Fall 2024 B561 Assignment 1
Relational Databases, Expressing Constraints and Queries in SQL, and in Tuple Relational Calculus (TRC)

Chenghong Wang

Released: September 10, 2024
Due: September 23, 2024 by 11:50pm

1 Introduction

The goals for this assignment are to

1. become familiar with the PostgreSQL system
2. create a relational database and populate it with data;
3. examine the side-effects on the state of a database by inserts and deletes in the presence or absence of primary and foreign key constraints;
4. formulate some queries in SQL and evaluate them in PostgreSQL; and
5. translate TRC queries to SQL and formulate queries and constraints in TRC

To turn in your assignment, you will need to upload to Canvas a file with name `assignment1.sql` which contains the necessary SQL statements that solve the graded problems in this assignment.

The assignment1.sql file must be such that the AI's can run it in their PostgreSQL environment. In addition, you will need to upload a separate assignment1.txt file that contains the results of running your queries. We have posted the exact requirements and an example for uploading your solution files. (See the module Instructions for turning in assignments.) Finally, you will need to upload an assignment1.pdf file that contains the solutions for problems related to TRC.

Note:
1. To solve this assignment, you will need to download and install PostgreSQL (version 12 or higher) on your computer.

2. To solve problems related to TRC, follow the syntax and semantics described in the lecture slides Tuple Relational Calculus and SQL.
3. For a good way to learn about Latex, look at
https://www.overleaf.com/learn/latex/Free_online_introduction_to_LaTeX_(part_1).

For the problems in this assignment we will use the following database schema:[1]

```
Student(sid, sname, homeCity)
Department(deptName, mainOffice)
Major(major)
employedBy(sid, deptName, salary)
departmentLocation(deptName, building)
studentMajor(sid, major)
hasFriend(sid1, sid2)
```

In this database we maintain a set of students (`Student`), a set of departments (`Department`), and a set of (major) majors (`Major`). The `sname` attribute in `Student` is the name of the student. The `homeCity` attribute in `Student` specifies the home city of the student. The `deptName` attribute in `Department` is the name of the department. The `mainOffice` attribute in `Department` is the name of the building where the department has its main office. The `major` attribute in `Major` is the name of a (major) major.

A student can be employed by at most one department. This information is maintained in the `employedBy` relation. (We permit that a student is not employed by any department and that a department may not employ any students.) The `salary` attribute in `employedBy` specifies the salary made by the student.

The `building` attribute in `departmentLocation` indicates a building in which the department is located. (A department may be located in multiple buildings.)

A student can have multiple majors. This information is maintained in the `studentMajor` relation. A major can be the major of multiple students. (A student may not have any majors, and a major may have no students with that major.)

A pair (s_1, s_2) in `hasFriend` indicates that student s_1 considers student s_2 as a friend. It is possible that s_1 considers s_2 as a friend, but not necessarily the other way around. In other words, the `hasFriend` relation can not be assumed to be symmetric.

The domain for the attributes `sid`, `salary`, `sid1`, and `sid2` is `integer`. The domain for all other attributes is `text`.

We assume the following foreign key constraints:

[1]The primary key, which may consist of one or more attributes, of each of these relations is underlined.

- `sid` is a foreign key in `employedBy` referencing the primary key `sid` in `Student`;
- `deptName` is a foreign key in `employedBy` referencing the primary key `deptName` in `Department`;
- `deptName` is a foreign key in `departmentLocation` referencing the primary key `deptName` in `Department`;
- `sid` is a foreign key in `studentMajor` referencing the primary key `sid` in `Student`;
- `major` is a foreign key in `studentMajor` referencing the primary key `major` in `Major`;
- `sid1` is a foreign key in `hasFriend` referencing the primary key `sid` in `Student`; and
- `sid2` is a foreign key in `hasFriend` referencing the primary key `sid` in `Student`;

The data for the assignment is included in the `schema.sql`.

Remark 1 *We will typically use the primary key of an object to specify that object. This should not cause any confusion since an object can be referenced uniquely with its primary key value. E.g., we will write '...* student *s ... ' rather than '...* the student with sid *s ...'. When posing a query, we may write '*Find each student who *...' instead of '*Find the sid of each student who *...'. The expected answer for such a query should be the set of sids of students who meet the criteria of the query.*

2 Database creation and impact of constraints on insert and delete statements.

Create a database in PostgreSQL that stores the data provided in the `schema.sql` file. Make sure to specify primary and foreign keys.

1. Provide 4 conceptually different examples that illustrate how the presence or absence of primary and foreign keys affect inserts or deletes in these relations. To solve this problem, you will need to experiment with the relation schemas and instances for this assignment. For example, you should consider altering primary keys and foreign key constraints and then consider various sequences of insert and delete operations. You may need to change some of the relation instances to observe the desired effects. Certain inserts and deletes should succeed but others should generate error conditions. (Consider the lecture notes about keys, foreign keys, and inserts and deletes as a guide to solve this problem.) [5 Points]

3 Expressing queries in SQL

For this assignment, you are required to use tuple variables in your SQL statements. For example, in formulating the query "Find the sid and sname of each student who lives in Bloomington" you should write the query

```
SELECT   s.sid, s.sname
FROM     Student s
WHERE    s.homeCity = 'Bloomington'
```

rather than

```
SELECT   sid, sname
FROM     Student
WHERE    homeCity = 'Bloomington'
```

Write SQL statements for the following queries. 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.

You can **not** use the SQL JOIN operations or SQL aggregate functions such as `COUNT`, `SUM`, `MAX`, `MIN`, etc in your solutions.

2. For each department, list its name along with the highest salary made by students who work for it. [5 Points]

3. Find the sid and sname of each student s who is employed by a department d and who has a salary that is strictly higher than the salary of each of his or her friends who is employed by that department d. (Student should only be considered if indeed he or she has a friend who is employed by department d.) [5 Points]

4. Find the deptName of each department that not only employs students whose home city is Indianapolis. (In other words, there exists at least one student who is employed by such a department whose home city is not Indianapolis.) [5 Points]

5. Find the sid, sname of each student who (a) has home city Bloomington, (b) works for a department where he or she earns a salary that is higher than 20000, and (c) has at least one friend. [5 Points]

6. Find the pairs (d_1, d_2) of names of different departments whose main offices are located in the same building. [5 Points]

7. Find the sid and sname of each student whose home city is different than those of his or her friends. [5 Points]

8. Find each major that is the major of at most 2 students. [5 Points]

9. Find the sid, sname, and salary of each student who has at least two friends such that these friends have a common major but provided that it is not the 'Mathematics' major. [5 Points]

4 Translating TRC queries to SQL

Consider the following queries formulated in TRC. Translate each of these queries to an equivalent SQL query. This underscores the close correspondence between TRC and SQL. The SQL queries should be included in the assignment1.sql file.

10.

$$\{(s_1.sid, s_1.sname) \mid Student(s_1) \wedge \exists d \in Department,\ w \in employedBy\ (d.deptName = w.deptName \wedge s_1.sid = w.sid \wedge d.mainOffice = LuddyHall \wedge \exists s_2 \in Student\ (hasFriend(s_1.sid, s_2.sid) \wedge s_2.homeCity \neq Bloomington))\}.$$

[5 Points]

11.

$$\{s_1.sid \mid Student(s_1) \wedge \forall s_2 \in Student\,(hasFriend(s_1.sid, s_2.sid) \rightarrow \exists sm_1 \in studentMajor,\ sm_2 \in studentMajor\,(sm_1.sid = s_1.sid \wedge sm_2.sid = s_2.sid \wedge sm_1.major = sm_2.major \wedge sm_1.sid \neq sm_2.sid))\}.$$

[5 Points]

12.

$$\{(s_1.sid, s_2.sid) \mid Student(s_1) \wedge Student(s_2) \wedge s_1.sid \leq s_2.sid \wedge \forall f_1 \in hasFriend\,(s_1.sid = f_1.sid_1 \rightarrow \exists f_2 \in hasFriend\,(f_2.sid_1 = s_2.sid \wedge f_1.sid_2 = f_2.sid_2))\}.$$

[5 Points]

13.

$$\{m.major \mid Major(m) \wedge \neg(\exists s \in Student\ \exists sm \in studentMajor(s.sid = sm.sid \wedge sm.major = m.major \wedge s.homeCity = \text{`Bloomington'})\}.$$

[5 Points]

5 Formulating Queries in the Tuple Relational Calculus

The TRC solutions of these problems should be included in the `assignment1.pdf` file. You need not write SQL queries for this section.

14.

(a) Find each pair (d, m) where d is the name of a department and m is a major of a student who is employed by that department and who earns a salary of at least 20000. [2.5 Points]

(b) Find each pair (s_1, s_2) of sids of different students who have the same (set of) friends who work for the CS department. [2.5 Points]

15.

(a) Find each major for which there exists a student with that major and who does not only have friends who also have that major. [2.5 Points]

(b) Find the sid and sname of each student whose home city is different than those of his or her friends. [2.5 Points]

16.

(a) Find the sid, sname, and salary of each student who has at least two friends such that these friends have a common major but provided that it is not the 'Mathematics' major. [2.5 Points]

(b) For each department, list its name along with the highest salary made by students who are employed by it. [2.5 Points]

17.

Find the sid, sname of each student who (a) has home city Bloomington, (b) works for a department where he or she earns a salary that is higher than 20000, and (c) has at least one friend. [5 Points]

6 Formulating constraints in the safe Tuple Relational Calculus and as boolean SQL queries

Formulate the following constraints in safe TRC and as boolean SQL queries.

The TRC solutions of these problems should be included in the `assignment1.pdf` file and the SQL solutions should be included in the `assignment1-SQL.sql` file.

Here is an example of what is expected for your answers.

Example 1 *Consider the constraint "*Each major is the major of a student.*" In safe TRC, this constraint can be formulated as follows:*

$$\forall m\ Major(n) \rightarrow \exists sm\,(studentMajor(sm) \wedge sm.major = m.major)$$

Alternatively, the constraint can be formulated in safe TRC as

$$\neg\exists m(Major(m) \wedge \neg\exists sm(studentMajor(sm) \wedge sm.major = m.major)).$$

This constraint can be specified using the following boolean SQL query:

```
select not exists (select 1
                   from   Major m
                   where  not exists (select 1
                                      from   studentMajor sm
                                      where  sm.major = m.major));
```

18. Consider the constraint "*Some major has fewer than 2 students with that major.*"

    (a) Formulate this constraint in safe TRC. [2.5 points]

    (b) Formulate this constraint as a boolean SQL query. [2.5 points]

19. Consider the constraint "*Each student is employed by a department and has at least two majors.*"

    (a) Formulate this constraint in safe TRC. [2.5 points]

    (b) Formulate this constraint as a boolean SQL query. [2.5 points]

20. Consider the constraint "*Each student and his or her friends work for the same department.*"

    (a) Formulate this constraint in safe TRC. [2.5 points]

    (b) Formulate this constraint as a boolean SQL query. [2.5 points]