# Examples of SQL translation and optimization

Dirk Van Gucht

October 18, 2020

In this note we show how to translate Pure SQL query to RA SQL queries using the translation algorimth presented in the lectures. We then show how to optimize these RA SQL queries into corresponding optimized RA SQL queries. We also show how the translated and optimed RA SQL queries are formulated in RA in standard notation.

## 1 Example 1

*Find the pid and name of each person who (a) works for a company located in 'Bloomington' and (b) knows as person who lives in 'Chicago'.*

```
select p.pid, p.name
from   person p
where  p.pid in (select w.pid
                 from   worksfor w
                 where  w.cname in (select c.cname
                                    from   company c
                                    where  c.city = 'Bloomington')) and
       p.pid in (select k.pid1
                 from   knows k
                 where  k.pid2 in (select p2.pid
                                   from   person p2
                                   where  p2.city = 'Chicago'));
```

We will translate this Pure SQL query into a RA SQL. We will then optimize this RA SQL query.

```
-- Step 1:  translate 'in' set predicates to 'exists' set predicates

select p1.pid, p1.name
```

```
from   person p1
where  exists (select w.pid
               from   worksfor w
               where  p1.pid = w.pid and
                      EXISTS (select c.cname
                              from   company c
                              where  w.cname = c.cname and
                                     c.city = 'Bloomington')) and
       exists (select k.pid1
               from   knows k
               where  p1.pid = k.pid1 and
                      EXISTS (select p2.pid
                              from   person p2
                              where  k.pid2 = p2.pid and
                                     p2.city = 'Chicago'))
order by 1,2;

-- Step 2 Translate the inner 'EXISTS' set predicates

select p1.pid, p1.name
from   person p1
where  exists (select w.pid
               from   worksfor w, company c
               where  p1.pid = w.pid and
                      w.cname = c.cname and
                      c.city = 'Bloomington') AND
       exists (select k.pid1
               from   knows k, person p2
               where  p1.pid = k.pid1 and
                      k.pid2 = p2.pid and
                      p2.city = 'Chicago')
order by 1,2;

-- Step 3 Translate the conjunction 'AND'
--        in the outermost 'where' clause
--        and introduce the 'intersect' operator

select pid, name
from  (select p1.*
       from   person p1
       where  EXISTS (select w.pid
                      from   worksfor w, company c
                      where  p1.pid = w.pid and
                             w.cname = c.cname and
                             c.city = 'Bloomington')
       intersect
       select p1.*
       from   person p1
       where  EXISTS (select k.pid1
```

```
                    from   knows k, person p2
                    where  p1.pid = k.pid1 and
                           k.pid2 = p2.pid and
                           p2.city = 'Chicago')) q
order by 1,2;



-- Step 4  Thranslate the 'exists' set predicates


select pid, name
from  (select p1.*
       from   person p1, worksfor w, company c
       where  p1.pid = w.pid and
              w.cname = c.cname and
              c.city = 'Bloomington'
       intersect
       select p1.*
       from   person p1, knows k, person p2
       where  p1.pid = k.pid1 and
              k.pid2 = p2.pid and
              p2.city = 'Chicago') q
order by 1,2;

-- Step 5
-- Move the condition 'c.city = 'Bloomington' to 'company'
-- Move the condition 'p2.city = 'Chicago'' to 'person'

with
   companyBloomington as (select cname, city
                          from company
                          where city = 'Bloomington'),
   personChicago as (select pid, name, city, birthyear
                     from   person
                     where  city = 'Chicago')
select pid, name
from  (select p1.*
       from   person p1, worksfor w, companyBloomington c
       where  p1.pid = w.pid and
              w.cname = c.cname
       intersect
       select p1.*
       from   person p1, knows k, personChicago p2
       where  p1.pid = k.pid1 and
              k.pid2 = p2.pid) q
order by 1,2;
```

```
-- Step 6
-- Introduce join operations
--         This gives us a query in RA SQL
--         which we can translate in standard RA notation

with
   companyBloomington as (select cname, city
                          from   company
                          where  city = 'Bloomington'),
   personChicago as (select pid, name, city, birthyear
                     from   person
                     where  city = 'Chicago')
select pid, name
from  (select p1.*
       from   person p1
              natural join worksfor w
              join companyBloomington c on (w.cname = c.cname)
       intersect
       select p1.*
       from   person p1
              join knows k on p1.pid = k.pid1
              join personChicago p2 on k.pid2 = p2.pid) q
order by 1,2;
```

We can formulate this RA SQL query as an RA expression in standard notation.

Consider the expressions

$$\begin{aligned} companyBloomington &= \sigma_{city=\texttt{Bloomington}}(C) \\ personChicago &= \sigma_{city=\texttt{Chicago}}(Person) \\ E &= \pi_{P.*}(P \bowtie W \bowtie_{W.cname=C.cname} companyBloomington) \\ F &= \pi_{P_1.*}(P_1 \bowtie_{P_1.pid=pid1} K \bowtie_{pid2=personChicago.cname)} personChicago) \end{aligned}$$

Then the RA expression for the query is

$$\pi_{pid,name}(E \cap F).$$

We can now begin with the optimization. The main rule is to push projections over joins In this case that is all we can do and we get the fully optimized RA SQL query. This query can be translated into standard RA notation

```
-- Step 1

with
   companyBloomington as (select distinct cname
                          from   company
                          where  city = 'Bloomington'),
```

```
   personChicago as (select pid
                     from   person
                     where  city = 'Chicago')
select pid, name
from  (select pid, name
       from   person
              natural join (select distinct pid
                            from   (select pid, cname
                                    from worksfor) w
                                    natural join companyBloomington) q
       intersect
       select distinct pid, name
       from   (select pid, name from person) p1
              join (select distinct pid1
                    from   knows
                           join personChicago p2 on pid2 = p2.pid) q on p1.pid = pid1) q
order by 1,2;

-- Step 2 Notice that outer project list (pid, name) is identical
-- to schema of the expression in the from clause.
-- So we can simplify to


with
   companyBloomington as (select distinct cname
                          from   company
                          where  city = 'Bloomington'),
   personChicago as (select pid
                     from   person

                     where  city = 'Chicago')
select pid, name
from   person
       natural join (select distinct pid
                     from   (select pid, cname
                             from worksfor) w
                                  natural join companyBloomington) q
      intersect
select distinct pid, name
from   (select pid, name from person) p1
       join (select distinct pid1
             from   knows
                    join personChicago p2 on pid2 = p2.pid) q on p1.pid = pid1
order by 1,2;
```

We can formulate this optimized RA SQL query as an RA expression in standard notation.

Consider the expressions

$$\begin{aligned} companyBloomington &= \pi_{cname}(\sigma_{city=\texttt{Bloomington}}(C)) \\ personChicago &= \pi_{pid,name}(\sigma_{city=\texttt{Chicago}}(Person)) \\ E &= \pi_{pid,name}(P \ltimes (\pi_{pid}(\pi_{pid,cname}(W) \ltimes companyBloomington))) \\ F &= \pi_{pid,name}(\pi_{pid,name}(P_1) \bowtie_{P_1.pid=pid1} (\pi_{pid1}(K \bowtie_{pid2=personChicago.cname} personChicago))) \end{aligned}$$

Then the RA expression for the query is

$$E \cap F.$$

# 2 Example 2

```
-- Find the cname of each company along with the pids and names of the
-- persons who work for that company and who have the next to
-- lowest salary (i.e., the second lowest salary) at that company.

select  distinct w.cname, p.pid, p.name
from    worksfor w, person p
where   w.pid = p.pid and
        exists (select 1
                from   worksfor w1
                where  w1.cname = w.cname and
                       w.salary > w1.salary) AND
        not exists (select 1
                    from   worksfor w1, worksfor w2
                    where  w.cname = w1.cname and
                           w1.cname = w2.cname and
                           w.salary > w1.salary and w1.salary > w2.salary)
order by 1,2,3;



-- Step 1: Introduce a natural join between 'worksfor' and 'person'

with
  pWorksfor as (select w.cname, w.salary, p.pid, p.name, p.city, p.birthyear
                from   person p natural join worksfor w)
select  distinct cname, pid, name
from    pworksfor w
where   exists (select 1
                from   worksfor w1
                where  w1.cname = w.cname and
                       w.salary > w1.salary) AND
        NOT exists (select 1
                    from   worksfor w1, worksfor w2
                    where  w.cname = w1.cname and
                           w1.cname = w2.cname and
                           w.salary > w1.salary and w1.salary > w2.salary)
```

```
order by 1,2,3;


-- Step 2: Translate AND NOT by introducing an EXCEPT

with
  pWorksfor as (select w.cname, w.salary, p.pid, p.name, p.city, p.birthyear
                from   person p natural join worksfor w)
select  distinct cname, pid, name
from (select  w.*
      from    pworksfor w
      where   exists (select 1
                      from   worksfor w1
                      where  w1.cname = w.cname and
                             w.salary > w1.salary)
      except
      select  w.*
      from    pworksfor w
      where   exists (select 1
                      from   worksfor w1, worksfor w2
                      where  w.cname = w1.cname and
                             w1.cname = w2.cname and
                             w.salary > w1.salary and w1.salary > w2.salary)) q
order by 1,2,3;


-- Step 3 Eliminate the 'exists' set predicates

with
  pWorksfor as (select w.cname, w.salary, p.pid, p.name, p.city, p.birthyear
                from   person p natural join worksfor w)
select  distinct cname, pid, name
from (select distinct w.*
      from   pworksfor w, worksfor w1
      where  w1.cname = w.cname and w.salary > w1.salary
      except
      select distinct w.*
      from   pworksfor w, worksfor w1, worksfor w2
      where  w.cname = w1.cname and
             w1.cname = w2.cname and
             w.salary > w1.salary and w1.salary > w2.salary) q
order by 1,2,3;


-- Step 4: Introduce join operations
-- Observe that we have a RA SQL query which can be translated directly
-- in RA in standard notation.
```

```
with
  pWorksfor as (select w.cname, w.salary, p.pid, p.name, p.city, p.birthyear
                from   person p natural join worksfor w)
select distinct cname, pid, name
from (select distinct w.*
      from   pWorksfor w
             join worksfor w1 on (w.cname = w1.cname and w.salary > w1.salary)
      except
      select distinct w.*
      from   pWorksfor w
             join worksfor w1 on ( w.cname = w1.cname and  w.salary > w1.salary)
             join worksfor w2 on (w1.cname = w2.cname and w1.salary > w2.salary)) q
order by 1,2,3;
```

We can formulate this RA SQL query as an RA expression in standard notation.
Consider the expressions

$$\begin{aligned} pWorksFor &= Person \bowtie worksFor \\ E &= \pi_{pWorksfor.*}(pWorksfor \bowtie_{pWorksfor.cname=W_1.cname \wedge p.Worksfor.salary>W_1.salary} W_1) \\ F &= \pi_{pWorksfor.*}(pWorksfor \bowtie_{pWorksfor.cname=W_1.cname \wedge p.Worksfor.salary>W_1.salary} W_1 \\ &\qquad \bowtie_{W_1.cname=W_2.cname \wedge W_1.salary>W_2.salary} W_2) \end{aligned}$$

Then the RA expression for the query is

$$\pi_{Worksfor.cname,Person.pid,Person.name}(E - F).$$

```
-- We can now start optimizing
-- The main rule is to push projection down over joins

with
  pWorksfor as (select w.cname, w.salary, p.pid, p.name
                from   person p natural join worksfor w),
  truncatedWorksFor as (select cname, salary
                        from   worksFor)
select cname, pid, name
from   (select distinct w.*
        from   pWorksfor w
               join truncatedWorksfor w1 on (w.cname = w1.cname and w.salary > w1.salary)
        except
        select distinct w.*
        from   pWorksfor w
               join (select distinct w1.cname, w1.salary
                     from   truncatedWorksfor w1
                            join truncatedWorksfor w2 on (w1.cname = w2.cname and
                                                          w1.salary > w2.salary)) q1
              on (w.cname = q1.cname and  w.salary > q1.salary)) q
order by 1,2,3;
```

We can formulate this optimized RA SQL query as an RA expression in standard notation.

Consider the expressions

$$
\begin{aligned}
pWorksFor &= \pi_{pid,name,cname,salary}(Person \bowtie worksFor) \\
T &= \pi_{cname,salary}(Company) \\
E &= \pi_{pWorksfor.*}(pWorksfor \bowtie_{pWorksfor.cname=W_1.cname \wedge p.Worksfor.salary>W_1.salary} T_1) \\
F &= \pi_{pWorksfor.*}(pWorksfor \bowtie_{pWorksfor.cname=W_1.cname \wedge p.Worksfor.salary>W_1.salary} \\
&\qquad \bowtie_{W_1.cname=W_2.cname \wedge W_1.salary>W_2.salary} (T_1 \bowtie_{W_1.cname=W_2.cname \wedge W_1.salary>W_2.salary} T_2))
\end{aligned}
$$

Then the optimized RA expression for the query is

$$\pi_{Worksfor.cname,Person.pid,Person.name}(E - F).$$

# 3 Example 3

```
-- Find the each job skill that is not the job skill of any person who
-- works for ‘Yahoo’ or for ‘Netflix’.

select js.skill
from   jobskill js
where  js.skill not in (select ps.skill
                        from   personskill ps
                        where  ps.pid in (select w.pid
                                          from   worksfor w
                                          where  w.cname = 'Yahoo') or
                               ps.pid in (select w.pid
                                          from   worksfor w
                                          where  w.cname = 'Netflix'));

-- We will translate this query and then optimize it.

--  Step 1:  Translate 'not in' and 'in' to 'not exists' and 'exists'

select js.skill
from   jobskill js
where  not exists (select ps.skill
                   from   personskill ps
                   where  js.skill = ps.skill and
                          (exists (select w.pid
                                   from   worksfor w
                                   where  ps.pid = w.pid and w.cname = 'Yahoo') or
                           exists (select w.pid
                                   from   worksfor w
                                   where  ps.pid = w.pid and w.cname = 'Netflix')));

-- Step 2:    Push the condition  'w.cname = 'Yahoo' to worksFor
--                Push the condition  'w.cname = 'Netflix' to worksFor
--                This can be done with temporary views
```

```
with
  worksForYahoo as (select * from worksfor where cname = 'Yahoo'),
  worksForNetflix as (select * from worksfor where cname = 'Netflix')
select js.skill
from   jobskill js
where  not exists (select ps.skill
                   from   personskill ps
                   where  js.skill = ps.skill AND
                          (exists (select w.pid
                                   from   worksforYahoo w
                                   where  ps.pid = w.pid) OR
                           exists (select w.pid
                                   from   worksforNetflix w
                                   where  ps.pid = w.pid)));

-- Step 3: we can apply the distribution law of 'AND' over 'OR'
--  I.e., the logical equivalence:  p AND (q OR r) <=> (p AND q) OR (p AND r)

with
  worksForYahoo as (select * from worksfor where cname = 'Yahoo'),
  worksForNetflix as (select * from worksfor where cname = 'Netflix')
select js.skill
from   jobskill js
where  not exists (select ps.skill
                   from   personskill ps
                   where  (js.skill = ps.skill AND
                           (exists (select w.pid
                                    from   worksforYahoo w
                                    where  ps.pid = w.pid)) OR
                          (js.skill = ps.skill AND
                           exists (select w.pid
                                   from   worksforNetflix w
                                   where  ps.pid = w.pid))));


-- Step 4:  Translate the 'OR' into a 'UNION'

with
  worksForYahoo as (select * from worksfor where cname = 'Yahoo'),
  worksForNetflix as (select * from worksfor where cname = 'Netflix')
select js.skill
from   jobskill js
where  not exists (select ps.skill
                   from   personskill ps
                   where  js.skill = ps.skill AND
                          EXISTS (select w.pid
                                  from   worksforYahoo w
                                  where  ps.pid = w.pid)
```

```
                  UNION
                  select  ps.skill
                  from    personskill ps
                  where   js.skill = ps.skill AND
                          EXISTS (select w.pid
                                  from   worksforNetflix w
                                  where  ps.pid = w.pid));

-- Step 5:  We can now eliminate the 'exists' statements

with
  worksForYahoo as (select * from worksfor where cname = 'Yahoo'),
  worksForNetflix as (select * from worksfor where cname = 'Netflix')
select js.skill
from   jobskill js
where  NOT EXISTS (select ps.skill
                  from   personskill ps, worksForYahoo w
                  where  js.skill = ps.skill and ps.pid = w.pid
                  union
                  select  ps.skill
                  from    personskill ps, worksForNetflix w
                  where   js.skill = ps.skill and ps.pid = w.pid);

-- Step 6:  we can now eliminate the 'NOT EXISTS' set predicate
--  by introducing and set difference 'EXCEPT' operation

with
  worksForYahoo as (select * from worksfor where cname = 'Yahoo'),
  worksForNetflix as (select * from worksfor where cname = 'Netflix')
select js.skill
from   jobskill js
EXCEPT
select js.skill
from   jobskill js
where  EXISTS (select ps.skill
               from   personskill ps, worksForYahoo w
               where  js.skill = ps.skill and ps.pid = w.pid
               UNION
               select  ps.skill
               from    personskill ps, worksForNetflix w
               where   js.skill = ps.skill and ps.pid = w.pid);


-- Step 7: notice that stating  'EXISTS( A UNION B)' is equivalent with
--             stating that 'EXISTS(A) OR EXISTS(B)'


with
  worksForYahoo as (select * from worksfor where cname = 'Yahoo'),
```

```
  worksForNetflix as (select * from worksfor where cname = 'Netflix')
select js.skill
from   jobskill js
except
select js.skill
from   jobskill js
where  EXISTS (select ps.skill
               from   personskill ps, worksForYahoo w
               where  js.skill = ps.skill and ps.pid = w.pid) OR
       EXISTS (select  ps.skill
               from    personskill ps, worksForNetflix w
               where   js.skill = ps.skill and ps.pid = w.pid);

-- Step 8:  the 'OR' can be turned again into a 'UNION' because a
--               this works because a projection distributes over an union

with
  worksForYahoo as (select * from worksfor where cname = 'Yahoo'),
  worksForNetflix as (select * from worksfor where cname = 'Netflix')
select js.skill
from   jobskill js
except
( select js.skill
 from   jobskill js
 where  exists (select ps.skill
                from   personskill ps, worksForYahoo w
                where  js.skill = ps.skill and ps.pid = w.pid)
UNION
 select js.skill
 from   jobskill js
 where  exists (select  ps.skill
                from    personskill ps, worksForNetflix w
                where   js.skill = ps.skill and ps.pid = w.pid))
order by 1;

-- Step 9: We now apply the set equivalence A-(B union C) = (A - B) intersect (A - C)


with
  worksForYahoo as (select * from worksfor where cname = 'Yahoo'),
  worksForNetflix as (select * from worksfor where cname = 'Netflix')

(select js.skill
from   jobskill js
EXCEPT
(select js.skill
 from   jobskill js
 where  exists (select ps.skill
                from   personskill ps, worksForYahoo w
```

```
               where  js.skill = ps.skill and ps.pid = w.pid)))
INTERSECT
(select js.skill
from   jobskill js
EXCEPT
( select js.skill
 from   jobskill js
 where  exists (select  ps.skill
                from    personskill ps, worksForNetflix w
                where   js.skill = ps.skill and ps.pid = w.pid)))

-- Step 10:  We now eliminate the 'EXISTS'

with
  worksForYahoo as (select * from worksfor where cname = 'Yahoo'),
  worksForNetflix as (select * from worksfor where cname = 'Netflix')

(select js.skill
from   jobskill js
except
(select js.skill
 from   jobskill js, personskill ps, worksForYahoo w
 where  js.skill = ps.skill and ps.pid = w.pid))
intersect
(select js.skill
from   jobskill js
except
(select js.skill
 from   jobskill js, personskill ps, worksForNetflix w
 where  js.skill = ps.skill and ps.pid = w.pid));


-- Stepp 11: we can now introduce 'join' operations
--                 Actually we can eliminate some variable names as well
-- Notice that the result is a RA SQL query

with
  worksForYahoo as (select * from worksfor where cname = 'Yahoo'),
  worksForNetflix as (select * from worksfor where cname = 'Netflix')

(select skill
from   jobskill
except
(select skill
 from   jobskill
        natural join personskill
        natural join worksForYahoo))
intersect
(select skill
```

```
from   jobskill
except
(select skill
 from   jobskill
        natural join personskill
        natural join worksForNetflix))
```

We can formulate this RA SQL query as an RA expression in standard notation. Consider the expressions

$$\begin{aligned} worksForYahoo &= \sigma_{cname=\texttt{Yahoo}}(worksFor) \\ worksForNetflix &= \sigma_{cname=\texttt{Netflix}}(worksFor) \\ E &= \pi_{skill}(jobSkill \bowtie personSkill \bowtie worksforYahoo) \\ F &= \pi_{skill}(jobSkill \bowtie personSkill \bowtie worksforNetflix) \end{aligned}$$

Then the RA expression for the query is

$$(jobSkill - E) \cap (jobskill - F).$$

```
-- Step 1: We can now start optimizing
-- The main rule is to push 'projections' down over 'join' operations

with
  worksForYahoo as (select pid from worksfor where cname = 'Yahoo'),
  worksForNetflix as (select pid from worksfor where cname = 'Netflix')
(select skill
from   jobskill
except
(select skill
 from   jobskill
        natural join personskill
        natural join worksForYahoo))
intersect
(select skill
from   jobskill
except
(select skill
 from   jobskill
        natural join personskill
        natural join worksForNetflix))

-- Step 2:  we can now apply a foreign key constraint
-- notice that 'skill' in 'personskill' references 'skill' in 'jobskill'
-- This permits us to eliminate the variable 'jobskill'

with
  worksForYahoo as (select pid from worksfor where cname = 'Yahoo'),
  worksForNetflix as (select pid from worksfor where cname = 'Netflix')
(select skill
from   jobskill
```

```
except
(select skill
 from   personskill
        natural join worksForYahoo))
intersect
(select skill
from   jobskill
except
(select skill
 from   personskill
        natural join worksForNetflix))

-- Step 3:  we can then apply the set equality
--  (A-B) intersect (A-C) = A - (B union C)
-- We get the optimized RA SQL expression
-- It is routine to turn this into a RA expression in RA notation

with
  worksForYahoo as (select pid from worksfor where cname = 'Yahoo'),
  worksForNetflix as (select pid from worksfor where cname = 'Netflix')
select skill
from   jobskill
except
(select skill
 from   personskill
        natural join worksForYahoo
 union
 select skill
 from   personskill
        natural join worksForNetflix)
order by 1;
```

We can formulate this optimized RA SQL query as an RA expression in standard notation.

Consider the expressions

$$\begin{aligned} worksForYahoo &= \pi_{pid}(\sigma_{cname=\texttt{Yahoo}}(worksFor)) \\ worksForNetflix &= \pi_{pid}(\sigma_{cname=\texttt{Netflix}}(worksFor)) \\ E &= \pi_{skill}(personSkill \ltimes worksforYahoo) \\ F &= \pi_{skill}(personSkill \ltimes worksforNetflix) \end{aligned}$$

Then the RA expression for the query is

$$jobSkill - (E \cup F).$$