Updated 1/31/2023

Topic 1: Data models and query languages
Unit 3: Relational Algebra (RA)
Lecture 7

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp23)

https://northeastern-datalab.github.io/cs7240/sp23/

1/31/2023

Pre-class conversations

- Last class summary
- Please keep on pointing out any errors on the slides
- It is time to start to hand in your first scribes (some ideas today)
- Project discussions (in 2 weeks: Fri 2/17: project ideas)

- today:
  - Algebra: independence and Codd's theorem
- next time:
  - Recursion (Datalog)

Algebra and the connection to logic and queries

- Algebra
- Relational Algebra
  - Operators
  - Independence
  - Power of algebra: optimizations
- Equivalence RA and safe RC (Codd's theorem)
  - RA → RC
  - RC → RA

Write in RA

Employee(<u>id</u>, name, salary)

Q: *Find the ID and name of those employees who earn more than the employee whose ID is 123?*

?

Write in RA

Employee(<u>id</u>, name, salary)

Q: *Find the ID and name of those employees who earn more than the employee whose ID is 123?*

$$\pi_{\text{e.id,e.name}}\left(\sigma_{\text{e.salary>o.salary}}(\rho_{\text{e}}(\text{employee})\times\sigma_{\text{id=123}}(\rho_{\text{o}}(\text{employee})))\right)$$

$$\pi_{\text{id,name}}\left(\sigma_{\text{salary>s}}(\text{employee} \times (\rho_{\text{salary}\to\text{s}}(\pi_{\text{salary}}\ (\sigma_{\text{id=123}}(\text{employee})))))\right)$$

$$\pi_{\$1,\$2}(\sigma_{\$4=123\ \wedge\ \$3>\$6}(\text{employee} \times \text{employee}))$$

Relational Algebra (RA) operators

- Five basic operators:
  1. Selection: σ ("sigma")
  2. Projection: Π
  3. Cartesian Product: ×
  4. Union: ∪
  5. Difference: –
- Auxiliary (or special) operator
  6. Renaming: ρ ("rho")
- Derived (or implied) operators
  7. Joins ⋈ (natural, theta join, equi-join, [semi-join: moved to T3-U1])
  8. Intersection / complement
  9. Division

Derived relational operators:
- can be expressed in basic RA; thus not needed

But enhancing the basic operator set with derived operators is a good idea:
- Queries become easier to write/understand/maintain
- Easier for DBMS to apply specialized optimizations (recall the conceptual evaluation strategy)

we discuss later in class in detail (SJs are at the heart of efficient algorithms)

most important

7a. Natural Join (⋈)

Product(<u>pname</u>, price, category, cid)
Company(<u>cid</u>, cname, stockprice, country)

316

- Notation: R ⋈ S
- Joins R and S on equality of all shared attributes
  - Only makes sense in named perspective!
  - If R has attribute set A, and S has attribute set B, and they share attributes A⋂B = C, can also be written as R $\bowtie_C$ S
- Natural join in basic RA:
  - Meaning: $R \bowtie S = \pi_{A \cup B}(\sigma_{R.C=S.C}(R \times S))$
  - Meaning: $R \bowtie S = \pi_{A \cup B}(\sigma_{C=D}(\rho_{C \to D}(R) \times S))$
    - The rename $\rho_{C \to D}$ renames the shared attributes in one of the relations
    - The selection $\sigma_{C=D}$ checks equality of the shared attributes
    - The projection $\pi_{A \cup B}$ eliminates the duplicate common attributes

SQL

```
SELECT pname, price, category,
P.cid, cname, stockprice, country
FROM Product P, Company C
WHERE P.cid= C.cid
```

SQL (alternative syntax)

?

7a. Natural Join (⋈)

Product(<u>pname</u>, price, category, cid)
Company(<u>cid</u>, cname, stockprice, country)

316

- Notation: R ⋈ S
- Joins R and S on equality of all shared attributes
  - Only makes sense in named perspective!
  - If R has attribute set A, and S has attribute set B, and they share attributes A⋂B = C, can also be written as R $\bowtie_C$ S

- Natural join in basic RA:
  - Meaning: $R \bowtie S = \pi_{A \cup B}(\sigma_{R.C=S.C}(R \times S))$
  - Meaning: $R \bowtie S = \pi_{A \cup B}(\sigma_{C=D}(\rho_{C \to D}(R) \times S))$
    - The rename $\rho_{C \to D}$ renames the shared attributes in one of the relations
    - The selection $\sigma_{C=D}$ checks equality of the shared attributes
    - The projection $\pi_{A \cup B}$ eliminates the duplicate common attributes

SQL

```
SELECT pname, price, category,
P.cid, cname, stockprice, country
FROM Product P, Company C
WHERE P.cid= C.cid
```

SQL (alternative syntax)

```
SELECT *
FROM Product
NATURAL JOIN Company
```

RA:

?

7a. Natural Join (⋈)

Product(pname, price, category, cid)
Company(cid, cname, stockprice, country)

- Notation: R ⋈ S
- Joins R and S on equality of all shared attributes
  - Only makes sense in named perspective!
  - If R has attribute set A, and S has attribute set B, and they share attributes $A \cap B = C$, can also be written as $R \bowtie_C S$
- Natural join in basic RA:
  - Meaning: $R \bowtie S = \pi_{A \cup B}(\sigma_{R.C=S.C}(R \times S))$
  - Meaning: $R \bowtie S = \pi_{A \cup B}(\sigma_{C=D}(\rho_{C \to D}(R) \times S))$
    - The rename $\rho_{C \to D}$ renames the shared attributes in one of the relations
    - The selection $\sigma_{C=D}$ checks equality of the shared attributes
    - The projection $\pi_{A \cup B}$ eliminates the duplicate common attributes

SQL

```
SELECT pname, price, category,
P.cid, cname, stockprice, country
FROM Product P, Company C
WHERE P.cid= C.cid
```

SQL (alternative syntax)

```
SELECT *
FROM Product
NATURAL JOIN Company
```

RA:

$$\text{Product} \bowtie \text{Company}$$

7a. Natural Join (⋈): an alternative perspective

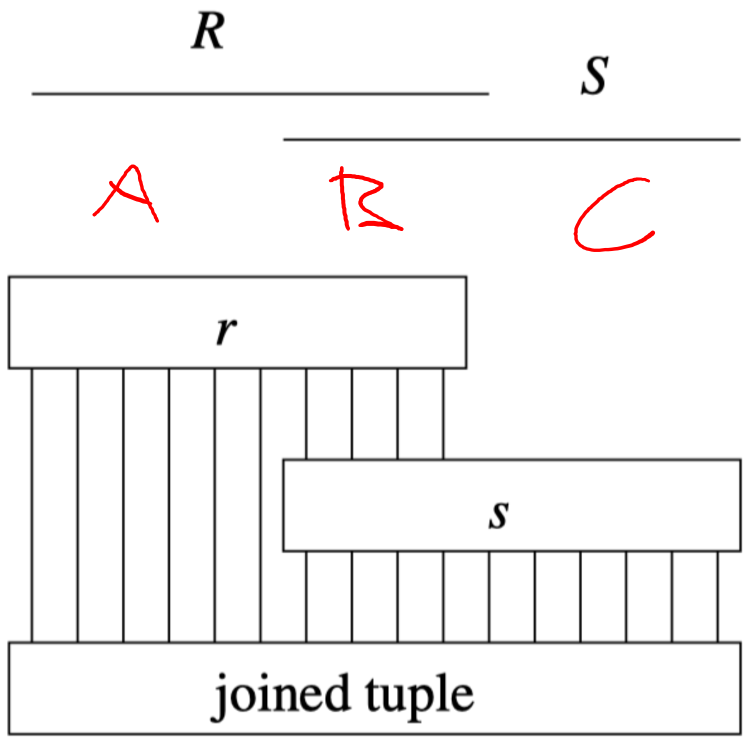


Figure 15: Joining tuples

We only want to pair those tuples that match in some way.

More formally the semantics of the natural join are defined as follows:

$$R \bowtie S = \{ r \cup s \mid r \in R \ \wedge \ s \in S \ \wedge \ \mathit{Fun}(r \cup s) \} \quad \textbf{(1)}$$

where *Fun(t)* is a predicate that is true for a relation *t* (in the mathematical sense) iff *t* is a function. It is usually required that *R* and *S* must have at least one common attribute, but if this constraint is omitted, and *R* and *S* have no common attributes, then the natural join becomes exactly the Cartesian product.

Source of Figure: Garcia-Molina, Ullman, Widom. Database Systems -- The Complete Book (2nd ed, international ed), 2014. http://infolab.stanford.edu/~ullman/dscb.html
Source of text: https://en.wikipedia.org/wiki/Relational_algebra#Natural_join

7a. Natural Join (⋈): An example

R

A	B
1	2
3	4

S

B	C	D
2	5	6
4	7	8
9	10	11

$$\rho_{B \rightarrow E}(\mathrm{R}) \times \mathrm{S}$$

?

7a. Natural Join (⋈): An example

R

A	B
1	2
3	4

S

B	C	D
2	5	6
4	7	8
9	10	11

R ⋈ S

?

$\rho_{B \to E}(\mathrm{R}) \times \mathrm{S}$

A	E	B	C	D
1	2	2	5	6
1	2	4	7	8
1	2	9	10	11
3	4	2	5	6
3	4	4	7	8
3	4	9	10	11

7a. Natural Join (⋈): An example

R

A	B
1	2
3	4

S

B	C	D
2	5	6
4	7	8
9	10	11

R ⋈ S

A	B	C	D
1	2	5	6
3	4	7	8

R ⋈ S =

in basic RA

?

$\rho_{B \rightarrow E}(R) \times S$

A	E	B	C	D
1	2	2	5	6
~~1~~	~~2~~	~~4~~	~~7~~	~~8~~
~~1~~	~~2~~	~~9~~	~~10~~	~~11~~
~~3~~	~~4~~	~~2~~	~~5~~	~~6~~
3	4	4	7	8
~~3~~	~~4~~	~~9~~	~~10~~	~~11~~

7a. Natural Join (⋈): An example

R

A	B
1	2
3	4

S

B	C	D
2	5	6
4	7	8
9	10	11

$R \bowtie S$

A	B	C	D
1	2	5	6
3	4	7	8

$R \bowtie S =$

$\Pi_{AR.BCD}(\sigma_{R.B=S.B}(R \times S)) =$

$\Pi_{ABCD}(\sigma_{B=E}(\rho_{B\to E}(R) \times S))$

$\rho_{B\to E}(R) \times S$

A	E	B	C	D
1	2	2	5	6
~~1~~	~~2~~	~~4~~	~~7~~	~~8~~
~~1~~	~~2~~	~~9~~	~~10~~	~~11~~
~~3~~	~~4~~	~~2~~	~~5~~	~~6~~
3	4	4	7	8
~~3~~	~~4~~	~~9~~	~~10~~	~~11~~

7a. Natural Join (⋈): practice

- Given schemas R(A, B, C, D), S(A, C, E), what is the schema of R ⋈ S ?

?

7a. Natural Join ($\bowtie$): practice

- Given schemas R(A, B, C, D), S(~~A~~, ~~C~~, E), what is the schema of R $\bowtie$ S ?

Answer(A, B, C, D,E)

- Given R(A, B, C), S(D, E), what is R $\bowtie$ S ?

?

7a. Natural Join (⋈): practice

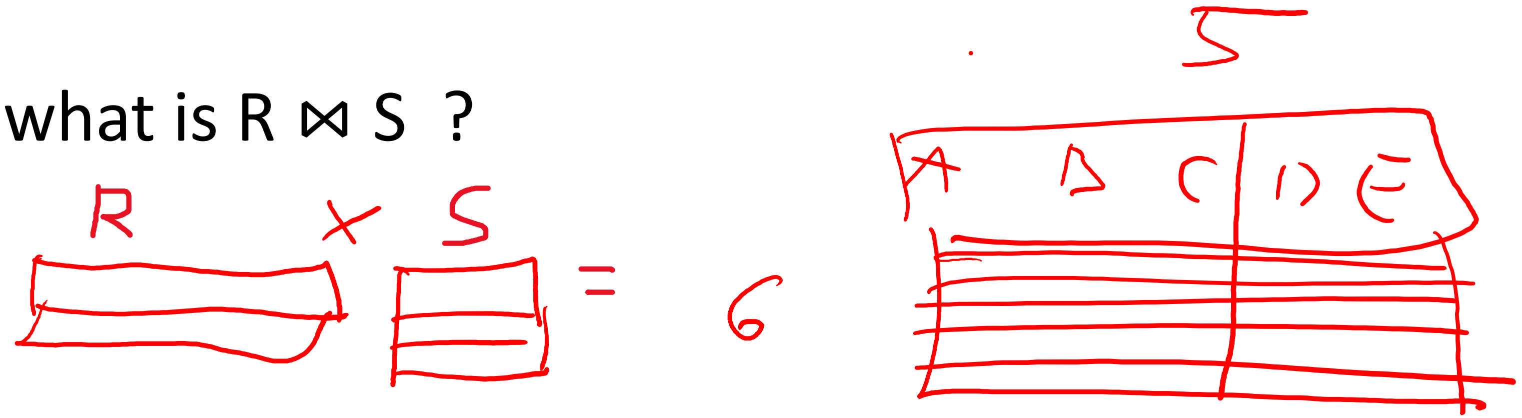
- Given schemas R(A, B, C, D), S(A, C, E), what is the schema of R ⋈ S ?

Answer(A, B, C, D,E)

no condition in the selection
that could be violated:

- Given R(A, B, C), S(D, E), what is R ⋈ S ?

R × S



- Given R(A, B), S(A, B), what is R ⋈ S ?

?

7a. Natural Join (⋈): practice

- Given schemas R(A, B, C, D), S(A, C, E), what is the schema of R ⋈ S ?

  Answer(A, B, C, D,E)

- Given R(A, B, C), S(D, E), what is R ⋈ S ?

  $R \times S$

- Given R(A, B), S(~~A~~, ~~B~~), what is R ⋈ S ?

  $R \cap S$

R ⋈ S

R:

A	B
1	2
1	3
2	2

S:

A	B
1	2
3	1
2	2

= ?

7a. Natural Join (⋈): practice

- Given schemas R(A, B, C, D), S(A, C, E), what is the schema of R ⋈ S ?

  Answer(A, B, C, D,E)

- Given R(A, B, C), S(D, E), what is R ⋈ S ?

  $R \times S$

- Given R(A, B), S(A, B), what is R ⋈ S ?

  $R \cap S$

R ⋈ S

R

A	B
1	2
1	3
2	2

S

A	B
1	2
3	1
2	2

=

Ans

A	B
1	2
2	2

7b. Theta Join ($\bowtie_\theta$)

- A join that involves a predicate

$$\boxed{R_1 \bowtie_\theta R_2 = \sigma_\theta(R_1 \times R_2)}$$

- θ ("theta") can be any condition
- No projection: #attributes in output = sum #attributes in input

Note that natural join is a theta join + a selection

- Example: band-joins for approx. matchings across tables

AnonPatient (age, zip, disease)
Voters (name, age, zip)

Assume relatively fresh data (within 1 year)

?

7b. Theta Join ($\bowtie_\theta$)

- A join that involves a predicate

$$R_1 \bowtie_\theta R_2 = \sigma_\theta(R_1 \times R_2)$$

- θ ("theta") can be any condition
- No projection: #attributes in output = sum #attributes in input
- Example: band-joins for approx. matchings across tables

Note that natural join is a theta join + a selection

AnonPatient (age, zip, disease)
Voters (name, age, zip)

Assume relatively fresh data (within 1 year)

$$A \bowtie_{P.zip=V.zip \wedge (P.age>=V.age -1 \wedge P.age<=V.age +1)} V$$

Student(sid,name,gpa)
People(ssn,name,address)

SQL:

```
SELECT *
FROM
    Students, People
WHERE θ
```

RA:

?

7b. Theta Join ($\bowtie_\theta$)

Student(sid,name,gpa)
People(ssn,name,address)

- A join that involves a predicate

$$\boxed{R_1 \bowtie_\theta R_2 = \sigma_\theta(R_1 \times R_2)}$$

- θ ("theta") can be any condition
- No projection: #attributes in output = sum #attributes in input

Note that natural join is a theta join + a selection

- Example: band-joins for approx. matchings across tables

AnonPatient (age, zip, disease)
Voters (name, age, zip)

Assume relatively fresh data (within 1 year)

$$A \bowtie_{P.zip=V.zip \land P.age>=V.age -1 \land P.age<=V.age +1} V$$

SQL:

```
SELECT *
FROM
   Students, People
WHERE θ
```

RA:

$$\text{Students} \bowtie_\theta \text{People}$$

7c. Equi-join ($\bowtie_{A=B}$)

- A theta join where q is an equality

$$R_1 \bowtie_{A=B} R_2 \;=\; \sigma_{A=B}(R_1 \times R_2)$$

- Example over Gizmo DB:
  - Product $\bowtie_{\text{manufacturer=cname}}$ Company
- Most common join in practice!

Student(sid,sname,gpa)
People(ssn,pname,address)

SQL:

```
SELECT *
FROM
   Students S, People P
WHERE sname = pname
```

RA:

?

7c. Equi-join ($\bowtie_{A=B}$)

- A theta join where q is an equality

$$\boxed{R_1 \bowtie_{A=B} R_2 = \sigma_{A=B}(R_1 \times R_2)}$$

- Example over Gizmo DB:
  - Product $\bowtie_{\text{manufacturer=cname}}$ Company
- Most common join in practice!

Student(sid,sname,gpa)
People(ssn,pname,address)

SQL:

```
SELECT *
FROM
    Students S, People P
WHERE sname = pname
```

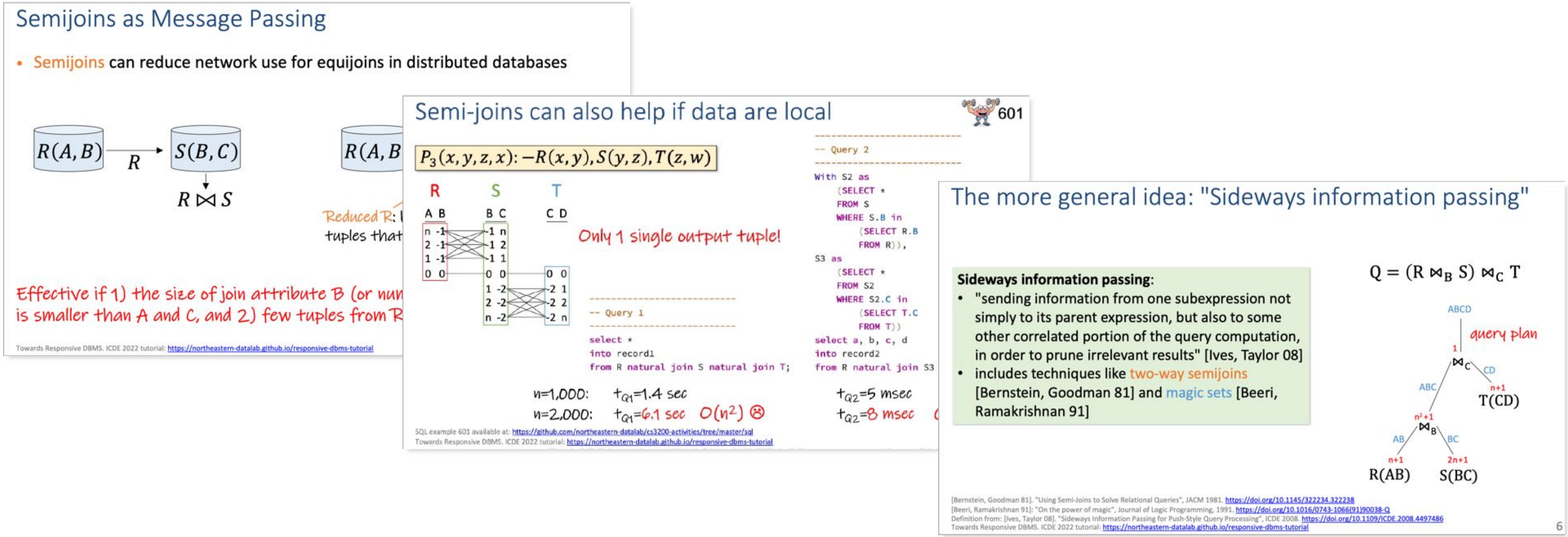
RA:

$$S \bowtie_{\text{sname=pname}} P$$

What is the connection with a natural join? ?

7d. Semi-join (⋉) [moved to T3-U1]

- $R \ltimes S$: Return tuples from R for which there is a matching tuple in S that is equal on their common attribute names.



See "Part 3: Acyclic queries & Enumeration": https://northeastern-datalab.github.io/responsive-dbms-tutorial/slides/Responsive-DBMS-tutorial-part-3-AcyclicQueries-Enumeration.pdf, https://www.youtube.com/watch?list=PL_72ERGKF6DTInW_P3a9zTYPSNLwbqOAx&v=toi7ysuyRkw from ICDE'22 tutorial "Toward Responsive DBMS: Optimal Join Algorithms, Enumeration, Factorization, Ranking, and Dynamic Programming" by Tziavelis et al. https://doi.org/10.1109/ICDE53745.2022.00299, https://northeastern-datalab.github.io/responsive-dbms-tutorial/

Join Summary

- **Theta-join**: $R \bowtie_\theta S = \sigma_\theta (R \times S)$
  - Join of R and S with a join condition θ
  - Cross-product followed by selection θ
  - No projection
- **Equijoin**: $R \bowtie_\theta S = \sigma_\theta (R \times S)$
  - Join condition θ consists only of equalities
  - No projection
- **Natural join**: $R \bowtie S = \pi_A (\sigma_\theta (R \times S))$
  - Equality on **all** fields with same name in R and in S
  - Projection π_A drops all redundant attributes

Example: Converting SFW Query to RA

```
Student(sid,name,gpa)
People(ssn,name,address)
```

```sql
SELECT DISTINCT gpa, address
FROM Student S, People P
WHERE S.name = P.name
AND gpa > 3.5
```

How do we represent this query in RA?

?

Example: Converting SFW Query to RA

Student(sid,name,gpa)
People(ssn,name,address)

```
SELECT DISTINCT gpa, address
FROM Student S, People P
WHERE S.name = P.name
AND gpa > 3.5
```

How do we represent this query in RA?

$$\Pi_{\text{gpa,address}}(\sigma_{\text{gpa}>3.5}(S \bowtie P))$$

$$\Pi_{\text{gpa,address}}(\sigma_{\text{gpa}>3.5 \land \text{S.name}=\text{P.name}}(S \times P))$$

$$\Pi_{\text{gpa,address}}(\sigma_{\text{gpa}>3.5 \land \text{name}=\text{name2}}(S \times \rho_{\text{name}\rightarrow\text{name2}} P))$$

Some Examples

Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supply(sno,pno,qty,price)

Find names of suppliers of parts with size greater than 10

?

Find names of suppliers of red parts or parts with size greater than 10

?

Some Examples

Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supply(sno,pno,qty,price)

Find names of suppliers of parts with size greater than 10

$\pi_{sname}(\sigma_{psize>10}(Supplier \bowtie Supply \bowtie Part))$

$\pi_{sname}(Supplier \bowtie Supply \bowtie (\sigma_{psize>10}(Part))$

Find names of suppliers of red parts or parts with size greater than 10

?

Some Examples

Supplier(<u>sno</u>,sname,scity,sstate)
Part(<u>pno</u>,pname,psize,pcolor)
Supply(<u>sno,pno</u>,qty,price)

Find names of suppliers of parts with size greater than 10

$\pi_{sname}(\sigma_{psize>10}(Supplier \bowtie Supply \bowtie Part))$

$\pi_{sname}(Supplier \bowtie Supply \bowtie (\sigma_{psize>10} (Part))$ → Representation of RA as tree?

?

Find names of suppliers of red parts or parts with size greater than 10

$\pi_{sname}(Supplier \bowtie Supply \bowtie (\sigma_{psize>10} (Part) \cup \sigma_{pcolor='red'} (Part)))$

$\pi_{sname}(Supplier \bowtie Supply \bowtie (\sigma_{psize>10 \vee pcolor='red'} (Part)))$

Some Examples

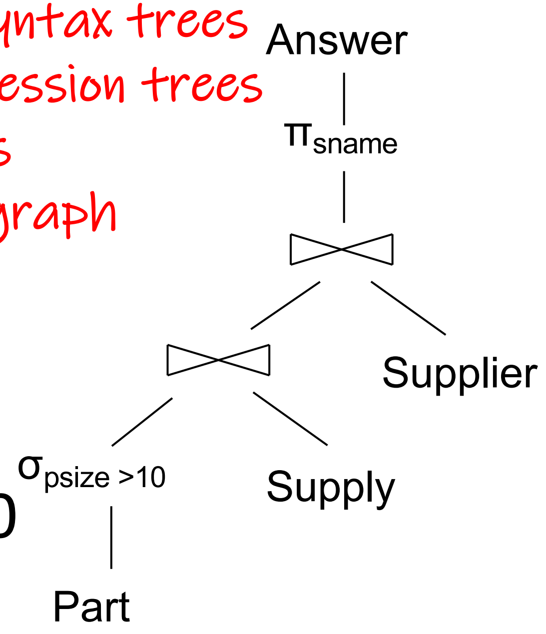
Supplier(<u>sno</u>,sname,scity,sstate)
Part(<u>pno</u>,pname,psize,pcolor)
Supply(<u>sno,pno</u>,qty,price)

Usually unary or binary. Think of:
- abstract syntax trees
- binary expression trees
- parse trees
- data flow graph

Find names of suppliers of parts with size greater than 10

$\pi_{sname}(\sigma_{psize>10}(\text{Supplier} \bowtie (\text{Supply} \bowtie \text{Part})))$

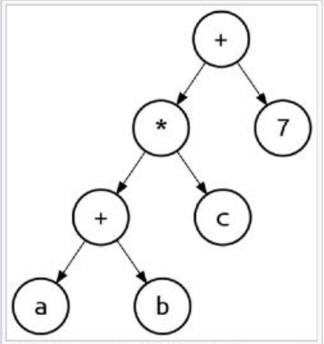
$\pi_{sname}(\text{Supplier} \bowtie \text{Supply} \bowtie (\sigma_{psize>10}(\text{Part}))$ → Representation of RA as tree?



Find names of suppliers of red parts or parts with size greater than 10

$\pi_{sname}(\text{Supplier} \bowtie \text{Supply} \bowtie (\sigma_{psize>10}(\text{Part}) \cup \sigma_{pcolor='red'}(\text{Part})))$

$\pi_{sname}(\text{Supplier} \bowtie \text{Supply} \bowtie (\sigma_{psize>10 \lor pcolor='red'}(\text{Part})))$



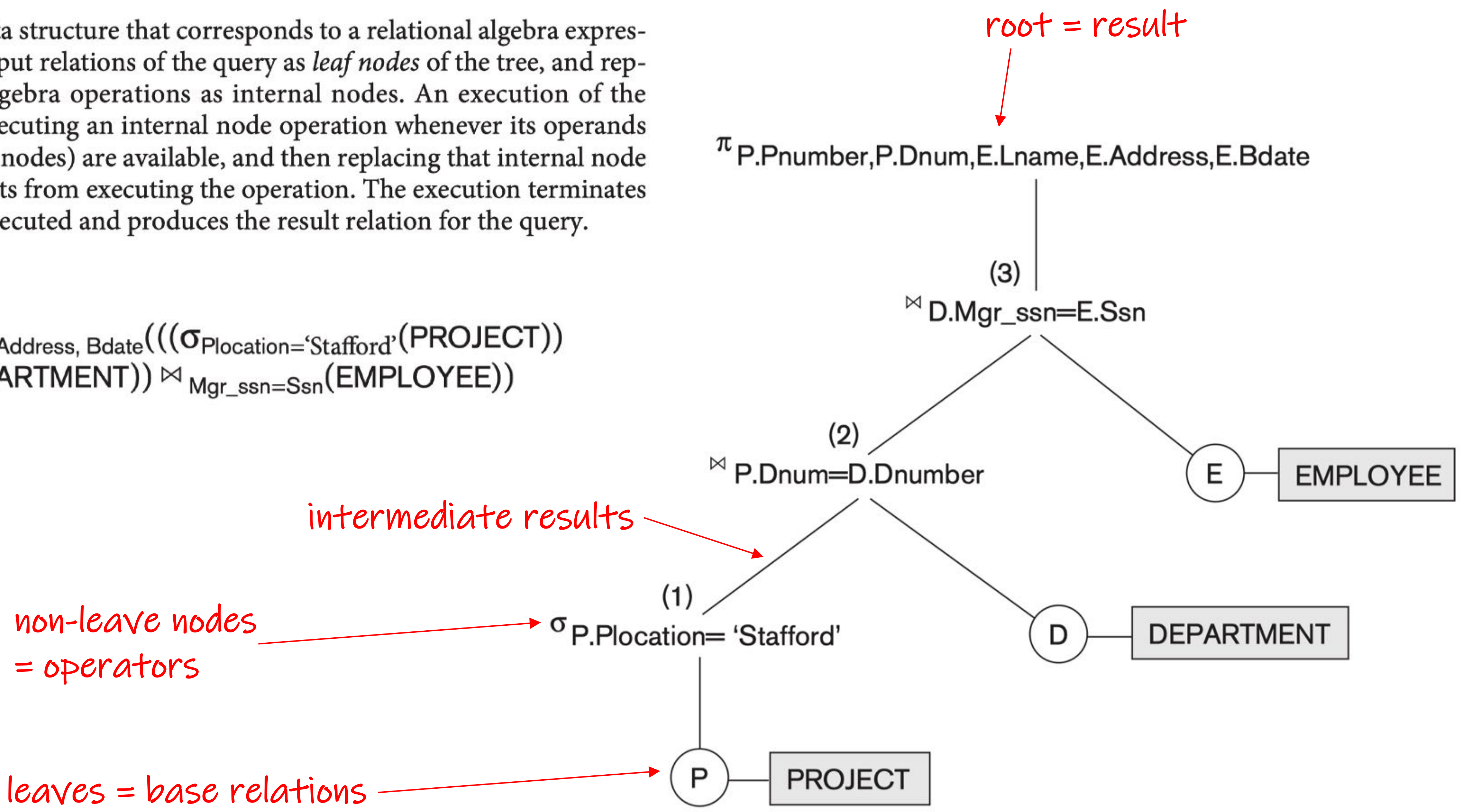
Expression tree of the expression (a+b)*c+7

https://en.wikipedia.org/wiki/Binary_expression_tree

Query (Evaluation / Execution) Tree, Data flow graph

A **query tree** is a tree data structure that corresponds to a relational algebra expression. It represents the input relations of the query as *leaf nodes* of the tree, and represents the relational algebra operations as internal nodes. An execution of the query tree consists of executing an internal node operation whenever its operands (represented by its child nodes) are available, and then replacing that internal node by the relation that results from executing the operation. The execution terminates when the root node is executed and produces the result relation for the query.

$$\pi_{\text{Pnumber, Dnum, Lname, Address, Bdate}}(((\sigma_{\text{Plocation='Stafford'}}(\text{PROJECT})) \bowtie_{\text{Dnum=Dnumber}}(\text{DEPARTMENT})) \bowtie_{\text{Mgr\_ssn=Ssn}}(\text{EMPLOYEE}))$$



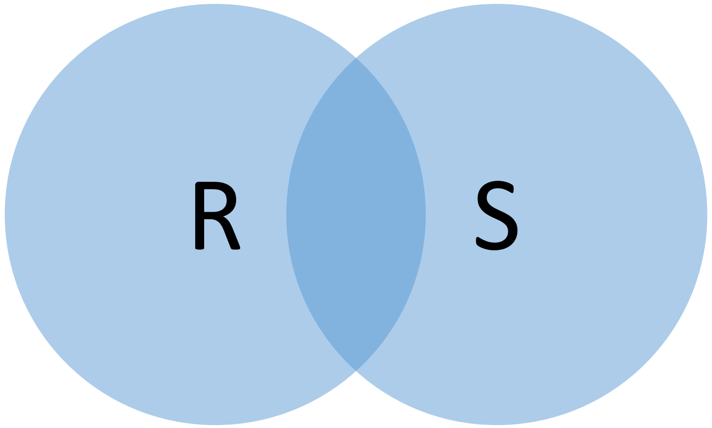
Relational Algebra (RA) operators

- Five basic operators:
  1. Selection: σ ("sigma")
  2. Projection: Π
  3. Cartesian Product: ×
  4. Union: ∪
  5. Difference: –
- Auxiliary (or special) operator
  6. Renaming: ρ ("rho")
- Derived (or implied) operators
  7. Joins ⋈ (natural, theta join, equi-join, [semi-join: moved to T3-U1])
  8. Intersection / complement
  9. Division

8. What about Intersection ∩?

- As derived operator using union and minus

  ?

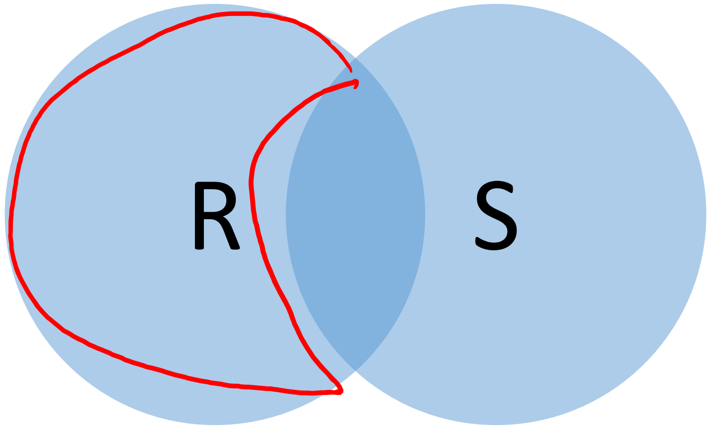


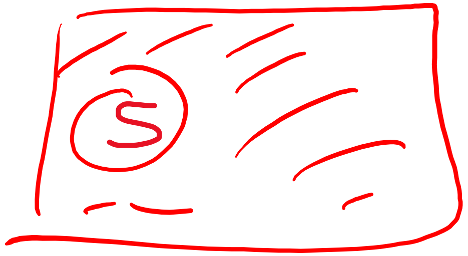
8. What about Intersection ∩?

- As derived operator using union and minus

(R-S)

?





$S^c = D - S$

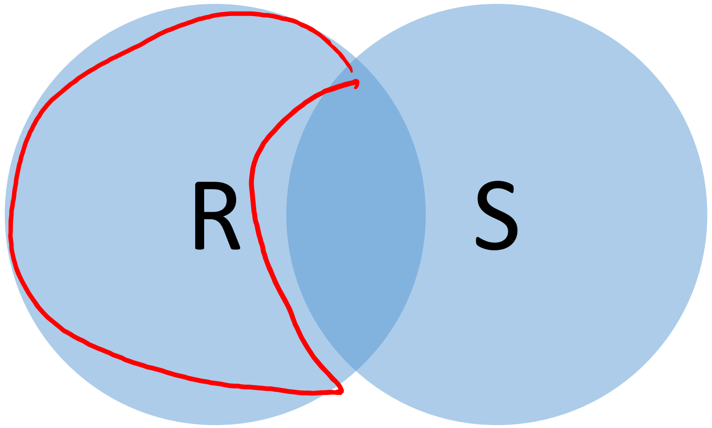
R

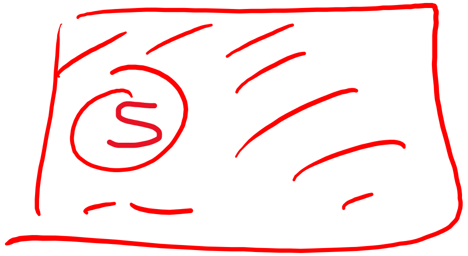
8. What about Intersection ∩?

- As derived operator using union and minus

$(R \cup S) - (R - S) - (S - R)$

?





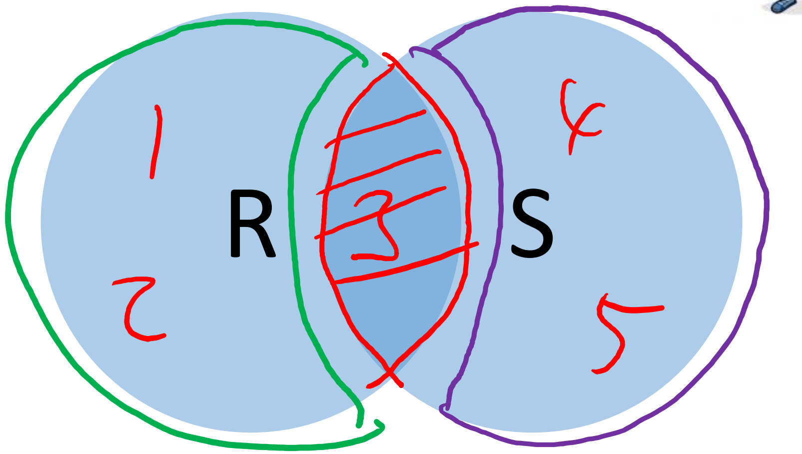
$S^c = \Delta - S$

R

8. What about Intersection ∩?

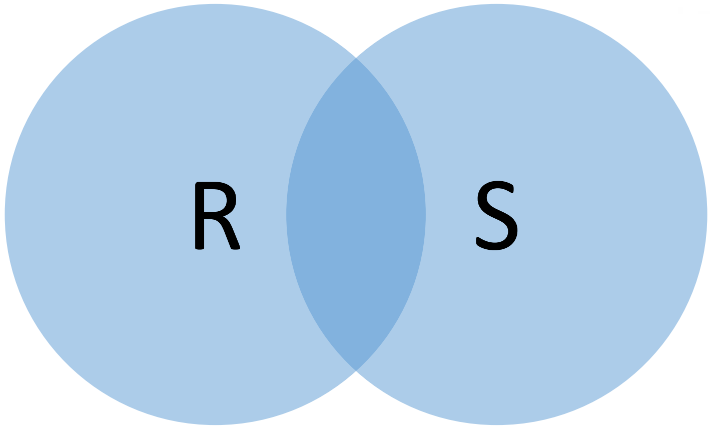
- As derived operator using union and minus

$$(R \cup S) - (\underline{(R - S)} \cup \underline{(S - R)})$$



$$\{1, 2, 3\} \cap \{3, 4, 5\} = 3$$

8. What about Intersection ∩?



- As derived operator using union and minus

$$R \cap S = ((R \cup S) - (R - S)) - (S - R)$$
$$R \cap S = (R \cup S) - ((R - S) \cup (S - R))$$

- Derived operator using minus only!

?

8. What about Intersection ∩?

- As derived operator using union and minus

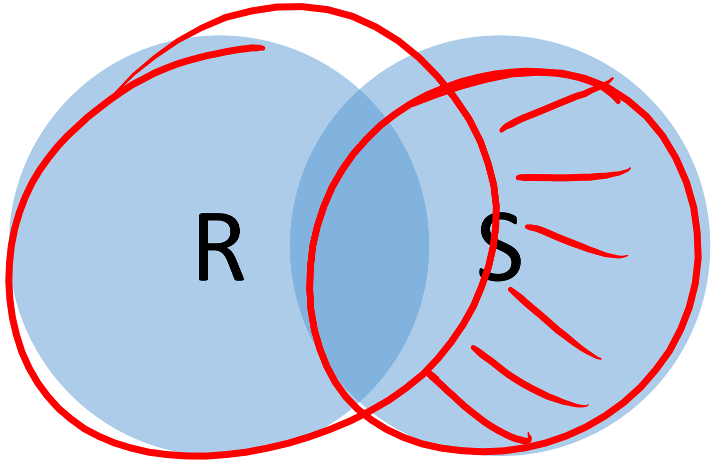
$$R \cap S = ((R \cup S) - (R - S)) - (S - R)$$
$$R \cap S = (R \cup S) - ((R - S) \cup (S - R))$$

- Derived operator using minus only!

$$R \cap S = \quad S \quad - (S - R)$$

- Derived using join

?



S − (S − R)

8. What about Intersection ∩?

- As derived operator using union and minus

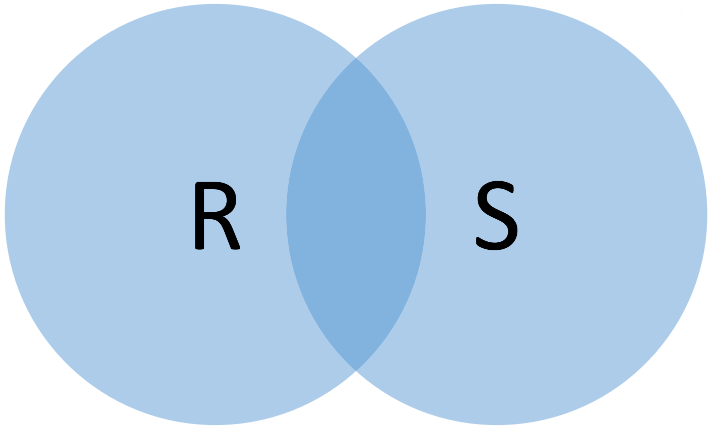
$$\boxed{\begin{array}{l} R \cap S = ((R \cup S) - (R - S)) - (S - R) \\ R \cap S = (R \cup S) - ((R - S) \cup (S - R)) \end{array}}$$

- Derived operator using minus only!

$$\boxed{R \cap S = \qquad S \qquad - (S - R)}$$

- Derived using join

$$\boxed{R \cap S = R \bowtie S}$$



Legal input: schemas need to be union compatible (same schema). E.g. not:
R(A,B,C)
S(A,B)

If R and S have the same schema, then $R \bowtie S$ and $R \ltimes S$ equal to $R \cap S$

Relational Algebra (RA) operators

- Five basic operators:
  1. Selection: σ ("sigma")
  2. Projection: Π
  3. Cartesian Product: $\times$
  4. Union: $\cup$
  5. Difference: $-$
- Auxiliary (or special) operator
  6. Renaming: ρ ("rho")
- Derived (or implied) operators
  7. Joins $\bowtie$ (natural, theta join, equi-join, [semi-join: moved to T3-U1])
  8. Intersection / complement
  9. Division

9. Division ($R \div S$)

- Consider two relations $R(X,Y)$ and $S(Y)$
- Then $R \div S$ is ...

X, Y are sets of attributes
Legal input: $att(R) \supset att(S)$

What could be a meaningful definition of division ?

Compare to Integer division: 7/2=3

3 is the biggest integer that multiplied with 2 is smaller or equal to 7

9. Division ($R \div S$)

- Consider two relations $R(X,Y)$ and $S(Y)$
- Then $R \div S$ is ...
  - ... the largest relation $T(X)$ s.t. $S \times T \subseteq R$

X, Y are sets of attributes
Legal input: $att(R) \supset att(S)$

(safety: $T \subseteq \pi_X R$)

9. Division (R ÷ S)

- Consider two relations R(X,Y) and S(Y)
- Then R ÷ S is ...
  - ... the largest relation T(X) s.t. S ⨯ T ⊆ R, or
  - ... the relation T(X) that contains the X's that occur with all Y's in S, or
  - ... {t(X) | ∀s(Y)∈S.[∃r(X,Y)∈R]} (+ safety)

X, Y are sets of attributes
Legal input: att(R) ⊃ att(S)
(safety: $T \subseteq \pi_X R$)

R Dividend

X	Y
Alice	1
Alice	2
Bob	1
Bob	2
Bob	3

S Divisor

Y
1
2
3

T

?

9. Division (R ÷ S)

- Consider two relations R(X,Y) and S(Y)
- Then R ÷ S is ...
  - ... the largest relation T(X) s.t. S × T ⊆ R, or
  - ... the relation T(X) that contains the X's that occur with all Y's in S, or
  - ... $\{t(X) \mid \forall s(Y) \in S.[\exists r(X,Y) \in R]\}$ (+ safety)

X, Y are sets of attributes
Legal input: att(R) ⊃ att(S)

(safety: $T \subseteq \pi_X R$)

R Dividend

X	Y
Alice	1
Alice	2
Bob	1
Bob	2
Bob	3

S Divisor

Y
1
2
3

T

X
Bob

Questions

Studies

sid	student	course
1	Alice	AI
1	Alice	DB
2	Bob	DB
2	Bob	ML
3	Charly	AI
3	Charly	DB
3	Charly	ML

÷

Course

course
ML

= ?

÷

course
AI
DB
ML

= ?

Questions

Studies

sid	student	course
1	Alice	AI
1	Alice	DB
2	Bob	DB
2	Bob	ML
3	Charly	AI
3	Charly	DB
3	Charly	ML

÷

Course

course
ML

=

sid	student
2	Bob
3	Charly

recall set semantics for RA

÷

course
AI
DB
ML

=

sid	student
3	Charly

Assume R,S have disjoint attribute sets (possibly by renaming)

(RxS)÷S = ?

(RxS)÷R = ?

Questions

recall set semantics for RA

Studies

sid	student	course
1	Alice	AI
1	Alice	DB
2	Bob	DB
2	Bob	ML
3	Charly	AI
3	Charly	DB
3	Charly	ML

÷ *Course*

course
ML

=

sid	student
2	Bob
3	Charly

÷

course
AI
DB
ML

=

sid	student
3	Charly

Assume R,S have disjoint attribute sets (possibly by renaming)

$$(R \times S) \div S = R$$

$$(R \times S) \div R = S$$

Q: If R has 1000 tuples and S has 100 tuples, how many tuples can be in R÷S?

?

Q: If R has 1000 tuples and S has 1001 tuples, how many tuples can be in R÷S?

?

Questions

3 3

R

recall set semantics for RA

Studies

sid	student	course
1	Alice	AI
1	Alice	DB
2	Bob	DB
2	Bob	ML
3	Charly	AI
3	Charly	DB
3	Charly	ML

÷

Course

course
ML

=

sid	student
2	Bob
3	Charly

÷

course
AI
DB
ML

=

sid	student
3	Charly

1000

10

3

Assume R,S have disjoint attribute sets (possibly by renaming)

(RxS)÷S = R

(RxS)÷R = S

Q: If R has 1000 tuples and S has 100 tuples, how many tuples can be in R÷S? 0-10

Q: If R has 1000 tuples and S has 1001 tuples, how many tuples can be in R÷S? 0

Questions

Studies

sid	student	course
1	Alice	AI
1	Alice	DB
2	Bob	DB
2	Bob	ML
3	Charly	AI
3	Charly	DB
3	Charly	ML

CourseType

course	type
AI	elective
DB	core
ML	core

Who took all core courses in RA with relational division?

?

Questions

Studies

sid	student	course
1	Alice	AI
1	Alice	DB
2	Bob	DB
2	Bob	ML
3	Charly	AI
3	Charly	DB
3	Charly	ML

CourseType

course	type
AI	elective
DB	core
ML	core

Who took all core courses in RA with relational division?

$$\text{Studies} \div \pi_{\text{course}}\left(\sigma_{\text{type}='\text{core}'}\text{CourseType}\right)$$

How to write R÷S in Primitive RA? (×,−, π)

$$R(X,Y) \div S(Y)$$

$R \div S = Q$

X	Y
a	0
a	1
a	2
b	1

Y
1
2

?

How to write R÷S in Primitive RA? (×,−, π)

$$R(X,Y) \div S(Y)$$

?

$R \div S = Q$

R:

X	Y
a	0
a	1
a	2
b	1
b	2

S:

Y
1
2

Q:

X
a

1

2

3

4: {a} = {a,b} – {b}

How to write R÷S in Primitive RA? (×,−, π)

$$R(X,Y) \div S(Y)$$

$$\boxed{\pi_X R \times S}$$

Each X of R w/ each Y of S

$R \div S = Q$

X	Y
a	0
a	1
a	2
b	1
b	2

Y
1
2

X
a

1

2

3

4: {a} = {a,b} – {b}

How to write R÷S in Primitive RA? (×,−, π)

$$R(X,Y) \div S(Y)$$

$$\boxed{\left(\boxed{\pi_X R \times S}\right) - R}$$

Each X of R w/ each Y of S

(X,Y) s.t. X in R, Y in S, but (X,Y) not in R

R ÷ S = Q

R:

X	Y
a	0
a	1
a	2
b	1
b	2

S:

Y
1
2

Q:

X
a

1

2

3

4: {a} = {a,b} − {b}

How to write R÷S in Primitive RA? (×,−, π)

$$R(X,Y) \div S(Y)$$

$$\pi_X\Big(\big(\boxed{\pi_X R \times S}\big) - R\Big)$$

Each X of R w/ each Y of S

(X,Y) s.t. X in R, Y in S, but (X,Y) not in R

Xs in R where for some Y in S, (X,Y) is not in R

R ÷ S = Q

R:

X	Y
a	0
a	1
a	2
b	1
b	2

S:

Y
1
2

Q:

X
a

1

2

3

4: {a} = {a,b} − {b}

How to write R÷S in Primitive RA? (×,−, π)

$$R(X,Y) \div S(Y)$$

$$\pi_X R - \pi_X\Big(\big(\pi_X R \times S\big) - R\Big)$$

Each X of R w/ each Y of S

(X,Y) s.t. X in R, Y in S, but (X,Y) not in R

Xs in R where for some Y in S, (X,Y) is not in R

R÷S

R ÷ S = Q

X	Y
a	0
a	1
a	2
b	1
b	2

Y
1
2

X
a

1

2

3

4: {a} = {a,b} − {b}

What if S=Ø?

$$R(X,Y) \div S(Y)$$

$R \div S = Q$

X	Y
a	0
a	1
a	2
b	1

Y

?

What if S=∅?

$$R(X,Y) \div S(Y)$$

$$\pi_X R - \pi_X\Big((\pi_X R \times S) - R\Big)$$

$R \div S = Q$

X	Y
a	0
a	1
a	2
b	1

Y

X
a
b

Recall: $\{t(X) \mid \forall s(Y) \in S.[\exists r(X,Y) \in R]\}$ (+ safety)

Now you see why we needed the safety condition "$T \subseteq \pi_X R$" when defining "$R \div S$ as the largest relation $T(X)$ s.t. $S \times T \subseteq R$"

R÷S in Primitive RA vs. RC

$$R(X,Y) \div S(Y)$$

In RA:

$$\pi_X R - \pi_X\big((\pi_X R \times S) - R\big)$$

In DRC:

?

$R \div S = Q$

X	Y
a	0
a	1
a	2
b	1
b	2

Y
1
2

X
a

R÷S in Primitive RA vs. RC

$$R(X,Y) \div S(Y)$$

In RA:

$$\pi_X R - \pi_X\big((\pi_X R \times S) - R\big)$$

In DRC:

$$\{\, X \mid \exists Z.[R(X,Z)] \wedge \quad \textbf{?} \quad \}$$

X is "guarded": safe and thus domain independent

$R \div S = Q$

X	Y
a	0
a	1
a	2
b	1
b	2

Y
1
2

X
a

R÷S in Primitive RA vs. RC

$$R(X,Y) \div S(Y)$$

In RA:

$$\pi_X R - \pi_X\Big((\pi_X R \times S) - R\Big)$$

In DRC: what if $S(Y)=\emptyset$?

?

$$\{\, X \mid \exists Z.[R(X,Z)] \land \forall Y.[S(Y) \rightarrow R(X,Y)] \,\}$$

$R \div S = Q$

R:

X	Y
a	0
a	1
a	2
b	1
b	2

S:

Y
1
2

Q:

X
a

R÷S in Primitive RA vs. RC

$$R(X,Y) \div S(Y)$$

In RA:

$$\pi_X R - \pi_X\Big(\big(\pi_X R \times S\big) - R\Big)$$

In DRC:

$$\{\, X \mid \exists Z.[R(X,Z)] \land \forall Y.[S(Y) \rightarrow R(X,Y)] \,\}$$

? without universal quantification

$R \div S = Q$

X	Y
a	0
a	1
a	2
b	1
b	2

Y
1
2

X
a

R÷S in Primitive RA vs. RC

$$R(X,Y) \div S(Y)$$

$R \div S = Q$

R:

X	Y
a	0
a	1
a	2
b	1
b	2

S:

Y
1
2

Q:

X
a

In RA:

$$\pi_X R - \pi_X\big((\pi_X R \times S) - R\big)$$

In DRC:

$$\{\, X \mid \exists Z.[R(X,Z)] \wedge \forall Y.[S(Y) \rightarrow R(X,Y)] \,\}$$

$$\{\, X \mid \exists Z.[R(X,Z)] \wedge \nexists Y.[S(Y) \wedge \neg R(X,Y)] \,\}$$

In TRC:

?

R÷S in Primitive RA vs. RC

$$R(X,Y) \div S(Y)$$

R ÷ S = Q

X	Y
a	0
a	1
a	2
b	1
b	2

Y
1
2

X
a

In RA:

$$\pi_X R - \pi_X\big((\pi_X R \times S) - R\big)$$

In DRC:

$$\{ X \mid \exists Z.[R(X,Z)] \wedge \forall Y.[S(Y) \rightarrow R(X,Y)] \}$$

$$\{ X \mid \exists Z.[R(X,Z)] \wedge \nexists Y.[S(Y) \wedge \neg R(X,Y)] \}$$

In TRC:

$$\{ r.A \mid \exists r \in R.[\nexists s \in S.[\qquad ? \qquad]]] \}$$

R÷S in Primitive RA vs. RC

$$R(X,Y) \div S(Y)$$

R		÷	S	=	Q
X	Y		Y		X
a	0		1		a
a	1		2		
a	2				
b	1				
b	2				

In RA:

$$\pi_X R - \pi_X\Big((\pi_X R \times S) - R\Big)$$

In DRC:

$$\{\, X \mid \exists Z.[R(X,Z)] \wedge \forall Y.[S(Y) \rightarrow R(X,Y)] \,\}$$

$$\{\, X \mid \exists Z.[R(X,Z)] \wedge \nexists Y.[S(Y) \wedge \neg R(X,Y)] \,\}$$

? in SQL

In TRC:

$$\{\, r.A \mid \exists r \in R.[\, \nexists s \in S.[\nexists r_2 \in R.[r_2.Y = s.Y \wedge r_2.X = r.X]]]\}$$

R÷S in Primitive RA vs. RC

In SQL

```
SELECT DISTINCT R.A
FROM R
WHERE not exists (
        SELECT *
        FROM S
        WHERE not exists (
                SELECT *
                FROM R AS R2
                WHERE R2.B=S.B
                AND R2.A=R.A))
```

R ÷ S = Q

X	Y
a	0
a	1
a	2
b	1
b	2

Y
1
2

X
a

In TRC:

$$\{ r.A \mid \exists r \in R.[\, \nexists s \in S.[\nexists r_2 \in R.[r_2.B = s.B \wedge r_2.A = r.A]]]\}$$

Parentheses Convention

- We have defined 3 unary operators and 3 binary operators
- It is acceptable to omit the parentheses from o(R) when o is unary
  - Then, unary operators take precedence over binary ones
- Example:

$$(\sigma_{course='DB'}(\text{Course})) \times (\rho_{cid \rightarrow cid1}(\text{Studies}))$$

becomes

$$\sigma_{course='DB'}\text{Course} \times \rho_{cid \rightarrow cid1}\text{Studies}$$

Algebra and the connection to logic and queries

- Algebra
- Relational Algebra
  - Operators
  - Independence
  - Power of algebra: optimizations
- Equivalence RA and safe RC (Codd's theorem)
  - RA $\rightarrow$ RC
  - RC $\rightarrow$ RA

5 Primitive Operators

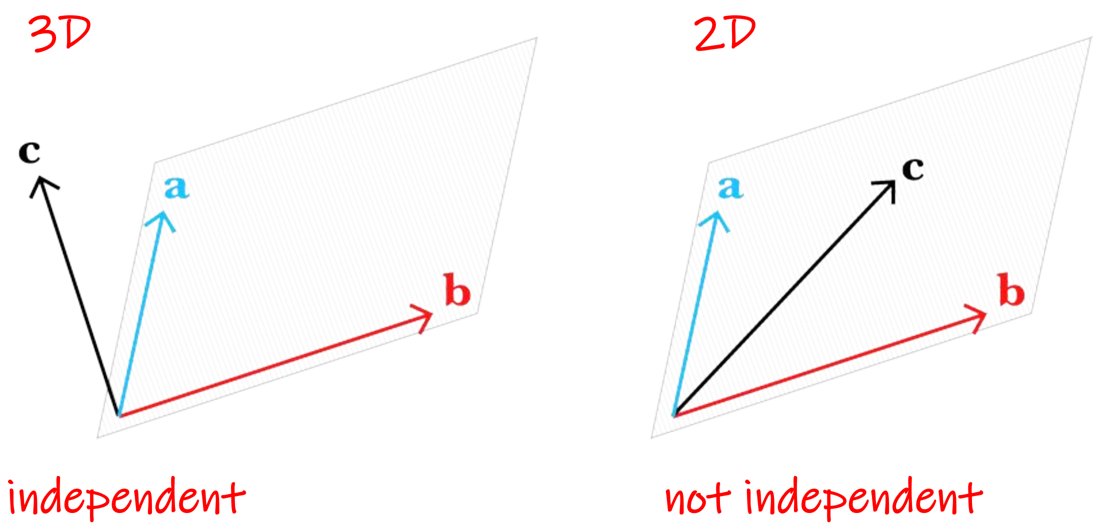
1. Projection (π)
2. Selection (σ)
3. Union ($\cup$)
4. Set Difference ($-$)
5. Cross Product ($\times$)

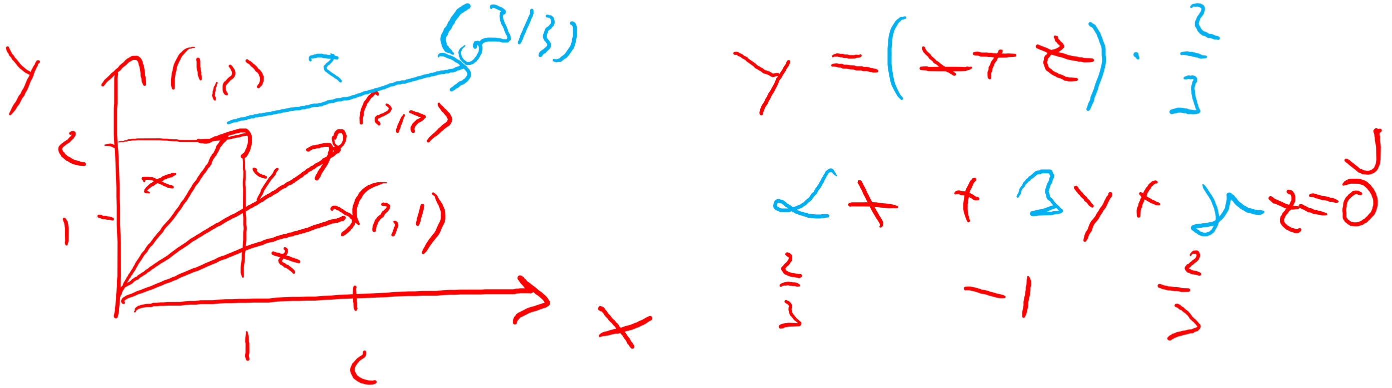
Is this a well chosen set of primitives?

?

5 Primitive Operators

1. Projection (π)
2. Selection (σ)
3. Union ($\cup$)
4. Set Difference ($-$)
5. Cross Product ($\times$)





Is this a well chosen set of primitives?

Could we drop an operator "without losing anything"?

Independence among Primitives

- Let o be an RA operator, and let A be a set of RA operators
- We say that o is independent of A if o cannot be expressed in A; that is, no expression in A is equivalent to o

$\boxed{\text{THEOREM: Each of the five primitives is independent of the other four}}$

$\{\pi, \sigma, \times, \cup, -\}$

Proof:

- Separate argument for each of the 5 (For each operator, we need to discover a property that is possessed by that operator, but not by any RA expression that involves only the other 4 operations)
- Arguments follow a common pattern (union as example next slides)

Recipe for Proving Independence of an operator o

1. Fix a schema S and an instance D over S

2. Find some property P over relations

3. Prove: for every expression φ that does not use o, the relation φ(D) satisfies P

   Such proofs are typically by induction on the size of the expression, since operators compose

4. Find an expression ψ such that ψ uses o and ψ(D) violates P

Concrete Example: Proving Independence of Union ∪

1. Fix a schema *S* and an instance D over *S*

   S: R(A), S(A)   D: {R(0), S(1)}

R
A
0

S
A
1

2. Find some property P over relations

   #tuples < 2

3. Prove: for every expression φ that does not use ∘, the relation φ(D) satisfies P

   Induction base: R and S have #tuples<2

   Induction step: If $\varphi_1(D)$ and $\varphi_2(D)$ have #tuples<2, then so do:

   $\sigma_c(\varphi_1(D))$, $\pi_A(\varphi_1(D))$, $\varphi_1(D) \times \varphi_2(D)$, $\varphi_1(D) - \varphi_2(D)$, $\rho_{A \to B}(\varphi_1(D))$

4. Find an expression ψ such that ψ uses ∘ and ψ(D) violates P

   $\psi = R \cup S$

Algebra and the connection to logic and queries

- Algebra
- Relational Algebra
  - Operators
  - Independence
  - Power of algebra: optimizations
- Equivalence RA and safe RC (Codd's theorem)
  - RA $\rightarrow$ RC
  - RC $\rightarrow$ RA

Commutativity and distributivity of RA operators

- The basic commutators:
  - Push <u>projection</u> through selection, join, union — $\pi_{\mathbf{A}}(R \cup S) = \pi_{\mathbf{A}}(R) \cup \pi_{A}(S)$
  - Push <u>selection</u> through projection, join, union — $\sigma_{\theta}(R \cup S) = \sigma_{\theta}(R) \cup \sigma_{\theta}(S)$
  - Also: Joins can be re-ordered! — $(R \cup S) \times T = (R \times T) \cup (S \times T)$

- Note that this is not an exhaustive set of operations

What about sorting and joins?

This simple set of tools allows us to greatly improve the execution time of queries by optimizing RA plans!

We next illustrate with an SFW (Select-From-Where) query

An example: SQL to RA to Optimized RA

```
R(A,B) S(B,C) T(C,D)
```

```
SELECT R.A,T.D
FROM   R,S,T
WHERE  R.B = S.B
   and S.C = T.C
   and R.A < 10;
```

in RA

?

An example: SQL to RA to Optimized RA

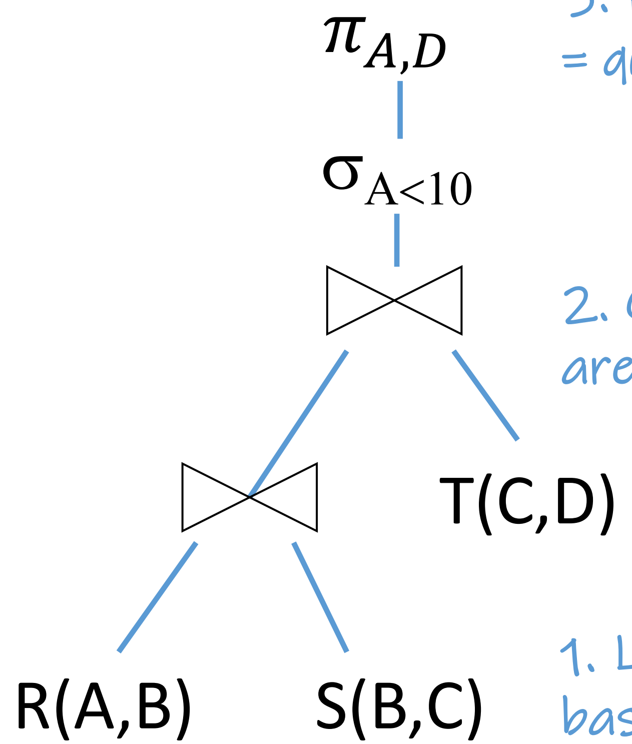
Heuristic: have selection and projection earlier to have fewer (or smaller) "intermediate" tuples

```
R(A,B) S(B,C) T(C,D)
```

```sql
SELECT R.A,T.D
FROM   R,S,T
WHERE  R.B = S.B
   and S.C = T.C
   and R.A < 10;
```

in RA

$$\pi_{A,D}\left(\sigma_{A<10}\left(T \bowtie (R \bowtie S)\right)\right)$$



3. Root node = query results

2. Other nodes are operators

1. Leaves are base relations

Query tree / expression tree / computation tree / data flow graph

An example: SQL to RA to Optimized RA

Heuristic: have selection and projection earlier to have fewer (or smaller) "intermediate" tuples

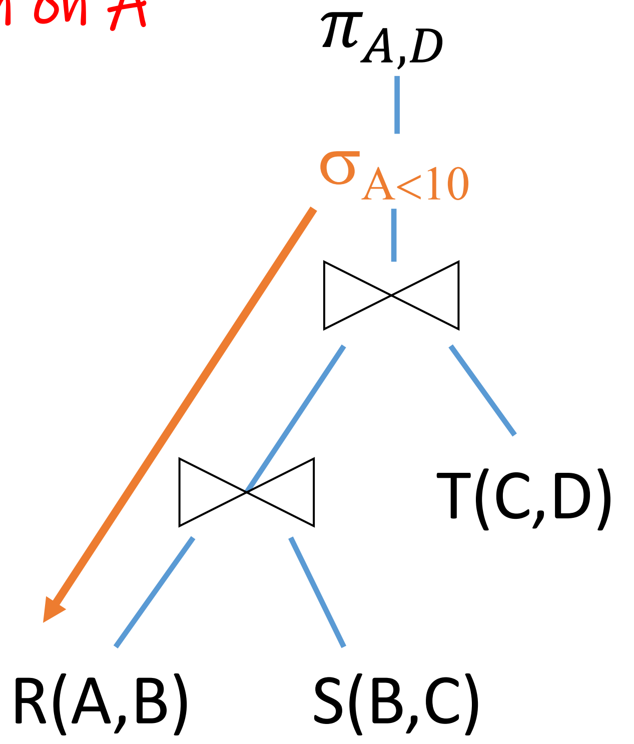
R(A,B) S(B,C) T(C,D)

```
SELECT R.A,T.D
FROM   R,S,T
WHERE  R.B = S.B
   and S.C = T.C
   and R.A < 10;
```

in RA

$$\pi_{A,D}\left(\sigma_{A<10}\left(T \bowtie (R \bowtie S)\right)\right)$$

1. Push down selection on A



Pushing down may be suboptimal if selection condition is very expensive (e.g. running some image processing algorithm). Projection could be unnecessary effort (but more rarely).

An example: SQL to RA to Optimized RA

Heuristic: have selection and projection earlier to have fewer (or smaller) "intermediate" tuples

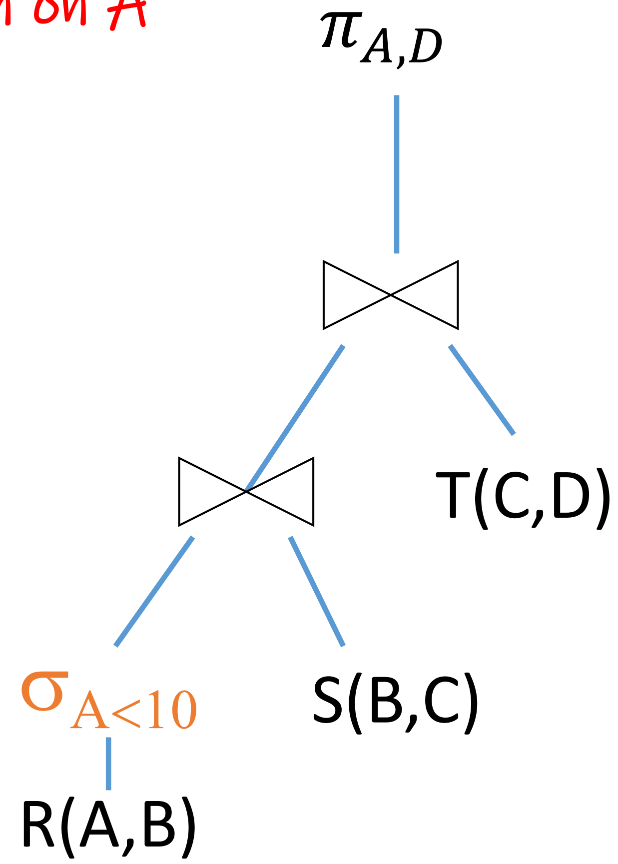
```
R(A,B) S(B,C) T(C,D)
```

```sql
SELECT R.A,T.D
FROM   R,S,T
WHERE  R.B = S.B
   and S.C = T.C
   and R.A < 10;
```

in RA

$$\pi_{A,D}\left(T \bowtie \left(\sigma_{A<10} R \bowtie S\right)\right)$$

1. Push down selection on A



An example: SQL to RA to Optimized RA

Heuristic: have selection and projection earlier to have fewer (or smaller) "intermediate" tuples

```
R(A,B) S(B,C) T(C,D)
```

```
SELECT R.A,T.D
FROM   R,S,T
WHERE  R.B = S.B
   and S.C = T.C
   and R.A < 10;
```

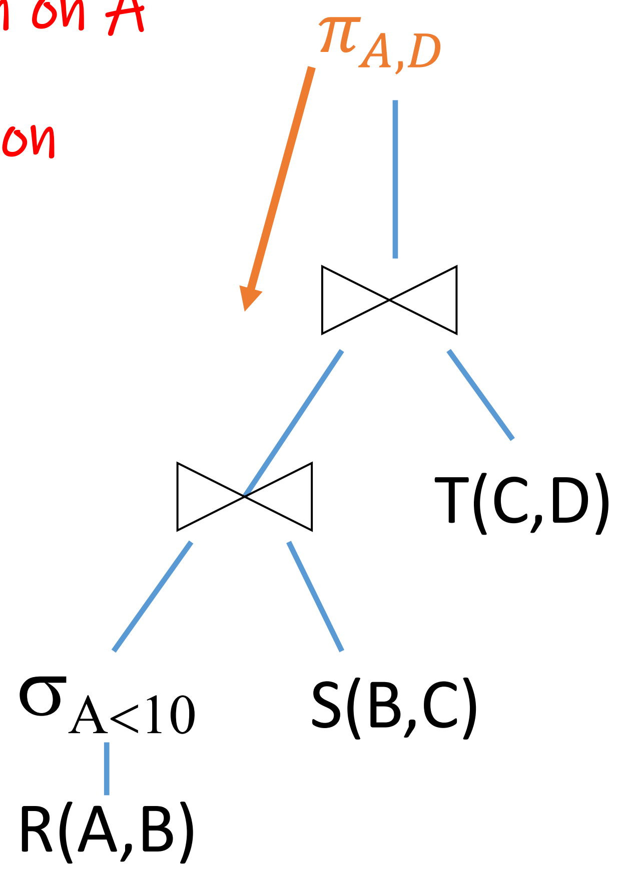
1. Push down selection on A

2. Push down projection

$\pi_{-B,C}$

in RA

$$\pi_{A,D}\left(T \bowtie \left(\sigma_{A<10} R \bowtie S\right)\right)$$



$\pi_{-B,C}$

An example: SQL to RA to Optimized RA

Variable Elimination!

`R(A,B) S(B,C) T(C,D)`

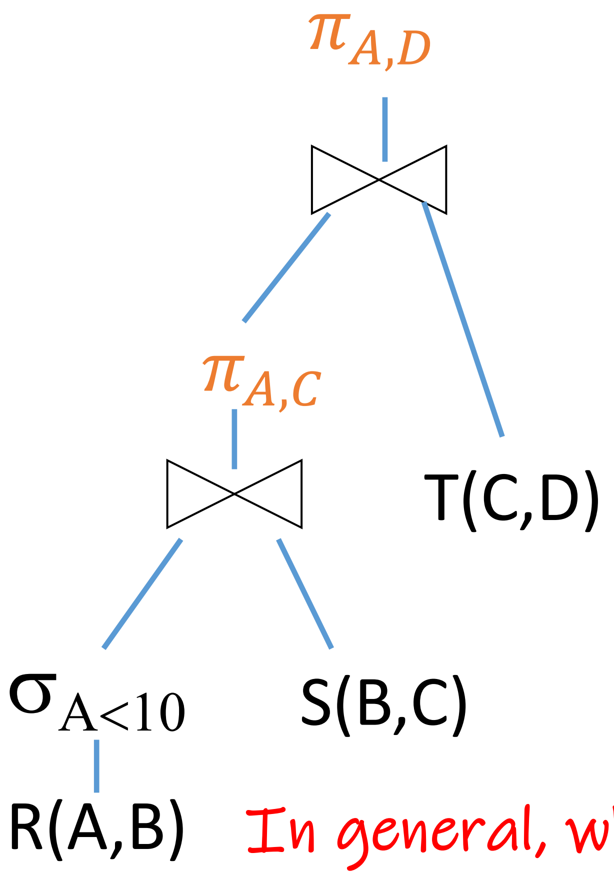
```sql
SELECT R.A,T.D
FROM   R,S,T
WHERE  R.B = S.B
   and S.C = T.C
   and R.A < 10;
```

in RA

$$\pi_{A,D}\left(T \bowtie \pi_{A,C}(\sigma_{A<10}R \bowtie S)\right)$$

π_{-C} π_{-B}

We now eliminate B earlier



π_{-C}

π_{-B}

In general, when is an attribute not needed?

Algebra and the connection to logic and queries

- Algebra
- Relational Algebra
  - Operators
  - Independence
  - Power of algebra: optimizations
- Equivalence RA and safe RC (Codd's theorem)
  - RA $\rightarrow$ RC
  - RC $\rightarrow$ RA

"Clear" variables*

Formula with clear variables : each quantifier "has its own variables" & each variable has only free or only bound occurrences

$$\forall x.\, \exists y.\, R(x, y, z) \wedge \neg \exists x.\, S(y, x)$$

? which variables are free or bound?

* "Clear variable" is a non-standard term used in excellent slides by Marie Duzi: http://www.cs.vsb.cz/duzi/

"Clear" variables*

Formula with clear variables : each quantifier "has its own variables" & each variable has only free or only bound occurrences

$$\forall x.\, \exists y. [R(x, y, z)] \land \neg \exists x.\, S(y, x)$$

bound free bound

notice operator precedence: ∃ before ∧
$\forall x. \exists y. [R(x,y,z)] \land \neg \exists x. [S(y,x)]$

Not "clear": Two x's and y's are different variables.

? how to make it "clear"

* "Clear variable" is a non-standard term used in excellent slides by Marie Duzi: http://www.cs.vsb.cz/duzi/

"Clear" variables*

Formula with clear variables : each quantifier "has its own variables" & each variable has only free or only bound occurrences

$\forall x. \exists y. R(x, y, z) \wedge \neg \exists x. S(y, x)$

bound free bound

notice operator precedence: ∃ before ∧

$\forall x. \exists y. [R(x,y,z)] \wedge \neg \exists x. [S(y,x)]$

Not "clear": Two x's and y's are different variables.

$\forall x. \exists y. R(x, y, z) \wedge \neg \exists u. S(v, u)$

now "clear"

$\{(z, v) \mid \forall x. \exists y. R(x, y, z) \wedge \neg \exists u. S(v, u)\}$

Now a query. But how to make it domain-independent ?

* "Clear variable" is a non-standard term used in excellent slides by Marie Duzi: http://www.cs.vsb.cz/duzi/

"Clear" variables*

Formula with clear variables : each quantifier "has its own variables" & each variable has only free or only bound occurrences

$\forall x. \exists y. R(x, y, z) \wedge \neg \exists x. S(y, x)$

bound free bound

notice operator precedence: ∃ before ∧

$\forall x. \exists y. [R(x,y,z)] \wedge \neg \exists x. [S(y,x)]$

Not "clear": Two x's and y's are different variables.

$\forall x. \exists y. R(x, y, z) \wedge \neg \exists u. S(v, u)$

now "clear"

$R(\_,\_,z) \wedge$ $S(\_,v) \wedge$

$\{(z, v) \mid \cancel{\forall x. \exists y. R(x, y, z)} \wedge \neg \exists u. S(v, u)\}$

$\forall x. [R(x,\_,\_) \rightarrow R(x,\_,z)]$

Now a query. But how to make it domain-independent

* "Clear variable" is a non-standard term used in excellent slides by Marie Duzi: http://www.cs.vsb.cz/duzi/

"Clear" variables*

Formula with clear variables : each quantifier "has its own variables" & each variable has only free or only bound occurrences

$$\forall x.\, \exists y.\, R(x, y, z) \land \neg\exists x.\, S(y, x)$$

bound free bound

notice operator precedence: ∃ before ∧

$\forall x. \exists y. [R(x,y,z)] \land \neg \exists x. [S(y,x)]$

Not "clear": Two x's and y's are different variables.

$$\forall x.\, \exists y.\, R(x, y, z) \land \neg\exists u.\, S(v, u)$$

now "clear"

$\exists s,t. R(s,t,z) \land$ $\exists p. S(p,v) \land$

$$\{(z, v) \mid \cancel{\forall x.\, \exists y.\, R(x, y, z)} \land \neg\exists u.\, S(v, u)\}$$

$\forall x. [\exists w,t. R(x,w,t) \rightarrow \exists y. R(x,y,z)]$

Now a query. But how to make it domain-independent

Repeated variable names

In sentences with multiple quantifiers, distinct variables do not need to range over distinct objects! (cp. homomorphism vs. isomorphism)

?

Which of the following formulas imply each other?

$\forall x. \forall y.\ E(x,y)$

$\forall x.\ E(x,x)$

$\exists x. \exists y.\ E(x,y)$

$\exists x.\ E(x,x)$

Repeated variable names

In sentences with multiple quantifiers, distinct variables do not need to range over distinct objects! (cp. homomorphism vs. isomorphism)

Assume DOM = {1, 2}:

E

s	t
1	1
1	2
2	1
2	2

$\forall x.\forall y.\ E(x,y) \quad \Rightarrow \quad \forall x.\ E(x,x)$

$\exists x.\exists y.\ E(x,y) \quad \Leftarrow \quad \exists x.\ E(x,x)$

E

s	t
1	2

Repeated variable names

In sentences with multiple quantifiers, <u>distinct variables do not need to range over distinct objects</u>! (cp. homomorphism vs. isomorphism)

Assume DOM = ∅ :

$$\forall x.\forall y.\, E(x,y) \quad \Rightarrow \quad \forall x.\, E(x,x)$$

$\Downarrow$ Only if domain is not empty! Dom $\neq \emptyset$ $\Downarrow$

$$\exists x.\exists y.\, E(x,y) \quad \Leftarrow \quad \exists x.\, E(x,x)$$

E

s	t

Example RC→RA

Person(id, name, country)
Spouse(id1, id2)

In DRC:

$$\{\, x \mid \exists z,w.\ \mathrm{Person}(x,z,w) \land \forall y.[\neg \mathrm{Spouse}(x,y)] \,\}$$

In RA:

?

Example RC→RA

Person(id, name, country)
Spouse(id1, id2)

In DRC:

$$\{\ x \mid \exists z,w.\ \text{Person}(x,z,w) \land \forall y.[\neg \text{Spouse}(x,y)]\ \}$$

$$\{\ x \mid \exists z,w.\ \text{Person}(x,z,w) \land \neg\exists y.[\text{Spouse}(x,y)]\ \}$$

In RA:

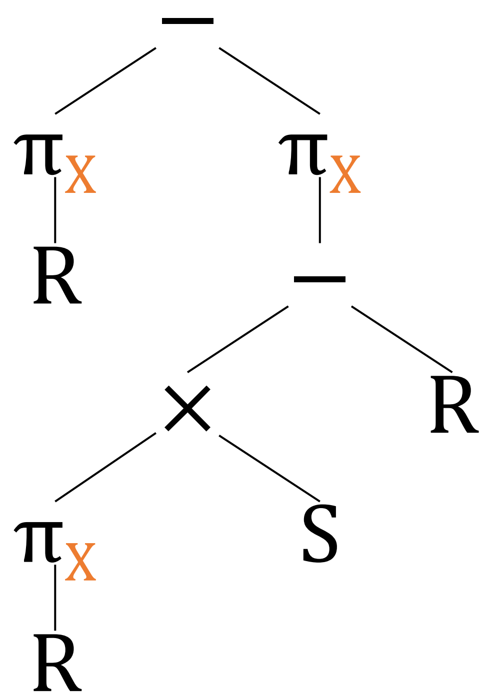
$$\pi_{\text{id}}\text{Person} - \pi_{\text{id1}}\text{Spouse}$$

$$\pi_{\text{id}}\text{Person} - \rho_{\text{id1}\to\text{id}}\,(\pi_{\text{id1}}\text{Spouse})$$

Recall: named vs ordered perspective

Example RA→RC

$$R(X,Y) \div S(Y)$$



In RA:

$$\pi_X R - \pi_X((\pi_X R \times S) - R)$$

In DRC:

$$\{\, X \mid \exists Z.[R(X,Z)] \land \forall Y.[S(Y) \rightarrow R(X,Y)] \,\}$$

$$\{\, X \mid \exists Z.[R(X,Z)] \land \neg\exists Y.[S(Y) \land \neg R(X,Y)] \,\}$$

Equivalence Between RA and Domain-Independent RC

CODD'S THEOREM:
RA and **domain-independent** RC
have the same expressive power.

More formally, on every schema $\mathbf{S}$:

1. For every RA expression E, there is a domain-independent RC query Q s.t. $\mathrm{Q} \equiv \mathrm{E}$
2. For every domain-independent RC query Q, there is an RA expression E s.t. $\mathrm{Q} \equiv \mathrm{E}$

The proof has two directions:

RA→RC:
by induction on the size of the RA expression

RC→RA:
more involved

See also: https://en.wikipedia.org/wiki/Codd%27s_theorem

Algebra and the connection to logic and queries

- Algebra
- Relational Algebra
  - Operators
  - Independence
  - Power of algebra: optimizations
- Equivalence RA and safe RC (Codd's theorem)
  - RA → RC
  - RC → RA

RA → DRC: Intuition

- Construction by induction
- Key technical detail: need to maintain a mapping b/w attribute names and variables

Intuition: $\{x \mid \exists y.[R(x,y)] \land \exists y.[S(x,y)]\}$

contrast with: $\{x \mid \exists y.[R(x,y)] \land \exists z.[S(x,z)]\}$

$Q(1) \leftarrow R(1,2), S(1,3)$

$y=2$ $y=3$

Here, ϕ_i is the formula constructed for expression E_i

RA expression	DRC formula ϕ
R (n columns)	$R(X_1,\ldots,X_n)$
$E_1 \times E_2$	
$E_1 \cup E_2$	
$E_1 - E_2$	
$\pi_{A_1,\ldots,A_k}(E_1)$	
$\sigma_c(E_1)$	

RA → DRC: Intuition

Intuition: $\{x \mid \exists y.[R(x,y)] \land \exists y.[S(x,y)]\}$

contrast with: $\{x \mid \exists y.[R(x,y)] \land \exists z.[S(x,z)]\}$

- Construction by induction
- Key technical detail: need to maintain a mapping b/w attribute names and variables

RA expression	DRC formula ϕ Here, ϕ_i is the formula constructed for expression E_i
R (n columns)	$R(X_1,\ldots,X_n)$
$E_1 \times E_2$	$\phi_1 \land \phi_2$ disjoint variables (rename)
$E_1 \cup E_2$	$\phi_1 \lor \phi_2$ use identical variables (rename) UNION COMPATIBLE
$E_1 - E_2$	$\phi_1 \land \neg\phi_2$ use identical variables (rename)
$\pi_{A_1,\ldots,A_k}(E_1)$	
$\sigma_c(E_1)$	

RA → DRC: Intuition

Intuition: $\{x \mid \exists y.[R(x,y)] \wedge \exists y.[S(x,y)]\}$

contrast with: $\{x \mid \exists y.[R(x,y)] \wedge \exists z.[S(x,z)]\}$

- Construction by induction
- Key technical detail: need to maintain a mapping b/w attribute names and variables

RA expression	DRC formula ϕ Here, ϕ_i is the formula constructed for expression E_i
R (n columns)	$R(X_1,\ldots,X_n)$
$E_1 \times E_2$	$\phi_1 \wedge \phi_2$ disjoint variables (rename)
$E_1 \cup E_2$	$\phi_1 \vee \phi_2$ use identical variables (rename)
$E_1 - E_2$	$\phi_1 \wedge \neg\phi_2$ use identical variables (rename)
$\pi_{A_1,\ldots,A_k}(E_1)$	$\exists X_1 \ldots \exists X_m . \phi_1$ where $X_1, \ldots, X_m$ are the variables not among $A_1, \ldots, A_k$
$\sigma_c(E_1)$	$\phi_1 \wedge c$

Correspondence more natural with project-away operator: $\pi_{-A_1,\ldots,A_m}(E_1)$

RA → DRC: Example R÷S

R(A,B) S(B)

RA	DRC	Mapping
R		
$\pi_A(R)$		
S		
$\pi_A(R)\times S$		
$(\pi_A(R)\times S) - R$		
$\pi_A\big((\pi_A(R)\times S) - R\big)$		
$\pi_A(R) - \pi_A\big((\pi_A(R)\times S) - R\big)$		

RA → DRC: Example R÷S

R(A,B) S(B)

RA	DRC	Mapping
R	$R(x, y)$	x:R.A, y:R.B
$\pi_A(R)$	$\exists y.\, R(x, y)$	x:R.A
S	$S(z)$	z:S.B
$\pi_A(R) \times S$		
$(\pi_A(R) \times S) - R$		
$\pi_A\big((\pi_A(R) \times S) - R\big)$		
$\pi_A(R) - \pi_A\big((\pi_A(R) \times S) - R\big)$		

RA → DRC: Example R÷S

R(A,B)   S(B)

RA	DRC	Mapping
R	$R(x, y)$	x:R.A, y:R.B
$\pi_A(R)$	$\exists y. R(x, y)$	x:R.A
S	$S(z)$	z:S.B
$\pi_A(R) \times S$	$\exists y. \boxed{R(x, y)} \land S(z)$ z needs to be different from y	x:R.A, z:S.B
$(\pi_A(R) \times S) - R$	$\big(\exists y. R(x, y) \land S(z)\big) \land \neg R(x, z)$	x:R.A, z:S.B
$\pi_A\big((\pi_A(R) \times S) - R\big)$	$\exists z\big[\big(\exists y. R(x, y) \land S(z)\big) \land \neg R(x, z)\big]$	x:R.A
$\pi_A(R) - \pi_A\big((\pi_A(R) \times S) - R\big)$	$\exists y. R(x, y) \land \neg \exists z\big[\big(\exists y. R(x, y) \land S(z)\big) \land \neg R(x, z)\big]$ x's need to be same variable	x:R.A

y's don't need to be same variable

RA → DRC: Example R÷S

R(A,B) S(B)

This is the DRC expression we got by translating from RA:

$$\{\, x \mid \exists y[R(x, y)] \land \neg\exists z[\exists y[R(x, y) \land S(z)] \land \neg R(x, z)]\}$$

This is the DRC expression for relational division that we saw earlier.

$$\{\, x \mid \exists y[R(x, y)] \land \neg\exists z[\qquad S(z) \quad \land \neg R(x, z)]\}$$

Claim: there is no logically equivalent RA expression that uses the table R only twice.
For details see: https://arxiv.org/pdf/2203.07284