Fall 2023 B461
Assignment 5

November 3, 2023

This assignment tests the following concepts:

- Triggers
- Object-relational DBs

To turn in your assignment, you will need to upload to Canvas a SQL file with the name `assignment5.sql` which contains the solutions and a text file `assignment5.txt` that contains the results of running your queries. The `assignment5.sql` file must be so that the AIs can run it in their PostgreSQL environment. You should use the `assignment5-script.sql` file to construct the `assignment5.sql` file. (Note that the data to be used for this assignment is included in this file.)

1 Database schema and instances

For the problems in this assignment, we will use the following database schema:

```
Person(pid, pname, city)
Company(cname, headquarter)
Skill(skill)
worksFor(pid, cname, salary)
companyLocation(cname, city)
personSkill(pid, skill)
hasManager(eid, mid)
Knows(pid1, pid2)
```

In this database, we maintain a set of persons (`Person`), a set of companies (`Company`), and a set of (job) skills (`Skill`). The `pname` attribute in `Person` is the name of the person. The `city` attribute in `Person` specifies the city in which the person lives. The `cname` attribute in `Company` is the name of the company. The `headquarter` attribute in `Company` is the name of the city wherein the company has its headquarters. The `skill` attribute in `Skill` is the name of a (job) skill.

A person can work for at most one company. This information is maintained in the `worksFor` relation. (We permit that a person does not work for any company.) The `salary` attribute in `worksFor` specifies the salary made by the person.

The `city` attribute in `companyLocation` indicates a city in which the company is located. (Companies may be located in multiple cities.)

A person can have multiple job skills. This information is maintained in the `personSkill` relation. A job skill can be the job skill of multiple persons. (A person may not have any job skills, and a job skill may have no persons with that skill.)

A pair (`e`;`m`) in `hasManager` indicates that person `e` has person `m` as one of his or her managers. We permit that an employee has multiple managers and that a manager may manage multiple employees. (It is possible that an employee has no manager and that an employee is not a manager.) We further require that an employee and his or her managers must work for the same company.

The relation `Knows` maintains a set of pairs (`p1`; `p2`) where `p1` and `p2` are `pids` of persons. The pair (`p1`; `p2`) indicates that the person with `pid` `p1` knows the person with `pid` `p2`. We do not assume that the relation `Knows` is symmetric: it is possible that (`p1`; `p2`) is in the relation but that (`p2`; `p1`) is not.

The domain for the attributes `pid`, `pid1`, `pid2`, `salary`, `eid`, and `mid` is integer. The domain for all other attributes is text.

We assume the following foreign key constraints:

- `pid` is a foreign key in `worksFor` referencing the primary key `pid` in `Person`;
- `cname` is a foreign key in `worksFor` referencing the primary key `cname` in `Company`;
- `cname` is a foreign key in `companyLocation` referencing the primary key `cname` in `Company`;
- `pid` is a foreign key in `personSkill` referencing the primary key `pid` in `Person`;
- `skill` is a foreign key in `personSkill` referencing the primary key `skill` in `Skill`;
- `eid` is a foreign key in `hasManager` referencing the primary key `pid` in `Person`;
- `mid` is a foreign key in `hasManager` referencing the primary key `pid` in `Person`;
- `pid1` is a foreign key in `Knows` referencing the primary key `pid` in `Person`;
- `pid2` is a foreign key in `Knows` referencing the primary key `pid` in `Person`.

The file assignment5.sql contains the data supplied for this assignment.

2 Triggers

Formulate the following queries in SQL. You are allowed to use aggregate functions, views, temporary views, parameterized views, and user-defined functions.

1. Write a trigger to check for primary key constraint. The trigger should include a definition and a function. [10 Points]

2. Write a trigger to check for referential integrity constraint. The trigger should include a definition and a function. [10 Points]

3. Consider two relations R(A:integer,B:integer) and S(B:integer) and a view with the following definition:
   ```
   select distinct r.A
   from R r, S s
   where r.A > 10 and r.B = s.B;
   ```
   Suppose we want to maintain this view as a materialized view called V(A:integer) upon the insertion of tuples in R and in S. (You do not have to consider deletions in this question.)
   Define SQL insert triggers and their associated trigger functions on the relations R and S that implement this materialized view. Write your trigger functions in the language 'plpgsql.'
   Make sure that your trigger functions act in an incremental way and that no duplicates appear in the materialized view. [10 Points]

4. Consider applying the following constraint over the relation personSkill. "Each skill of a person who works for Apple should also be the skill of the person who works for Google". Write a trigger that maintains the constraint when inserting new pairs of (pid,skill) into the personSkill relation.(You can ignore the constraint restriction to hold upon the already existing previous records).

   - Consider adding the data records given in the data file.
   - Return the personSkill records across each of the newly added PIDs given in the data file.
   - Drop the records and retain the original data as provided in the data file. [10 Points]

5. Consider applying the following constraint over the relation knows. "Whenever a person moves from company A to company B, they should know all the managers working at the new company B." To meet this constraint, add the required entries in the Knows table. Test your trigger across the below updates:

   (a) 1005 moving from Google to Apple

   (b) 1012 moving from Apple to Google [10 Points]

3 Formulating Query in Object-Relational SQL

For the problems in the section, you will need to use the defined functions and predicates that are defined in the document.

Functions

- `set intersection(A,B)` : $A \cap B$
- `set difference(A,B)` : $A \setminus B$
- `add element(x,A)` : $x \cup A$
- `remove element(x,A)` : $A \setminus \{x\}$
- `make singleton(x)` : $\{x\}$
- `choose element(A)` : choose some element from A
- `bag union(A,B)` : the bag union of A and B
- `bag to set(A)` : coerce the bag A to the corresponding set

Predicates

- `is in(x,A)` : $x \in A$
- `is not in(x,A)` : $x \notin A$
- `is empty(A)` : $A =$
- `is not empty(A)` : $A \neq$
- `subset(A,B)` : $A \subseteq B$
- `superset(A,B)` : $A \supseteq B$
- `equal(A,B)` : $A = B$
- `overlap(A,B)` : $A \cap B \neq$
- `disjoint(A,B)` : $A \cap B =$

```
But before turning to the problems, we will introduce various object-
relational views defined over these relations in the schema:
```

1. The view companyHasEmployees(cname,employees) which as- sociates with each company, identified by a cname, the set of pids of persons who work for that company.

   ```
   create or replace view companyHasEmployees as
   select cname, array(select pid
   from worksfor w
   where w.cname = c.cname order by 1) as employees
   from company c order by 1;
   ```

2. The view cityHasCompanies(city,companies) which associates with each city the set of cnames of companies that are located in that city.

   ```
   create or replace view cityHasCompanies as
   select city, array_agg(cname order by 1) as companies
   from companyLocation
   group by city order by 1;
   ```

3. The view companyHasLocations(cname,locations) which asso- ciates with each company, identified by a cname, the set of cities in which that company is located.

```
create or replace view companyHasLocations as
select cname, array(select city
from companyLocation cc
where c.cname = cc.cname order by 1) as locations
from company c order by 1;
```

4. The view knowsPersons(pid,persons) which associates with each per- son, identified by a pid, the set of pids of persons he or she knows.

```
create or replace view knowsPersons as
select p.pid, array(select k.pid2
from knows k
where k.pid1 = p.pid order by pid2) as persons
from person p order by 1;
```

5. The view isKnownByPersons(pid,persons) which associates with each person, identified by a pid, the set of pids of persons who know that person. Observe that there may be persons who are not known by anyone.

```
create or replace view isKnownByPersons as
select distinct p.pid, array(select k.pid1
from knows k
where k.pid2 = p.pid) as persons
from person p order by 1;
```

6. The view personHasSkills(pid,skills) which associates with each person, identified by a pid, his or her set of job skills.

```
create or replace view personHasSkills as
select distinct p.pid, array(select s.skill
from personSkill s
where s.pid = p.pid order by 1) as skills
from person p order by 1;
```

7. The view skillOfPersons(skills,persons) which associates with each job skill the set of pids of persons who have that job skill.

```
create or replace view skillOfPersons as
select js.skill, array(select ps.pid
from personSkill ps
where ps.skill = js.skill order by pid) as persons
from jobSkill js order by skill;
```

```
In the problems in this section, you are asked to formulate queries in
object- relational SQL. You should use the set operations and set predi-
cates defined in the document SetOperationsAndPredicates.sql,
THE RELATIONS:
                Person
                Company
                Skill
                worksFor
AND THE VIEWS:
                companyHasEmployees
                cityHasCompanies
                companyHasLocations
                knowsPersons
                isKnownByPersons
                personHasSkills
                skillOfPersons
```

However, you are not permitted to use the Knows, companyLocation,and personSkill relations in the object relation SQL formulation of the queries. Observe that you actually don't need these relations since they are encapsulated in these views.

Before listing the queries that you are asked to formulate, we present some examples of queries that are formulated in object-relational SQL using the assumptions stated in the previous paragraph. Your solutions need to be in the style of these examples. The goals is to maximize the utilization of the functions and predicates defined in document `SetOperationsAndPredicates.sql`.

1. **Example 1:** Consider the query "Find the pid of each person who knows a person who has a salary greater than 55000."

   ```
   select distinct pk.pid
   from knowsPersons pk, worksfor w
   where is_in(w.pid, pk.persons) and w.salary > 55000
   order by 1;
   ```

   Note that the following formulation for this query is not allowed since it uses the relation `Knows` which is not permitted.

   ```
   select distinct k.pid1
   from knows k, worksfor w
   where k.pid2 = w.pid and w.salary > 55000;
   ```

2. **Example 2:** Consider the query "Find the pid and name of each person along with the set of his or her skills that are not among the skills of persons who work for 'Netflix'."

   ```
   select p.pid, p.pname, set_difference((select ps.skills
   from personHasSkills ps
   where ps.pid = p.pid),
   array(select
   unnest(ps.skills)
   from personHasSkills ps
   where is_in(ps.pid, (select employees
   from companyHasEmployees
   where cname = 'Netflix'))))
   from person p;
   ```

4 Object Relational Queries

6. Find the cname and headquarter of each company that employs at least two persons who have a common job skill. [10 Points]

7. Find the pid and name of each person p along with the set of pids of persons who (1) know p and (2) who have the AI skill but not the Networks skill. [10 Points]

8. Find the pid and name of each person who has all the skills of the combined set of job skills of the highest-paid persons who work for Google. [10 Points]

9. Find the set of companies that employ at least 3 persons who each know at least five persons. (So this query returns only one object, i.e., the set of companies specified in the query.) [10 Points]

10. Find the following set of sets

$$\{S \mid S \subseteq Skill \wedge \mid S \mid \leq 4\}$$

I.e., this is the set consisting of each set of job skills whose size (cardinality) is at most 4. [10 Points]

Hint: Let S1 be the set of skills with cardinality 1, S2 be the set of skills with cardinality 2, S3 be the set of skills with cardinality 3, S4 be the set of skills with cardinality 4. The resultant should be S1 $\cup S2 \cup S3 \cup S4$