Assignment 5: Relational Algebra

For this assignment, you will need to submit 3 files. The first file is a .sql file that should contain all the SQL code relating to problems requesting the development of such code. The second file is a .txt file that should contain the output of the queries in the problems in Part 2. The third file is a .pdf file that should contain your solutions for problems where RA expressions in their standard (i.e., non SQL) notation are requested. Ideally you should use latex to construct this .pdf file. Latex offers a convenient syntax to formulate RA expessions.

Before you solve the problems in this section, we briefly review how you can express RA expressions in SQL in a way that closely mimics their RA specifications. (For more detail, consult the lectures relating to RA and joins.)

Consider a relation $R(A, B)$ and a relation $S(C)$ and consider the following RA expression F:

$$\pi_A(R) - \pi_A(\sigma_{B=1}(R \bowtie_{B=C} S))$$

Then we can write this query in SQL in a variety of ways that closely mimics its RA formulation. One way to write this RA expression in SQL is as follows:

```
SELECT DISTINCT A
FROM   R
EXCEPT
SELECT A
FROM   (SELECT DISTINCT A, B, C
        FROM   R JOIN S ON (B = C)
        WHERE  A = 1) q
```

An alternative way to write this query is to use the `WITH` statement of SQL.[1] To do this, we separate the RA expression F into sub-expressions as follows. (In this case, notice that each sub-expression corresponds to the application of a single RA operation. More generally, one can of course use sub-expressions that can contain multiple RA operations.)

Expression Name	RA expression
E_1	$\pi_A(R)$
E_2	$R \bowtie_{B=C} S$
E_3	$\sigma_{B=1}(E_2)$
E_4	$\pi_A(E_3)$
F	$E_1 - E_4$

[1] This is especially convenient when the RA expression is long and complicated.

Then we write the following SQL query. Notice how the expressions $E1$, $E2$, $E3$, and $E4$ occur as separate queries in the WITH statement and that the final query gives the result for the expression F.[2]

```
WITH
E1 AS (SELECT DISTINCT A FROM R),
E2 AS (SELECT DISTINCT A, B, C FROM (R JOIN S ON (B = C)) e2),
E3 AS (SELECT A, B, C FROM E2 WHERE B = 1),
E4 AS (SELECT DISTINCT A FROM E3)
(SELECT A FROM E1) EXCEPT (SELECT A FROM E4);
```

In your answer to a problem, you may write the resulting RA expression with or without the WITH statement. (Your SQL query should of course closely resemble the RA expression it is aimed to express.)

1 Theoretical Problems about RA

1. (a) Consult the lecture on set joins and semijoins. Using the techniques described in that lecture, develop a general RA expression for the "`all-but-two`" set semijoin.

   Solution: Let E_1 be an RA expression with schema $(\mathbf{A}, \mathbf{C})$ and let E_2 be an RA expression with schema $(\mathbf{C}, \mathbf{B})$. Here $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ denote sequences of attributes $(A_1, \ldots, A_m)$, $(C_1, \ldots, C_k)$, and $(B_1 \ldots B_n)$, respectively. We assume that there is no overlap among these attributes.

   Of interest for this problem is the RA expression, denoted F,

   $$dom(\mathbf{A}) \times \pi_{\mathbf{C}}(E_2) - E_1$$

   where $dom(\mathbf{A})$ denotes $dom(A_1) \times \cdots \times dom(A_m)$. Here $dom(A_1)$ through $dom(A_m)$ are the domains of the attributes A_1 through A_m, respectively. Also note that the schema of F is $(\mathbf{A}, \mathbf{C})$.

   For the "`all-but-two`" semijoin, what we need to find are those $\mathbf{a} \in dom(\mathbf{A})$ such that

   $$|\sigma_{\mathbf{A}=\mathbf{a}}(F)| = 2.$$

   Since we are in RA, we need to formulate that for each $\mathbf{a} \in \mathbf{A}$, $\sigma_{\mathbf{A}=\mathbf{a}}(F)$ has at least 2 elements but not at least 3 elements.

   The "`at-least 2`" RA expression is

   $$\pi_{\mathbf{A}}(F_1 \bowtie_{F_1.\mathbf{A}=F_2.\mathbf{A} \wedge F_1.\mathbf{B} \neq F_2.\mathbf{B}} F_2).$$

[2] For better readability, I have used relational-name overloading. Sometimes, you may need to introduce new attribute names in SELECT clauses using the AS clause. Also, use DISTINCT were needed.

and the "`at-least 3`" RA expression is

$$\pi_{\mathbf{A}}((F_1 \bowtie_{F_1.\mathbf{A}=F_2.\mathbf{A} \wedge F_1.\mathbf{B} \neq F_2.\mathbf{B}} F_2) \bowtie_{F_1.\mathbf{A}=F_3.\mathbf{A} \wedge F_1.\mathbf{B} \neq F_3.\mathbf{B} \wedge F_2.\mathbf{B} \neq F_3.\mathbf{B}} F_3).$$

So the RA expression for the "`all-but-two`" semijoin is

$$\pi_{\mathbf{A}}(F_1 \bowtie_{F_1.\mathbf{A}=F_2.\mathbf{A} \wedge F_1.\mathbf{B} \neq F_2.\mathbf{B}} F_2) - \pi_{\mathbf{A}}((F_1 \bowtie_{F_1.\mathbf{A}=F_2.\mathbf{A} \wedge F_1.\mathbf{B} \neq F_2.\mathbf{B}} F_2) \bowtie_{F_1.\mathbf{A}=F_3.\mathbf{A} \wedge F_1.\mathbf{B} \neq F_3.\mathbf{B} \wedge F_2.\mathbf{B} \neq F_3.\mathbf{B}} F_3).$$

(b) Apply this RA expression to the query "Find the bookno and title of each book that is bought by all but two students who major in 'CS'.

Solution: For this query, E_1 and E_2 are as follows:

$$E_1 = Buys$$
$$E_2 = \pi_{sid}(\sigma_{major=\text{“CS”}}(Major)).$$

Assume that we substitute these expressions into the RA expression for the "`all-but-two`" semijoin developed in Problem 1a and, as such, get an expression which we denote by G.
Then, since we need to get both the bookno and title of the sought for book, the final RA expression for the query becomes

$$\pi_{bookno,title}(Book) \ltimes G.$$

(c) Formulate the RA expression obtained in Problem 1b in SQL with relational operators. (So no SQL set predicates are allowed in your solution.)

Solution:

```
WITH E1 AS (SELECT bookno, sid FROM Buys),
     E2 AS (SELECT DISTINCT sid FROM Major WHERE major = 'CS'),
     B  AS (SELECT bookno FROM Book),
     F  AS (SELECT * FROM B CROSS JOIN E2
            EXCEPT
            SELECT * FROM E1),
     G2 AS (SELECT DISTINCT f1.bookno
            FROM F f1
                 JOIN F f2 ON (f1.bookno = f2.bookno AND
                               f1.sid <> f2.sid)),
     G3 AS (SELECT DISTINCT f1.bookno
            FROM F f1
                 JOIN F f2 ON (f1.bookno = f2.bookno AND
                               f1.sid <> f2.sid)
                 JOIN F f3 ON (f1.bookno = f3.bookno AND
                               f1.sid <> f3.sid AND f2.sid <> f3.sid))
```

```
SELECT DISTINCT b.bookno, b.title
FROM   Book b
       NATURAL JOIN (SELECT DISTINCT * FROM G2
                     EXCEPT
                     SELECT DISTINCT * FROM G3) q;
```

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

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

   $$\texttt{if } F \neq \emptyset \quad \texttt{then return } E_1 \quad \texttt{else return } E_2$$

   So this query evaluates to the expression E_1 if $F \neq \emptyset$ and to the expression E_2 if $F = \emptyset$.

   We can formulate this query in SQL as follows[3]:

```
select e1.*
from   E1 e1
where  exists (select 1 from F)
union
select e2.*
from   E2 e1
where  not exists (select 1 from F);
```

   (a) i. Write an RA expression, in function of E_1, E_2, and F, that expresses this "`if-then-else`" statement.

   Solution: We assume that the schemas of E_1 and E_2 are different than the schema of F. If not, then we need to rename attributes to make this happen.
   We can then express the "`if-then-else`" statement in RA as follows:

   $$\pi_{E_1.*}(E_1 \times F) \cup (E_2 - \pi_{E_2.*}(E_2 \times F)) \quad (1).$$

   Notice how this expression works. When $F \neq \emptyset$, the expression evaluation to E_1 since in this case $\pi_{E_1.*}(E_1 \times F) = E_1$ and $(E_2 - \pi_{E_2.*}(E_2 \times F)) = E_2 - E_2 = \emptyset$, and therefore

   $$\pi_{E_1.*}(E_1 \times F) \cup (E_2 - \pi_{E_2.*}(E_2 \times F)) = E_1 \cup \emptyset = E_1.$$

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

When $F = \emptyset$, $\pi_{E_1.*}(E_1 \times F) = \emptyset$, since, in this case, $E_1 \times F = \emptyset$, and $(E_2 - \pi_{E_2.*}(E_2 \times F)) = E_2 - \emptyset = E_2$. Thus, in case,

$$\pi_{E_1.*}(E_1 \times F) \cup (E_2 - \pi_{E_2.*}(E_2 \times F)) = \emptyset \cup E_2 = E_2.$$

An alternative way to express the "`if-then-else`" statement is as follows:

$$E_1 \times \pi_{()}(F) \cup (E_2 - E_2 \times \pi_{()}(F)) \quad (2).$$

Here, $\pi_{()}(F)$ is the projection of F on the empty set of attribute. Observe that $\pi_{()}(F) = \{()\}$ if $F \neq \emptyset$, and $\pi_{()}(F) = \emptyset$ if $F = \emptyset$. Expression (2) is better than expression (1), since its complexity is $O(|E_1|+|E_2|+|F|)$ as compared to the complexity of expression (2) which is $O((|E_1| + |E_2) * |F|)$.
Therefore, we will use Expression (2) in several of problems that follow below.

ii. Then express this RA expression in SQL with RA operators. In particular, you can not use SQL set predicates in your solution.

Solution:
We can formulate Expression in SQL with RA operators as follows:

```
SELECT  e1.*
FROM    E1 e1 CROSS JOIN (SELECT row() FROM F) f
UNION
(SELECT  e2.*
 FROM   E2 e2
 EXCEPT
 SELECT e2.*
 FROM   E2 e2 CROSS JOIN (SELECT row() FROM F) f);
```

Notice that in SQL, `row()` in SQL evaluates to the empty tuple.

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

```
select exists(select 1 from A) as A_isNotEmpty;
```

This boolean query returns the constant "`true`" if $A \neq \emptyset$ and returns the constant "`false`" otherwise. Using the insights you gained from Problem 2a, solve the following problems:

i. Write an RA expression that expresses the above boolean SQL query.
Hint: recall that, in general, a constant value "**a**" can be represented in RA by an expression of the form "`(C: a)`". (Here, `C` is some arbitrary attribute name.) Furthermore, recall that

we can express "`(C: a)`" in SQL as "`select a as C`". Thus, we can use RA expressions "`(C: true)`" and "`(C: false)`" to represent for the constants "`true`" and "`false`", respectively.

Solution: Notice that this problem can be expressed as the following "`if-then-else`" statement:

`if` $A \neq \emptyset$ `then return` "`true`"
`else return` "`false`"

We can then use Expression (2) and obtain the RA expression

$$(\texttt{A\_isNotExmpty} : \texttt{true}) \times \pi_{()}(A) \cup ((\texttt{A\_isNotEmpty} : \texttt{false}) - (\texttt{A\_isNotEmpty} : \texttt{false}) \times \pi_{()}(A))).$$

ii. Write a SQL query with relational operators, thus without set predicates, that expresses the above boolean SQL query.

Solution

```
SELECT  e1.A_isNotEmpty
FROM     (SELECT true as A_isNotEmpty) e1 CROSS JOIN (SELECT row() FROM A) a
UNION
(SELECT  e2.A_isNotEmpty
 FROM   (SELECT false as A_isNotEmpty) e2
 EXCEPT
 SELECT e2.A_isNotEmpty
 FROM   (SELECT false as A_isNotEmpty) e2 CROSS JOIN (SELECT row() FROM A) a);
```

3. Let $f : A \to B$ be a function from a set A to a set B and let $g : B \to C$ be a function from a set B to a set C. The *composition* of the functions f and g, denoted $g \circ f$, is a function from A to C such that for $x \in A$, $g \circ f(x)$ is defined as the value $g(f(x))$.

   Represent f in a binary relation `f` with schema `(A,B)` and represent g in a binary relation `g` with schema `(B,C)`.

   (a) Write an RA expression that computes the function $g \circ f$. I.e., your expression should compute the binary relation $\{(x, g \circ f(x)) \mid x \in A\}$.
   Solution:

   $$\pi_{A,C}(F \bowtie G).$$

   In SQL:

   ```
   SELECT F.A, G.C
   FROM F NATURAL JOIN G;
   ```

   (b) Let y be a value in C. Write an RA expression that computes the set $\{x \in A \mid g \circ f(x) = y\}$. I.e., these are the values in A that are mapped by the function $g \circ f$ to the value y.

Solution:

$$\pi_A(\sigma_{C=y}(F \bowtie G)).$$

In SQL:

```
SELECT DISTINCT F.A
FROM F NATURAL JOIN G
WHERE G.C = y;
```

4. Let $f : A \to B$ be a function from a set A to a set B. We say that f is a *one-to-one* function if for each pair x_1 and x_2 of different values in A (i.e., $x_1 \neq x_2$) it is the case that $f(x_1) \neq f(x_2)$. Represent f by a relation `f` with schema `(A,B)`.

   Write an RA expression that returns the value "`true`" if f (as stored in `f`) is a one-one-one function, and returns the value "`false`" otherwise.

   Solution:

   It is instructive to first express the one-to-one condition of f in TRC:

$$\forall t_1 \forall t_2((\mathtt{f}(t_1) \wedge \mathtt{f}(t_2) \wedge t_1.A \neq t_2.A) \to t_1.B \neq t_2.B)$$

   or, equivalently

$$\neg(\exists t_1 \exists t_2(\mathtt{f}(t_1) \wedge \mathtt{f}(t_2) \wedge t_1.A \neq t_2.A \wedge t_1.B = t_2.B)$$

   Next, focus on the subformula:

$$\mathtt{f}(t_1) \wedge \mathtt{f}(t_2) \wedge t_1.A \neq t_2.A \wedge t_1.B = t_2.B.$$

   The corresponding RA expression for this is

$$\mathtt{f}_1 \bowtie_{\mathtt{f}_1.A \neq \mathtt{f}_2.A \wedge \mathtt{f}_1.B = \mathtt{f}_2.B} \mathtt{f}_2.$$

   Let us call this expression E. Notice that if E evaluates to a non-empty relation, then f is not one-to-one, and if E evaluates to the empty relation, then f is one-to-one.

   We can now express the one-to-one condition of f in RA as follows:

$$(is\_One\_to\_One : \mathtt{false}) \times \pi_{()}(E) \cup \\ ((is\_One\_to\_One : \mathtt{true}) - ((is\_One\_to\_One : \mathtt{true}) \times \pi_{()}(E)).$$

5. Let $f : A \to B$ be a function from a set A to a set B. We say that f is an *onto* function if for each value y in B, there exists a value x in A such that $f(x) = y$. Represent f by a relation `f` with schema `(A,B)`.

Write an RA expression that returns the value "`true`" if f (as stored in `f`) is an onto function, and returns the value "`false`" otherwise.

Solution: If we represent B by a relation `B(y)`, then the onto condition of f states that the `B` attribute of `f` is a foreign key referencing the attribute `y` in `B`.

Now consider the RA expression

$$\texttt{B} - \pi_{\texttt{B}}(\texttt{f}).$$

If indeed f is onto then this expression evaluates to the empty set, and if f is not onto then this expression evaluates to a nonempty set.

Therefore, the RA expression for the onto condition of f can be expressed in RA using the following expression:

$$\begin{aligned}&(is\_Onto : \texttt{false}) \times \pi_{()}(\texttt{B} - \pi_{\texttt{B}}(\texttt{f})) \cup \\ &\quad ((is\_Onto : \texttt{true}) - ((is\_Onto : \texttt{true}) \times \pi_{()}(\texttt{B} - \pi_{\texttt{B}}(\texttt{f})))).\end{aligned}$$

6. A *graph* G is a structure (V, E) where V is a set of vertices and wherein E is a set of edges between these vertices. Thus $E \subseteq V \times V$.

   A *path* in G is a sequence of vertices $(v_0, v_1, \ldots, v_n)$ such that for each $i \in [0, n-1]$, $(v_i, v_{i+1}) \in E$. We call n the *length* of this path.

   Represent E by a binary relation `E(source,target)`. A pair (s, t) is in `E` if s and t are vertices in V and (s, t) is an edge in E. Think of s as the source of this edge and t as the target of this edge.

   We say that two vertices v and w in V are connected in G by a path of length n if there exists a path $(v_0, v_1, \ldots, v_n)$ such that $v = v_0$ and $w = v_n$.

   Write an RA expression that returns the set of pairs (v, w) that are connected by a path who length is less than or equal n. (You may assume that $n \geq 1$.)

   Solution: What we need to do is find each pair of vertices v and w that are connected by a path of length m with $1 \leq m \leq n$.

   We begin with the case $m = 1$. For this case, the RA expression is simply `E`. Indeed a pair of vertices v and w are connected by a path of length at least 1 if $(v = v_0, v_1 = w) \in E$. I.e., $(v, w) \in E$.

   Let us assume that we have found an RA expression P_m that finds the pairs (v, w) such that v is connected to w by a path of length m with $m \in [1, n-1]$. (Observe that $P_1 = E$.)

   Let us assume further that the schema of P_m is `(source, target)`. Then the set of pair that are connected by a path whose length is upper bounded by n is given by the RA expression

$$P_1 \cup \cdots \cup P_{n-1} \cup \pi_{P_{n-1}.source, E_n.target}(P_{n-1} \bowtie_{P_{n-1}.target = E_n.source} E_n).$$

Here E_n denotes a copy of E.

For $n = 1$, the expression is E.

For $n = 2$, using proper naming, the expression

$$E_1 \cup \pi_{E_1.source, E_2.target}(E_1 \bowtie_{E_2.target=E_2.source} E_2).$$

And, for $n = 3$, the expression is

$$\begin{array}{l} E_1 \cup \\ \pi_{E_1.source, E_2.target}(E_1 \bowtie_{E_1.target=E_2.source} E_2) \cup \\ \pi_{E_1.source, E_3.target}(\pi_{E_1.source, E_2.target}(E_1 \bowtie_{E_1.target=E_2.source} E_2) \bowtie_{E_2.target=E_3.source} E_3). \end{array}$$

2 Formulating Queries in RA

In the following problems, we will use the data that you can find in the `data.sql` file provided for these problems.

Write the following queries as RA expressions in the standard RA notation. Submit these queries in a .pdf document. In these expressions, you can use the following notations for the relations:

Student	S, S_1, S_2, etc
Book	B, B_1, B_2 etc
Cites	C, C_1, C_2 etc
Major	M, M_1, M_2, etc
Buys	T, T_1, T_2, etc

Then, for each such RA expression, write a SQL query (possibly using the WITH statement) that mimics this expression as discussed above. Submit these queries in a .sql file as usual.

Furthermore, and where possible, avoid using $\times$ or `CROSS JOIN` operator. In addition, where possible, using semijoin operations instead of join operations.

Each of the problem relates back to a problem in Assignment 2. You can consult the SQL solutions for these problem as they may help you in formulating the queries as RA expressions.

7. Find the sid and name of each student who majors in CS and who bought a book that cost more than \$10. (Assignment 2, Problem 1.)

   Solution:

   $\pi_{sid,sname}(S) \ltimes \pi_{sid}(\sigma_{major=CS}(Major) \ltimes \pi_{sid}(T \ltimes (\pi_{bookno}(\sigma_{price>\$10}(B)))))$.

   Equivalently,

   $\pi_{sid,sname}(S) \ltimes (\pi_{sid}(\sigma_{major=CS}(Major)) \cap \pi_{sid}(T \ltimes (\pi_{bookno}(\sigma_{price>\$10}(B)))))$.

   We will use this second expression

```
WITH  CS AS (SELECT sid FROM Major WHERE major = 'CS'),
      B  AS (SELECT bookno FROM Book WHERE price > 10),
      S  AS (SELECT sid FROM Buys NATURAL JOIN B)
SELECT DISTINCT s.sid, s.sname
FROM   Student s NATURAL JOIN (SELECT sid
                               FROM   CS
                               INTERSECT
                               SELECT sid
                               FROM   S) q;
```

8. Find the bookno, title, and price of each book that cites at least two books that cost less than $60. (Assignment 2, Problem 3.)

   Solution:

   Let us the following expressions:

$$\begin{array}{lcl} B & = & \pi_{bookno}(\sigma_{price<60}(Book)) \\ CB & = & \pi_{bookno,citedbookno}(Cites \bowtie_{citedbookno=B.bookno} B) \end{array}$$

   Then the RA expression is

$$Book \ltimes \pi_{bookno}(CB_1 \bowtie_{CB_1.bookno=CB_2.bookno \wedge CB_1.citedbooko \neq C_2.citedbookno} CB_2)$$

```
WITH   B AS (SELECT bookno FROM Book WHERE price < 60),
      CB AS (SELECT c.bookno, c.citedbookno
             FROM    Cites c JOIN B b ON (c.citedbookno = b.bookno))
SELECT bookno, title, price
FROM   Book
       NATURAL JOIN (SELECT DISTINCT cb1.bookno
                     FROM   CB cb1 JOIN CB cb2 ON(cb1.bookno = cb2.bookno AND
                                                  cb1.citedbookno <> cb2.citedbookno)) q;
```

9. Find the bookno, title, and price of each book that was not bought by any Math student. (Assignment 2, Problem 2.)

   Solution:

$$B \ltimes (\pi_{bookno}(B) - \pi_{bookno}(T \ltimes \pi_{bookno}(\sigma_{major='Math'}(M))))$$

```
SELECT b.bookno, b.title, b.price
FROM   Book b
       NATURAL JOIN (SELECT bookno
                     FROM   Book
                     EXCEPT
                     SELECT t.bookno
                     FROM   Buys t NATURAL JOIN (SELECT sid
                                                 FROM   Major
                                                 WHERE  major = 'Math') q1) q2;
```

10. Find the sid and name of each student along with the title and price of the most expensive book(s) bought by that student. (Assignment 2, Problem 4.)

    Solution: Consider the expression

$$T = \pi_{S.sid,S.sname,B.bookno,B.title,B.price}(Student \bowtie Buys \bowtie Book)$$

    Then the expression for the query is

$$\pi_{sid,sname,title,price}(T - (T_1 \bowtie_{T_1.sid=T_2.sid \wedge T_1.price<T_2.price} T_2)).$$

```
WITH T AS (SELECT s.sid, s.sname, b.*
           FROM  student s NATURAL JOIN buys t NATURAL JOIN Book b)

SELECT q.sid, q.sname, q.title, q.price
FROM   (SELECT T.*
        FROM   T
        EXCEPT
        SELECT t1.*
        FROM   T t1 JOIN T t2 ON (t1.sid = t2.sid and t1.price < t2.price)) q;
```

11. Find the booknos and titles of books with the next to highest price. (Assignment 2, Problem 6.)

    Solution:

$$\pi_{B_1.bookno,B_1.title}(B_1 \bowtie_{B_1.price<B_2.price} B_2) - \pi_{B_1.bookno,B_1.title}(\pi_{B_1.bookno,B_1.title,B_2.price}(B_1 \bowtie_{B_1.price<B_2.price} B_2) \bowtie_{B_2.price<B_3.price} B_3)$$

```
WITH E AS (SELECT DISTINCT b1.bookno, b1.title, b2.price
           FROM   Book b1 JOIN Book b2 ON (b1.price < b2.price)),
     F AS (SELECT DISTINCT e.bookno, e.title
           FROM   E e JOIN Book b3 ON (e.price < b3.price))

SELECT bookno, title
FROM   E
EXCEPT
SELECT bookno, title
FROM   F;
```

12. Find the bookno, title, and price of each book that cites a book which is not among the most expensive books. (Assignment 2, Problem 7.)

    Solution:

    Let E be the expression for the bookno's of books that are not the most expensive

$$E \quad = \quad \pi_{B_1.bookno}(B_1 \bowtie_{B_1.price<B_2.price} B_2)$$

$$\pi_{B.*}(Book \ltimes (C \bowtie_{C.citedbookno=E.bookno} E).$$

```
WITH E AS (SELECT DISTINCT b1.bookno AS bno
           FROM   Book b1 JOIN Book b2 ON (b1.price < b2.price))
SELECT DISTINCT b.*
FROM   Book b NATURAL JOIN (Cites c JOIN E e ON (c.citedbookno = e.bno)) q;
```

13. Find the sid and name of each student who has a single major and such that none of the book(s) bought by that student cost less than $40. (Assignment 2, Problem 8.)

    Solution

Consider the following expressions:

$$\begin{aligned} U &= \pi_{sid}(M) - \pi_{M_1.sid}(M_1 \bowtie_{M_1.sid=M_2.sid \wedge M_1.major \neq M_2.major} M_2) \\ B40 &= \pi_{bookno}(\sigma_{price<40}(B)) \end{aligned}$$

Then the expression for the query is

$$\pi_{sid,sname}(S \ltimes (U \cap (\pi_{sid}(S) - \pi_{sid}(T \ltimes B40)))).$$

```
WITH U AS (SELECT sid FROM Major
           EXCEPT
           SELECT m1.sid
           FROM   Major m1 JOIN Major m2 ON (m1.sid=m2.sid AND m1.major <> m2.major)),

           B40 AS (SELECT bookno FROM Book WHERE price < 40)

            SELECT s.sid, s.sname
            FROM   Student s NATURAL JOIN (SELECT sid
                                           FROM   U
                                           INTERSECT
                                           (SELECT sid
                                            FROM   Student
                                            EXCEPT
                                            SELECT sid
                                            FROM   Buys NATURAL JOIN B40)) q;
```

14. Find the bookno and title of each book that is bought by all students who major in both "CS" and in "Math". (Assignment 2, Problem 9.)

    Solution:

    Consider the following expressions:

$$E = \pi_{sid}(\sigma_{major=CS}(M)) \cap \pi_{sid}(\sigma_{major=Math}(M))$$

    Then the RA expression for the query is

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

```
WITH E AS (SELECT sid FROM Major WHERE major = 'CS'
           INTERSECT
           SELECT sid FROM Major WHERE major = 'Math')

SELECT b.bookno, b.title
FROM    Book b
        NATURAL JOIN (SELECT bookno
                      FROM   Book
                      EXCEPT
                      SELECT bookno
                      FROM  (SELECT sid, b.bookno
                             FROM   E CROSS JOIN (SELECT bookno FROM Book) b
                             EXCEPT
                             SELECT sid, bookno FROM Buys) q) q1;
```

15. Find the sid and name of each student who, if he or she bought a book that cost at least than $70, also bought a book that cost less than $30. (Assignment 2, Problem 10.)

Solution:

Consider the following expressions:

$$\begin{aligned} B70 &= \pi_{bookno}(\sigma_{price \geq 70}(B)) \\ B30 &= \pi_{bookno}(\sigma_{price < 30}(B)) \\ T &= \pi_{S.sid,S.sname,B70.bookno}(S \bowtie Buys \bowtie B70) \end{aligned}$$

Then the RA expression for the query is

$$S - \pi_{sid,sname}(T - T \ltimes \pi_{sid}(Buys \ltimes B30))$$

```
WITH B70 AS (SELECT bookno FROM Book WHERE price >= 70),
     B30 AS (SELECT bookno FROM Book WHERE price < 30),
     T   AS (SELECT s.sid, s.sname, b.bookno
             FROM   Student s
                    NATURAL JOIN Buys t
                    NATURAL JOIN (SELECT bookno FROM B70) b)

SELECT sid, sname
FROM   Student
EXCEPT
SELECT sid, sname
FROM   (SELECT sid, sname, bookno
        FROM   T
        EXCEPT
        SELECT sid, sname, bookno
        FROM   T
               NATURAL JOIN (SELECT sid
                             FROM   Buys NATURAL JOIN B30) q1) q;
```

16. Find each pair (s1, s2) where s1 and s2 are the sids of students who have a common major but who did not buy the same set of books. (Assignment 2, Problem 11.)

Solution:

Consider the following expression

$$\begin{aligned} E &= \pi_{M_1.sid,M_2.sid}(M_1 \bowtie_{M_1.major=M_2.major} M_2) \\ S &= \pi_{sid}(Student) \\ D_1 &= \pi_{Buys_1.sid,S_2.sid}(Buys_1 \times S_2 - S_1 \times \pi_{bookno,sid}(Buys_2)) \\ D_2 &= \pi_{S_1.sid,Buys_2.sid}(S_1 \times \pi_{bookno,sid}(Buys_2) - Buys_1 \times S_2) \end{aligned}$$

Then the RA expression for the query is

$$E \cap (D_1 \cup D_2).$$

```
WITH E  AS (SELECT m1.sid as s1, m2.sid as s2
            FROM   Major m1 JOIN Major m2 ON (m1.major = m2.major)),
     S  AS (SELECT sid FROM Student),
     D1 AS (SELECT t1.sid as sid1, bookno, s2.sid as sid2
            FROM   Buys t1 CROSS JOIN S s2),
     D2 AS (SELECT s1.sid as sid1, bookno, t2.sid as sid2
            FROM   S s1 CROSS JOIN Buys t2)
SELECT DISTINCT s1, s2
FROM   E
INTERSECT
SELECT   q.sid1, q.sid2
FROM      ((SELECT sid1, bookno, sid2
            FROM   D1
            EXCEPT
            SELECT sid1, bookno, sid2
            FROM   D2)
            UNION
            (SELECT sid1, bookno, sid2
             FROM   D2
             EXCEPT
             SELECT sid1, bookno, sid2
             FROM   D1)) q;
```