CS660: Graduate Intro to Database Systems

Class 12: *Relational Algebra*

Instructor: Manos Athanassoulis

https://bu-disc.github.io/CS660/

Reminders

Relation

Schema: relation name, attributes (type & name)

Students(*sid*: string, *name*: string, *login*: string, *age*: integer, *gpa*: real)

Instance

a *table* containing *rows* of such *columns*

every relation instance is a <u>set</u> (all rows distinct)

Relational Algebra

Relational Query Languages

Selection & Projection

Union, Set Difference & Intersection

Cross product & Joins

Examples

Division

Multisets

Units

Relational Algebra

Relational Query Languages

Selection & Projection

Union, Set Difference & Intersection

Cross product & Joins

Examples

Division

Multisets

Units

Relational Query Languages

Query languages: manipulation and retrieval of data

Relational model supports simple, powerful QLs:

Strong formal foundation based on logic.

Allows for much optimization.

Query Languages != programming languages!

QLs not expected to be "Turing complete".

QLs not intended to be used for complex calculations.

QLs support easy, efficient access to large data sets.

Turing completeness

A system of data-manipulation rules (e.g., a computer's instruction set, a programming language, or a cellular automaton) is said to be **Turing Complete** or *computationally universal* if it can be used to simulate any single-taped **Turing** machine.

"Can perform any general-purpose computation"

6

Formal Relational Query Languages

Two mathematical Query Languages form the basis for "real" languages (e.g. SQL), and for implementation:

user-facing

Relational Calculus: Lets users describe what they want, rather than how to compute it. (Non-procedural, *declarative*.)

Relational Algebra: More operational, very useful for representing execution plans.

system-facing

Understanding Algebra is key to understanding query processing!

Preliminaries

Query

from a **relation instance** to a **relation instance**

input & output schema

different but *fixed*
queries run over any legal instances
output schema defined by the query constructs

attribute notation

positional & name-field

Relational Algebra: 5 Basic Operations

Selection (σ) Selects a subset of *rows* from relation (horizontal).

Projection (π) Retains only wanted *columns* from relation (vertical).

Cross-product ($\times$) Allows us to combine two relations.

Set-difference ($-$) Tuples in R_1, but not in R_2.

Union ($\cup$) Tuples in R_1 and/or in R_2.

each operation returns a relation : ***composability*** (Algebra is "closed")

Example Instances

Boats

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Sailors: S₁

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

Reserve: R₁

sid	bid	day
22	101	10/10/16
58	103	11/12/16

Sailors: S₂

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

Relational Algebra

Relational Query Languages

Selection & Projection

Union, Set Difference & Intersection

Cross product & Joins

Examples

Division

Multisets

Units

Projection

Examples: $\pi_{age}(S_2)$; $\pi_{sname, rating}(S_2)$

retains only attributes that are in the *"projection list"*

schema of result:

    fields of projection list (with the same names)

projection operator has to *eliminate duplicates*
  why we may have duplicates? why remove them?

    Note: systems typically don't do duplicate elimination unless the user explicitly asks for it (Why not?)

Projection

sname	rating
yuppy	9
lubber	8
guppy	5
rusty	10

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S₂

$$\pi_{sname,rating}(S_2)$$

Projection

sname	rating
yuppy	9
lubber	8
guppy	5
rusty	10

$$\pi_{sname,rating}(S_2)$$

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S_2

age
35.0
55.5

$$\pi_{age}(S_2)$$

Selection (σ)

selects rows that satisfy a *selection condition*

result: has the same *schema* as the input relation

do we need to do <u>duplicate elimination</u>?

sid	sname	rating	age
28	yuppy	9	35.0
~~31~~	~~lubber~~	~~8~~	~~55.5~~
~~44~~	~~guppy~~	~~5~~	~~35.0~~
58	rusty	10	35.0

$$\sigma_{rating > 8}(S_2)$$

Selection (σ)

selects rows that satisfy a *selection condition*

result: has the same *schema* as the input relation

do we need to do <u>duplicate elimination</u>?

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

$$\sigma_{rating>8}(S_2)$$

sname	rating
yuppy	9
rusty	10

$$\pi_{sname,rating}(\sigma_{rating>8}(S_2))$$

Relational Algebra

Relational Query Languages

Selection & Projection

Union, Set Difference & Intersection

Cross product & Joins

Examples

Division

Multisets

Units

Union and Set-Difference

the _set_ operations take two input relations
which must be _union-compatible_

(i) same number of fields

(ii) "corresponding" fields have the same type

for which, if any, is duplicate elimination required?
(union/~~set-difference~~)

Union

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S_1

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S_2

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0
44	guppy	5	35.0
28	yuppy	9	35.0

$S_1 \cup S_2$

Set Difference

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S_1

sid	sname	rating	age
22	dustin	7	45.0

$S_1 - S_2$

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S_2

sid	sname	rating	age
28	yuppy	9	35.0
44	guppy	5	35.0

$S_2 - S_1$

Compound Operator: Intersection

in addition to the 5 basic operators

several additional ***compound operators***

no new computational power, but useful shorthands

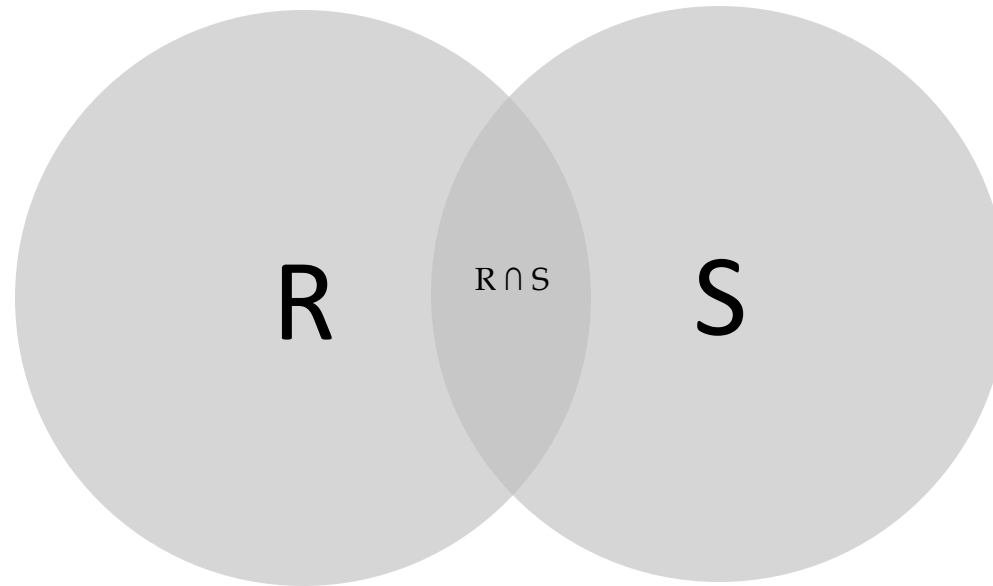*can be expressed solely with the basic ops*

intersection takes two *union-compatible* relations

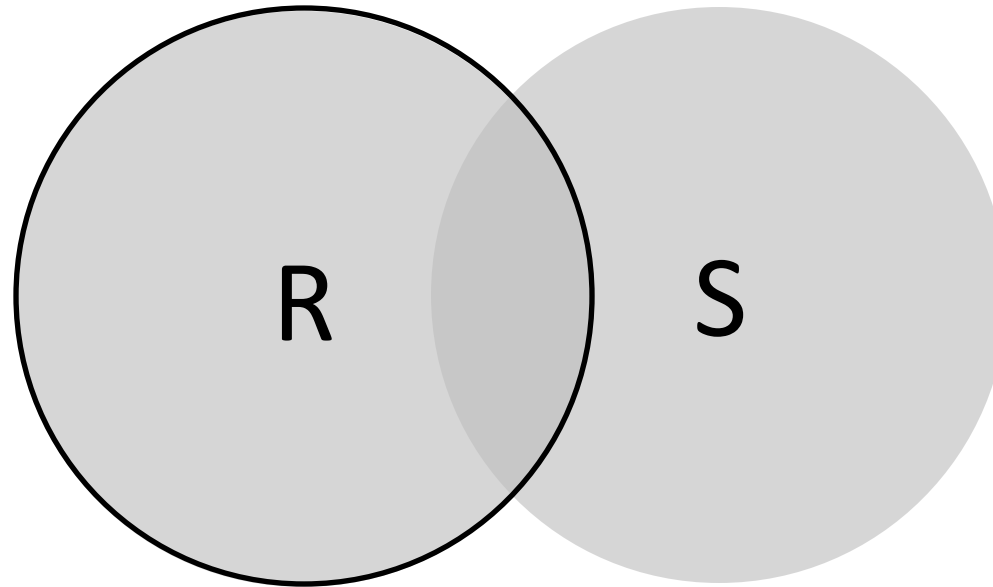Q: How to express it using basic operators?
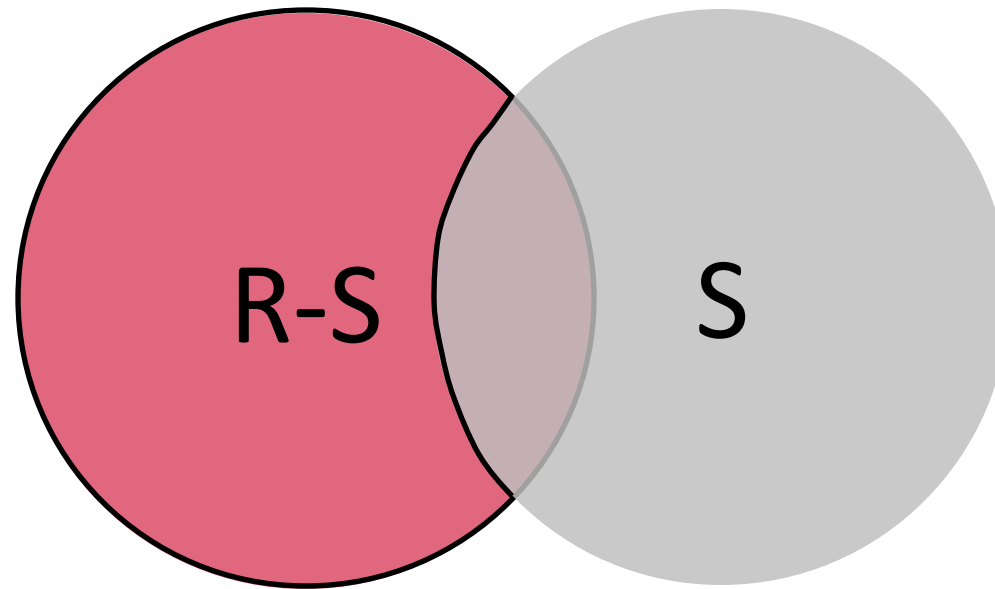
$$R \cap S = R - (R - S)$$

Intersection

Intersection

R   S

R

Intersection

R – S

Intersection

R

R-S   R∩S   S

R – (R – S)

Intersection

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S₁

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S₂

sid	sname	rating	age
31	lubber	8	55.5
58	rusty	10	35.0

S₁ ∩ S₂

Relational Algebra

Relational Query Languages

Selection & Projection

Union, Set Difference & Intersection

Cross product & Joins

Examples

Division

Multisets

Units

Cross-Product

$S_1 \times R_1$: each row of S_1 paired with each row of R_1

how many rows in the result?

result schema has one field per field of S_1 and R_1, with field names "inherited" (if possible)

    may have a naming conflict:

    both S_1 and R_1 have a field with the same name

    in this case, can use the *renaming operator*:

$$\rho(C(1 \rightarrow sid_1, 5 \rightarrow sid_2), S_1 \times R_1)$$

Cross Product Example

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S₁

sid	bid	day
22	101	10/10/16
58	103	11/12/16

R₁

S₁ × R₁ =

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/16
22	dustin	7	45.0	58	103	11/12/16
31	lubber	8	55.5	22	101	10/10/16
31	lubber	8	55.5	58	103	11/12/16
58	rusty	10	35.0	22	101	10/10/16
58	rusty	10	35.0	58	103	11/12/16

Compound Operator: Join

Joins are <u>compound operators</u> : $\times$, σ, (sometimes) π

frequent type is "*natural join*" (often called "join")

R ⋈ S <u>conceptually</u> is:

compute R×S

select rows <u>where</u> attributes in both **R**, **S** have equal values

project all unique attributes and one copy of the common ones

Note: Usually done much more efficiently than this

Useful for putting ***normalized*** relations back together

Natural Join Example

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S_1

sid	bid	day
22	101	10/10/16
58	103	11/12/16

R_1

$S_1 \bowtie R_1 =$

sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/16
58	rusty	10	35.0	103	11/12/16

Natural Join Example

(1) $S_1 \times R_1 =$

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/16
22	dustin	7	45.0	58	103	11/12/16
31	lubber	8	55.5	22	101	10/10/16
31	lubber	8	55.5	58	103	11/12/16
58	rusty	10	35.0	22	101	10/10/16
58	rusty	10	35.0	58	103	11/12/16

Natural Join Example

(1) $S_1 \times R_1 =$

(2) σ

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/16
~~22~~	~~dustin~~	~~7~~	~~45.0~~	~~58~~	~~103~~	~~11/12/16~~
~~31~~	~~lubber~~	~~8~~	~~55.5~~	~~22~~	~~101~~	~~10/10/16~~
~~31~~	~~lubber~~	~~8~~	~~55.5~~	~~58~~	~~103~~	~~11/12/16~~
~~58~~	~~rusty~~	~~10~~	~~35.0~~	~~22~~	~~101~~	~~10/10/16~~
58	rusty	10	35.0	58	103	11/12/16

Natural Join Example

(1) $S_1 \times R_1 =$

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/16
~~22~~	~~dustin~~	~~7~~	~~45.0~~	~~58~~	~~103~~	~~11/12/16~~
~~31~~	~~lubber~~	~~8~~	~~55.5~~	~~22~~	~~101~~	~~10/10/16~~
~~31~~	~~lubber~~	~~8~~	~~55.5~~	~~58~~	~~103~~	~~11/12/16~~
~~58~~	~~rusty~~	~~10~~	~~35.0~~	~~22~~	~~101~~	~~10/10/16~~
58	rusty	10	35.0	58	103	11/12/16

(2) σ

(3) π

$S_1 \bowtie R_1 =$

sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/16
58	rusty	10	35.0	103	11/12/16

Other Types of Joins

condition join (or "theta-join")

$$R\bowtie_c S=\sigma_c(R\times S)$$

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	58	103	11/12/16
31	lubber	8	55.5	58	103	11/12/16

$$S\bowtie_{S.sid<R.sid} R$$

result schema <u>same</u> as that of cross-product

may have <u>fewer tuples</u> than cross-product

Equi-Join: Special case: condition *c* contains only conjunction of *equalities*.

Relational Algebra

Relational Query Languages

Selection & Projection

Union, Set Difference & Intersection

Cross product & Joins

Examples

Division

Multisets

Units

Examples

Reserves

sid	bid	day
22	101	10/10/16
58	103	11/12/16

Sailors

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

Boats

bid	bname	color
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green
104	Marine	Red

Reserves (sid, bid, day)   **Sailors** (sid, sname, rating, age)

Boats (bid, bname, color)

Find names of sailors who have reserved boat #103

Solution 1: $\pi_{sname}((\sigma_{bid=103} \text{Re}serves) \bowtie Sailors)$

Solution 2: $\rho_{sname}(S_{bid=103}(\text{Re}serves \bowtie Sailors))$

another solution?

Reserves (sid, bid, day)   **Sailors** (sid, sname, rating, age)

Boats (bid, bname, color)

Find names of sailors who have reserved a red boat

boat color only available in Boats; need an <u>extra</u> join:

$$\pi_{sname}((\sigma_{color='red'} Boats) \bowtie \mathrm{Re}serves \bowtie Sailors)$$

a *<u>more efficient</u>* solution: why more efficient?

$$\pi_{sname}(\pi_{sid}((\pi_{bid}\sigma_{color='red'} Boats) \bowtie \mathrm{Re}s) \bowtie Sailors)$$

a *<u>query optimizer</u> can find this given the first solution!*

Reserves (sid, bid, day)　　　　**Sailors** (sid, sname, rating, age)

Boats (bid, bname, color)

Find sailors who have reserved a red or a green boat

identify <u>all red or green</u> boats first

$$\rho \ (Tempboats, (\sigma_{color='red' \vee color='green'} \ Boats))$$

then <u>find sailors who have reserved one of these</u> boats:

$$\pi_{sname}(Tempboats \bowtie Reserves \bowtie Sailors)$$

Reserves (sid, bid, day)   **Sailors** (sid, sname, rating, age)

Boats (bid, bname, color)

Find sailors who have reserved a red <u>and</u> a green boat

Previous approach will not work! **Why?**

identify sailors who have reserved <u>red</u> boats

$$\rho \ (Tempred, \ \pi_{sid}((\sigma_{color='red'} \ Boats) \bowtie \text{Re} serves))$$

sailors who have reserved <u>green</u> boats

$$\rho \ (Tempgreen, \ \pi_{sid}((\sigma_{color='green'} \ Boats) \bowtie \text{Re} serves))$$

then find the intersection *(sid* is a key for Sailors)

$$\rho_{sname}((Tempred \subsetneq Tempgreen) \bowtie Sailors)$$

More examples – Your turn!

1. Find (the name of) all sailors whose rating is above 9

2. Find all sailors who reserved a boat prior to November 1, 2016

3. Find (the names of) all boats that have been reserved at least once

4. Find all pairs of sailors with the same rating

5. Find all pairs of sailors in which the <u>older</u> sailor has a <u>lower</u> rating

Reserves (sid, bid, day)   **Sailors** (sid, sname, rating, age)

Boats (bid, bname, color)

(1) Find (the name of) all sailors whose rating is above 9

$$\rho_{sname}(\sigma_{rating>9}(Sailors))$$

Reserves (sid, bid, day)   **Sailors** (sid, sname, rating, age)

Boats (bid, bname, color)

(2) Find all sailors who reserved a boat prior to November 1, 2016

$$\pi_{sname}(Sailors \bowtie \sigma_{day<'11/1/16'}(Reserves))$$

Reserves (sid, bid, day)   **Sailors** (sid, sname, rating, age)

Boats (bid, bname, color)

(3) Find (the names of) all boats that have been reserved at least once

$$\rho_{bname}(Boats \bowtie Reserves)$$

Reserves (sid, bid, day)   **Sailors** (sid, sname, rating, age)

Boats (bid, bname, color)

(4) Find all pairs of sailors with the same rating

$$\rho(S_1(1 \longrightarrow sid_1, 2 \longrightarrow sname_1, 3 \longrightarrow rating_1, 4 \longrightarrow age_1), Sailors)$$
$$\rho(S_2(1 \longrightarrow sid_2, 2 \longrightarrow sname_2, 3 \longrightarrow rating_2, 4 \longrightarrow age_2), Sailors)$$

$$\pi_{sname_1, sname_2}(S_1 \bowtie_{rating_1 = rating_2 \wedge sid_1 \neq sid_2} S_2)$$

is this ok?

$sid_1 < sid_2$

Reserves (sid, bid, day)   **Sailors** (sid, sname, rating, age)

Boats (bid, bname, color)

(5) Find all pairs of sailors in which the older sailor has a lower rating

$$\pi_{sname_1,sname_2}(S_1 \bowtie_{rating_1 < rating_2 \wedge age_1 > age_2} S_2)$$

Relational Algebra

Relational Query Languages

Selection & Projection

Union, Set Difference & Intersection

Cross product & Joins

Examples

Division

Multisets

Units

<u>Last</u> Compound Operator: Division

useful for expressing "*<u>for all</u>*" queries like:
"find sids of sailors who have reserved <u style="color:red">all</u> boats"

for A/B attributes of B are subset of attributes of A

may need to "project" to make this happen.

e.g., let *A* have 2 fields, *x and y* ; *B* have only field *y* :

$$A/B = \left\{ \langle x \rangle \big|^{\shortparallel} \ \langle y \rangle \ \hat{1} \ \ B(\$\langle x,y \rangle \ \hat{1} \ \ A) \right\}$$

A/B contains all x tuples such that for <u>every</u> y tuple in B, there is an xy tuple in A

Examples of Division A/B

sno	pno
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

A

pno
p2

B1

sno
s1
s2
s3
s4

A/B1

pno
p2
p4

B2

pno
p1
p2
p4

B3

Examples of Division A/B

sno	pno
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

A

pno
p2

B1

sno
s1
s2
s3
s4

A/B1

pno
p2
p4

B2

sno
s1
s4

A/B2

pno
p1
p2
p4

B3

Examples of Division A/B

sno	pno
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

A

pno
p2

B1

pno
p2
p4

B2

pno
p1
p2
p4

B3

sno
s1
s2
s3
s4

A/B1

sno
s1
s4

A/B2

sno
s1

A/B3

Expressing A/B Using Basic Operators

division is not essential op; just a shorthand

(true for joins, but *so common* that are implemented specially)

Idea: For *A/B*, compute all *x* values that are not "disqualified" by some *y* value in *B*

Remember:
A: <x,y>
B: <y>

x value is *disqualified* if by <u>attaching *y*</u> value from *B*,
we obtain <u>an *xy* tuple that is not in *A*</u>

only x part
all possible x from A   combined with B

Disqualified *x* values: $\pi_x((\pi_x(A) \times B) - A)$

except the (x,y) pairs that exist already

A/B: $\pi_x(A) - $ Disqualified *x* values

Expressing A/B: $\rho_{sno}(A) - \rho_{sno}((\rho_{sno}(A) \times B) - A)$

sno	pno
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

A

sno	pno
s1	p1
s1	p2
s1	p4
s2	p1
s2	p2
s2	p4
s3	p1
s3	p2
s3	p4
s4	p1
s4	p2
s4	p4

$T1 = \rho_{sno}(A) \times B$

$\cup$

sno
s1
s2
s3
s4

pno
p1
p2
p4

B

Expressing A/B: $\rho_{sno}(A) - \rho_{sno}((\rho_{sno}(A) \times B) - A)$

sno	pno
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

A

sno	pno
~~s1~~	~~p1~~
~~s1~~	~~p2~~
~~s1~~	~~p4~~
~~s2~~	~~p1~~
~~s2~~	~~p2~~
s2	p4
s3	p1
~~s3~~	~~p2~~
s3	p4
s4	p1
~~s4~~	~~p2~~
~~s4~~	~~p4~~

$T1 = \rho_{sno}(A) \times B$

sno	pno
s2	p4
~~s3~~	~~p1~~
s3	p4
s4	p1

$T1 - A$

pno
p1
p2
p4

B

sno
s2
s3
s4

$T2 = \rho_{sno}(T1 - A)$

Expressing A/B: $\rho_{sno}(A) - \rho_{sno}((\rho_{sno}(A) \times B) - A)$

sno	pno
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

A

sno	pno
~~s1~~	~~p1~~
~~s1~~	~~p2~~
~~s1~~	~~p4~~
~~s2~~	~~p1~~
~~s2~~	~~p2~~
s2	p4
s3	p1
~~s3~~	~~p2~~
s3	p4
s4	p1
~~s4~~	~~p2~~
~~s4~~	~~p4~~

$T1 = \rho_{sno}(A) \times B$

sno	pno
s2	p4
~~s3~~	~~p1~~
s3	p4
s4	p1

$T1 - A$

pno
p1
p2
p4

B

sno
s1
s2
s3
s4

−

sno
s2
s3
s4

$T2 = \rho_{sno}(T1-A)$

=

sno
s1

$A/B = \pi_{sno}(A) - T2$

Reserves (sid, bid, day)   **Sailors** (sid, sname, rating, age)

Boats (bid, bname, color)

Find the names of sailors who have reserved all boats

use division; schemas of the input relations to / must be carefully chosen (**why?**)

$$\rho \ (Tempsids, (\pi_{sid,bid} \text{Re}serves) \ / \ (\pi_{bid} Boats))$$

$$\pi_{sname} (Tempsids \bowtie Sailors)$$

To find sailors who have reserved all "Interlake" boats:

$$..... \ / \ \pi_{bid} (\sigma_{bname = 'Interlake'} Boats)$$

Reserves (sid, bid, day)   **Sailors** (sid, sname, rating, age)

Boats (bid, bname, color)

Find the names of sailors who have reserved all boats

use division; schemas of the input relations to / must
be carefully chosen (**why?**)

$$\rho \ (Tempsids, \ (\pi_{sid,bid} \mathrm{Re}serves) \ / \ (\pi_{bid} Boats))$$

$$\pi_{sname} (Tempsids \bowtie Sailors)$$

what if we divided *Reserves* / π_{bid}*(Boats)* ?

this would return the pairs of (sid,date) that have a value for evey boat,
*i.e., the <u>sids</u> that <u>rented</u> **every** <u>boat</u>, **every** <u>day</u> they made any reservation!!!! Not so useful!*

Relational Algebra

Relational Query Languages

Selection & Projection

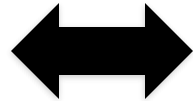Union, Set Difference & Intersection

Cross product & Joins

Examples

Division

Multisets

Units

SQL uses Multisets

Multiset X

Tuple
(1, a)
(1, a)
(1, b)
(2, c)
(2, c)
(2, c)
(1, d)
(1, d)

Equivalent Representations of a Multiset

Multiset X

Tuple	$\lambda(X)$
(1, a)	2
(1, b)	1
(2, c)	3
(1, d)	2

$\lambda(X)$= "Count of tuple in X" (Items not listed have implicit count 0)

Note: In a set all counts are {0,1}.

60

Generalizing Set Operations to Multiset Operations

Multiset X

Tuple	$\lambda(X)$
(1, a)	2
(1, b)	0
(2, c)	3
(1, d)	0

∩

Multiset Y

Tuple	$\lambda(Y)$
(1, a)	5
(1, b)	1
(2, c)	2
(1, d)	2

=

Multiset Z

Generalizing Set Operations to Multiset Operations

Multiset X

Tuple	$\lambda(X)$
(1, a)	2
(1, b)	0
(2, c)	3
(1, d)	0

∩

Multiset Y

Tuple	$\lambda(Y)$
(1, a)	5
(1, b)	1
(2, c)	2
(1, d)	2

=

Multiset Z

Tuple	$\lambda(Z)$
(1, a)	2
(1, b)	0
(2, c)	2
(1, d)	0

$$\lambda(Z)=min(\lambda(X),\lambda(Y))$$

For sets, this is **intersection**

Generalizing Set Operations to Multiset Operations

Multiset X

Tuple	$\lambda(X)$
(1, a)	2
(1, b)	0
(2, c)	3
(1, d)	0

$\cup$

Multiset Y

Tuple	$\lambda(Y)$
(1, a)	5
(1, b)	1
(2, c)	2
(1, d)	2

=

Multiset Z

Generalizing Set Operations to Multiset Operations

Multiset X

Tuple	$\lambda(X)$
(1, a)	2
(1, b)	0
(2, c)	3
(1, d)	0

$\cup$

Multiset Y

Tuple	$\lambda(Y)$
(1, a)	5
(1, b)	1
(2, c)	2
(1, d)	2

$=$

Multiset Z

Tuple	$\lambda(Z)$
(1, a)	7
(1, b)	1
(2, c)	5
(1, d)	2

$\lambda(Z) = \lambda(X) + \lambda(Y)$

For sets,
this is **union**

38

Operations on Multisets (bags)

All RA operations need to be defined carefully on bags

- $\sigma_C(R)$: **preserve** the number of occurrences

- $\Pi_A(R)$: **no duplicate elimination**

- Cross-product, join: **no duplicate elimination**

This is important: relational engines work on multisets, not sets!

RA has Limitations !

Cannot compute **transitive closure**

Name1	Name2	Relationship
Fred	Mary	Father
Mary	Joe	Cousin
Mary	Bill	Spouse
Nancy	Lou	Sister

"Find all direct and indirect relatives of Fred"
→ Cannot express in RA !!!
Need to write Java program, use a graph engine, or modern SQL…