Set Joins and Semijoins in Relational Algebra

Dirk Van Gucht[1]

[1] Indiana University

Joins and semijoins

- Two types of joins
  - Regular joins: these are joins that compute relationships between objects (o_1, o_2) based on tuple component comparisons (see the previous lecture on regular joins.)
  - Set joins: these are joins that compute relationships between objects (o_1, o_2) based on set comparisons between sets $S(o_1)$ and $S(o_2)$ that are associated with objects o_1 and o_2, respectively.
    - Think for example of finding pairs (s, p) such that student s only takes courses taught by professors.
    - If $S(s)$ denotes the set of courses taken by s and $S(p)$ denotes the set of coursed taught by p, then we need to check if $C(s) \subseteq C(p)$.
- Analogously, there are also two types of semijoins: regular semijoins and set semijoins

Set joins (Motivation)

- Consider the relations Enroll(sid,cno) and TaughtBy(cno,pid).
- (s, c) is in Enroll when student s is enrolled in course c
- (c, p) is in TaughtBy when course c is taught by professor p
- We are interested in student-professor relationships (s, p) based on some complex join condition.
- "Find the student-professor pairs (s, p) such that student s takes $[some|not\ only|not\ all|no|only|all|at\ least\ 2|\cdots]$ course(s) taught by professor p."

Set joins (Motivation)

- Consider the relations Patient(pid, sympton) and Disease(sympton,disease).
- (p, s) is in Patient when patient p exhibits health sympton s
- (s, d) is in Disease when s is a sympton of disease d
- "Find the patient-disease pairs (p, d) such that patient p has $[some|not\ only|not\ all|no|only|all|at\ least\ 2|\cdots]$ symptons associated with disease d."

Set semijoins (Motivation)

- Consider the relations Patient(pid,sympton) and Flu(sympton).
- We are interested in finding patient that satisfy some complex condition.
- "Find each patient p such that p has $[some|not\ only|not\ all|no|only|all|at\ least\ 2|\cdots]$ flu symptoms.

Set semijoins (Advanced Google Search Engine)

- Consider the relations Document(doc,word) and the set of words $\{Advanced, Database, Concept\}$.
- A pair $(d, w) \in Document$ when document d contains the word w.
- Advanced Google Search supports the following queries: "Find each document d such that d contains $[some(any)|no|all]$ words in $\{Advanced, Database, Concepts\}$.
- You can try this out for the *all* query and you will see documents that refer to B561.
- $https://www.google.com/advanced\_search$
- So set semijoins are frequently used in real search applications

Set joins and semijoin in RA (Objective)

- Derive relational algebra expressions for set joins and semijoins that supports queries with quantifiers.
- The technique we use for this follows the Venn-diagram method used in the lecture on "Queries with Quantifiers" to derive SQL queries for such queries.
- We will discuss how the derived RA expressions provide deep insight into the time and space complexities for different set joins
- Note: this is a very technical lecture that relies on understanding of logic and set theory

Set joins in RA (Introduction-Terminology)

We begin by introducing some terminology and state our assumptions.

- Denote by $\mathcal{S}$ the set of all student sids
- Denote by $\mathcal{P}$ the set of all professor pids
- Denote by $\mathcal{C}$ the set of all course cnos
- Denote by E the relation *Enroll*
- Denote by T the relation *TaughtBy*
- We will assume that the natural foreign key constraints hold:

$$\begin{aligned} \pi_{sid}(E) &\subseteq \mathcal{S} \\ \pi_{cno}(E) &\subseteq \mathcal{C} \\ \pi_{cno}(T) &\subseteq \mathcal{C} \\ \pi_{pid}(T) &\subseteq \mathcal{P} \end{aligned}$$

Set joins in RA (Parameterized sets)

- Let s be a student sid (i.e., $s \in \mathcal{S}$) and let p be a professor pid (i.e., $p \in \mathcal{P}$).
- We will consider sets parameterized by s and p.
- Both these sets are subsets of $\mathcal{C}$, i.e., the set of cnos of courses.
  - $E(s)$ is the set of cnos of courses in which student s is enrolled.[1]

$$E(s) = \{c \mid (s, c) \text{ is a tuple in the Enroll relation}\}$$

  - $T(p)$ is the set of cnos of courses taught by professor p.[2]

$$T(p) = \{c \mid (c, p) \text{ is a tuple in the TaughtBy relation}\}$$

[1] If s is not enrolled in any course then $E(s) = \emptyset$.

[2] If p does not teach any course then $T(p) = \emptyset$.

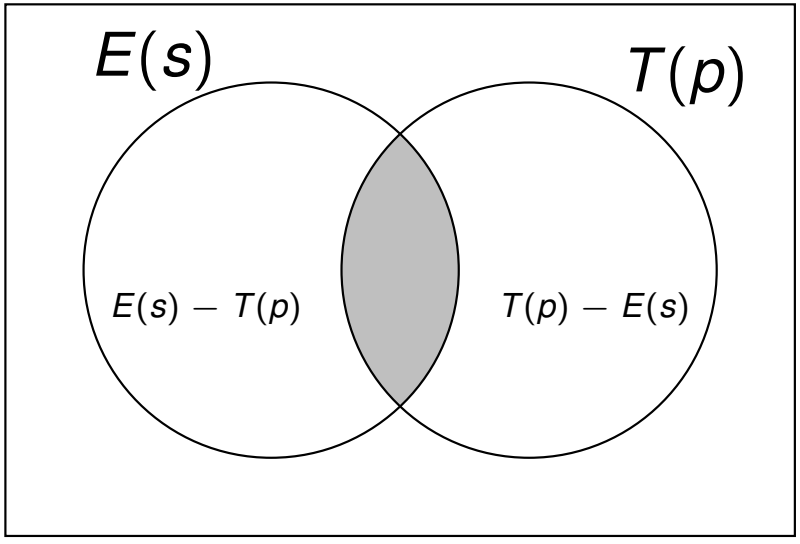
Set joins (Regions)

- For each student-professor pair (s, p), the sets $E(s)$ and $T(p)$ define 4 pairwise disjoint "regions" in the Venn diagram associated with these sets.[3]
- These regions partition the set of cno of courses $\mathcal{C}$:

$E(s) \cap T(p)$	$=$	the courses enrolling student s and taught by professor p
$E(s) - T(p)$	$=$	the courses enrolling student s but not taught by professor p
$T(p) - E(s)$	$=$	the courses not enrolling student s but taught by professor p
$\mathcal{C} - (E(s) \cup T(p))$	$=$	the courses not enrolling student s and not taught by professor p

[3] It is recommended that you revisit the lecture of "Queries with Quantifiers" to further understand this Venn diagram.

Venn diagram and regions for $E(s)$ and $T(p)$



The gray region is $E(s) \cap T(p)$

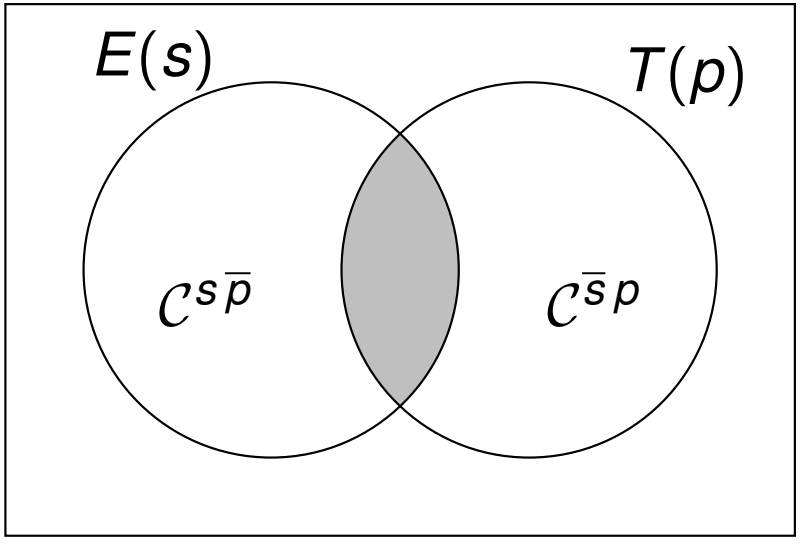
The region outside both $E(s)$ and $T(p)$ is $\mathcal{C} - (E(s) \cup T(p))$

Set joins (Regions)

- For each student-professor pair (s, p), the sets $E(s)$ and $T(p)$ define 4 pairwise disjoint "regions" that partition the set of cnos of courses $\mathcal{C}$:
- We use the following notations to denote these regions:

Region	Notation
$E(s) \cap T(p)$	$\mathcal{C}^{sp}$
$E(s) - T(p)$	$\mathcal{C}^{s\overline{p}}$
$T(p) - E(s)$	$\mathcal{C}^{\overline{s}p}$
$\mathcal{C} - (E(s) \cup T(p))$	$\mathcal{C}^{\overline{s}\overline{p}}$

Venn diagram and regions for $E(s)$ and $T(p)$



The gray region is $\mathcal{C}^{s\,p}$

The region outside both $E(s)$ and $T(p)$ is $\mathcal{C}^{\bar{s}\,\bar{p}}$. (From now on, we will ignore this region in the lecture.)

Set joins (Region analysis)

- Consider again the regions $\mathcal{C}^{sp}$, $\mathcal{C}^{s\overline{p}}$, and $\mathcal{C}^{\overline{s}p}$.
- These regions can be either not-empty or empty.
- Depending on this, we can distinguish 6 set joins. They correspond to the some, no, not only, only, not all, and all quantifiers we considered earlier:

$$\begin{aligned}
\mathcal{C}^{sp} \neq \emptyset &\quad \leftrightarrow \quad \text{some} \\
\mathcal{C}^{sp} = \emptyset &\quad \leftrightarrow \quad \text{no} \\
\mathcal{C}^{s\overline{p}} \neq \emptyset &\quad \leftrightarrow \quad \text{not only} \\
\mathcal{C}^{s\overline{p}} = \emptyset &\quad \leftrightarrow \quad \text{only} \\
\mathcal{C}^{\overline{s}p} \neq \emptyset &\quad \leftrightarrow \quad \text{not all} \\
\mathcal{C}^{\overline{s}p} = \emptyset &\quad \leftrightarrow \quad \text{all}
\end{aligned}$$

- Once the analysis of these cases is understood, it becomes straightforward to study a very large set of other interesting (and practical) set joins.

Set join (not-empty cases)

- We begin with the set joins for some, not only, and not all. I.e., those corresponding to non-empty regions.
- To perform our analysis, it helps to formulate precisely the queries we want to express as RA expressions:

$$
\begin{array}{lll}
\text{some} & \leftrightarrow & \{(s,p)|\exists c : c \in \mathcal{C}^{sp}\} \\
 & \leftrightarrow & \{(s,p)|\exists c : c \in E(s) \cap T(p)\} \\
 & \leftrightarrow & \{(s,p)|\exists c : c \in E(s) \wedge c \in T(p)\} \\
\text{not only} & \leftrightarrow & \{(s,p)|\exists c : c \in \mathcal{C}^{s\bar{p}}\} \\
 & \leftrightarrow & \{(s,p)|\exists c : c \in (E(s) - T(p))\} \\
 & \leftrightarrow & \{(s,p)|\exists c : c \in E(s) \wedge c \notin T(p)\} \\
\text{not all} & \leftrightarrow & \{(s,p)|\exists c : c \in \mathcal{C}^{\bar{s}p}\} \\
 & \leftrightarrow & \{(s,p)|\exists c : c \in (T(p) - E(s))\} \\
 & \leftrightarrow & \{(s,p)|\exists c : c \in T(p) \wedge c \notin E(s))\}
\end{array}
$$

Set join (some case)

- Recall

$$\begin{aligned} \text{some} \quad &\leftrightarrow \quad \{(s,p)|\exists c : c \in \mathcal{C}^{s\,p}\} \\ &\leftrightarrow \quad \{(s,p)|\exists c : c \in E(s) \wedge c \in T(p)\} \\ &\leftrightarrow \quad \{(s,p)|\exists c : E(s,c) \wedge T(c,p)\} \end{aligned}$$

- We first need to find an RA expression for the set

$$\{(s,c,p)|E(s,c) \wedge T(c,p)\}.$$

- But this is easy since, by the definition of the natural join $\bowtie$

$$\{(s,c,p)|E(s,c) \wedge T(c,p)\} = E \bowtie T.$$

Consequently, the RA expression for some is

$$\pi_{sid,pid}(E \bowtie T)$$

Set join (not only case)

- Recall

$$\begin{aligned} \text{not only} &\leftrightarrow \{(s,p)|\exists c : c \in \mathcal{C}^{s\bar{p}}\} \\ &\leftrightarrow \{(s,p)|\exists c : c \in E(s) \wedge \neg(c \in T(p))\} \\ &\leftrightarrow \{(s,p)|\exists c : E(s,c) \wedge \neg T(c,p)\} \end{aligned}$$

- We first need to find an RA expression for the set

$$\{(s,c,p)|E(s,c) \wedge \neg T(c,p))\}.$$

$$\begin{aligned} \{(s,c,p)|E(s,c) \wedge \neg T(c,p)\} &= \{(s,c,p)|(E(s,c) \wedge p \in \mathcal{P}) \wedge \neg(E(s,c) \wedge T(c,p))\} \quad (1) \\ &= \{(s,c,p)|(s,c,p) \in (E \times \mathcal{P}) \wedge \neg((s,c,p) \in E \bowtie T)\} \\ &= (E \times \mathcal{P}) - (E \bowtie T) \end{aligned}$$

The equality (1) will be shown on the next slides.
Consequently, the RA expression for not only is

$$\pi_{sid,pid}((E \times \mathcal{P}) - (E \bowtie T))$$

Set join (not only case using natural join)

- To prove equality (1), we need to show that the following statements are logically equivalent

$$E(s, c) \wedge \neg T(c, p)$$

$$(E(s, c) \wedge p \in \mathcal{P}) \wedge \neg(E(s, c) \wedge T(c, p))$$

Since, clearly $p \in \mathcal{P}$ is true, we have to show that the following statements are equivalent:[4]

$$E(s, c) \wedge \neg T(c, p)$$

$$E(s, c) \wedge \neg(E(s, c) \wedge T(c, p)$$

[4] We are assuming that there is at least one professor.

Set join (not only case using natural join)

We need to show that the following statements are equivalent:

$$E(s,c) \wedge \neg T(c,p) \qquad \text{(I)}$$
$$E(s,c) \wedge \neg(E(s,c) \wedge T(c,p)) \qquad \text{(II)}$$

To see this, you need to determine if statements (I) and (II) have the same truth values. This is the case as shown in the following truth table:

$E(s,c)$	$T(c,p)$	Statement (I)	Statement (II)
true	true	false	false
true	false	true	true
false	true	false	false
false	false	false	false

Set join (not all case)

- Recall

$$\begin{aligned} \text{not all} \quad &\leftrightarrow \quad \{(s,p)|\exists c : c \in \mathcal{C}^{\bar{s}\,p}\} \\ &\leftrightarrow \quad \{(s,p)|\exists c : c \in T(p) \wedge \neg(c \in E(s))\} \\ &\leftrightarrow \quad \{(s,p)|\exists c : T(c,p) \wedge \neg E(s,c))\} \end{aligned}$$

- We first need to find an RA expression for the set

$$\{(s,c,p)|T(c,p) \wedge \neg E(s,c))\}.$$

$$\begin{aligned} \{(s,c,p)|T(c,p) \wedge \neg E(s,c))\} &= \{(s,c,p)|(s \in \mathcal{S} \wedge T(c,p)) \wedge \neg(E(s,c) \wedge T(c,p))\} \\ &= \{(s,c,p)|(s,c,p) \in (\mathcal{S} \times T) \wedge \neg((s,c,p) \in E \bowtie T)\} \\ &= (\mathcal{S} \times T) - (E \bowtie T) \end{aligned} \quad (1)$$

The equality (1) will be shown on the next slides.
Consequently, the RA expression for not all is

$$\pi_{sid,pid}((\mathcal{S} \times T) - (E \bowtie T))$$

Set join (not all case using natural join)

- To prove equality (1), we need to show that the following statements are logically equivalent

$$T(c,p)) \wedge \neg E(s,c)$$

$$(s \in \mathcal{S} \wedge T(c,p)) \wedge \neg(E(s,c) \wedge T(c,p))$$

Since, clearly, $s \in \mathcal{S}$ is true, we need to show that the following statements are equivalent:[5]

$$T(c,p) \wedge \neg E(s,c)$$

$$T(c,p) \wedge \neg(E(s,c) \wedge T(c,p))$$

[5] We assume that there is at least one student.

Set join (not all case using natural join)

We need to show that the following statements are equivalent:

$$T(c,p) \wedge \neg E(s,c) \qquad \text{(I)}$$
$$T(c,p) \wedge \neg(E(s,c) \wedge T(c,p)) \qquad \text{(II)}$$

To see this, you need to determine if statements (I) and (II) have the same truth values. This is the case as shown in the following truth table:

$E(s,c)$	$T(c,p)$	Statement (I)	Statement (II)
true	true	false	false
true	false	false	false
false	true	true	true
false	false	false	false

Set joins (not-empty cases)

- We have now developed the RA expressions for the following set joins:

$$\begin{array}{lcl} \text{some} & = & \pi_{sid,pid}(E \bowtie T) \\ \text{not only} & = & \pi_{sid,pid}((E \times \mathcal{P}) - (E \bowtie T)) \\ \text{not all} & = & \pi_{sid,pid}((\mathcal{S} \times T) - (E \bowtie T)) \end{array}$$

- Notice that the time and space complexities for *not only* and *not all* are at least $O(|E| * |\mathcal{P}|)$ and $O(|T| * |\mathcal{S}|)$, respectively. (At IU, $160000 * 2000$ and $6000 * 40000$.)
- These set joins can be very expensive to evaluate.
- Observe that the time and space complexity for $E \bowtie S$ is never more than $O(|E| + |S| + |\mathcal{S}| * |\mathcal{P}|)$[6]

[6] If we use the hash join algorithm.

Set joins (generalized intersection and difference)

A more compact way to remember these expressions is by using the following notations:

$$
\begin{array}{lcl}
E \sqcap T & \leftrightarrow & E \bowtie T \\
E \ominus T & \leftrightarrow & (E \times \mathcal{P}) - (E \bowtie T) \\
T \ominus E & \leftrightarrow & (\mathcal{S} \times T) - (E \bowtie T)
\end{array}
$$

With these notations, we get

$$
\begin{array}{lcl}
\text{some} & = & \pi_{sid,pid}(E \sqcap T) \\
\text{not only} & = & \pi_{sid,pid}(E \ominus T) \\
\text{not all} & = & \pi_{sid,pid}(T \ominus E)
\end{array}
$$

We call $\sqcap$ the generalized intersection and $\ominus$ the generalized difference.

Set joins (empty cases)

- Finding RA expressions for the empty cases is now easy: they consist of subtracting the corresponding not-empty cases from the set of all possible student-professor pairs (s, p), i.e., the set $\mathcal{S} \times \mathcal{P}$.

- So, we get the following RA expressions for the set joins associated with no, not only, and all:

$$\begin{aligned} \text{no} &= \mathcal{S} \times \mathcal{P} - (\pi_{sid,pid}(E \sqcap T)) \\ \text{only} &= \mathcal{S} \times \mathcal{P} - (\pi_{sid,pid}(E \ominus T)) \\ \text{all} &= \mathcal{S} \times \mathcal{P} - (\pi_{sid,pid}(T \ominus E)) \end{aligned}$$

Set joins (Summary)

The following are then the RA expressions for all of the cases considered:

$$
\begin{array}{lcl}
\text{some} & = & \pi_{sid,pid}(E \sqcap T) \\
\text{not only} & = & \pi_{sid,pid}(E \ominus T) \\
\text{not all} & = & \pi_{sid,pid}(T \ominus E) \\
\text{no} & = & \mathcal{S} \times \mathcal{P} - (\pi_{sid,pid}(E \sqcap T)) \\
\text{only} & = & \mathcal{S} \times \mathcal{P} - (\pi_{sid,pid}(E \ominus T)) \\
\text{all} & = & \mathcal{S} \times \mathcal{P} - (\pi_{sid,pid}(T \ominus E))
\end{array}
$$

Set joins some and no in SQL

- some

```
SELECT   DISTINCT sid, pid
FROM     Enroll NATURAL JOIN TaughtBy
```

- no

```
SELECT   sid, pid
FROM     Student CROSS JOIN Professor
EXCEPT
SELECT   DISTINCT sid, pid
FROM     Enroll NATURAL JOIN TaughtBy
```

Set joins not only and only in SQL

not only

```
SELECT  sid, pid
FROM    (SELECT sid, cno, pid
         FROM Enroll CROSS JOIN Professor
         EXCEPT
         SELECT sid, cno, pid
         FROM Enroll NATURAL JOIN TaughtBy) q
```

only

```
SELECT  sid, pid
FROM    Student     CROSS JOIN Professor
EXCEPT
SELECT  sid, pid
FROM    (SELECT     sid, cno, pid
         FROM       Enroll CROSS JOIN Professor
         EXCEPT
         SELECT     sid, cno, pid
         FROM       Enroll NATURAL JOIN TaughtBy) q
```

Set joins not all and all in SQL

not all

```
SELECT  sid, pid
FROM    (SELECT sid, cno, pid
         FROM Student CROSS JOIN TaughtBy
         EXCEPT
         SELECT sid, cno, pid
         FROM Enroll NATURAL JOIN TaughtBy) q
```

all

```
SELECT  sid, pid
FROM    Student     CROSS JOIN Professor
EXCEPT
SELECT  sid, pid
FROM    (SELECT     sid, cno, pid
         FROM       Student CROSS JOIN TaughtBy
         EXCEPT
         SELECT     sid, cno, pid
         FROM       Enroll NATURAL JOIN TaughtBy) q
```

The queries run in $O(|Student| * |TaughtBy| + |Enroll| * |TaughtBy|)$

Set joins all in Pure SQL

```
SELECT   sid, pid
FROM     Student      CROSS JOIN Professor
EXCEPT
SELECT   sid, pid
FROM     (SELECT      sid, cno, pid
          FROM        Student CROSS JOIN TaughtBy
          EXCEPT
          SELECT      sid, cno, pid
          FROM        Enroll NATURAL JOIN TaughtBy) q
```

Contrast this with the all set join expressed in Pure SQL

```
SELECT   sid, pid
FROM     Student s, Professor p
WHERE    NOT EXISTS (SELECT 1
                     FROM TaughtBy t
                     WHERE t.pid = p.pid AND
                  t.cno NOT IN (SELECT e.cno
                                FROM Enroll e
                                WHERE e.sid = s.sid))
```

This query runs in $O(|Student| * |Professor| * |TaughtBy| * |Enroll|)$ and is thus orders of magnitude slower than the above RA SQL query which which runs in $O(|Student| * |TaughtBy| + |Enroll| * |TaughtBy|)$.

Set joins (Example)

Consider the following queries:

"Find the student sid pairs (s_1, s_2) such that student s_1 takes [*some*|*not only*|*not all*|*no*|*only*|*all*] course(s) that student s_2 takes."

The queries are very similar to those we have been considering.

The only difference is that we now have to involve the *Enroll* relation twice: ones for student s_1 and the other for student s_2.

Set joins (Example)

Let us denote by $\widetilde{E}$ the RA expression $\rho_{sid \to \widetilde{sid}}(\pi_{cno,sid}(E))$. So $\widetilde{E}$ is the same as E, except that its schema is $(cno, \widetilde{sid})$ instead of (sid, cno). We have renamed sid by $\widetilde{sid}$.

For this example, the search space is the set of all student pairs $\mathcal{S} \times \mathcal{S}$ and we assume its schema is $(sid, \widetilde{sid})$.

Then the RA expressions for the above six queries are

$$
\begin{array}{lcl}
\text{some} & = & \pi_{sid,\widetilde{sid}}(E \bowtie \widetilde{E}) \\
\text{not only} & = & \pi_{sid,\widetilde{sid}}(E \ominus \widetilde{E}) \\
\text{not all} & = & \pi_{sid,\widetilde{sid}}(\widetilde{E} \ominus E) \\
\text{no} & = & \mathcal{S} \times \mathcal{S} - (\pi_{sid,\widetilde{sid}}(E \bowtie \widetilde{E})) \\
\text{only} & = & \mathcal{S} \times \mathcal{S} - (\pi_{sid,\widetilde{sid}}(E \ominus \widetilde{E})) \\
\text{all} & = & \mathcal{S} \times \mathcal{S} - (\pi_{sid,\widetilde{sid}}(\widetilde{E} \ominus E))
\end{array}
$$

More set joins

- The RA region expressions $E \sqcap T$, $E \ominus T$, and $T \ominus E$ permit us to express many more set joins. We will consider one such example next.
- "Find the student-professor pairs (s, p) such that student s takes at least 2 courses not taught by professor p."
- More formally, we need to find a RA expression for the query

$$\{(s,p) \mid \exists c_1, c_2 : c_1 \neq c_2 \wedge c_1 \in (E(s) - T(p)) \wedge c_2 \in (E(s) - T(p))\}$$

- Alternatively,

$$\{(s,p) \mid \exists c_1, c_2 : c_1 \neq c_2 \wedge \{c_1, c_2\} \subseteq (E(s) - T(p))\}$$

More set joins

- "Find the student-professor pairs (s, p) such that student s takes at least 2 courses not taught by professor p."

$$\{(s,p) \mid \exists c_1, c_2 : c_1 \neq c_2 \wedge \{c_1, c_2\} \subseteq (E(s) - T(p))\}$$

- From this formulation, it is clear that we need to involve the region expression $E \ominus T$, but we need two copies of it, one to reason about c_1 and the other to reason about c_2.
- Let these two copies be $E_1 \ominus T_1$ with schema (sid, cno_1, pid) and $E_2 \ominus T_2$ with schema (sid, cno_2, pid).
- Then the RA expression for the query is as follows:

$$\pi_{sid,pid}(\sigma_{cno_1 \neq cno_2}((E_1 \ominus T_1) \bowtie (E_2 \ominus T_2)))$$

Set semijoins (Motivation)

- Consider the relations Enroll(sid,cno) and the relation of CS courses CS(cno).
- We are interested in students satisfying some property.
- "Find each student sid *s* such that student *s* takes [*some*|*not only*|*not all*|*no*|*only*|*all*] CS course(s)."

Set semijoins (Solution)

An analysis very similar to that for set joins gives the following RA queries for these queries.
First observe that $E \bowtie CS = E \ltimes CS$.

$$
\begin{array}{lcll}
\text{some} & = & \pi_{sid}(E \bowtie CS) & = \quad \pi_{sid}(E \ltimes CS) \\
\text{not only} & = & \pi_{sid}(E \ominus CS) & \\
\text{not all} & = & \pi_{sid}(CS \ominus E) & \\
\text{no} & = & \mathcal{S} - (\pi_{sid}(E \ltimes CS)) & \\
\text{only} & = & \mathcal{S} - (\pi_{sid}(E \ominus CS)) & \\
\text{all} & = & \mathcal{S} - (\pi_{sid}(CS \ominus E)) &
\end{array}
$$

Where
$$
\begin{array}{lcll}
E \ominus CS & = & E - (E \ltimes CS) & = \quad E \,\overline{\ltimes}\, CS \\
CS \ominus E & = & (\mathcal{S} \times CS) - (E \ltimes CS) &
\end{array}
$$

Set semijoins (Efficiency consideration)

Reconsider the 6 semijoins:

$$
\begin{array}{lcl}
\text{some} & = & \pi_{sid}(E \ltimes CS) \\
\text{not only} & = & \pi_{sid}(E \,\overline{\ltimes}\, CS)) \\
\text{not all} & = & \pi_{sid}((\pi_{sid}(S) \times CS) - (E \ltimes CS)) \\
\text{no} & = & \pi_{sid}(S) - (\pi_{sid}(E \ltimes CS)) \\
\text{only} & = & \pi_{sid}(S) - \pi_{sid}(E \,\overline{\ltimes}\, CS) \\
\text{all} & = & \pi_{sid}(S) - \pi_{sid}((\pi_{sid}(S) \times CS) - (E \ltimes CS))
\end{array}
$$

Let $k = |S|$, $l = |CS|$, and $n = |E|$.

- Then, using hashing methods, the some, no, not only, only set semijoins can be implemented in linear time and space $O(|k| + |l| + |n|)$. (I.e., extremely fast.)
- However, the not all and all set semijoins require quadratic time and space $O((k * l) + n)$ because of the need to compute $\pi_{sid}(S) \times CS$. (I.e., can be extremely expensive.)

Set semijoin some and no in SQL

- some

```
SELECT   DISTINCT sid
FROM     Enroll NATURAL JOIN CS
```

- no

```
SELECT   sid
FROM     Student
EXCEPT
SELECT   DISTINCT sid
FROM     Enroll NATURAL JOIN CS
```

Set semijoin not only in SQL

- not only

```
SELECT   DISTINCT sid
FROM     (SELECT sid, cno
          FROM   Enroll
          EXCEPT
          SELECT sid, cno
          FROM   Enroll NATURAL JOIN CS) q
```

Set semijoin only SQL

- only

```
SELECT   sid
FROM     Student
EXCEPT
SELECT   DISTINCT sid
FROM     (SELECT sid, cno
          FROM   Enroll
          EXCEPT
          SELECT sid, cno
          FROM   Enroll NATURAL JOIN CS) q
```

Set semijoin not all in SQL

- not all

```
SELECT   DISTINCT sid
FROM     (SELECT sid, cno
          FROM   (SELECT sid from Student) s CROSS JOIN CS
          EXCEPT
          SELECT sid, cno
          FROM   Enroll NATURAL JOIN CS) q
```

Set semijoin all in SQL

- all

```
SELECT    sid
FROM      Student s
EXCEPT
SELECT    DISTINCT sid
FROM      (SELECT sid, cno
           FROM   (SELECT sid from Student) s CROSS JOIN CS
           EXCEPT
           SELECT sid, cno
           FROM   Enroll NATURAL JOIN CS) q
```

Set joins (General case)

- Let $E_1(A_1, \ldots, A_m, C_1, \ldots, C_k)$ and $E_2(C_1, \ldots, C_k, B_1, \ldots, B_n)$ be RA expressions.
- We assume that each attribute has an associated domain: these are denoted $\mathcal{A}_1, \ldots, \mathcal{A}_m, \mathcal{C}_1, \ldots, \mathcal{C}_k, \mathcal{B}_1, \ldots, \mathcal{B}_n$.
- We can then build RA expressions $E_1 \sqcap E_2$, $E_1 \ominus E_2$, and $E_2 \ominus E_1$ as follows:

$$\begin{aligned} E_1 \sqcap E_2 &= E_1 \bowtie E_2 \\ E_1 \ominus E_2 &= (E_1 \times \mathcal{B}_1 \times \cdots \times \mathcal{B}_n) - (E_1 \bowtie E_2) \\ E_2 \ominus E_1 &= (\mathcal{A}_1 \times \cdots \times \mathcal{A}_m \times E_2) - (E_1 \bowtie E_2) \end{aligned}$$

- Using these region expressions we can build numerous other set-joins.

Set joins (From general case to special cases)

$$
\begin{array}{lcl}
E_1 \sqcap E_2 & = & E_1 \bowtie E_2 \\
E_1 \ominus E_2 & = & (E_1 \times \mathcal{B}_1 \times \cdots \times \mathcal{B}_n) - (E_1 \bowtie E_2) \\
E_2 \ominus E_1 & = & (\mathcal{A}_1 \times \cdots \times \mathcal{A}_m \times E_2) - (E_1 \bowtie E_2)
\end{array}
$$

We can now consider special cases:

- When $m = k = n = 1$ we have the region expressions for the set joins we have seen above.
- When $m = k = 1$ and $n = 0$ we have the region expressions for semi set joins we have seen above.
- When $m = n = 0$, then

$$
\begin{array}{lcl}
E_1 \sqcap E_2 & = & E_1 \cap E_2 \\
E_1 \ominus E_2 & = & E_1 - E_2 \\
E_2 \ominus E_1 & = & E_2 - E_1
\end{array}
$$