Assignment 6 Query Translation and Optimization Object-Relational Databases

Sanket Pandilwar

1 Theoretical Problems

1. (a) Show how such SQL queries can be translated to equivalent RA expressions. Be careful in the translation since you should take into account that projections do not in general distribute over intersections or over set differences. To get practice, first consider the following special case where k = 1, m = 1, and n = 1. I.e., the following case:

   Let's consider special case first:

```
SELECT L(r1)
FROM R1 r1
WHERE C1(r1) AND r1.A1 [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))
```

   Solution: With IN:

```
select distinct q(r1) from
(
select r1.* from R1 r1 where C1(r1)
intersect
select r2.* from
(
select r3.* from R3 r3,S1 s1 where C2(s1,r3) and r3.A1=s1.B1
[union | intersect | except]
select r4.* from R4 r4,T1 t1 where C3(t1,r4) and r4.A1=t1.C1
)r2)q
```

   Which in RA can be represented as:

$$\pi_{L(R_1)}(\sigma_{C_1}(R_1) \cap \pi_*(R_3 \bowtie_{C_2(R_3,S_1) \wedge R_3.A_1=S_1.B_1} S_1[\cup|\cap|-]R_4 \bowtie_{C_3(R_4,T_1) \wedge R_4.A_1=T_1.C_1} T_1))$$

   With NOT IN:

```
select distinct q(r1) from
(
select r1.* from R1 r1 where C1(r1)
except
select r2.* from
(
select r3.* from R3 r3,S1 s1 where C2(s1,r3) and r3.A1=s1.B1
[union | intersect | except]
select r4.* from R4 r4,T1 t1 where C3(t1,r4) and r4.A1=t1.C1
)r2)q
```

Which in RA can be represented as:

$$\pi_{L(R_1)}(\sigma_{C_1}(R_1) - \pi_*(R_3 \bowtie_{C_2(R_3,S_1) \wedge R_3.A_1=S_1.B_1} S_1[\cup|\cap|-]R_4 \bowtie_{C_3(R_4,T_1) \wedge R_4.A_1=T_1.C_1} T_1))$$

Now for general case, RA can be written as:
1. In operator:

$$\pi_{L(R_1,..,R_k)}(\sigma_{C_1}(R_1,..,R_k) \cap \pi_*(R \bowtie_{C_2(R,S) \wedge R_1.A_1=S_1.B_1} S[\cup|\cap|-]R \bowtie_{C_3(R,T) \wedge R_1.A_1=T_1.C_1} T))$$

where

$$R = R_1 \times ..... \times R_k$$
$$S = S_1 \times ..... \times S_m$$
$$T = T_1 \times ..... \times T_n$$

2. Not In operator:

$$\pi_{L(R_1,..,R_k)}(\sigma_{C_1}(R_1,..,R_k) - \pi_*(R \bowtie_{C_2(R,S) \wedge R_1.A_1=S_1.B_1} S[\cup|\cap|-]R \bowtie_{C_3(R,T) \wedge R_1.A_1=T_1.C_1} T))$$

where

$$R = R_1 \times ..... \times R_k$$
$$S = S_1 \times ..... \times S_m$$
$$T = T_1 \times ..... \times T_n$$

(b) Show how your translation can be improved when the variable r1 does not occur in the conditions C2 and C3 in the sub query. You don't have to solve this query for the general case, but rather for the SQL query.

If we don't have r1 occurring in C2 and C3, then we can improve our query as follows:

```
with E as
(
SELECT DISTINCT s1.B1 as a
FROM S1 s1
WHERE C2(s1)
[UNION|INTERSECT|EXCEPT]
SELECT DISTINCT t1.C1 as a
FROM T1 t1
WHERE C3(t1)
)
select L(r) from R1 r1 natural join E e where C1(r1)
```

Let E be the expression:

$$E = \pi_*(\pi_{b1}(\sigma_{C_2}(S_1))[\cup| \cap |-]\pi_{c1}(\sigma_{C_3}(T_1)))$$

With In Operator:

$$\pi_{L(R_1)}(R1 \ltimes_{C_1(R_1) \wedge R_1.A=E.A} E)$$

With Not In Operator:

$$\pi_{L(R_1)}(\sigma_{C_1}(R_1) - (R_1 \ltimes E)$$

2. Let R be a relation with schema (a, b, c) and let S be a relation with schema (a, b, d). You may assume that the domains for the attributes a, b, and c are the same. Prove the correctness of the following rewrite rule:
(Reference: Discussed the approach with some of the colleagues)
We have LHS as:

$$\begin{aligned}
&\pi_a(R \bowtie_{R.a=S.b \wedge R.b=S.a} S) \\
&= \{(a)|\exists a \exists b \exists c \exists d(R(a,b,c) \wedge R.a = S.b \wedge R.b = S.a \wedge S(a,b,d))\} \\
&= \{(a)|\exists a \exists b(\exists c(R(a,b,c)) \wedge R.a = S.b \wedge R.b = S.a \wedge \exists d(S(a,b,d)))\} \\
&= \{(a)|\exists a \exists b(((a,b) \in \pi_{a,b}(R)) \wedge R.a = S.b \wedge R.b = S.a \wedge ((a,b) \in \pi_{a,b}(S))) \\
&= \{(a)|\exists a \exists b(((a,b) \in \pi_{a,b}(R)) \bowtie_{R.a=S.b \wedge R.b=S.a} ((a,b) \in \pi_{a,b}(S))) \\
&= \{(a)|\exists a \exists b(\pi_{a,b}(R) \bowtie_{R.a=S.b \wedge R.b=S.a} \pi_{a,b}(S)) \\
&= \{(a)|\exists a \exists b(\pi_{a,b}(R) \cap \pi_{b,a}(S)) \\
&= \pi_a(\pi_{a,b}(R) \cap \pi_{b,a}(S)) \\
&= RHS
\end{aligned}$$

2 Translating and Optimizing SQL Queries to Equivalent RA Expressions

3. Find the sid and name of each student who majors in CS and who bought a book that cites a higher priced book.

   (a) Using the translation algorithm presented in class, translate this SQL into an equivalent RA expression formulated in the standard RA syntax. Specify this RA expression in your .pdf file.

$$\pi_{S.sid,S.sname}(S \bowtie_{S.sid=M.sid \wedge M.major='CS'} M \bowtie_{S.sid=T.sid} T \bowtie_{C.bookno=T.bookno} C$$

$$\bowtie_{C.bookno=B_1.bookno} B_1 \bowtie_{C.citedbookno=B_2.bookno \wedge B_1.price<B_2.price} B_2)$$

   (b) Use the optimization rewrite rules, including those that rely on constraints, to optimize this RA expression. Specify this optimized RA expression in your .pdf file.

   Consider following expressions:

$$\begin{aligned} S &= \pi_{sid,sname}(S) \\ CS &= \pi_{sid}(\sigma_{M.major='CS'}(M)) \\ B &= \pi_{bookno,price}(B) \\ T &= \pi_{sid,bookno}(T) \end{aligned}$$

   Hence original expression can be translated as:

$$\pi_{S.sid,S.sname}(S \bowtie_{S.sid=CS.sid} CS \bowtie_{S.sid=T.sid} T \bowtie_{C.bookno=T.bookno} C \bowtie_{C.bookno=B_1.bookno} B_1$$

$$\bowtie_{C.citedbookno=B_2.bookno \wedge B_1.price<B_2.price} B_2)$$

   Following are the optimization steps and optimized RA.

$\pi_{S.sid,S.sname}(S \ltimes CS \bowtie_{S.sid=T.sid} T \bowtie_{C.bookno=T.bookno} C \bowtie_{C.bookno=B_1.bookno} B_1$
$\bowtie_{C.citedbookno=B_2.bookno \wedge B_1.price<B_2.price} B_2)$
$= \pi_{S.sid,S.sname}(S \ltimes CS \bowtie_{S.sid=T.sid} (\pi_{T.sid}(T \bowtie_{C.bookno=T.bookno}$
$(\pi_{C.bookno}(C \bowtie_{C.bookno=B_1.bookno} (\pi_{B_1.bookno,B_2.bookno}(B_1 \bowtie_{B_1.price<B_2.price} B_2)))))))$
$= \pi_{S.sid,S.sname}(S \ltimes CS \bowtie (\pi_{T.sid}(T \bowtie (\pi_{C.bookno}(C \bowtie_{C.bookno=B_1.bookno \wedge C.citedbookno=B_2.bookno}$
$(\pi_{B_1.bookno,B_2.bookno}(B_1 \bowtie_{B_1.price<B_2.price} B_2)))))))$
$= \pi_{S.sid,S.sname}(S \ltimes CS \ltimes (\pi_{T.sid}(T \ltimes (\pi_{C.bookno}(C \ltimes_{C.bookno=B_1.bookno \wedge C.citedbookno=B_2.bookno}$
$(\pi_{B_1.bookno,B_2.bookno}(B_1 \bowtie_{B_1.price<B_2.price} B_2)))))))$

$Let,$
$E_1 = \pi_{S.sid,S.sname,bookno}(S \bowtie CS \bowtie (\pi_{T.sid}(T)))$
$E_2 = \pi_{C.bookno}(C \ltimes_{C.bookno=B_1.bookno \wedge C.citedbookno=B_2.bookno}$
$(\pi_{B_1.bookno,B_2.bookno}(B_1 \bowtie_{B_1.price<B_2.price} B_2)))$

Then optimized expression becomes:

$$\pi_{E_1.sid,E_1.sname}(E_1 \ltimes E_2)$$

4. Find the sid, name, and major of each student who does not major in CS,did not buy a book that less than 30, and bought some book(s) that cost less than less than 50

   (a) Using the translation algorithm presented in class, translate this SQL into an equivalent RA expression formulated in the standard RA syntax. Specify this RA expression in your .pdf file.

   Let E be the expression.

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

   Query:

   $$\pi_{E.*}(E-(E_1 \bowtie T \bowtie_{T.bookno=B.bookno \wedge B.price<30} B))-\pi_{E.*}(E- \bowtie_{E.sid=M.sid \wedge M.major='CS'} M)$$

   (b) Use the optimization rewrite rules, including those that rely on constraints, to optimize this RA expression. Specify this optimized RA expression in your .pdf file.

   Consider following expressions:

   $$\begin{array}{lcl} S & = & \pi_{sid,sname}(S) \\ CS & = & \pi_{sid,major}(\sigma_{M.major='CS'}(M)) \\ B_1 & = & \pi_{bookno}(\sigma_{B.price<60}(B)) \\ B_2 & = & \pi_{bookno}(\sigma_{B.price<30}(B)) \\ T & = & \pi_{sid,bookno}(T) \end{array}$$

   Expression E can be optimized by pushing down selections and projections as follows:

   $$\pi_{S.sid,S.sname,M.major}(M \bowtie (S \ltimes (T \ltimes B_1)))$$

   Following are the optimization steps and optimized RA.

   $$\pi_{E.*}(E - (E \bowtie \pi_{T.sid}(T \bowtie_{T.bookno=B_2.bookno} B_2))) - \pi_{E.*}(E- \bowtie_{E.sid=CS.sid} (CS))$$

   $$= \pi_{E.*}(E \bar{\ltimes} \pi_{T.sid}(T \ltimes B_2)) - \pi_{E.*}(E \ltimes (CS))$$

   Let,

   $$E_1 = \pi_{E.*}(E \bar{\ltimes} \pi_{T.sid}(T \ltimes B_2))$$
   $$E_2 = \pi_{E.*}(E \ltimes (CS))$$

   Then optimized expression becomes:

   $$E_1 - E_2$$

5. Find each (s, n, b) triple where s is the sid of a student, n is the name of this student, and where b is the bookno of a book whose price is the most expensive among the books bought by that student.

   (a) Using the translation algorithm presented in class, translate this SQL into an equivalent RA expression formulated in the standard RA syntax. Specify this RA expression in

your .pdf file.

Let E be the expression:

$$\pi_{S.sid,S.sname,B.bookno,B.price}(S \bowtie T \bowtie B)$$

Now Query can be written as:

$$\pi_{E.sid,E.sname,E.bookno}(E) - \pi_{S.sid,S.sname,B.bookno}(S \bowtie T \bowtie \pi_{B.bookno}(B \bowtie_{\neg(B.price>=B1.price)})(T_1 \bowtie B_1))$$

(b) Use the optimization rewrite rules, including those that rely on constraints, to optimize this RA expression. Specify this optimized RA expression in your .pdf file.

Consider following expressions:

$$\begin{aligned} S &= \pi_{sid,sname}(S) \\ B &= \pi_{bookno,price}(B) \\ T &= \pi_{bookno,sid}(T) \end{aligned}$$

Expression E can be optimized as follows:

$$\pi_{S.sid,S.sname,B.bookno,B.price}((S \bowtie B) \ltimes_{T.sid=S.sid} T)$$

Following are the optimization steps and optimized RA.

$$\pi_{E.sid,E.sname,E.bookno}(E) -$$

$$\pi_{S.sid,S.sname,B.bookno}(S \bowtie T \bowtie \pi_{B.bookno}(B \bowtie_{\neg(B.price>=B_1.price)} \pi_{T_1.sid,T_1.bookno,B_1.price}(T_1 \bowtie B_1)))$$

$$= \pi_{E.sid,E.sname,E.bookno}(E) - \pi_{E.sid,E.sname,E.bookno}(E \bowtie_{\neg(E.price>=B_1.price)} (T_1 \bowtie B_1))$$

6. Find the bookno and title of each book that is not bought by all students who major in both CS or in Math.

   (a) Using the translation algorithm presented in class, translate this SQL into an equivalent RA expression formulated in the standard RA syntax. Specify this RA expression in your .pdf file.

   Let E be the expression:

   $$\pi_{sid}(\pi_{S.sid}(S \bowtie_{S.sid=M.sid \wedge M.major='CS'} M) \cup \pi_{S.sid}(S \bowtie_{S.sid=M.sid \wedge M.major='Math'} M))$$

   Now Query can be written as:

   $$\pi_{B_1.bookno,B_1.title}(B_1 \bowtie (\pi_{B.bookno,E.sid}(B \times E) - \pi_{T.bookno,T.sid}(T)))$$

   (b) Use the optimization rewrite rules, including those that rely on constraints, to optimize this RA expression. Specify this optimized RA expression in your .pdf file.

Consider following expressions:

$$\begin{array}{lcl} S & = & \pi_{sid}(S) \\ CS & = & \pi_{sid}(\sigma_{M.major='CS'}(M)) \\ MATH & = & \pi_{sid}(\sigma_{M.major='Math'}(M)) \\ B_1 & = & \pi_{bookno,title}(B) \\ T & = & \pi_{bookno,sid}(T) \end{array}$$

Expression E can be optimized by pushing down selections and projections as follows:

$$\pi_{sid}((S \ltimes CS) \cup (S \ltimes MATH)) = \pi_{sid}(CS \cup MATH)$$

Following are the optimization steps and optimized RA.

$$\pi_{B_1.bookno,B_1.title}(B_1 \bowtie (\pi_{B.bookno,E.sid}(\pi_{bookno}(B) \times \pi_{sid}(E)) - \pi_{T.bookno,T.sid}(T)))$$

$$= \pi_{B_1.bookno,B_1.title}(B_1 \ltimes \pi_{B.bookno,E.sid}((B \times E) - T))$$

Then optimized expression becomes:

$$B_1 \ltimes \pi_{B.bookno,E.sid}((B \times E) - T))$$

3 Experiments to Test the Effectiveness of Query Optimization

7. We have Query Q3 as:

```
select distinct r1.a
from R r1, R r2, R r3
where r1.b = r2.a and r2.b = r3.a;
```

After translation and optimization, Q4 becomes:

```
select distinct r1.a
from R r1 natural join
(select distinct r2.a as b from R r2 natural join
 (select r3.a as b from R r3) r3) r4;
```

Comparison using different experiments:

makerandomR	**Q3 (in ms)**	**Q4 (in ms)**
(100,100,1000)	23.1	2.1
(300,300,10000)	2511.34	97.965
(500,500,25000)	13150.718	214.686
(1000,1000,100000)	208203.47	1693.89
(2000,2000,400000)	–	16053.09

Conclusion:
1. For larger data, in memory sorting did not work hence db engine switched to external merge sort and used disk space instead of main memory, whereas for smaller data it was using main memory and for sorting, quick sort was being used.
2. For larger data, even Q4 used external merge sort and disk space to execute the query.
3. Query Q3 was running in cubic time because of two joins hence execution time increased drastically with every experiment whereas, Q4 has linear time complexity and makes use of hashing hence it is significantly faster than Q3.
4. As the data increases, both the queries takes longer time to execute.

8. We have Query Q5 as:

```
select ra.a
from Ra ra
where not exists (select r.b from R r where r.a = ra.a and
r.b not in (select s.b from S s));
```

After translation and optimization, Q6 becomes:

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

Comparison using different experiments:

makerandomR	**makerandomS**	**Q5 (in ms)**	**Q6 (in ms)**
(100,100,1000)	(100,50)	0.703	5.26
(100,100,1000)	(100,90)	0.776	4.157
(300,300,10000)	(300,100)	5.822	40.396
(300,300,10000)	(300,280)	5	51.58
(500,500,25000)	(500,200)	23.41	163.197
(500,500,25000)	(500,490)	7	135.50
(1000,1000,100000)	(1000,990)	69.6	454

Conclusion:
1. For query Q5, there are very less operations and hashing in used hence first is performing better. Q5 used hash anti join which is faster.
2. Query Q6 has multiple joins hence little time consuming as well as lots of operations being done with two except operations as well.
3. As data increases execution time also increases.
4. For Query Q6, its using lots of joins. A hash join has expected complexity O(M + N), but has unfavorable memory access patterns (random disk access is really bad). Because of lots of joins, there is lots of disk access.

9. We have Query Q7 as:

```
select ra.a from Ra ra
where not exists (select s.b from S s
where s.b not in (select r.b from R r where r.a = ra.a));
```

After translation and optimization, Q8 becomes:

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

Comparison using different experiments:

makerandomR	**makerandomS**	**Q7 (in ms)**	**Q8 (in ms)**
(100,100,1000)	(100,50)	16.57	19.026
(100,100,1000)	(100,90)	22.11	15.53
(300,300,10000)	(300,100)	606	65.54
(300,300,10000)	(300,280)	544	231.8
(500,500,25000)	(500,200)	2541.63	355.85
(500,500,25000)	(500,490)	2266.25	912
(1000,1000,100000)	(1000,990)	19097	2878.756

Conclusions:
1. Query Q7 has nested loop anti join, hence quadratic in time. It also has some sequential scan and it doesnt use hash join. Hence performance is slower.
2. Query Q8 also has nested loop hence quadratic in time because of cross join but it used hashing. Q8 also uses except with hash hence it is faster.
3. As it can be seen from experiment, Q8 performs faster than Q7.
4. ONLY vs ALL, optimized query was faster than original for ALL but it was opposite in case of ONLY.
5. ONLY executes faster than ALL query for both the normal and optimized query for the same data size.
6. ONLY and ALL operates on same set but positioning is different. For ONLY, it was easier to check except/not exists operation than ALL.
7. We used cross join in ALL query and ONLY used just natural joins which might contribute to the time difference.

10. We have Query Q9 as:

```
with NestedR as (select r.a, array_agg(r.b order by 1) as Bs
from R r
group by (r.a)),
SetS as (select array(select s.b from S s order by 1) as Bs)
select r.a
from NestedR r, SetS s
where r.Bs <@ s.Bs
union
select r.a
from (select a from ra
except
select distinct a from R) r;
```

Working:
1. nestedR stores all values of a from R relation and the list of b's occurred for each of the element a.
2. setS stores list of all b's in relation S.
3. The given query checks if the list of b's from relation R is a subset of b's from relation S which is ONLY set operation

Comparison using different experiments:

makerandomR	**makerandomS**	**Q5 (in ms)**	**Q6 (in ms)**	**Q9 (in ms)**
(100,100,1000)	(100,50)	0.703	5.26	0.813
(100,100,1000)	(100,90)	0.776	4.157	0.707
(300,300,10000)	(300,100)	5.822	40.396	5.576
(300,300,10000)	(300,280)	5	51.58	6.524
(500,500,25000)	(500,200)	23.41	163.197	13.41
(500,500,25000)	(500,490)	7	135.50	20.59
(1000,1000,100000)	(1000,990)	69.6	454	116.095

Conclusion:
1. Q9 and Q5 has almost similar running times. They both are faster than Q6.
2. Q9 uses group by, set operations and in-place quick sort.
3. As size increases, running time also increases.

11. We have Query Q10 as:

```
with NestedR as (select r.a, array_agg(r.b order by 1) as Bs
from R r
group by (r.a)),
SetS as (select array(select s.b from S s order by 1) as Bs)
select r.a
from NestedR r, SetS s
where s.Bs <@ r.Bs
```

Working:
1. This is same as query Q9 but with the subset condition reversed i.e the given query checks if the list of b's from relation S is a subset of b's from relation R which is ALL set operation.

Comparison using different experiments:

makerandomR	**makerandomS**	**Q7 (in ms)**	**Q8 (in ms)**	**Q10 (in ms)**
(100,100,1000)	(100,50)	16.57	19.026	0.535
(100,100,1000)	(100,90)	22.11	15.53	0.551
(300,300,10000)	(300,100)	606	65.54	5.415
(300,300,10000)	(300,280)	544	231.8	4.986
(500,500,25000)	(500,200)	2541.63	355.85	13.783
(500,500,25000)	(500,490)	2266.25	912	15.53
(1000,1000,100000)	(1000,990)	19097	2878.756	55.28

Extra experiment with Q8 and Q10:

makerandomR	**makerandomS**	**Q8 (in ms)**	**Q10 (in ms)**
(500,500,25000)	(500,490)	912	15.53
(1000,1000,100000)	(1000,990)	2878.756	55.28
(2000,2000,400000)	(2000,1000)	2010	216.62
(2000,2000,400000)	(2000,1990)	3750	217
(5000,5000,2500000)	(5000,5000)	4881	220

Conclusion:
1. From above experiment, We can say that Q10 is faster than Q7 and Q8
2. Q10 uses group by, set operations and in-place quick sort.
3. When u run Q8 and Q10 alone, both performs very well but Q10 is still faster because Q8 is quadratic in time.

4 Formulating Queries in the Object-Relational Model

12. Set Union

```
create or replace function setunion(A anyarray, B anyarray)
returns anyarray as
$$
with
Aset as (select UNNEST(A)),
Bset as (select UNNEST(B))
select array( (select * from Aset) union
(select * from Bset) order by 1);
$$ language sql;
```

Set Intersection

```
create or replace function setintersection(A anyarray, B anyarray)
returns anyarray as
$$
with
Aset as (select UNNEST(A)),
Bset as (select UNNEST(B))
select array( (select * from Aset) intersect
(select * from Bset) order by 1);
$$ language sql;
```

Set Difference

```
create or replace function setdifference(A anyarray, B anyarray)
 returns anyarray as
$$
with
Aset as (select UNNEST(A)),
Bset as (select UNNEST(B))
select array( (select * from Aset) except
(select * from Bset) order by 1);
$$ language sql;
```

Is In

```
create or replace function isIn(x anyelement, S anyarray)
returns boolean as
$$
select x = SOME(S);
$$ language sql;
```

13. student_books:

```
create or replace view student_books as
select s.sid, array(select t.bookno
from buys t
where t.sid = s.sid order by bookno) as books
from student s order by sid;
```

```
book_students:
create or replace view book_students as
select b.bookno, array(select t.sid from buys t
where t.bookno=b.bookno order by t.sid) as students
from book b order by b.bookno;
```

```
book_citedbooks:
create or replace view book_citedbooks as
select b.bookno, array(select c.citedbookno from cites c
where c.bookno=b.bookno order by c.citedbookno) as citedbooks
```

```
from book b order by b.bookno;

book_citingbooks:
create or replace view book_citingbooks as
select b.bookno, array(select c.bookno from cites c
where c.citedbookno=b.bookno order by c.bookno) as citingbooks
from book b order by b.bookno;


major_students:
create or replace view major_students as
select m.major, array_agg(m.sid) as students from major m
group by m.major order by m.major;


student_majors:
create or replace view student_majors as
select s.sid, array(select m.major from major m
where m.sid=s.sid order by m.major) as majors from student s
order by s.sid;
```

14. (a) Find the bookno and title of each book that cites at least three books that cost less than 50.

```
with E as
(select bookno, unnest(citedbooks) as citedbooks
from book_citedbooks where cardinality(citedbooks)>=3),
F as
(select e.bookno from E e join book b
on(e.citedbooks=b.bookno) where b.price<50
group by e.bookno having count(e.bookno)>=3 )
select b.bookno,b.title from F f natural join book b;
```

(b) Find the bookno and title of each book that was not bought by any students who majors in CS.

```
with E as
(select unnest(students) as students
from major_students where major='CS')
select b.bookno, b.title
from book b natural join book_students bs where
not exists(select 1 from E e where isIn(e.students, bs.students));
```

(c) Find the sid of each student who bought all books that cost at least 50.

```
with E as
(select b.bookno from book b where b.price>=50)
select sb.sid from student_books sb where not exists
(select 1 from E e where not isIn(e.bookno,sb.books));
```

(d) Find the bookno of book that was not only bought by students who major in CS.

```
select bs.bookno from book_students bs where
not setdifference(bs.students,
(select students from major_students where major='CS'))='{}';
```

(e) Find the bookno and title of each book that was not only bought by students who bought all books that cost more than 45.

```
with E as
(select b.bookno from book b where b.price>45),
F as
(select array_agg(sb.sid)as student_list from student_books sb where
not exists
(select 1 from E e where not isIn(e.bookno,sb.books)))
select bs.bookno,b.title from book_students bs natural join book b
where
not setdifference(bs.students,(select student_list from F))='{}';
```

(f) Find sid-bookno pairs (s, b) such that not all books bought by student s are books that cite book b.

```
select distinct sb.sid, bc.bookno
from student_books sb, book_citingbooks bc
where not(sb.books <@ bc.citingbooks)
order by sb.sid,bc.bookno;
```

(g) Find the pairs (b1, b2) of booknos of books that where bought by the same set of students.

```
select bs1.bookno, bs2.bookno
from book_students bs1, book_students bs2
where bs1.students<@ bs2.students
and bs2.students<@ bs1.students;
```

(h) Find the pairs (b1, b2) of booknos of books that where bought by the same number of students.

```
select bs1.bookno,bs2.bookno
from book_students bs1, book_students bs2
where cardinality(bs1.students)=cardinality(bs2.students);
```

(i) Find the sid of each student who bought all but four books.

```
select sb.sid from student s natural join student_books sb
where
(select count(1) from
(select bo.bookno from book bo
except
select unnest(sb1.books) from student_books sb1
where sb1.sid=sb.sid ) c)=4;
```

(j) Find the sid of each student who bought no more books than the combined number of books bought by the set of students who major in Psychology.

```
with E as
(select unnest(students) as sid from major_students
where major='Psychology'),
F as
(select count(1) from
(select e.sid, unnest(sb.books)
from student_books sb natural join E e)q)
select sb.sid from student_books sb
where cardinality(sb.books) <= (select * from F);
```