Animal Shelter

This is a complicated design that offers a lot of opportunity to apply JOINs.

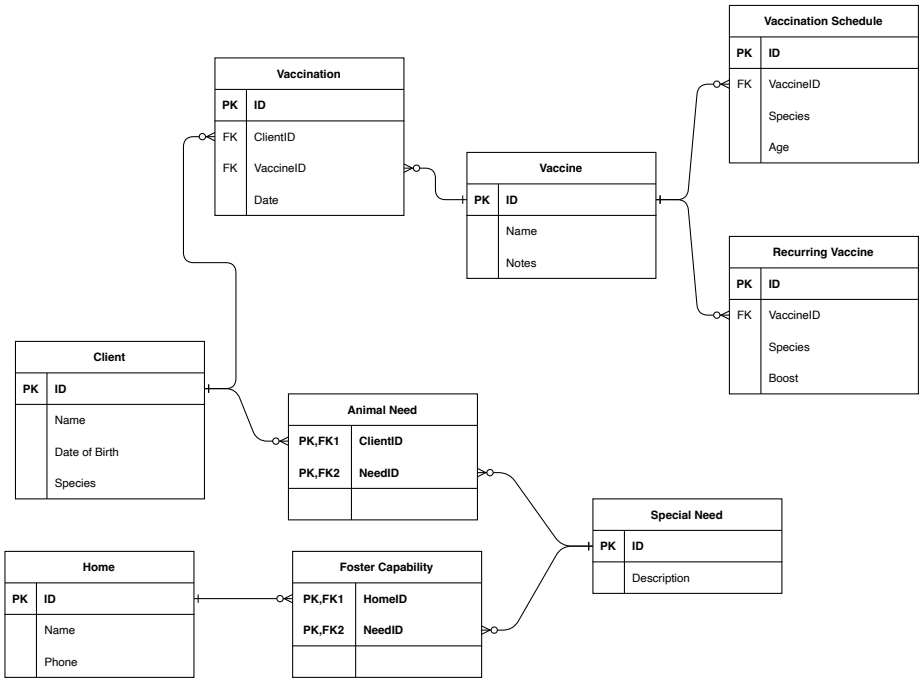
About the organization

We serve our animals first, which is why we refer to the animals as the "client."

We make sure our clients are fully vaccinated according to guidelines before they are adopted.

We understand our clients may require special environments and/or support. We strive to find homes for all our clients, and qualify potential adopters for their capacity to provide the right environment for our clients.

The Database



Link Tables

There are a number of link tables that implement many-to-many relationships.

Each one of these requires *two* joins to resolve: basically one to resolve each foreign key.

An effective way to approach these is to write and debug one query that resolves half of the relation, returning a table that has all the data you need from one of the tables, and leaves the foreign key for the other table from the link table.

Then do a join on that, resolving the remaining foreign key.

Queries are nested in parentheses. **They need a temporary name, even if that name is not needed to resolve the columns.**

Client-to-NeedID

Here is a query that resolves half of the AnimalNeed table:

```
SELECT client.name, animalneed.needid
FROM client
INNER JOIN animalneed
ON client.id = animalneed.clientid;
```

Note that this query fully qualifies column names. This is not always necessary, but will make the next step clearer.

This returns:

```
   name  | needid
--------+--------
 Krypto |      2
 Vivi   |      1
 Io     |      1
 Io     |      2
 Pepper |      3
```

Completing the link table: nested join

The above table needs to have the "need description" pulled from the specialneed table. Note how the above SELECT is embedded in parentheses and given the temporary name "nested."

```
SELECT nested.name, specialneed.description
FROM specialneed
INNER JOIN (
    SELECT client.name, animalneed.needid
    FROM client
    INNER JOIN animalneed
    ON client.id = animalneed.clientid
) nested
ON specialneed.id = nested.needid;
```

returns:

```
  name  |    description
--------+--------------------
 Krypto | no children
 Vivi   | older
 Io     | older
 Io     | no children
 Pepper | extra medical care
```

Clarifying the intermediate role

This is the same as above, but renames the columns using AS so they are easier to trace:

```
SELECT temporary_table.tmp_name AS "NAME", specialneed.description
FROM specialneed
INNER JOIN (
    SELECT client.name AS tmp_name, animalneed.needid AS tmp_needid
```

```
    FROM client
    INNER JOIN animalneed
    ON client.id = animalneed.clientid
) temporary_table
ON specialneed.id = temporary_table.tmp_needid;
```

```
  NAME  |    description
--------+--------------------
 Krypto | no children
 Vivi   | older
 Io     | older
 Io     | no children
 Pepper | extra medical care
```

(Notice the uppercase column names made using double quotes.)

Queries

- what is the vaccination schedule for a species?
- what is the vaccines have the clients had?
- what is the series vaccination schedule for clients?
- survey of homes, listing capabilities if available

Complicated: - timing of first three rounds of boosters (use CROSS JOIN and function generate_series(first, last)) - recurring schedule - what are upcoming vaccines? - booster schedule for client - match clients to homes

Suggestions for making JOINs

- ALWAYS START WITH THE SELECT
- identifies what columns you are interested in
- ask: where do these columns come from?
- -> gives you TABLEs.
- first table: SELECT FROM
- each additional table: SELECT and JOIN: one table is new; one is nested
  - name the nested table
- for a JOIN: ON is foreign key to primary key
- check the clause order: (SELECT FROM JOIN ON WHERE GROUP ORDER)

Notes

Medications

Medications and vaccinations are very similar, but may need different treatment. Feel free to expand the design to track medications.

INTERVAL

The SQL INTERVAL datatype is new in this course. Compare the INSERT statements:

```
INSERT INTO vaccinationschedule(id, vaccineid, species, age)
VALUES
(1, 1, 'dog', '6 weeks'),
(2, 2, 'dog', '6 weeks'),
(6, 3, 'dog', '12 months'),
(1010, 1003, 'cat', '12 weeks'),
(1011, 1003, 'cat', '1 year');
```

with the tabular output for these rows:

```
shelter=> SELECT * FROM vaccinationschedule ;
  id  | vaccineid | species |   age
------+-----------+---------+----------
    1 |         1 | dog     | 42 days
    2 |         2 | dog     | 42 days
    6 |         3 | dog     | 1 year
 1010 |      1003 | cat     | 84 days
 1011 |      1003 | cat     | 1 year
```

"Weeks" and "months" are converted into "days" and "years." The database uses this representation to make date calculations easier. If you export the database it will use the days/years notation.

This storage pattern means there is a small amount of data loss when the INSERT is processed. In the hand-written INSERTs, it is nice to use the same measures as the source document: it makes it easier to check that the vaccination schedule was entered correctly. Depending on how this table will be used and updated, you may want to be extra-sure to keep the original SQL INSERTs, or you may even want to add another attribute for "human-readable schedule" that matches the recommendation units exactly.

Different schedules

Vaccination schedules are often not "one size fits all," but are adapted for circumstance. Some factors might include:

- shelter animals have more social exposure than animals in a home

- animals in a multi-pet home have more social exposure than single pets
- outdoor cats encounter different pathogens than cats that stay indoors
- precise age or vaccination history may not be known for stray animals

References

AKC Puppy Shots Complete Guide

American Animal Hospital Association, American Association of Feline Practitioners, and International Society of Feline Medicine