B561 Advanced Database Concepts
Assignment 3
Spring 2023

Muazzam Siddiqui, Srinivas Kini, Pavan Kalyan Thota

This assignment tests the following concepts:

- Pure SQL
- Relational Algebra (RA)
- Joins and semi-joins
- Pure SQL to RA SQL and RA Expressions
- Query optimization

with particular focus on the last two lectures.

To turn in your assignment, you will need to upload to Canvas a single file with name `assignment3.sql` which contains the necessary SQL statements that solve the problems in this assignment. The `assignment3.sql` file must be so that the AI's can run it in their PostgreSQL environment. You should use the script file to construct the `assignment3.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 `assignment3.txt` file that contains the results of running your queries. Finally, you need to upload a file `assignment3.pdf` that contains the solutions to the problems that require it.

- Include all problems with the blue bullet ●in assignment3.sql
- Include all problems with the red bullet ●in assignment3.pdf

Database schema and instances

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

$\texttt{Person(}\underline{\texttt{pid}}\texttt{, pname, city)}$
$\texttt{Company(}\underline{\texttt{cname}}\texttt{, headquarter)}$
$\texttt{Skill(}\underline{\texttt{skill}}\texttt{)}$
$\texttt{worksFor(}\underline{\texttt{pid}}\texttt{, cname, salary)}$
$\texttt{companyLocation(}\underline{\texttt{cname, city}}\texttt{)}$
$\texttt{personSkill(}\underline{\texttt{pid, skill}}\texttt{)}$
$\texttt{hasManager(}\underline{\texttt{eid, mid}}\texttt{)}$
$\texttt{Knows(}\underline{\texttt{pid1, pid2}}\texttt{)}$

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

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

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 (p_1, p_2) where p_1 and p_2 are pids of persons. The pair (p_1, p_2) indicates that the person with pid p_1 knows the person with pid p_2. We do not assume that the relation `Knows` is symmetric: it is possible that (p_1, p_2) is in the relation but that (p_2, p_1) 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`; and
- `pid2` is a foreign key in `Knows` referencing the primary key `pid` in `Person`

Pure SQL and RA SQL

In this assignment, we distinguish between Pure SQL and RA SQL.

Pure SQL *Rules*:

- `SELECT, FROM, WHERE`
- Set operations: `UNION, INTERSECT, EXCEPT`
- Predicates: `EXISTS, NOT EXISTS`
- Predicates: `IN, NOT IN`
- Predicates: `[NOT] (ALL, SOME)`
- Window functions `PARTITION, RANK` etc. and aggregate functions `SUM, AVG` etc. are **not** allowed
- `VIEWs, WITH` clauses that obey the rules above

RA SQL *Rules*:

- `SELECT, FROM, WHERE`
- The `WHERE` clause can **only** be used with constants.
- Set operations: `UNION, INTERSECT, EXCEPT`
- `JOINs, CROSS JOINs, NATURAL JOINs`
- Commas are **not** allowed
- Window functions `PARTITION, RANK` etc. and aggregate functions `SUM, AVG` etc. are **not** allowed
- `VIEWs, WITH` clauses that obey the rules above

In particular, any other elements of SQL that are not mentioned in the lecture slides (like `LIMIT`) are **not** allowed for Pure SQL **or** RA SQL

1 Theoretical problems related to query translation and optimization

1. Consider two RA expressions E_1 and E_2 over the same schema and an RA expression F with a schema that is not necessarily the same as that of E_1 and E_2.

   Consider the following `if-then-else` query:

   $$\begin{array}{lll}\texttt{if } F=\emptyset & \texttt{then} & \texttt{return } E_1\\ & \texttt{else} & \texttt{return } E_2\end{array}$$

   We can formulate this query in SQL as follows[2]:

   ```
   select e1.*
   from   E1 e1
   where  not exists (select distinct row() from F)
   union
   select e2.*
   from   E2 e2
   where  exists (select distinct row() from F);
   ```

   Remark 1 `select distinct row() from F`
   returns the empty set if $F=\emptyset$ and returns the tuple `()` *if $F\neq\emptyset$.*[3]
   In RA, this query can be written as

   $$\pi_{()}(F).$$

   I.e., the projection of F on an empty list of attributes.

   • Using E_1, E_2, and F, write an RA expression in standard notation that expresses the above `if-then-else` query:[4] [`5 pts`]

2. Let `R(x)` be a unary relation that can store a set of integers R. Consider the following Sample boolean SQL query:

[2]In this SQL query `E1`, `E2`, and `F` denote SQL queries corresponding to the RA expressions E_1, E_2, and F, respectively.

[3]The tuple `()` is often referred to as the *empty tuple*, i.e., the tuple without components. It is akin to the empty string ϵ in the theory of formal languages. I.e., the string without alphabet characters.

[4]Hint: consider using the Pure SQL to RA SQL translation algorithm.

```
select not exists(select 1
                  from   R r1, R r2
                  where  r1.x <> r2.x) as fewerThanTwo;
```

This boolean query returns the constant "`true`" if R has fewer than two elements and returns the constant "`false`" otherwise. [5]

● Using the insights you gained from Problem 1 and the sample boolean query:

(a) Formulate the boolean query in RA SQL (in the pdf file) : *Return the pair (c, t) where c is the cname and t is '1' if c has at least 1 employee that earns more than 55000 and knows their manager and '0' otherwise.* [`3 pts`]

(b) Write the RA Expression for this query in standard RA Notation. [`2 pts`]

3. In the translation algorithm from Pure SQL to RA we assumed that the argument of each set predicate was a (possibly parameterized) Pure SQL query that did not use a `union`, `intersect`, nor an `except` operation.

   In this problem, you are asked to extend the translation algorithm from Pure SQL to RA such that the argument of set predicate is a (parameterized) pure SQL query containing `UNION` , `INTERSECT, EXCEPT` operations.

   More specifically, consider the following types of queries using the [`not`] `exists` set predicate.

```
select L1(r1,...,rn)
from   R1 r1, ..., Rn rn
where  C1(r1,...,rn) and
              [not] exists (select L2(s1,...,sm)
                            from   S1 s1,..., S1 sm
                            where  C2(s1,...,sm,r1,...,rn)
                            [union | intersect | except]
                            select L3(t1,...,tk)
                            from   T1 t1, ..., Tk tk
                            where  C3(t1,...,tk,r1,...,rn))
```

[5] Hint: recall that, in general, a constant value "`a`" can be represented in RA by an expression of the form `(A: a)`. (Here, `A` is some arbitrary attribute name.) Furthermore, recall that we can express `(A: a)` in SQL as "`select a as A`". Thus RA expressions for the constants "`true`" and "`false`" can be the expressions `(A: true)` and `(A: false)`, respectively.

Observe that there are six cases to consider:

```
(a) exists (...  union ...)
(b) exists (...  intersect ...)
(c) exists (...  except ...)
(d) not exists (...  union ...)
(e) not exists (...  intersect ...)
(f) not exists (...  except ...)
```

● Show how such SQL queries can be translated to equivalent RA expressions in standard notation. In the translation, you should take into account that projections do not in general distribute over intersections and over set differences. [`10 pts`]

To get practice, first consider the following special case where $n = 1$, $m = 1$, and $k = 1$. I.e., the following case: [6]

```
select L1(r)
from   R r
where  C1(r) and [not] exists (select L2(s)
                               from   S s
                               where  C2(s,r)
                               [union | intersect | except]
                               select L3(t)
                               from   T t
                               where  C3(t,r))
```

4. ● Let R be a relation with schema (a, b, c) and let S be a relation with schema (d, e).

   Prove, from first principles[7], the correctness of the following rewrite rule:

$$\pi_{a,d}(R \bowtie_{c=d} S) = \pi_{a,d}(\pi_{a,c}(R) \bowtie_{c=d} \pi_d(S)).$$

   [`5 pts`]

[6]Once you can handle this case, the general case is a similar.

[7]In particular, do not use the rewrite rule of pushing projections over joins. Rather, use Predicate Logic or TRC to provide a proof.

2 Translating Pure SQL to RA SQL and optimized RA expressions

In this section, you are asked to *translate* Pure SQL queries into RA SQL queries as well as standard RA expressions using the *translation algorithm.*

You are required to show the intermediate steps that you took during the translation. After the translation, you are asked to *optimize* the resulting RA expressions.

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 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);
```

Translation of '`and`*' in the '*`where`*' clause.*

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

Translation of '`in`*' and '*`<= ALL`*'.*

```
select q.pid, q.cname
from   (select w.*
        from   worksfor w, companyLocation cl
        where  w.cname = cl.cname and cl.city = 'Bloomington'
        intersect
        (select w.*
         from   worksfor w
         except
         select w.*
         from   worksfor w, worksfor w1
         where  w.salary > w1.salary and w1.cname = w.cname)) q;
```

Move 'constant' condition.

```
select q.pid, q.cname
from   (select w.*
        from worksfor w,
        (select cl.* from companyLocation cl  where cl.city = 'Bloomington') cl
        where  w.cname = cl.cname
        intersect
        (select w.*
         from   worksfor w
         except
         select w.*
         from   worksfor w, worksfor w1
         where  w.salary > w1.salary and w1.cname = w.cname)) q;
```

Introduction of natural join and join.

```
select q.pid, q.cname
from   (select w.*
        from   worksfor w
               natural join
               (select cl.* from companyLocation cl  where cl.city = 'Bloomington') cl
               intersect
              (select w.*
               from   worksfor w
               except
               select w.*
               from   worksfor w join worksfor w1 on
               (w.salary > w1.salary and w1.cname = w.cname))) q;
```

This RA SQL query can be formulated as an RA expression in standard notation 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

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

We can now commence the optimization.

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.,

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

5. “*Find the pid of each person that knows at least 2 people, such that at least 1 of them works at Apple or Netflix .*”
   A possible way to write this query in Pure SQL is

```
select distinct p.pid from Person p, Knows k1, Knows k2
where k1.pid1 = p.pid
and k2.pid1 = p.pid
and k1.pid2 <> k2.pid2
and exists (
            select 1 from worksFor w
            where (w.pid = k1.pid2 or w.pid = k2.pid2)
            and   (w.cname = 'Apple' or w.cname = 'Netflix')
);
```

   (a) ● Translate the Pure SQL query to RA SQL using the translation algorithm step-by-step. [5 pts]
   (b) ● Write the RA Expression of the translated RA SQL query in standard notation. [2 pts]
   (c) ● Optimize this RA Expression and mention at-least 2 conceptually different rewrite rules you used. [8 pts]

6. “*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.*”
   A possible way to write this query in Pure SQL is

```
select p.pid, c.cname
from person p , company c, worksfor w
where p.pid = w.pid and c.cname = w.cname
and exists (
     select 1 from hasManager m, personSkill ps1, personSkill ps2
     where m.eid = p.pid and ps1.pid = m.mid and ps2.pid = m.mid
     and ps1.skill <> ps2.skill
)
and p.pid not in (
     select k1.pid1 from knows k1
     where k1.pid2 in (select p1.pid from person p1 where p1.city='Seattle')
);
```

   (a) ● Translate the Pure SQL query to RA SQL using the translation algorithm step-by-step. [5 pts]
   (b) ● Write the RA Expression of the translated RA SQL query in standard notation. [2 pts]

(c) ● Optimize this RA Expression and mention at-least 2 conceptually different rewrite rules you used. [8 pts]

7. "*Return each skill that is the skill of at least 2 persons, such that at least 1 of them lives in Bloomington*"
   A possible way to write this query in Pure SQL is

```
select s.skill from skill s
where exists (
    select 1 from personskill ps1, personskill ps2
    where ps1.pid <> ps2.pid
    and ps1.skill = ps2.skill and s.skill = ps1.skill
    and exists (
        select 1 from person p where p.city = 'Bloomington'
        and (ps1.pid = p.pid or ps2.pid = p.pid)
    )
);
```

   (a) ● Translate the Pure SQL query to RA SQL using the translation algorithm step-by-step. [5 pts]

   (b) ● Write the RA Expression of the translated RA SQL query in standard notation. [2 pts]

   (c) ● Optimize this RA Expression and mention at-least 2 conceptually different rewrite rules you used. [8 pts]

8. "*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.*"

   A possible way to write this query in Pure SQL is

```
with mv
 as (select P.pid
     from   person P,worksfor W,company C
     where  P.pid = W.pid and W.cname = C.cname
            and C.headquarter = 'MountainView'),
 all_that_know_p
 as (select mv.pid,W.salary
     from   mv MV,knows K,worksfor W
     where  K.pid2 = mv.pid and W.pid = K.pid1
     order  by 1),
 min_salary
 as (select distinct ATP.pid,ATP.salary
     from   all_that_know_p ATP
     where  not exists (select 1 from   all_that_know_p ATP1 where  ATP.salary >
         ATP1.salary and ATP1.pid = ATP.pid))
select * from   min_salary;
```

(a) ● Translate the Pure SQL query to RA SQL using the translation algorithm step-by-step. [`5 pts`]

(b) ● Write the RA Expression of the translated RA SQL query in standard notation. [`2 pts`]

(c) ● Optimize this RA Expression and mention at-least 2 conceptually different rewrite rules you used. [`8 pts`]

9. "*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"

A possible way to write this query in Pure SQL is

```
select C.cname from company C
where  exists (select 1
               from   worksfor W,personskill PS
               where  W.cname = C.cname
                      and W.pid = PS.pid
                      and PS.skill = 'OperatingSystems')
       and exists (select 1
                   from   worksfor W1,worksfor W2,person P1,person P2
                   where  W1.cname = C.cname
                          and W2.cname = C.cname and W1.pid = P1.pid
                          and W2.pid = P2.pid and W1.pid <> W2.pid
                          and P1.city <> P2.city);
```

(a) ● Translate the Pure SQL query to RA SQL using the translation algorithm step-by-step. [`5 pts`]

(b) ● Write the RA Expression of the translated RA SQL query in standard notation. [`2 pts`]

(c) ● Optimize this RA Expression and mention at-least 2 conceptually different rewrite rules you used. [`8 pts`]