Fall 2023 B461 Assignment 4

Released: 16 Oct, 2023
Due: Oct 26, 2023

This assignment relies on the lectures:

- Query Optimization
- Aggregate functions
- Functions and Expressions
- Queries with quantifiers

To turn in your assignment, you will need to upload to Canvas a single file with name `assignment4.sql` which contains the necessary SQL statements that solve the problems in this assignment. The `assignment4.sql` file must be so that the AI's can run it in their PostgreSQL environment. You should use the `SQL Script file` to construct the `assignment4.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 `assignment4.txt` file that contains the results of running your queries and `assignment4.pdf` file that contains the theoretical answers(Optimization steps) with conditions.

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 headquarter. 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 Assignment4Script.sql contains the data supplied for this assignment.

2 Optimizing RA expressions

In this section, you are asked to use the RA expressions obtained by translation in your previous assignment(**Assignment 3**).

You are required to show the intermediate steps that you took during the optimization.

You can use the following notation to denote relation names in RA expressions:

P, P_1, P_2, $\cdots$	`Person`
C, C_1, C_2, $\cdots$	`Company`
S, S_1, S_2, $\cdots$	`Skill`
W, W_1, W_2, $\cdots$	`worksFor`
cL, cL_1, cL_2, $\cdots$	`companyLocation`
pS, pS_1, pS_2, $\cdots$	`personSkill`
hM, hM_1, hM_2, $\cdots$	`hasManager`
K, K_1, K_2, $\cdots$	`Knows`

Note: Please make note of the following example, and use it as a template to construct your answers for this section. You should write all the steps and RA expressions in Latex or a word editor. Images of handwritten notes will NOT be accepted.

Example 1 *Consider the query* "Find each (p, c) pair where p is the pid of a person who works for a company c located in Bloomington and whose salary is the lowest among the salaries of persons who work for that company.

A possible formulation of this query in Pure SQL is

```
select w.pid, w.cname
from   worksfor w
where  w.cname in  (select cl.cname
                    from   companyLocation cl
                    where  cl.city = 'Bloomington') and
       w.salary <= ALL (select w1.salary
                        from   worksfor w1
                        where  w1.cname = w.cname);
```

The RA expression for the pure SQL query(that you have from Assignment 3) in standard notation is as follows:

$$\pi_{W.pid,W.cname}(\mathbf{E} \cap (W - \mathbf{F}))$$

where

$$\mathbf{E} = \pi_{W.*}(W \bowtie \sigma_{city=\mathbf{Bloomington}}(cL))$$

and

$$\mathbf{F} = \pi_{W.*}(W \bowtie_{W.salary>W_1.salary \wedge W_1.cname=W.cname} W_1).$$

Note: You have to start directly with the RA expression(of the Pure SQL query) for the following questions, and show the optimization steps. Finally, the last step should be the optimized RA expression.

Step 1 *Observe the expression* $\mathbf{E} \cap (W - \mathbf{F})$*. This expression is equivalent with* $(\mathbf{E} \cap W) - \mathbf{F}$*. Then observe that, in this case,* $\mathbf{E} \subseteq W$*. Therefore* $\mathbf{E} \cap W = \mathbf{E}$*, and therefore* $\mathbf{E} \cap (W - \mathbf{F})$ *can be replaced by* $\mathbf{E} - \mathbf{F}$*. So the expression for the query becomes*

$$\pi_{W.pid,W.cname}(\mathbf{E} - \mathbf{F}).$$

Step 2 *We now concentrate on the expression*

$$\mathbf{E} = \pi_{W.*}(W \bowtie \sigma_{city=\mathbf{Bloomington}}(cL)).$$

We can push the projection over the join and get

$$\pi_{W.*}(W \bowtie \pi_{cname}(\sigma_{city=\mathbf{Bloomington}}(cL))).$$

Which further simplifies to

$$W \ltimes \sigma_{city=\mathbf{Bloomington}}(cL).$$

We will call this expression $\mathbf{E}^{opt}$.

Step 3 *We now concentrate on the expression*

$$\mathbf{F} = \pi_{W.*}(W \bowtie_{W.salary>W_1.salary \wedge W_1.cname=W.cname} W_1).$$

We can push the projection over the join and get the expression

$$\pi_{W.*}(W \bowtie_{W.salary>W_1.salary \wedge W_1.cname=W.cname} \pi_{W_1.cname,W_1.salary}(W_1)).$$

We will call this expression $\mathbf{F}^{opt}$.

Therefore, the fully optimized RA expression is

$$\pi_{W.pid,W.cname}(\mathbf{E}^{opt} - \mathbf{F}^{opt}).$$

I.e.,

$$\begin{aligned}&\pi_{W.pid,W.cname}(W \ltimes \sigma_{city=\mathbf{Bloomington}}(cL) - \\ &\qquad \pi_{W.*}(W \bowtie_{W.salary>W_1.salary \wedge W_1.cname=W.cname} \pi_{W_1.cname,W_1.salary}(W_1))).\end{aligned}$$

1. "*Find the pid of each person that knows at least 2 people, such that at least 1 of them works at Apple or Netflix .*"

   Optimize the RA Expression that you came up with for the above query in Assignment 3 and mention at-least 2 conceptually different rewrite rules you used. [`10 pts`]
   We can start with what we were given:

$$\pi_{p.pid}\Bigg(p \bowtie_{k1.pid1=p.pid} k1 \\ \bowtie_{k2.pid1=p.pid \wedge k1.pid2 \neq k2.pid2} k2 \\ \bowtie_{(w.pid=k1.pid2 \vee w.pid=k2.pid2) \wedge (w.cname='Apple' \vee w.cname='Netflix')} w\Bigg)$$

   We can start by separating the W join thing to make W a seperate entity. This can be done by making W equal to

$$\mathbf{W} = \pi_{W.pid}(\sigma_{cname='Apple'}(W)) \cup \pi_{W.pid}(\sigma_{cname='Netflix}(W))$$

   This W can be optimized by pushing the projection inside, because projection distributes over union:

$$\mathbf{W} = \pi_{pid}(\pi_{pid}(\sigma_{cname='Apple'}(W)) \cup \pi_{pid}(\sigma_{cname='Netflix}(W)))$$

   and to optimize the original answer, we are only using the pid from the Person table, so we can push the projection inside of the expression to the P

$$\textbf{Answer} = \pi_{pid}(\pi_{pid}(P) \bowtie_{pid=K1.pid1} K_1 \bowtie_{pid=K2.pid1, \wedge\, K1.pid2 \neq K2.pid2} \\ K_2 \bowtie_{W.pid=K1.pid2, \vee\, W.pid=K2.pid2} W)$$

   These are two optimizations we can make, and we cant push that projection anywhere else because each of the K1, K2, and Wx tables use all of the columns in their relations, so we cannot simply grab just one column

2. "*Return the the pair (p, c) where p is the pid of a person, and c is the cname of the company where p works, such that (1) p is managed by someone who has at-least 2 skills and (2) p does not know anyone that lives in Seattle.*"

Optimize the RA Expression that you came up with for the above query in Assignment 3 and mention at-least 2 conceptually different rewrite rules you used. [`10 pts`]
what I wrote for the last assignment

$$\textbf{FIRST} = \pi_{W.pid,C.cname}(C \bowtie_{C.cname=W.cname} W \bowtie_{W.pid=M.eid} M \bowtie_{M.mid=PS1.pid} PS1 \bowtie_{M.mid=PS2.pid,\wedge PS1.skill \neq PS2.skill} PS2)$$

and

$$\textbf{SECOND} = \pi_{P.pid,C.cname}(P \bowtie W \bowtie C)$$

and

$$\textbf{THIRD} = \pi_{P.pid,C.cname}(P \bowtie W \bowtie C \bowtie_{K.pid1=P.pid} K \bowtie_{P1.pid=K.pid2\ P1.city=Seattle} P1)$$

Making the result:

$$\mathbf{F}IRST \cap (SECOND - THIRD)$$

Now first we can optimize this by projecting only the columns we need from each step. For the first one:

$$\textbf{FIRST} = \pi_{W.pid,C.cname}((\pi_{cname}(C)) \bowtie_{C.cname=W.cname} (\pi_{pid,cname}(W)) \bowtie_{W.pid=M.eid} M \bowtie_{M.mid=PS1.pid} PS1 \bowtie_{M.mid=PS2.pid,\wedge PS1.skill \neq PS2.skill} PS2)$$

Here we didn't need the salary from WorksFor, and the headquarter of the company For the second step:

$$\textbf{SECOND} = \pi_{P.pid,C.cname}((\pi_{pid}(P)) \bowtie (\pi_{pid,cname}(W)) \bowtie (\pi_{cname}(C)))$$

We didn't need pname, city from Person, and the salary from worksFor, and the headquarters from company. For the third:

$$\textbf{THIRD} = \pi_{P.pid,C.cname}((\pi_{pid}(P)) \bowtie (\pi_{pid,cname}(W)) \bowtie (\pi_{cname}(C)) \bowtie_{K.pid1=P.pid} K \bowtie_{P1.pid=K.pid2 \wedge P1.city=Seattle} P1)$$

Once again, didnt need the pname or city from Person, and didnt need salary from w, and didnt need headquarters from C. In addition, for our second optimization, we can separate the P in third to make the constant its own small query.

$$\mathbf{P} = \pi_{P.pid}(\sigma_{city='Seattle'}(P))$$

Now, we can change the third to be this:

$$\textbf{THIRD} = \pi_{P.pid,C.cname}((\pi_{pid}(P)) \bowtie (\pi_{pid,cname}(W)) \bowtie (\pi_{cname}(C)) \bowtie_{K.pid1=P.pid} K \bowtie_{P1.pid=K.pid2} P)$$

Now, here is the final answer:

$$\mathbf{F}IRST \cap (SECOND - THIRD)$$

3. "*Return each skill that is the skill of at least 2 persons, such that at least 1 of them lives in Bloomington*"

   Optimize the RA Expression that you came up with for the above query in Assignment 3 and mention at-least 2 conceptually different rewrite rules you used. [`10 pts`]
   Here is what was given to start.

   $=\pi_{S.skill}($
   $S \bowtie pS1 \bowtie_{ps1.skill=ps2.skill \wedge\ ps1.pid \neq ps2.pid}\ Ps2 \bowtie_{P.city='Bloomington'}$
   $(P.pid = pS1.pid \vee\ P.pid = pS2.pid)\ P)$

   Can you please pretend there is an

   $$\wedge$$

   right after the bloomington, I could not get this to work in Latex to be formatted correctly. Now we can first optimize it by taking out the constant to be seperated:

   $$\mathbf{P} = \pi_{pid}(\sigma_{city='Bloomington'}(P))$$

   Which makes the answer:

   $=\pi_{S.skill}($
   $S \bowtie pS1 \bowtie_{ps1.skill=ps2.skill \wedge\ ps1.pid \neq ps2.pid}\ Ps2 \bowtie_{P.pid=pS1.pid \vee\ P.pid=pS2.pid}\ P)$

   Now, I am going to do something that I'm not sure is allowed, we can just get rid of the Skill table entirely, which is 100 percent an optimization and still returns the correct results, because even if a skill is present in the skill table that is not in personSkill, it wouldn't show up in this output because we want skills that at least two people have. So we can eliminate the entire skill table from this query.

$$=\pi_{Ps1.skill}($$
$$pS1 \bowtie_{ps1.skill=ps2.skill\wedge\ ps1.pid\neq ps2.pid}\ Ps2 \bowtie_{P.pid=pS1.pid\vee\ P.pid=pS2.pid}\ P)$$

This will give us the same results, and is optimized with two steps.

4. "*Return the pair (p, s) where p is the pid of a person that works at a company headquartered in MountainView and s is the minimum salary among all people that know p.*"

   Optimize the RA Expression that you came up with for the above query in Assignment 3 and mention at-least 2 conceptually different rewrite rules you used. [`10 pts`]
   Here is what we have from the solutions: Let,

   $$\textbf{mv} = \pi_{p.pid}(P \bowtie_{P.pid=W.pid}\ W \bowtie_{W.cname=C.cname\wedge\ C.headquarter='MountainView'}\ C)$$

   and (all_that_know_p gets abreivated to ATP)

   $$\textbf{all\_that\_know\_p} = \pi_{MV.pid,W.salary}(MV \bowtie_{MV.pid=K.pid2}\ K \bowtie_{K.pid1=W.pid}\ W)$$

   and

   $$\textbf{q} = \pi_{ATP.pid,ATP.salary}(ATP)$$
   $$-\ \pi_{ATP.pid,ATP.salary}(ATP \bowtie_{ATP.salary>ATP1.salary,\wedge\ ATP1.pid=ATP.pid}\ ATP1)$$

   With the answer being:

   $$=\pi_{pid,salary}(q)$$

   For the optimizations, for the start we can remove the mountainview check from mv and put it into its own subquery in order to remove that and, speeding up our query. Let,

   $$\textbf{C} = \pi_{cname}(\sigma_{headquarter='MountainView'}C)$$

   therefore

   $$\textbf{mv} = \pi_{p.pid}(P \bowtie_{P.pid=W.pid}\ W \bowtie_{W.cname=C.cname}\ C)$$

Then, we can remove the unnecessary columns from the steps. For mv, we can project only the pid, W we can project only the pid and cname, and c we already have changed it to only get the cname therefore

mv $= \pi_{p.pid}((\pi_{pid}(P)) \bowtie_{P.pid=W.pid} (\pi_{pid,cname}(W)) \bowtie_{W.cname=C.cname} C)$

For all that know, we cannot do anthing except for the W, where we can project only the pid and salary, since mv already only gets the pid, and we use both pid1 and pid2 from K

all_that_know_p $= \pi_{MV.pid,W.salary}(MV \bowtie_{MV.pid=K.pid2} K \bowtie_{K.pid1=W.pid} (\pi_{pid,salary}(W)))$

Then q still is the same because we are using our own views that only contain the columns we need already, we can even change the first project to a project * if we wanted, but I dont think thats an optimization.

q $=\pi_{ATP.pid,ATP.salary}(ATP)$
$\quad - \pi_{ATP.pid,ATP.salary}(ATP \bowtie_{ATP.salary>ATP1.salary, \wedge ATP1.pid=ATP.pid} ATP1)$

But there we go, the answer is still

$$=\pi_{pid,salary}(q)$$

,but we have completed two optimizations, one by moving the constants to a subquery, and the other by projecting out redundant columns from our tables.

5. "*Return each cname such that*
(1) at least 1 person working there has the OperatingSystems skill
(2) at least 2 persons working there live in different cities"

Optimize the RA Expression that you came up with for the above query in Assignment 3 and mention at-least 2 conceptually different rewrite rules you used. [`10 pts`]
what I had before

Answer $=\pi_{C.cname}(C \bowtie W \bowtie_{W.pid=PS.pid \wedge ps.Skill='OperatingSystems'})$
$PS \bowtie_{W1.cname=C.cname} W1 \bowtie_{W2.cname=C.cname, \wedge W1.pid \neq W2.pid}$
$W2 \bowtie_{P1.pid=W1.pid} P1 \bowtie_{W2.pid=P2.pid, \wedge P1.city \neq P2.city} P2$

To first optimize this, we can take the constant out and make it its own thing like we have been doing.

$$\mathbf{PS} = \pi_{pid}(\sigma_{skill='OperatingSystems'} PS)$$

Now our ra looks like this

$\textbf{Answer} = \pi_{C.cname}(C \bowtie W \bowtie_{W.pid=PS.pid})$
$PS \bowtie_{W1.cname=C.cname} W1 \bowtie_{W2.cname=C.cname, \wedge W1.pid \neq W2.pid}$
$W2 \bowtie_{P1.pid=W1.pid} P1 \bowtie_{W2.pid=P2.pid, \wedge P1.city \neq P2.city} P2$

Now, we can project out columns we do not need.

$\textbf{Answer} = \pi_{C.cname}((\pi_{cname}(C)) \bowtie (\pi_{pid,cname}(W)) \bowtie_{W.pid=PS.pid})$
$PS \bowtie_{W1.cname=C.cname} (\pi_{pid,cname}(W1)) \bowtie_{W2.cname=C.cname, \wedge W1.pid \neq W2.pid}$
$(\pi_{pid,cname}(W2)) \bowtie_{P1.pid=W1.pid} (\pi_{pid,city}(P1)) \bowtie_{W2.pid=P2.pid, \wedge P1.city \neq P2.city} (\pi_{pid,cit}$

Optimizations: We can grab only the cname column from C, because thats all we need, we can grab only the pid and cname from W, we don't need the salary, we can grab the pid, cname from W1, as well as W2, and can grab the pid and city from P1 and P2, which this along with moving the constant out is my final optimized answer, with the two optimizations being applied.

3 Solving queries using Aggregate Functions

Formulate the following queries in SQL. You must use aggregate functions in ALL these queries and must not use set predicates where it is mentioned explicitly. You can use views, temporary views, parameterized views, and user-defined functions and expressions.

6. Find each pair (c, p) where c is the city and p is the pid of the person that lives in c, and earns the lowest salary among all persons living in c. You must not use set predicates in this query. [`10 pts`]

7. Let p1 be a person and N be the set of skills of Netflix employees. Find the pid and name of each person p1 if p1 has less then 2 of the skills in N i.e. the combined set of job skills of persons who work for Netflix.

$$\{s \mid s\ is\ a\ jobskill\ of\ an\ employee\ of\ Netflix\ \}$$

   [`10 pts`]

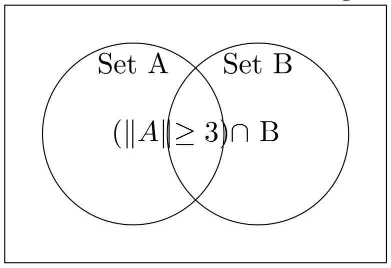
8. Find each pid of a person who knows at least two people who (a) work for Apple and (b) who make less than 60000.You must not use set predicates in this query. [`10 pts`]

4 Queries with quantifiers

Use the idea of Venn diagrams to write SQL queries for the following queries with quantifiers.
To get full credit in these problems, you must write appropriate views and parameterized views for the sets A and B that occur in the Venn diagram with conditions for these queries. (See the lecture on Queries with Quantifiers.)

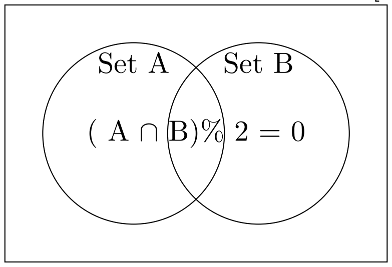
Make the following query using the COUNT function:

9. Find the pid and name of each person who knows at least 3 people who each have at most 2 managers. [`10 pts`] To make this venn diagram,



In this venn diagram, Set A represents the set of all people that a person knows, and Set B represents the people who have at most 2 managers. The intersection of this set is where the person knows at least 3 people, and the person has less than or equal to two managers.(Can you pretend like theres only one set of bar around the A, couldnt get it for format correctly) The rest of set A has the people who know less than 3 people and who have more than 3 managers, A-B, and set B is just the people who have at most two managers, so there isn't any people who do not fulfill that condition.

10. Find the cname of each company that employs an even number of persons who have at least 2 skills. [`10 pts`] To make this venn diagram,



CAN YOU PLEASE PRETEND LIKE THE PARENTHESIS ARE BARS, I COULDNT GET

THE VENN DIAGRAM TO RENDER WITH THEM AGAIN In this venn diagram, set A is the people(pids) of people who work at a company. Set B in this people who have at least two skills. The intersection of these two is all the people who work at a company who have at least two skills, and the total of these people who work there is an even number. This means that the rest of Set A is the people who work at a company who do not have two or more skills, or who work for a company that does not have an even number of these two people, and set B is just everyone with more than 1 skill so there is no rest of set B.