University of Regina

CS 215
Web Oriented Programming

Database Fundamentals

Dr. Orland Hoeber
orland.hoeber@uregina.ca
http://www.cs.uregina.ca/~hoeber/cs215/

Persistent Data Storage For Web Apps

- If we want to maintain data for any non-trivial length of time, we need to store it on the server
- Compared to storing it on the client (browser), storing it on the server is more:
  - reliable
  - maintainable
  - secure
- In almost all web application cases, such data is stored in a database
  - the goal of this lecture is to explain to you why this is the case

The Problems with File Storage

- Suppose we have a collection of data that is relatively large (e.g., 500 GB of customer records)
- If we want to access some of this within a web application, we will need to manage a few details:
  - load the entire dataset into memory
  - If the data is too large, devise a scheme for only loading relevant elements into memory
  - develop a scheme for addressing the data elements
  - write software that will support adding, editing, or changing the data
  - ensure that the software will support concurrent access and modifications
  - ensure that the software will handle crashes and reboots gracefully
- There is a lot of software development overhead here

Database Management Systems

- Database management systems (DBMS) take care of all of these problems for you
  - Data independence
    - data representation and storage are handled independently of the application program
  - Efficient data access
    - data storage and retrieval can occur with minimal delay
  - Data integrity and security
    - rules regarding valid data can be automatically enforced
    - access to the data can be restricted depending on who or what application is accessing it

DBMS (continued)

- Data administration
  - the data can be administered independent of the application
- Concurrent access
  - multiple users can access the data at the same time
  - maintains the illusion that each has dedicated access
- Crash recovery
  - if the computer crashes, no data is corrupted or lost
  - any changes to the data either happen completely or don't happen at all

- Fundamentally, the use of a DBMS will result in significant reductions in application development time
- However, in some special cases, databases may not be appropriate (e.g., real-time data processing)

Data Model

- A **data model** is a set of high-level data description constructs that allow us to specify what the data should look like without having to worry about the low-level storage details
- The specific data model used in most commonly is the **relational data model**
  - the central data construct in this model is the relation, which can be though of as a set of records
  - don't confuse this with a relationship, which is also present in the relational model

Schema

- A specific description of data in terms of a data model is a **schema**
- Within a relational data model, the schema for a relation specifies:
  - the name of the relation
  - the name of each field (or attribute, or column)
  - the type of each field

```
Students (sid: integer, name: string, login: string,
dob: date, gpa: float)
```

- Logically, this gives us a template for defining a table structure, where each row is a record that describes a student

Levels of Abstraction

- The data in a database can be described at three levels of abstraction:
  - conceptual
  - physical
  - external

- These all represent the database, but from different perspectives and purposes

Conceptual Schema

- The **conceptual schema** describes the database in terms of the data model
  - information about entities
  - relationships between the entities
- All of this information can be stored in a series of relations (tables)
- The choice of relations and fields in a relation is not always obvious
- The process of arriving at a good conceptual schema is called **conceptual database design**

Conceptual Database Design

```
Student (sid: integer, name: string, login: string,
dob: date, gpa: float)

Faculty (fid: integer, name: string)

Courses (cid: integer, cname: string,
credits:integer)

Enrolled (sid: integer, cid: integer, grade: float)

Teaches (fid: integer, cid: integer)
```

Physical Schema

- The **physical schema** specifies storage details for the conceptual schema
  - which file organizations to use to store the relations
  - what auxillary data structures can be created to speed up storage and/or retrieval operations (indexes)

- Decisions on the physical schema are made based on an understanding of how the data is typically accessed

External Schema

- **External schemas** allow data access to be customized and authorized at the level of individual users and groups
  - different users can share access to the same database, but with different rights and privileges
  - a simple example is table-level access controls
  - a more complex example is custom views that combine different elements from different related tables
    - e.g., a view that shows student enrollment in a course, their grades, but not their login information
- Such external schemas are guided by the end-user requirements for the application

Data Independence

- A fundamental feature of using a DBMS within any application is that it offers **data independence**
  - application programs are insulated from changes in the way the data is structured and stored
- There are two different types of data independence:
  - logical data independence
    - users are shielded from changes in the logical structure of the data or changes in the relations
    - achieved through the use of views
  - physical data independence
    - changes in how the data is stored internally within the database have no impact on how the users access the data

Queries

- What makes databases popular is the ease by which information can be accessed
- Relational databases allow for a rich class of questions to be easily posed to the database
  - such questions are called **queries**
  - they are written in a **query language**
  - in addition, users can create, modify, and delete data through a **data manipulation language**
- The language we will use for these purpose is called the **structured query language**, or **SQL**

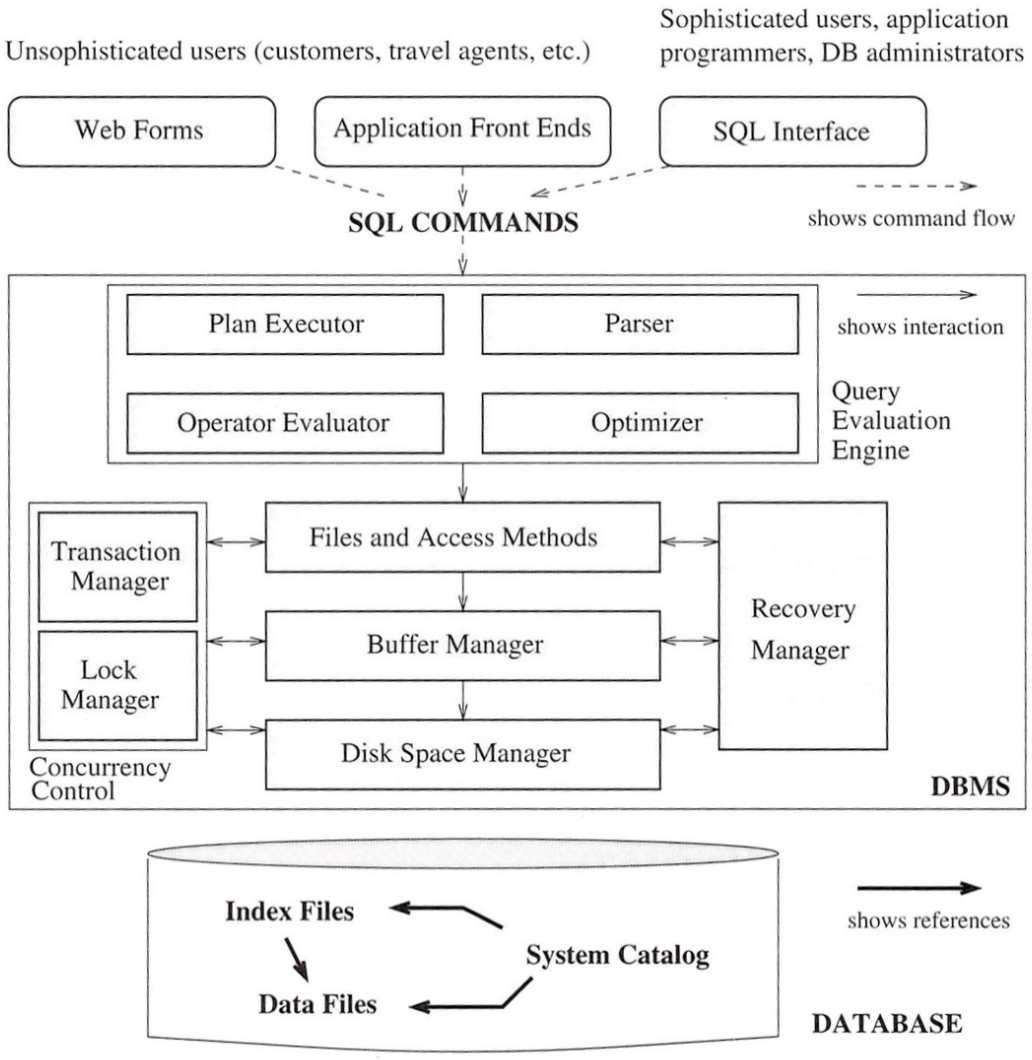
Transaction Management

- An important feature of databases is the ability to allow concurrent users of the database
- In order to avoid conflicts, the database must have a concept of the basic unit of change: the **transaction**
- The illusion of concurrent access is achieve by carefully interleaving the transactions of each user
- Problems are avoided by using a locking protocol on the relations
  - queries can use a **shared lock**
  - modifications require the use of an **exclusive lock**

Incomplete Transactions & Crashes

- When a transaction is interrupted (e.g., due to a crash), it is important to be able to recover from it gracefully
- This is done using a **write-ahead log**
  - write the details of the transaction to the log first
  - force the log to be written to the disk
  - execute the transaction on the database
- In this way, if a crash occurs, the DBMS can check the log and the status of the database to find out if the last few transactions were completed or not

Components & Structure



Databases & Tables (Relations)

- Tables
  - data is organized over rows and columns
- Columns
  - columns are named
  - assigned specific data types
- Rows
  - rows are unnamed
  - values are (normally) assigned for each column
  - the rows represent entities, where the column values are the attributes for the entities
- Primary Keys
  - most tables will have a special column that contains a unique identifier for each row – the primary key
  - normally an integer value

user_id	username	password	last_access
1	tim	xxkkddi	07-11-14
2	jill	33kk99d	05-11-14

Simple Database Tables

- The simplest database table will contain a primary key, and a column for each attribute of the data
  - this can result in a lot of wasted space and redundant data
  - a better approach is to think about the meaning of the data in a careful and logical manner, decomposing it into multiple related tables that eliminate the redundancy

car_id	make	model	engine	hp	doors	price
1	Chevrolet	Volt	electric	149	4	$28,000
2	Audi	A4	4 cyl turbo	210	4	$44,000

Normalization

- The process of separating your data into tables and creating keys that link them together is called normalization
- Goal: ensure that each piece of information appears in the database only once
  - duplicate data is a waste of space
  - when data that is duplicated and the database needs to be updated, there is a risk of the data becoming inconsistent (updated in one location, but not others)
- There are three standard forms of normalization: first, second, and third

First Normal Form (1NF)

- The database must fulfill the following requirements:
  - there should be no repeating columns containing the same data
  - all columns should contain a single value
  - there should be a primary key to uniquely identify each row
- This is simple to do with careful conceptual database design

Second Normal Form (2NF)

- To achieve second normal form, all the tables must start in first normal form
- Then, the goal is to eliminate redundancy across multiple rows by creating new tables that hold this repeating information once
  - e.g., suppose we have a database table that holds information about product orders. For repeat purchases, we should not duplicate the name and address of the customer within the orders table, but instead create a new table that holds customers independently from their orders.

Third Normal Form (3NF)

- Once you have the database in both first and second normal form, it is probably in good shape and may not need any further modifications
- Third normal form is the process of eliminating all other redundancy from the database
  - any data that is dependent on another value in the table should also be moved to a separate table
  - e.g., a city can be derived from the postal code, so this should be in a separate table
- **For our purposes, 3NF will result in too many tables and rather complex queries**

Entity Relationship Diagrams (ERD)

- ERDs are a useful tool for modeling the conceptual schema associated with a given problem or domain
  - describes the data (entities and attributes of the entities)
  - describe the relationships between the entities

- We can use language constructs to identify the features of an ERD
  - nouns: entities
  - adjectives: attributes
  - verbs: relationships

Entities

- Entities are the basic objects of ERDs
  - represent a “thing” with an independent existence
  - may be physical or conceptual
    - car, car type, manufacturer, engine
    - student, course, faculty, instructor

- Represented as blocks in the ERD

Relationships

- Relationships are the associations between the entities
  - can be inclusion (like in OO programming) or a more abstract connection

- There are three different types of relationships:
  - one to one:
  - one to many: 
  - many to many: 

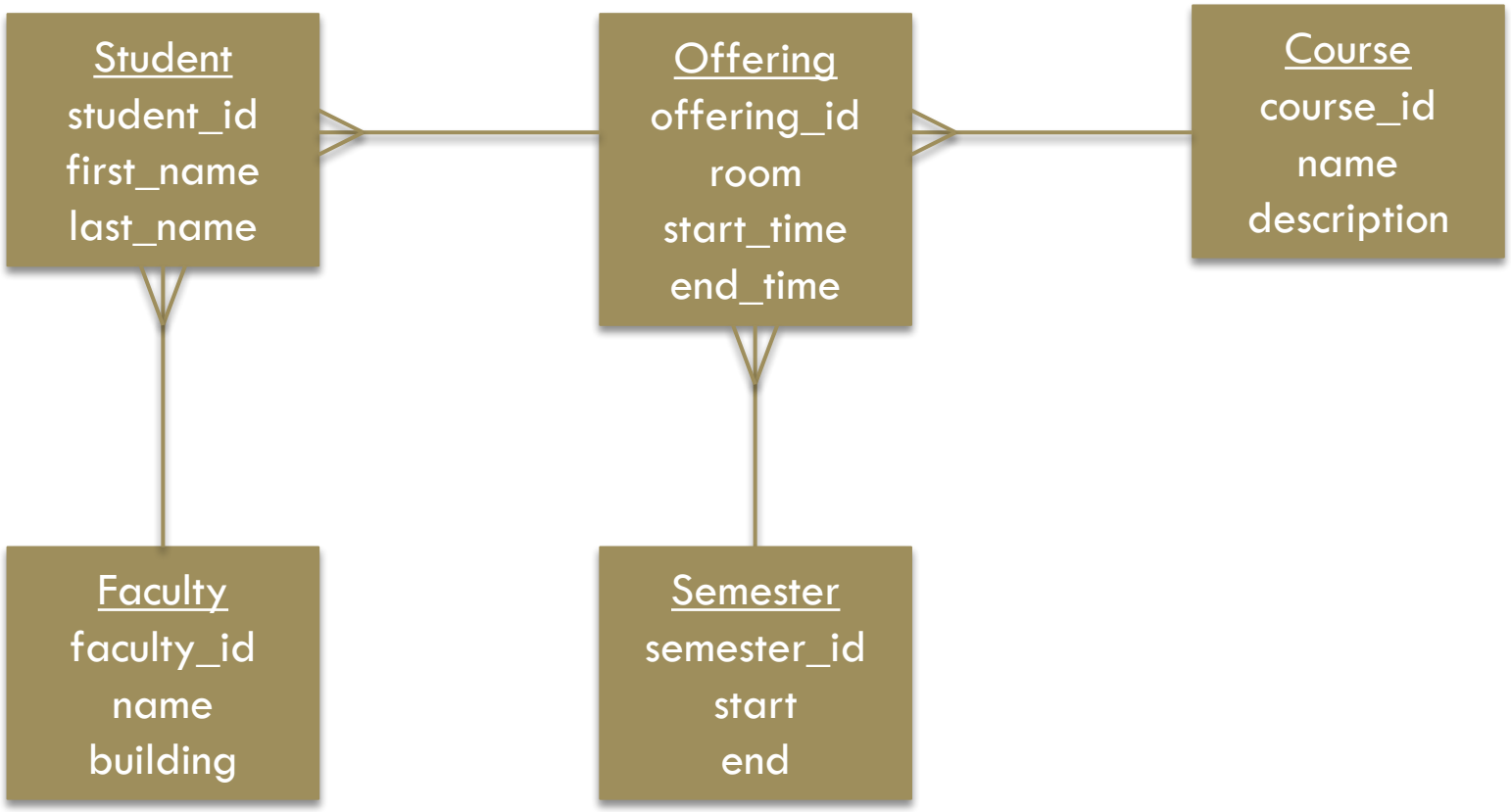
Attributes

- Attributes can be assigned to the entities and/or the relationships
  - for entity attributes, these can be listed within the entity box
  - relationship attributes are normally added as bubbles connected to the relationship line

A Simple Example

- Suppose we are modeling data for students attending courses across different faculties:
  - students can take multiple courses
  - courses are offered in semesters
  - students belong to faculties

- What are the entities?
- What are the attributes?
- What are the relationships?


Student
student_id
first_name
last_name
Offering
offering_id
room
start_time
end_time
Course
course_id
name
description
Faculty
faculty_id
name
building
Semester
semester_id
start
end

The Value of ERDs

- The main value of carefully constructing an ERD is that it can readily be converted into a database structure:
  - entities become tables
  - relationships are represented with keys from one table into another
  - many to many relationships require separate tables to link the two entity tables

Final Words on Data Models

- Although our focus is on databases that implement the relational data model, there are other types of data models that are useful in special circumstances:
  - hierarchical data model
  - network data model
  - object-oriented data model
  - object-relational data model
  - semi-structured data model

- We have seen one of these data models in this class already (DOM), and will see another very soon (JSON)

Homework

- Read Chapter 8 & 9
- Next topic: Databases & MySQL
- Upcoming deadlines:
  - Assignment 4: Tuesday Nov 21@ 11:55PM