

# L4

# Relational Algebra

Eugene Wu

# Supplemental Materials

## Helpful References

[https://en.wikipedia.org/wiki/Relational\\_algebra](https://en.wikipedia.org/wiki/Relational_algebra)

# Overview

Last time, learned about  
pre-relational models  
an informal introduction to relational model  
an introduction to the SQL query language.

Learn about formal relational query languages  
Relational Algebra (algebra: perform operations)  
Relational Calculus (logic: are statements true?)

Keys to understanding SQL and query processing

# Who Cares?

Clean query semantics & rich program analysis

Helps/enables optimization

Opens up rich set of topics

- Materialized views

- Data lineage/provenance

- Query by example

- Distributed query execution

- ...

# What's a Query Language?

Allows manipulation and **retrieval of data** from a database.

Traditionally: QL != programming language

- Doesn't need to be turing complete

- Not designed for computation

- Supports easy, efficient access to large databases

Recent Years

- Scaling to large datasets is a reality

- Query languages are a powerful way to

  - think about data algorithms scale

  - think about asynchronous/parallel programming

# What's a Query Language?

Re-use permitted when acknowledging the original © Daniel Abadi, Shivnath Babu, Fatma Ozcan, and Ippokratis Pandis (2015)

4

## MapReduce is **not** the answer

- MapReduce is a powerful primitive to do many kinds of parallel data processing
- BUT
  - Little control of data flow
  - Fault tolerance guarantees not always necessary
  - Simplicity leads to inefficiencies
  - Does not interface with existing analysis software
  - Industry has existing training in SQL



**SQL interface for Hadoop critical for mass adoption**

SQL-on-Hadoop Tutorial

23/02/2015

Tutorial at VLDB 2015 Abadi et al.  
<http://www.slideshare.net/abadid/sqlonhadoop-tutorial>

# Formal Relational Query Languages

Formal basis for real languages e.g., SQL

## Relational Algebra

Operational, used to represent execution plans

## Relational Calculus

Logical, describes what data users want  
(not operational, fully declarative)

# Prelims

Query is a function over **relation instances**

$$Q(R_1, \dots, R_n) = R_{\text{result}}$$

Schemas of input and output relations are *fixed* and well defined by the query  $Q$ .

Positional vs Named field notation

- Position easier for formal defs

  - one-indexed (not 0-indexed!!!)

- Named more readable

- Both used in SQL



# Prelims

Relation (for this lecture)

Instance is a set of tuples

Schema defines field names and types (domains)

Students(sid int, name text, major text, gpa int)

How are relations different than generic sets ( $\mathbb{R}$ )?

Can assume item structure due to schema

Some algebra operations (x) need to be modified

Will use this later

# Relational Algebra Overview

## Core 5 operations

PROJECT ( $\pi$ )

SELECT ( $\sigma$ )

UNION ( $\cup$ )

SET DIFFERENCE ( $-$ )

CROSSPRODUCT ( $\times$ )

## Additional operations

RENAME ( $\rho$ )

INTERSECT ( $\cap$ )

JOIN ( $\bowtie$ )

DIVIDE ( $/$ )

# Instances Used Today: Library

Students, Reservations

Use positional or named field notation

Fields in query results are inherited from input relations (unless specified)

RI

sid	rid	day
1	101	10/10
2	102	11/11

SI

sid	name	gpa	age
1	eugene	4	20
2	barb	3	21
3	tanya	2	88

S2

sid	name	gpa	age
4	aziz	3.2	21
2	barb	3	21
3	tanya	2	88
5	rusty	3.5	21

# Project

$$\pi_{\langle \text{attr1}, \dots \rangle}(A) = R_{\text{result}}$$

Pick out desired attributes (subset of columns)

Schema is subset of input schema in the projection list

$\pi_{\langle a, b, c \rangle}(A)$  has output schema  $(a, b, c)$  w/ types carried over

# Project

S2

sid	name	gpa	age
4	aziz	3.2	21
2	barb	3	21
3	tanya	2	88
5	rusty	3.5	21

$$\pi_{\text{name,age}}(S2) =$$

name	age
aziz	21
barb	21
tanya	88
rusty	21

# Project

S2

sid	name	gpa	age
4	aziz	3.2	21
2	barb	3	21
3	tanya	2	88
5	rusty	3.5	21

$$\pi_{\text{name,name,age}}(S2) =$$

name	name	age
aziz	aziz	21
barb	barb	21
tanya	tanya	88
rusty	rusty	21

# Project

S2

sid	name	gpa	age
4	aziz	3.2	21
2	barb	3	21
3	tanya	2	88
5	rusty	3.5	21

$$\pi_{\text{age}}(S2) =$$

age
21
88

Where did all the rows go?

Real systems typically don't remove duplicates. Why?

# Select

$$\sigma_{\langle p \rangle}(A) = R_{\text{result}}$$

Select subset of rows that satisfy condition  $p$

Won't have duplicates in result. Why?

Result schema same as input



# Select

S1

sid	name	gpa	age
1	eugene	4	20
2	barb	3	21
3	tanya	2	88

$$\sigma_{\text{age} < 30} (S1) =$$

sid	name	gpa	age
1	eugene	4	20
2	barb	3	21

$$\pi_{\text{name}}(\sigma_{\text{age} < 30} (S1)) =$$

name
eugene
barb

# Commutatively & Associativity

$$A + B = B + A$$

$$A * B = B * A$$

$$A + (B * C) = (B * C) + A$$

$$A + (B + C) = (A + B) + C$$

$$A + (B * C) = (A + B) * C$$

# Commutatively & Associativity

$$A + B = B + A$$

$$A * B = B * A$$


$$A + (B * C) = (B * C) + A$$

$$A + (B + C) = (A + B) + C$$


~~$$A + (B * C) = (A + B) * C$$~~


# Commutatively

$$\pi_{\text{age}}(\sigma_{\text{age} < 30} (SI))$$

$\sigma_{\text{age} < 30}$  

sid	name	gpa	age
1	eugene	4	20
2	barb	3	21
3	tanya	2	88





sid	name	gpa	age
1	eugene	4	20
2	barb	3	21

# Commutatively

$\pi_{\text{age}}(\sigma_{\text{age} < 30} (SI))$

$\sigma_{\text{age} < 30}$

sid	name	gpa	age
1	eugene	4	20
2	barb	3	21
3	tanya	2	88

$\pi_{\text{age}}$

sid	name	gpa	age
1	eugene	4	20
2	barb	3	21

$=$

age
20
21

# Commutatively

$$\sigma_{\text{age} < 30}(\pi_{\text{age}}(SI))$$

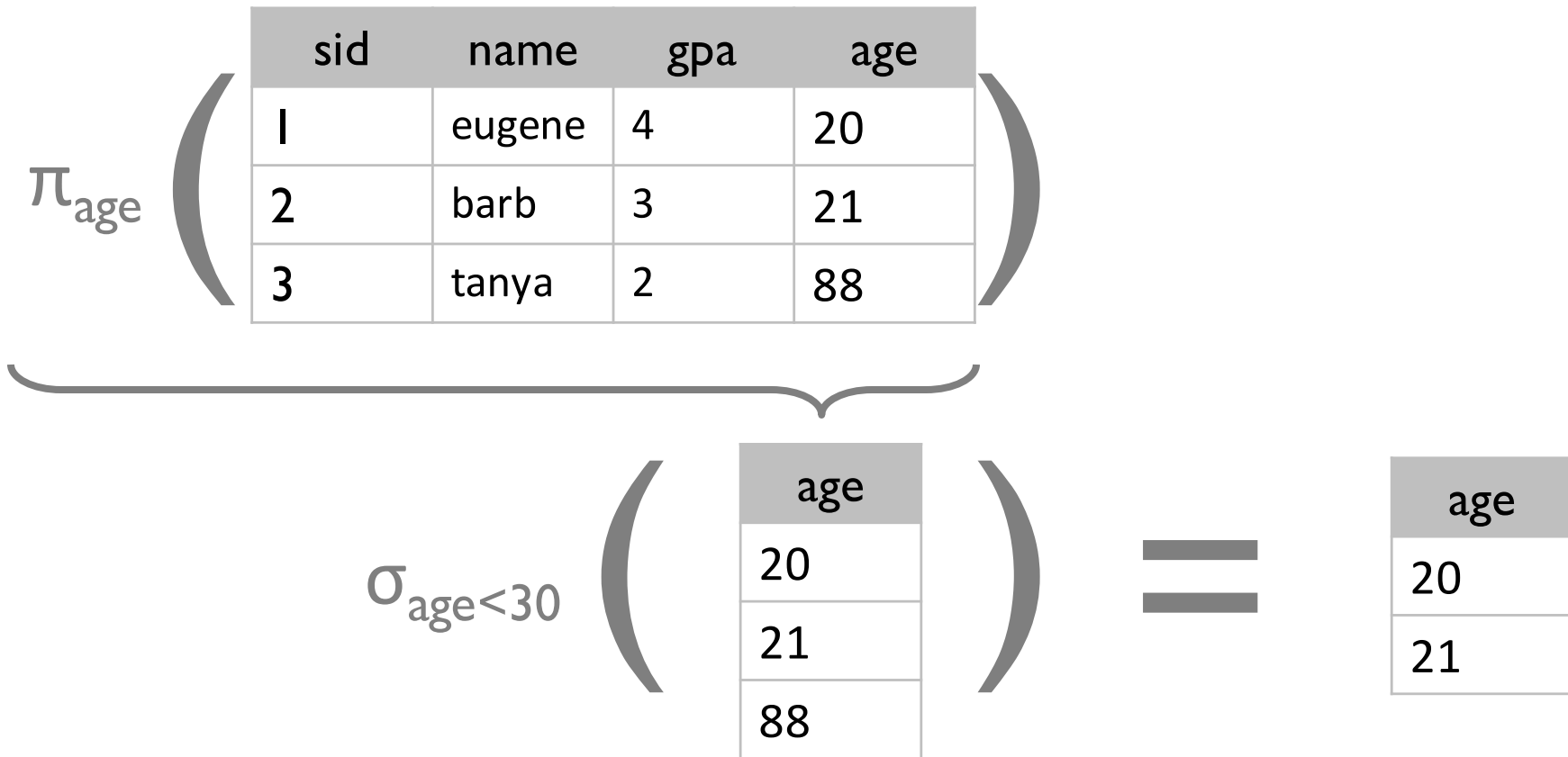
$\pi_{\text{age}}$	sid	name	gpa	age	
	1	eugene	4	20	
	2	barb	3	21	
	3	tanya	2	88	

age
20
21
88

# Commutatively

$$\sigma_{\text{age} < 30}(\pi_{\text{age}}(S))$$



# Commutatively

Does Project and Select commute?

$$\pi_{\text{age}}(\sigma_{\text{age} < 30}(SI)) = \sigma_{\text{age} < 30}(\pi_{\text{age}}(SI))$$

What about

$$\pi_{\text{name}}(\sigma_{\text{age} < 30}(SI))?$$



# Commutatively

Does Project and Select commute?

$$\pi_{\text{age}}(\sigma_{\text{age} < 30}(SI)) = \sigma_{\text{age} < 30}(\pi_{\text{age}}(SI))$$

What about

$$\pi_{\text{name}}(\sigma_{\text{age} < 30}(SI)) \neq \sigma_{\text{age} < 30}(\pi_{\text{name}}(SI))$$

# Commutatively

Does Project and Select commute?

$$\pi_{\text{age}}(\sigma_{\text{age} < 30}(SI)) = \sigma_{\text{age} < 30}(\pi_{\text{age}}(SI))$$

What about

$$\pi_{\text{name}}(\sigma_{\text{age} < 30}(SI)) \neq \sigma_{\text{age} < 30}(\pi_{\text{name, age}}(SI))$$

# Commutatively

Does Project and Select commute?

$$\pi_{\text{age}}(\sigma_{\text{age} < 30}(SI)) = \sigma_{\text{age} < 30}(\pi_{\text{age}}(SI))$$

What about

$$\pi_{\text{name}}(\sigma_{\text{age} < 30}(SI)) = \pi_{\text{name}}(\sigma_{\text{age} < 30}(\pi_{\text{name, age}}(SI)))$$

OK!

# Union, Set-Difference

$$A \text{ op } B = R_{\text{result}}$$

A, B must be *union-compatible*

Same number of fields

Field i in each schema have same type

Result Schema borrowed from first arg (A)

A(big int, poppa int)  $\cup$  B(thug int, life int) = ?

# Union, Set-Difference

$$A \text{ op } B = R_{\text{result}}$$

A, B must be *union-compatible*

Same number of fields

Field i in each schema have same type

Result Schema borrowed from first arg (A)

$A(\text{big int, poppa int}) \cup B(\text{thug int, life int}) =$

$R_{\text{result}}(\text{big int, poppa int})$

# Union, Intersect, Set-Difference

S1

sid	name	gpa	age
1	eugene	4	20
2	barb	3	21
3	tanya	2	88

S2

sid	name	gpa	age
4	aziz	3.2	21
2	barb	3	21
3	tanya	2	88
5	rusty	3.5	21

$S1 \cup S2 =$

sid	name	gpa	age
1	eugene	4	20
4	aziz	3.2	21
5	rusty	3.5	21
3	tanya	2	88
2	barb	3	21

# Union, Intersect, Set-Difference

S1

sid	name	gpa	age
1	eugene	4	20
2	barb	3	21
3	tanya	2	88

S2

sid	name	gpa	age
4	aziz	3.2	21
2	barb	3	21
3	tanya	2	88
5	rusty	3.5	21

$S1 - S2 =$

sid	name	gpa	age
1	eugene	4	20

# Note on Set Difference & Performance

Notice that most operators are monotonic  
increasing size of inputs  $\rightarrow$  outputs grow  
if  $A \supseteq B \rightarrow Q(A, T) \supseteq Q(B, T)$   
can compute *incrementally*

Set Difference is *not monotonic*

if  $A \supseteq B \rightarrow T - A \subseteq T - B$   
e.g.,  $5 > 1 \rightarrow 9 - 5 < 9 - 1$

Set difference is *blocking*:

For  $T - S$ , must wait for all  $S$  tuples before any results



# Cross-Product

$$A(a_1, \dots, a_n) \times B(a_{n+1}, \dots, a_m) = R_{\text{result}}(a_1, \dots, a_m)$$

Each row of A paired with each row of B

Result schema concatenates A and B's fields, inherit if possible

Conflict: students and reservations have *sid* field

Different than mathematical “X” by flattening results:

$$\text{math } A \times B = \{ (a, b) \mid a \in A \wedge b \in B \}$$

$$\text{e.g., } \{1, 2\} \times \{3, 4\} = \{ (1, 3), (1, 4), (2, 3), (2, 4) \}$$

what is  $\{1, 2\} \times \{3, 4\} \times \{5, 6\}$ ?

# Cross-Product

SI

sid	name	gpa	age
1	eugene	4	20
2	barb	3	21
3	tanya	2	88

RI

sid	rid	day
1	101	10/10
2	102	11/11

SI x RI =

(sid)	name	gpa	age	(sid)	rid	day
1	eugene	4	20	1	101	10/10
2	barb	3	21	1	101	10/10
3	tanya	2	88	1	101	10/10
1	eugene	4	20	2	102	11/11
2	barb	3	21	2	102	11/11
3	tanya	2	88	2	102	11/11

# Rename

$p(<\text{new\_name}>(<\text{mappings}>), Q)$

Explicitly defines/changes field names of schema

$p(C(I \rightarrow \text{sid1}, 5 \rightarrow \text{sid2}), S1 \times R1)$

C =

sid1	name	gpa	age	sid2	rid	day
1	eugene	4	20	1	101	10/10
2	barb	3	21	1	101	10/10
3	tanya	2	88	1	101	10/10
1	eugene	4	20	2	102	11/11
2	barb	3	21	2	102	11/11
3	tanya	2	88	2	102	11/11

Project

$$\pi\left(\begin{array}{|c|c|} \hline \text{blue} & \text{orange} \\ \hline \end{array}\right) = \begin{array}{|c|} \hline \text{blue} \\ \hline \end{array}$$

Select

$$\sigma\left(\begin{array}{|c|} \hline \text{blue} \\ \hline \text{orange} \\ \hline \end{array}\right) = \begin{array}{|c|} \hline \text{blue} \\ \hline \end{array}$$

Cross product

$$\begin{array}{|c|} \hline \text{blue} \\ \hline \end{array} \times \begin{array}{|c|} \hline \text{orange} \\ \hline \end{array} = \begin{array}{|c|c|} \hline \text{blue} & \text{orange} \\ \hline \end{array}$$

Difference

$$\begin{array}{|c|} \hline \text{orange} \\ \hline \text{blue} \\ \hline \end{array} - \begin{array}{|c|} \hline \text{orange} \\ \hline \end{array} = \begin{array}{|c|} \hline \text{blue} \\ \hline \end{array}$$

Union

$$\begin{array}{|c|} \hline \text{blue} \\ \hline \end{array} \cup \begin{array}{|c|} \hline \text{orange} \\ \hline \end{array} = \begin{array}{|c|} \hline \text{blue} \\ \hline \text{orange} \\ \hline \end{array}$$

Intersect

$$\begin{array}{|c|} \hline \text{orange} \\ \hline \text{blue} \\ \hline \end{array} \cap \begin{array}{|c|} \hline \text{purple} \\ \hline \text{orange} \\ \hline \end{array} = \begin{array}{|c|} \hline \text{orange} \\ \hline \end{array}$$

# Compound/Convenience Operators

INTERSECT ( $\cap$ )

JOIN ( $\bowtie$ )

DIVIDE ( $/$ )

# Intersect

$$A \cap B = R_{\text{result}}$$

A, B must be *union-compatible*

# Intersect

S1

sid	name	gpa	age
1	eugene	4	20
2	barb	3	21
3	tanya	2	88

S2

sid	name	gpa	age
4	aziz	3.2	21
2	barb	3	21
3	tanya	2	88
5	rusty	3.5	21

$S1 \cap S2 =$

sid	name	gpa	age
2	barb	3	21
3	tanya	2	88

# Intersect

$$A \cap B = R_{\text{result}}$$

A, B must be *union-compatible*

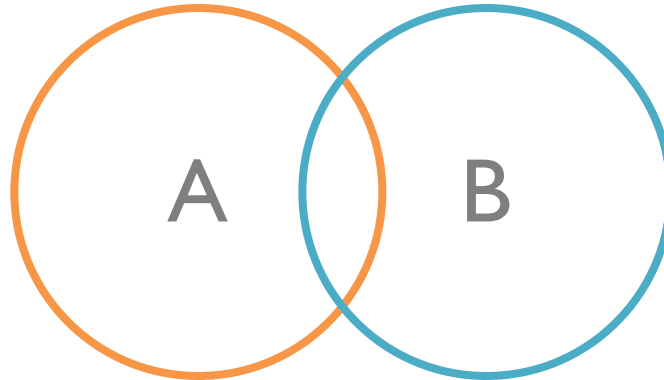
Can we express using core operators?

$$A \cap B = ?$$



# Intersect

$$A \cap B = R_{\text{result}}$$

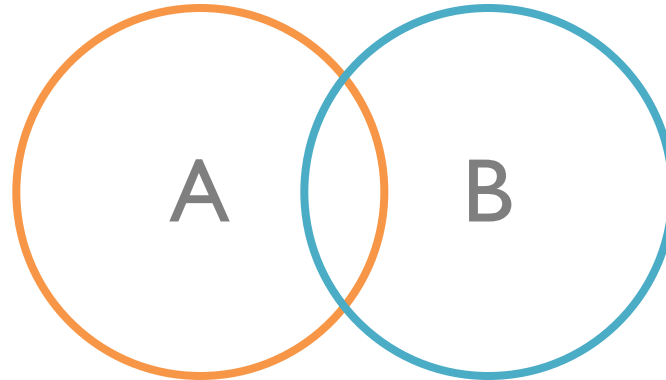


Can we express using core operators?

$A \cap B = A - ?$  (think venn diagram)

# Intersect

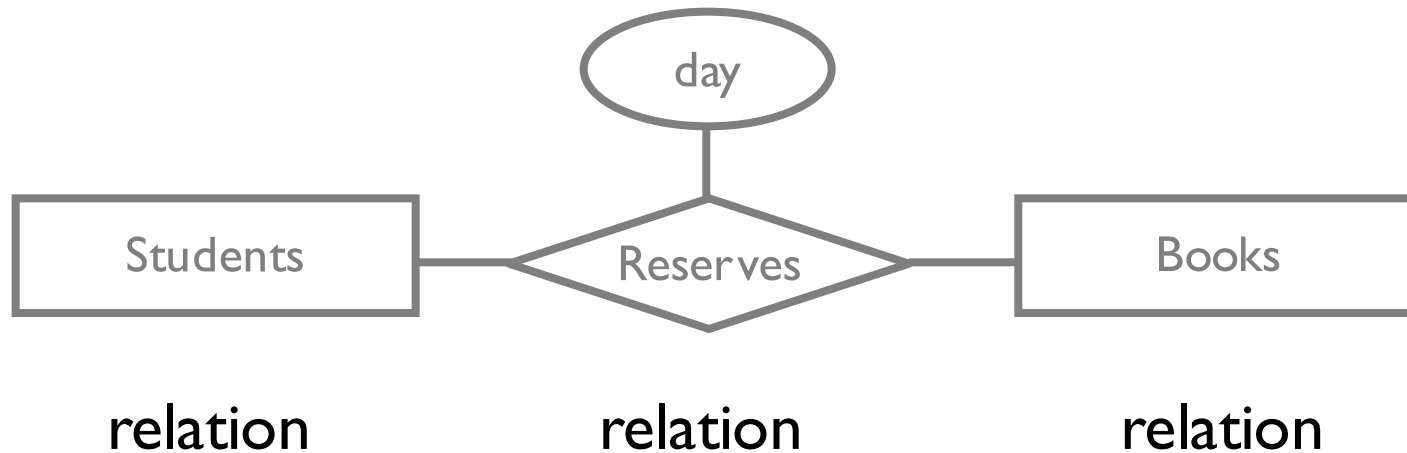
$$A \cap B = R_{\text{result}}$$



Can we express using core operators?

$$A \cap B = A - (A - B)$$

# Joins (high level)



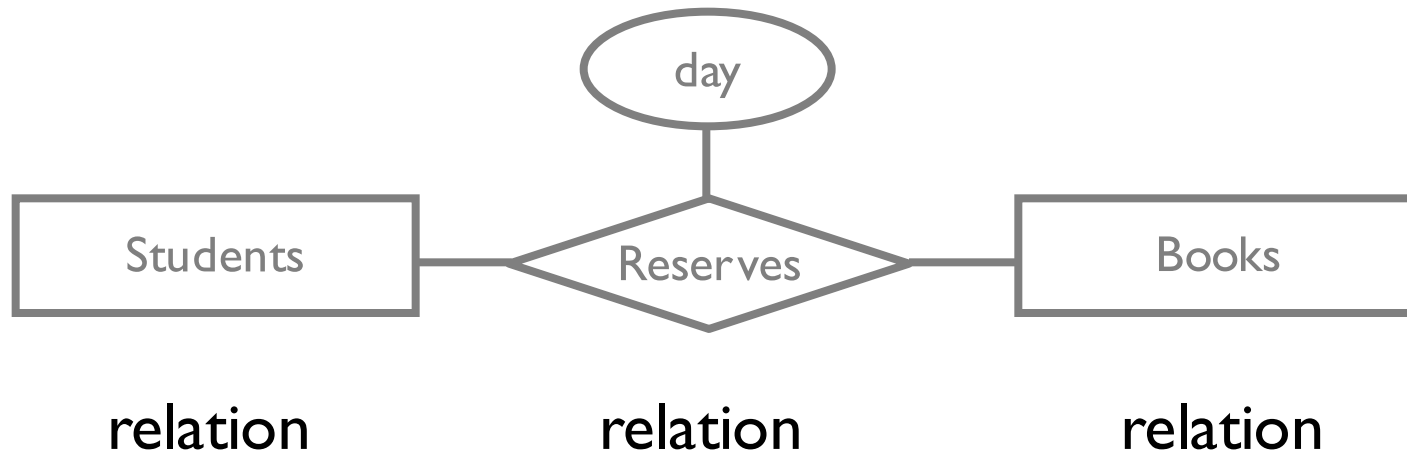
What if you want to query across all three tables?

e.g., all names of students that reserved “The Purple Crayon”

Need to combine these tables

Cross product? But that ignores foreign key references

# Joins (high level)



SI

sid	name	gpa	age
1	eugene	4	20
2	barb	3	21
3	tanya	2	88

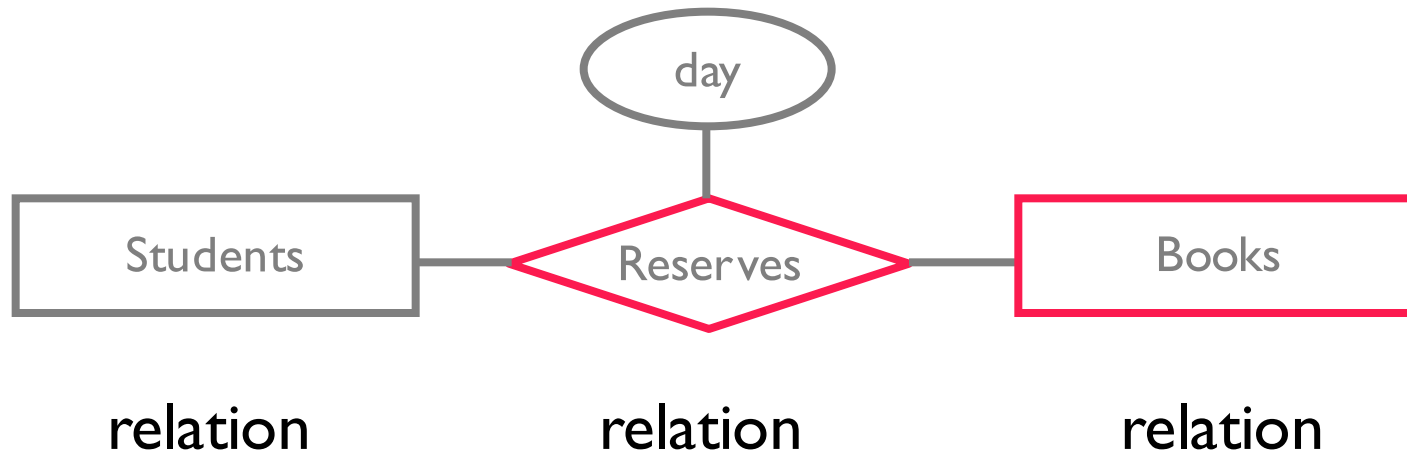
RI

sid	rid	day
1	101	10/10
2	102	11/11

BI

rid	name
101	The Purple Crayon
102	1984

# Joins (high level)



SI

sid	name	gpa	age
1	eugene	4	20
2	barb	3	21
3	tanya	2	88

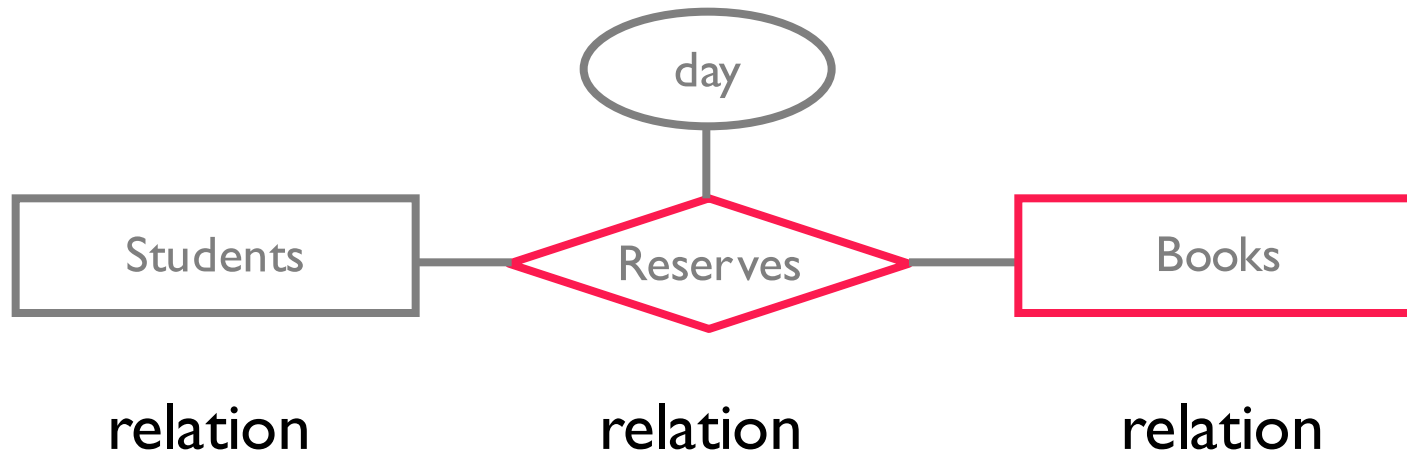
RI

sid	rid	day
1	101	10/10
2	102	11/11

BI

rid	name
101	The Purple Crayon
102	1984

# Joins (high level)



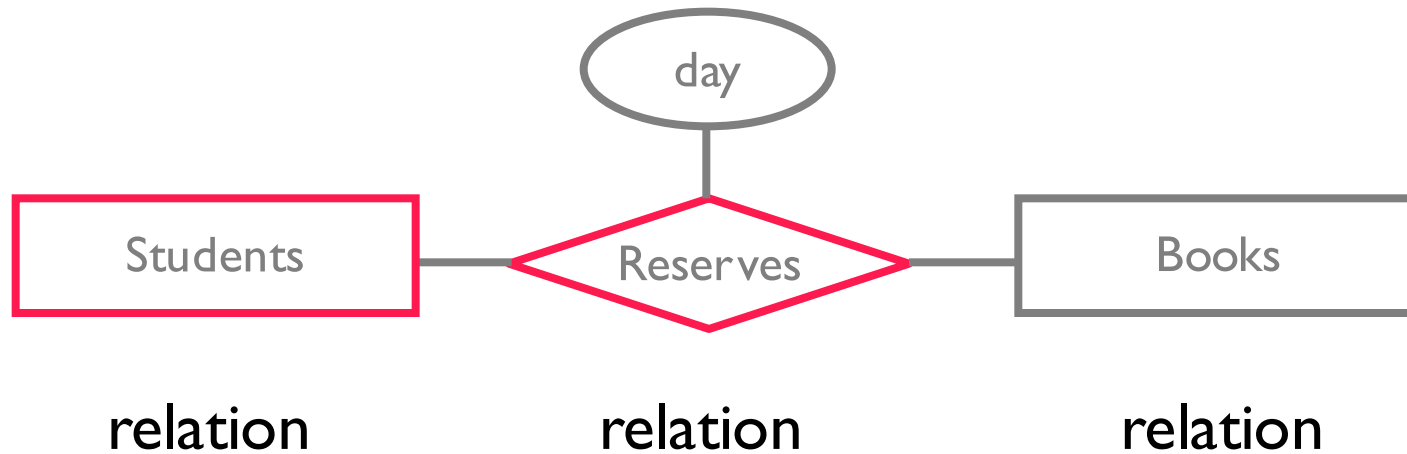
SI

sid	name	gpa	age
1	eugene	4	20
2	barb	3	21
3	tanya	2	88

RBI

sid	(rid)	day	(rid)	name
1	101	10/10	101	The Purple Crayon
2	102	11/11	102	1984

# Joins (high level)



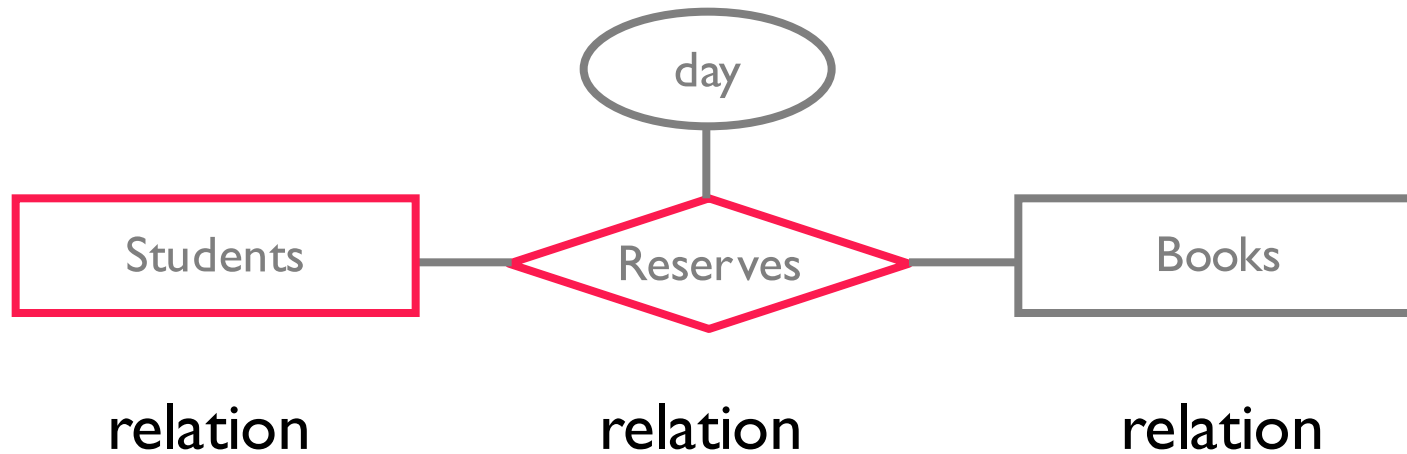
SI

sid	name	gpa	age
1	eugene	4	20
2	barb	3	21
3	tanya	2	88

RBI

sid	(rid)	day	(rid)	name
1	101	10/10	101	The Purple Crayon
2	102	11/11	102	1984

# Joins (high level)



SRBI

(sid)	(name)	gpa	age	(sid)	(rid)	day	(rid)	(name)
1	eugene	4	20	1	101	10/10	101	The Purple Crayon
2	barb	3	21	2	102	11/11	102	1984



# theta ( $\theta$ ) Join

$$A \bowtie_c B = \sigma_c(A \times B)$$

Most general form

Result schema same as cross product

Often *far* more efficient to compute than cross product

Commutative

$$(A \bowtie_c B) \bowtie_c C = A \bowtie_c (B \bowtie_c C)$$

# theta ( $\theta$ ) Join

SI

sid	name	gpa	age
1	eugene	4	20
2	barb	3	21
3	tanya	2	88

RI

sid	rid	day
1	101	10/10
2	102	11/11

$$SI \bowtie_{SI.sid \leq RI.sid} RI =$$

$$\sigma_{SI.sid \leq RI.sid} (SI \times RI) =$$

(sid)	name	gpa	age	(sid)	rid	day
1	eugene	4	20	1	101	10/10
2	barb	3	21	1	101	10/10
3	tanya	2	88	1	101	10/10
1	eugene	4	20	2	102	11/11
2	barb	3	21	2	102	11/11
3	tanya	2	88	2	102	11/11

# theta ( $\theta$ ) Join

SI

sid	name	gpa	age
1	eugene	4	20
2	barb	3	21
3	tanya	2	88

RI

sid	rid	day
1	101	10/10
2	102	11/11

$$SI \bowtie_{SI.sid \leq RI.sid} RI =$$

$$\sigma_{SI.sid \leq RI.sid} (SI \times RI) =$$

(sid)	name	gpa	age	(sid)	rid	day
1	eugene	4	20	1	101	10/10
2	barb	3	21	1	101	10/10
3	tanya	2	88	1	101	10/10
1	eugene	4	20	2	102	11/11
2	barb	3	21	2	102	11/11
3	tanya	2	88	2	102	11/11

# theta ( $\theta$ ) Join

SI

sid	name	gpa	age
1	eugene	4	20
2	barb	3	21
3	tanya	2	88

RI

sid	rid	day
1	101	10/10
2	102	11/11

$$SI \bowtie_{SI.sid \leq RI.sid} RI =$$

$$\sigma_{SI.sid \leq RI.sid}(SI \times RI) =$$

(sid)	name	gpa	age	(sid)	rid	day
1	eugene	4	20	1	101	10/10
1	eugene	4	20	2	102	11/11
2	barb	3	21	2	102	11/11

# Equi-Join

$$A \bowtie_{\text{attr}} B = \pi_{\text{all attrs except B.attr}}(A \bowtie_{A.\text{attr} = B.\text{attr}} B)$$

Special case where the condition is attribute equality

Result schema only keeps *one copy* of equality fields

Natural Join ( $A \bowtie B$ ):

Equijoin on *all* shared fields (fields w/ same name)

# Equi-Join

SI

sid	name	gpa	age
1	eugene	4	20
2	barb	3	21
3	tanya	2	88

RI

sid	rid	day
1	101	10/10
2	102	11/11

$SI \bowtie_{sid} RI =$

sid	name	gpa	age	rid	day
1	eugene	4	20	101	10/10
2	barb	3	21	102	11/11

# Division

Let us have relations  $A(x, y)$ ,  $B(y)$

$$A/B = \{ \langle x \rangle \mid \forall y \in B \langle x, y \rangle \in A \}$$

*Find all students that have reserved all books*

$A/B =$  all  $x$  (students) s.t. for every  $y$  (reservation),  $\langle x, y \rangle \in A$

Good to ponder, not supported in most systems (why?)

Generalization

$y$  can be a list of fields in  $B$

$x \cup y$  is fields in  $A$

# Examples

A

sid	rid
1	1
1	2
1	3
1	4
2	1
2	2
3	2
4	2
4	4

R1

rid
2

A/R1

R2

rid
2
4

A/R2

R3

rid
1
2
4

A/R3

Which sid values have relationship with all vals in Rx?



# Examples

A

sid	rid
1	1
1	2
1	3
1	4
2	1
2	2
3	2
4	2
4	4

R1

rid
2

sid
1
2
3
4

A/R1

R2

rid
2
4

A/R2

R3

rid
1
2
4

A/R3

Which sid values have relationship with all vals in Rx?

# Examples

A

sid	rid
1	1
1	2
1	3
1	4
2	1
2	2
3	2
4	2
4	4

R1

rid
2

sid
1
2
3
4

A/R1

R2

rid
2
4

sid
1
4

A/R2

R3

rid
1
2
4

A/R3

Which sid values have relationship with all vals in Rx?

# Examples

A

sid	rid
1	1
1	2
1	3
1	4
2	1
2	2
3	2
4	2
4	4

R1

rid
2

sid
1
2
3
4

A/R1

R2

rid
2
4

sid
1
4

A/R2

R3

rid
1
2
4

sid
1

A/R3

Which sid values have relationship with all vals in Rx?

# Is $A/B$ a Fundamental Operation?

No. Shorthand like Joins

joins natively supported: ubiquitous, can be optimized

**Hint:** Find all  $x$ s not 'disqualified' by some  $y$  in  $B$ .

$A(x, y), B(y)$

$x$  value is *disqualified* if

1. by attaching  $y$  value from  $B$  (e.g., create  $\langle x, y \rangle$ )
2. we obtain an  $\langle x, y \rangle$  that is not in  $A$ .

A

sid	rid
1	1
1	2
1	3
1	4
2	1
2	2
3	2
4	2
4	4

B

rid
2
4

Disqualified =

A/B =

A

sid	rid
1	1
1	2
1	3
1	4
2	1
2	2
3	2
4	2
4	4

B

rid
2
4

 $\pi_{\text{sid}}(A)$ 

sid
1
2
3
4

Disqualified =  
A/B =

A

sid	rid
1	1
1	2
1	3
1	4
2	1
2	2
3	2
4	2
4	4

B

rid
2
4

 $\pi_{\text{sid}}(A)$ 

sid
1
2
3
4

 $\pi_{\text{sid}}(A) \times B$ 

sid	rid
1	2
1	4
2	2
2	4
3	2
3	4
4	2
4	4

Disqualified =  
A/B =

$(\pi_{\text{sid}}(A) \times B)$

A

sid	rid
1	1
1	2
1	3
1	4
2	1
2	2
3	2
4	2
4	4

B

rid
2
4

$\pi_{\text{sid}}(A)$

sid
1
2
3
4

 $\pi_{\text{sid}}(A) \times B$ 

sid	rid
1	2
1	4
2	2
2	4
3	2
3	4
4	2
4	4

 $(\pi_{\text{sid}}(A) \times B) - A$ 

sid	rid
2	4
3	4

Disqualified =  $((\pi_{\text{sid}}(A) \times B) - A)$   
 A/B =



A

sid	rid
1	1
1	2
1	3
1	4
2	1
2	2
3	2
4	2
4	4

B

rid
2
4

 $\pi_{\text{sid}}(A)$ 

sid
1
2
3
4

 $\pi_{\text{sid}}(A) \times B$ 

sid	rid
1	2
1	4
2	2
2	4
3	2
3	4
4	2
4	4

 $(\pi_{\text{sid}}(A) \times B) - A$ 

sid	rid
2	4
3	4

sid

sid
1
4

A/B

$$\text{Disqualified} = \pi_{\text{sid}}((\pi_{\text{sid}}(A) \times B) - A)$$

$$A/B = \pi_{\text{sid}}(A) - \text{Disqualified}$$

# Different Plans, Same Results

Semantic equivalence:  
results are *always* the same

Note that it is independent of the database  
instance!

# Names of students that reserved book 2

$$\pi_{\text{name}}(\sigma_{\text{rid}=2} (R1) \bowtie SI)$$

## Equivalent Queries

$$\begin{aligned} & p(\text{tmp1}, \sigma_{\text{rid}=2} (R1)) \\ & p(\text{tmp2}, \text{tmp1} \bowtie SI) \\ & \pi_{\text{name}}(\text{tmp2}) \end{aligned}$$
$$\pi_{\text{name}}(\sigma_{\text{rid}=2}(R1 \bowtie SI))$$

# Names of students that reserved db books

Book(rid, type)    Reserve(sid, rid)    Student(sid, name)

Need to join DB books with reserve and students

$\sigma_{\text{type}='db'}$  (Book)

# Names of students that reserved db books

Book(rid, type)    Reserve(sid, rid)    Student(sid, name)

Need to join DB books with reserve and students

$\sigma_{\text{type}='db'} (\text{Book}) \bowtie \text{Reserve}$

# Names of students that reserved db books

Book(rid, type)    Reserve(sid, rid)    Student(sid, name)

Need to join DB books with reserve and students

$\sigma_{\text{type}='db'}$  (Book)  $\bowtie$  Reserve  $\bowtie$  Student

# Names of students that reserved db books

Book(rid, type)    Reserve(sid, rid)    Student(sid, name)

Need to join DB books with reserve and students

$\pi_{\text{name}}(\sigma_{\text{type}='db'} (\text{Book}) \bowtie \text{Reserve} \bowtie \text{Student})$

# Names of students that reserved db books

Book(rid, type)    Reserve(sid, rid)    Student(sid, name)

Need to join DB books with reserve and students

$\pi_{\text{name}}(\sigma_{\text{type}='db'} (\text{Book}) \bowtie \text{Reserve} \bowtie \text{Student})$

More efficient query

$\pi_{\text{name}}(\pi_{\text{sid}}((\pi_{\text{rid}} \sigma_{\text{type}='db'} (\text{Book})) \bowtie \text{Reserve}) \bowtie \text{Student})$

Query optimizer can find the more efficient query!



# Names of students that reserved db books

Book(rid, type)    Reserve(sid, rid)    Student(sid, name)

Need to join DB books with reserve and students

$\pi_{\text{name}}(\sigma_{\text{type}='db'}(\text{Book}) \bowtie \text{Reserve} \bowtie \text{Student})$

More efficient query

$\pi_{\text{name}}(\pi_{\text{sid}}((\pi_{\text{rid}} \sigma_{\text{type}='db'}(\text{Book})) \bowtie \text{Reserve}) \bowtie \text{Student})$

Query optimizer can find the more efficient query!

# Names of students that reserved db books

Book(rid, type)    Reserve(sid, rid)    Student(sid, name)

Need to join DB books with reserve and students

$\pi_{\text{name}}(\sigma_{\text{type}='db'}(\text{Book}) \bowtie \text{Reserve} \bowtie \text{Student})$

More efficient query

$\pi_{\text{name}}(\pi_{\text{sid}}((\pi_{\text{rid}} \sigma_{\text{type}='db'}(\text{Book})) \bowtie \text{Reserve}) \bowtie \text{Student})$

Query optimizer can find the more efficient query!

# Names of students that reserved db books

Book(rid, type)    Reserve(sid, rid)    Student(sid, name)

Need to join DB books with reserve and students

$\pi_{\text{name}}(\sigma_{\text{type}='db'}(\text{Book}) \bowtie \text{Reserve} \bowtie \text{Student})$

More efficient query

$\pi_{\text{name}}(\pi_{\text{sid}}((\pi_{\text{rid}} \sigma_{\text{type}='db'}(\text{Book})) \bowtie \text{Reserve}) \bowtie \text{Student})$

Query optimizer can find the more efficient query!

# Students that reserved DB or HCI book

1. Find all DB or HCI books
2. Find students that reserved one of those books

$\rho(\text{tmp}, (\sigma_{\text{type}='DB' \vee \text{type}='HCI'}(\text{Book})))$   
 $\pi_{\text{name}}(\text{tmp} \bowtie \text{Reserve} \bowtie \text{Student})$

## Alternatives

define tmp using UNION (how?)

what if we replaced  $\vee$  with  $\wedge$  in the query?

# Students that reserved a DB and HCI book

Does previous approach work?

$\rho(\text{tmp}, (\sigma_{\text{type}='DB' \wedge \text{type}='HCI'}(\text{Book})))$   
 $\pi_{\text{name}}(\text{tmp} \bowtie \text{Reserve} \bowtie \text{Student})$

NO

# Why?

$\rho(\text{tmp}, (\sigma_{\text{type}='DB' \wedge \text{type}='HCI'} (\text{Book})))$   
 $\pi_{\text{name}}(\text{tmp} \bowtie \text{Reserve} \bowtie \text{Student})$

for b in Book:

if b.type = 'DB' and b.type = 'HCI':

for r in Reserve:

for s in Student:

if r.sid = s.sid and r.bid = b.bid:

yield b.name

# Students that reserved a DB and HCI book

Does previous approach work?

1. Find students that reserved DB books
2. Find students that reversed HCI books
3. Intersection

$$\begin{aligned} & \rho(\text{tmpDB}, \pi_{\text{sid}}(\sigma_{\text{type}='DB'} \text{ Book}) \bowtie \text{ Reserve}) \\ & \rho(\text{tmpHCI}, \pi_{\text{sid}}(\sigma_{\text{type}='HCI'} \text{ Book}) \bowtie \text{ Reserve}) \\ & \pi_{\text{name}}((\text{tmpDB} \cap \text{tmpHCI}) \bowtie \text{ Student}) \end{aligned}$$

# Students that reserved all books

Use division

Be careful with schemas of inputs to / !

Books(rid, type)      Reserves(sid, rid, date)      Students(sid,)

$p(tmp, (Reserves / Books))$

$tmp \bowtie Students$

Type error



# Students that reserved all books

Use division

Be careful with schemas of inputs to / !

Books(rid, type)      Reserves(sid, rid, date)      Students(sid,)

$\rho(\text{tmp}, (\text{Reserves} / \pi_{\text{rid}}(\text{Books})))$

$\text{tmp} \bowtie \text{Students}$

Each student must have reserved all books on the same date!

# Students that reserved all books

Use division

Be careful with schemas of inputs to / !

Books(rid, type)      Reserves(sid, rid, date)      Students(sid)

$\rho(\text{tmp}, (\pi_{\text{sid,rid}} (\text{Reserves})) / (\pi_{\text{rid}} (\text{Books})))$

$\text{tmp} \bowtie \text{Student}$

What if want students that reserved all horror books?

$\rho(\text{tmp}, (\pi_{\text{sid,rid}} \text{Reserves}) / (\pi_{\text{rid}} (\sigma_{\text{type}='horror'} \text{Book})))$

# Let's step back

Relational algebra is expressiveness benchmark

A language equal in expressiveness as relational algebra is relationally complete

But has limitations

nulls

aggregation

recursion

duplicates

# Equi-Joins are a way of life

Matching of two sets based on shared attributes

Yelp: Join between your location and restaurants

Market: Join between consumers and suppliers

High five: Join between two hands on time and space

Comm.: Join between minds on ideas/concepts

**PLANES**



**ARE JOINS BETWEEN CITIES**

imgflip.com

# What can we do with RA?

Query(DB instance)  $\rightarrow$  Relation instance

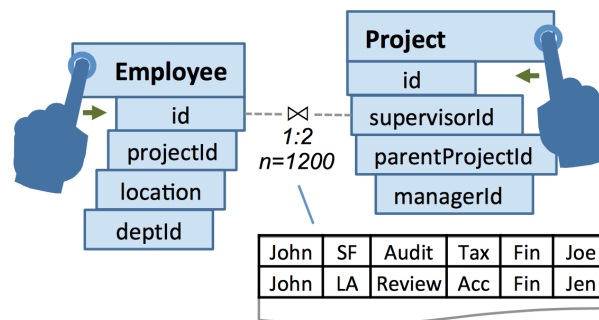
# What can we do with RA?

**Query**(DB instance) = **Relation instance**

## Query by example

Here's data and examples of result, *generate the query for me*

Novel relationally  
complete interfaces



GestureDB. Nandi et al.

# Summary

Relational Algebra (RA) operators

Operators are closed

inputs & outputs are relations

Multiple Relational Algebra queries can be equivalent

It is operational

Same semantics but different performance

Forms basis for optimizations



# Next Time

~~Relational Calculus~~

SQL