# CCT1343, Week 3

## SQL Queries Using Multiple Tables

Yuri Takhteyev
University of Toronto
January 17, 2011

# Projection

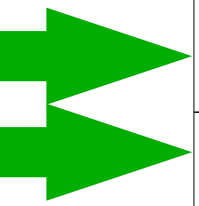
| name | owner | species | sex | birth |
|---|---|---|---|---|
| Fluffy | Harold | cat | f | 1993-02-04 |
| Bluffy | Harold | dog | f | 1989-05-13 |
| Chirpy | Gwen | bird | f | 1998-09-11 |

# Projection

| name | species | sex |
| --- | --- | --- |
| Fluffy | cat | f |
| Bluffy | dog | f |
| Chirpy | bird | f |

# Selection

("Restriction" in Harrington)



| name | owner | species | sex | birth |
|---|---|---|---|---|
| Fluffy | Harold | cat | f | 1993-02-04 |
| Bluffy | Harold | dog | f | 1989-05-13 |
| Chirpy | Gwen | bird | f | 1998-09-11 |

projection

```
select name, species
from pet
where weight < 1;
```

selection

# Also:

- Ordering the results
- Limit
- Aggregation (count, sum, avg)
- Grouping

# Putting It All Together

```
    select
6.  weight
1.  from pet
2.  where weight>1
3.  group by species
4.  having count(name)>1
5.  order by sum(weight)
7.  limit 1;
```

# More Data

pet

| name | owner | species |
|---|---|---|
| Fluffy | Harold | cat |
| Buffy | Harold | dog |
| Chirpy | Gwen | bird |
| Fang | Benny | dog |

# More Data

pet

| name | owner | species | food |
|---|---|---|---|
| Fluffy | Harold | cat | cat food |
| Buffy | Harold | dog | dog food |
| Chirpy | Gwen | bird | seeds |
| Fang | Benny | dog | dog food |

# More Data

pet

| name | owner | species | food | owner tel |
|---|---|---|---|---|
| Fluffy | Harold | cat | cat food | 416.123.1234 |
| Buffy | Harold | dog | dog food | 416.123.1234 |
| Chirpy | Gwen | bird | seeds | 647.987.6543 |
| Fang | Benny | dog | dog food | 901.129.2832 |

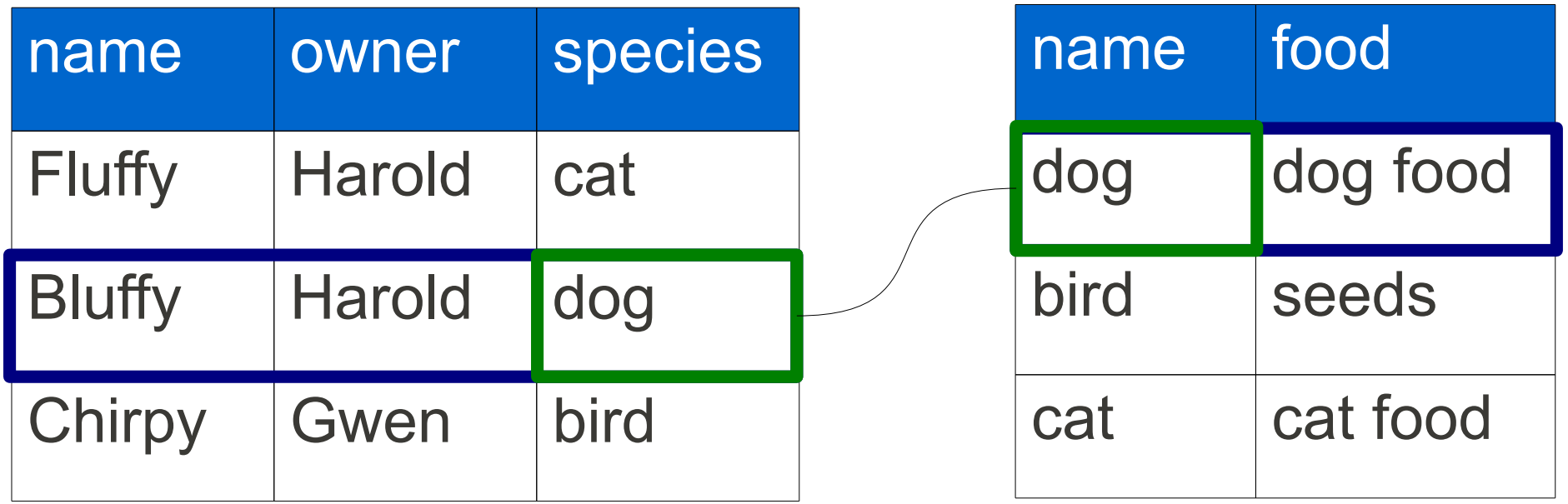
# Multiple Tables

pet

| name | owner | species |
|---|---|---|
| Fluffy | Harold | cat |
| Bluffy | Harold | dog |
| Chirpy | Gwen | bird |

species

| name | food |
|---|---|
| dog | dog food |
| bird | seeds |
| cat | cat food |



# Join

| name | owner | species | food |
| --- | --- | --- | --- |
| Fluffy | Harold | cat | cat food |
| Bluffy | Harold | dog | dog food |
| Chirpy | Gwen | bird | seeds |

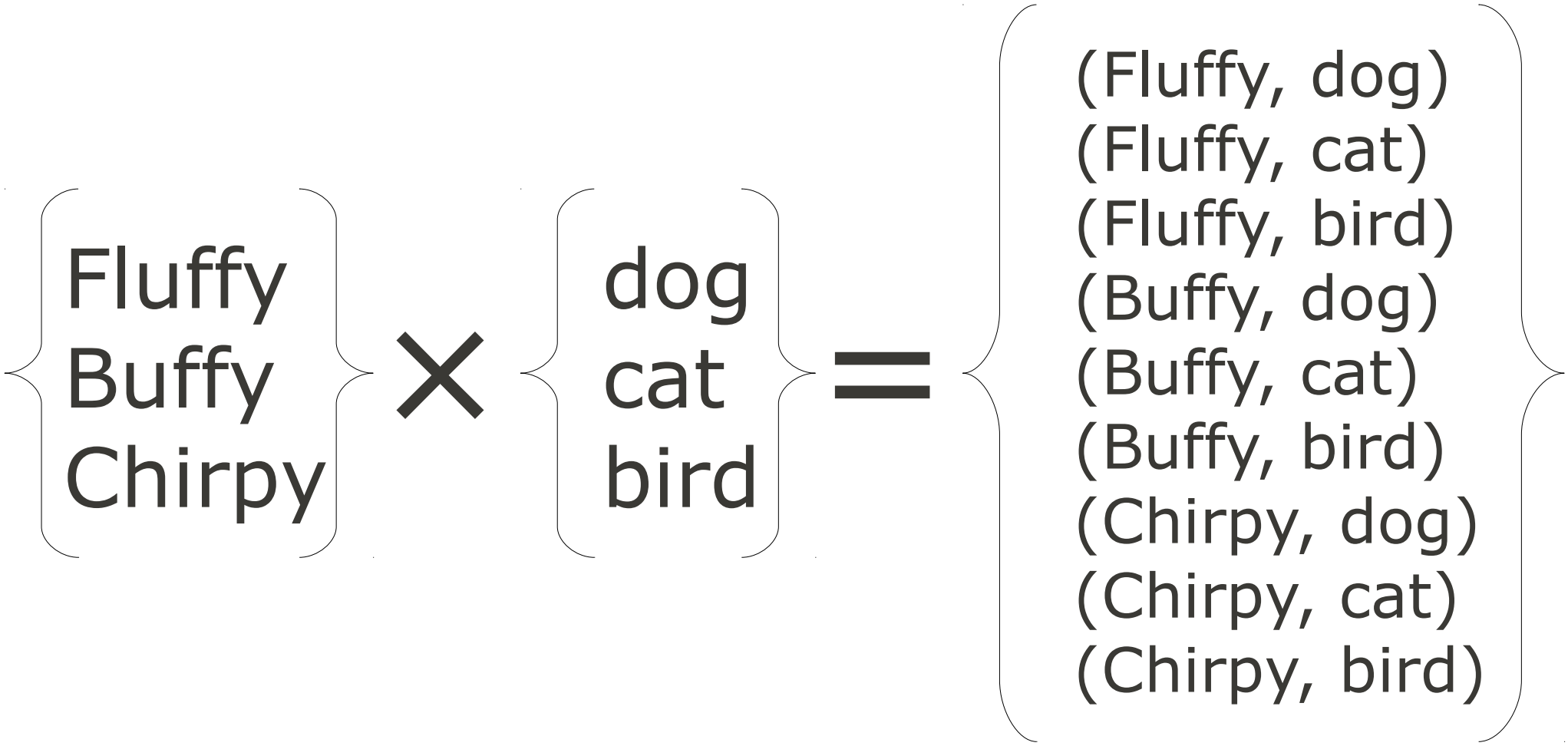
# SQL Select

1. identifying a source table
2. selection
3. grouping
4. group selection
5. ordering
6. projecting fields
7. taking some of results (“limit”)

# SQL Select

1. identifying a source table
1a. joining additional tables
2. selection
3. grouping
4. group selection
5. ordering
6. projecting fields
7. taking some of results (“limit”)

# Cartesian Product



# Product of Tables

pet

| name | owner | species |
|---|---|---|
| Fluffy | Harold | cat |
| Bluffy | Harold | dog |
| Chirpy | Gwen | bird |

×

species

| name | food |
|---|---|
| dog | dog food |
| bird | seeds |
| cat | cat food |

| name | owner | species | species | food |
|---|---|---|---|---|
| Fluffy | Harold | cat | cat | cat food |
| Bluffy | Harold | dog | cat | cat food |
| Chirpy | Gwen | bird | cat | cat food |
| Fluffy | Harold | cat | dog | dog food |
| Bluffy | Harold | dog | dog | dog food |
| Chirpy | Gwen | bird | dog | dog food |
| Fluffy | Harold | cat | bird | seeds |
| Bluffy | Harold | dog | bird | seeds |
| Chirpy | Gwen | bird | bird | seeds |

| name | owner | species | species | food |
|---|---|---|---|---|
| Fluffy | Harold | cat | cat | cat food |
| Bluffy | Harold | dog | cat | cat food |
| Chirpy | Gwen | bird | cat | cat food |
| Fluffy | Harold | cat | dog | dog food |
| Bluffy | Harold | dog | dog | dog food |
| Chirpy | Gwen | bird | dog | dog food |
| Fluffy | Harold | cat | bird | seeds |
| Bluffy | Harold | dog | bird | seeds |
| Chirpy | Gwen | bird | bird | seeds |

| name | owner | species | species | food |
|---|---|---|---|---|
| Fluffy | Harold | cat | cat | cat food |
| Bluffy | Harold | dog | cat | cat food |
| Chirpy | Gwen | bird | cat | cat food |
| Fluffy | Harold | cat | dog | dog food |
| Bluffy | Harold | dog | dog | dog food |
| Chirpy | Gwen | bird | dog | dog food |
| Fluffy | Harold | cat | bird | seeds |
| Bluffy | Harold | dog | bird | seeds |
| Chirpy | Gwen | bird | bird | seeds |

cartesian product + selection

=

relational join

selection based on equality



“equi-join”

# SQL-92 Inner Join

(aka “ANSI Join”)

```
select ... from «table1»
join «table2» on «conditions»;
```

For instance:

```
select pet.name, species.food
from pet join species
on pet.species = species.name;
```

# SQL-92 Inner Join

```
select ... from «table1»
join «table2» on «conditions»;
```

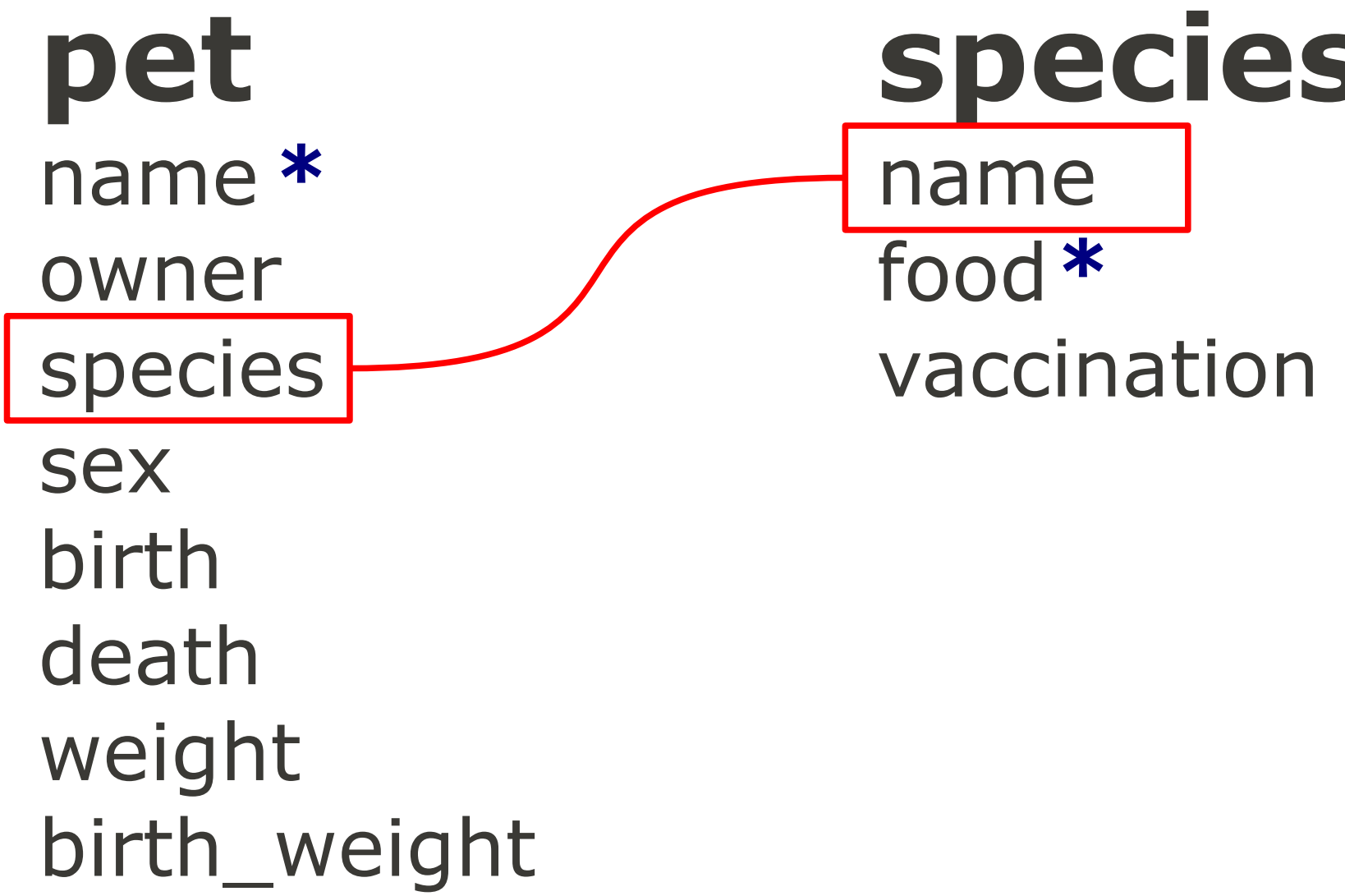
For instance:

```
select pet.name, species.food
from pet join species
on pet.species = species.name;
```

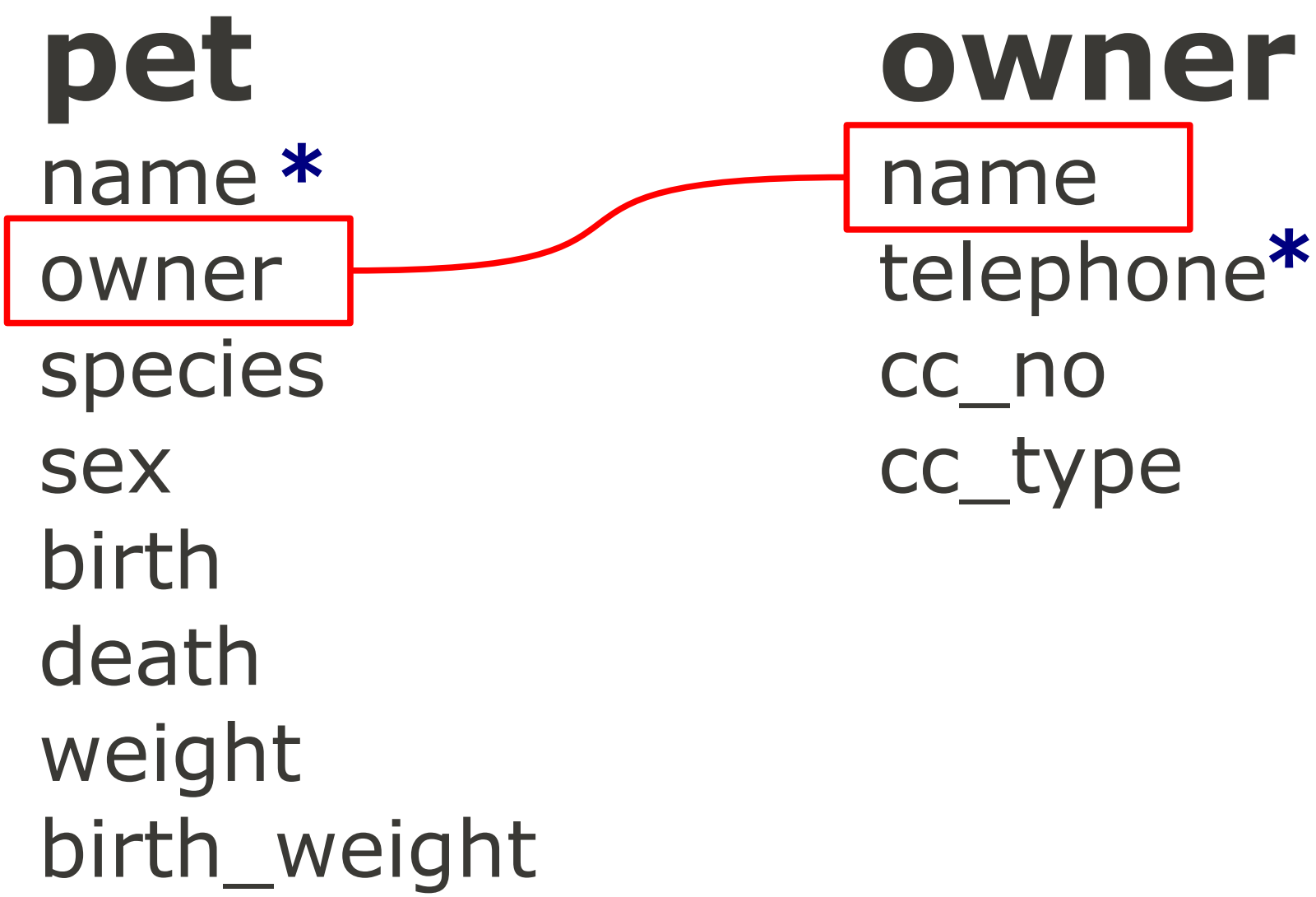
```
+----------+----------+
| name     | food     |
+----------+----------+
| Fluffy   | cat food |
| Claws    | cat food |
| Buffy    | dog food |
| Fang     | dog food |
| Bowser   | dog food |
| Chirpy   | seeds    |
| Whistler | seeds    |
| Slim     | mice     |
+----------+----------+
8 rows in set (0.00 sec)
```

without the “on” clause (depends on the db) →

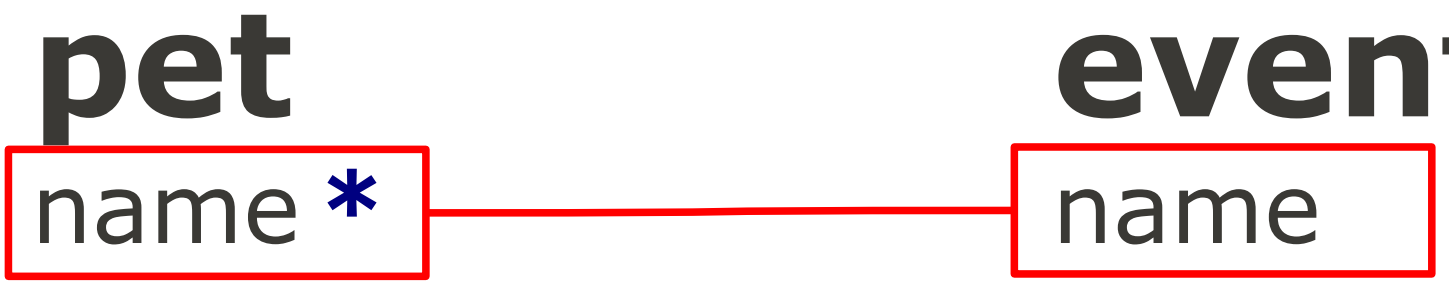
```
+----------+----------+
| name     | food     |
+----------+----------+
| Fluffy   | cat food |
| Fluffy   | dog food |
| Fluffy   | seeds    |
| Fluffy   | mice     |
| Claws    | cat food |
| Claws    | dog food |
| Claws    | seeds    |
| Claws    | mice     |
| Buffy    | cat food |
             ...
| Slim     | cat food |
| Slim     | dog food |
| Slim     | seeds    |
| Slim     | mice     |
| Puffball | cat food |
| Puffball | dog food |
| Puffball | seeds    |
| Puffball | mice     |
+----------+----------+
36 rows in set (0.00 sec)
```




```
select pet.name, species.food
from pet join species on
pet.species=species.name;
```




```
select pet.name, owner.telephone
from pet join owner on
pet.owner=owner.name;
```




```
select pet.name, event.type
from pet join event on
pet.name=event.name;
```

# Table Aliases

```
select pet.name, species.food
from pet join species
on pet.species = species.name;
```

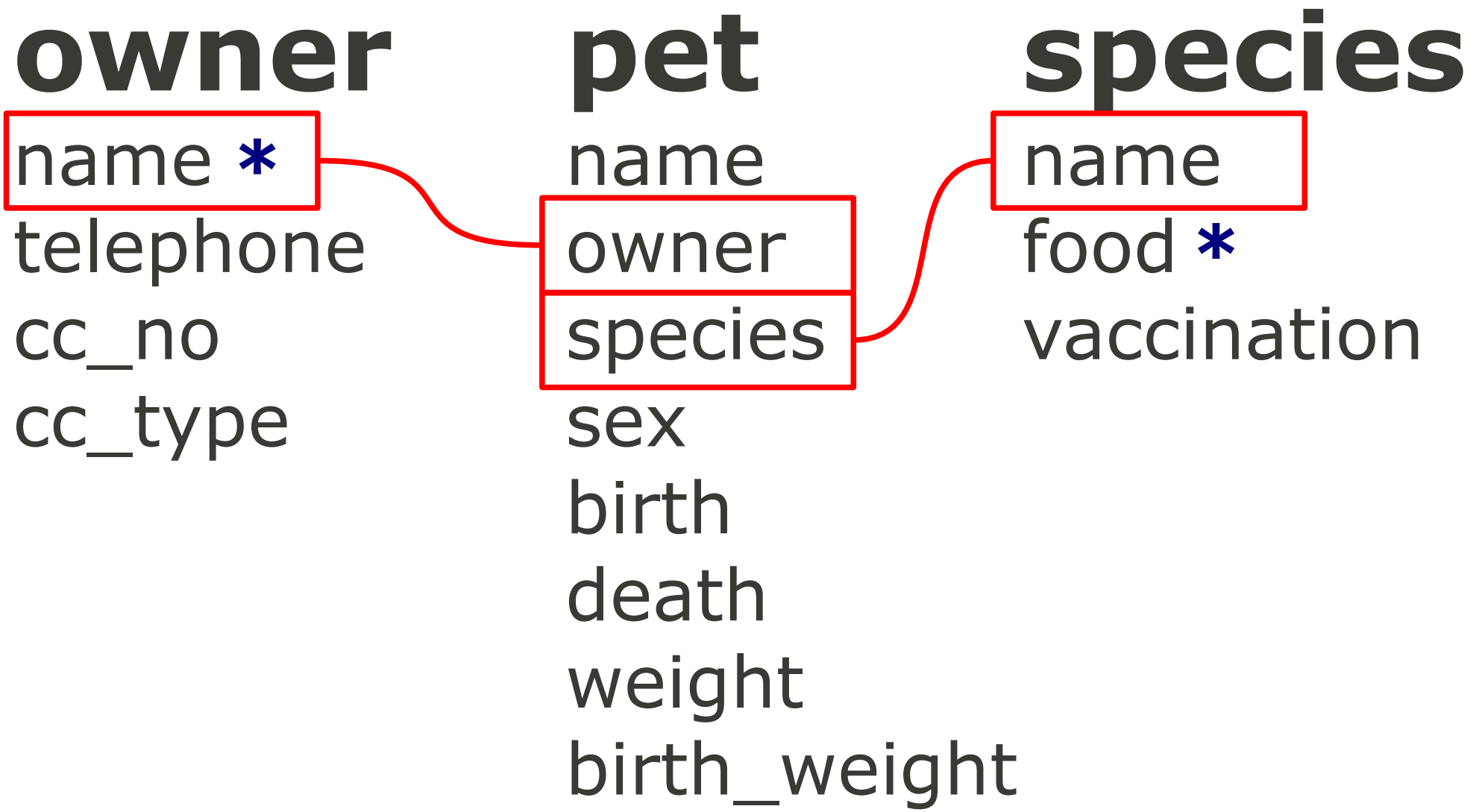
```
select p.name, s.food
from pet as p join species as s
on p.species = s.name;
```

# Self-Join

```
select …
from «table» as «alias1»
join «table» as «alias1»
on «conditions»;
```

```
select p1.name, p2.name
from pet as p1 join pet as p2
on p1.species = p2.species
where p1.name < p2.name;
```

```
+--------+-----------+
| name   | name      |
+--------+-----------+
| Claws  | Fluffy    |
| Bowser | Buffy     |
| Buffy  | Fang      |
| Bowser | Fang      |
| Chirpy | Whistler  |
+--------+-----------+
5 rows in set (0.00 sec)
```




```
owner join pet on...
join species on...
```

# Multiple Joins

```
select ... from «table1»
join «table2» on «condition1»
join «table3» on «condition2»;
```

```
select owner.name, food from owner
join pet
 on pet.owner = owner.name
join species
 on pet.species = species.name;
```

```
+--------+-----------+
| name   | food      |
+--------+-----------+
| Harold | cat food  |
| Gwen   | cat food  |
| Harold | dog food  |
| Diane  | dog food  |
| Gwen   | seeds     |
| Gwen   | seeds     |
+--------+-----------+
6 rows in set (0.00 sec)
```

# Easier Equi-Joins

pet

| name | owner |
|---|---|
| Fluffy | Harold |
| Bluffy | Harold |
| Chirpy | Gwen |
| Fang | Benny |

owner

| name | telephone |
|---|---|
| Gwen | 16472939823 |
| Harold | 14092938489 |
| Diane | 552122347849 |

# Easier Equi-Joins

pet

| pet_name | owner_name |
|---|---|
| Fluffy | Harold |
| Bluffy | Harold |
| Chirpy | Gwen |
| Fang | Benny |

owner

| owner_name | telephone |
|---|---|
| Gwen | 16472939823 |
| Harold | 14092938489 |
| Diane | 552122347849 |

# Join... Using...

```
select ... from «table1»
join «table2»
using («columns»);
```

For instance:

```
select pet_name, food
from pet join owner
using (owner_name);
```

# Yet Easier Equi-Joins

pet

| pet_name | owner_name |
|---|---|
| Fluffy | Harold |
| Bluffy | Harold |
| Chirpy | Gwen |
| Fang | Benny |

owner

| owner_name | telephone |
|---|---|
| Gwen | 16472939823 |
| Harold | 14092938489 |
| Diane | 552122347849 |

# Natural Join

```
select ... from «table1»
natural join «table2»;
```

For instance:

avoid

```
select pet_name, food
from pet natural join owner;
```

# Why Avoid It?

implicit selection of columns

=

bad idea

# “Traditional” Join

```
select ...
from «table1», «table2»
where «join_conditions»;
```

For instance:

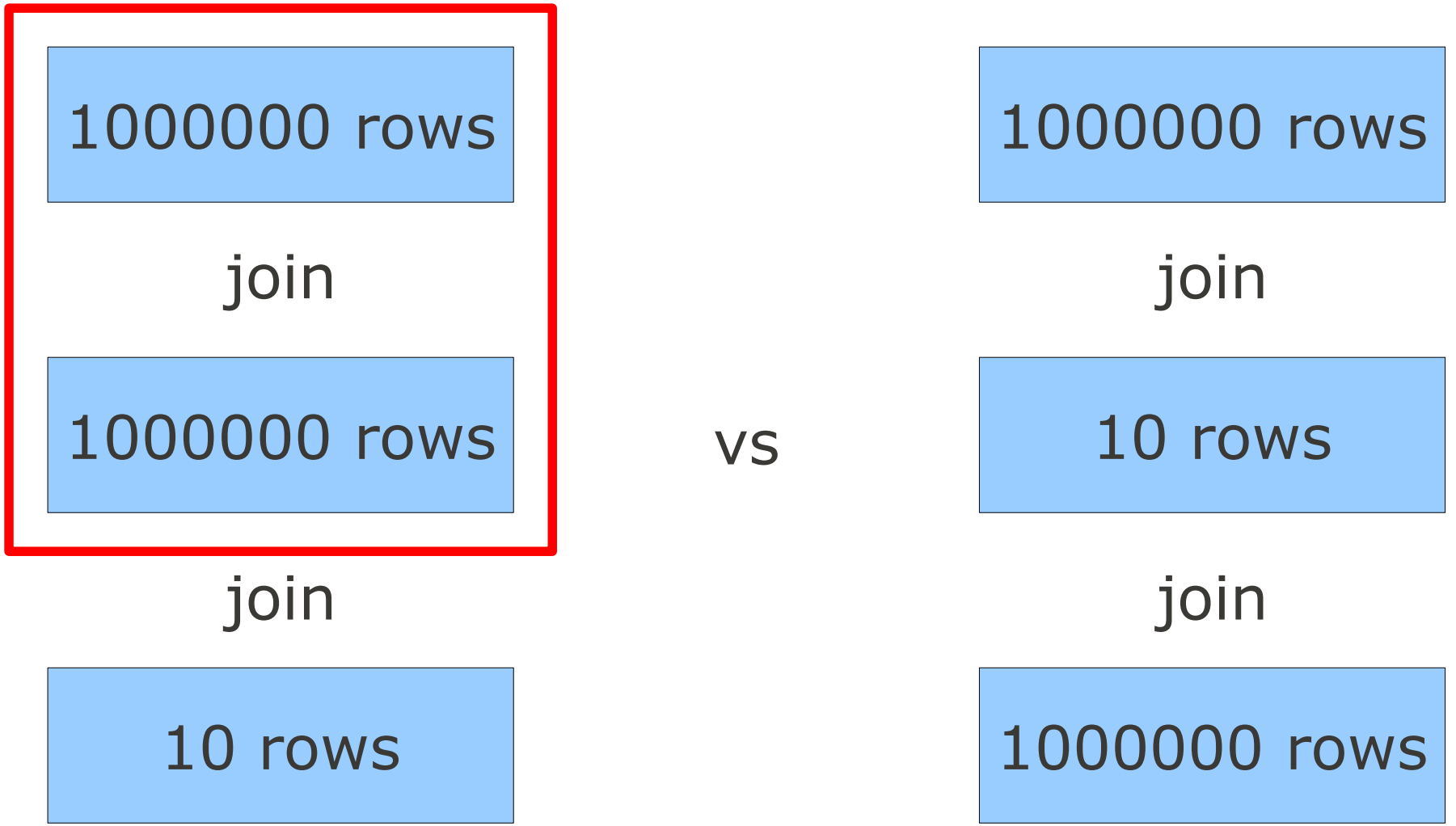
avoid

```
select name, food
from pet, owner
where pet.owner=owner.name;
```

# Use SQL-92 “Join”

- More options
  - e.g., “outer”
- More clear
  - avoids making “where” ambiguous
- Control over the order of joins

# Order of Joins



Questions?

# Inner vs. Outer

**Inner Join:**

only pairs that satisfy the condition

**Outer Joins:**

includes non-matched rows from one of the tables, or both

-> “left”, “right”, “full”

# Why Do Outer Joins?

pet

| name | owner |
|---|---|
| Fluffy | Harold |
| Buffy | Harold |
| Chirpy | Gwen |
| Fang | Benny |

owner

| name | telephone |
|---|---|
| Gwen | 16472939823 |
| Harold | 14092938489 |
| Diane | 552122347849 |

# An Inner Join

pet join owner on pet.owner=owner.name

| name | owner | telephone |
|---|---|---|
| Fluffy | Harold | 14092938489 |
| Buffy | Harold | 14092938489 |
| Chirpy | Gwen | 552122347849 |

What happened to Fang?

# What We Might Want

| name | owner | telephone |
|---|---|---|
| Fluffy | Harold | 14092938489 |
| Bluffy | Harold | 14092938489 |
| Chirpy | Gwen | 552122347849 |
| Fang | Benny | NULL |

# Left Outer Join

```
select ... from «table1»
left outer join «table2»
on «conditions»;
```

include unmatched rows from the **left** table

For instance:

```
select pet.name, pet.owner,
owner.telephone
from pet left outer join owner
on pet.owner = owner.name;
```

left table

right table

```
+----------+--------+--------------+
| name     | owner  | telephone    |
+----------+--------+--------------+
| Fluffy   | Harold | 14092938489  |
| Claws    | Gwen   | 16472939823  |
| Buffy    | Harold | 14092938489  |
| Fang     | Benny  | NULL         |
| Bowser   | Diane  | 552122347849 |
| Chirpy   | Gwen   | 16472939823  |
| Whistler | Gwen   | 16472939823  |
| Slim     | Benny  | NULL         |
| Puffball | Diane  | 552122347849 |
+----------+--------+--------------+
9 rows in set (0.00 sec)
```

# Other Outer Joins

**Right Outer Join:**

unmatched rows from the *right* table

**Full Outer Join:**

unmatched rows from *both* sides (not available in mysql)

# StarWars!

1. Which characters belong to species with typical lifespan > 200 years?

2. Which characters belong to species  that come from worlds that are more than 50% water (by surface area)?

```
select
   persona.name
from
   persona
join world



where
   world.percent_water > 50;
```

```
select
  persona.name
from
  persona
join ???

join world


where
  world.percent_water > 50;
```

```
select
  persona.name
from
  persona
join species
  using (species)
join world
  on species.homeworld=
  world.world_name
where
  world.percent_water > 50;
```

# Advanced Joins

3. Which characters come from worlds that have more water than the world where their species originated?

Hint: join **world** twice, with different aliases

# The Two Worlds

```
select
   w1.world_name, w2.world_name
from world as w1
join world as w2
where
   w1.percent_water >
   w2.percent_water;
```

# Persona + Species

```
select
  persona.name, species.species
from persona
join species on
  persona.species=species.species;
```

# ... + w1

```
select
  persona.name, species.species,
  w1.percent_water
from persona
join species on
  persona.species=species.species
join world as w1 on
  persona.homeworld=w1.world_name;
```

# ... + w2

```
select
  persona.name, species.species,
  w1.percent_water, w2.percent_water
from persona
join species on
  persona.species=species.species
join world as w1 on
  persona.homeworld=w1.world_name
join world as w2 on
  species.homeworld=w2.world_name;
```

# Adding WHERE

```
select
  persona.name, species.species,
  w1.percent_water, w2.percent_water
from persona
join species on
  persona.species=species.species
join world as w1 on
  persona.homeworld=w1.world_name
join world as w2 on
  species.homeworld=w2.world_name
where
 w1.percent_water>w2.percent_water;
```

# The Final Answer

```
select
  persona.name
from persona
join species on
  persona.species=species.species
join world as w1 on
  persona.homeworld=w1.world_name
join world as w2 on
  species.homeworld=w2.world_name
where
 w1.percent_water>w2.percent_water;
```

# Keys

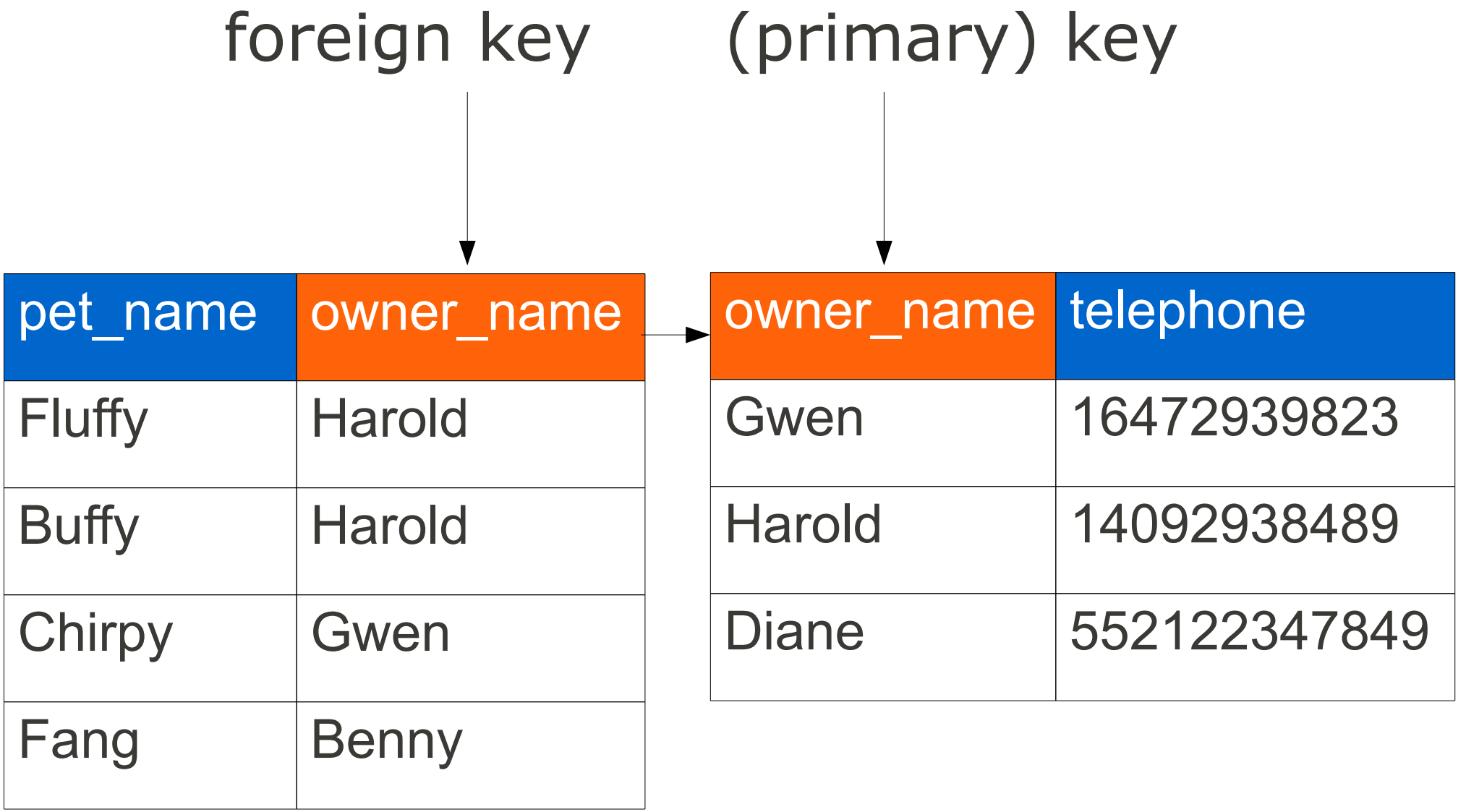
pet

| pet_name | owner_name |
|---|---|
| Fluffy | Harold |
| Buffy | Harold |
| Chirpy | Gwen |
| Fang | Benny |

owner

| owner_name | telephone |
|---|---|
| Gwen | 16472939823 |
| Harold | 14092938489 |
| Diane | 552122347849 |

owner names are unique here


foreign key
(primary) key
pet_name
owner_name
Fluffy
Harold
Buffy
Harold
Chirpy
Gwen
Fang
Benny
owner_name
telephone
Gwen
16472939823
Harold
14092938489
Diane
552122347849

# “Harold” is now “Bob”

| pet_name | owner_name |
|---|---|
| Fluffy | Harold |
| Buffy | Harold |
| Chirpy | Gwen |
| Fang | Benny |

| owner_name | telephone |
|---|---|
| Gwen | 16472939823 |
| Bob | 14092938489 |
| Diane | 552122347849 |

# The Solution

| pet_name | owner_id |
|---|---|
| Fluffy | 2 |
| Buffy | 2 |
| Chirpy | 1 |
| Fang | ??? |



| owner_id | owner_name | telephone |
|---|---|---|
| 1 | Gwen | 1647293 |
| 2 | Harold | 1409293 |
| 3 | Diane | 5521223 |

# The Solution

| pet_name | owner_id |
|---|---|
| Fluffy | 2 |
| Buffy | 2 |
| Chirpy | 1 |
| Fang | 4 |



| owner_id | owner_name | telephone |
|---|---|---|
| 1 | Gwen | 1647293 |
| 2 | Harold | 1409293 |
| 3 | Diane | 5521223 |
| 4 | Benny | NULL |

# The Solution

| pet_name | owner_id |
|---|---|
| Fluffy | 2 |
| Buffy | 2 |
| Chirpy | 1 |
| Fang | 4 |



| owner_id | owner_name | telephone |
|---|---|---|
| 1 | Gwen | 1647293 |
| 2 | Bob | 1409293 |
| 3 | Diane | 5521223 |
| 4 | Benny | NULL |

Meaningless keys are (usually) best.

# IN

```
select ... from «some_table»
where «value» in («values»);
```

For instance:

```
select name from pet
where owner in ("Harold",
"Gwen");
```

# Subqueries

```
select ... from «some_table»
where «column» in («query2»);
```

For instance:

```
select name from pet
where owner in
(select name from owner where
 cc_type="visa");
```

```
select name from
owner where
cc_type="visa";
```

```
+--------+
| name   |
+--------+
| Gwen   |
| Harold |
+--------+
```

```
select name
from pet where
owner in (result);
```

```
+----------+
| name     |
+----------+
| Fluffy   |
| Claws    |
| Buffy    |
| Chirpy   |
| Whistler |
+----------+
```

compare with

```
select name from pet
where owner in ("Gwen",
"Harold");
```

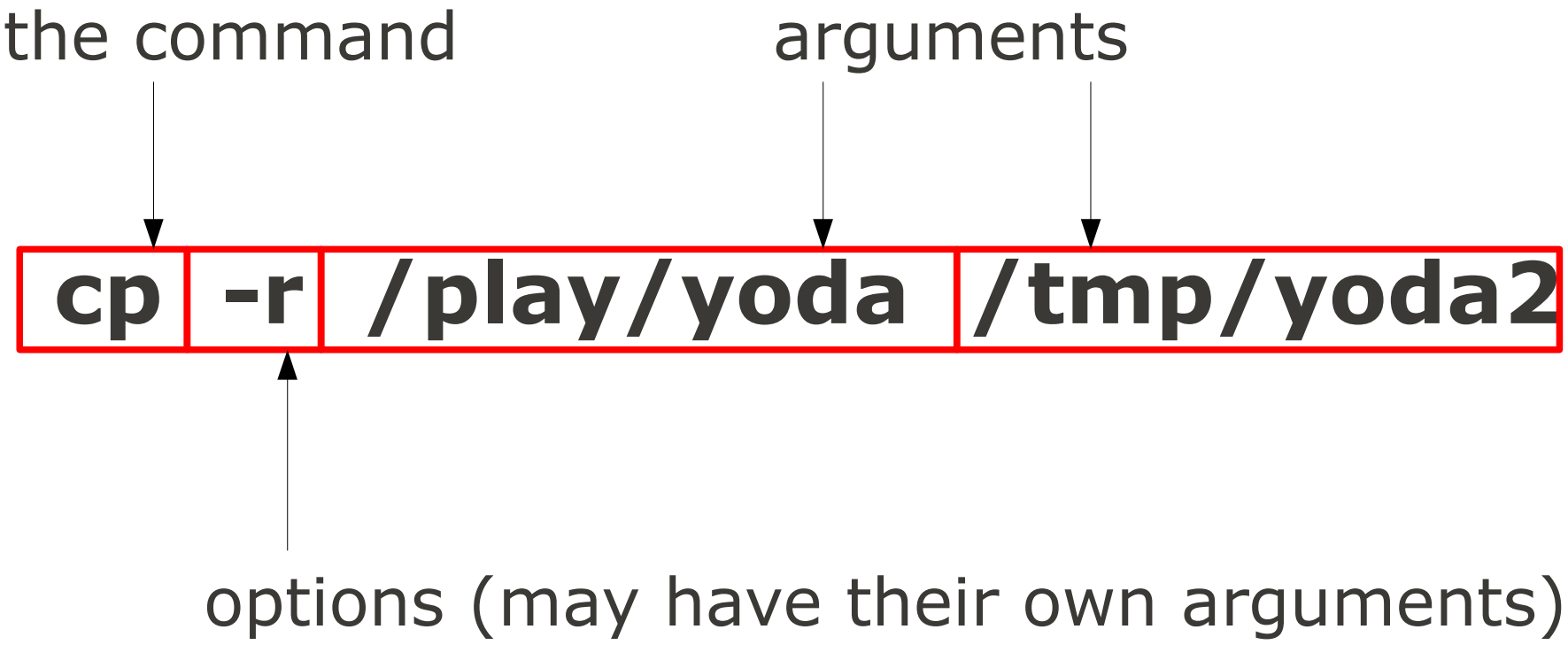
# Questions?

# More Unix Commands

**ls** – **l**i**s**t files in a directory
**cd** – **c**hange **d**irectory
**mkdir** – create (**m**a**k**e) a **dir**ectory
**rm** – delete ("**r**e**m**ove") a file or directory
**cp** – **c**o**p**y a file or directory
**less** – view a text file
**nano** – edit a text file
**mysql** – start mysql client

some of those commands are available both in your local and remote bash, some just on the server

# Anatomy of the Unix Command



# Some Examples

**cd /play**

go to directory “/play”

Hint: press [Tab] after typing “/pl”

**ls**

list the files in the current directory

**cd yoda**

go to directory “yoda”

Hint: press [Tab] after typing “y”

**ls**

Hint: use [↑] for earlier commands

# Some Examples

**less force.txt**

Hint: press [Tab] after typing “f”

Hint: press “q” to exit less

**cd ..**

go to up one level

**ls**

**cd locked**

go to directory “sandbox”

Hint: you don't have the permissions

# Some Examples

**cd sandbox**

**mkdir obiwan**

create a directory “obiwan”
(use your own name)

**ls**

we should see everyone's directory

**cd obiwan**

go to your directory

# Some Examples

**ls /play/yoda/**

What was that file called again?

**less /play/yoda/force.txt**

Let's look at it again.

**cp /play/yoda/force.txt .**

copy “force.txt” to the local directory

**nano force.txt**

edit force.txt

Hint: **^** means [Control]

# Options

**ls -sh**

list files with file sizes

**cp -r /play/yoda .**

copy “recursively”

**less -N force.txt .**

show the file with line numbers

# Getting Help

**man ls**

user **man**ual for the **ls** command

# Directories

**/home/kenobio7**
user's “home” directory

**~**
alias for user's home directory
e.g. “ls ~”

**.**
current directory

**..**
parent of the current directory

# Redirection

**command > file.txt**

write the output to file

**command < file.txt**

feed the content of file to the command

**command1 | comman2**

send the output of command1 to command2

(We'll see examples in a second.)

# MySQL

**mysql**
connect to mysql
**mysql -u *username* -p**
connect to mysql as a *kenobio7*, with a password

# SQL From a File

**cd ~**

**cp /play/yoda/humans.sql .**

**mysql < humans.sql**

run mysql client feeding it the contents of “non-humans.sql”

**mysql < humans.sql > h.txt**

save the output into “h.txt”

Exercise: create a file “ewoks.sql” that would give us a list of **Ewoks**.

# Using SCP

**scp = s**ecure **c**o**p**y (or **s**sh + **cp**)
copy files over an ssh connection

Hint: You will usually be running this in your **local** bash session (i.e. on your laptop/desktop).

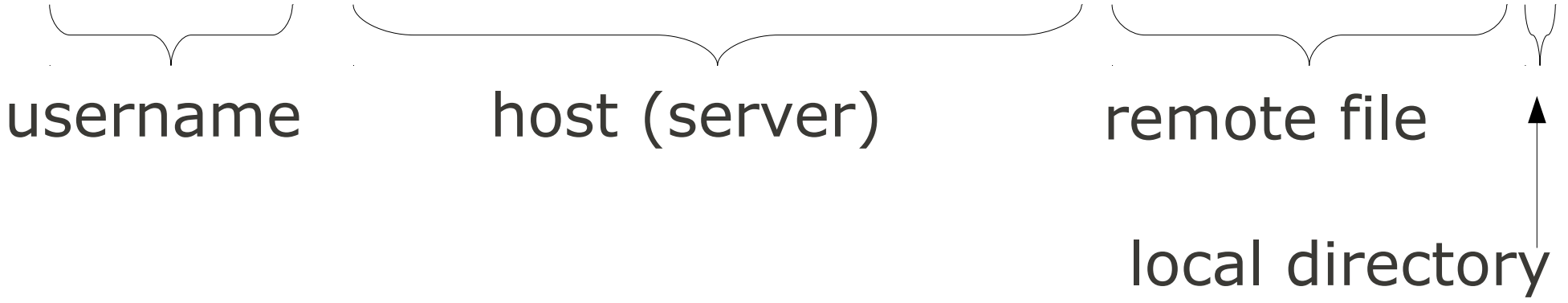
Hint: Windows users can use WinSCP instead.

# Remote to Local

scp user@host:/remote/file /local/dir

e.g.:

scp kenobio7@yoda.ischool.utoronto.ca:~/humans.txt .



# Local to Remote

scp /local/file user@host:/remote/dir

e.g.:

scp ewoks.sql kenobio7@yoda.ischool.utoronto.ca:~/

# Editing Files Locally

Windows: **Notepad++**
Mac: **TextWrangler**
Linux: **gedit** (or emacs, vi)

Key feature: syntax highlighting