SQLAlchemy

Robert M. Dondero, Ph.D
Princeton University

Objectives

- The lecture will:
  - Provide *a taste of* the **SQLAlchemy** object-relational mapper
  - Give you enough information about **SQLAlchemy** to:
    - Help you decide if you want to use it in your project
    - Help you get started with it

Motivation

- **Problem**:
  - Relational DBs create an ***impedence mismatch***
    - Awkward to map the relational data model (tables, rows, fields) to the OOP data model (objects, object references, object composition, class inheritance)

Cursors

- **Solution 1**: Cursors
  - As we've seen…
    - Cursor = array + indication of current element
  - DB driver maps each DB **table** to an array
    - Each element represents a row
    - Cursor keeps track of current row
  - DB driver maps each DB **row** to an array
    - Each element represents a field

ORMs

- **Solution 2**: ***Object-relational mapper (ORM)***
  - ORM maps each DB **table** to a **class**
    - E.g., books table => class Book
  - ORM maps each DB **row** to an **object**
    - E.g., row of books table => object of class Book
  - ORM maps each DB **field** to an object **field**
    - E.g. isbn field of some row of books table => isbn field of some object of class Book
  - OO pgm fetches/stores data by sending messages to (calling methods of) objects, not by executing SQL statements

Popular ORMs

- Some popular ORMs:
  - See https://en.wikipedia.org/wiki/List_of_object-relational_mapping_software
  - For Python the most popular is…

SQLAlchemy

- ***SQLAlchemy***
  - **Who**: Michael Bayer
  - **When**: 2006
  - **Why**: ORM for Python

Installing SQLAlchemy

- Linux, Mac, and MS Windows
  - At a bash shell or Command prompt:
    - Activate your cos333 virtual environment
    - `python -m pip install SQLAlchemy`

SQLAlchemy Pgmming in Python

- See **SQLAlchemy/database.py**
  - Informs SQLAlchemy of database schema
    - Names of tables
    - Names & data types of fields
    - Note:
      - Unusual use of class-level (static) fields
      - For each table, must specify which fields comprise the table's primary key

SQLAlchemy Pgmming in Python

- See **SQLAlchemy/create.py**
  - To create DB
    - Create engine
    - Create session
    - Create all tables
    - Create books, add to session, commit session
    - Same for authors, customers, zipcodes, orders

SQLAlchemy Pgmming in Python

- See **<u>SQLAlchemy/display.py</u>**
  - To display DB
    - Create engine
    - Create session
    - Send `query()` message to session
      - Specify table
    - Send `all()` message to result
      - Alternative: send `one()` message to result

SQLAlchemy Pgmming in Python

- See **SQLAlchemy/authorsearch.py**
  - To query DB
    - Create engine
    - Create session
    - Send `query()` message to session
    - Send `filter()` message to result
    - …
    - Send `all()` or `one()` message to result

SQLAlchemy Pgmming in Python

- See **SQLAlchemy/order.py**
  - To update DB
    - Same as query, and then…
    - Update object fields
      - SQLAlchemy marks changed objects as “dirty”
    - Send `commit()` message to session
      - SQLAlchemy writes dirty objects to DB

SQLAlchemy Pgmming in Python

- See **SQLAlchemy/purchase.py**
  - SQLAlchemy supports transactions
    - Can send `commit()` or `rollback()` message to session

SQLAlchemy Pgmming in Python

- See **SQLAlchemy/recovery.py**
  - Transactions work!

SQLAlchemy Assessment

- SQLAlchemy assessment
  - (pro) Eliminates impedence mismatch
  - (pro) Insulates programmer from SQL
    - Good if you don't know SQL, or don't want to deal with SQL
  - (con) Insulates programmer from SQL!
    - Bad if your intention is to learn SQL

- You **may not** use SQLAlchemy for assignments
- You **maybe should** use SQLAlchemy for your project

Summary

- The lecture has:
  - – Provided *a taste of* the **SQLAlchemy** object-relational mapper
  - – Given you enough information about **SQLAlchemy** to:
    - Help you decide if you want to use it in your project
    - Help you get started with it