CS W4111.001
Introduction to Databases
Spring 2017

Computer Science Department
Columbia University

Application Programming
SQL != Programming Language

Not a general purpose programming language
Tailored for data access/manipulation
Easy to optimize and parallelize
Can't perform "business logic"

Options
1. Extend existing languages to understand SQL natively
2. Provide an API between programming languages and DBMS

Many Database API Options

Fully embed into language (embedded SQL)

Low-level library with core database calls (DBAPI)

Object-Relational Mapping (ORM)

Ruby on Rails, Django, Hibernate, SQLAlchemy, …
defines database-backed classes
magically maps between database rows & objects
“Magic” is a double-edged sword; we will not use in our class
Embedded SQL

Extend host language (e.g., Java) with SQL syntax
   e.g., EXEC SQL sql-query
   goes through a preprocessor

Compiled into program that interacts with DBMS directly

Embedded SQL

Java + embedded SQL
  ↓
  Preprocessor
  ↓
  Java + DB library calls
  ↓
  Java Compiler
  ↓
  Executable
  ↑
  DBMS library
  ↓
  DBMS

    ... if (user == 'admin'){
    ...     EXEC SQL select * ...
    } else {
    ...     ...
    }
SQL Programming Models

Embedded SQL
- Host language (e.g., C, Java) program with special SQL directives goes through a preprocessor.
- It is then compiled into a program that interacts with the DBMS directly.

API
- Vendor specific or standardized (ODBC/JDBC).
- Write a program using classes implemented by the DBMS vendor, which implement a standard set of interfaces.
- Pass SQL statements as arguments to functions.
- SQL statements are processed at run-time, and sent to DBMS via a driver provided by the DBMS.

What does a library need to do?

Single interface to possibly multiple DBMS engines
Connect to a database
Manage transactions
Map objects between host language and DBMS
Manage query results
Overview

Library Components

Impedance Mismatches
  1. Types
  2. Result sets

Engines

Abstraction for a database engine
  tries to hide DBMS language differences

driver://username:password@host:port/database

from sqlalchemy import create_engine

db1 = create_engine(
    "postgresql://localhost:5432/testdb"
)

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

Before running queries need to create a connection

• Tells DBMS to allocate resources for the connection
• Are relatively expensive to set up, so libraries often cache connections for future use
• Defines scope of a transaction

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

Should close connections when done!

Query Execution

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

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

Challenges

- What is the return type of `execute()`?
- Type impedance
- How to pass data between DBMS and host language?

(Type) Impedance Mismatch

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

<table>
<thead>
<tr>
<th>SQL types</th>
<th>C types</th>
<th>Python types</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR(20)</td>
<td>char[20]</td>
<td>str</td>
</tr>
<tr>
<td>INTEGER</td>
<td>int</td>
<td>int</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>short</td>
<td>int</td>
</tr>
<tr>
<td>REAL</td>
<td>float</td>
<td>float</td>
</tr>
</tbody>
</table>

What about complex objects `{ x:‘I’, y:‘hello’ }`
Query Execution

Pass *sanitized* values to the database

```python
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*

(Results) Impedance Mismatch

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

```python
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
Cursor similar to an iterator (next() calls)

cursor = execute("SELECT * FROM bigtable")

Cursor attributes/methods (logical)

rowcount
keys()
previous()
next()

Actual Cursor methods vary depending on implementation
Some Useful Names

DBMS vendors provide libraries for most languages

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.*

What to Understand

Impedance mismatch
Different uses of a DBAPI
Why Embedded SQL is no good
What good are cursors?

Will use **SQLAlchemy** with Python for Part 3 of Project 1, but **without ORM component**