Advanced Database Design Lecture 1

Gregory S. DeLozier, Ph.D.

gdelozie@kent.edu

Objectives

- SQL/relational database concepts
- Use of databases in web apps
- Object/relationship mappers (ORM)
- Relational database optimization
- Document store (NoSQL) databases
- Distributed database strategies
- New developments in database technology
- A bit of big data experience

Activities

- Review some historical high points
- Examine some database research topics
- Conduct some database experiments
- Do a database project
- Present some research results
- Impress your friends

- Enjoy yourself!

Not In Scope

- Remedial SQL
- General software development skills
- Detailed training in commercial products
- DBA preparation
- Fear of breaking things
- Expensive books

General Approach

- History and Context
- Problems to be addressed
- History of a specific product or technology
- Examination of the product or technology
- Implementation and playing around
- Making something with the technology
- Examination, homework, etc.

Grading

- Weekly homework. 20%
- Midterm 20%
- Database project 20%
- Final exam. Open notes, etc. 40%

Missing Class or Deadlines

- Missing class is not good. If you miss a lecture or a lab exercise, you will have to get that information and experience somewhere.
- On weekly assignments, there are no makeups. Some grading is automated, and I will discuss the assignments the following week. Get them done.
- On other assignments, except the final, if you are having trouble with a date, see me.

Class Conduct

- The usual rules about adult behavior apply.
- Kent is serious about academic honesty.
- Keep the laptop and phone distraction to a minimum when we're doing things together.
- I have no problem with snacks and drinks, and I will check on Kent's rules. This is a nice room, and was just remodeled, so keep it clean.
- We will have one or two breaks. If you need to leave between breaks, be discreet, please.

Extreme Conditions

- If something unusual is happening at the university, we might not have class if the university is closed.
- If something unusual happens to me, and I'm not here by 7:30, we won't have class.
- If either one of these happens, adjusted homework and lecture notes will be posted on the web within 24 hours. *You will still be responsible for getting assignments done.*

Skills you will need

- Programming basics
  - How to use an IDE, how to debug
  - Basic language skills
    - SQL
    - Python
    - Javascript
    - Java
- Know your way around a Linux command line
- Some idea of how the web works
- Some way to write scientific English

Stuff you need to get

- A reasonable computer
  - Windows, Mac, Linux
  - Chrome
  - Possibly some free software (Python, Mongo, etc.)
- A solid web connection
- Something to keep a lot of information safe
  - (i.e. a notebook, a document, etc.)
  - You will be storing passwords

Stuff you need to read

- Open source books as assigned
- Various system documentation
- On-line articles and industry commentary
- Research papers regarding database topics

- We are not using a conventional textbook. You can learn from the original sources.

Stuff you need to sign up for

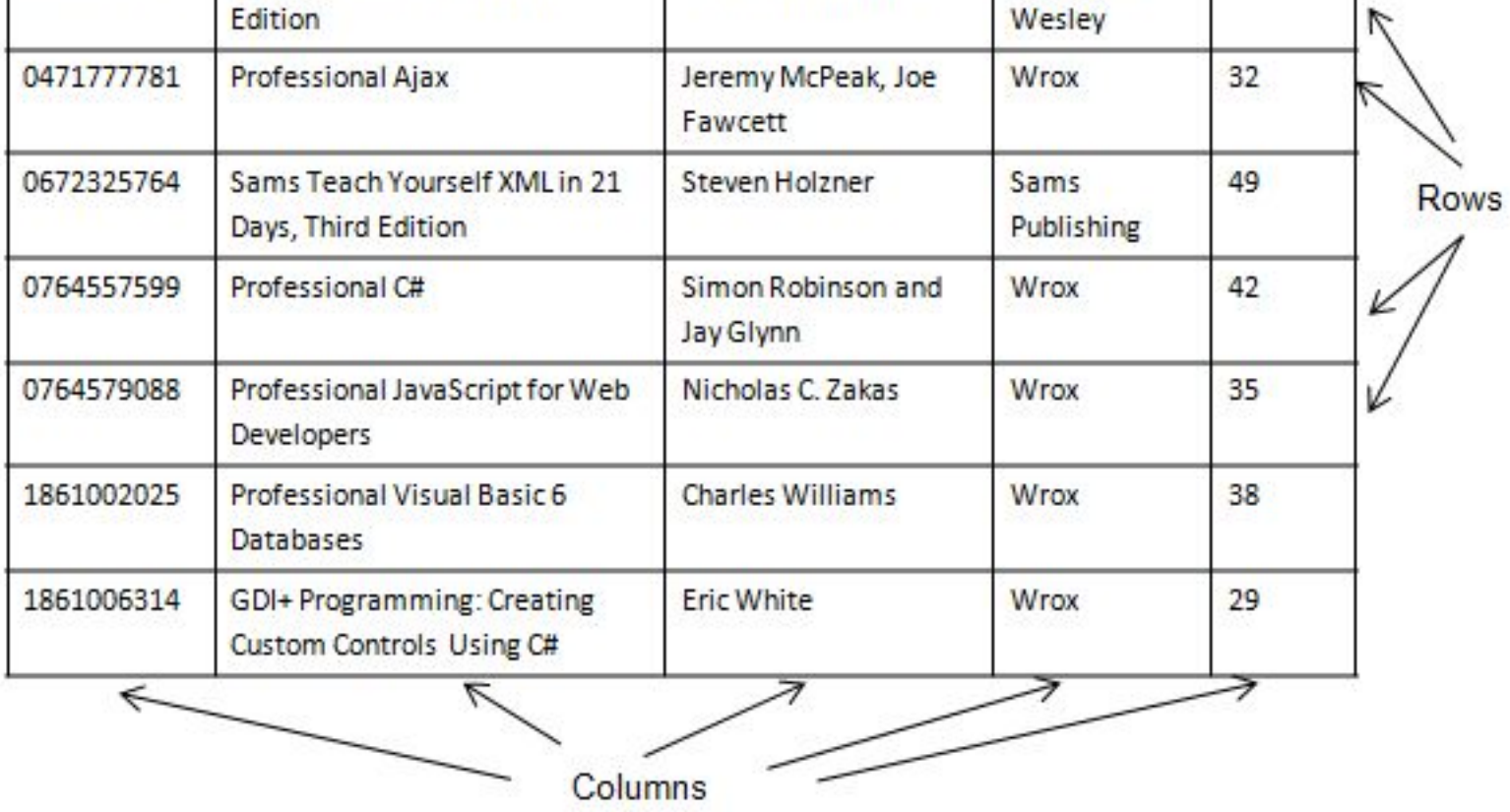
- PythonAnywhere
  - General workstation-in-the-cloud
  - Very easy to serve web apps
- Codio
  - Ubuntu/Linux workstations
  - Multiple computers per person
  - More like "real" servers
- GitHub
  - Store your code responsibly. It's a good habit

Demo Time!

A little bit of SQL

Tables

ISBN_NO	SHORT_DESC	AUTHOR	PUBLISHER	PRICE
0201703092	The Practical SQL, Fourth Edition	Judith S. Bowman	Addison Wesley	39
0471777781	Professional Ajax	Jeremy McPeak, Joe Fawcett	Wrox	32
0672325764	Sams Teach Yourself XML in 21 Days, Third Edition	Steven Holzner	Sams Publishing	49
0764557599	Professional C#	Simon Robinson and Jay Glynn	Wrox	42
0764579088	Professional JavaScript for Web Developers	Nicholas C. Zakas	Wrox	35
1861002025	Professional Visual Basic 6 Databases	Charles Williams	Wrox	38
1861006314	GDI+ Programming: Creating Custom Controls Using C#	Eric White	Wrox	29



Create

ID	Name	Kind	Age
1	Suzy	Dog	12
2	Sandy	Cat	4

```
create table animals(
        id integer primary key,
        name TEXT,
        kind TEXT,
        age NUMERIC)
```

Insert

ID	Name	Kind	Age
1	Suzy	Dog	12
2	Sandy	Cat	4
3	Whiskers	Hamster	2

```
insert
        into animals(id,name,kind,age)
       values (3,'Whiskers','Hamster',2)
```

Update

ID	Name	Kind	Age
1	Suzy	Dog	12
2	Sandy	Cat	5
3	Whiskers	Hamster	2

update animals set age = 5 where kind='Cat'

Delete

ID	Name	Kind	Age
1	Suzy	Dog	12
2	Sandy	Cat	4
3	Whiskers	Hamster	2

delete from animals where age < 3

Break Time!

More Topics

- Quiz
- Selects and Joins
- ACID concepts
- Programming the database API
- Homework

Select

- For a single table
  - Returns a subset of columns
  - Returns a subset of rows (based on where)
  - Orders rows
  - Groups rows

Select

- For a multiple tables
  - Creates a larger view onto both tables
  - “where” defines connection between tables
  - Returns a subset of columns and rows as before
  - Orders rows, groups rows

Select Example

ID	Name	Kind	Age
1	Suzy	Dog	12
2	Sandy	Cat	4
3	Whiskers	Hamster	2

- This is a “flat file”

Select Example

ID	Name	Kind	Age
1	Suzy	Dog	12
2	Sandy	Cat	4
3	Whiskers	Hamster	2

select name,kind from animals

Name	Kind
Suzy	Dog
Sandy	Cat
Whiskers	Hamster

Select Example

ID	Name	Kind	Age
1	Suzy	Dog	12
2	Sandy	Cat	4
3	Whiskers	Hamster	2

select name,kind from animals where age > 10

Name	Kind
Suzy	Dog

Select Example

ID	Name	KindID	Age
1	Suzy	22	12
2	Sandy	31	4
3	Whiskers	45	2
4	Heidi	22	13

ID	Kind
22	Dog
31	Cat
45	Hamster

- This is relational data

Select Example

ID	Name	KindID	Age
1	Suzy	22	12
2	Sandy	31	4
3	Whiskers	45	2
4	Heidi	22	13

ID	Kind
22	Dog
31	Cat
45	Hamster

- This is a relation

Select Example

ID	Name	KindID	Age
1	Suzy	22	12
2	Sandy	31	4
3	Whiskers	45	2
4	Heidi	22	13

ID	Kind
22	Dog
31	Cat
45	Hamster

- This is a many-to-one relation

Select Example

ID	Name	KindID	Age
1	Suzy	22	12
2	Sandy	31	4
3	Whiskers	45	2
4	Heidi	22	13

ID	Kind
22	Dog
31	Cat
45	Hamster

select name,kind

Select Example

ID	Name	KindID	Age
1	Suzy	22	12
2	Sandy	31	4
3	Whiskers	45	2
4	Heidi	22	13

ID	Kind
22	Dog
31	Cat
45	Hamster

```
select name,kind
    from animals, kinds
```

Select Example

ID	Name	KindID	Age
1	Suzy	22	12
2	Sandy	31	4
3	Whiskers	45	2
4	Heidi	22	13

ID	Kind
22	Dog
31	Cat
45	Hamster

```
select name,kind
    from animals, kinds
```

Select Example

ID	Name	KindID	Age
1	Suzy	22	12
2	Sandy	31	4
3	Whiskers	45	2
4	Heidi	22	13

ID	Kind
22	Dog
31	Cat
45	Hamster

```
select name,kind
    from animals, kinds
    where animals.kindid = species.id
```

Select Example

ID	Name	KindID	Age
1	Suzy	22	12
2	Sandy	31	4
3	Whiskers	45	2
4	Heidi	22	13

ID	Kind
22	Dog
31	Cat
45	Hamster

```
select name,kind
    from animals, kinds
    where animals.kindid = species.id
```

Name	Kind
Suzy	Dog
Sandy	Cat
Whiskers	Hamster

Select Example

ID	Name	KindID	Age
1	Suzy	22	12
2	Sandy	31	4
3	Whiskers	45	2
4	Heidi	22	13

ID	Kind
22	Dog
31	Cat
45	Hamster

```
select name,kind
    from animals, kinds
    where animals.kindid = species.id
    and animals.age > 10
```

Name	Kind
Suzy	Dog
Heidi	Dog

Join

- For a multiple tables
- Creates views on multiple tables
- Rows can be “connected” as with select

Join Example

ID	Name	KindID	Age
1	Suzy	22	12
7	Chipper	25	6
2	Sandy	31	4
3	Whiskers	45	2
4	Heidi	22	13

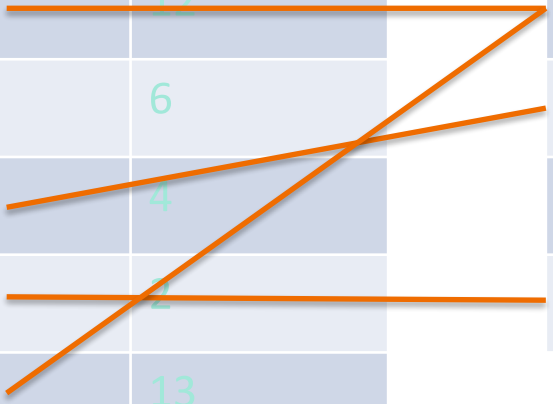
ID	Kind
22	Dog
31	Cat
37	Fish
45	Hamster

Add some rows that do not relate…

Join Example

ID	Name	KindID	Age
1	Suzy	22	12
7	Chipper	25	6
2	Sandy	31	4
3	Whiskers	45	2
4	Heidi	22	13

ID	Kind
22	Dog
31	Cat
37	Fish
45	Hamster



Related rows…

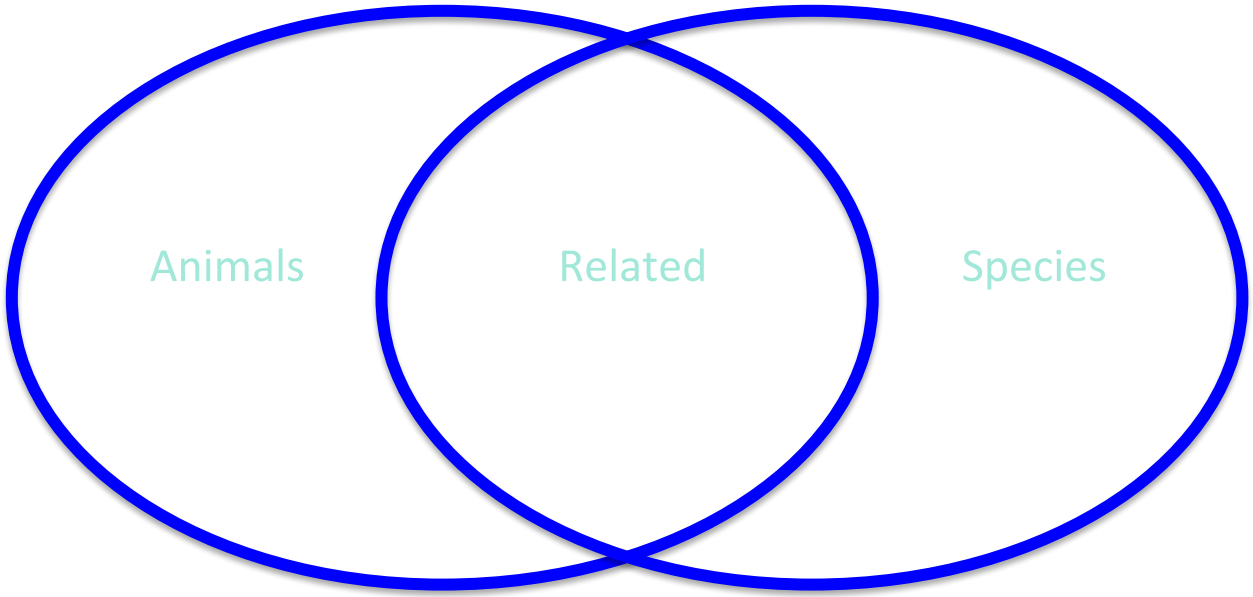
Join Example

ID	Name	KindID	Age
1	Suzy	22	12
7	**Chipper**	**25**	**6**
2	Sandy	31	4
3	Whiskers	45	2
4	Heidi	22	13

ID	Kind
22	Dog
31	Cat
37	**Fish**
45	Hamster

Unrelated rows… how do we handle these?

Join Types



Cross Join

LETTERS	NUMBERS
A	1
B	2
C	3
D	

Cross Join

LETTERS	NUMBERS
A	1
B	2
C	3
D	

Select * from letters, numbers

Cross Join

LETTERS	NUMBERS
A	1
B	2
C	3
D	

Select * from letters, numbers

A 1
A 2
A 3
B 1
B 2
B 3
C 1
C 2
C 3
D 1
D 2
D 3

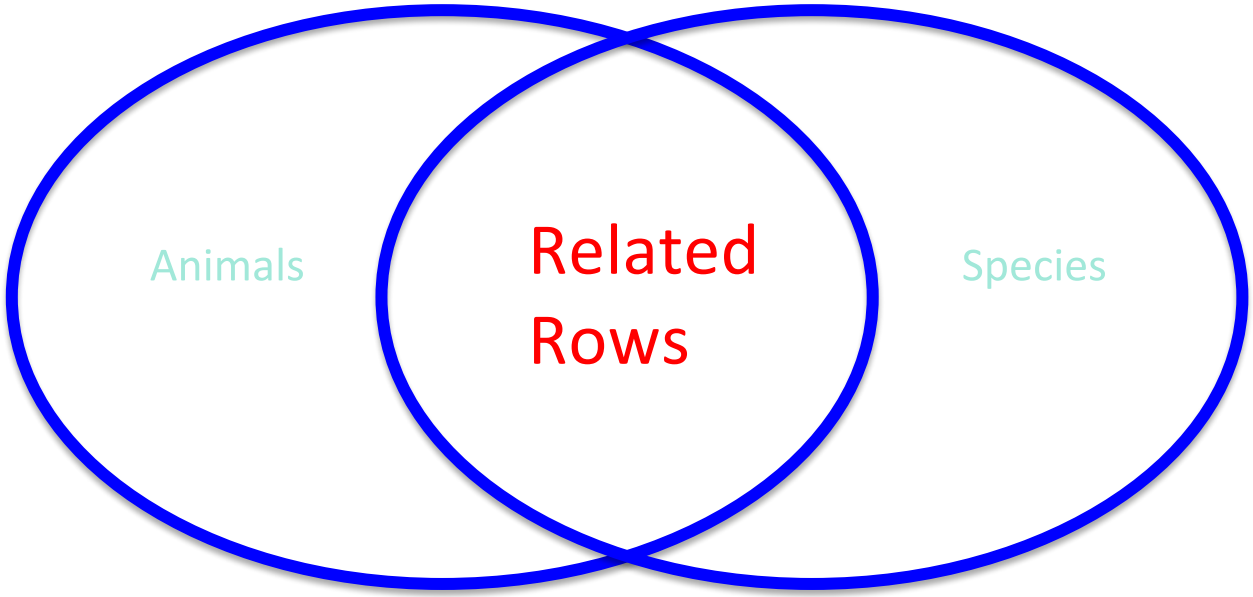
Cross Join

LETTERS	NUMBERS
A	1
B	2
C	3
D	

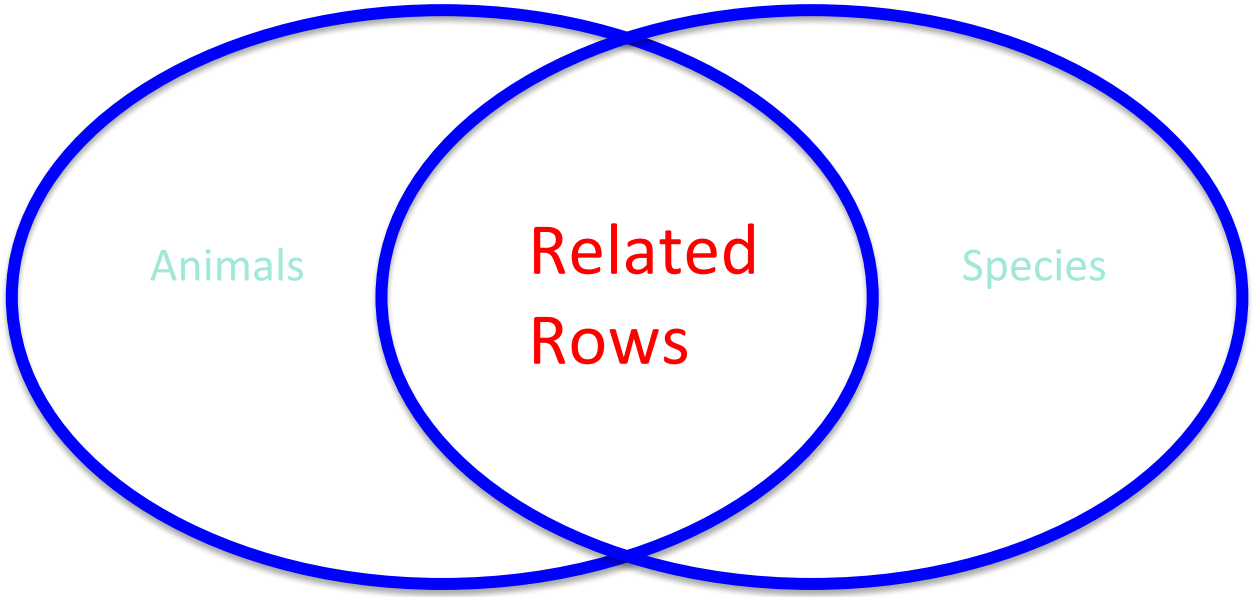
Select * from letters, numbers where n <= 2

A 1
A 2
B 1
B 2
C 1
C 2
D 1
D 2

Inner Join



Inner Join



Name	Kind
Suzy	Dog
Sandy	Cat
Whiskers	Hamster
Heidi	Dog

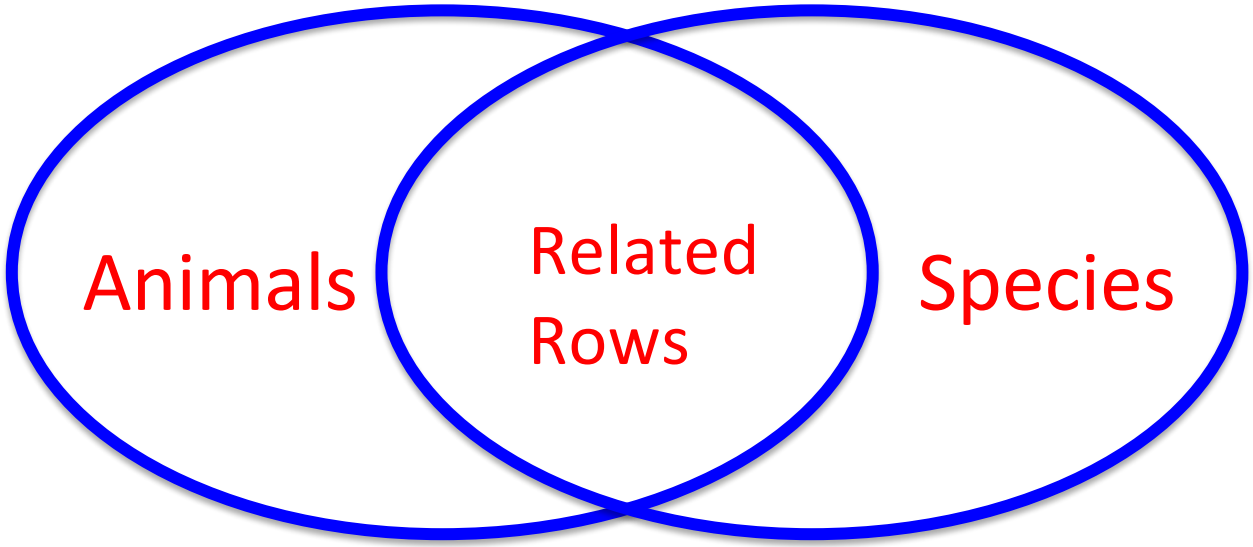
Join Example

ID	Name	KindID	Age
1	Suzy	22	12
7	**Chipper**	**25**	**6**
2	Sandy	31	4
3	Whiskers	45	2
4	Heidi	22	13

ID	Kind
22	Dog
31	Cat
37	**Fish**
45	Hamster

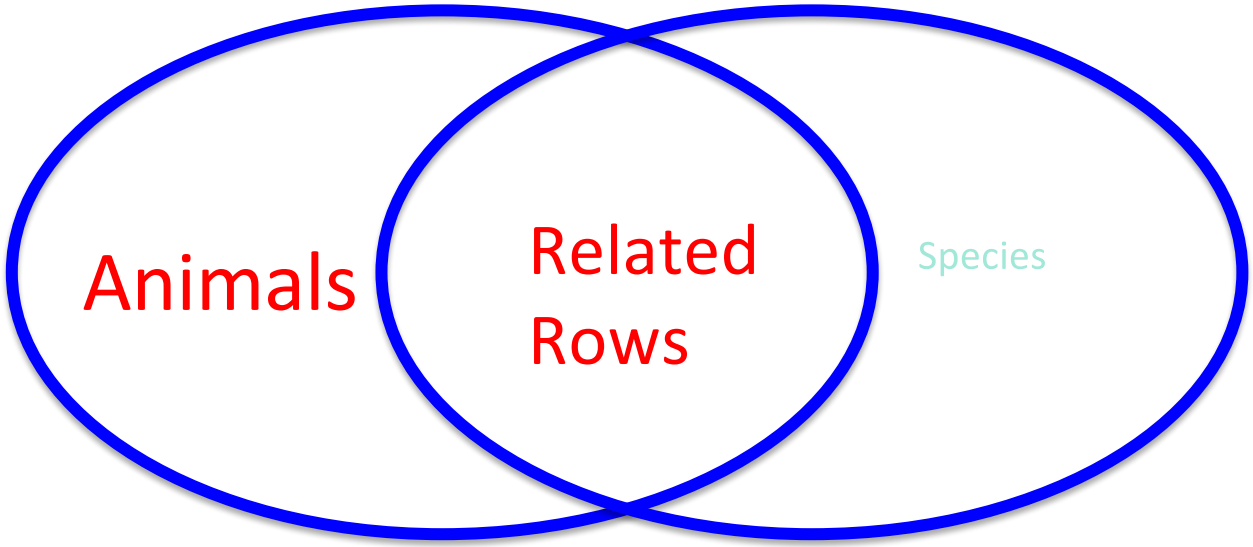
Unrelated rows…

Outer Join



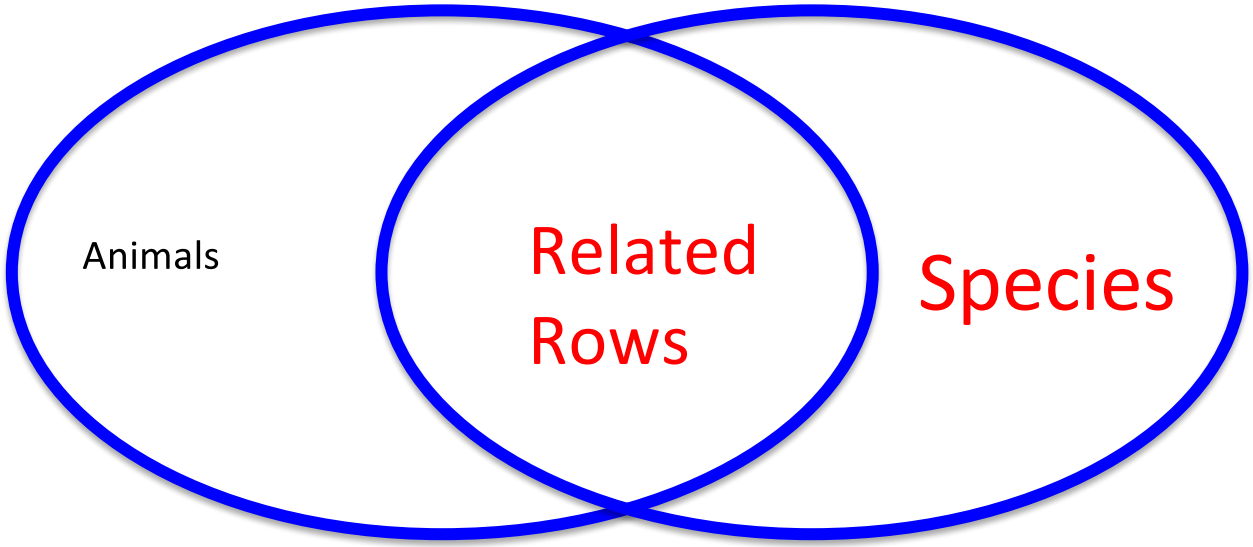
Name	Kind
Suzy	Dog
Sandy	Cat
Whiskers	Hamster
<null>	Fish
Chipper	<null>
Heidi	Dog

Left Outer Join



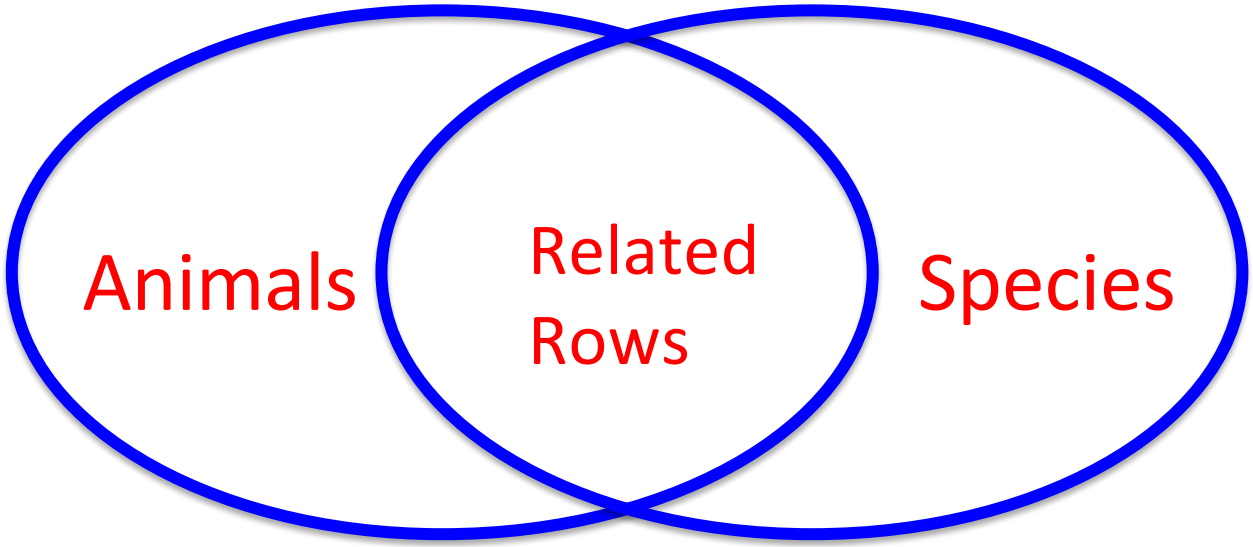
Name	Kind
Suzy	Dog
Sandy	Cat
Whiskers	Hamster
Chipper	<null>
Heidi	Dog

Right Outer Join



Name	Kind
Suzy	Dog
Sandy	Cat
Whiskers	Hamster
<null>	Fish
Heidi	Dog

(Full) Outer Join



Name	Kind
Suzy	Dog
Sandy	Cat
Whiskers	Hamster
<null>	Fish
Chipper	<null>
Heidi	Dog

ACID

- Atomic
- Consistent
- Isolated
- Durable

Atomic

- All or nothing
- No partial transactions

Consistent

- All data meets constraints
- No violation states

Isolated

- Concurrent transactions cant see each other’s stats
- There must be a sequential equivalent

Durable

- Any committed transaction must be permanent
- No excuses
- Not even power outages or crashes

.SQL files

- You can put SQL commands in a file
- Use the sqlite3 '.read' command

Python SQLite API

- import the sqlite3 library
  - *import sqlite3*
- open a database
  - connection = sqlite3.connect("sample.db")
- create a cursor
  - cursor = connection.cursor()
- execute statements with the cursor
  - cursor.execute(….)

Python SQLite Statements

- execute statements with the cursor
  - cursor.execute("create table …")
- commit (save) the changes
  - connection.commit()
- close the database
  - connection.close()

Python Statement Injection

- Putting values in SQL statements is risky
  - (A few words about SQL injection here…)
  - More on this later
- Use the value substitutions in SQLite
  - c.execute("select … where id>?",t)
  - c.execute("select … where id>? and x<?,(3,4))
- You can execute many:
  - c.executemany("insert…",[(),(),(),…])

Python SQLite Queries

- execute queries with the cursor
  - cursor.execute("select * from …")
- iterate over the result
  - for row in cursor.execute("…"):
- get one value
  - cursor.execute("…")
  - row = cursor.fetchone()
- or get all the values
  - rows = cursor.fetchall()

Demo Time!