B561 Assignment 1

Relational Databases and Basic SQL queries

Due: Wednesday January 22, 2020 by 11:45 pm

The goal of this assignment is to become familiar with the PostgreSQL system, to create a relational database, and to write and evaluate some simple SQL statements and queries in that system.

You need to upload a single file with name `assignment1.sql` file to Canvas which contains the necessary SQL statements that solve the problems in this assignment. The `assignment1.sql` file must be so that the AI's can run it in their PostgreSQL shell. (We will post a sample .sql file that gives the template for your `ssignment1.sql` file.)

Before you can solve this assignment, you will need download PostgreSQL (version 10) and install it on your computer.

Consider the following relation schemas for a database that maintains sailors, boats, and reservations of boats by sailors.

```
Sailor(sid integer, sname text, rating integer)
Boat(bid integer, bname text, color text)
Reserves(sid integer, bid integer, day text)
```

You should assume that `sid` in `Reserves` is a foreign key that references the primary key `sid` in `Sailor`, and that `bid` in `Reserves` is a foreign key that references the primary key `bid` in `Boat`.

Note the files `sailor.sql`, `boat.sql`, and `reserves.sql` that contain the relation instances for the `Sailor`, `Boat`, and `Reserves` relations that are supplied with this assignment.

1 Database creation and impact of constraints on INSERT and DELETE statements.

1. Create a database with name `assignment1` in PostgreSQL that stores these relations. Make sure to specify primary and foreign keys. Then write SQL queries that return each of the relation instances `Sailor`, `Boat`, and `Reserves`.

2. Provide 6 examples that illustrate how the presence or absence of primary and foreign keys affects insert and deletes in these relations. To solve this problem, you will need to experiment with the `Sailor`, `Boat`, and `Reserves` relation schemas and instances. For example, you should consider altering primary keys and foreign key constraints and then consider

various sequences of insert and delete operations. You may also need to change some of the relation instances. Certain inserts and deletes should succeed but other should create error conditions. (Consider the lecture notes about keys, foreign keys, and inserts and deletes as a guide to solve this problem.)

2 Formulating queries in SQL

Write SQL statements for the following queries. For this assignment, make sure to always use tuple variables in your SQL statements. For example, in formulating the query "Find the name of each boat" you should write the query

```
SELECT   b.bname
FROM     Boat b
```

instead of

```
SELECT   bname
FROM     Boat
```

Make sure that each of your queries returns a set but not a bag. In other words, make appropriate use of the `DISTINCT` clause where necessary.

1. Find the `sid` and age of each sailor.
2. Find the `sid`, name, and rating of each sailor whose rating is in the range $[2, 11]$ but not in the range $[8, 10]$.
3. Find the `bid`, name, and color of each non-red boat that was reserved by some sailor whose rating is more than 7.
4. Find the `bid` and name of each boat that was reserved by a sailor on a weekend day but that was not reserved by a sailor on a Tuesday.
5. Find the `sid` of each sailor who reserved both a red boat and a green boat.
6. Find the `sid` and name of each each sailor who reserved at least two different boats. (You should write this query without using the `COUNT` aggregate function.)
7. Find the pairs of `sid`s (s_1, s_2) of different sailors who both reserved a same boat.
8. Find the `sid` of each sailor who did not reserve any boats on a Monday or on a Tuesday.
9. Find the pairs (s, b) such that the sailor with `sid` s reserved the boat with `bid` b, provided that the sailor s has a rating greater than 6 and the color of boat b is not red.

10. Find the `bid` of each boat that where reserved by just one sailor. (You should write this query without using the `COUNT` aggregate function.)

11. Find the `sid` of each sailor who reserved fewer than 3 boats. (You should write this query without using the `COUNT` aggregate function.)

3 Formulating queries in Predicate Logic

For each query in Section 2, specify an expression in the Predicate Logic (also called Relational Calculus). This will illustrate how closely this logic is associated with SQL.

To learn more about this logic, you can consult the book *Database Management Systems* by Ramakrishnan and Gehrke which you can find online.

You should upload your answers for these problems in a separate pdf file named `Assignment1.pdf`.

Assume that associated with each of the relation `Sailor`, `Boat`, and `Reserves`, there is a predicate as follows:

Relation	Predicate
`Sailor`	$\texttt{Sailor}(x, y, z)$
`Boat`	$\texttt{Boat}(x, y, z)$
`Sailor`	$\texttt{Sailor}(x, y, z)$

As an example consider the query "Find the `bid` and name of each boat that was reserved by fewer than 2 sailors." This query can be formulated in the Predicate Logic with *domain variables* as follows[1]:

$$\{(b, n) \mid \exists c\, \texttt{Boat}(b, n, c) \wedge \neg(\exists s_1 \exists d_1 \exists s_2 \exists d_2(\texttt{Reserves}(s_1, b, d_1) \wedge \texttt{Reserves}(s_2, b, d_2) \wedge s_1 \neq s_2)\}.$$

And alternative formulation using a Predicate Logic with *tuple variables* is as follows and is more inline with the SQL formulation:[2]:

$$\{(\mathbf{b}.\texttt{bid}, \mathbf{b}.\texttt{bname}) \mid \texttt{Boat}(\mathbf{b}) \wedge \\ \neg(\exists \mathbf{r}_1 \exists \mathbf{r}_2(\texttt{Reserves}(\mathbf{r}_1) \wedge \texttt{Reserves}(\mathbf{r}_2) \wedge \mathbf{r}_1.\texttt{bid} = \mathbf{b}.\texttt{bid} \wedge \mathbf{r}_2.\texttt{bid} = \mathbf{b}.\texttt{bid} \wedge \mathbf{r}_1.\texttt{sid} \neq \mathbf{r}_2.\texttt{sid})\}.$$

[1] This logic is also called the Domain Relational Calculus.
[2] This logic is also called the Tuple Relational Calculus.