CCT396, Fall 2011

Database Design and Implementation

Yuri Takhteyev
University of Toronto

1:1 Relationships

Option 1:

Use the same table.

Option 2:

Use a single-attribute FK
as the PK in one of the tables.

Multivalued Attributes

customer:

- name
- phone number(s)
- email addresse(s)

restaurant:

- name
- address
- tag(s)

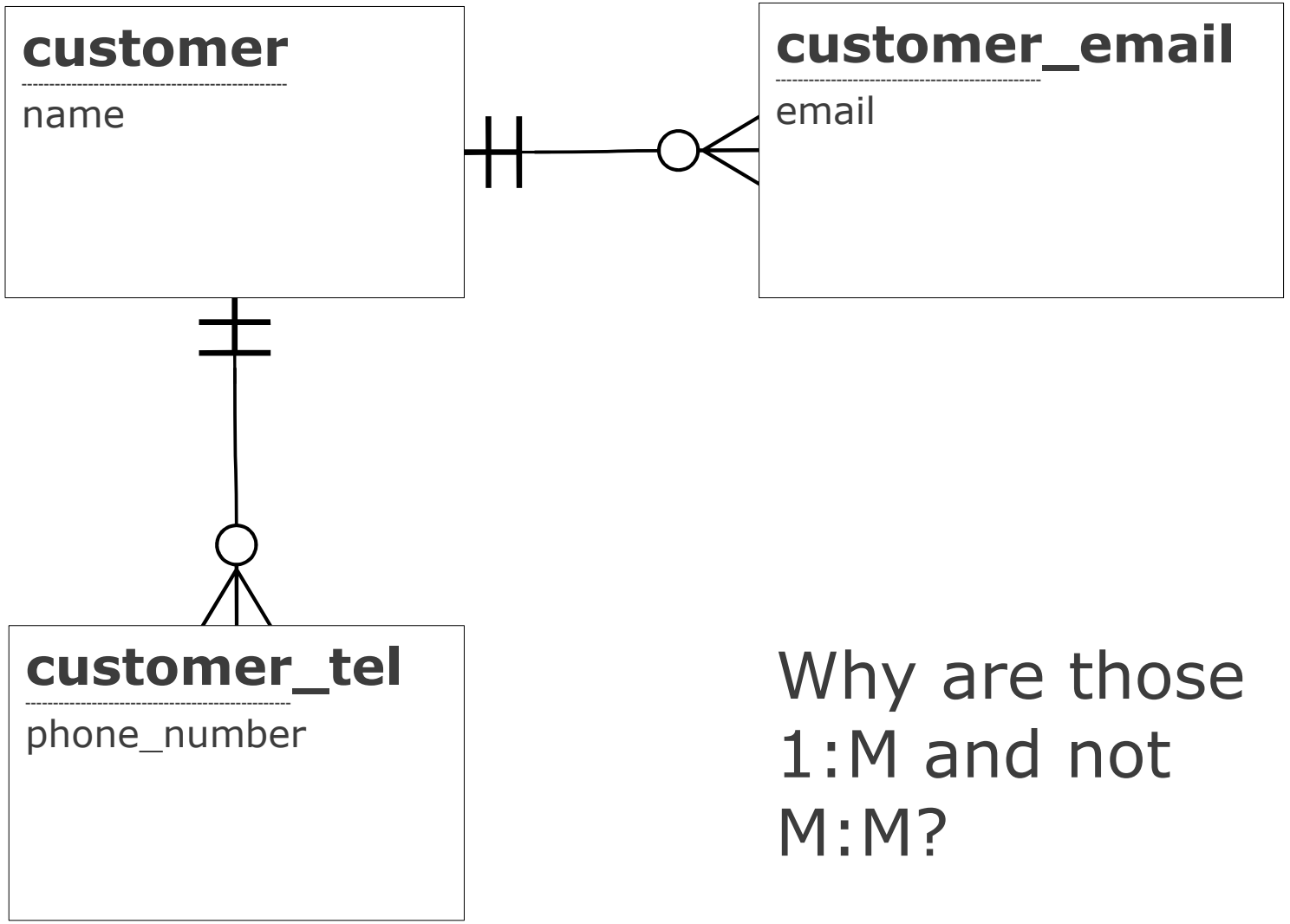
Multivalued Attributes

Problem:

Multivalued attributes may be ok in ER, but definitely not in a relational database.

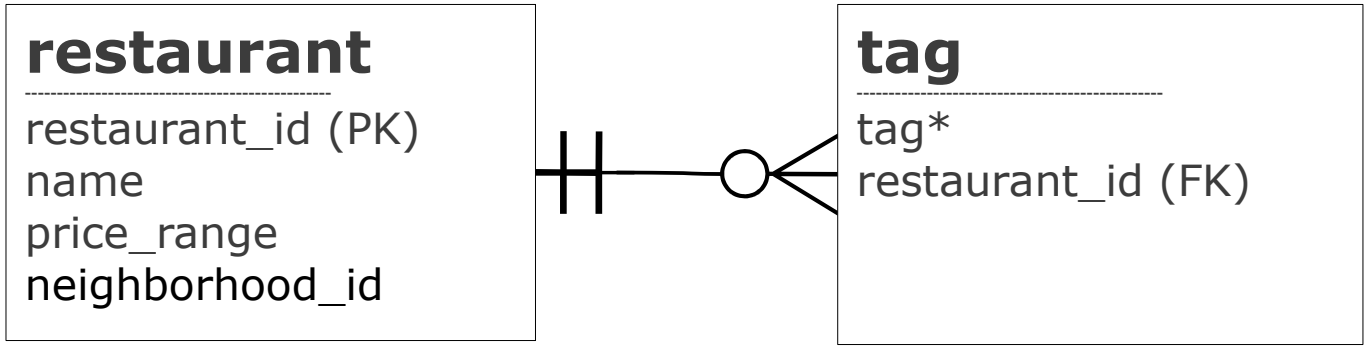
Solution:

Treat multivalued attributes as simple entities.

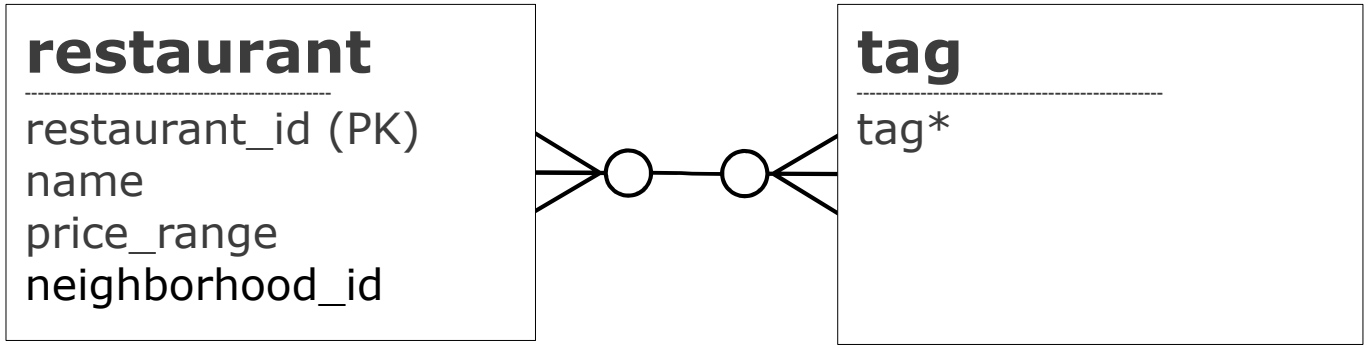


Why are those 1:M and not M:M?

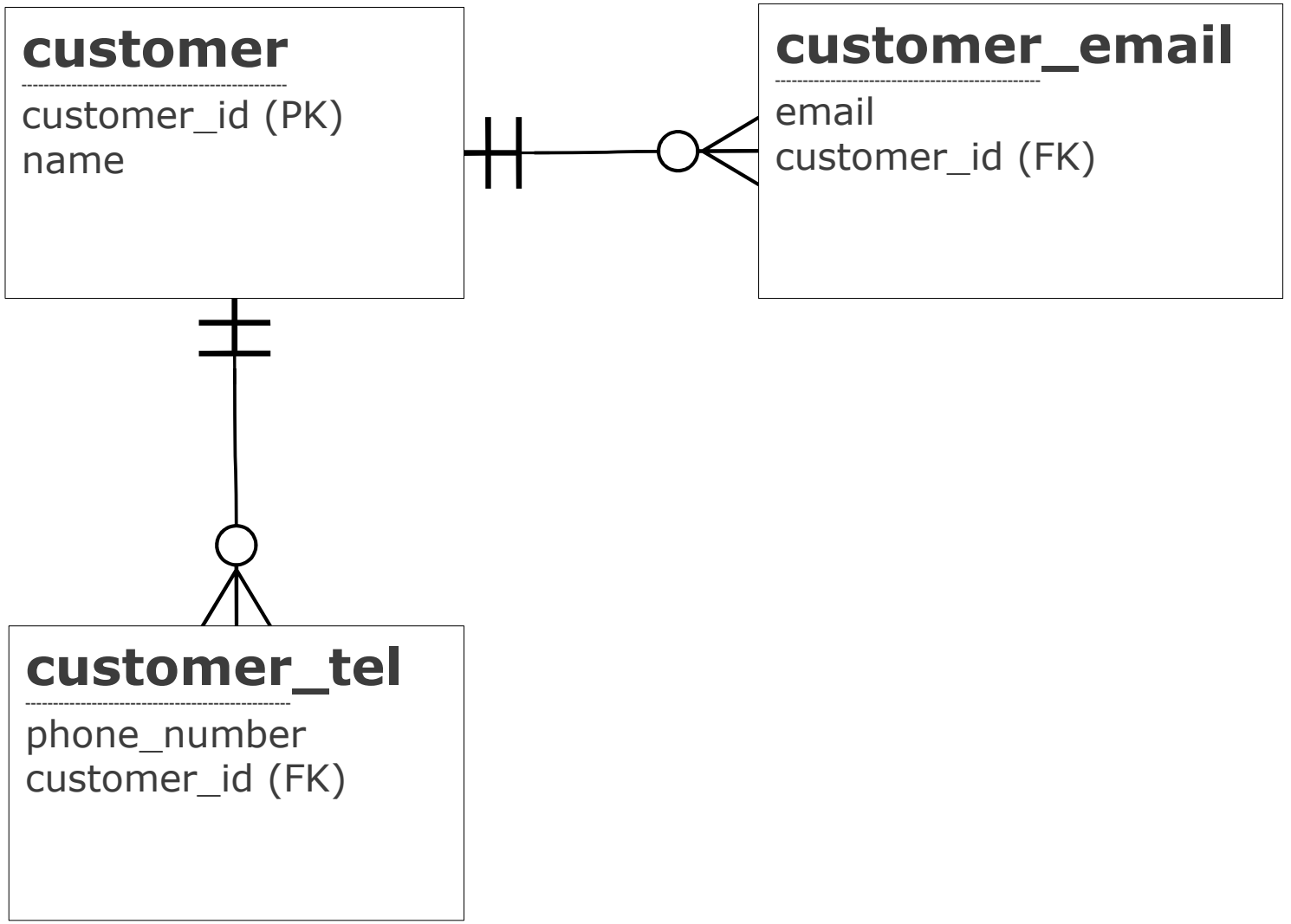
```
create table customer_email (
   email varchar(100),
   customer_id integer not null,
   position integer,
   primary key
     (customer_id, email),
   foreign key (customer_id)
    references customer(list_id)
);
```


restaurant
restaurant_id (PK)
name
price_range
neighborhood_id
tag
tag*
restaurant_id (FK)

or


restaurant
restaurant_id (PK)
name
price_range
neighborhood_id
tag
tag*

?


customer
customer_id (PK)
name
customer_email
email
customer_id (FK)
customer_tel
phone_number
customer_id (FK)

```
create table customer_email (
   email varchar(100),
   customer_id integer not null,
   primary key
     (customer_id, email),
   foreign key (customer_id)
     references customer(list_id)
);
```

Are we missing anything?

```
create table customer_email (
   email varchar(100),
   customer_id integer not null,
   position integer,
   primary key
     (customer_id, email),
   foreign key (customer_id)
    references customer(list_id)
);
```

CASE Tools

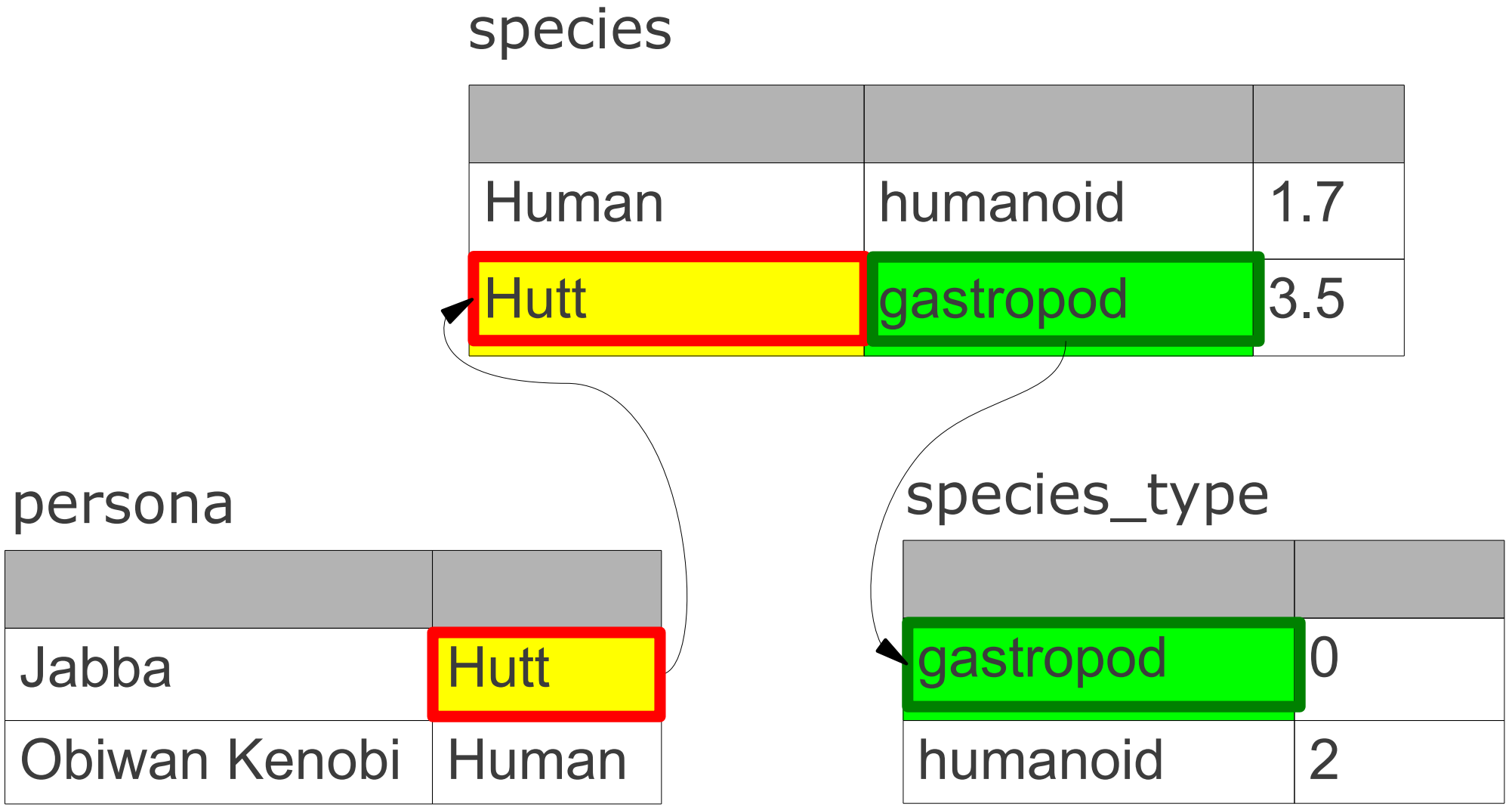
Allows designing in ER-like form and getting SQL generated automatically.

(Don’t this use in this class.)

Week 5

Queries Using Multiple Tables

Linking the Tables



Projection

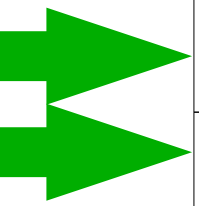
name	owner	species	sex	birth
Fluffy	Harold	cat	f	1993-02-04
Bluffy	Harold	dog	f	1989-05-13
Chirpy	Gwen	bird	f	1998-09-11

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

Multiple Tables

pet

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

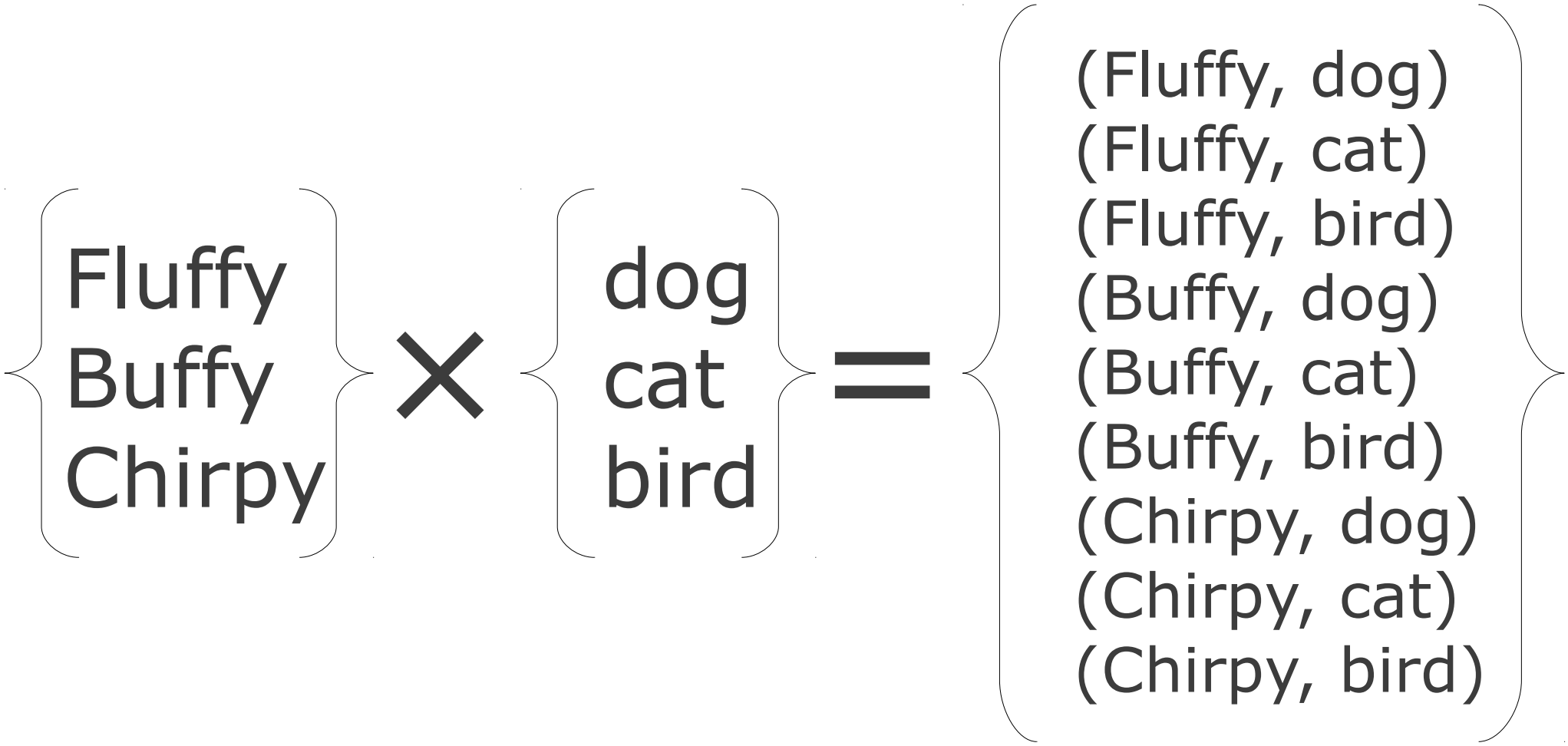
species

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

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»;
```

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

SQL-92 Inner Join

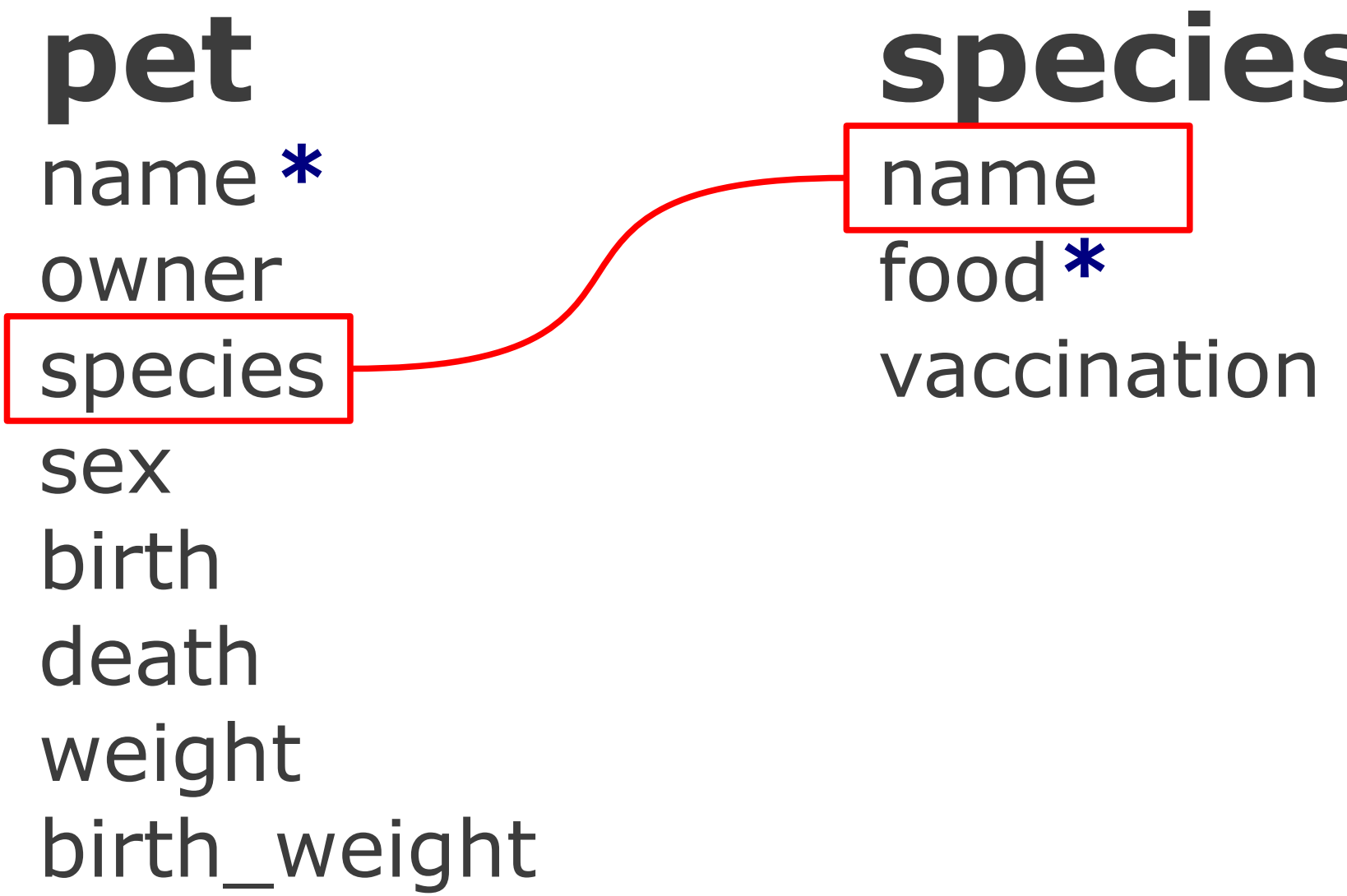
```
select ... from «table1»
join «table2» on «conditions»;
```

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

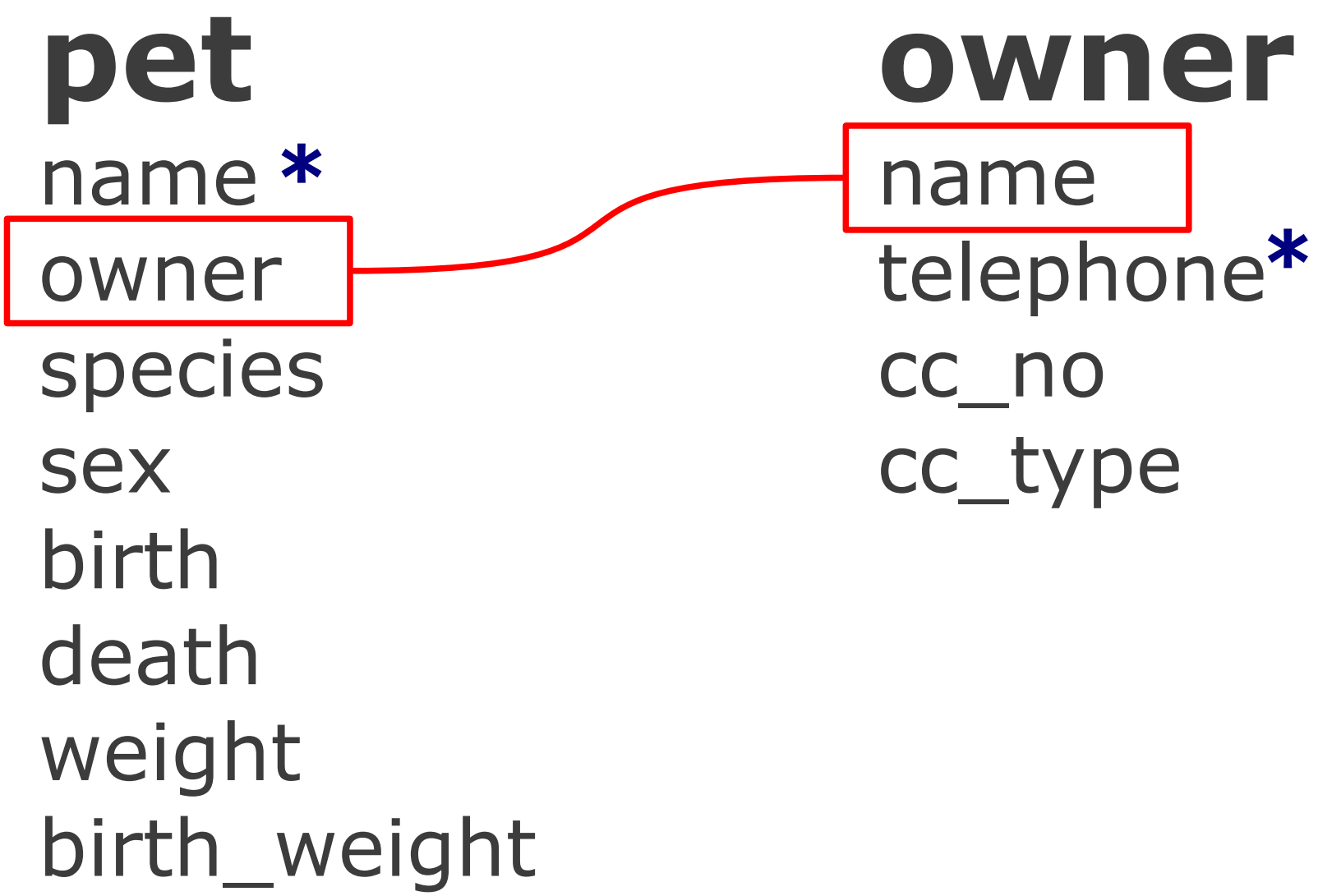
```
+----------+----------+
| name     | food     |
+----------+----------+
| Fluffy   | cat food |
| Claws    | cat food |
| Buffy    | dog food |
| Fang     | dog food |
| Bowser   | dog food |
| Chirpy   | seeds    |
| Whistler | seeds    |
| Slim     | mice     |
+----------+----------+
8 rows in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

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

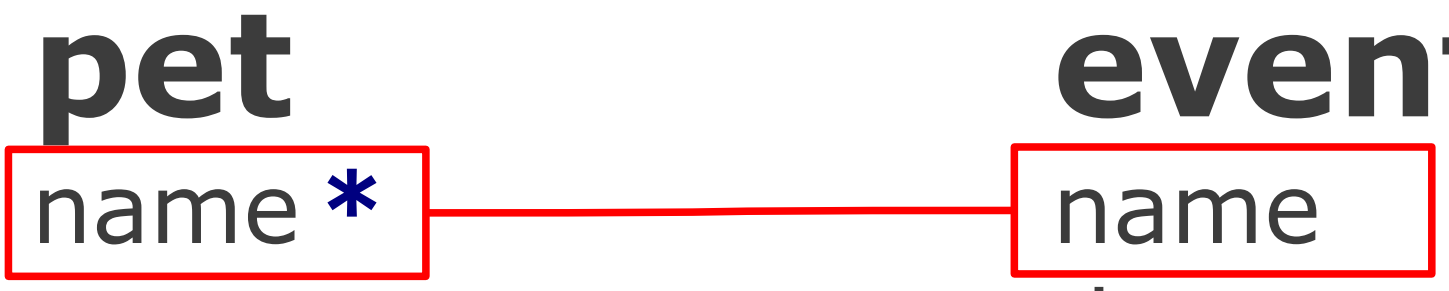
```
+----------+----------+
| name     | food     |
+----------+----------+
| Fluffy   | cat food |
| Fluffy   | dog food |
| Fluffy   | seeds    |
| Fluffy   | mice     |
| Claws    | cat food |
| Claws    | dog food |
| Claws    | seeds    |
| Claws    | mice     |
| Buffy    | cat food |
             ...
| 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;
```

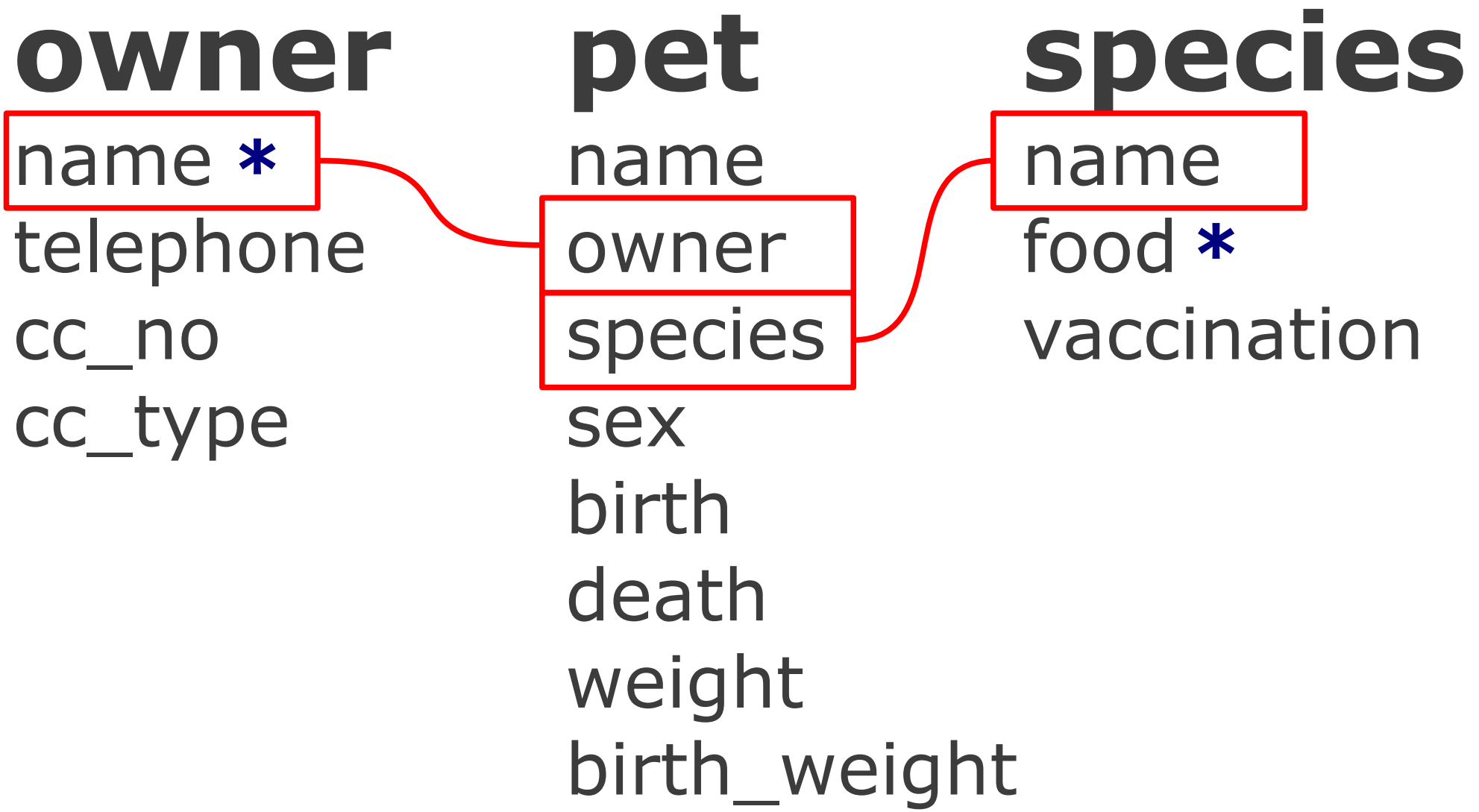
```
select p.name, s.food
from pet as p join species as s
on p.species = s.name;
```

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»);
```

```
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»;
```

---

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»;
```

---

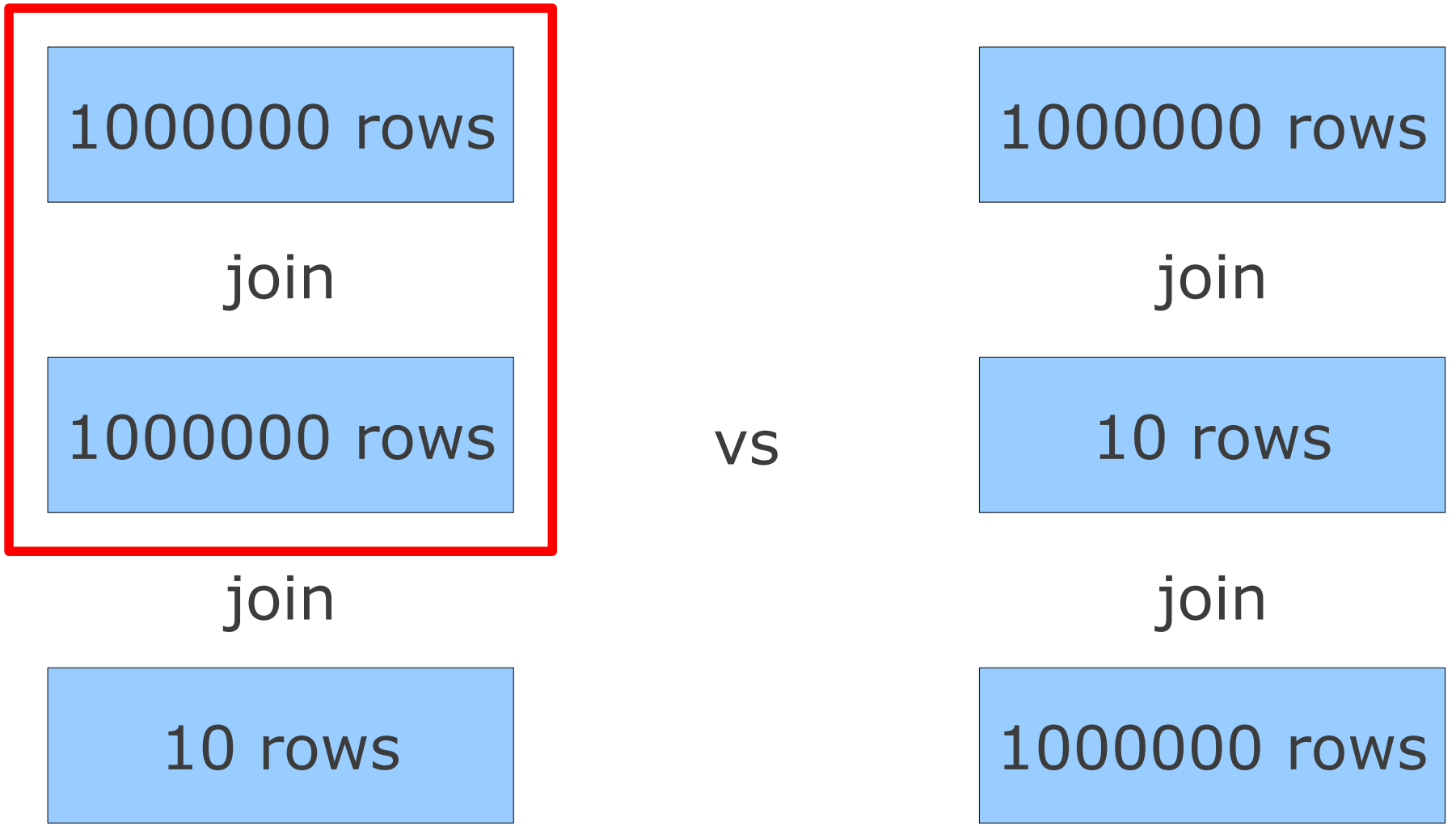
avoid

```
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

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

left 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)

Back to 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

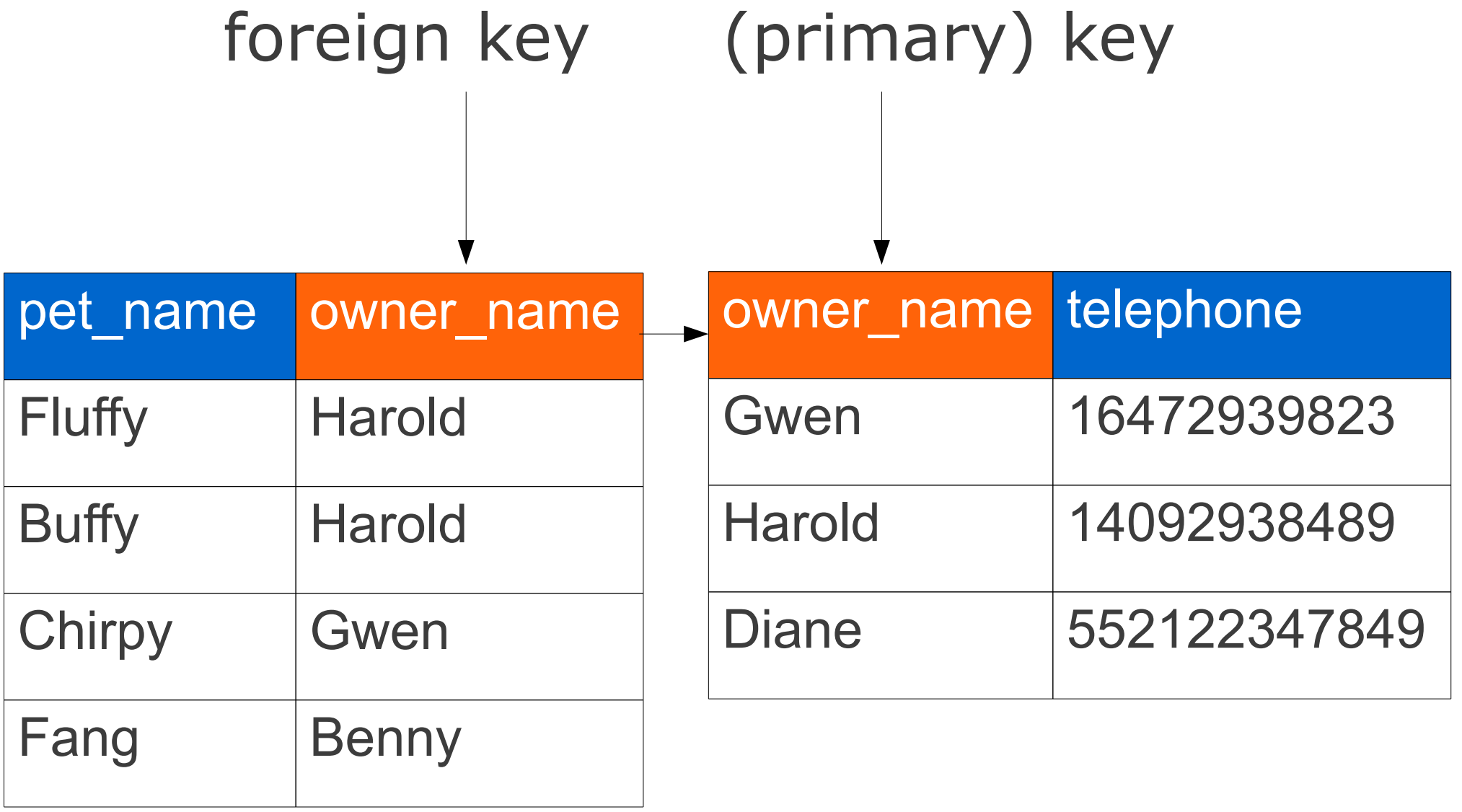
```
select
  persona.name, species.species,
  w1.percent_water, w2.percent_water
from persona
join species on
  persona.species=species.species
join world as w1 on
  persona.homeworld=w1.world_name
join world as w2 on
  species.homeworld=w2.world_name;
```

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;
```


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.

Diveshop

What is the winter temperature in C° at the place where the customer from Michigan (“MI”) booked his vacation?

IN

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

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

Subqueries

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

```
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");
```

With Comparison

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

```
select name from pet
where weight < (select
avg(weight) from pet);
```

"Uncorrelated"

```
select name from pet
where weight < (select
avg(weight) from pet);
```

Inner query is independent from the outer query

"Correlated"

```
select name from pet as p1
where weight < (select
avg(weight) from pet as p2
where p2.species=p1.species);
```

The result of the inner query depends on where we are in the outer query!
(The inner query will need to be evaluated one per row of the outer query.)

"Correlated"

```
select name from pet as p
where weight < (select
average(weight) from pet);
```

Inner query is independent from the outer query

"Correlated"

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

Inner query is independent from the outer query

Questions?