Lecture 15: Midpoint

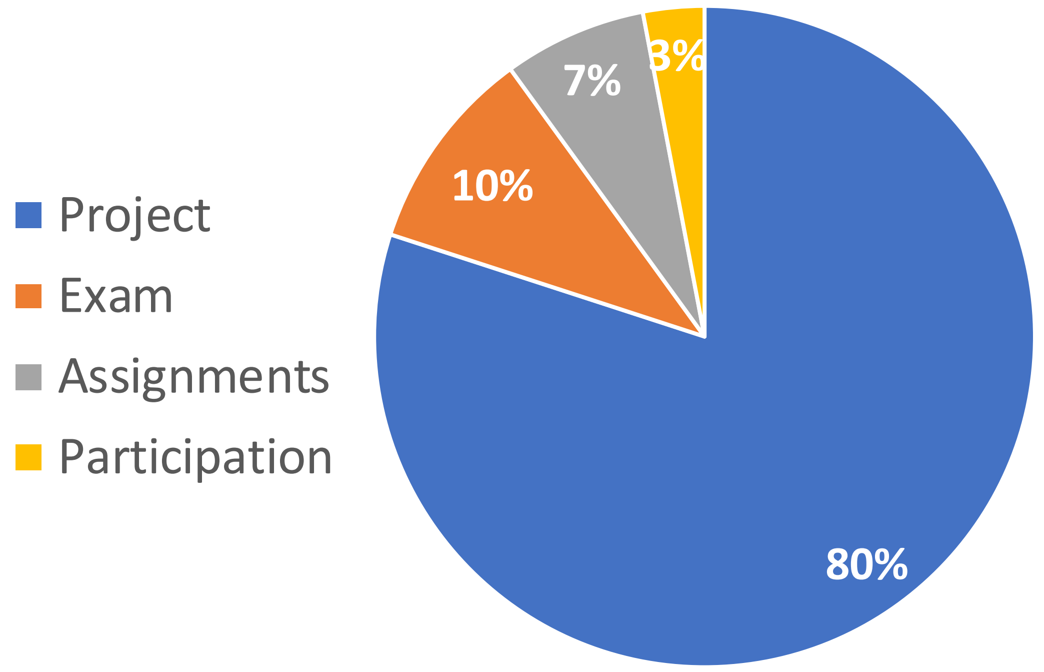
CS 5150, Spring 2022

Midpoint snapshot

- After lecture, grade summaries will be posted to CMS
  - As usual, do not try to interpret as a "percentage grade"
  - Rubrics in assignment descriptions (also summarized on Canvas)

- Assessments so far:
  - Polling participation
  - Assignments 1-3 + peer evals submission
  - Peer evals feedback
  - Project reports 1 & 2

Grading breakdown



(subject to change by up to 5%)

Project components:

- Reports
- Presentations
- Code & process quality
- Meeting etiquette / client feedback
- Peer evaluations

Rubric interpretations

- Participation
  - 3: On track for full credit
  - 2: Trending below full credit
  - 1: Below expectations
  - 0: No record of any participation

- Peer evaluation submission
  - 2: Followed directions
  - 1: Did not follow directions; manual cleanup required
    - Title lines
    - Numbers for comments
    - Excel instead of CSV
  - 0: No submission

- Tip: self-verify your submissions

Rubric interpretations: Peer feedback

- Peer feedback can *shift* your project grade
  - Relative, not absolute
  - By definition, most students earn a 3
    - Giving everyone maximum scores gives everyone a "3"
- Still have three more sessions over which to improve contributions

- 4: Peers have recognized your standout contributions
  - May receive minor grade boost
- 3: Meeting team expectations
  - Will inherit overall project grade
- 2: Peers have identified room for improvement
  - May receive minor grade penalty
- 1: Peers have some major concerns
  - If unaddressed, penalty depends on specific concerns

Retrospectives

- How to improve based on team feedback? Share it!
  - Hold a "retrospective" or "post-mortem" after major deliverables
    - Reports, presentations
- Need structure
  - Team improvement won't just happen on its own
  - Appoint a facilitator, have an agenda

- General tips
  - Don't make things personal, don't take things personally
    - Focus on specific actions/non-actions and their consequences
  - Focus on concrete steps for improvement
    - Don't dwell on blame
  - Be open, honest, and human
    - But stay calm, avoid getting defensive
  - Embrace sticky notes, timers

Retrospective outline

1. How are you feeling about the project today?
2. Individual reflection
   1. What went well? / What team accomplishment are you most proud of?
   2. What did not go well? / Where did the team fall short of its potential?
   3. What would you like to see improved?
3. Share and organize
4. Brainstorm: How could we improve?
5. Decide on actions

- Everyone participates
- Facilitator times reflection, brainstorming; calls on each member in turn

Poll: What is an appropriate way to address an individual's shortcomings in a retrospective?

A. "Alice, when you arrive late to meetings, the team loses productive time waiting for you, or else we risk rework because you missed important information. It put our last milestone at risk, and it lowers my impression of your contributions. Do you see where I'm coming from?"

B. "Bob, you're simply bad at writing Java. You clearly can't learn it fast enough, so just leave the coding to us."

C. "Chris, your UI designs are great. It'd be nice if you came to our planning meetings, though. But we appreciate your proofreading too."

D. Do nothing and hope the problem solves itself.

PollEv.com/cs5150

Web application frameworks

Django

- Architectural styles
  - Client-server separation
  - Three-tier architecture
  - Model-View-Template
    - "View" is analogous to "Controller"
    - "Template" is analogous to "View"

- Model
  - Object-Relational Mapper (ORM)
- View (aka "Template")
  - Templates
- Controller (aka "View")
  - Routes configuration
  - "View functions" that interpret a request, perform business logic on model data, and construct a response

Objects vs. Relational databases

Objects

- Encapsulate data in fields, expose behavior via methods
- Form a graph by referencing other objects
- Support inheritance and polymorphism

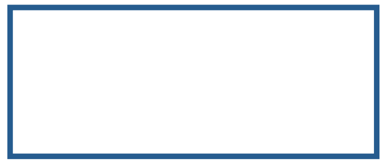
Relational databases

- Store data in rows of tables with fixed-type columns

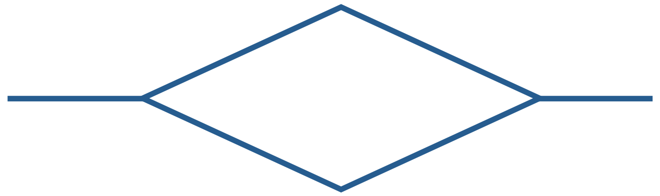
"Object-relational impedance mismatch"

Entity-relation model

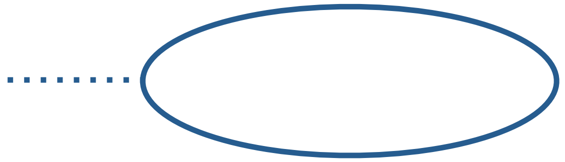
- Can be used for requirements modeling
- Can be used for database design



An entity (noun)

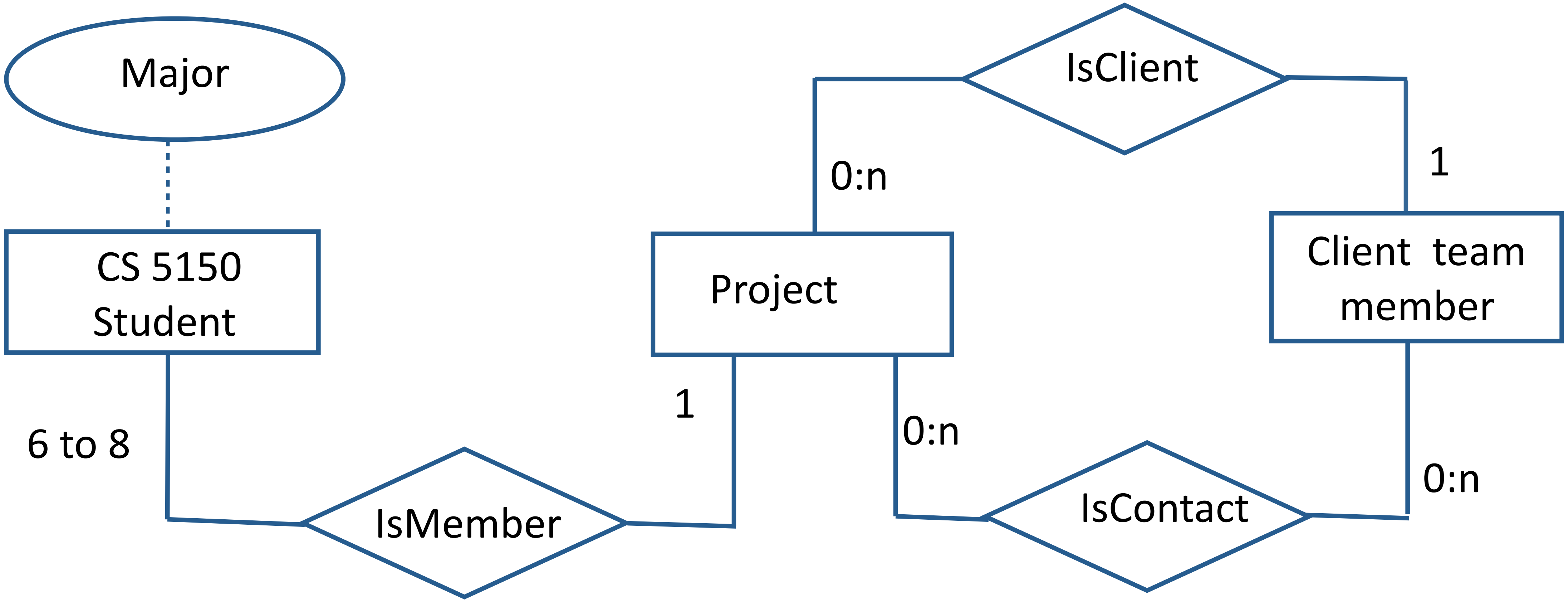


A relation between entities (verb)



An entity or relation attribute

Example


Major
CS 5150 Student
6 to 8
IsMember
1
Project
0:n
IsClient
1
Client team member
0:n
IsContact
0:n

Example representations

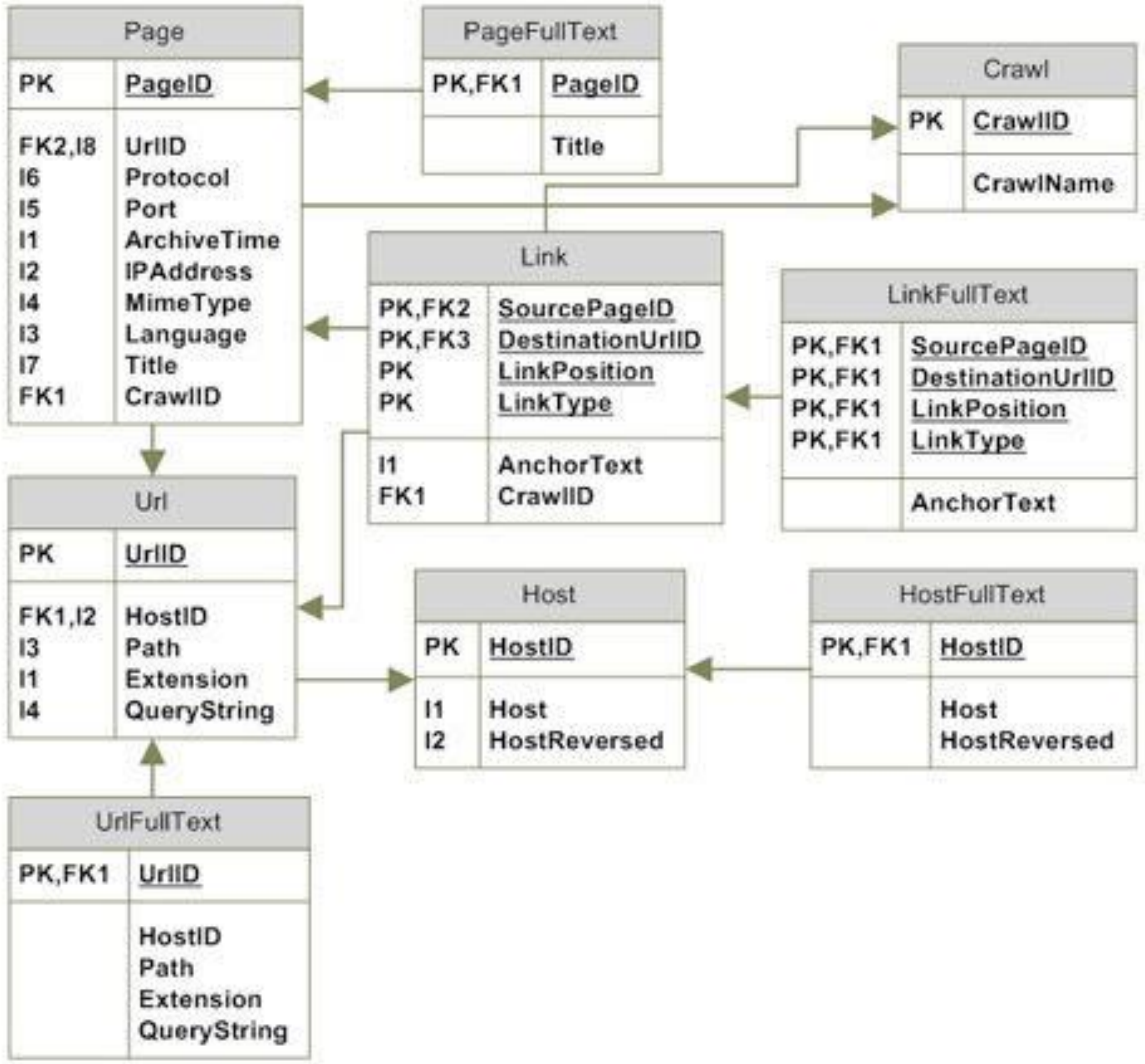
Objects

```
class Student {
  String major;
  Project project;
}

class Project {
  Stakeholder client;
  List<Stakeholder> contacts;
}

class Stakeholder {}
```

Database tables

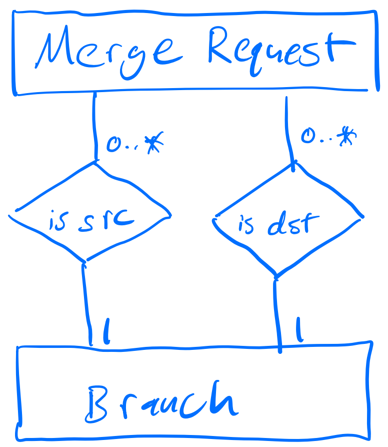
```
TABLE Students (
  id, major, projectId);

TABLE Projects (
  id, clientId);

TABLE Stakeholders (id);


TABLE Contacts (
  projectId, contactId);
```

Example: Web craw data

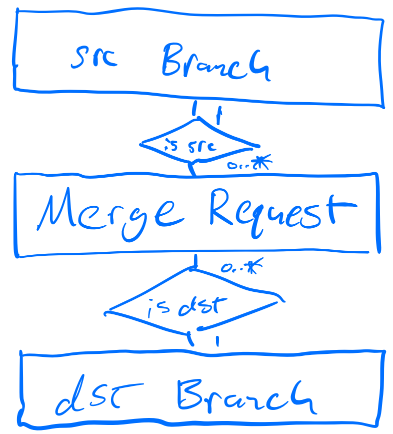


Poll: PollEv.com/cs5150

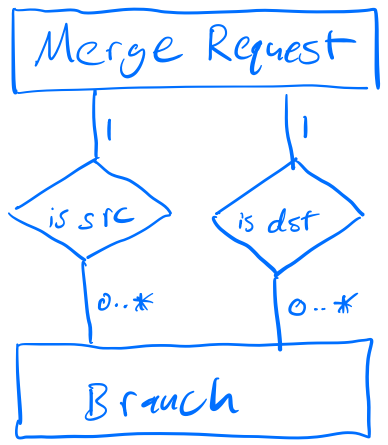
A



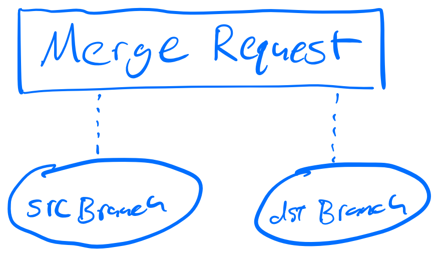
B



C



D



Demo: database exploration

Django ORM

```python
from django.db import models

class Person(models.Model):
  first_name =
      models.CharField(max_length=30)
  last_name =
      models.CharField(max_length=30)
```

```sql
CREATE TABLE myapp_person (
  "id" serial NOT NULL PRIMARY KEY,
  "first_name" varchar(30) NOT NULL,
  "last_name" varchar(30) NOT NULL
);
```

CRUD

	SQL	HTTP
Create	INSERT	PUT
Read	SELECT	GET
Update	UPDATE	PUT
Delete	DELETE	DELETE

Using introspection, many frameworks can dynamically generate graphical interfaces for managing application-specific persistent data

Django CRUD

- Create: create(), save()
- Read: get()
- Update: save(), add()
- Delete: delete()

```
from blog.models import Entry,
Author


entry = Entry.objects.get(pk=1)

entry.title = "Hello"

entry.save()


joe = Author.objects.create(
    name="Joe")

entry.authors.add(joe)
```

Django "views" (Controller)

- Routes (url.py)

```
from django.urls import path

from . import views

urlpatterns = [
    path('articles/2003/', views.special_case_2003),
    path('articles/<int:year>/', views.year_archive),
    path('articles/<int:year>/<int:month>/', views.month_archive),
    path('articles/<int:year>/<int:month>/<slug:slug>/',
      views.article_detail),
]
```

Django "views" (Controller)

```
def index(request):
    latest_question_list = Question.objects.order_by('-
pub_date')[:5]
    template = loader.get_template('polls/index.html')
    context = {
        'latest_question_list': latest_question_list,
    }
    return HttpResponse(template.render(context,
request))
```

Django templates

```
{% if latest_question_list %}
    <ul>
    {% for question in latest_question_list %}
        <li><a href="/polls/{{ question.id }}/">{{
question.question_text }}</a></li>
    {% endfor %}
    </ul>
{% else %}
    <p>No polls are available.</p>
{% endif %}
```

Logging

- Advantages over printing
    - Flexibility in output destination
    - Filtering (by severity, source component)
    - Automatic metadata (timestamp, line number)
    - Structured formats
    - Efficiency
- For global configuration, use Singleton pattern
    - `import logging`
    - `logger = logging.getLogger(__name__)`