Intro to Computer Science

Previous

- Some UNIX
- SQL
  - select
  - join
  - alter

Next

- SQL review
- Python + MySQL

SQL

- Structured query language (SQL) is a way of extracting information from relational databases
  - query
  - define
  - modify
- Allows the user to specify what data they want, not necessarily how

Selection

Extract a subset of the data

UID	Name	Major
1	Alex	CS
2	Lisa	Bio
3	Shaun	CS

```
SELECT *
FROM students
```

UID	Name	Major
1	Alex	CS
2	Lisa	Bio
3	Shaun	CS

Selection

Extract a subset of the data

UID	Name	Major
1	Alex	CS
2	Lisa	Bio
3	Shaun	CS

```
SELECT *
FROM students
WHERE Major = 'CS'
```

UID	Name	Major
1	Alex	CS
3	Shaun	CS

Projection

Select all the data, but only show the UID and name

```
SELECT UID, Name
FROM students
```

UID	Name	Major
1	Alex	CS
2	Lisa	Bio
3	Shaun	CS

UID	Name
1	Alex
2	Lisa
3	Shaun

Projection

Select students whose first name is ‘Alex’, but only give me the name and major

```
SELECT Name, Major
FROM students
WHERE Name = ‘Alex’
```

UID	Name	Major
1	Alex	CS
2	Lisa	Bio
3	Shaun	CS

UID	Name	Major
1	Alex	CS

Name	Major
Alex	CS

In SQL

- SELECT [attributes]
  - SELECT uid, name, major
  - SELECT *
- FROM [relations]
  - FROM students
- WHERE [predicates]
  - WHERE uid > 2 AND major = 'CS'
- GROUP BY [attribute]
- ORDER

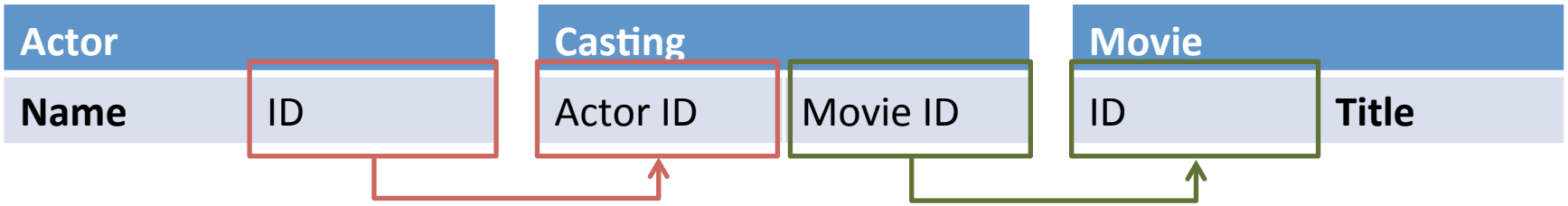
- Predicate gotchas
  - Not equal is '<>'
  - Special comparisons for NULL values
    - a NOT NULL
    - a IS NULL
  - String values go between single quotes (' '), not double quotes (" ")

Joins

Type	Names	
Cross join	• Cartesian product	Every tuple from R_1 combined with ever tuple from R_2
Inner join	• Natural join • **Equi-join**	Tuples in R_1 and R_2 with some matching attribute
Outer join	• **Left-outer join** • Right-outer join • Full-outer join	Tuples from one relation that may not have a matching attribute with the other

➢ We will mostly focus on equi-joins

Going to the movies




```
SELECT actor.name, movie.title FROM actor
JOIN casting ON actor.id = casting.actorid
JOIN movie ON casting.movieid = movie.id
```

Insert/Update

- Insert allows you to add data to the database

```
INSERT INTO table (a1, a2, …, an) VALUES (v1, v2, …, vn)
```

- Update allows you to alter data within the table

```
UPDATE table SET a1=v1, a2=v2 …, an=vn WHERE condition
```

Insert/Update/Delete

- Insert allows you to add data to the database

INSERT INTO table (a_1, a_2, …, a_n) VALUES (v_1, v_2, …, v_n)

```
INSERT INTO students (uid, name, major)
VALUES (4, 'Bob', 'Physics')
```

UID	Name	Major
1	Alex	CS
2	Lisa	Bio
3	Shaun	CS

UID	Name	Major
1	Alex	CS
2	Lisa	Bio
3	Shaun	CS
4	Bob	Physics

Insert/Update/Delete

- Update allows you to update data in the database

  `UPDATE table SET` $a_1=v_1$, $a_2=v_2$, …, $a_n=v_n$ `WHERE …`

```
UPDATE students SET Major = 'Bio' WHERE Name = 'Alex'
```

UID	Name	Major
1	Alex	CS
2	Lisa	Bio
3	Shaun	CS

UID	Name	Major
1	Alex	Bio
2	Lisa	Bio
3	Shaun	CS

Back to Python!

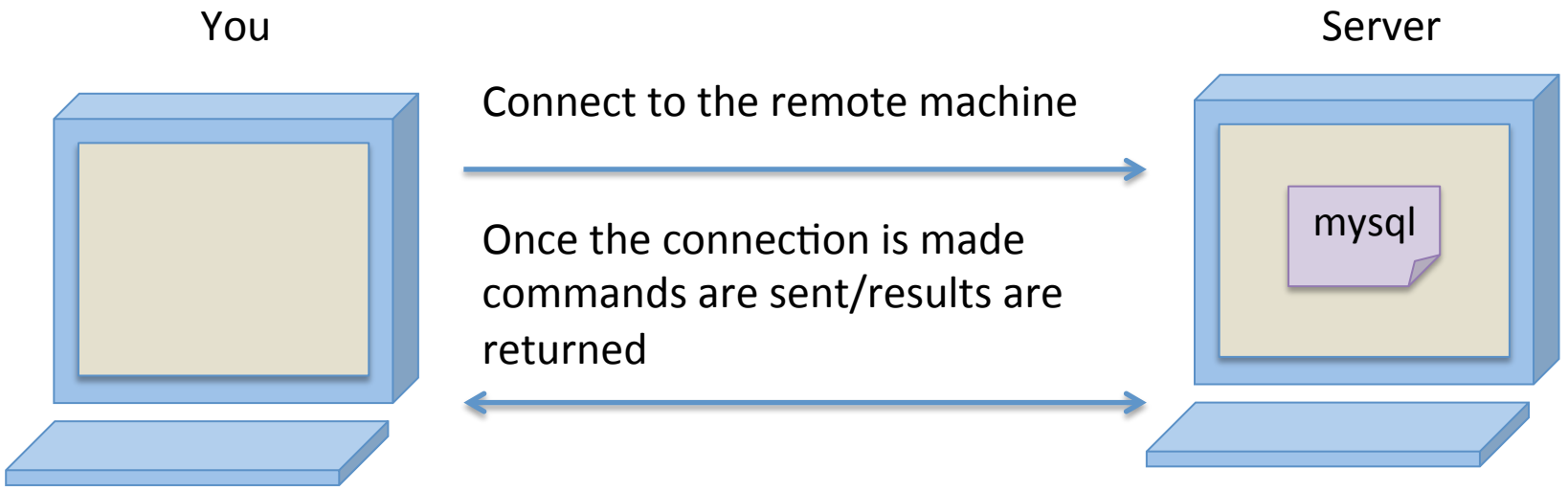
- It's much more convenient (and realistic) to integrate the power of a database into your program
- Several intermediary programs that facilitate this
  - Within Python
  - And even across languages
- We need something that supports remote procedures

A little background

- A *server* is a program that sits around waiting for instructions (*requests*)
- The programs we've written that wait for user input are essentially "servers"

Connection to MySQL

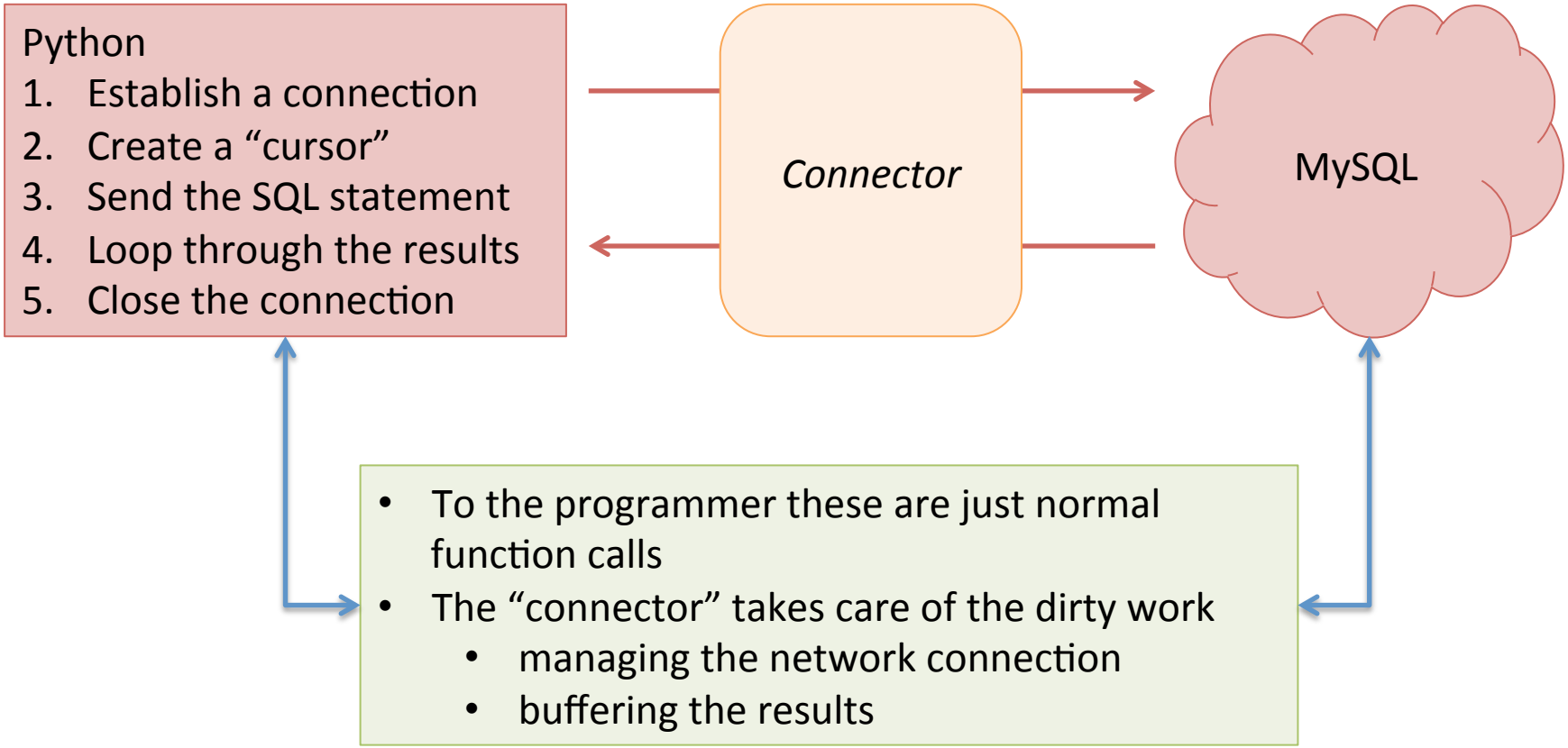
- Interacting with a server over the Internet is a lot like a phone conversation
  - Dial the persons phone number
  - Once the person picks up the connection is established
  - Commands are sent back and forth by talking to each other
- Generally never worry about the infrastructure (how it's accomplished)



Connecting to MySQL

- MySQL sits around and waits for SQL statements
- The format you see when operating in the terminal is just a printing of the data
  - Just like with our board: the data and the presentation are separate!
- It's the *data* that's sent to Python

The idea


Python
1. Establish a connection
2. Create a "cursor"
3. Send the SQL statement
4. Loop through the results
5. Close the connection
Connector
MySQL
• To the programmer these are just normal function calls
• The "connector" takes care of the dirty work
  • managing the network connection
  • buffering the results