Advanced Topics in Database Systems

Prof. Yanlei Diao

University of Massachusetts Amherst
Outline

• Overview of database systems

• Course topics and requirements
Database

Database: a large, integrated collection of data.
Database Management System (DBMS): a software package designed to store and manage a large amount of data.
Database Design

• “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 1: 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 2: 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),
adress 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,
PRIMARY KEY(ordernum, isbn),
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 3: 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>

### 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>
Example SQL Queries

**Search Page**

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

**Login Page**

```
SELECT cid, username, password
FROM Customers
WHERE username = '<SpecifiedUsername>'
```
Step 4: Physical Design

- **Indexes:** auxiliary data structures to speed up searches. B+trees, hash indexes, R trees...

### 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

- **isbn number**
SQL Refresher

```
SELECT {DISTINCT} <list of columns>
FROM <list of relations>
{WHERE <list of "Boolean Factors">}
{GROUP BY <list of columns>}
{HAVING <list of Boolean Factors>}
{ORDER BY <list of columns>};
```

- **Query Semantics:**
  - Take Cartesian product (a.k.a. cross-product) of relns in FROM
  - If a WHERE clause exists, apply all filters in it
  - If a GROUP BY clause exists, form groups on the retained tuples
  - If a HAVING clause exists, filter groups with it
  - If an ORDER BY clause exists, make sure tuples are output in the right order
  - Project tuples onto a set of columns. If there is a DISTINCT modifier, remove duplicates
Basics of Query Optimization

LOCATION: WHERE condition

- Convert the WHERE condition to **conjunctive normal form**: 
  - (day<8/9/94 OR bid=5 OR sid=3 ) AND (rname='Paul' OR sid=3)

LOCATION: Plan tree

- Interleave FROM and WHERE into a plan tree for optimization.

LOCATION: ORDER

- Apply GROUP BY, HAVING, DISTINCT and ORDER BY at the end, pretty much in that order.

SELECT {DISTINCT} <list of columns>
FROM <list of relations>
{WHERE <list of "Boolean Factors">}
{GROUP BY <list of columns>}
{HAVING <list of Boolean Factors>}
{ORDER BY <list of columns>};
Expression in Relational Algebra (RA):

\[ \pi_{\text{cname, ordernum, order\_date}} (\sigma_{\text{cname}=\text{``John''}} (\text{Customers} \bowtie_{\text{cid}=\text{cid}} \text{Orders})) \]

The algebraic expression partially specifies how to evaluate the query:
- Compute the natural join of Reserves and Sailors
- Perform the selections
- Project the \textit{sname} field
Query Evaluation Plan

• **Query evaluation plan** is an extended *RA tree*, with additional annotations:
  – *access method* for each relation;
  – *implementation method* for each relational operator.

• **Optimization opportunities:**
  – Pushing selections down the plan
  – Using indexes
  – More efficient join algorithms
  – …
DBMS Architecture

Query Processor

Query Parser
Query Rewriter
Query Optimizer
Query Executor

Transactional Storage Manager

Lock Manager
Access Methods
Buffer Manager
Log Manager

Disk Space Manager

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

SELECT C.cname, F.ordernum, F.order_date FROM Customers C, OrderInfo F WHERE C.cname = “John” AND C.cid = F.cid

SELECT C.cname, F.ordernum, F.order_date FROM Customers C, OrderInfo F WHERE C.cname = “John” AND C.cid = F cid

- Syntax checking
- Internal representation
- Handling views
- Search algorithm
- Pull-based execution of a plan
- Each operator is an Iterator: init(), next(), close()
Transactional Storage Manager

Access Methods

- Heap file, B+tree, Hash
- Lock Manager: Concurrency: 2PL
- Buffer Manager: Replacement policy
- Log Manager: Recovery: WAL

Indexed Join:
- cid=cid
- (Indexed Join)
- C.cname, O.ordernum, O.order_date

On-the-fly:
- IndexScan Customers: cname=“John”
- IndexScan Orders
Disk Manager

Allocate/Deallocate a page or contiguous seq. of pages; Read/Write a page.

Database

Data

Indexes

Log

Catalog

Buffer Manager

Disk Space Manager
DBMS: Theory + Systems

- Query Parser
- Query Rewriter
- Query Optimizer
- Query Executor
- Lock Manager
- Access Methods
- Buffer Manager
- Log Manager
- Disk Space Manager

Theory!
Systems!