Assignment 6

Pauravi Wagh

3/15/2020

Contents

Question 1:

Original Query:
k = 1, m = 1, n = 1.

```
SELECT L(r1)
FROM R1 r1
WHERE C1(r1) AND r1.A [NOT] IN (SELECT DISTINCT s1.B1
                                FROM S1 s1
                                WHERE C2(s1,r1)
                                [UNION|INTERSECT|EXCEPT]
                                SELECT DISTINCT t1.C1
                                FROM T1 t1
                                WHERE C3(t1,r1))
```

IN case:

```
SELECT DISTINCT L(q.r1)
FROM ((SELECT r1.* FROM R1 r1 WHERE C1(r1)) r1
       NATURAL JOIN
       (SELECT r2.*
       FROM  R1 r2 JOIN S1 s1 ON (C2(s1, r2) and r2.A = s1.B1 )
       [UNION | INTERSECT | EXCEPT ]
       SELECT r3.*
       FROM  R1 r3 JOIN T1 t1 ON (C3(t1, r3) and r3.A = t1.C1)))q;
```

Equivalent RA expression:

$\pi_{L(R_1)}(\sigma_{C_1}(R_1) \bowtie (\pi_{r_2.*}(R_2 \bowtie_{C_2(S_1,R_2)\wedge R_2.A=S_1.B_1} (S_1))[\cup|\cap|-]\pi_{r_3.*}(R_3 \bowtie_{C_3(T_1,R_3)\wedge R_3.A=T_1.C_1} (T_1))))$

In case of union,

$\pi_{L(R_1)}(\sigma_{C_1}(R_1) \bowtie_{C_2(S_1,R_1)\wedge R_1.A=S_1.B_1} S_1) \cup \pi_{L(R_1)}(\sigma_{C_1}(R_1) \bowtie_{C_3(T_1,R_1)\wedge R_1.A=T_1.C_1} T_1)$

General Case:

```
    WITH R_new AS (SELECT r1.*,....,rk.* FROM R1 r1 cross join.... cross join Rk rk),
         S_new AS (SELECT s1.*,....,sm.* FROM S1 s1 cross join.... cross join Sm Sm),
         T_new AS (SELECT t1.*,....,tn.* FROM T1 t1 cross join.... cross join Tn tn),
    SELECT DISTINCT L(q.r1)
    FROM ((SELECT r1.* FROM R_new r1 WHERE C1(r1)) r1
          NATURAL JOIN
          (SELECT r2.*
           FROM R_new r2 join S_new s1 on (C2(s1,r2) and r2.A=s1.B1)
           [UNION | INTERSECT | EXCEPT]
```

```
            (SELECT r3.*
            FROM R_new r3 join T_new t1  on (C3(t1,r3) and r3.A=t1.C1))))q
```

RA Expression:

$$\pi_{L(R_1 \ldots R_k)}(\sigma_{C_1}(R_1) \bowtie (R_2 \bowtie_{C_2(S,R_2) \wedge R_2.A=S.B_1} (S)[\cup| \cap |-]R_3 \bowtie_{C_3(T,R_3) \wedge R_3.A=T.C_1} (T)))$$

where

$$R = R_1 \times ... \times R_k$$

$$S = S_1 \times ... \times S_m$$

$$T = T_1 \times ... \times T_n$$

In case of union,

$\pi_{L(R)}((\sigma_{C_1}(R) \bowtie_{C_2(S,R) \wedge R.A=S.B_1} S) \cup \pi_{L(R)}(\sigma_{C_1}(R) \bowtie_{C_3(T,R) \wedge R.A=T.C_1} T))$

NOT IN Case:
k = 1, m = 1, n = 1.

```
SELECT DISTINCT L(q.r1)
FROM ((SELECT r1.* FROM R1 r1 WHERE C1(r1)) r1
       EXCEPT
       (SELECT L(q1.r2)
        FROM (SELECT r2.*
           FROM R1 r2 join S1 s1 on (C2(s1,r2) and r2.A=s1.B1)
           [UNION | INTERSECT | EXCEPT]
           SELECT r3.*
           FROM R1 r3 join T1 t1  on (C3(t1,r3) and r3.A=t1.C1))q1))q
```

RA Expression:

$\pi_{L(R_1)}(\sigma_{C_1}(R_1) - \pi_{r_2.*}(\pi_{r_2.*}(R_2 \bowtie_{(C_2(s_1,r_2) \wedge r_2.A=s1.B1)} S_1)[\cup| \cap |-]\pi_{r_3.*}(R_3 \bowtie_{(C_3(t_1,r_3) \wedge r_3.A=t1.C1)} T_1)))$

General case:

```
WITH R_new AS (SELECT r1.*,....,rk.* FROM R1 r1 cross join.... cross join Rk rk),
     S_new AS (SELECT s1.*,....,sm.* FROM S1 s1 cross join.... cross join Sm Sm),
     T_new AS (SELECT t1.*,....,tn.* FROM T1 t1 cross join.... cross join Tn tn),
    SELECT DISTINCT L(q.r1)
    FROM ((SELECT r1.* FROM R_new r1 WHERE C1(r1)) r1
           EXCEPT
           (SELECT L(q1.r2)
            FROM (SELECT r2.*
                  FROM R_new r2 join S_new s1 on (C2(s1,r2) and r2.A=s1.B1)
                  [UNION | INTERSECT | EXCEPT]
                  SELECT r3.*
                  FROM R_new r3 join T_new t1  on (C3(t1,r3) and r3.A=t1.C1))q1)
                 )q
```

RA Expression:

$$\pi_{L(R_1)}(\sigma_{C_1}(R_1) - \pi_{r_2.*}(\pi_{r_2.*}(R_2 \ltimes_{C_2(S,R_2)\wedge r_2.A=s.B1} (S))[\cup|\cap|-]\pi_{r_3.*}(R_3 \ltimes_{C_3(T,R_3)\wedge r_3.A=t.C1} (T))))$$

where

$$R = R_1 \times ... \times R_k$$

$$S = S_1 \times ... \times S_m$$

$$T = T_1 \times ... \times T_n$$

Question 1.b)

```
SELECT L(r1)
FROM R1 r1
WHERE C1(r1) AND r1.A [NOT] IN (SELECT DISTINCT s1.B1
                                FROM S1 s1
                                WHERE C2(s1,r1)
                                [UNION|INTERSECT|EXCEPT]
                                SELECT DISTINCT t1.C1
                                FROM T1 t1
                                WHERE C3(t1,r1))
```

IN Case:

If we remove the R1 condition from the C2 and C3, then we no longer need joins, thus we can just alias the columns to perform a natural join with relation R

```
  SELECT DISTINCT L(q.r1)
  FROM ((SELECT r1.* FROM R1 r1 WHERE C1(r1)) r1
        NATURAL JOIN
       (SELECT s1.B1 AS A
        FROM   S1 s1
        WHERE C2(s1)
        [UNION | INTERSECT | EXCEPT ]
        SELECT t1.C1 AS A
        FROM  T1 t1
        WHERE C3(t1))q;
```

RA Expression:

$$\pi_{L(r_1)}(\pi_{r_1.*}(\sigma_{C_1}(R_1)) \bowtie (\pi_{s_1.A}(\sigma_{C_2}(S_1))[\cup|\cap|-]\pi_{t_1.A}(\sigma_{C_3}(T_1))))$$

NOT IN CASE:

```
SELECT DISTINCT L(q.r1)
FROM ((SELECT r1.* FROM R1 r1 WHERE C1(r1)) r1
       EXCEPT
       SELECT q1.r2
       FROM ((SELECT r2.* FROM R1 r2 WHERE C1(r2)) r2
       NATURAL JOIN (SELECT s1.B1 AS A
                     FROM   S1 s1
```

```
                        WHERE C2(s1)
                        [UNION | INTERSECT | EXCEPT ]
                        SELECT t1.C1 AS A
                        FROM  T1 t1
                        WHERE C3(t1) )q1)q
```

RA Expression:

Let,

$$R_1 = \pi_{r_1.*}(\sigma_{C_1}(R_1))$$
$$S_1 = \pi_{s_1.*}(\sigma_{C_2}(S_1))$$
$$T_1 = \pi_{t_1.*}(\sigma_{C_3}(T_1))$$
$$\pi_{L(r_1)}(R_1 - \pi_{r_2.*}(R_1 \bowtie (S_1[\cup|\cap|-]T_1)))$$

Question 2:

To prove $\pi_a(R \bowtie_{r.a=s.b \wedge r.b=s.a} S) = \pi_a(\pi_{a,b}(R) \cap \pi_{b,a}(S))$

$\pi_a(R \bowtie_{r.a=s.b \wedge r.b=s.a} S) = \{(a|\exists b \exists c \exists d(R(a,b,c) \wedge r.a = s.b \wedge r.b = s.a \wedge S(a,b,d)))\}$

$= \{(a|\exists b \exists c(R(a,b,c) \wedge r.a = s.b \wedge r.b = s.a \wedge (\exists d S(a,b,d))))\}$

$= \{(a|\exists b((\exists c R(a,b,c)) \wedge r.a = s.b \wedge r.b = s.a \wedge (\exists d S(a,b,d)))\}$

$= \{(a|\exists b((a,b) \in \pi_{a,b}(R) \wedge r.a = s.b \wedge r.b = s.a \wedge (b,a) \in \pi_{b,a}(S))\}$

$= \{(a|\exists b((a,b) \in \pi_{a,b}(R) \cap (b,a) \in \pi_{b,a}(S))\}$

$= \pi_a(\pi_{a,b}(R) \cap \pi_{b,a}(S))$

Question 3:

(a.) Translated Query :

Translated Query:

```
with CSmajor as (select m.sid from major m where m.major = 'CS')
select distinct s.sid, s.sname
from (student s natural join CSmajor natural join buys t)
join cites c on (c.bookno = t.bookno)
join book b1 on (b1.bookno = c.bookno )
join book b2 on (b2.bookno = c.citedbookno and b1.price < b2.price)
```

Let

$$E = (S \bowtie \sigma_{major='CS'}(M) \bowtie (T)) \bowtie_{t.bookno=c.bookno} (C) \bowtie_{c.bookno=b1.bookno} (B_1)$$

The query then becomes,

$$\pi_{sid,sname}(E \bowtie_{c.citedbookno=b_2.bookno \wedge b1.price<b2.price} (B_2))$$

(b.) Optimized Query:

```
  with
  csmajor as
      (select m.sid from major m where m.major = 'CS'),
  T as
      (select distinct s.sid, s.sname, b.bookno
       from student s
            natural join csmajor c
            natural join buys b),
  B as
      (select distinct bookno, price from book),
  C as
      (select c.bookno
       from cites c
       join book b1 on (c.bookno=b1.bookno)
       join book b2 on (c.citedbookno = b2.bookno and b1.price<b2.price))

  select distinct t.sid, t.sname
  from T natural join  C
  order by 1,2;
```

RA Expression:

$CS = \pi_{sid}(\sigma_{major='CS'}(Major))$

$T = \pi_{sid,sname,bookno}(S \bowtie CS \bowtie Buys)$

$B = \pi_{bookno,price}(Book)$

$C = \pi_{cites.bookno}(Cites \bowtie_{cites.bookno=b1.bookno} (B_1) \bowtie_{cites.citedbookno=b_2.bookno \wedge b_1.price<b_2.price} (B_2))$

The optimized expression is

$$\pi_{sid,sname}(T \bowtie C)$$

Question 4:

Question 4.a

Translated Query:

```
        select distinct q.sid, q.sname, q.major
      from (select distinct s.sid, s.sname, m.major
            from student s natural join major m natural join buys t
            join book b on (t.bookno = b.bookno and b.price < 60 )
            except
            select distinct q1.sid, q1.sname, q1.major
            from (select distinct s.sid, s.sname, m.major, t.bookno, b.title, b.price
                  from student s natural join major m natural join buys t
                  join book b on (t.bookno = b.bookno and b.price < 60 )
                  join major m1 on (m1.sid = s.sid and m1.major = 'CS')
                  union
                  select distinct s.sid, s.sname, m.major, t.bookno, b.title, b.price
                  from student s natural join major m natural join buys t
                  join book b on (t.bookno = b.bookno and b.price < 60 )
                  join buys t1 on(s.sid = t1.sid)
                  join book b1 on (b1.bookno = t1.bookno and b1.price < 30)
            )q1)q;
```

Translated RA Expression:

Let $E = \pi_{s.sid,s.sname,m.major}((S \bowtie M \bowtie T) \bowtie_{t.bookno=b.bookno \wedge b.price<60} (B))$

The equation now becomes,
$\pi_{sid,sname,major}(E-$

$\pi_{e.sid,e.sname,e.major}$
$(\pi_{e.sid,e.sname,e.major,e.bookno,b.title,b.price}(E \bowtie_{m1.sid=s.sid \wedge m1.major='CS'} (M_1))$
$\cup$
$\pi_{e.sid,e.sname,e.major,e.bookno,b.title,b.price}(E \bowtie_{e.sid=t_1.sid} (T_1) \bowtie_{b_1.bookno=t_1.bookno \wedge b_1.price<30} (B_1)))$

4.b)
Optimized Query:

```
with
CSMajors as (select m.sid from major m where m.major = 'CS'),
booklessthan60 as (select b.bookno, b.price from book b where b.price < 60),
booklessthan30 as (select b.bookno, b.price from book b where b.price < 30),
commonjoin as (select s.sid, s.sname, m.major, t.bookno from student s natural join major m natural
join buys t)

select q.sid, q.sname, q.major
from (select distinct s.sid, s.sname, s.major
      from commonjoin s natural join booklessthan60 b
      except
      select distinct q1.sid, q1.sname, q1.major
      from (select distinct s.sid, s.sname, s.major
            from commonjoin s
            natural join CSMajors m1

            union

            select distinct s.sid, s.sname, s.major
            from commonjoin s
            natural join booklessthan30 b1
      )q1)q;
```

Optimized RA Expression:

Let
$CS = \pi_{sid}(\sigma_{major='CS'}(M))$

$B60 = \pi_{bookno,price}(\sigma_{price<60}(B))$

$B30 = \pi_{bookno,price}(\sigma_{price<30}(B))$

$CJ = \pi_{s.sid,s.sname,s.major,t.bookno}(S \bowtie M \bowtie T)$

The optimized query is,

$\pi_{sid,sname,major}(\pi_{s.sid,s.sname,s.major}(CJ \bowtie B60) - \pi_{sid,sname,major}(\pi_{sid,sname,major}(CJ \bowtie CS) \cup \pi_{sid,sname,major}(CJ \bowtie B30)))$

Question 5:

Equivalent query:

5.a)

```
select distinct q.sid, q.sname, q.bookno
from (select distinct s.*, t.sid as sd, t.bookno as bno, b.*
      from student s natural join buys t natural join book b
      except
      select distinct s.*, t.sid as sd, t.bookno as bno, b.*
      from student s natural join buys t natural join book b
      join (book b1 natural join buys t1)
      on (b.price < b1.price and s.sid = t1.sid ))q order by 1;
```

Translated Query:

$\pi_{sid,sname,bookno}(\pi_{s.*,t.*,b.*}(S \bowtie T \bowtie B) - \pi_{s.*,t.*,b.*}((S \bowtie T \bowtie B) \bowtie_{b.price<b_1.price \wedge t.sid=t_1.sid} (B_1 \bowtie T_1)))$

5.b)

Optimized query:

```
with B as (select bookno, price from book)

select distinct q.sid, q.sname, q.bookno
from (select s.sid, s.sname, t.bookno
      from student s natural join buys t natural join B b
      except
      select s.sid, s.sname, t.bookno
      from student s natural join buys t natural join book b
      join (B b1 natural join buys t1)
      on (b.price < b1.price and s.sid = t1.sid ))q
order by 1;
```

Let

$$B = \pi_{bookno,price}(Book)$$

The Final Query is :

$\pi_{sid,sname,bookno}(\pi_{s.sid,s.sname,t.bookno}(S \bowtie T \bowtie B) - \pi_{s.sid,s.sname,t.bookno}(S \bowtie T \bowtie B) \bowtie_{b.price<b_1.price \wedge s.sid=t_1.sid} (B_1 \bowtie T_1))$

Question 6:

6.a)
Translated Query:

```
with MATHCS as (select m.sid from major m
                where m.major = 'CS'
                UNION
                select m.sid from major m
                where m.major = 'Math')

select distinct q.bookno, q.title
from (select distinct b.*, s.*
      from book b cross join student s natural join  mathcs c
      except
      select distinct b.*, s.*
      from student s natural join mathcs c natural join buys t natural join book b
      )q order by 1;
```

Let $U = \pi_{sid}(\sigma_{major='CS'}(M) \cup \sigma_{major='Math'}(M))$

Translated Query is:

$\pi_{bookno,title}(\pi_{b.*,s.*}((B \times S) \bowtie U) - \pi_{b.*,s.*}(S \bowtie U \bowtie T \bowtie B))$

6.b)

Optimized version :

```
  with MATHCS as (select m.sid from major m
                  where m.major = 'CS'
```

```
                UNION
                select m.sid from major m
                where m.major = 'Math'),
      B as (select bookno, title from book b)

 select distinct q.bookno, q.title
 from (select distinct b.bookno, b.title, c.sid
       from B b cross join mathcs c
       except
       select distinct b.bookno, b.title, t.sid
       from  buys t natural join B b
       )q order by 1;
```

We have removed the student table from all the queries as it had a foreign key relations with major table and buys table.
Final query is :

Let $U = \pi_{sid}(\sigma_{major='CS'}(M) \cup \sigma_{major='Math'}(Major))$

Let $B = \pi_{bookno,title}(Book)$

Let $T = Buys$

The final query is:

$$\pi_{bookno,title}(\pi_{b.bookno,b.title,u.sid}(B \times U) - \pi_{b.bookno,b.title,u.sid}(T \bowtie B))$$

Question 7:

7.a)
Optimized query (Q_4) :

```
with R3 as (select distinct r.a as b from r r),
R2 as (select distinct r2.a as b from R r2 natural join R3)
select distinct r1.a
from R r1 natural join r2;
```

7.b)
Comparison:

makerandomR	Q3 (in ms)	Q4 (in ms)
(1000, 1000, 1000)	48	43
(1000, 10000, 10000)	89	74
(3000,3000,20000)	273	80
(2500,2500,40000)	2303	100
(2000,2000,400000)	—	344

7.c)
We can see that as we increase the size of the relation R, the non-optimized query works poorly whereas the optimized query gives us results in a very short duration. We can see that in Q_3, there are 2 cross join operations, which means the running time is in order of $O(|R|^3)$. On the other hand, the running time of query Q_4 is in the order $O(|R|)$ Using natural joins reduced the time needed to execute each query, but more than that use of the keyword distinct reduced the time significantly. Q3 query had 3 nested loops, where as the Q4 query executes in linear time, thus this reduced the time significantly.

Question 8:

8.a)

Translated and optimized query Q_6:

```
select q.a
from (select distinct ra.a
      from Ra ra
      except
      select q1.a
      from (select r.a, r.b
            from R r
            except
            select distinct r.a, r.b
            from r r natural join s s
            )q1
      )q;
```

8.b)
Comparison:

makerandomR	makerandomS	Q5(in ms)	Q6(in ms)
(1000,1000,10000)	(1000,100)	58	48
(1000,1000,10000)	(1000,1000)	54	57
(2000,2000,40000)	(2000, 200)	61	79
(2000,2000,40000)	(2000, 1800)	70	100
(5000,5000,80000)	(5000, 500)	77	117
(5000,5000,80000)	(5000, 5000)	87	148
(8000,8000,100000)	(8000, 100)	90	140
(8000,8000,100000)	(8000, 8000)	68	174

8.c)
We can see here, that when the size of relations R and S is not that huge, then the optimized and the non-optimized query work similarly. But when the size is increased, the optimized query works poorly than the non-optimized one. The post-gres internal system finds a way to solve the non-optimized query in a way that we get better results for it than the optimized one. So, clearly optimization does not always work.

Question 9:

9.a)
In this query, we realize that the attribute A in R will always have values less than the values in Ra, thus when we perform a natural join operation of R with Ra, we will always get R. Thus, we reduced our query from $Ra \bowtie R$ to just R.

Translated and Optimized Query Q_8:

```
select distinct q.a
from (select ra.a
from Ra ra
except
select q1.a
from (select  ra.a, s.b
     from ra ra cross join s s
     except
     select r.a, r.b
     from  r r natural join s s)q1)q;
```

9.b)

makerandomR	makerandomS	Q7(in ms)	Q8(in ms)
(500,500,2000)	(500,100)	139	63
(500,500,2000)	(500,500)	98	213
(1000,1000,5000)	(1000,200)	285	146
(1000,1000,5000)	(1000,800)	282	645
(2000,2000,6000)	(2000,200)	687	249
(2000,2000,6000)	(2000,1000)	646	1650
(5000,5000,10000)	(5000,1000)	2610	2863
(5000,5000,10000)	(5000,5000)	2604	27068

9.c)
We see a different pattern here than all the previous examples. As we can see in the table, the size of relation R an S goes on increasing as we move down in the table. Along with that, I have varied the value of S for the fixed values of m, n, and l of R. On using such data, I observed that, when the value of S is small for the fixed values of l,m and n of R , the optimized query works well and takes less running time, whereas the non-optimized query has a higher running time. But as we increase the limit of the relation S by keeping other values constant, we see that optimized query performs badly. This is because of the presence of a cross join operation which takes the running time to $O(|R|^2)$.

9.d)
When we compare the running times of optimized queries for the ALL and ONLY queries, we find that running time for the ALL query increases faster than that of the ONLY query. This difference is clear as we increase the sizes of relations R and S. This can be attributed to the presence of the cross join in the ALL query. Although both the optimized queries do not work well on larger data than their corresponding non-optimized versions, still we can say that the ONLY query isn't as bad as the ALL query. The presence of a $O(|R|^2)$ term in the running time of the ALL query degrades its performance on larger relations.

Question 10

Working of the Query:
Our goal here is to find ONLY those values of A attribute in relation R, whose corresponding B values are also present in S. So, using object relational model, we check the set containment to see if all the B values pertaining to each A attribute is present in S or not.. In this query, we firstly aggregate all the values in the B attribute corresponding to the values of A attributes by grouping them from relation R. Then we make an array of all the values in the Relation S. We select ONLY those A attributes from R whose corresponding B values are contained in the array of S values. Along with that, we select those A attributes, which have no corresponding B attribute using the UNION operation.

10.a)

makerandomR	makerandomS	Q5(in ms)	Q6(in ms)	Q9(in ms)
(1000,1000,10000)	(1000,100)	58	48	54
(1000,1000,10000)	(1000,1000)	54	57	58
(2000,2000,40000)	(2000, 200)	70	74	69
(2000,2000,40000)	(2000, 1800)	61	80	118
(5000,5000,80000)	(5000, 500)	71	95	117
(5000,5000,80000)	(5000, 5000)	65	132	517
(8000,8000,100000)	(8000, 100)	90	140	104
(8000,8000,100000)	(8000, 8000)	68	174	1001
(10000,10000,1000000)	(10000, 100)	380	970	568
(10000,10000,1000000)	(10000, 10000)	219	1346	10583

10.b)
Looking at the table above, We can see that query Q_5, the non-optimized one, works the best. For smaller

size of relations R and S, all the three queries are comparable, but for bigger relations, we can see a lot of difference expecially in case of object relational query. The optimized version Q_6 does not perfrom better than Q_5. And the query Q_9 works the worst of the lot. Optimization and object-relational query, both perform worse than the non-optimized query.

Question 11)

Working: In this query, we aggregate all the B attributes in R grouped by the A attribute. Thus, we have for each value of A in R, all the values of B associated with it. Then we aggregate all the S values in one array. We now select such A attributes in R, whose B values contain **ALL** the values in the aggregation of S. Thus if the aggreation of B values in R contains all the values in S, then the corresponding A attributes will be chosen.

11.a)
Comparison:

makerandomR	makerandomS	Q7(in ms)	Q8(in ms)	Q10(in ms)
(500,500,2000)	(500,100)	139	63	44
(500,500,2000)	(500,500)	98	213	43
(1000,1000,5000)	(1000,200)	285	146	47
(1000,1000,5000)	(1000,800)	282	645	47
(2000,2000,6000)	(2000,200)	687	249	46
(2000,2000,6000)	(2000,1000)	646	1650	45
(5000,5000,10000)	(5000,1000)	2610	3863	69
(5000,5000,10000)	(5000,5000)	2604	27068	49

11.b)
Comparison:

makerandomR	makerandomS	Q8(in ms)	Q10(in ms)
(2500,2500,8000)	(2500,200)	297	61
(2500,2500,8000)	(2500,500)	703	57
(2500,2500,8000)	(2500,2000)	2862	60
(3000,3000,8000)	(3000,300)	558	58
(3000,3000,8000)	(3000,2000)	3462	62
(5000,5000,2500000)	(5000,4000)	26414	1457

11.c)
I think that when we use the object relational query, the time reduces considerably. Q_7 and Q_8 perform worse, but Q_{10} works exceptionally well. The method of aggregating the values has worked really well for this question.Query Q_7 is optimized by postgres, and it finds out a better way to optimize than the optimization done by us in query Q_8. The Query Q_8 does not scale well at all.
By running additional experiments on Q_8 and Q_{10}, we realize that, Q_{10} is way more efficient than Q_8. Presence of a cross join i.e.(running time in the order $O(|R|^2)$), really degrades the performance and cannot be put to use practically. Even, with such less number of attributes if we get such poor results, it is not prudent to use Q_8 in practical applications. In the query Q_{10}, we use aggregation, which reduces the running time by a lot.

Thus overall, owing to the structure of each query, we see that object relational query works the best here.