Worksheet 3 - SQL joins

Jo Hardin

2023-01-10

Your Name: _____

Names of people you worked with: _____

- Introduce yourself. What is your favorite extra-curricular/club activity at college?
- What is your major? Why? What do you love about your major?

Task:[1]

Consider the following two tables. The first table, `dogs`, lists different dogs as observational units. The owners table, `users`, contains information on each dog owner.

```
CREATE TABLE dogs (
    dogid integer,
    ownerid integer,
    name varchar,
    breed varchar,
    age integer,
    PRIMARY KEY (dogid),
    FOREIGN KEY (ownerid) REFERENCES users(userid)
);
```

```
CREATE TABLE users (
    userid integer,
    name varchar,
    age integer,
    PRIMARY KEY (userid)
);
```

[1] Questions come from CS 186 at UC Berkeley, https://cs186berkeley.net/notes/note2/#practice-questions.

The `users` own dogs. The `ownerid` column in the `dogs` table corresponds to the `userid` column of the `users` table (`ownerid` is a foreign key that references the `users` table).

1. Write a query that lists the names of all the dogs that "Josh Hug" owns.
2. Write a query that finds the name of the user and the number of dogs that user owns for the user that owns the most dogs in the database. Assume that there are no ties (i.e., this query should only return 1 user). Users may share the same name.
3. Now write the same query again, but you can no longer assume that there are no ties.

Solution

1. Write a query that lists the names of all the dogs that "Josh Hug" owns.

```
SELECT dogs.name
FROM dogs INNER JOIN users ON dogs.ownerid = users.userid
WHERE users.name="Josh Hug";
```

We now need information from both tables (the dog name is only in the `dogs` table and the owner name is only in the `users` table). The join condition is `dogs.ownerid=users.userid` because we only want to get rows with the dog and its owner in it. Finally we add the predicate to the `WHERE` clause to only get Josh's dogs.

2. Write a query that finds the name of the user and the number of dogs that user owns for the user that owns the most dogs in the database. Assume that there are no ties (i.e., this query should only return 1 user). Users may share the same name.

```
SELECT users.name, COUNT(*)
FROM users INNER JOIN dogs on users.userid = dogs.ownerid
GROUP BY users.userid, users.name
ORDER BY COUNT(*) DESC
LIMIT 1;
```

We can use an `ORDER BY` combined with a `LIMIT` to select the first n most rows (with n being 1 in this case). We `GROUP BY` the name because we want our groups to be all about one user. We have to include `userid` in the `GROUP BY`, because users may share the same name.

3. Now write the same query again, but you can no longer assume that there are no ties.

```
SELECT users.name, COUNT(*)
FROM users INNER JOIN dogs ON users.userid = dogs.ownerid
GROUP BY users.userid, users.name
HAVING COUNT(*) >= all(
    SELECT COUNT(*)
    FROM dogs
    GROUP BY ownerid
);
```

The inner query gets the number of dogs owned by each owner. The owner(s) with the max number of dogs must have a number of dogs that is >= all these rows in order to be the max. We put this condition in the `HAVING` rather than the `WHERE` clause because it pertains to the groups not the individual rows (that is, `HAVING` filters on the results set).