University of Massachusetts Amherst 2/13/2006
An Example Internet Store

Welcome to The Internet BookShop. We have just recently opened for business!!!

We've spend a great deal of time in creating a website really suited to the desires of today's book buyers. We offer fast and convenient service with operators available at all hours of the day to process your requests.

And of course, membership is free of charge! Sign up today!

You may log into your account by clicking here, or create a new account by clicking here. If you want to add items to your shopping cart without logging in, feel free to...
Example SQL Queries

**Search Page**

```
SELECT isbn, title, author, price
FROM Books
WHERE isbn = '%<SearchString>%'
ORDER BY title
```

**Login Page**

```
SELECT cid, username, password
FROM Customers
WHERE username = '<SpecifyUsername>'
```

**Confirm Page**

```
INSERT INTO Orders
(cid, cardnum, order_date)
VALUES (<Cid>,<CreditCardNumber>,<OrderDate>)

SELECT ordernum
FROM Orders
WHERE CID = <Cid>
ORDER BY ordernum DESC

INSERT INTO Orderlists
(ordernum, isbn, qty)
VALUES
(<OrderNumber>,<ISBN>,<Quantity>)
```
Step 6: Physical Design

- Good performance for **typical workloads**
- Auxiliary data structures (**indices**) to speed up searches

### Books

<table>
<thead>
<tr>
<th>isbn</th>
<th>title</th>
<th>author</th>
<th>price</th>
<th>year</th>
<th>qty</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-07-11</td>
<td>Legacies of the Turf</td>
<td>Edward L. Bowen</td>
<td>29.95</td>
<td>2003</td>
<td>10</td>
</tr>
<tr>
<td>1-12-23</td>
<td>Seattle Slew</td>
<td>Dan Mearns</td>
<td>24.95</td>
<td>2000</td>
<td>0</td>
</tr>
<tr>
<td>0-07-24</td>
<td>Spectacular Bid</td>
<td>Timothy Capps</td>
<td>16.95</td>
<td>2001</td>
<td>3</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

**Hash Index on Books.isbn**

Yanlei Diao, University of Massachusetts Amherst 2/13/2006
Step 6: Physical Design

- Good performance for typical workloads
- Auxiliary data structures (indices) to speed up searches

Books

<table>
<thead>
<tr>
<th>isbn</th>
<th>title</th>
<th>author</th>
<th>price</th>
<th>year</th>
<th>qty</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-07-11</td>
<td>Legacies of the Turf</td>
<td>Edward L. Bowen</td>
<td>29.95</td>
<td>2003</td>
<td>10</td>
</tr>
<tr>
<td>1-12-23</td>
<td>Seattle Slew</td>
<td>Dan Mearns</td>
<td>24.95</td>
<td>2000</td>
<td>0</td>
</tr>
<tr>
<td>0-07-24</td>
<td>Spectacular Bid</td>
<td>Timothy Capps</td>
<td>16.95</td>
<td>2001</td>
<td>3</td>
</tr>
</tbody>
</table>

Hash Index on Books.isbn
Hash Index on Books.title
Hash Index on Books.author
Hash Index on Customers.cid
B+Tree on Orders.ordernum

Yanlei Diao, University of Massachusetts Amherst
2/13/2006
Outline

• An outside look: DB Application
• An inside look: Anatomy of DBMS
• Project ideas: DB Application
• Project ideas: DBMS Internals
An Inside Look: Anatomy of DBMS

DBMS Architecture

Query Processor

Transactionsal Storage Manager

Disk Manager

Yanlei Diao, University of Massachusetts Amherst
SELECT C.cname, F.ordernum, F.order_date
FROM Customers C, OrderInfo F
WHERE C.cname = “John”
C.cid = F.cid

SELECT C.cname, O.ordernum, O.order_date
FROM Customers C, Orders O
WHERE C.cname = “John”
C.cid = O.cid

CREATE VIEW OrderInfo
(ordernum, cid, order_date)
AS
SELECT O.ordernum, O.cid, O.order_date,
FROM Orders O

Query Processor

• Syntax checking
• Internal representation

• Handling views
• Logical/semantic rewriting
• Flattening subqueries
Query Processor

- Building a query execution plan
  - Efficient, if not optimal
    - Define plan space
    - Cost estimation for each
    - Search algorithm
- Pull-based execution of a plan
  - Each operator is an Iterator: init(), next(), close()
Transactional Storage Manager

Access Methods
- IndexScan
  - Customers
    - cname="John"
  - Orders
- (On-the-fly)
  - C.cname,
    - O.ordernum,
    - O.order_date
- (Indexed Join)
  - cid=cid

Replacement policy,
Support for Concurrency & Recovery

Concurrency:
- 2PL

Recovery:
- WAL

Lock Manager
- Concurrency: 2PL

Buffer Manager
- Heap file, B+tree, Hash

Log Manager
- Recovery: WAL
Disk Manager

- Buffer Manager
  - Allocate/Deallocate a page
  - Read/Write a page
  - Contiguous seq. of pages

- Disk Space Manager

- Database
  - Data
  - Log
  - Indices
  - Catalog

Heap file
Page format
Record format
Outline

• An outside look: DB Application
• An inside look: Anatomy of DBMS
• Project ideas: DB Application
• Project ideas: DBMS Internals
Application: UMass CS Pub DB

• UMass Computer Science Publication Database
  – All papers on professors’ web pages and in their DBLP records
  – All technical reports

• Search:
  – Catalog search (author, title, year, conference, etc.)
  – Text search (using SQL “LIKE”)

• Navigation
  – Overview of the structure of document collection
  – Area-based “drill down” and “roll up” with statistics

• Add document
• Top hits

• Deliverables: useful software, user-friendly interface
Application: RFID Database

- RFID technology

reader_id, tag_id, timestamp
Application: RFID Database

- RFID technology
- RFID supply chain
  - Locations

Product Flow:
- Manufacturer
- Supplier DC
- Retail DC
- Retail Store
Application: RFID Database

- RFID technology
- RFID supply chain
  - Locations
  - Objects
Application: RFID Database

• RFID technology
• RFID Supply chain
• Database propagation
  – Streams of (reader_id, tag_id, time)
  – Semantics: reader_id → location, tag_id → object
  – Containment
    • Location-based, items in a case, cases on a pallet, pallets in a truck…
    • Duration of containment
  – History of movement: (object, location, time_in, time_out)
  – Data compression for duplicate readings
  – Integration with sensors: temperature, location…

• Track and trace queries
Outline

• An outside look: DB Application
• An inside look: Anatomy of DBMS
• Project ideas: DB Application
• Project ideas: DBMS Internals
New Directions for DB Research

- **XML**: new data model
- **Embedded DB**: new architecture
- **Streams**: new execution model
- **Data quality and provenance**: new services
- **Security**: new service
XML (Extensible Markup Language)

```
<bibliography>
  <book>
    <title> Foundations… </title>
    <author> Abiteboul </author>
    <author> Hull </author>
    <author> Vianu </author>
    <publisher> Addison Wesley </publisher>
    <year> 1995 </year>
  </book>
  ...
</bibliography>

XML: a tagging mechanism to describe content!
```
XML Data Model (Graph)

Main structure: ordered, labeled tree
References between node: becoming a graph
XQuery: XML Query Language

- A declarative language for querying XML data

- **XPath: path expressions**
  - Patterns to be matched against an XML graph
  - `/bib/paper[author/lastname='Croft']/title`

- **FLOWR expressions**
  - Combining matching and restructuring of XML data
  - for $p in distinct(document("bib.xml")//publisher)
    let $b := document("bib.xml")/book[publisher = $p]
    where count($b) > 100
    order by $p/name
    return $p
Metadata Management using XML

- File Searches
  - “all the files generated on Oct 1, 2005”
  - “all the files whose name is like ‘*simu*.txt’”
  - “all the files that were generated from the file ‘basic-measures.txt’”

- Build an XML store to manage directory trees!
  - XML data model
  - XML Query language
  - XML Indices
XML Document Processing

• Multi-hierarchical XML markup of text documents
  – Multi-hierarchies: part-of-speech, page-line
  – Features in different hierarchies overlap in scope
  – Need a query language & querying mechanism
  – References [Nakov et al., 2005; Iacob & Dekhtyar, 2005]

• Querying and ranking of XML data
  – XML fragments returned as results
  – Fuzzy matches
  – Ranking of matches
  – References [Amer-Yahia et al., 2005; Luo et al., 2003]

• Well-defined problems → identify your contributions!
X Wiki

• Collaborative authoring of structured data
• Application ideas
  – Balanced potluck
  – Calendar intersection
  – Schema evolution
• Build general but adaptable prototype
• Use XML data model, tools, principles
Lightweight DB in Applications

• Lightweight DB implementations linked to application programs (not client-server)

• In this project, investigate:
  – Usage (what’s driving their growth)
  – Comparison with RDBMS
  – Performance analysis
Sensor DB on Proxy Node

- **Sensor:**
  - temperature,
  - lighting...

- **Proxy:**
  - single board computer
  - limited storage compared to DB servers

- **Multi-resolution storage on proxy**
  - Per-sec information for a few hours
  - Per-hour information for a few days
  - Per-day information for a few months...

- **Queries**: max(), count(), avg(), percentile, etc. over period T with high accuracy
Sensor DB on Flash Memory

- **Sensor:**
  - GPS
  - Measurements...

- **Flash memory:**
  - local storage
  - reads/writes: different characteristics from magnetic disks!

- **Dynamic sensor network**
  - Each sensor stores its location (e.g., every minute)
  - When two turtles meet, exchange data
  - Other turtles/scientists ask turtle X, “where/when/how often have you seen turtle Y?”
  - Temporal-spatial indices in flash memory!
Data Stream Management

Traditional Database

- Data at rest
- One-shot or periodic queries
- Query-driven execution

Data Stream Processor

- Data in motion, unending
- Continuous, long-running queries
- Data-driven execution
XPath Stream Processing

• XPath widely used for handling XML messages
  – Authentication
  – Authorization
  – Transformation
  – Message routing

• Gigabit rate XPath processing using hardware
  – thwarted by buffering overhead

• Minimizing memory use of XPath processing

• DataPower / IBM
RFID Stream Processing

Shoplifting: an item was taken out of store without being checked out.

<pm1>
<tag>01.01298.6EF.0A</tag>
<time>00129038</time>
<location>shelf 2</location>
</pm1>

<pm1>
<tag>01.01298.6EF.0A</tag>
<time>02183947</time>
<location>exit1</location>
</pm1>
RFID Stream Processing

- Monitoring tasks (e.g., shoplifting, out-of-stocks) encoded as queries
- **Real time query processing** over RFID readings
  - No DB propagation!
- Extending existing languages
- Query plan-based fast implementation
- Handling incomplete readings
- **Walmart, Johnson&Johnson, DHL…**
Data Quality

- **Closed world assumption**: not any more!
- **Various sources of data loss**
  1) Sensing noise
  2) Data compression
  3) Lossy wireless links
  4) Incomplete merging
- **Probabilistic query processing**
  - Model sources of data loss
  - Quantify the effect on queries max(), avg(), percentile...
  - Output query results with confidence level
Versioning Databases

- Most databases do not preserve past states of the database after modifications.
- Design and implement a versioning database:
  - Deleted/modified tuples not removed, just marked.
  - Efficient queries/updates on current instance
  - Efficient queries on past instances
- Evaluate performance
  - Trade-offs: operations on current v. past instances
Fine-Grained Access Control

- Most DBMSs don’t provide tuple-level access control
- Extend an open-source DBMS with fine-grained access control
  - Tuples tagged with ownership
  - Indexes to improve performance
- Evaluate performance on workload
Passive Access Control in Embedded DB

• Passive access control
  – Crypto, instead of trusted process

• Design file format for embedded DB
  – Owner can write file in encrypted form
  – Owner can generate credentials
  – Reader process can access data only with credential
Questions