Theory_part_Assignment 6

Vibhas Vats

4/01/2020

Contents

Part 1: Theoratical questions

Question 1.a

IN case

For m = n = k = 1 case

```
select L(r1)
from R1 r1
where C1(r1) and r1.A 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));
```

converting to equivalent RA SQL:

```
select L(q.r1)
from ((select r1.*
     from R1 r1
     where C1(r1)) r1
     natural join (select distinct r2.*
                from S1 s1 join R1 r2 on (C2(s1,r2) and r2.A = s1.B1)
                [union | except | intersect]
                select distinct r3.*
                from T1 t1 join R1 r3 on (C3(t1, r3) and r3.A = t1.c1)) q2) q;
```

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 we can do better

$$\pi_{L(R_1)}(\sigma_{C_1}(R_1) \bowtie_{C_2(S_1,R_1) \wedge R_1.A=S_1.B_1} S_1) \cup \pi_{L(R_1)}((\sigma_{C_1}(R_1) \bowtie_{C_3(T_1,R_1) \wedge R_1.A=T_1.c_1} T_1))$$

Generalized case

```
with R as (select r1.*, ......rk.*
          from R1 r1 cross join .....cross join Rk rk),
    S as (select s1.*,.... sm.*
          from S1 s1 cross join ..... cross join Sm sm),
    T as (select t1.*,.... tn.*
          from T1 t1 cross join ..... cross join Tn tn)
select L(r_1...r_k)
from ((select r1.*
      from R r1
      where C1(R1)) q1
      natural join (select r2.*
                    from S s1 join R r2 on (C2(S1,R2) and R2.A = S1.B1)
                    [Union | Except | intersect]
                    select r3.*
                    from T t1 join R r3 on (C3(T1, R3) and T1.C1 = R3.A)) q2) q;
```

$\mathbf{R} = R_1 \times ..... \times R_K$
$\mathbf{S} = S_1 \times ..... \times S_m$
$\mathbf{T} = T_1 \times ..... \times T_n$

RA Expression:

$$\pi_{L(r_1 ... r_k)}(\sigma_{C_1}(\mathbf{R_1}) \bowtie (\pi_{R_2.*}(\mathbf{R_2} \bowtie_{C_2(S_1,R_2) \wedge R_2.A = S_1.B_1} \mathbf{S_1}[\cup| \cap |-]\pi_{R_3.*}(\mathbf{R_3} \bowtie_{C_3(T_1,R_3) \wedge R_3.A = T_1.C_1} \mathbf{T_1}))))$$

In case of **union** we can do better:

$$\pi_{L(r_1 ... r_k)}(\sigma_{C_1}(\mathbf{R}) \bowtie_{C_2(S,R) \wedge R.A = S.B_1)} \mathbf{S} \cup \pi_{L(r_1 ... r_k)}(\sigma_{C_1}(\mathbf{R}) \bowtie_{C_3(T,R) \wedge R.A = T.C_1} \mathbf{T})))$$

NOT IN case

For m = n = k = 1 case

```
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));
```

converting to equivalent RA SQL:

```
select L(q.r1)
from (select r1.*
      from R1 r1
      where C1(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:

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

Generalized case

```
with R as (select r1.*, ......rk.*
           from R1 r1 cross join .....cross join Rk rk),
    S as (select s1.*,.... sm.*
           from S1 s1 cross join ..... cross join Sm sm),
    T as (select t1.*,.... tn.*
           from T1 t1 cross join ..... cross join Tn tn)
select L(r_1...r_k)
from (select R1.*
      from R r1
      where C1(r1)
      except
      select L(q1.r2)
      from (select r2.*
```

```
            from S s1 join R r2 on (C2(s1,r2) and r2.A = s1.B1)
            [union | intersect | except]
            select r3.*
            from R r3 join T t1 on (C3(t1, r3) and r3.A = t1.C1)) q1) q;
```

$\mathbf{R} = R_1 \times ..... \times R_K$
$\mathbf{S} = S_1 \times ..... \times S_m$
$\mathbf{T} = T_1 \times ..... \times T_n$

RA Expression:

$$\pi_{L(r_1 ... r_k)}(\sigma_{C_1}(\mathbf{R_1}) - \pi_{R_2.*}(\pi_{R_2.*}(\mathbf{R_2} \bowtie_{C_2(S_1,R_2) \wedge R_2.A = S_1.B_1} \mathbf{S_1})[\cup|\cap|-]\pi_{R_3.*}(\mathbf{R_3} \bowtie_{C_3(T_1,R_3) \wedge R_3.A = T_1.C_1} \mathbf{T_1})))$$

Question 1.b

given query: IN Case

```
SELECT L(r)
FROM R1 r1
WHERE C1(r1) IN r1.A (SELECT DISTINCT s1.B1
                            FROM S1 s1
                            WHERE C2(s1)
                            [UNION|INTERSECT|EXCEPT]
                            SELECT DISTINCT t1.C1
                            FROM T1 t1
                            WHERE C3(t1))
```

Translated query:

```
with
  New_R1 as (select r1.*
             from R1 r1
             where C1(r1)),
  New_S1 as (select s1.B1 as A
             from S1 s1
             where C2(s1)),
  New_T1 as (select t1.c1 as A
             from T1 t1
             where C3(t1))
select L(q.r1)
from ((select r1.*
     from New_R1 r1) r1
     natural join (select distinct s1.A
                   from New_S1 s1
                   [union | except | intersect]
                   select distinct t1.A
                   from New_T1 t1) q2) q;
```

Explanation: When there is no r1 present in C2 and C3 conditions then we can take advantage of natural join which is very fast improves the translation. The RA expression becomes:
RA Expression:
New_R1 $\rightarrow R1 = \pi_{r1.*}(\sigma_{C_1}(R_1))$
New_S1 $\rightarrow S1 = \pi_{s1.A}(\sigma_{C_2}(S_1))$
in above query I have renamed attribute s1.B1 as s1.A
New_T1 $\rightarrow T1 = \pi_{t1.A}(\sigma_{C_3}(T_1))$ in above query I have renamed attribute t1.C1 as t1.A

Final RA expression:

$$\pi_{R_1}(R_1 \bowtie (\pi_{s_1.A}(S_1)[\cup| \cap |-]\pi_{t_1.A}(T_1)))$$

given query: NOT IN Case

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

Translated query:

```
with
New_R1 as (select r1.*
        from R1 r1
        where C1(r1)),
New_S1 as (select s1.B1 as A
        from S1 s1
        where C2(s1)),
New_T1 as (select t1.c1 as A
        from T1 t1
        where C3(t1))
select L(q.r1)
from (select r1.*
    from New_R1 r1
    except
    select r1.*
    from New_R1 r1 natural join (select s1.A
                                 from New_S1 s1
                                 [union | intersect | except]
                                 select t1.A
                                 from New_T1 t1) q1) q;
```

Explanation: When there is no r1 present in C2 and C3 conditions then we can take advantage of natural join which is very fast improves the translation. The RA expression becomes:
New_R1 $\rightarrow R1 = \pi_{r1.*}(\sigma_{C_1}(R_1))$
New_S1 $\rightarrow S1 = \pi_{s1.*}(\sigma_{C_2}(S_1))$
New_T1 $\rightarrow T1 = \pi_{t1.*}(\sigma_{C_3}(T_1))$

$$\pi_{L(R_1)}(R_1 - \pi_{R_2.*}(R_2 \bowtie (S_1[\cup| \cap |-]T_1)))$$

question 2

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

$$\{(a)|\exists_b \exists_c \exists_d((R|a,b,c) \wedge R.a = S.b \wedge R.b = S.a \wedge (S|a,b,d))\}$$

$$\{(a)|\exists_b \exists_c((R|a,b,c) \wedge R.a = S.b \wedge R.b = S.a \wedge \exists_d(S|a,b,d))\}$$

$$\{(a)|\exists_b(\exists_c(R|a,b,c) \wedge R.a = S.b \wedge R.b = S.a \wedge (\exists_d(S|a,b,d)))\}$$

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

Since, we are checking equality condition on (R.a, S.b) and (R.b, S.a), and in above expression we have projected attribute 'b' ahead of attribute 'a', so now, we can just take intersection to get the required result

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

part 2 : Translating and Optimizing SQL Queries to Equivalent RA Expressions

Question 3

Abbreviation:
student → S
Book → B
Buys → T
cites → C
Major → M

Equivalen SQL query:

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

Equivalent RA expression:

$$\pi_{sid,sname}((S \bowtie \sigma_{major=CS}(M) \bowtie T) \bowtie_{t.bookno=c.bookno} C \bowtie_{c.bookno=b_1.bookno} B_1 \bowtie_{c.citedbookno=b_2.bookno \wedge b1.price<b2.price} B_2)$$

optimised SQL query:

```
with
CSMajor as (select sid
            from major
            where major = 'CS'),
StudentCSMajorBuys as (select distinct s.sid, s.sname, t.bookno
                       from student s
                          natural join CSMajor m
                          natural join buys t),
Books as (select bookno, price
          from book),
CitedBooks as (select c.bookno
     from cites c
     natural join Books b1
     join Books b2 on (c.citedbookno = b2.bookno and b1.price < b2.price))
select distinct t.sid, t.sname
from StudentCSMajorBuys t
natural join CitedBooks c
order by 1,2;
```

Equivalent RA expression:

CSMajor $\rightarrow CS = \pi_{sid}(\sigma_{major='CS'}(M))$
StudentCSMajorBuys $= \pi_{sid,sname,bookno}(S \bowtie CS \bowtie T)$
Books $\rightarrow BK = \pi_{bookno,price}(B)$
CitedBooks $= \pi_{c.bookno}(C \bowtie BK_1 \bowtie_{c.citedboono=b_2.bookno \wedge b1.price<b2.price} BK_2)$

final expression:

$$\pi_{sid,sname}(StudentCSMajorBuys \bowtie CitedBooks)$$

Question 4

Equivalen SQL query:

```
select q.sid, q.sname, q.major
from (select 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 q2.sid, q2.sname, q2.major
     from (select 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)
           join major m2 on (s.sid = m2.sid and m2.major = 'CS')
          union
          select 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)
        join buys t2 on (t2.sid = s.sid)
        join book b2 on (t2.bookno = b2.bookno and b2.price < 30) ) q2) q;
```

Equivalent RA expression:

$\pi_{sid,sname,major}(\pi_{sid,sname,major}(S \bowtie M \bowtie T \bowtie_{t.bookno=b.bookno \wedge b.price<60} B) - \pi_{sid,sname,major}((S \bowtie M \bowtie T \bowtie_{t.bookno=b.bookno \wedge b.price<60} B \bowtie_{s.sid=m_2.sid \wedge m_2.major=CS} M_2) \cup \pi_{sid,sname,major}(S \bowtie M \bowtie T \bowtie_{t.bookno=b.bookno \wedge b.price<60} B \bowtie_{t_2.sid,s.sid} T_2 \bowtie_{t_2.bookno=b_2.bookno \wedge b_2.price<30} B_2)))$

optimised SQL query:

```
with CSMajor as( select m.sid from major m where m.major = 'CS'),
    bookLessThan60 as (select b.bookno from book b where b.price < 60),
    bookLessThan30 as (select b.bookno from book b where b.price < 30),
    StudentMajorbuys as (select s.sid, s.sname, m.major, t.bookno
                         from Student s
                         natural join Major m
                         natural join buys t),
    Q1 as (select smb.sid, smb.sname, smb.major
          from StudentMajorbuys smb
           natural join bookLessThan60 b ),
    Q21 as (select smb.sid, smb.sname, smb.major
           from StudentMajorbuys smb
```

```
            natural join CSmajor m2),
    Q22 as (select smb.sid, smb.sname, smb.major
           from StudentMajorbuys smb
            natural join BookLessThan30 b)
select q.sid, q.sname, q.major
from (select q1.sid, q1.sname, q1.major
     from Q1 q1
     except
     (select q21.sid, q21.sname, q21.major
     from Q21 q21
     union
     select q22.sid, q22.sname, q22.major
     from Q22 q22 ))q;
```

optimised RA expression:
CSmajor $\rightarrow CS = \pi_{sid}(\sigma_{major='CS'}(M))$
BookLessThan60 $\rightarrow B60 = \pi_{bookno}(\sigma_{price<60}(B))$
BookLessThan30 $\rightarrow B30 = \pi_{bookno}(\sigma_{price<30}(B))$
StudentMajorbuys $\rightarrow SMB = \pi_{sid,sname,major}(S \bowtie M \bowtie T)$
Q1 $\rightarrow Q_1 = \pi_{sid,sname,major}(SMB \bowtie B60)$
Q21 $\rightarrow Q_{21} = \pi_{sid,sname,major}(SMB \bowtie CS)$
Q22 $\rightarrow Q_{22} = \pi_{sid,sname,major}(SMB \bowtie B30)$

Main query:

$$\pi_{sid,sname,major}(Q_1 - (Q_{21} \cup Q_{22}))$$

The above query can further be re-written using Relativized De-Morgan for $\cup$ as:

$$\pi_{sid,sname,major}((Q_1 - Q_{21}) \cap (Q_1 - Q_{22}))$$

But both optimization should work in same order of execution time as set intersection, set difference and set unions are linear time operation.

Question 5

Equivalent SQL query:

```
select distinct q.sid, q.sname, q.bookno
from (select s.sid, s.sname, b.*
     from student s natural join buys t natural join book b
     except
     select s.sid, s.sname, b.*
     from student s natural join buys t natural join book b
     join buys t2 on (s.sid = t2.sid)
     join book b2 on (b2.bookno = t2.bookno and b.price < b2.price)) q order by 1;
```

Equivalent RA expression: $\pi_{sid,sname,bookno}(\pi_{sid,sname,b.*}(S \bowtie T \bowtie B) - \pi_{sid,sname,b.*}(S \bowtie T \bowtie B \bowtie_{s.sid=t2.sid} T_2 \bowtie_{b_2.bookno=t_2.bookno \wedge b.price<b_2.price} B_2))$

optimised SQL query:

```
with Books as (select b.bookno, b.price
               from book b),
    StudentBook as (select distinct s.sid, s.sname, b.bookno, b.price
```

```
               from student s
                    natural join buys t
                    natural join Books b),
    BuysBooks as (select t.sid, t.bookno, b.price
                  from buys t
                  natural join Books b)
select q.sid, q.sname, q.bookno
from (select sb.sid, sb.sname, sb.bookno
     from StudentBook sb
     except
     select sb.sid, sb.sname, sb.bookno
     from StudentBook sb join BuysBooks bt on (bt.sid = sb.sid and sb.price < bt.price)) q
     order by 1;
```

optimised RA Expression:
Books $\rightarrow B = \pi_{bookno,price}(B)$
StudentsBook $\rightarrow SB = \pi_{sdid,sname,bookno,price}(S \bowtie T \bowtie B)$
BuysBooks $\rightarrow BT = \pi_{sid,bookno,price}(T \bowtie B)$
Main query:

$$\pi_{sid,sname,bookno}(\pi_{sid,sname,bookno}(SB) - \pi_{sid,sname,bookno}(SB \bowtie_{bt.sid=sb.sid \wedge sb.price<bt.price} BT))$$

Question 6

Equivalent SQL query:

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

Equivalent RA expression:

MathAndCS $\rightarrow MCS : \pi_{sid}(\pi_{sid}(\sigma_{major='CS'}(M)) \cup \pi_{sid}(\sigma_{major='Math'}(M)))$
Main RA expression:
$\pi_{bookno,title}(\pi_{bookno,title,sid}(B \times S \bowtie MCS) - \pi_{bookno,title,sid}(S \bowtie MCS \bowtie T \bowtie B))$

optimised SQL query:

```
with  MathAndCS as (select m.sid from major m where m.major = 'CS'
                    union
                     select m.sid from major m where m.major = 'Math'),
    Books as (select b.bookno, b.title from book b)
select distinct q.bookno, q.title
from (select distinct b.bookno, b.title, m.sid
     from MathAndCS m
```

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

Equivalent RA expression:

MathAndCS $\rightarrow MCS : \pi_{sid}(\pi_{sid}(\sigma_{major='CS'}(M)) \cup \pi_{sid}(\sigma_{major='Math'}(M)))$
Books $\rightarrow BK : \pi_{bookno,title}(B)$
$\pi_{bookno,title}(\pi_{bookno,title,sid}(BK \times MCS) - \pi_{bookno,title,sid}(T \bowtie BK))$

Part 3: Experiments to Test the Effectiveness of Query Optimization

Question 7

optimised* query:

```
with
R3 as (select distinct r.a as b from R r),
R2R3 as (select distinct r2.a as b from R r2 natural join R3)
select distinct r1.a
from R r1
natural join R2R3;
```

*: optimization explanation is in .sql file.

$makeRandomR$	$Q_3(in\,msec)$	$Q_4(in\,msec)$
(100,100,1000)	83	43
(200,200,2000)	135	49
(500,500,25000)	23686	84
(1000,1000,100000)	360060	188
(2000,2000,400000)	--	605

The -- symbol indicates manual interruption.

Experiment conclusion:

We can notice the significant difference between the execution time of the non-optimised query Q_3 and optimised query Q_4, which is indicative of effectiveness of optimization on Q_3, clearly, optimization works on Q_3. The reason for this massive improvement of execution time on larger relation can be asserted to the structure of query, if we see closely, in Q_3, there are two cartesian products (between r1, r2 and r3), which makes the running time of order $O(|R|^3)$ and for Q_4, the query has been optimised to to utilize the advantage of natural joins, which runs in $O(|R|)$ time making is very fast.

Question 8

optimised* query:

```
select q.a
from (select distinct ra.a
     from Ra ra
     except
```

```
select q2.a
from (select r.a, r.b
     from R r
     except
     select r.a, r.b
     from R r Natural join S s) q2) q;
```

*: optimization explanation is in .sql file.

Execution time comparison table for Q_5 and Q_6.

$makeRandomR$	$makeRandomS$	$Q_5(in\,msec)$	$Q_6(in\,msec)$
(100,100,1000)	(100,50)	42	37
(100,100,1000)	(100,100)	44	38
(500,500,25000)	(500,250)	44	46
(500,500,25000)	(500,500)	45	48
(1000,1000,10000)	(1000,500)	48	50
(1000,1000,10000)	(1000,1000)	52	55
(2000,2000,40000)	(2000,2000)	63	108
(5000,6000,100000)	(6000,6000)	87	187
(8000,8000,100000)	(8000,8000)	89	201

Experiment conclusion:

We can notice that there is not any significant difference between the execution time of the non-optimised query Q_5 and optimised query Q_6, which might be indicative of the fact the optimization doesn't always work. As per the table above, it can be noticed that changes in execution time is not different in order of relation size. Though there is slighly increase in the execution time for larger sized realtions but the difference in execution time is not in the order of magnitude of relation size.

Question 9

optimised* query:

```
select q.a
from (select ra.a
     from Ra ra
     except
     select distinct q2.a
     from (select ra.a, s.b
          from Ra ra cross join S s
          except
          select r.a, s.b
          from R r natural join S s) q2) q;
```

*: optimization explanation is in .sql file.

$makeRandomR$	$makeRandomS$	$Q_7(in\,msec)$	$Q_8(in\,msec)$
(100,100,1000)	(100,50)	57	39
(100,100,1000)	(100,100)	56	46
(500,500,25000)	(500,250)	191	161
(500,500,25000)	(500,500)	195	283
(1000,1000,10000)	(1000,500)	1167	658
(1000,1000,10000)	(1000,1000)	1272	1196
(2000,2000,40000)	(2000,2000)	9362	4893
(3000,5000,80000)	(5000,5000)	28512	29242
(8000,8000,100000)	(8000,8000)	60636	120002

9.c: Experiment conclusion:

Looking at the table above, we can observe that the execution time for Q_7 and Q_8 are in the same order (big 'O') till the size of the relations are small or medium sized. Once the relation becomes very large, the performence of the optimised query Q_8 becomes worse than Q_7. Apart form this, we also observe that keeping all paramters (n and m) same and only increasing the value of 'L' makes the performance of the optimised query bad as compared to the non-optimised query Q_7. This pattern can be observed in the first six entries of the table above. In all those entries, intially I have kept the value of L in relation S small and then increased it to its maximum value, upon doing this, the execution time of query Q_8 when relation S has greater number of values shows significant increase as compared to the Q_7. This jump in execution time can be attributed to a cross join present in optimised query Q_8, where The same reason can be attributed for comparatively bad performace of optimised query Q_8 when the size of relations becomes very large as the gain of optimization of query is overcome by the cross join being performed between Ra and S, which is very significant for large relations

9.d : comparison with query in problem 8.

Looking at the execution time tables of problem 8 and problem 9 for optimised version of both the problems a pattern is observed that, with increase in the size of relations,the query execution time of question 9 (ALL set condition) increases much faster than the query execution time of question 8 (ONLY set condition). this suggests that finding out ONLY set condition questions takes lesser time as compared to solving ALL set condition queries in general. The difference in execution time becomes fairly visible with the larger size of relation.

Question 10

Working and how this query solves the problem: In object-relational method, the query creaste two remporary views, NesterR and SetS, both of these are aggretated arrays. NesterR is grouped for the unique values of R.a, which reduce the size of relation R considerably and SetS, which has reduced the relation S into one row by array aggregation. The query then checks the set containlment condition using '<@' operator for containement of set r.Bs into s.Bs. The set containment operator returns true only when all the elements of first set is present in second set, here it will return true if all elements of r.Bs is present in set s.Bs. At the end we also take union of all those R.a values for which NestedR.Bs is empty to cover all cases of ONLY set condition.

In the table, I have recorded the execution time for Q_5, Q_6 and Q_9. A closer look into the table indicates that the query execution time has not really improved for Q_9 as compared to Q_5 or Q_6. It might be because of the set containment checking we are doing. For smaller sizes of relations, the execution time is almost similar but for very large size of relation, the different in execution time starts to show up. This indicates that object-relational model doesn't always guarantee to make the execution faster.

Conclusion of experiment: The close observation of above table shows that in this particular query (and its optimised and object relational version) Q_5 the pure SQL query is the fastest operating one, the RA optimization, Q_6 hasn't really drastically improved the query execution time, in-fact, it performs either

$makeRandomR$	$makeRandomS$	$Q_5(in\,msec)$	$Q_6(in\,msec)$	$Q_9(in\,msec)$
(100,100,1000)	(100,50)	42	37	38
(100,100,1000)	(100,100)	44	38	39
(500,500,25000)	(500,250)	44	46	44
(500,500,25000)	(500,500)	45	48	50
(1000,1000,10000)	(1000,500)	48	50	58
(1000,1000,10000)	(1000,1000)	52	55	96
(2000,2000,40000)	(2000,2000)	63	108	364
(8000,8000,80000)	(5000,5000)	87	187	1565
(3000,5000,100000)	(5000,5000)	89	201	3248

equally or worse than Q_5, the object-relational version, Q_9 also doesn't' improve the execution time. It performs worse than Q_6. This experiment shows that neither of RA optimization or object-relational query guarantee execution time improvement always. It might be the case that some methods are more suited for some particular types of set operations.

Question 11

Working explanation: This query operates much like the query in question 10 but this time for ALL set condition. The query creaste two remporary views, NesterR and SetS, both of these are aggretated arrays. NesterR is grouped for the unique values of R.a, which reduce the size of relation R considerably and SetS, which has reduced the relation S into one row by array aggregation. The query then checks the set containlment condition using '<@' operator for containement of set s.Bs into r.Bs. The set containment operator returns true only when all the elements of first set is present in second set, here it will return true if all elements of s.Bs is present in set r.Bs.

11.a : Execution time comparison

$makeRandomR$	$makeRandomS$	$Q_7(in\,msec)$	$Q_8(in\,msec)$	$Q_{10}(in\,msec)$
(100,100,1000)	(100,50)	57	39	37
(100,100,1000)	(100,100)	56	46	44
(500,500,25000)	(500,250)	191	161	46
(500,500,25000)	(500,500)	195	283	46
(1000,1000,10000)	(1000,500)	1167	658	56
(1000,1000,10000)	(1000,1000)	1272	1196	55
(2000,2000,40000)	(2000,2000)	9362	4893	86
(3000,5000,80000)	(5000,5000)	28436	29242	143
(8000,8000,100000)	(8000,8000)	60636	120002	173

11.b: additional comparison

$makeRandomR$	$makeRandomS$	$Q_8(in\,msec)$	$Q_{10}(in\,msec)$
(1000,1000,10000)	(1000,10000)	1828	55
(1500,1500,21000)	(1500,15000)	4087	64
(2000,2000,400000)	(2000,400000)	9769	487
(2500,2500,600000)	(2500,600000)	15752	662

11.c: Conclusion of experiment: It is clearly evident from the execution time table comparison of Q_7, Q_8 and Q_{10} that queries written in pure SQL, i.e. Q_7, are not possible to run practically on large data sets, the RA SQL when optimised query, i.e. Q_8, gives significant improvement in execution time but this too is only good and most effective for middle sized relations. The object-relational query, i.e. Q_{10} outperforms all other methodologies and seems suitable for handling larger data.

One of the possible cause for such an improvement could be that the query uses array aggregation and set operations like, set union, set difference and set intersection, which are a linear times operation. Q_{10} uses array aggregation, which significantly reduces the size of relation by aggregating values which is grouped by attribute R.a. The size of relation S is also significantly reduced due to array aggregation, essentially, it has reduced to only one row, which also significantly boost the execution time. So, in Q_{10}, the two array aggregations significantly reduces the size of relations, which greatly speeds up the execution time. In later part of query, there is one Cartesian product but this will be executed in $O(|NestedR|$ as S has only one row. Each modification that has come about object-relation SQL model reduces the number of rows in each relation and significantly speed up the process. This effect can be seen in two tables above.