

L4 Relational Algebra

Eugene Wu
Fall 2015

Reading

Ramakrishnan
Sections 4.1 and 4.2

Helpful References
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?

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

Tutorial at VLDB 2015 Abadi et al.
<http://www.slideshare.net/abadi/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{M})?

Can assume item structure due to schema

Some algebra operations (\times) need to be modified

Will use this later

Relational Algebra Overview

Core 5 operations

PROJECT (π)

SELECT (σ)

UNION (\cup)

SET DIFFERENCE ($-$)

CROSSPRODUCT (\times)

Additional operations

RENAME (ρ)

INTERSECT (\cap)

JOIN (\bowtie)

DIVIDE ($/$)

Instances Used Today: Library

Students, Reservations

R1	sid	rid	day
1		101	10/10
2		102	11/11

Use positional or named field notation

S1	sid	name	gpa	age
1		eugene	4	20
2		barak	3	21
3		trump	2	88

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

S2	sid	name	gpa	age
4		aziz	3.2	21
2		barak	3	21
3		trump	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	barak	3	21
3	trump	2	88
5	rusty	3.5	21

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

name	age
aziz	21
barak	21
trump	88
rusty	21

Project

S2

sid	name	gpa	age
4	aziz	3.2	21
2	barak	3	21
3	trump	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	barak	3	21
3	trump	2	88

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

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

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

name
eugene
barak

Commutatively

$$A + B = B + A$$

$$A * B = B * A$$

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

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

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

Commutatively

$$A + B = B + A$$

$$A * B = B * A$$

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

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

$$\textcolor{red}{A + (B * C) = (A + B) * C}$$

Commutatively

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

sid	name	gpa	age
1	eugene	4	20
2	barak	3	21
3	trump	2	88

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

Commutatively

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

sid	name	gpa	age
1	eugene	4	20
2	barak	3	21
3	trump	2	88

$$\pi_{\text{age}} \left(\begin{array}{|c|c|c|c|} \hline \text{sid} & \text{name} & \text{gpa} & \text{age} \\ \hline 1 & \text{eugene} & 4 & 20 \\ 2 & \text{barak} & 3 & 21 \\ 3 & \text{trump} & 2 & 88 \\ \hline \end{array} \right) = \begin{array}{|c|} \hline \text{age} \\ \hline 20 \\ 21 \\ \hline \end{array}$$

Commutatively

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

sid	name	gpa	age
1	eugene	4	20
2	barak	3	21
3	trump	2	88

age
20
21
88

Commutatively

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

sid	name	gpa	age
1	eugene	4	20
2	barak	3	21
3	trump	2	88

$$\sigma_{\text{age} < 30} \left(\begin{array}{|c|} \hline \text{age} \\ \hline 20 \\ 21 \\ 88 \\ \hline \end{array} \right) = \begin{array}{|c|} \hline \text{age} \\ \hline 20 \\ 21 \\ \hline \end{array}$$

Commutatively

Does Project and Select commute?

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

What about

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

Commutatively

Does Project and Select commute?

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

What about

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

Commutatively

Does Project and Select commute?

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

What about

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

Commutatively

Does Project and Select commute?

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

What about

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

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)

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

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	barak	3	21
3	trump	2	88

S2			
sid	name	gpa	age
4	aziz	3.2	21
2	barak	3	21
3	trump	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	trump	2	88
2	barak	3	21

Union, Intersect, Set-Difference

S1			
sid	name	gpa	age
1	eugene	4	20
2	barak	3	21
3	trump	2	88

S2			
sid	name	gpa	age
4	aziz	3.2	21
2	barak	3	21
3	trump	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:

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

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

S1			
sid	name	gpa	age
1	eugene	4	20
2	barak	3	21
3	trump	2	88

R1		
sid	rid	day
1	101	10/10
2	102	11/11

$S1 \times R1 =$

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

Rename

$$\rho(\langle \text{new_name} \rangle (\langle \text{mappings} \rangle), Q)$$

Explicitly defines/changes field names of schema

$$\rho(C(1 \rightarrow \text{sid1}, 5 \rightarrow \text{sid2}), S1 \times R1)$$

$C =$

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

Project

$$\pi(\text{blue, orange}) = \text{blue}$$

Select

$$\sigma(\text{blue, orange}) = \text{blue, orange}$$

Cross product

$$\text{blue} \times \text{orange} = \text{blue, orange}$$

Difference

$$\text{blue} - \text{orange} = \text{blue}$$

Union

$$\text{blue} \cup \text{orange} = \text{blue, orange}$$

Intersect

$$\text{blue} \cap \text{orange} = \text{orange}$$

Compound/Convenience Operators

INTERSECT (\cap)

JOIN (\bowtie)

DIVIDE ($/$)

Intersect

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

A, B must be *union-compatible*

Intersect

S1				S2			
sid	name	gpa	age	sid	name	gpa	age
1	eugene	4	20	4	aziz	3.2	21
2	barak	3	21	2	barak	3	21
3	trump	2	88	3	trump	2	88
				5	rusty	3.5	21

$$S1 \cap S2 =$$

sid	name	gpa	age
2	barak	3	21
3	trump	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?

$$A \cap B = A - ? \quad (\text{think venn diagram})$$

Intersect

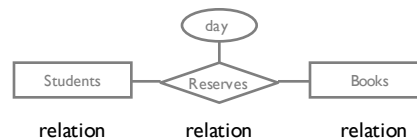
$$A \cap B = R_{\text{result}}$$



Can we express using core operators?

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

Joins (high level)



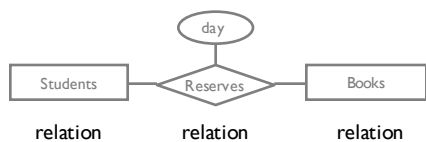
What if you want to query across all three tables?

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

Need to combine these tables

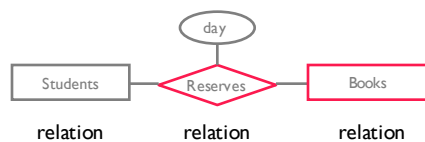
Cross product? But that ignores foreign key references

Joins (high level)



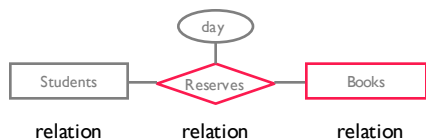
S I				R I			B I	
sid	name	gpa	age	sid	rid	day	rid	name
1	eugene	4	20	1	101	10/10	101	The Purple Crayon
2	barak	3	21	2	102	11/11	102	1984
3	trump	2	88					

Joins (high level)



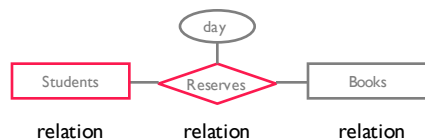
S I				R I			B I	
sid	name	gpa	age	sid	rid	day	rid	name
1	eugene	4	20	1	101	10/10	101	The Purple Crayon
2	barak	3	21	2	102	11/11	102	1984
3	trump	2	88					

Joins (high level)



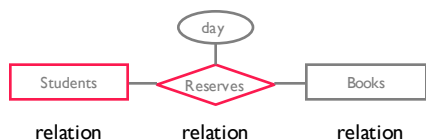
S I				RBI				
sid	name	gpa	age	sid	(rid)	day	(rid)	name
1	eugene	4	20	1	101	10/10	101	The Purple Crayon
2	barak	3	21	2	102	11/11	102	1984
3	trump	2	88					

Joins (high level)



S I				RBI				
sid	name	gpa	age	sid	(rid)	day	(rid)	name
1	eugene	4	20	1	101	10/10	101	The Purple Crayon
2	barak	3	21	2	102	11/11	102	1984
3	trump	2	88					

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	barak	3	21	2	102	11/11	102	1984

theta (θ) Join

$$A \bowtie_{\theta} B = \sigma_{\theta}(A \times B)$$

Most general form

Result schema same as cross product

Often *far* more efficient to compute than cross product

Commutative

$$(A \bowtie_{\theta} B) \bowtie_{\theta} C = A \bowtie_{\theta} (B \bowtie_{\theta} C)$$

theta (θ) Join

S1

sid	name	gpa	age
1	eugene	4	20
2	barak	3	21
3	trump	2	88

R1

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

$S1 \bowtie_{S1.sid \leq R1.sid} R1 =$

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

Equi-Join

$$A \bowtie_{attr} B = \pi_{all\ attr\ except\ Battr}(A \bowtie_{Aattr = Battr} B)$$

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

S1

sid	name	gpa	age
1	eugene	4	20
2	barak	3	21
3	trump	2	88

R1

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

$S1 \bowtie_{sid} R1 =$

sid	name	gpa	age	rid	day
1	eugene	4	20	101	10/10
2	barak	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

R2

rid
2
4

R3

rid
1
2
4

A/R1

A/R2

A/R3

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

R2

rid
2
4

R3

rid
1
2
4

A/R1

A/R2

A/R3

Examples

A		R1		R2		R3	
sid	rid	rid		rid		rid	
1	1	2		2		1	
1	2			4		2	
1	3					4	
1	4						
2	1						
2	2						
3	2						
4	2						
4	4						

A/R1

A/R2

A/R3

Examples

A		R1		R2		R3	
sid	rid	rid		rid		rid	
1	1	2		2		1	
1	2			4		2	
1	3					4	
1	4						
2	1						
2	2						
3	2						
4	2						
4	4						

A/R1

A/R2

A/R3

Is A/B a Fundamental Operation?

No. Shorthand like Joins

joins so common, it's natively supported

Hint: Find all xs not 'disqualified' by some y in B.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		B	
sid	rid	rid	
1	1	2	
1	2	4	
1	3		
1	4		
2	1		
2	2		
3	2		
4	2		
4	4		

Disqualified =
A/B =

A		B		$\pi_{\text{sid}}(A) \times B$	
sid	rid	rid		sid	rid
1	1	2		1	2
1	2	4		1	4
1	3			2	2
1	4			2	4
2	1			3	2
2	2			3	4
3	2			4	2
4	2			4	4
4	4				

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

A		B		$\pi_{\text{sid}}(A) \times B$		$\pi_{\text{sid}}(A) \times B - A$	
sid	rid	rid		sid	rid	sid	rid
1	1	2		1	2	2	4
1	2	4		1	4	3	4
1	3			2	2		
1	4			2	4		
2	1			3	2		
2	2			3	4		
3	2			4	2		
4	2			4	4		
4	4						

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

A		B		$\pi_{sid}(A) \times B$		$(\pi_{sid}(A) \times B) - A$	
sid	rid	rid		sid	rid	sid	rid
1	1	2		1	2	2	4
1	2	4		1	4	3	4
1	3			2	2		
1	4			2	4		
2	1			3	2		
2	2			3	4		
3	2			4	2		
4	2			4	4		
4	4						

sid
1
4

A/B

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

Names of students that reserved book 2

$\pi_{name}(\sigma_{rid=2}(R1) \bowtie S1)$

Equivalent
Queries

$p(tmp1, \sigma_{rid=2}(R1))$
 $p(tmp2, tmp1 \bowtie S1)$
 $\pi_{name}(tmp2)$

$\pi_{name}(\sigma_{rid=2}(R1 \bowtie S1))$

Names of students that reserved db
books

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

Need to join DB books with reserve and students

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

Names of students that reserved db
books

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

Need to join DB books with reserve and students

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

Names of students that reserved db
books

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

Need to join DB books with reserve and students

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

Names of students that reserved db
books

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

Need to join DB books with reserve and students

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

Names of students that reserved db books

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

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)

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)

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)

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

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

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?

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

NO

Students that reserved a DB and HCI book

Does previous approach work?

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

$$p(tmpDB, \pi_{sid}(\sigma_{type='DB'} Book) \bowtie Reserve)$$

$$p(tmpHCI, \pi_{sid}(\sigma_{type='HCI'} Book) \bowtie Reserve)$$

$$\pi_{name}((tmpDB \cap tmpHCI) \bowtie Student)$$

Students that reserved all books

Use division

Be careful with schemas of inputs to / !

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

$$\pi_{name}(tmp \bowtie Student)$$

What if want students that reserved all horror books?

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

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

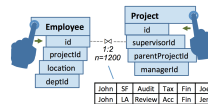


What can we do with RA?

Query by example

Here's my data and examples of the 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