L16
Application Programming

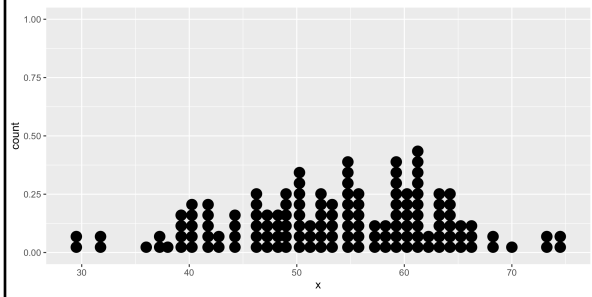
Administrivia

Thank you for not discussing midterm

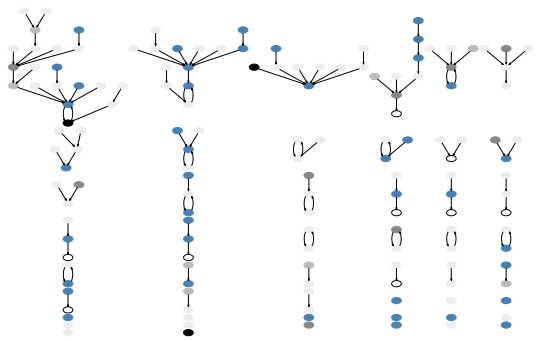
Midterm stats, feedback, problems?
- solutions out later this week

Project Part 3 out
- web application
- demos week ~11/9

HW3 out Monday



Median 54.5
Mean 53.5
Stddev 9.53



SQL != Programming Language

Designed for data access/manipulation
Not Turing complete: missing recursion/iteration
Can't perform "business logic"

Many Database API options

Embedded SQL: Mix SQL and another language

Low-level library with core database calls (DBAPI)

Object-relational mapping (ORM)

Embedded SQL

Previously popular approach; Not currently in fashion
Extend host language (C, C++, Fortran, Pascal) with SQL syntax
Compiled into program that interacts with DBMS directly

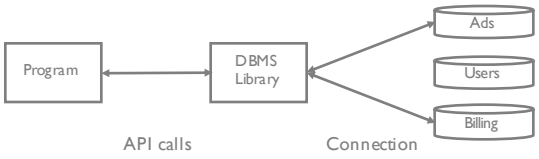
Oracle Pro*C (still supported)

```
int a;
/* ... */
EXEC SQL SELECT salary INTO :a
         FROM Employee
         WHERE SSN=876543210;
/* ... */
printf("The salary is %d\n", a);
/* ... */
```

Embedded SQL

What does a library need to do?

Interface to multiple DBMS engines
Map objects between host language and DBMS
Manage query results



Engines

Common interface to all databases:
hides DBMS differences

```
from sqlalchemy import create_engine
db1 = create_engine(
   "postgresql://localhost:5432/testdb"
)

db2 = create_engine("sqlite:///testdb.db")
// note: sqllite has no host name (sqlite:///)
```

http://docs.sqlalchemy.org/en/rel_1_0/core/engines.html

Connections

Communication channel: send query and receive results
Relatively expensive; often cached for future use
Defines scope of a transaction (later)

```
conn1 = db1.connect()
conn2 = db2.connect()
```

Should close connections when done! Otherwise resource leak.

Query Execution

```
conn1.execute("update table test set a = 1")
conn1.execute("update table test set s = 'wu'")
```

Query Execution

```
foo = conn1.execute("select * from big_table")
```

Challenges

- What is the return type of `execute()`?
- How to pass data between DBMS and host language?
- Can we pass *code* between the two?

Query Execution

How to pass values into a query?

```
Users(id int serial, name text)

name = "eugene"

conn1.execute("""
   INSERT INTO users(name)
   VALUES(<what to put here??>)""")
```

Query Execution

How to pass values into a query?

```
Users(id int serial, name text)

name = "eugene"

conn1.execute ("""
   INSERT INTO users(name)
   VALUES('{name}')""".format(name=name))
```

Why is this a *really* bad idea?

Detour: SQL Injections

http://w4111db.eastus.cloudapp.azure.com:8112/

code on github:
syllabus/src/injection/


```
@app.route('/', methods=["POST", "GET"])
def index():
  if request.method == "POST":
    name = request.form['name']
    q = "INSERT INTO bad_table(name) VALUES('%s');" % name
    print q
    g.conn.execute(q)
```

Detour: SQL Injections

If we submit:
'); DELETE FROM bad_table; --

Query is
INSERT INTO bad_table(name) VALUES('');
DELETE FROM bad_table; -- ');

```
@app.route('/', methods=["POST", "GET"])
def index():
  if request.method == "POST":
    name = request.form['name']
    q = "INSERT INTO bad_table(name) VALUES('%s');" % name
    print q
    g.conn.execute(q)
```

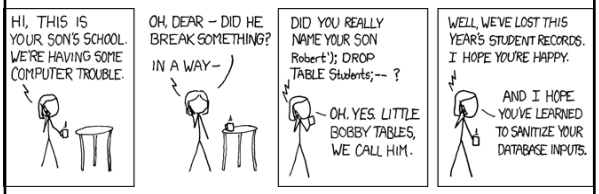
Detour: SQL Injections

Safe implementation

Pass form values as arguments to the execute() function
Library sanitizes inputs automatically (and correctly!)

```
@app.route('/safe/', methods=["POST", "GET"])
def safe_index():
  if request.method == "POST":
    name = request.form['name']
    q = "INSERT INTO bad_table(name) VALUES(%s);"
    print q
    g.conn.execute(q, (name,))
```

Detour: SQL Injections



Project: You'll need to protect against simple SQL injections

Query Execution

Pass *sanitized* values to the database

```
args = ('Dr Seuss', '40')
conn1.execute(
    "INSERT INTO users(name, age) VALUES(%s, %s)",
    args)
```

Pass in a tuple of query arguments
DBAPI library will *properly escape* input values
Most libraries support this
Never construct raw SQL strings

Placeholders

Not standardized: Vary between languages and databases

Postgres in Python: Use %s
Postgres in Go: Use ?
SQLite in Python: Use ?

Impedance Mismatch

Electronics: Maximize power transfer:
Match output impedance of source to the input impedance of load

Relational Impedance Mismatch

Mismatch between the relational database model and the programming model, particularly objects

Object (programming) != Row (database)

(Type) Impedance Mismatch

SQL defines mappings between several languages
Most libraries can deal with common types

SQL types	C types	Python types
CHAR(20)	char[20]	str
INTEGER	int	int
SMALLINT	short	int
REAL	float	float

What about complex objects { x: 'l', y: 'hello' }?

(Object) Impedance Mismatch

Programming languages have objects/structs
Setting an attribute in User should save it

```
user.name  = "Dr Seuss"
user.job  = "writer"
```

```
class User { … }
class Employee extends User { … }
class Salaries {
    Employee worker;
    …
}
```

Object Relational Mappings designed to address this

Object-Relational Mappers

Use objects in your program; read/write relations
Widely used; avoids writing conversion code

Can cause inefficient queries
Tricky when upgrading apps
Complex queries: may need raw SQL

```
class User(Base):
  __tablename__  = 'users'

  id = Column(Integer,  primary_key=True)
  name = Column(String)
  fullname = Column(String)
  password = Column(String)
```

```
class User(Base):
  __tablename__  = 'users'

  id = Column(Integer,  primary_key=True)
  name = Column(String)
  fullname = Column(String)
  password = Column(String)


CREATE TABLE users(
  id INT PRIMARY KEY,
  name TEXT,
  fullname TEXT,
  password TEXT
);
```

```
>>> ed_user = User(
  name='ed',  fullname='Ed Jones',
  password='edspassword')
>>> ed_user.name
'ed'
>>> ed_user.password
'edspassword'
>>> session.add(ed_user)
```

```
session.query(User).filter(User.name.in_(
  ['Edwardo',  'fakeuser']
).all()
```

```
SELECT  * FROM users
WHERE name IN ('Edwardo',  'fakeuser')
```

ORM Relationship Challenges

Recall Sailors Reserve Boats

Should the Sailors object have a "reservations" object?
Should Boats object have "reservations"? Both?

```
for reservation in sailorEvan.reservations:
  print "reservation on: " reservation.day
  print "reserved boat: " reservation.boat.name
```

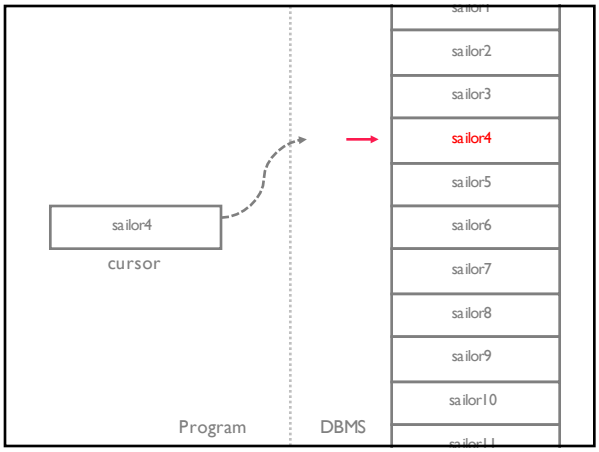
(results) Impedance Mismatch

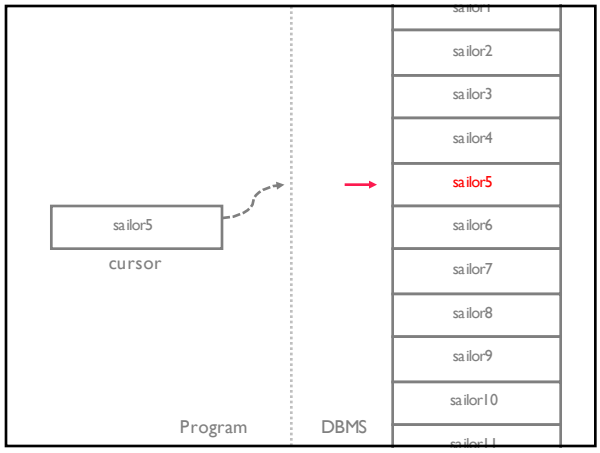
SQL relations and results are sets of records
What is the type of `table`?

```
table = execute("SELECT * FROM big_table")
```

Cursor over the Result Set

- similar to an iterator interface
- Note: relations are unordered!
  - Cursors have no ordering guarantees
  - Use ORDER BY to ensure an ordering





Cursors

Similar to an iterator (next() calls)

```
cursor = execute("SELECT * FROM bigtable")
```

Cursor attributes/methods (logical)

```
rowcount
keys()
previous()
next()
get(idx)
```

Cursors

Similar to an iterator (next() calls)

```
cursor = execute("SELECT * FROM bigtable")
cursor.rowcount()  # 1000000
cursor.fetchone()  # (0, 'foo', …)
for row in cursor: # iterate over the rest
   print row
```

Actual Cursor methods vary depending on implementation

(functions) Impedance Mismatch

What about functions?

```
def add_one(val):
   return val + 1

conn1.execute("SELECT add_one(1)")
```

Would need to embed a language runtime into DBMS
- Many DBMSes support runtimes e.g., python
- Can register User Defined Functions (UDFs)

(constraints) Impedance Mismatch

DB-style constraints often as conditionals or exceptions
Constraints often duplicated throughout program

JS
```
age = get_age_input();
if (age > 100 or age < 18)
    show_error("age should be 18 – 100");
```

DBMS
```
CREATE TABLE Users (
   …
   age int CHECK(age >= 18 and age <= 100)
   …
)
```

(constraints) Impedance Mismatch

Some ORMs try to have one place to define constraints

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

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

Some Useful Names

DBMS vendors provide libraries for most libraries

Two heavyweights in enterprise world

ODBC Open DataBase Connectivity
- Microsoft defined for Windows libraries

JDBC Java DataBase Connectivity
- Sun developed as set of Java interfaces
- java.sql.*
- javax.sql.* (recommended)

Modern Database APIs

Linq, Scalding, SparkSQL

DBMS executor in same language (C#, Scala) as app code
- what happens to language impedance?
- what happens to exception handling?
- what happens to host language functions?

```
val lines = spark.textFile("logfile.log")
val errors = lines.filter(_ startswith "Error")
val msgs = errors.map(_.split("\t")(2))

msgs.filter(_ contains "foo").count()
```

What to Understand

Impedance mismatch
- Examples, and possible solutions

SQL injection and how to protect

The different uses of a DBAPI

Why Embedded SQL is no good

What good are cursors?