Lecture 2
Entity-Relationship Model

Eugene Wu
Fall 2015

HW 0

VM vs virtualenv vs folder vs python shell

Maybe talk about GIT workflows

How to ask for help
https://github.com/w4111/syllabus#help

The TA Situation

Why the different results?

Result: 69

```
import csv
file = open('iowa-liquor-sample.csv')
file_reader = csv.reader(file)
n = 0
for row in file_reader:
  for el in row:
  if "single malt scotch" in el.lower():
    n += 1
print n
```

Result: 51

```
file = open('iowa-liquor-sample.csv','r')
n = 0
for line in file:
  temp = line.lower()
  if 'single malt scotch' in temp:
    n += 1
print n
```

HW0 Stats

enrolled 79
on waitlist 84

http://eugenewu.net/students.html

Overview of DBMS Components

Classic Components in Databases

- Concurrency Control
- Transactions
- Atomicity
- Recovery and Logs

Transaction: Execution of a DB Program

Def: atomic sequence of DBMS actions

```
Begin;
<read beth's account>
<deduct from beth's account>
<increase eugene's account>
Commit;  (or Abort;)
```

Transaction: Execution of a DB Program

Def: atomic sequence of DBMS actions

Each fully executed transaction must leave DB in *consistent state* if DB is consistent before transaction

- Users specify simple *integrity constraints* on data, and DBMS enforces the constraints.
- DBMS does not understand semantics of its data
  e.g., doesn't know how bank interest is computed
- User's responsibility to ensure transaction (run alone) preserves consistency

Concurrency Control

Concurrently running multiple user programs needed for good performance
- Disk accesses are frequent & slow.
- Keep CPU working on several user programs while waiting.

Concurrency can cause inconsistencies
- e.g., check cleared while account balance being computed.
- *Really* hard to program against

DBMS ensures such problems don't arise
- programmers can pretend to use a single-user system.

Scheduling Concurrent Transactions

Transactions T1, …, Tn are run concurrently
Equivalent to a *serial* ordering (as if no concurrency)

Locks: T_i requests and waits for lock before read/write.
- e.g., T_i locks the database, updates, then releases
- e.g., T_i locks the table, updates, then releases
- e.g., T_i locks rows, updates, then releases

Will talk about how this works later in course.

Atomicity

Def: Xact fully completes, or never happened even after failures e.g., crashes

Record all actions Xact did during execution in a log

1. Write ahead logging: before making any change, ensure the change is safely recorded in log
2. After failure, read log and undo any incomplete Xacts

The Log

A log record contains enough info to undo actions:
- Transaction id
- T_i writes an object: old and new values
- Log record *must* be safely stored before the changed data
- T_i commits/aborts: store commit/abort action

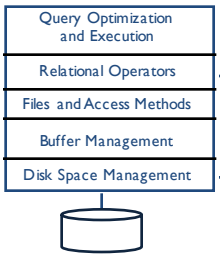
All logging, recovery and concurrency control activities hidden away from user.

Classic Structure of a DBMS

These layers must consider concurrency control and recovery

Typical layered architecture
DBMS, not OS, manages memory and disk

Doesn't show concurrency control & recovery components



Steps for a New Application

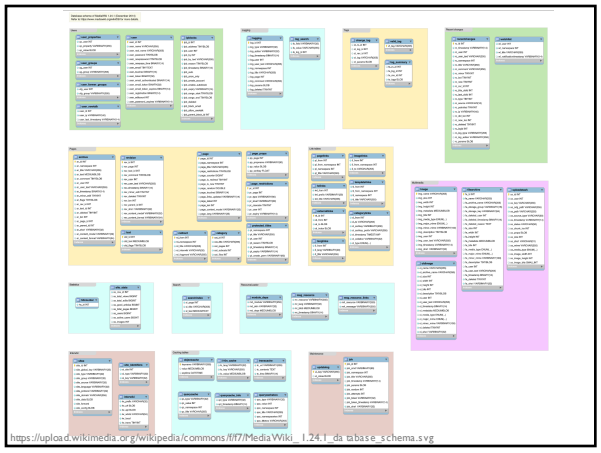
Requirements
what are you going to build?
Conceptual Database Design
pen-and-pencil description
Logical Design
formal database schema
Schema Refinement:
fix potential problems, normalization
Physical Database Design
use sample of queries to optimize for speed/storage
App/Security Design
prevent security problems

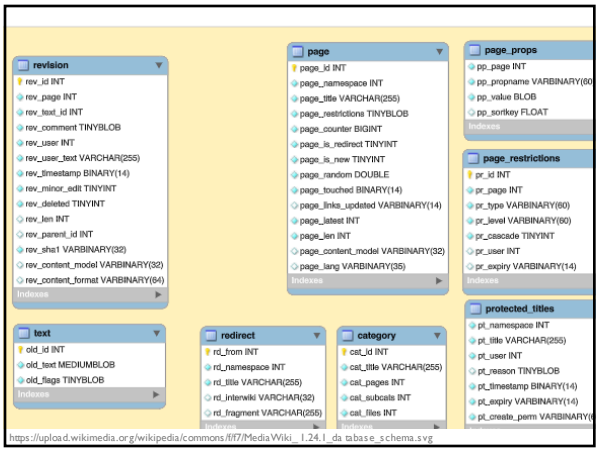
Steps for a New Application

Requirements
what are you going to build?
Conceptual Database Design
pen-and-pencil description
ER Modeling
Logical Design
formal database schema
Schema Refinement:
fix potential problems, normalization
Physical Database Design
use sample of queries to optimize for speed/storage
App/Security Design
prevent security problems

Database Apps Are Complicated

Typical Fortune 100 Company
~10k different information (data) systems
90% relational databases (DBMSes)
Typical database has >100 tables
Typical table has 50 – 200 attributes

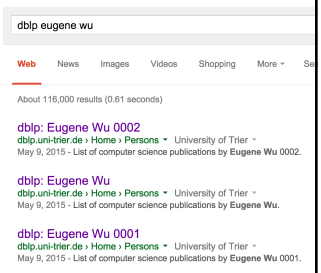




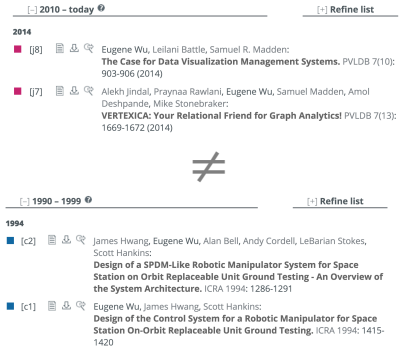
Inconsistencies/Constraint Violations

Huge amount of effort to avoid inconsistencies

DBLP is *the* site for computer science publications

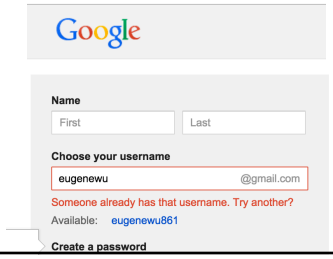


Inconsistencies/Constraint Violations

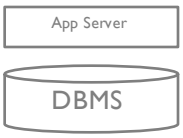


Inconsistencies/Constraint Violations

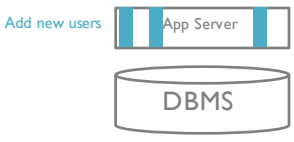
Giving me eugenewu@gmail would violate constraints



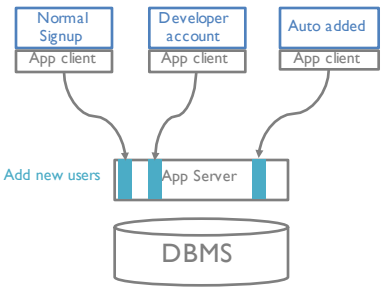
It is Hard to Design Applications



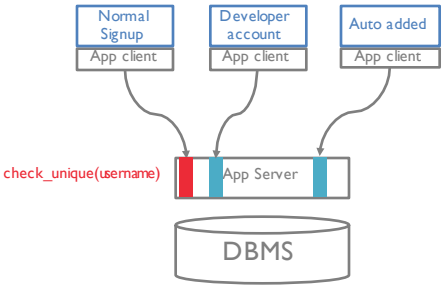
It is Hard to Design Applications



It is Hard to Design Applications

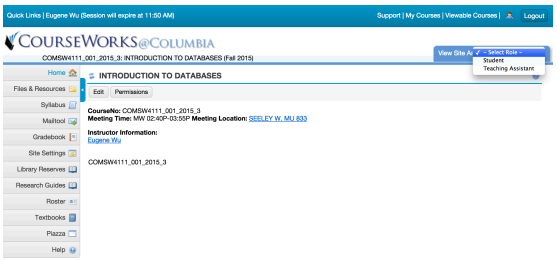


It is Hard to Design Applications



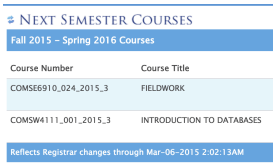
Let's make ~~a webapp~~ $$$

live exercise time

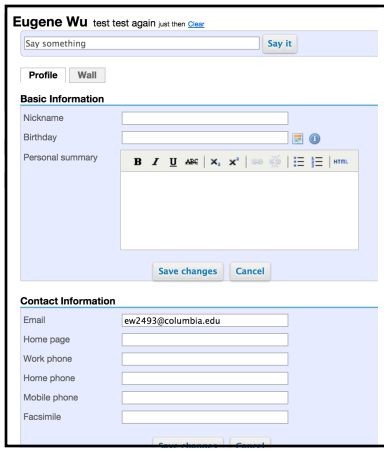


Entity-Relationship Modeling

Entities (objects) to store and their attributes
Relationships between entities and their attrs.
Integrity constraints & business rules
Visually modeled, easy to turn into DB schema



Courses
- Course Number
- CourseTitle
- Year
- Semester



Users
- Nickname
- Name
- Birthday
- Summary
- Email
- …

Basics: Entities

Entity e.g., intro to databases
- real-world object distinguishable from other objects
- described as set of attribute & the values
- (think one record)

Entity Set e.g., all courses
- collection of similar entities
- all entities have same attributes (unless Is-A)
- must have one or more keys
- attributes have domains
- ≈ table

Example: Entity

Keys (cid, uid) are underlined

Values must be unique

(think: can use as hashtable key to lookup table)

Basics: Relationships

Relationship: association between 2 or more entities

e.g., alice is taking Introduction to DBs

Relationship Set: collection of similar relationships

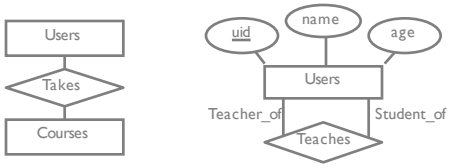
N-ary relationship set R relates N entity sets $E_1 \ldots E_n$

Each $r \in R$ involves entities $e_1 \ldots e_n$

An E_i can be part of diff. relationship sets or diff. roles in same set

Basics: Relationships

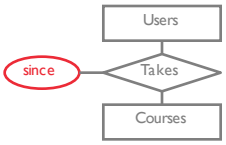
Users takes diff roles in same relationships set



Basics: Relationships

Relationships sets can have descriptive attributes

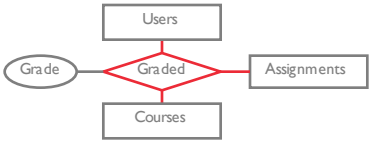
e.g., the *since* attribute of Instructs



Basics: Ternary Relationships

Connects three entities

N-ary relationships possible too.



Constraints

Help avoid corruption, inconsistencies

- Key constraints
- Participation constraints
- Weak entities
- Overlap and covering constraints

Key Constraints

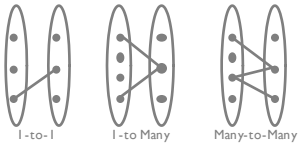
Defines cardinality requirements on relationships

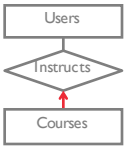
Many to many e.g., consider *Takes*

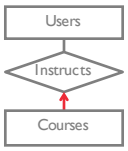
- a user can take many courses
- a course can have many users that take the course

One to Many e.g., consider *Instructs*

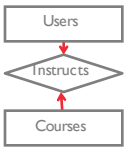
- a course has at most one instructor







A course has at most one instructor

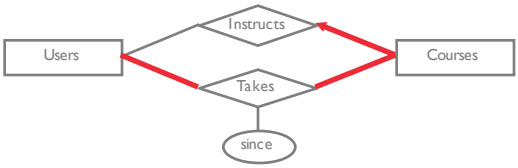


???

Participation Constraints

Does every course need an instructor?

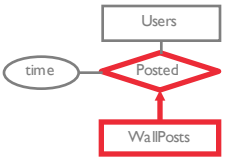
- If yes, it's a participation constraint
  - e.g., participation of Courses in instructs is *Total*
- Otherwise, *partial* participation constraint

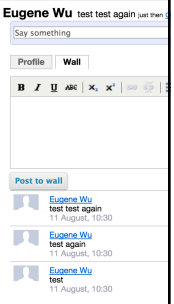


Weak Entities

A *weak entity* can only be uniquely identified by using the primary key of its owner entity

- Owner and weak entity sets must be in one to many relationship set
- Weak entity set must have total participation in this *identifying* relationships set





At least one

At most one

Exactly one

Weak Entity

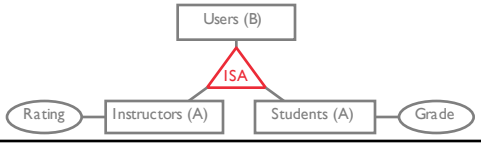
ISA (is a) Hierarchies

Inheritance rules similar to programming languages

- A ISA B → every A also considered a B
- When querying for Bs, must consider As (unlike e.g., C++)

Why use ISA?

- add descriptive attributes specific to a subclass e.g., grade
- identify entities that participate in a relationship



ISA (is a) Hierarchies

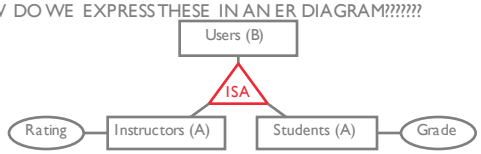
Overlap Constraint

can eugene be an instructor and a student? (allow/disallow)

Covering Constraint

must every user be an instructor or student? (yes/no)

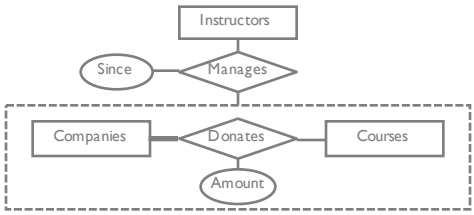
HOW DO WE EXPRESS THESE IN AN ER DIAGRAM???????



Aggregation

Relationships between (entities – relationships)

Lets us treat a Relationship Set like an Entity Set so it can participate in other relationships

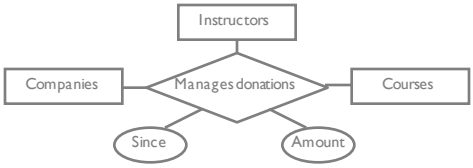


Aggregation vs Ternary Relationships

Manages and Donates are distinct relationships with own attrs

Can define constraints on relationship sets

e.g., a donation can be managed by at most one instructor

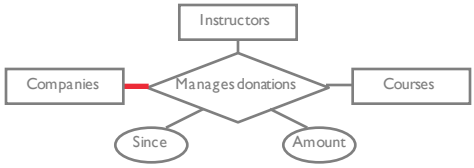


Aggregation vs Ternary Relationships

Manages and Donates are distinct relationships with own attrs

Can define constraints on relationship sets

e.g., a donation can be managed by at most one instructor



Using the ER Model

Design Choices for a concept

- Entity or Attribute?
- Entity or Relationship?
- Binary or Ternary relationship?
- Aggregation or Ternary relationship?

Constraints in ER Modeling

- Many types of data semantics can be captured using ER
- Some constraints not captured (discuss limitations later)

Need further schema refinement

- ER Model is still subjective, need further refinement after translated into relational schema

Entity or Attribute?

Is users.address an attribute of Users or an entity connected to Users by a relationship?

Depends (and may change over time!)

- If a user has >1 addresses, must be an entity
- If an address has attrs (structure), must be entity

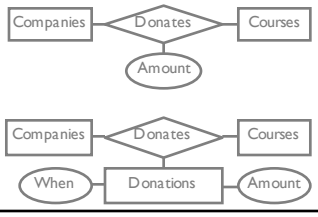
e.g., want to search for users by city, state, or zip

Entity or Attribute?

A company can't donate multiple amounts (top fig)

Use ternary relationship (bottom fig)

Turn it into a one-to-many relationship



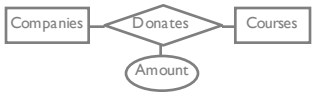
Entity or Relationship?

OK if company donates to courses individually

What if company donates to school for all data-related courses?

Redundancy of *amount*, need to remember to update every one

Misleading implies *amount* tied to individual courses



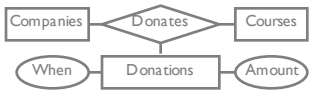
Company	Course	Amount
Amazon	4111	2000
Amazon	4112	2000
Amazon	5111	2000

These amounts are logically the same (redundant)!

Entity or Relationship?

If company donates once to school for all courses data related courses.

Refactor amount into an entity

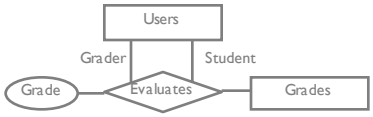


Company	Course	Donation
Amazon	4111	1
Amazon	4112	1
Amazon	5111	1

Donation	When	Amount
1	Today	2000

Binary or Ternary Relationship?

What if grades have at most one grader?

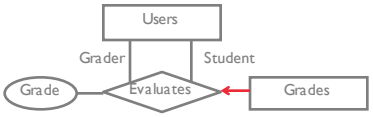


Binary or Ternary Relationship?

What if grades have at most one grader?

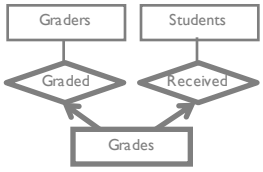
Only one student can have a grade!

Actually two separate relationships



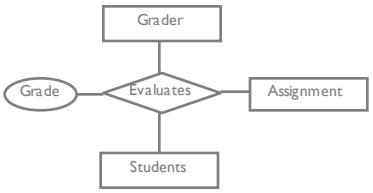
Binary or Ternary Relationship?

Binary relationships allows additional constaints



Binary or Ternary Relationship?

Sometimes have true ternary relationship that is defined by all three entities.



Summary

Conceptual design follows *requirements analysis*

ER model helpful for conceptual design

- constraints are expressive
- matches how we often think about applications

Core constructs

- entity, relationship, attribute
- weak entities, ISA, aggregation

Many variations beyond today's discussion

Summary

ER design is subjective based on usage+needs

- Today we saw multiple ways to model same idea

ER design is not complete/perfect

- Developed in an enterprise-oriented world
- Doesn't capture semantics (what does "instructor" *mean?)*
- Doesn't capture e.g., processes/state machines
- How to combine multiple ER models automatically?
- Limitation of imagination
- Open problems!

ER design is a useful way of thought

Summary

Requirements
- what are you going to build?

Conceptual Database Design
- pen-and-pencil description

(Today) ER Modeling

Logical Design
- formal database schema

Schema Refinement:
- fix potential problems, normalization

Physical Database Design
- use sample of queries to optimize for speed/storage

App/Security Design
- prevent security problems

Next Time

Relational Model – de-facto DBMS standard

Set up for ER diagrams → Relational models