

SQLAlchemy

# The Problems Came First

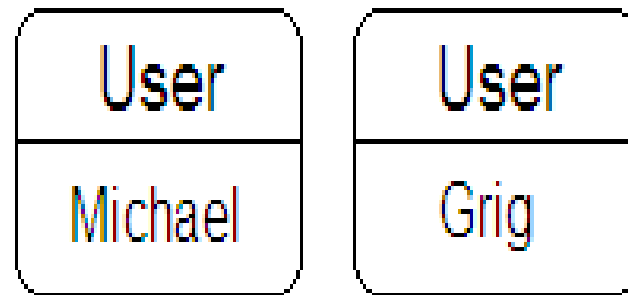
- Data Persistence
- Data size
- Data sharing

# Objects vs. Rows

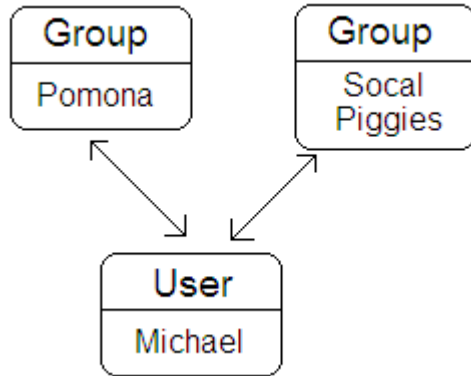
- Relational DB
  - Tables

<i>user_id</i>	<i>user_name</i>	<i>password</i>
1	Michael	Secret
2	<u>Grig</u>	Piggies
...	...	...

- Object DB
  - Serialized Objects



# Pointers vs. Relations, Round 1



Of which groups is Michael a member?

## Relational DB

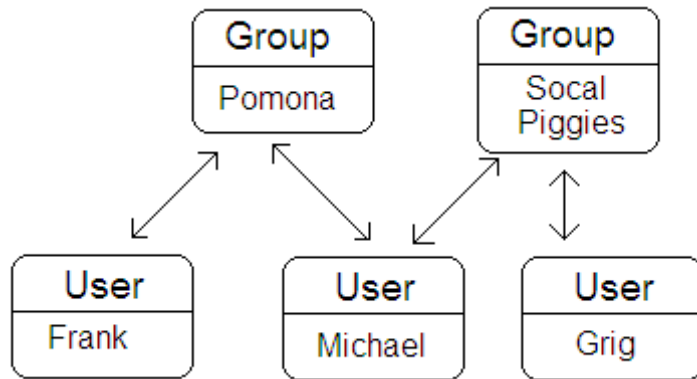
Use SQL:

```
select * from users_groups  
where user_id = 1
```

## Object DB

Easy, we have the list  
already: **Michael.groups**

# Pointers vs. Relations, Round 1



Who is in Michael's Network?

## Relational DB

Use SQL:

1. Issue Select on `user_groups`
2. Join result of 1 to `groups`
3. Join result of 2 to `user_groups`
4. Select from result of 3

## Object DB

```
1 users = []
2 for group in michael.groups:
3     for user in group.users:
4         users.append(user)
5
```

# Best Option?

## Relational DB

- Faster
  - Algorithmically
  - HD Access
- Data relations are easy

## Object DB

- Simple, Obvious
- Pythonic
  - Easy to add a persistence layer to old programs.
  - Practically no difference in how you think about problems

# SQLAlchemy

Enter SA

# SQLAlchemy: Overview

- Object Relational Mapper
- Best of Two Worlds
- Levels of Abstraction
  - SQL generation
  - Database connection
  - Data Mappers
  - Sessions

# SQLAlchemy: The Very Basics

- Tables
  - Example
- Queries
  - Example
- Data Mapping
  - Plain old python
  - Example
  - Sessions
  - Unit of Work

# SQLAlchemy: Advanced Uses

- Self-referential Mapping
- Mapping Inheritance Trees
  - Vertical
  - Horizontal
- Polymorphic Loading
- Mapping against arbitrary selects

