L8
Relational Algebra (Joins+More)

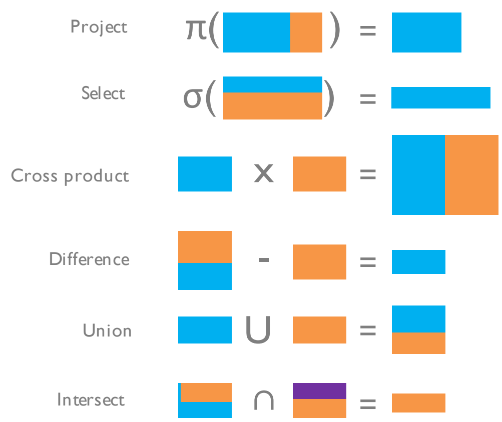
Eugene Wu
Fall 2016

Administrivia

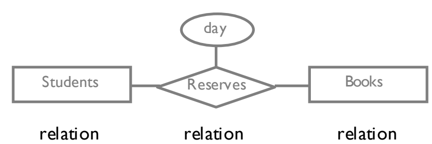
Today:
- Project 1 Part 1 DUE
- Project 1 Part 2 out!

Future:
- HW1 due Wed in class
- HW2 out next Mon

Lost a partner? Qi will post a message on piazza



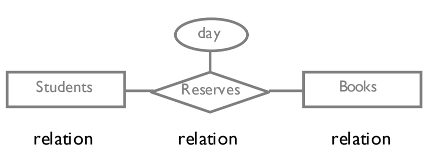
Joins (high level)



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)



S1

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

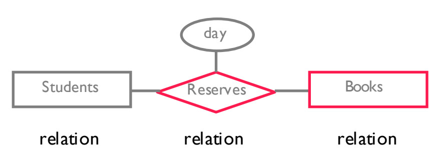
R1

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

B1

rid	name
101	The Purple Crayon
102	1984

Joins (high level)



S1

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

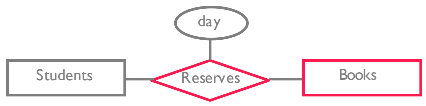
R1

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

B1

rid	name
101	The Purple Crayon
102	1984

Joins (high level)



relation relation relation

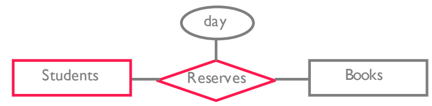
SI

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

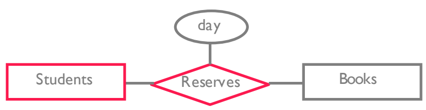
SI

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

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

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

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

SI

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

RI

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

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

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

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

Equi-Join

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

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	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 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

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

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

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

Is A/B a Fundamental Operation?

No. Shorthand like Joins

joins so common, it's natively supported

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

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 = $\pi_{sid}(A)$

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

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

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

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

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

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

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

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

Names of students that reserved book 2

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

$\pi_{name}(\sigma_{rid=2}\ (\text{Reserve}) \bowtie \text{Student})$

Equivalent Queries

$p(tmp1,\ \sigma_{rid=2}\ (\text{Reserve}))$
$p(tmp2,\ tmp1 \bowtie \text{Student})$
$\pi_{name}(tmp2)$

$\pi_{name}(\sigma_{rid=2}(\text{Reserve} \bowtie \text{Student}))$

Names of students that reserved db books

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

Need to join DB books with reserve and students

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

Names of students that reserved db books

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

Need to join DB books with reserve and students

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

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

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

Names of students that reserved db books

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

Need to join DB books with reserve and students

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

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

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

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

Alternatives

define tmp using UNION (how?)

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

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

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

Students that reserved all books

Use division
Be careful with schemas of inputs to / !

$\rho(tmp, (\pi_{sid,rid}\ Reserves) / (\pi_{rid}\ Books))$
$\pi_{name}(tmp \bowtie Student)$

What if want students that reserved all horror books?

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

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

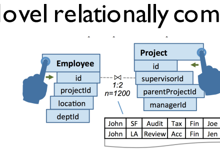
High five: Join between two hands on time and space

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