Spring 2023 DS321 Assignment 7
Expressing Queries in Relational Algebra and RA SQL
Expressing Queries in Object-Relational Databases

Dirk Van Gucht

Due Wednesday April 26 by 11:45pm

1 Preliminaries

The goals for this assignment are to

- Expressing queries in RA and RA SQL;
- Expressing queries in Object-Relational Databases.

To turn in your assignment, you will need to upload to Canvas a single file with name `assignment7.sql` which contains the necessary SQL statements that solve the problems in this assignment. The `assignment7.sql` file must be so that the AI's can run it in their PostgreSQL environment. You should use the `assignment7Script.sql` file to construct the `assignment7.sql` file. (Note that the data to be used for this assignment is included in this file.) In addition, you will need to upload a separate `assignment7.txt` file that contains the results of running your SQL queries. Finally, you need to upload a file `assignment7.pdf` that contains the solutions to the problems that require it.

2 Expressing queries in RA SQL

2.1 Database Schema

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

```
Student(sid, sname, city)
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 `city` 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.) 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 `Knows` indicates that student s_1 knows student s_2.

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

2.2 Problems

Write RA SQL statements for each of the following queries. (Before expressing these queries in RA SQL, you are encouraged to first formulate these these queries in Relational Algebra in standard notation using σ, π, $\times$, $\cup$, $\cap$, $-$, $\bowtie$, and $\ltimes$.)

You are also encouraged to use regular views and temporary views. You can not use user-defined views.

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

You can **not** use subquery expressions `[NOT] EXISTS`, `[NOT] IN`, and θ `[SOME|ANY|ALL]`. However, you should using `CROSS JOIN, JOIN ON, and NATURAL JOIN` where necessary.

1. ● 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.

2. ● Find the pairs (d_1, d_2) of names of different departments whose main offices are located in the same building.

3. ● Find the sid and sname of each student whose home city is different than those of his or her friends.

4. ● Find each major that is the major of at most 2 students.

5. ● 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.

6. ● 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.)

7. ● For each department, list its name along with the highest salary made by students who are employed by it.

8. ● 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 s should only be considered if indeed he or she has a friend who is employed by department d.)

3 Expressing queries in Object Relational Databases

For the problems in the section, you will need to use (some of) the polymorphically defined functions and predicates that are included in the file

`Assignment-Script-2023-Spring-assignment7.sql`

Functions

set_union(A,B)	$A \cup B$
set_intersection(A,B)	$A \cap B$
set_difference(A,B)	$A - B$

Predicates

is_in(x,A)	$x \in A$
is_not_in(x,A)	$x \notin A$
is_empty(A)	$A = \emptyset$
is_not_emptyset(A)	$A \neq \emptyset$
subset(A,B)	$A \subseteq B$
superset(A,B)	$A \supseteq B$
equal(A,B)	$A = B$
overlap(A,B)	$A \cap B \neq \emptyset$
disjoint(A,B)	$A \cap B = \emptyset$

We now turn to the problems in this section. You will need use the data provided for the `Person`, `Company`, `companyLocation`, `worksFor`, `jobSkill`, `personSkill`, and `Knows` relations. But before turning to the problems, we will introduce various object-relational functions defined over these relations:[2]

- The function `companyHasEmployees(cname)` which associates with each company, identified by a cname, the set of pids of persons who work for that company.

```
create or replace function companyHasEmployees(c text)
returns int[] as
$$
select array(select pid
             from   worksFor
             where  cname = c order by 1);
$$ language sql;
```

[2]The various `order by` clauses in these functions are not essential: they simply aid to read the data more easily.

- The function `cityHasCompanies(city)` which associates with each city the set of cnames of companies that are located in that city.

  ```
  create or replace function cityHasCompanies(c text)
  returns text[] as
  $$
  select array(select cname
               from   companyLocation
               where  city = c order by 1);
  $$ language sql;
  ```

- The function `companyHasLocations(cname)` which associates with each company, identified by a cname, the set of cities in which that company is located.

  ```
  create or replace function companyHasLocations(c text)
  returns text[] as
  $$
  select array(select city
               from   companyLocation
               where  cname = c order by 1);
  $$ language sql;
  ```

- The function `knowsPersons(pid)` which associates with each person, identified by a pid, the set of pids of persons who he or she knows.

  ```
  create or replace function knowsPersons (p int)
  returns int[] as
  $$
  select array(select pid2
               from   Knows
               where  pid1 = p order by 1);
  $$ language sql;
  ```

- The function `isKnownByPersons(pid)` which associates with each person, identified by a pid, the set of pids of persons who know this person.

  ```
  create or replace function isKnownByPersons(p int) as
  $$
  select array(select pid1
               from   Knows
               where  pid2 = p order by 1);
  $$ language sql;
  ```

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

  ```
  create or replace function personHasSkills(p int)
  returns text[] as
  $$
  select array(select skill
               from   personSkill
  ```

```
             where  pid = p order by 1);
$$ language sql;
```

- The function `skillOfPersons(skill)` which associates with each skill the set of pids of persons who have that skill.

```
create or replace function skillOfPersons(s text)
returns int[] as
$$
select array(select pid
             from   personSkill
             where  skill = s order by 1);
$$ language sql;
```

In the problems in this section, you are asked to formulate queries in object-relational SQL. You should use the set operations and set predicates defined in the document `Assignment-Script-2023-Spring-assignment7.sql`, the relations

`Person`
`Company`
`Skill`
`worksFor`

and the functions

`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 functions.

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 `Assignment-Script-2023-Spring-assignment7.sql`.

Example 1 *Consider the query* "Find the pid of each person who knows a person who has a salary greater than 55000." *(In this example, focus on the* `is_in` *predicate.)*

```
select distinct p.pid
from   Person p, worksFor w
where  is_in(w.pid,knowsPersons(p.pid)) 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 order by 1;
```

Example 2 *Consider the query* "Find the pid and name of each person p who (1) has both the AI and Programming skills and (2) knows at least 5 persons, and report the number of persons who know p."[3]

```
select p.pid, p.pname,
       cardinality(isKnownByPersons(p.pid)) as ct_knownByPersons
from   Person p
where  subset('{"AI","Programming"}', personHasSkills(p.pid)) and
       cardinality(knowsPersons(p.pid)) >= 5;
```

Example 3 *Consider the query* "Find the pid and name of each person along with the set of his of her skills that are not among the skills of persons who work for 'Netflix'".[4]

```
select p.pid, p.pname, set_difference(personHasSkills(p.pid),
                                      array(select unnest(personHasSkills(p1.pid))
                                            from   Person p1
                                            where  is_in(p1.pid, companyHasEmployees('Netflix'))))
from   Person p;
```

9. Express the following query in object-relational SQL.

   Find the cname and headquarter of each company that employs at least three persons who have a common job skill.

10. Express the following query in object-relational SQL. Find each person who has no skill in common with those of the persons who works for Yahoo or for Netflix.

11. Express the following query in object-relational SQL. Find the set of companies that employ at least 2 persons who each know fewer than 3 persons. (So this query returns **just one** object, i.e., the set of companies specified in the query.)

12. Express the following query in object-relational SQL. Find the pid and name of each person p along with the set of pids of persons who (1) are known by p and (2) who have both the Programming and AI skills.

[3]In this example, focus on the set (array) construction `'{"AI", "Programming"}'` and the `subset` predicate. Also focus on the use of `cardinality` function.

[4]In this example, focus on (1) the `set_difference` operation and (2) the `unnest` operation followed by a set (`array`) construction.

13. Express the following query in object-relational SQL. Find each pair (s_1, s_2) of different skills s_1 and s_2 such that the number of employees who have skill s_1 and who make strictly less than 55000 is strictly smaller than the number of employees who have skill s_2 and who make strictly more than 55000.

14. Express the following query in object-relational SQL.

    Find each (c, p) pair where c is the cname of a company and p is the pid of a person who works for that company and who is known by all other persons who work for that company.

15. Express the following query in object-relational SQL.

    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 Yahoo.