Administrivia

HW1 Part 1 Solutions. Deep apologies.

Deadline for Part 2 extended to end of week.

A wonderful article about database modeling

https://ovid.github.io/articles/death-by-database.html

They'd ask all customers for a billing address and if they had a different delivery address. At the database level, they had a "one-to-many" relationship between customers and addresses.

That was their first problem. A customer's partner might come into Bob's and order something and if the address was entered correctly it would be flagged as "in use" and we had to use a different address or *deliberately* enter a typo. Fortunately, addresses were case-sensitive, so many people had UPPER-CASE ADDRESSES.

L4
Relational Algebra

Eugene Wu

Supplemental Materials

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/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

$$Q(R_1,\ldots,Rn) = R_{result}$$

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 (π)
- SELECT (σ)
- UNION (U)
- SET DIFFERENCE (-)
- CROSSPRODUCT (x)

Additional operations

- RENAME (ρ)
- 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)

R1

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

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

Project

$$\pi_{<attr1,\ldots>}(A) = R_{result}$$

Pick out desired attributes (subset of columns)

Schema is subset of input schema in the projection list

- $\pi_{<a,b,c>}(A)$ 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

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

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

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

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

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

age
21
88

Where did all the rows go?

Real systems typically don't remove duplicates. Why?

Select

$$\sigma_{<p>}(A) = R_{result}$$

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

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

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

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

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

$$A + B = B + A$$

$$A * B = B * A$$

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

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

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

Commutatively

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

$$\underbrace{\sigma_{age<30}\left(\begin{array}{|l|l|l|l|}\hline \text{sid} & \text{name} & \text{gpa} & \text{age} \\ \hline 1 & \text{eugene} & 4 & 20 \\ \hline 2 & \text{barb} & 3 & 21 \\ \hline 3 & \text{tanya} & 2 & 88 \\ \hline \end{array}\right)}$$

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

Commutatively

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

$\sigma_{age<30}$

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

π_{age}

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

=

age
20
21

Commutatively

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

π_{age} (

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

)

age
20
21
88

Commutatively

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

π_{age} (

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

)

$\sigma_{age<30}$ (

age
20
21
88

) =

age
20
21

Commutatively

Does Project and Select commute?

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

What about

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

Commutatively

Does Project and Select commute?

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

What about

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

Commutatively

Does Project and Select commute?

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

What about

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

Commutatively

Does Project and Select commute?

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

What about

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

OK!

Union, Set-Difference

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

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) U B(thug int, life int) = ?

Union, Set-Difference

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

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) U B(thug int, life int) = R_{result}(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 U 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 → 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

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

Each row of A paired with each row of B

- Result schema concats A and B's fields, inherit if possible
- Conflict: students and reservations have *sid* field

Different than mathematical "X" by flattening results:

- 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	barb	3	21
3	tanya	2	88

R1

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

S1 x R1 =

(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

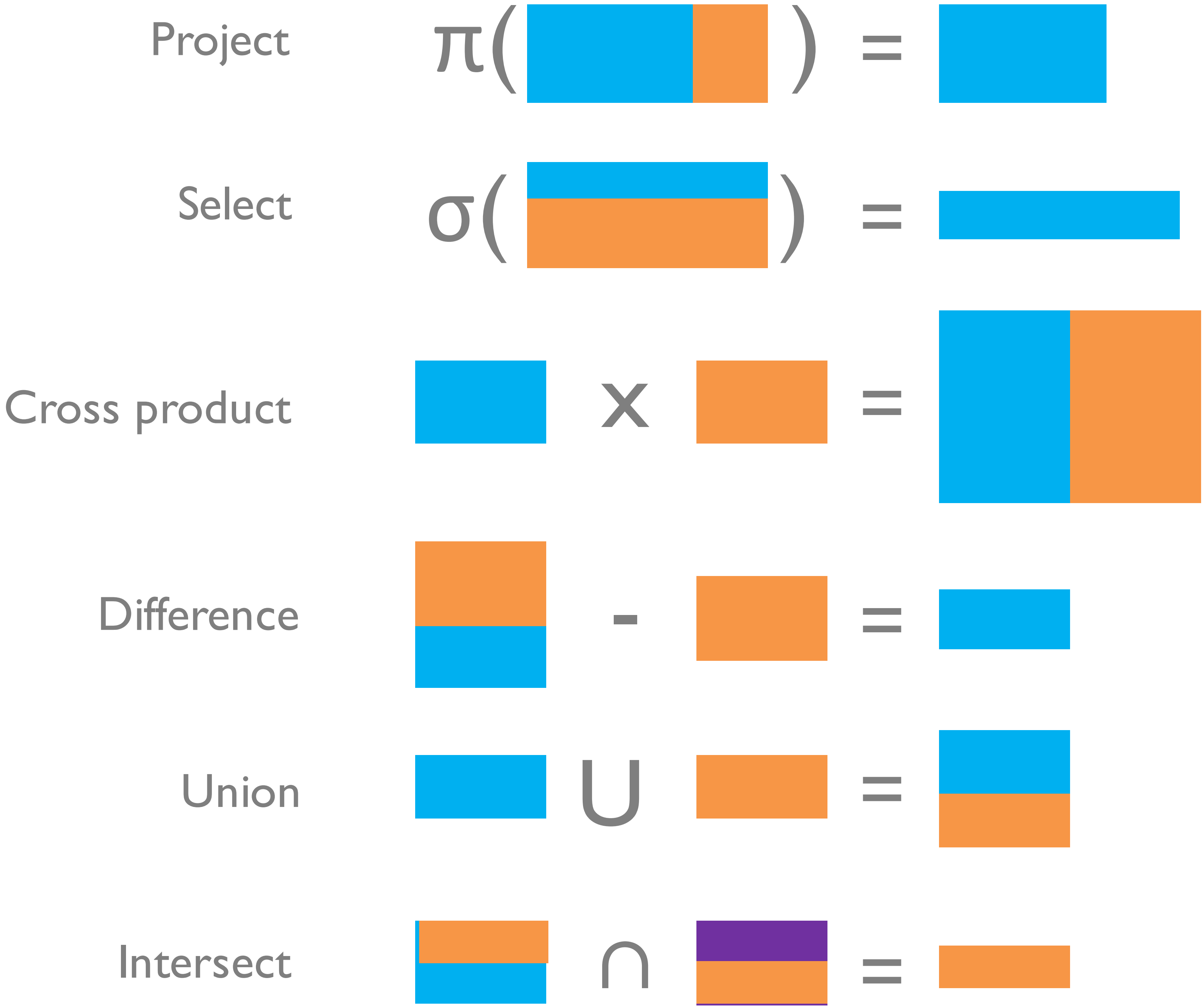
p(<new_name>(<mappings>), Q)

Explicitly defines/changes field names of schema

p(C(1 → sid1, 5 → sid2), S1 x 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
π(
)
=
Select
σ(
)
=
Cross product
x
=
Difference
-
=
Union
∪
=
Intersect
∩
=

Compound/Convenience Operators

INTERSECT ($\cap$)

JOIN ($\bowtie$)

DIVIDE (/)

Intersect

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

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

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

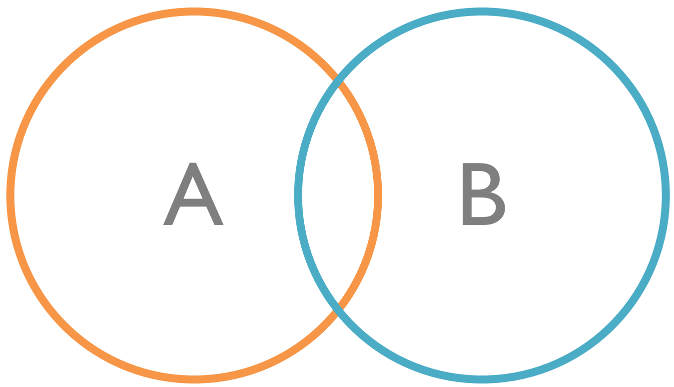
A, B must be *union-compatible*

Can we express using core operators?

$$A \cap B \;=\; ?$$

Intersect

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

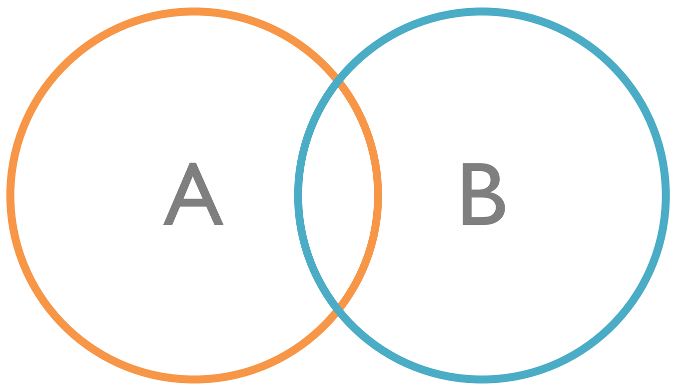


Can we express using core operators?

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

Intersect

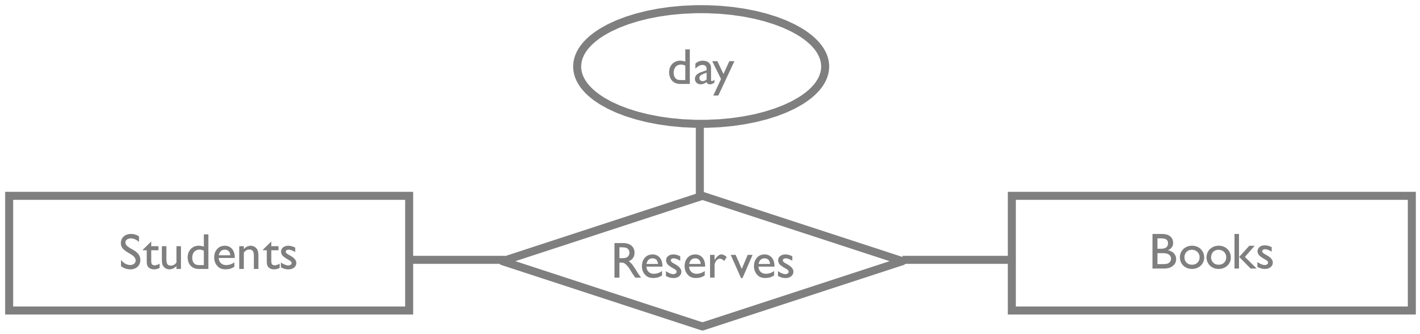
$$A \cap B = R_{result}$$



Can we express using core operators?

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

Joins (high level)



relation relation relation

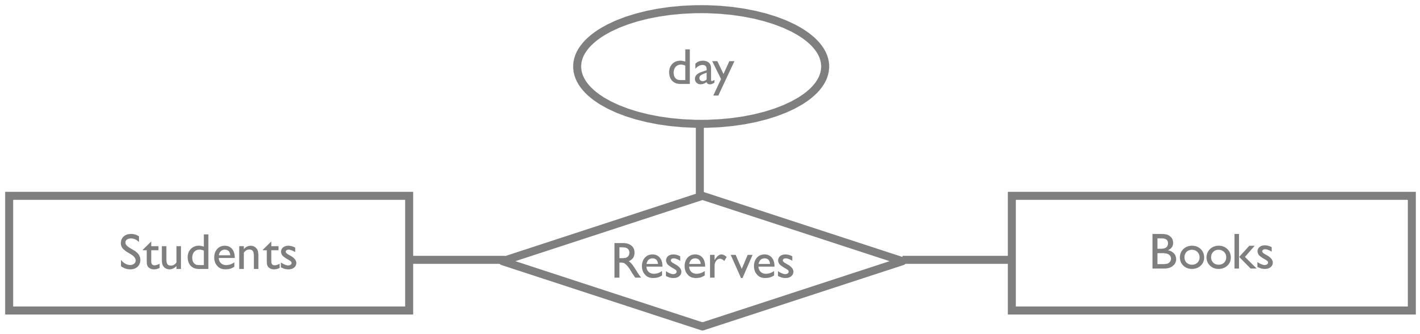
What if you want to query across all three tables?

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)



relation relation relation

S1

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

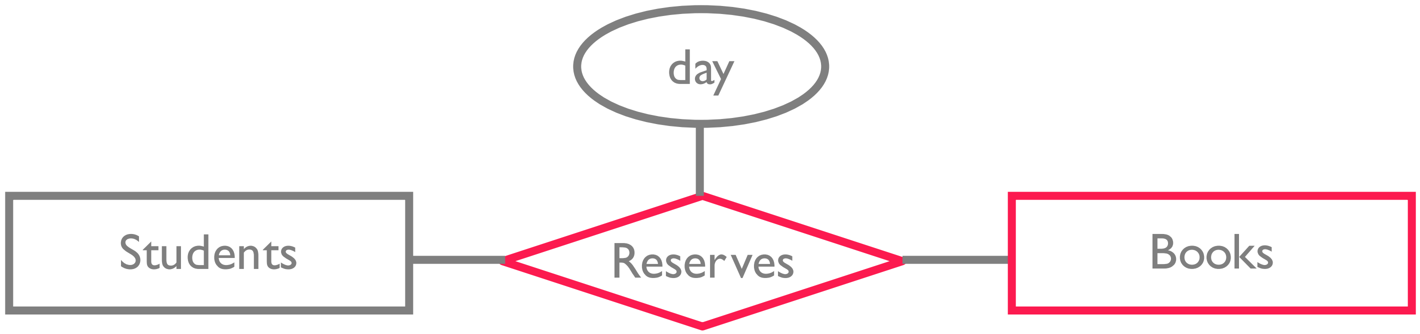
R1

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

B1

rid	name
101	The Purple Crayon
102	1984

Joins (high level)



relation relation relation

S1

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

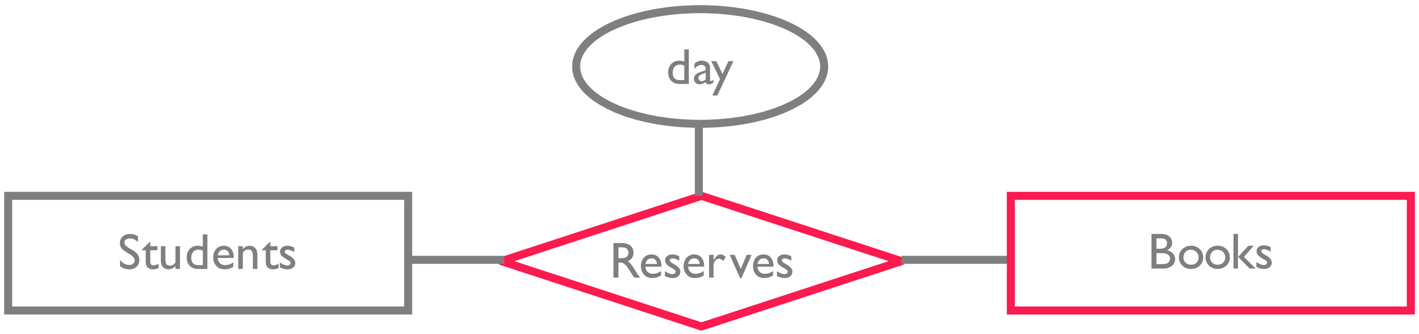
R1

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

B1

rid	name
101	The Purple Crayon
102	1984

Joins (high level)



relation relation relation

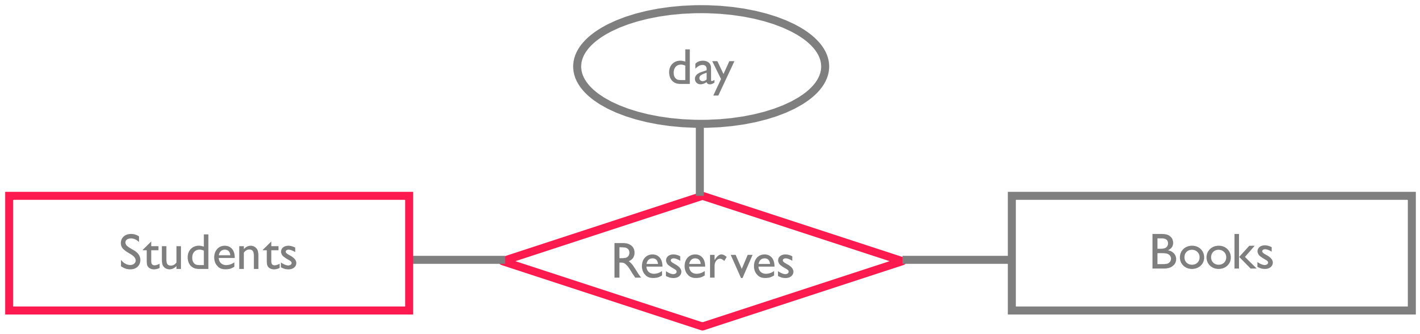
S1

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

RB1

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

Joins (high level)



relation relation relation

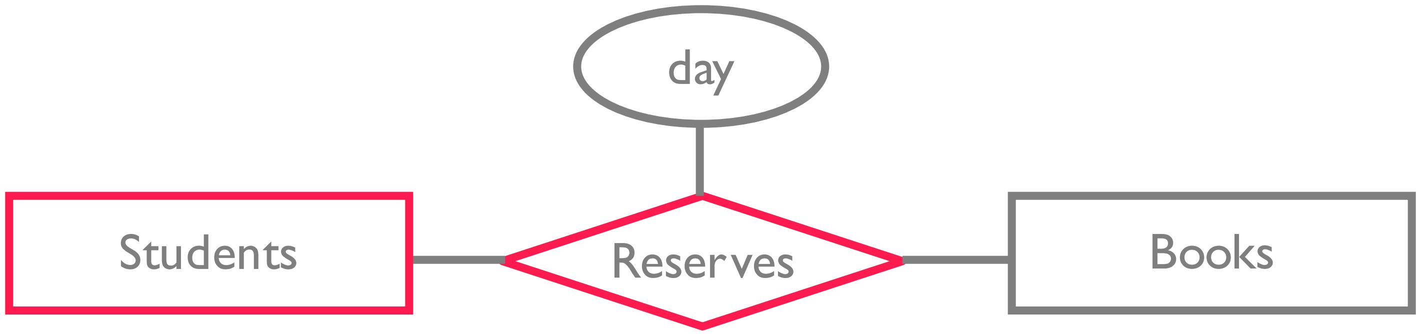
SI

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)



relation relation relation

SRB1

(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

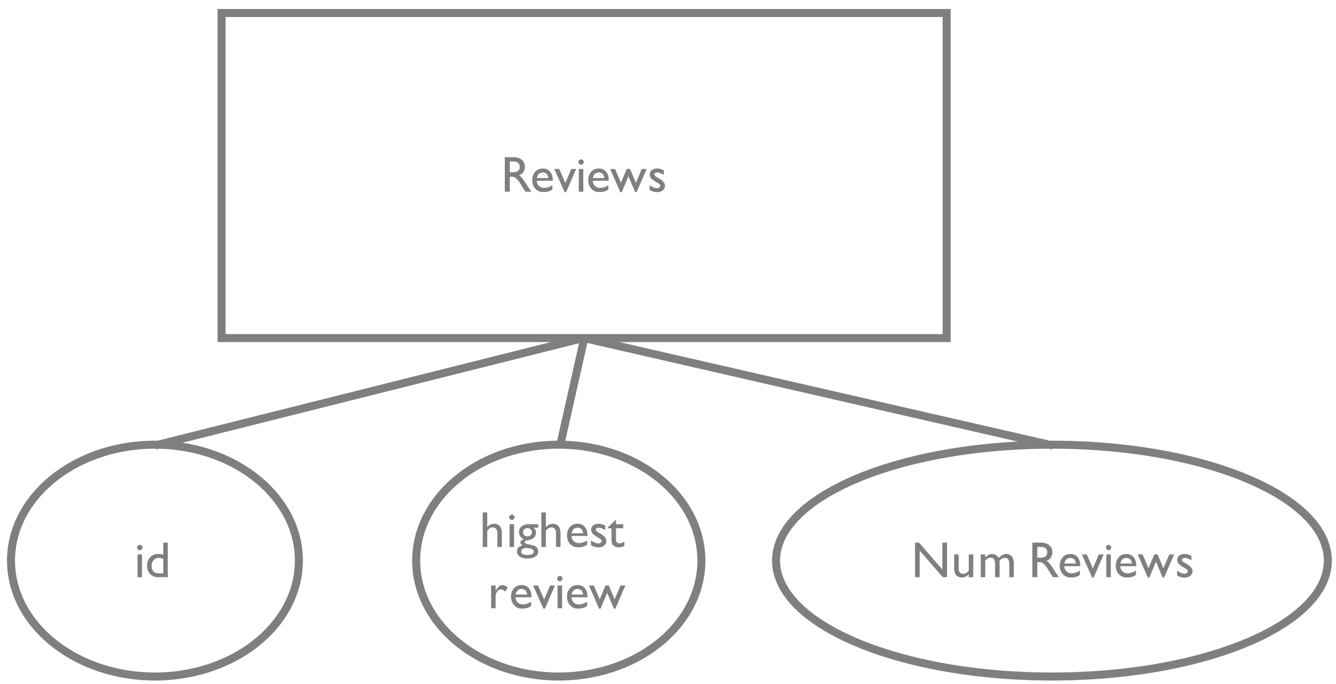
Administrative Trivia

Google cloud codes will be released soon

- Instructions on how to set it up.
- Use to have a linux env and for project 1

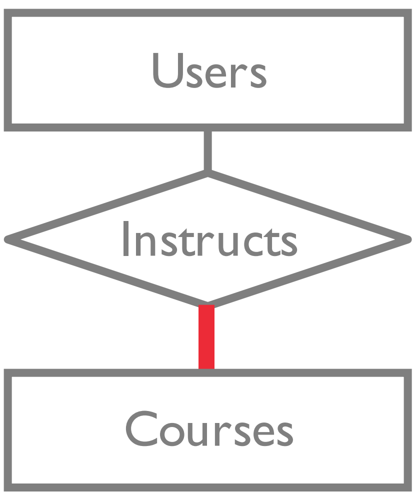
Comments on Part 1

Performance optimizations



Comments on Part 1

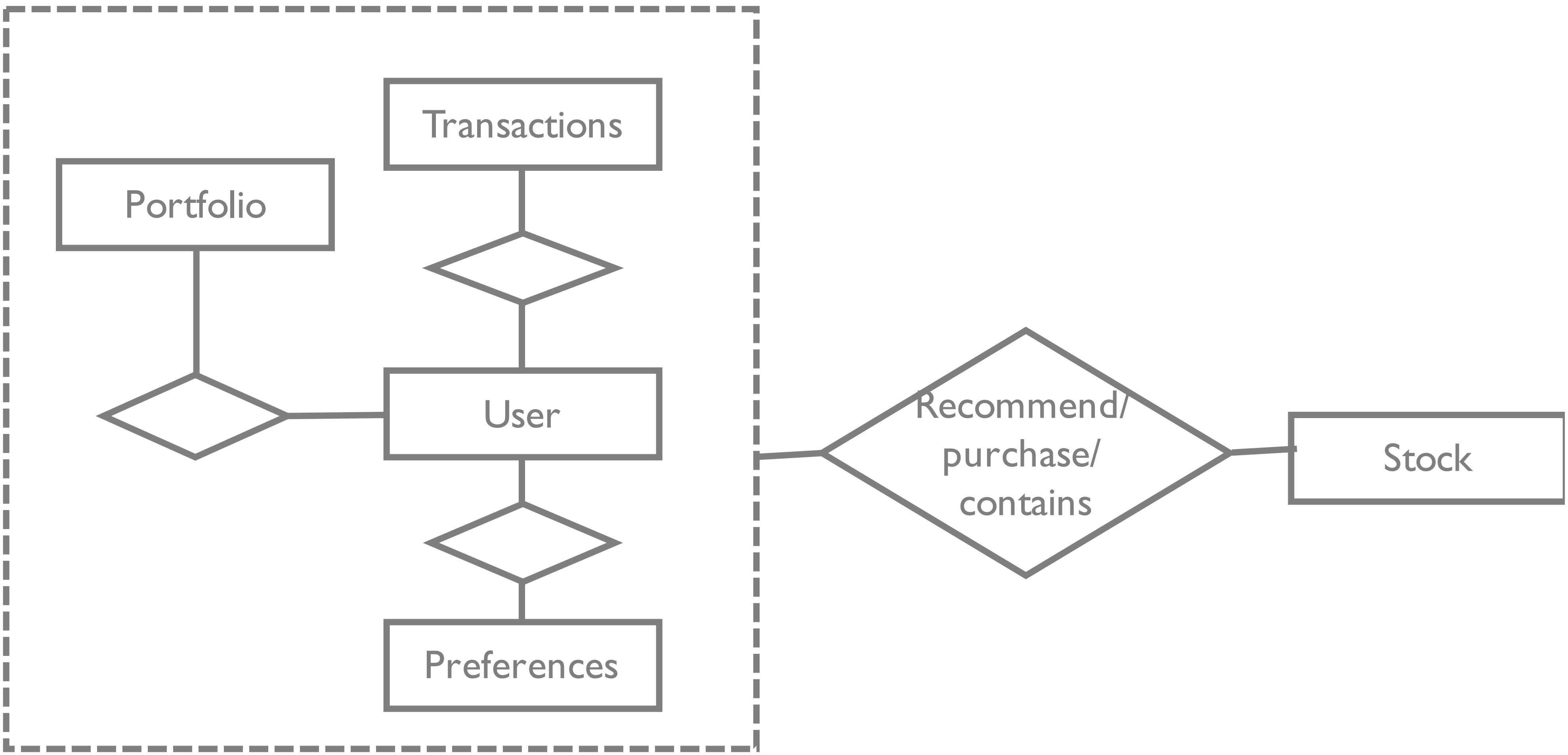
Total Participation




```
CREATE TABLE Course_Instructs(
    cid int
    uid int NOT NULL,
    name text,
    loc text,
    PRIMARY KEY (cid),
    FOREIGN KEY (uid) REFERENCES Users
        ON DELETE NO ACTION
)
```

Comments on Part 1

Aggregation around many relationships



theta (θ) 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 (θ) Join

S1

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

R1

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

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

$\sigma_{S1.sid \leq R1.sid}(S1 \times R1) =$

(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 (θ) Join

S1

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

R1

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

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

$\sigma_{S1.sid \leq R1.sid}(S1 \times R1) =$

(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 (θ) Join

S1

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

R1

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

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

$\sigma_{S1.sid \leq R1.sid}(S1 \times R1) =$

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

Equi-Join

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

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

S1

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

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

R2

rid
2
4

R3

rid
1
2
4

sid
1
2
3
4

A/R1

A/R2

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 <x, *y*>)
2. we obtain an <*x*,*y*> 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

$\pi_{sid}(A)$

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

$\pi_{sid}(A)$

sid
1
2
3
4

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

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

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

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

$\pi_{sid}(A)$

sid
1
2
3
4

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

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

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

sid	rid
2	4
3	4

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

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

$\pi_{sid}(A)$

sid
1
2
3
4

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

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

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

sid	rid
2	4
3	4

sid
1
4

A/B

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

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

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

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

store in temporary table

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

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

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

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

More efficient query

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

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

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

More efficient query

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

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

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

More efficient query

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

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

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

More efficient query

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

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

$$\rho(tmp, (\sigma_{type='DB' \vee type='HCI'} (Book))$$

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

Alternatives

define tmp using UNION (how?)

what if we replaced v with ^ in the query?

Students that reserved a DB and HCI book

Does previous approach work?

$$p(tmp, (\sigma_{type='DB' \wedge type='HCI'} (Book))$$

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

NO

Why?

$$p(tmp, (\sigma_{type='DB' \wedge type='HCI'} (Book))$$

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

```
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

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

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

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

Students that reserved all books

Use division

Be careful with schemas of inputs to / !

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,)

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

$$tmp \bowtie Students$$

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)

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

$$tmp \bowtie Student$$

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

Mat

oute

What can we do with RA?

Query(DB instance) → Relation instance

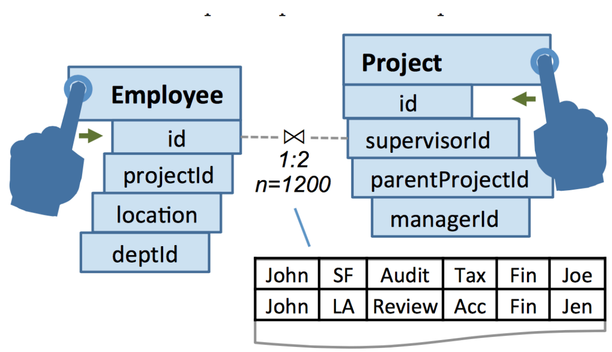
What can we do with RA?

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