Course Overview

- **Highlights:**
  - Modeling: Scenario to ER-diagrams to Schema to Relational Models
  - Schema Refinement: Various Normal Forms, BCNF holy grail
  - Query Optimization: Indexing, relational algebra, algorithms, sorting, math!
  - LAMP Development: LinuxApacheMySQLPHP
  - SQL Server 2014 (beautiful query optimizer)
  - Amazon AWS grant for MongoDB in cloud! (just be sure to shut your server down…)
  - SQL…the language! (including DDL)
  - Fun group projects at the end of class (more later)
  - Some days in CS lab for hands-on work (about 5-6)
    - You will be Linux pros.
Course Goals

Musicbrainz.org ER Diagram

Course Tools

Servers:
- SQL Server 2014 on NELSON\SQLSERVERCS445 (on campus only)
- LAMP on gray.cs.pacificu.edu (on campus only)
- AWS for MongoDB

Software:
- Aptana 3 for Web/PHP Development (Linux & Win)
- MS SQL Server Management Studio 2014 (Win)
- MySQL Workbench 6.3 (Win)
- SSH/FTP client (Win)
Introduction

- Read Chapter 1
- What is a database?
- Why do we use them?
- Who uses them?
- How do we use them?

Example: Airline Reservations

Build a system for making airline reservations
- What is done in the three different tiers of the application?
- Database System:
  - Airline info, available seats, customer info, etc.
- Application Server:
  - Logic to make reservations, cancel reservation, add new airlines, etc.
- Client Program:
  - Log in as different users, display forms and human-readable output
Example 1: Airline Reservations, cont.

Why Store Data in a DBMS?

- Benefits:
  - Transactions (concurrent data access, recovery from system crashes)
  - High-level abstractions for data access, manipulation, and administration
  - Data integrity and security
  - Performance and scalability
What is a Transaction?

- The execution of a program that performs a function by accessing a database.
- Examples:
  - Reserve an airline seat. Buy an airline ticket.
  - Withdraw money from an ATM.
  - Verify a credit card sale.
  - Order an item from an Internet retailer.
- A transaction will commit after completing all of its actions, or it could abort (or be aborted by the DBMS) after executing some actions.

Transactions

- A transaction is an atomic sequence of actions
- Each transaction must leave the system in a consistent state (if system is consistent when the transaction starts.)
- The ACID Properties
  - Atomicity
  - Consistency
  - Isolation
  - Durability
Example Transaction: Online Store

- Your purchase transaction:
  - Atomicity: Either the complete purchase happens, or nothing.
  - Consistency: The inventory and internal accounts are updated correctly.
  - Isolation: It does not matter whether other customers are also currently making a purchase.
  - Durability: Once you have received the order confirmation number, your order information is permanent, even if the site crashes

Concurrency Control for Isolation

(Assume A=$100, B=$100)

Consider two transactions
- T1: Start: A=A+100, B=B-100, COMMIT
- T2: Start: A=1.06*A, B=1.06*B, COMMIT

The first transaction is transferring $100 from B’s account to A’s account. The second is crediting both accounts with a 6% interest payment.

Database systems try to do as many operations concurrently as possible, to increase performance.
Example (cont).

(Start A=$100, B=$100)

- Consider a possible interleaving (schedule):
  T1: A=A+100, B=B-100 COMMIT
  T2: A=1.06*A, B=1.06*B COMMIT
  End Result: A=$212, B=$0

- Consider a possible interleaving (schedule):
  T1: A=A+100, COMMIT
  B=B-100 COMMIT
  T2: A=1.06*A, B=1.06*B COMMIT
  End Result: A=$212, B=$6

NOTE: The second interleaving is incorrect! The concurrency control of a database system makes sure that the second schedule does not happen.

Ensuring Atomicity

- DBMS ensures atomicity (all-or-nothing property) even if the system crashes in the middle of a transaction.
- IDEA: Keep a log (history) of all actions carried out by the DBMS while executing:
  - Before a change is made to the database, the corresponding log entry is forced to a safe location.
  - After a crash, the effects of partially executed transactions are undone using the log.
Why Store Data in a DBMS?

- **Benefits**
  - Transactions (concurrent data access, recovery from system crashes)
  - High-level abstractions for data access, manipulation, and administration
  - Data integrity and security
  - Performance and scalability

Data Modeling

- To do their job, databases store information in a structured manner.
- Structure used to store information is called the **Data Model**
- Most common model is the **Relational Model**
  - stores data in tables, rows and columns like a spreadsheet
  - other models exist (E-R, object oriented, etc.)
  - this course concentrates on Relational Model
The Relational Model

A relational database is a set of relations. Turing Award (Nobel Prize in CS) for Codd in 1980 for his work on the relational model.

- Example relation:

```
Customers (cid: integer, name: string, byear: integer, state: string)
```

<table>
<thead>
<tr>
<th>cid</th>
<th>Name</th>
<th>byear</th>
<th>state</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Jones</td>
<td>1960</td>
<td>NY</td>
</tr>
<tr>
<td>2</td>
<td>Smith</td>
<td>1974</td>
<td>CA</td>
</tr>
<tr>
<td>3</td>
<td>Smith</td>
<td>1955</td>
<td>OR</td>
</tr>
</tbody>
</table>

The Relational Model: Terminology

- Relation instance and schema
  ```
  Customers (cid: integer, name: string, byear: integer, state: string)
  ```

- Field (column)
- Record or tuple (row)
Product Relation

- Relation schema:
  
  Products(pid:integer, pname: string, price:float, category:string)

- Relation instance:

<table>
<thead>
<tr>
<th>pid</th>
<th>pname</th>
<th>price</th>
<th>category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Intel PIII-700</td>
<td>300.00</td>
<td>hardware</td>
</tr>
<tr>
<td>2</td>
<td>MS Office PRO</td>
<td>500.00</td>
<td>software</td>
</tr>
<tr>
<td>3</td>
<td>IBM DB2</td>
<td>5000.00</td>
<td>software</td>
</tr>
</tbody>
</table>

Transaction relation

- Relation schema:
  
  Transactions (tid:integer, tdate: date, cid: integer, pid: integer)

- Relation instance:

<table>
<thead>
<tr>
<th>tid</th>
<th>tdate</th>
<th>cid</th>
<th>pid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1/1/2015</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1/23/2015</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2/1/2015</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>
Query Languages

We need a high-level language to describe and manipulate the data.

Requirements

- Precise semantics
- Easy integration into applications written in C++/JAVA/Visual Basic/HTML/etc.
- Easy to learn
- DBMS needs to be able to efficiently evaluate queries written in the language.

SQL: Structured Query Language

- Developed by IBM (System R) in the 1970's
- ANSI standard since 1986
  - SQL-86
  - SQL-89 (minor revision)
  - SQL-92 (major revision)
  - SQL-99 (major extensions, triggers, embedded SQL)
  - SQL:2003 (XML)
  - SQL:2006 (mainly more XML support, XQuery)
  - SQL:2008
  - SQL:2011
- Much more about SQL in upcoming lectures.
Example Query

- Example Schema:

```sql
Customers (cid: integer,
    name: string,
    byear: integer,
    State: string)
```

- Query:

```sql
SELECT Customers.cid, Customers.name, Customers.byear, Customers.state
FROM Customers
WHERE Customers.cid = 1
```

<table>
<thead>
<tr>
<th>cid</th>
<th>name</th>
<th>byear</th>
<th>state</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Jones</td>
<td>1960</td>
<td>NY</td>
</tr>
<tr>
<td>2</td>
<td>Smith</td>
<td>1974</td>
<td>CA</td>
</tr>
<tr>
<td>3</td>
<td>Smith</td>
<td>1955</td>
<td>OR</td>
</tr>
</tbody>
</table>

Why Store Data in a DBMS?

- Benefits
  - Transactions (concurrent data access, recovery from system crashes)
  - High-level abstractions for data access, manipulation, and administration
  - **Data integrity and security**
  - Performance and scalability
**Integrity Constraints**

- Integrity Constraints (ICs): Condition that must be true for any instance of the database.
  - ICs are specified when the schema is defined.
  - ICs are checked when relations are modified.
  - A legal instance of relation is one that satisfies all specified ICs

- Example: Domain Constraints

---

**Primary Key Constraints**

- A set of fields is a **superkey** for a relation if no two distinct tuples can have the same values in all key fields.
- A set of fields is a **key** if the set is a superkey and none of its subsets is a superkey (so, smallest superkey)

- Example:
  - \{cid, name\} is a superkey for Customers
  - \{cid\} is a key for Customers
Foreign Keys, Referential Integrity

- Foreign Key: Set of fields in one relation that refers to a unique tuple in another relation. (NOTE: The foreign key must be a superkey of the second relation).
- If all foreign key constraints are enforced, we say that referential integrity is achieved.
  - No dangling references.
  - Compare to links in HTML

Foreign Keys: Example

- The cid field of the Transactions relation refers to the cid field of the Customer relation:

<table>
<thead>
<tr>
<th>tid</th>
<th>tdate</th>
<th>cid</th>
<th>pid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1/1/2004</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1/23/2004</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2/1/2004</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cid</th>
<th>Name</th>
<th>byear</th>
<th>state</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Jones</td>
<td>1960</td>
<td>NY</td>
</tr>
<tr>
<td>2</td>
<td>Smith</td>
<td>1974</td>
<td>CA</td>
</tr>
<tr>
<td>3</td>
<td>Smith</td>
<td>1955</td>
<td>OR</td>
</tr>
</tbody>
</table>
Enforcing Referential Integrity

- What should be done in a Transaction tuple with a non-existent Customer ID is inserted? (Reject it)

- What should be done if a Customer tuple is deleted?
  - Also delete all Transaction tuples that refer to it.
  - Disallow deletion of a Customer tuple that has associated Transactions.
  - Set cid in transactions tuples to a default or special cid.

- SQL supports all three choices.

Security

- Secrecy: Users should not be able to see things they are not suppose to:
  - For example. A student should not be able to see another students' grades.

- Integrity: Users should not be able to modify things they are not supposed to.
  - For example: Only instructors can assign grades.

- Availability: Users should be able to see and modify things they are allowed to see and modify.
Why Store data in a DBMS

- Benefits
  - Transactions (concurrent data access, recovery from system crashes)
  - High-level abstractions for data access, manipulation, and administration
  - Data integrity and security
  - Performance and scalability

DBMS and Performance

- Efficient implementation of all database operations
- Indexes: Auxiliary structures that allow fast access to the portion of data that a query is addressing.
- Smart buffer management
- Query optimization: Finds the best way to execute a query.
- Automatic high-performance concurrent query execution
- Allows for Parallel and Distributed architectures.
Summary of DBMS Benefits

- Transactions
  - ACID properties, concurrency control, recover
- High-level abstractions for data access
  - Data models
- Data integrity and security
  - Key constraints, foreign key constraints, access control
- Performance and scalability
  - Parallel DBMS, distributed DBMS, performance tuning.