Translating SQL Queries to Relational Algebra Expressions

Dirk Van Gucht[1]

[1] Indiana University

February 26, 2019

Outline

- Objective: Discuss an algorithm that translates a SQL query into an equivalent RA expression
- Motivation: Translate a declaratively specified query into a procedurally specified query
- Restriction: We do not attempt to get an efficient RA expression. Finding an equivalent efficient RA expression is done during query optimization

Strategy

- SQL queries with set predicates will be translated to equivalent SQL queries without set predicates
- WHERE conditions will be eliminated by translating them into FROM clauses using selections and join operations, or by decomposing them into more basic components (then translate these) and then use the set operations union, intersection, and set difference
- These SQL queries will then be translated into RA expressions

Query Forms (Basic case)

SELECT	[DISTINCT] $L(t_1, \ldots, t_n)$
FROM	$R_1\ t_1, \ldots, R_n\ t_n$
WHERE	$C(t_1, \ldots, t_n)$

- $L(t_1, \ldots, t_n)$ is a list of (named) components of the tuple variables t_1 through t_n[1]
- R_1 through R_n are either relations or non-parameterized SQL queries aliased by the tuple variables t_1 through t_n
- $C(t_1, \ldots, t_n)$ is any valid SQL condition involving the components of the variables t_1 through t_n
- We do not handle SQL queries with aggregate functions or queries with subqueries in the FROM clause

[1] There may also appear constants in L; these need special treatment

Query Forms (queries with set operations)

Assuming that Q, Q_1, and Q_2 are SQL queries, we consider the following SQL query forms

```
Q1
UNION
Q2

Q1
INTERSECT
Q2

Q1
EXCEPT
Q2

(Q)
```

Discussion: interaction between projection π and set operations $\cup$, $\cap$, and $-$

- Before we can start with the translation algorithm, it is crucial to discuss how the projection operator π interacts with the set operations $\cup$, $\cap$, and $-$.
- Understanding this is vital for the correct translations of SQL WHERE clauses that use the OR, AND, and NOT boolean operations
- We will consider the following interactions:
  - Projection π and union $\cup$
  - Projection π and intersection $\cap$
  - Projection π and set difference $-$

Projection π distributes over $\cup$

Given RA expressions E_1 and E_2, it is the case that

$$\pi_L(E_1 \cup E_2) = \pi_L(E_1) \cup \pi_L(E_2)$$

An application of this is the following: (C_1 and C_2 are some conditions)

$$\pi_L(\sigma_{C_1 \vee C_2}(E)) = \pi_L(\sigma_{C_1}(E) \cup \sigma_{C_2}(E)) = \pi_L(\sigma_{C_1}(E)) \cup \pi_L(\sigma_{C_2}(E))$$

Translating OR in WHERE clause

```
SELECT   $L(t_1, \ldots, t_n)$
FROM     $R_1\ t_1, \ldots, R_n\ t_n$
WHERE    $C_1(t_1, \ldots, t_n)$ OR $C_2(t_1, \ldots, t_n)$
```

can be translated to become

```
SELECT   $L(t_1, \ldots, t_n)$
FROM     $R_1\ t_1, \ldots, R_n\ t_n$
WHERE    $C_1(t_1, \ldots, t_n)$
UNION
SELECT   $L(t_1, \ldots, t_n)$
FROM     $R_1\ t_1, \ldots, R_n\ t_n$
WHERE    $C_2(t_1, \ldots, t_n)$
```

Projection π does not distributes over $\cap$

Given RA expressions E_1 and E_2, it is the case that

$$\pi_L(E_1 \cap E_2) \subseteq \pi_L(E_1) \cap \pi_L(E_2)$$

But there exist cases where

$$\pi_L(E_1 \cap E_2) \neq \pi_L(E_1) \cap \pi_L(E_2)$$

This complexity features in reasoning about the expression $\pi_L(\sigma_{C_1 \wedge C_2}(E))$ since

$$\pi_L(\sigma_{C_1 \wedge C_2}(E)) = \pi_L(\sigma_{C_1}(E) \cap \sigma_{C_2}(E)) \subseteq \pi_L(\sigma_{C_1}(E)) \cap \pi_L(\sigma_{C_2}(E))$$

But there are cases where

$$\pi_L(\sigma_{C_1 \wedge C_2}(E)) \neq \pi_L(\sigma_{C_1}(E)) \cap \pi_L(\sigma_{C_2}(E))$$

Projection π does not distribute over intersection $\cap$

R

a	b	c
1	1	2
1	2	1

$$\pi_a(\sigma_{b=1 \wedge c=1}(R))$$

a

correct

$$\pi_a(\sigma_{b=1}(R) \cap \sigma_{c=1}(R))$$

a

incorrect

$$\pi_a(\sigma_{b=1}(R)) \cap \pi_a(\sigma_{c=1}(R))$$

a
1

Projection π does not distribute over intersection $\cap$ (Predicate Logic)

The underlying reason why projection π does not distribute over intersection $\cap$ is that an existential quantifier $\exists$ does not distribute over a conjunction $\wedge$ in Predicate Logic

$$\begin{gathered}
\pi_a(\sigma_{b=1 \wedge c=1}(R)) \\
= \\
\pi_a(\sigma_{b=1}(R) \cap \sigma_{c=1}(R)) \\
= \\
\{a \,|\, \exists b \exists c[(R(a,b,c) \wedge b = 1) \wedge (R(a,b,c) \wedge c = 1)]\} \\
\subseteq \\
\{a \,|\, \exists b \exists c(R(a,b,c) \wedge b = 1) \;\wedge\; \exists b \exists c(R(a,b,c) \wedge c = 1)\} \\
= \\
\pi_{b=1}(R) \cap \pi_{c=1}(R)
\end{gathered}$$

Translating AND in WHERE clause (Correct)

```
SELECT   $L(t_1, \ldots, t_n)$
FROM     $R_1\ t_1, \ldots, R_n\ t_n$
WHERE    $C_1(t_1, \ldots, t_n)$ AND $C_2(t_1, \ldots, t_n)$
```

can be translated to become

```
SELECT   $L^q(t_1, \ldots, t_n)$[2]
FROM     (SELECT      $t_1.*, \ldots, t_n.*$
          FROM        $R_1\ t_1, \ldots, R_n\ t_n$
          WHERE       $C_1(t_1, \ldots, t_n)$
          INTERSECT
          SELECT      $t_1.*, \ldots, t_n.*$
          FROM        $R_1\ t_1, \ldots, R_n\ t_n$
          WHERE       $C_2(t_1, \ldots, t_n)$
         ) q
```

[2] $L^q(t_1, \ldots, t_n)$ indicates that the components of t_1 through t_n in L may need to be renamed as components of q

Translating AND in WHERE clause (Incorrect)

```
SELECT   L(t_1, ..., t_n)
FROM     R_1 t_1, ..., R_n t_n
WHERE    C_1(t_1, ..., t_n) AND C_2(t_1, ..., t_n)
```

is not equivalent[3] with

```
SELECT      L(t_1, ..., t_n)
FROM        R_1 t_1, ..., R_n t_n
WHERE       C_1(t_1, ..., t_n)
INTERSECT
SELECT      L(t_1, ..., t_n)
FROM        R_1 t_1, ..., R_n t_n
WHERE       C_2(t_1, ..., t_n)
```

[3] Projection does not distribute over intersection

Translating AND in WHERE clause (Correct)

```
SELECT    e.sid
FROM      Enroll e
WHERE     e.cno = 100 AND e.grade = 'A'
```

can be translated to become

```
SELECT    q.sid
FROM      (SELECT      e.sid, e.cno, e.grade
           FROM        Enroll e
           WHERE       e.cno = 100
           INTERSECT
           SELECT      e.sid, e.cno, e.grade
           FROM        Enroll e
           WHERE       e.grade = 'A'
          ) q
```

Translating AND in WHERE clause (Incorrect)

```
SELECT   e.sid
FROM     Enroll e
WHERE    e.cno = 100 AND e.grade = 'A'
```

is not equivalent[4] with

```
SELECT      e.sid
FROM        Enroll e
WHERE       e.cno = 100
INTERSECT
SELECT      e.sid
FROM        Enroll e
WHERE       e.grade = 'A'
```

[4] Projection does not distribute over intersection

Projection π does not distributes over $-$

Given RA expressions E_1 and E_2, it is the case that

$$\pi_L(E_1 - E_2) \supseteq \pi_L(E_1) - \pi_L(E_2)$$

But there exist cases where

$$\pi_L(E_1 - E_2) \neq \pi_L(E_1) - \pi_L(E_2)$$

This complexity features in reasoning about the expression $\pi_L(\sigma_{\neg C}(E))$ since

$$\pi_L(\sigma_{\neg C}(E)) = \pi_L(E - \sigma_C(E)) \supseteq \pi_L(E) - \pi_L(\sigma_C(E))$$

But there are cases where

$$\pi_L(\sigma_{\neg C}(E)) \neq \pi_L(E) - \pi_L(\sigma_C(E))$$

Projection π does not distribute over set difference $-$

R

a	b
1	1
1	2

$\pi_a(\sigma_{\neg(b=1)}(R))$

a
1

correct

$\pi_a(R - \sigma_{b=1}(R))$

a
1

incorrect

$\pi_a(R) - \pi_a(\sigma_{b=1}(R))$

a

Projection π does not distribute over set difference $-$ (Predicate Logic)

The underlying reason why projection π does not distribute over distribution $\cap$ is that an existential quantifier $\exists$ does not distribute over a conjunction-negation $\wedge\neg$ sequence in Predicate Logic

$$
\begin{gathered}
\pi_a(\sigma_{\neg(b=1)}(R)) \\
= \\
\pi_a(R - \sigma_{b=1}(R)) \\
= \\
\{a \mid \exists b\,(R(a,b) \;\wedge\; \neg(R(a,b) \wedge b = 1))\} \\
\supseteq \\
\{a \mid \exists b\, R(a,b) \;\wedge\; \neg\,\exists b\,(R(a,b) \;\wedge\; b = 1))\} \\
= \\
\pi_a(R) - \pi_a(\sigma_{b=1}(R))
\end{gathered}
$$

Translating NOT in WHERE clause (Correct)

```
SELECT   $L(t_1, \ldots, t_n)$
FROM     $R_1\ t_1, \ldots, R_n\ t_n$
WHERE    NOT $C(t_1, \ldots, t_n)$
```

can be translated to become

```
SELECT   $L^q(t_1, \ldots, t_n)$
FROM     (SELECT   $t_1.*, \ldots, t_n.*$
          FROM     $R_1\ t_1, \ldots, R_n\ t_n$
          EXCEPT
          SELECT   $t_1.*, \ldots, t_n.*$
          FROM     $R_1\ t_1, \ldots, R_n\ t_n$
          WHERE    $C(t_1, \ldots, t_n)$
          ) q
```

Translating NOT in WHERE clause (Incorrect)

```
SELECT   $L(t_1, \ldots, t_n)$
FROM     $R_1\ t_1, \ldots, R_n\ t_n$
WHERE    NOT $C(t_1, \ldots, t_n)$
```

is not equivalent[5] with

```
SELECT   $L(t_1, \ldots, t_n)$
FROM     $R_1\ t_1, \ldots, R_n\ t_n$
EXCEPT
SELECT   $L(t_1, \ldots, t_n)$
FROM     $R_1\ t_1, \ldots, R_n\ t_n$
WHERE    $C(t_1, \ldots, t_n)$
```

[5] Projection do not distribute over set difference

Translating NOT in WHERE clause (Correct)

```
SELECT    e.sid
FROM      Enroll e
WHERE     NOT e.grade = 'A'
```

can be translated to become

```
SELECT    q.sid
FROM      (SELECT    e.sid, e.cno, e.grade
           FROM      Enroll e
           EXCEPT
           SELECT    e.sid, e.cno, e.grade
           FROM      Enroll e
           WHERE     e.grade = 'A'
          ) q
```

Translating NOT in WHERE clause (Incorrect)

```
SELECT    e.sid
FROM      Enroll e
WHERE     NOT e.grade = 'A'
```

is not equivalent[6] with

```
SELECT    e.sid
FROM      Enroll e
EXCEPT
SELECT    e.sid
FROM      Enroll e
WHERE     e.grade = 'A'
```

[6] Projection does not distribute over set difference

Projection π does not distribute over set difference $-$

R

A	B	C
1	1	2
1	2	1

$\pi_A(\sigma_{B=1 \wedge \neg(C=1)}(R))$

A
1

correct

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

A
1

incorrect

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

A

Translating AND NOT in WHERE clause (Correct)

```
SELECT   $L(t_1, \ldots, t_n)$
FROM     $R_1\ t_1, \ldots, R_n\ t_n$
WHERE    $C_1(t_1, \ldots, t_n)$ AND NOT $C_2(t_1, \ldots, t_n)$
```

can be translated to become

```
SELECT   $L^q(t_1, \ldots, t_n)$
FROM     (SELECT   $t_1.*, \ldots, t_n.*$
          FROM     $R_1\ t_1, \ldots, R_n\ t_n$
          WHERE    $C_1(t_1, \ldots, t_n)$
          EXCEPT
          SELECT   $t_1.*, \ldots, t_n.*$
          FROM     $R_1\ t_1, \ldots, R_n\ t_n$
          WHERE    $C_2(t_1, \ldots, t_n)$
         ) q
```

Translating AND NOT in WHERE clause (Incorrect)

```
SELECT   $L(t_1, \ldots, t_n)$
FROM     $R_1\ t_1, \ldots, R_n\ t_n$
WHERE    $C_1(t_1, \ldots, t_n)$ AND NOT $C_2(t_1, \ldots, t_n)$
```

is not equivalent[7] with

```
SELECT   $L(t_1, \ldots, t_n)$
FROM     $R_1\ t_1, \ldots, R_n\ t_n$
WHERE    $C_1(t_1, \ldots, t_n)$
EXCEPT
SELECT   $L(t_1, \ldots, t_n)$
FROM     $R_1\ t_1, \ldots, R_n\ t_n$
WHERE    $C_2(t_1, \ldots, t_n)$
```

[7] Projection do not distribute over set difference

Translating AND NOT in WHERE clause (Correct)

```
SELECT   e.sid
FROM     Enroll e
WHERE    e.cno = 100 AND NOT e.grade = 'A'
```

can be translated to become

```
SELECT   q.sid
FROM     (SELECT   e.sid, e.cno, e.grade
          FROM     Enroll e
          WHERE    e.cno = 100
          EXCEPT
          SELECT   e.sid, e.cno, e.grade
          FROM     Enroll e
          WHERE    e.grade = 'A'
         ) q
```

Translating AND NOT in WHERE clause (Incorrect)

```
SELECT   e.sid
FROM     Enroll e
WHERE    e.cno = 100 AND NOT e.grade = 'A'
```

is not equivalent[8] with

```
SELECT   e.sid
FROM     Enroll e
WHERE    e.cno = 100
EXCEPT
SELECT   e.sid
FROM     Enroll e
WHERE    e.grade = 'A'
```

[8] Projection does not distribute over set difference

Translating AND NOT in WHERE clause (Also correct)

```
SELECT   $L(t_1, \ldots, t_n)$
FROM     $R_1\ t_1, \ldots, R_n\ t_n$
WHERE    $C_1(t_1, \ldots, t_n)$ AND NOT $C_2(t_1, \ldots, t_n)$
```

can be also be translated to become

```
SELECT   $L^q(t_1, \ldots, t_n)$
FROM     (SELECT   $t_1.*, \ldots, t_n.*$
          FROM     $R_1\ t_1, \ldots, R_n\ t_n$
          WHERE    $C_1(t_1, \ldots, t_n)$
          EXCEPT
          SELECT   $t_1.*, \ldots, t_n.*$
          FROM     $R_1\ t_1, \ldots, R_n\ t_n$
          WHERE    $C_1(t_1, \ldots, t_n)$ AND $C_2(t_1, \ldots, t_n)$
         ) q
```

Translating AND NOT in WHERE clause (Also correct)

```
SELECT    e.sid
FROM      Enroll e
WHERE     e.cno = 100 AND NOT e.grade = ‘A’
```

can be translated to become

```
SELECT    q.sid
FROM      (SELECT    e.sid, e.cno, e.grade
           FROM      Enroll e
           WHERE     e.cno = 100
           EXCEPT
           SELECT    e.sid, e.cno, e.grade
           FROM      Enroll e
           WHERE     e.cno= 100 AND e.grade = ‘A’
          ) q
```

Translating EXISTS in WHERE clause (Example)

Let $R(a, b)$ and $S(b, c)$ be two relations.

```
SELECT   r.a
FROM     R r
WHERE    EXISTS (SELECT S.c
                 FROM    S s
                 WHERE   r.b = s.b)
```

In Predicate Logic,

$$\{a \mid \exists b(R(a, b) \wedge \{c \mid S(b, c)\} \neq \emptyset)\}$$
$$=$$
$$\{a \mid \exists b(R(a, b) \wedge \exists c\, S(b, c)))\}$$
$$=$$
$$\{a \mid \exists b \exists c(R(a, b) \wedge S(b, c)\}$$
$$=$$
$$\pi_{R.a}(\sigma_{R.b=S.b}(R \times S))$$

Translating EXISTS in WHERE clause (Example in Predicate Logic)

$$\{a \mid \exists b(R(a,b) \wedge \{c \mid S(b,c)\} \neq \emptyset)\}$$
$$=$$
$$\{a \mid \exists b(R(a,b) \wedge \exists c\, S(b,c)))\}$$
$$=$$
$$\{a \mid \exists b \exists c(R(a,b) \wedge S(b,c)\}$$
$$=$$
$$\pi_{R.a}(\sigma_{R.b=S.b}(R \times S))$$

In SQL,

```
SELECT   DISTINCT r.a
FROM     R r, S s
WHERE    r.b = s.b
```

Translating EXISTS in WHERE clause (Example)

Let $R(a, c)$ and $S(b, c)$ be two relations.

```
SELECT   r.a
FROM     R r
WHERE    EXISTS (SELECT S.c
                 FROM   S s
                 WHERE  r.b = s.b)
```

is translated to

```
SELECT   DISTINCT r.a
FROM     R r, S s
WHERE    r.b = s.b
```

Translating EXISTS in WHERE clause (General case)

```
SELECT  $L(t_1, \ldots, t_n)$
FROM    $R_1\ t_1, \ldots, R_n\ t_n$
WHERE   EXISTS (SELECT 1
                FROM   $S_1\ u_1, \ldots, S_m\ u_m$
                WHERE  $C(u_1, \ldots, u_m, t_1, \ldots, t_n)$)
```

is translated to

```
SELECT  DISTINCT $L(t_1, \ldots, t_n)$
FROM    $R_1\ t_1, \ldots, R_n\ t_n, S_1\ u_1, \ldots, S_m\ u_m$
WHERE   $C(u_1, \ldots, u_m, t_1, \ldots, t_n)$
```

Translating EXISTS in WHERE clause

```
SELECT    s.sid
FROM      Student s
WHERE     EXISTS (SELECT 1
                  FROM    Enroll e, Course c
                  WHERE   e.sid = s.sid AND e.cno = c.cno AND c.dept = ‘CS’)
```

is translated to

```
SELECT    DISTINCT s.sid
FROM      Student s, Enroll e, Course c
WHERE     e.sid = s.sid AND e.cno = c.cno AND c.dept = ‘CS’
```

Translating NOT EXISTS in WHERE clause (Example)

Let $R(a, b)$ and $S(b, c)$ be two relations.

```
SELECT   r.a
FROM     R r
WHERE    NOT EXISTS (SELECT S.c
                     FROM   S s
                     WHERE  r.b = s.b)
```

In Predicate Logic,

$$\{a \mid \exists b(R(a, b) \wedge \{c \mid S(b, c)\} = \emptyset)\}$$
$$=$$
$$\{a \mid \exists b(R(a, b) \wedge \neg\exists c\, S(b, c)))\}$$
$$=$$
$$\{a \mid \exists b(R(a, b) \wedge \neg\exists c\, (R(a, b) \wedge S(b, c)))\}$$
$$=$$
$$\pi_{R.a}(R - \pi_{R.a,R.b}(\sigma_{R.b=S.b}(R \times S)))$$

Translating NOT EXISTS in WHERE clause (Example)

$$\{a \mid \exists b(R(a,b) \land \{c \mid S(b,c)\} = \emptyset)\}$$
$$=$$
$$\pi_{R.a}(R - \pi_{R.a,R.b}(\sigma_{R.b=S.b}(R \times S)))$$

In SQL,

```
SELECT    DISTINCT q.a
FROM      (SELECT       R.a, R.b
           FROM         R r
           EXCEPT
           SELECT       R.a, R.b
           FROM         R r, S s
           WHERE        r.b = s.b
          ) q
```

Translating NOT EXISTS in WHERE clause (Example)

```
SELECT   r.a
FROM     R r
WHERE    NOT EXISTS (SELECT S.c
                     FROM   S s
                     WHERE  r.b = s.b)
```

is translated to

```
SELECT   DISTINCT q.a
FROM     (SELECT      R.a, R.b
          FROM        R r
          EXCEPT
          SELECT      R.a, S.b
          FROM        R r, S s
          WHERE       r.b = s.b
         ) q
```

Translating NOT EXISTS in WHERE clause (General case)

```
SELECT   $L(t_1, \ldots, t_n)$
FROM     $R_1\ t_1, \ldots, R_n\ t_n$
WHERE    NOT EXISTS (SELECT 1
                     FROM   $S_1\ u_1, \ldots, S_m\ u_m$
                     WHERE  $C(u_1, \ldots, u_m, t_1, \ldots, t_n)$)
```

is translated to

```
SELECT   DISTINCT $L^q(t_1, \ldots, t_n)$
FROM     (SELECT          $t_1.*, \ldots, t_n.*$
          FROM            $R_1\ t_1, \ldots, R_n\ t_n$
          EXCEPT
          SELECT          $t_1.*, \ldots, t_n.*$
          FROM            $R_1\ t_1, \ldots, R_n\ t_n, S_1\ u_1, \ldots, S_m\ u_m$
          WHERE           $C(u_1, \ldots, u_m, t_1, \ldots, t_n)$
         ) q
```

Translating NOT EXISTS in WHERE clause

```
SELECT   s.sid
FROM     Student s
WHERE    NOT EXISTS (SELECT 1
                     FROM   Enroll e
                     WHERE e.sid = s.sid AND e.grade = ‘A’)
```

is translated to

```
SELECT   q.sid
FROM     (SELECT   s.sid, s.sname
          FROM     Student s
          EXCEPT
          SELECT   s.sid, s.sname
          FROM     Student s, Enroll e
          WHERE    e.sid = s.sid AND e.grade = ‘A’
         ) q
```

Translating AND NOT EXISTS in WHERE clause

```
SELECT    s.sid
FROM      Student s
WHERE     s.sname = ‘Ann’ AND
          NOT EXISTS (SELECT 1
                      FROM   Enroll e
                      WHERE e.sid = s.sid AND e.grade = ‘A’)
```

is translated to

```
SELECT    q.sid
FROM      (SELECT s.sid, s.sname
           FROM Student s
           WHERE s.sname = ‘Ann’
           EXCEPT
           SELECT s.sid, s.sname
           FROM Student s, Enroll e
           WHERE e.sid = s.sid AND e.grade = ‘A’
          ) q
```

Translating AND NOT EXISTS in WHERE clause (Alternative)

```
SELECT    s.sid
FROM      Student s
WHERE     s.sname = ‘Ann’ AND
          NOT EXISTS (SELECT 1
                      FROM   Enroll e
                      WHERE e.sid = s.sid AND e.grade = ‘A’)
```

is translated to

```
SELECT    q.sid
FROM      (SELECT s.sid, s.sname
           FROM Student s
           WHERE s.sname = ‘Ann’
           EXCEPT
           SELECT s.sid, s.sname
           FROM Student s, Enroll e
           WHERE s.sname=‘Ann’ AND e.sid = s.sid AND e.grade = ‘A’
          ) q
```

Translating IN in WHERE clause

SELECT $L(t_1, \ldots, t_n)$
FROM $R_1\ t_1, \ldots, R_n\ t_n$
WHERE $(t_{i_1}.A_{j_1}, \ldots, t_{i_k}.A_{j_k})$ IN
(SELECT $u_{l_1}.B_{m_1}, \ldots, u_{l_k}.B_{m_k}$
FROM $S_1\ u_1, \ldots, S_m\ u_m$
WHERE $C(u_1, \ldots, u_m, t_1, \ldots, t_n)$)

is translated to

SELECT DISTINCT $L(t_1, \ldots, t_n)$
FROM $R_1\ t_1, \ldots, R_n\ t_n, S_1\ u_1, \ldots, S_m\ u_m$
WHERE $C(u_1, \ldots, u_m, t_1, \ldots, t_n)$ AND
$t_{i_1}.A_{j_1} = u_{l_1}.B_{m_1}$ AND $\cdots$ AND $t_{i_k}.A_{j_k} = u_{l_k}.B_{m_k}$

Translating NOT IN in WHERE clause

```
SELECT    $L(t_1, \ldots, t_n)$
FROM      $R_1\ t_1, \ldots, R_n\ t_n$
WHERE     $(t_{i_1}.A_{j_1}, \ldots, t_{i_k}.A_{j_k})$ NOT IN
              (SELECT $u_{l_1}.B_{m_1}, \ldots, u_{l_k}.B_{m_k}$
               FROM   $S_1\ u_1, \ldots, S_m\ u_m$
               WHERE  $C(u_1, \ldots, u_m, t_1, \ldots, t_n)$)
```

is translated to

```
SELECT   $L^q(t_1, \ldots, t_n)$
FROM     (SELECT    $t_1.*, \ldots, t_n.*$
          FROM      $R_1\ t_1, \ldots, R_n\ t_n$
          EXCEPT
          SELECT    $t_1.*, \ldots, t_n.*$
          FROM      $R_1\ t_1, \ldots, R_n\ t_n, S_1\ u_1, \ldots, S_m\ u_m$
          WHERE     $C(u_1, \ldots, u_m, t_1, \ldots, t_n)$ AND
                    $t_{i_1}.A_{j_1} = u_{l_1}.B_{m_1}$ AND $\cdots$ AND $t_{i_k}.A_{j_k} = u_{l_k}.B_{m_k}$

         ) q
```

Translating θ SOME in WHERE clause

```
SELECT   $L(t_1, \ldots, t_n)$
FROM     $R_1\ t_1, \ldots, R_n\ t_n$
WHERE    $t_{i_1}.A_{j_1}$ $\theta$ SOME
                  (SELECT $u_{l_1}.B_{m_1}$
                   FROM   $S_1\ u_1, \ldots, S_m\ u_m$
                   WHERE  $C(u_1, \ldots, u_m, t_1, \ldots, t_n)$)
```

is translated to

```
SELECT   DISTINCT $L(t_1, \ldots, t_n)$
FROM     $R_1\ t_1, \ldots, R_n\ t_n, S_1\ u_1, \ldots, S_m\ u_m$
WHERE    $C(u_1, \ldots, u_m, t_1, \ldots, t_n)$ AND
         $t_{i_1}.A_{j_1}\ \theta\ u_{l_1}.B_{m_1}$
```

Translating θ ALL in WHERE clause

```
SELECT   $L(t_1, \ldots, t_n)$
FROM     $R_1\ t_1, \ldots, R_n\ t_n$
WHERE    $t_{i_1}.A_{j_1}$ $\theta$ ALL
                    (SELECT $u_{l_1}.B_{m_1}$
                     FROM    $S_1\ u_1, \ldots, S_m\ u_m$
                     WHERE   $C(u_1, \ldots, u_m, t_1, \ldots, t_n)$)
```

is translated to

```
SELECT   $L^q(t_1, \ldots, t_n)$
FROM     (SELECT     $t_1.*, \ldots, t_n.*$
          FROM       $R_1\ t_1, \ldots, R_n\ t_n$
          EXCEPT
          SELECT     $t_1.*, \ldots, t_n.*$
          FROM       $R_1\ t_1, \ldots, R_n\ t_n, S_1\ u_1, \ldots, S_m\ u_m$
          WHERE      $C(u_1, \ldots, u_m, t_1, \ldots, t_n)$ AND
                     NOT $t_{i_1}.A_{j_1}$ $\theta$ $u_{l_1}.B_{m_1}$

         ) q
```

Translating θ ALL in WHERE clause

```
SELECT   p.pid
FROM     Person p
WHERE    p.age ≤ ALL
                (SELECT p1.age
                 FROM    Person p1)
```

is translated to

```
SELECT   DISTINCT q.pid
FROM     (SELECT     p.pid, p.age
          FROM       Person p
          EXCEPT
          SELECT     p.pid, p.age
          FROM       Person p, Person p1
          WHERE      NOT p.age ≤ p1.age
         ) q
```

Translating SQL with doubly-nested set predicates

“Find the sid of each student who is only enrolled CS courses."

```
SELECT    s.sid
FROM      Student s
WHERE     NOT EXISTS (SELECT 1
                      FROM   Enroll e
                      WHERE e.sid = s.sid AND
                       e.cno NOT IN (SELECT c.cno
                                     FROM    Course c
                                     WHERE   c.dept = ‘CS’))
```

Eliminate first-level NOT EXISTS predicate

“Find the sid of each student who is only enrolled in CS courses."

```
SELECT   q1.ssid
FROM     (SELECT s.sid AS ssid, s.sname
          FROM   Student s
          EXCEPT
          SELECT s.sid, s.sname
          FROM   Student s, Enroll e
          WHERE e.sid = s.sid AND
                      e.cno NOT IN (SELECT c.cno
                                    FROM   Course c
                                    WHERE  c.dept = ‘CS’)) q1
```

Translating NOT EXISTS in WHERE

```
SELECT L(t1, ..., tn)
FROM R1 t1, ..., Rn tn
WHERE NOT EXISTS (SELECT 1
           FROM S1 u1, ..., Sm um
           WHERE C(u1, ..., um, t1, ..., tn))
```

is translated to

```
SELECT DISTINCT Lq(t1, ..., tn)
FROM (SELECT t1.*, ..., tn.*
    FROM R1 t1, ...,Rn tn
    EXCEPT
    SELECT t1.*, ..., tn.*
    FROM R1 t1, ...,Rn tn, S1 u1, ..., Sm um
    WHERE C(u1, ..., um, t1, ..., tn)
    ) q
```

Eliminate second-level AND NOT IN

“Find the sid of each student who is only enrolled in CS courses."

```
SELECT    q1.ssid
FROM      (SELECT s.sid AS ssid, s.sname
           FROM   Student s
           EXCEPT
           SELECT q2.ssid, s.sname
           FROM (SELECT s.sid AS ssid, s.sname, e.sid, e.cno, e.grade
                 FROM   Student s, Enroll e
                 WHERE e.sid = s.sid
                 EXCEPT
                 (SELECT s.sid, s.sname, e.sid, e.cno, e.grade
                  FROM   Student s, Enroll e, Course c
                  WHERE e.cno = c.cno AND c.dept = ‘CS’ ) q2) q1
```

Translating NOT IN in WHERE

```
SELECT L(t1, ..., tn)
FROM R1 t1, ...,Rn tn
WHERE (ti1.Aj1, ..., tik.Ajk ) NOT IN
        (SELECT ul1.Bm1 , ..., ulk.Bmk
        FROM S1 u1, ..., Sm um
        WHERE C(u1, ..., um; t1, ..., tn))
```

is translated to

```
SELECT Lq(t1, ..., tn)
FROM (SELECT t1:  , ..., tn:
     FROM R1 t1, ...,Rn tn
     EXCEPT
     SELECT t1:  , ..., tn:
     FROM R1 t1, ...,Rn tn; S1 u1, ..., Sm um
     WHERE C(u1, ..., um; t1, ..., tn) AND
     ti1.Aj1 = ul1.Bm1 AND ...  AND tik.Ajk = ulk.Bmk
     ) q
```

Translating SQL with doubly-nested set predicates

“Find the sid of each student who is enrolled in all CS courses."

```
SELECT    s.sid
FROM      Student s
WHERE     NOT EXISTS (SELECT 1
                      FROM   Course c
                      WHERE c.dept = ‘CS’ AND
                       c.cno NOT IN (SELECT e.cno
                                     FROM    Enroll e
                                     WHERE    e.sid = s.sid))
```

Eliminate first-level NOT EXISTS predicate

“Find the sid of each student who is only enrolled in CS courses."

```
SELECT    q1.ssid
FROM       (SELECT s.sid AS ssid, s.sname
            FROM   Student s
            EXCEPT
            SELECT s.sid, s.sname
            FROM   Student s, Course c
            WHERE c.dname = ’CS’ AND
                           c.cno NOT IN (SELECT e.cno
                                         FROM    Enroll e
                                         WHERE   e.sid = s.sid)) q1
```

Eliminate second-level AND NOT IN

"Find the sid of each student who is enrolled in all CS courses."

```
SELECT    q1.ssid
FROM      (SELECT s.sid AS ssid, s.sname
           FROM   Student s
           EXCEPT
           SELECT q2.ssid, s.sname
           FROM (SELECT s.sid AS ssid, s.sname, c.cno, c.dname
                 FROM   Student s, Course c
                 WHERE c.dname = 'CS'
                 EXCEPT
                 (SELECT s.sid, s.sname, c.cno, c.dname
                  FROM   Student s, Enroll e, Course c
                  WHERE e.cno = c.cno AND e.sid = s.sid ) q2) q1
```

Moving WHERE condition to FROM clause

- In the previous slides we have shown how set predicates can be translated
- After that process, we will have SQL queries wherein the WHERE clauses consist of boolean combinations of conditions of the form
  - $t.A\,\theta\,\mathbf{a}$; or
  - $t_i.A\,\theta\,t_j.B$
- In the following slides we will show how these WHERE clauses can be moved to FROM clauses
- We might also have queries without a WHERE clause and/or without a FROM clause; these require special treatment

SQL queries without WHERE and FROM clauses

SELECT **a** AS A

This query is translated to the RA expression

$$(A : \mathbf{a})$$

SQL queries without WHERE clause

$$\begin{array}{ll} \text{SELECT} & L(t_1, \ldots, t_n) \\ \text{FROM} & R_1\ t_1, \ldots, R_n\ t_n \end{array}$$

This query is translated to the RA expression

$$\pi_{L(t_1,\ldots,t_n)}(R_1 \times \cdots \times R_n)$$

Moving WHERE condition to FROM clause (condition on at least three relations)

Assume that the condition C applies to a least three relations, i.e., the condition is of the form $C(t_{i_1}, t_{i_2}, t_{i_3}, \dots, t_{i_k})$ with $k \geq 3$

SELECT	$L(t_1, \dots, t_n)$
FROM	$R_1\ t_1, R_2\ t_2, R_3\ t_3, \dots, R_n\ t_n$
WHERE	$C(t_{i_1}, t_{i_2}, t_{i_3}, \dots, t_{i_k})$

We can now introduce the CROSS JOIN in the FROM clause by replacing each ',' with CROSS JOIN

SELECT	$L(t_1, \dots, t_n)$
FROM	$R_1\ t_1$ CROSS JOIN $R_2\ t_2$ CROSS JOIN $R_3\ t_3$ CROSS JOIN $\cdots$ CROSS JOIN $R_n\ t_n$
WHERE	$C(t_{i_1}, t_{i_2}, t_{i_3}, \dots, t_{i_k})$

In RA,

$$\pi_{L(t_1,\dots,t_n)}(\sigma_{C(t_{i_1},t_{i_2},t_{i_3},\dots,t_{i_k})}(R_1 \times R_2 \times R_3 \times \cdots \times R_n))$$

Moving WHERE condition to FROM clause (condition on at least three relations) Example

$$\begin{array}{ll}
\text{SELECT} & L(t_1, t_2, t_3) \\
\text{FROM} & R_1\ t_1, R_2\ t_2, R_3\ t_3 \\
\text{WHERE} & t_1.A_1\ \theta_1\ t_2.A_2\ \text{OR}\ t_2.A_3\ \theta_2\ t_3.A_4
\end{array}$$

is translated to

$$\pi_{L(t_1,t_2,t_3)}(\sigma_{t_1.A_1\ \theta_1\ t_2.A_2 \vee t_2.A_3\ \theta_2\ t_3.A_4}(R_1 \times R_2 \times R_3))$$

Moving WHERE condition to FROM clause (condition on single relation)

SELECT	$L(t_1, \ldots, t_n)$
FROM	$R_1\ t_1, R_2\ t_2, \ldots, R_i\ t_i, \ldots, R_n\ t_n$
WHERE	$C(t_i)$ [AND $C'(t_{i_1}, \ldots, t_{i_k})$]

Observe that $C(t_i)$ is only a condition on R_i. This query is translated to

SELECT $L(t_1, \ldots, t_n)$
FROM $R_1\ t_1, R_2\ t_2 \ldots,$
(SELECT $t_i.*$ FROM R_i WHERE $C(t_i)$) $t_i, \ldots, R_n\ t_n$
[WHERE $C'(t_{i_1}, \ldots, t_{i_k})$]

Moving WHERE condition to FROM clause (condition on single relation)

```
SELECT $L(t_1, \dots, t_n)$
FROM    $R_1\ t_1, R_2\ t_2, \dots,$
           (SELECT $t_i.*$ FROM $R_i$ WHERE $C(t_i)$) $t_i, \dots, R_n\ t_n$
[WHERE  $C'(t_{i_1}, \dots, t_{i_k})$]
```

We can now introduce the CROSS JOIN in the FROM clause by replacing each ‘,’ with CROSS JOIN

```
SELECT $L(t_1, \dots, t_n)$
FROM    $R_1\ t_1$ CROSS JOIN $R_2\ t_2$ CROSS JOIN $\cdots$ CROSS JOIN
           (SELECT $t_i.*$ FROM $R_i$ WHERE $C(t_i)$) $t_i$ CROSS JOIN $\cdots$ CROSS JOIN $R_n\ t_n$
[WHERE  $C'(t_{i_1}, \dots, t_{i_k})$ ]
```

Which, in the notation of RA, corresponds to the expression

$$\pi_{L(t_1,\dots,t_n)}(\sigma_{C'(t_{i_1},\dots,t_{i_k})}(R_1 \times R_2 \times \cdots \times \sigma_{C(t_i)}(R_i) \times \cdots \times R_n))$$

or, when $C'(t_{i_1}, \dots, t_{i_k})$ is missing,

$$\pi_{L(t_1,\dots,t_n)}(R_1 \times R_2 \times \cdots \times \sigma_{C(t_i)}(R_i) \times \cdots \times R_n)$$

Moving WHERE condition to FROM clause (condition on two relations)

SELECT $L(t_1, \ldots, t_n)$
FROM $R_1\ t_1, \ldots, R_i\ t_i, \ldots, R_j\ t_j, \ldots, R_n\ t_n$
WHERE $C(t_i, t_j)$ [AND $C'(t_{i_1}, \ldots, t_{i_k})$]

Observe that $C(t_i, t_j)$ is a condition relating R_i and R_j. This query is translated to

SELECT $L(t_1, \ldots, t_n)$
FROM $R_1\ t_1, \ldots, R_{i-1}\ t_{j-1}, R_{i+1}\ t_{i+1}, \ldots, R_{j-1}\ t_{j-1}, R_{j+1}\ t_{j+1}, \ldots R_n\ t_n,$
$R_i\ t_i$ JOIN $R_j\ t_j$ ON $C(t_i, t_j)$
[WHERE $C'(t_{i_1}, \ldots, t_{i_k})$]

Moving WHERE condition to FROM clause (condition on two relations)

SELECT $L(t_1, \ldots, t_n)$
FROM $R_1\ t_1, \ldots, R_{i-1}\ t_{j-1}, R_{i+1}\ t_{i+1}, \ldots, R_{j-1}\ t_{j-1}, R_{j+1}\ t_{j+1}, \ldots R_n\ t_n,$
$R_i\ t_i$ JOIN $R_j\ t_j$ ON $C(t_i, t_j)$
[WHERE $C'(t_{i_1}, \ldots, t_{i_k})$]

Recalling that each ',' in the FROM clause corresponds to a CROSS JOIN, this query can be formulated in RA as follows:

$$\pi_{L(t_1,\ldots,t_n)}(\sigma_{C'(t_{i_1},\ldots,t_{i_k})}(R_1 \times \cdots \times R_{i-1} \times R_{i+1} \times \cdots \times R_{j-1} \times R_{j+1} \times \cdots \times R_n \times (R_i \bowtie_{C(t_i,t_j)} R_j)))$$

or, when $C'(t_{i_1}, \ldots, t_{i_k})$ is missing

$$\pi_{L(t_1,\ldots,t_n)}(R_1 \times \cdots \times R_{i-1} \times R_{i+1} \times \cdots \times R_{j-1} \times R_{j+1} \times \cdots \times R_n \times (R_i \bowtie_{C(t_i,t_j)} R_j))$$

Moving WHERE condition to FROM clause (natural join condition on two relations)

Assume that $A_1, \ldots, A_k$ are the common attributes of R_i and R_j and that $C(t_i, t_j)$ is the condition

$$t_i.A_1 = t_j.A_1 \text{ AND} \cdots \text{ AND } t_i.A_k = t_j.A_k$$

then

```
SELECT $L(t_1, \ldots, t_n)$
FROM   $R_1\ t_1, \ldots, R_{i-1}\ t_{j-1}, R_{i+1}\ t_{i+1}, \ldots, R_{j-1}\ t_{j-1}, R_{j+1}\ t_{j+1}, \ldots R_n\ t_n,$
         $R_i\ t_i$ JOIN $R_j\ t_j$ ON $C(t_i, t_j)$
[WHERE $C'(t_{i_1}, \ldots, t_{i_k})$]
```

is translated to

```
SELECT $L(t_1, \ldots, t_n)$
FROM   $R_1\ t_1, \ldots, R_{i-1}\ t_{j-1}, R_{i+1}\ t_{i+1}, \ldots, R_{j-1}\ t_{j-1}, R_{j+1}\ t_{j+1}, \ldots R_n\ t_n,$
         $R_i\ t_i$ NATURAL JOIN $R_j\ t_j$
[WHERE $C'(t_{i_1}, \ldots, t_{i_k})$]
```

In RA,

$$\pi_{L(t_1,\ldots,t_n)}(\sigma_{C'(t_{i_1},\ldots,t_{i_k})}(R_1 \times \cdots \times R_{i-1} \times R_{i+1} \times \cdots \times R_{j-1} \times R_{j+1} \times \cdots \times R_n \times (R_i \bowtie R_j)))$$

SQL queries with set operations UNION, INTERSECT, or EXCEPT

Assuming Q_1 and Q_2 SQL queries, the queries of the form

Q_1
UNION [INTERSECT | EXCEPT]
Q_2

can be translated to RA as follows

$$E_{Q_1} \cup [\cap \mid -] E_{Q_2}$$

where E_{Q_1} and E_{Q_2} are the RA expressions corresponding to Q_1 and Q_2

Example

“Find the sid of each student who is only enrolled in CS courses."

```
SELECT    q1.ssid
FROM      (SELECT s.sid AS ssid, s.sname
           FROM    Student s
           EXCEPT
           SELECT q2.ssid, s.sname
           FROM (SELECT s.sid AS ssid, s.sname, e.sid, e.cno
                 FROM    Student s, Enroll e
                 WHERE e.sid = s.sid
                 EXCEPT
                 (SELECT s.sid, s.sname, e.sid, e.cno
                  FROM    Student s, Enroll e, Course c
                  WHERE e.cno = c.cno AND c.dept = ‘CS’ ) q2) q1
```

Example

“Find the sid of each student who is only enrolled in CS courses."

```
SELECT      q1.ssid
FROM         (SELECT s.sid AS ssid, s.sname
              FROM    Student s
              EXCEPT
              SELECT q2.ssid, s.sname
              FROM (SELECT s.sid AS ssid, s.sname, e.sid, e.cno
                          FROM    Student s NATURAL JOIN Enroll e
                          EXCEPT
                          (SELECT s.sid, s.sname, e.sid, e.cno
                           FROM    Student s CROSS JOIN Enroll E
                                      NATURAL JOIN (SELECT c.* FROM Course c WHERE dept = ‘CS’) c) q2) q1
```

is translated to

$$\pi_{sid}(\pi_{sid,sname}(S) - \pi_{S.sid,sname}(\pi_{S.sid,sname,E.sid,cno}(S \bowtie E) - \pi_{S.sid,sname,E.sid,cno}(S \times E \bowtie \sigma_{dept=\text{‘}CS\text{’}}(C))))$$

where S, E, and C denote Student, Enroll, and Course, respectively.

Example (Optimization)

“Find the sid of each student who is only enrolled in CS courses."

$$\pi_{sid}(\pi_{sid,sname}(S) - \pi_{S.sid,sname}(\pi_{S.sid,sname,E.sid,cno}(S \bowtie E) - \pi_{S.sid,sname,E.sid,cno}(S \times E \bowtie \sigma_{dept='CS'}(C))))$$

This can be optimized to the RA expression

$$\pi_{sid}(S) - \pi_{sid}(\pi_{sid,cno}(E) - \pi_{sid,cno}(E) \ltimes \pi_{cno}(\sigma_{dept='CS'}(C)))$$

Notice that this is the RA expression for the only set semijoin

If furthermore the schema of Enroll is (sid,cno), this expression becomes

$$\pi_{sid}(S) - \pi_{sid}(E - E \ltimes CS)) = \pi_{sid} - \pi_{sid}(E \overline{\ltimes} CS)$$

where CS denotes the RA expression $\pi_{cno}(\sigma_{dept='CS'}(C))$