Persistence

- Data must be preserved, even when application not running.
- Usually, data is stored in a relational database.
- Object oriented environment such as Java or ruby: we wish to persist and object, so that it outlives the process.
- Problem: paradigm mismatch between OO and relational databases.

Paradigm Mismatch Examples

- class Student
  - String id
  - String name
  - Address address
  - Professor advisor
  - Set courses
- class Course
  - String title
  - Set students
  - Professor professor
- class Professor
  - String id
  - String name
  - Set courses

Paradigm Mismatch

- Granularity
  - How to map address object to database table?

```
create table students (
  id integer not null primary key,
  advisor_id integer foreign key references professor(id),
  name varchar(255)
) ...
```

Paradigm Mismatch: subtypes

- In OO, relationships are expressed as references.
- In RDBMS, relationships expressed as foreign keys.
- Object references are directional. They are pointers. If you need to navigate in both directions, you must define the relationship twice.
- FK associates are not directional. You can create arbitrary relationships with joins.
Problems with data navigation

• In Java, you can access data like `student.course[i].professor` (or something similar).
• In SQL, you do not want to navigate from object to object like that. It is better to minimize requests to the database. (The dreaded n+1 selects problem)
• If you just want a professor’s data:
  `select * from professors where id=123;`
• But if you want related information, you would use a join:
  `select * from professors p, courses c
   where p.id=c.instructor_id;`

References

• This site has a good article, and links to other sites:
  [http://www.agiledata.org/essays/mappingObjects.html](http://www.agiledata.org/essays/mappingObjects.html)