The active record design pattern

Chris Mitchell
Email: chrism@lclark.edu

But then comes the database…

Tedium with SQL and tuples.

```
con = sqlite3.connect('gradebook.db')
cursor = con.execute("SELECT * FROM student")
for row in cursor:
    print "{0} {1}".format(row[1], row[2])
```

Tedium with SQL and tuples.

```
con = sqlite3.connect('gradebook.db')
cursor = con.execute("SELECT * FROM student")
for row in cursor:
    print "{0} {1}".format(row[1], row[2])
```

Fun with objects!

```
students = Students.all()
for student in students:
    print student.full_name()
```

The active record pattern at work.

Fun with objects!

```
students = Students.all()
for student in students:
    print student.full_name()
```

Active record…

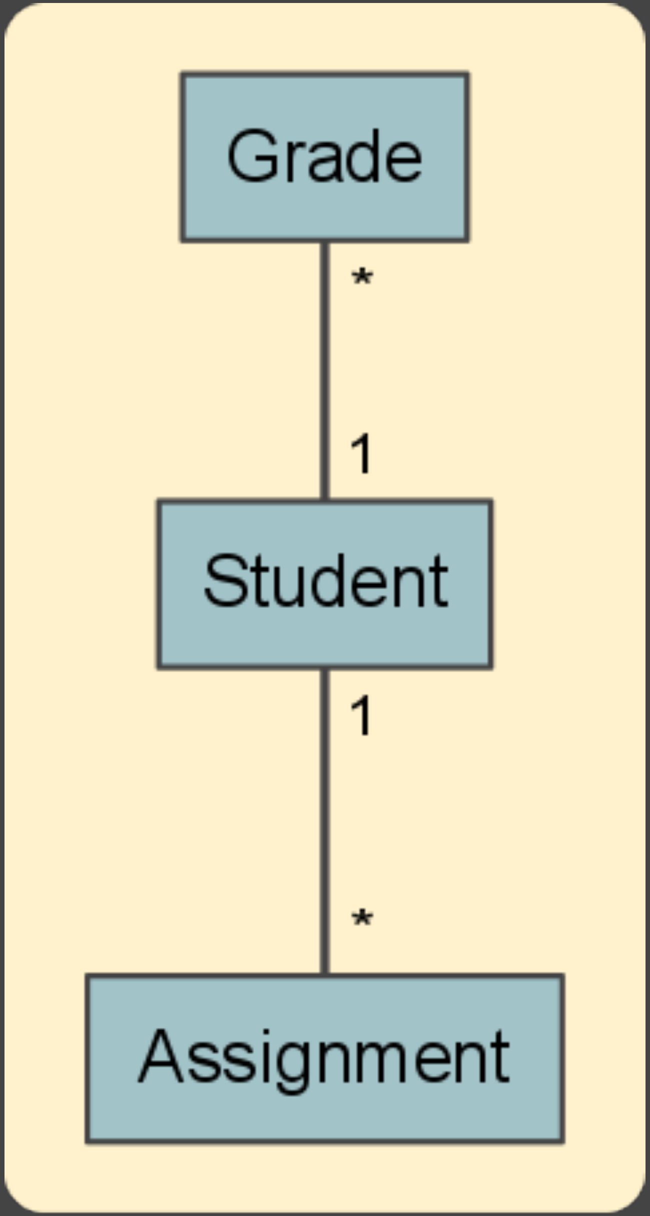
- is a design pattern
- wraps access to table rows
- instances represent rows
- adds business logic

For your reference:

```
students = Students.all()
for student in students:
    print student.full_name()
```

Ways it can be used:

- Student.all()
- Student.get(3)
- student.pk
- student.first_name = "Reginald"
- student.save()
- student.full_name()
- student.get_grades()
- grade.get_assignment()



How it can be implemented.

- One class per table
- Hand-coded parameterized SQL
- Class methods for retrieving/creating new rows
- Instance methods for business logic
- Strike a balance between simplicity and DRY

Active record is not a hammer AKA not everything is a nail.

- AR encourages coupling
- Some queries not easily expressible

But active record is still great for:

- CRUD — Create Update Delete
- That one guy on your team who still doesn’t know SQL.

In short, active record can help you make **database access** more congruent with **object orientation**.

http://www.lclark.edu/~chrism/talks/active-record/
has a working example.

I am: Chris Mitchell
My email address is: chrism@lclark.edu