

Database Applications (15-415)

Relational Algebra

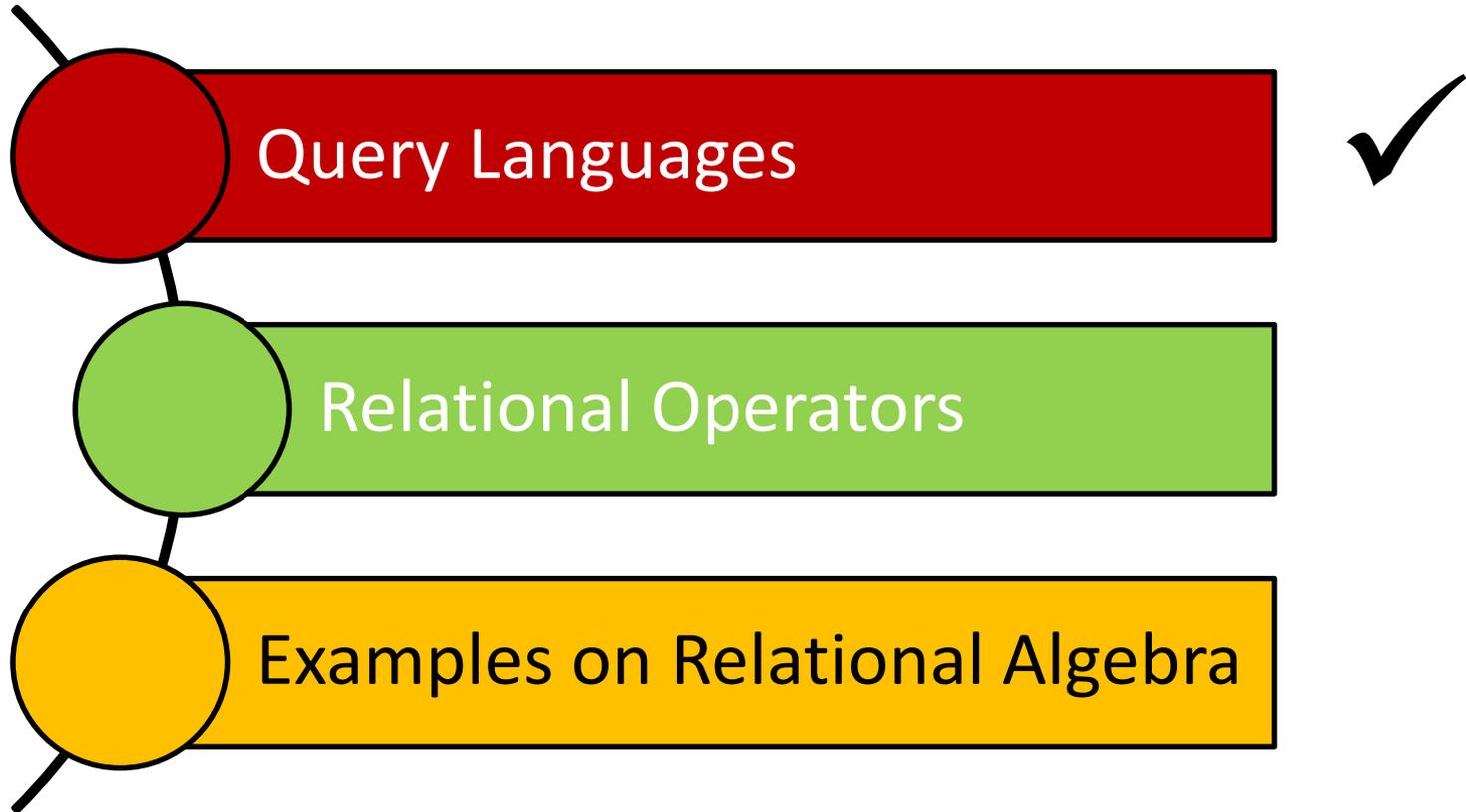
Lecture 5, January 24, 2016

Mohammad Hammoud

Today...

- **Last Session:**
 - The relational model
- **Today's Session:**
 - Relational algebra
 - Relational query languages (in general)
 - Relational operators
 - Few examples
- **Announcements:**
 - PS2 will be posted on Tuesday, Jan 26th
 - PS1 grades will be out on Tuesday, Jan 26th
 - Your Postgres DB VMs will be ready by Thursday, Jan 28th. We will use them during the upcoming recitation.

Outline



Relational Query Languages

- **Query languages** (QLs) allow *manipulating* and *retrieving* data from databases
- The relational model supports simple and powerful QLs:
 - Strong formal foundation based on logic
 - High amenability for effective optimizations
- **Query Languages != programming languages!**
 - QLs are not expected to be “Turing complete”
 - QLs are not intended to be used for complex calculations
 - QLs support easy and efficient access to large datasets

Formal Relational Query Languages

- There are two mathematical Query Languages which form the basis for commercial languages (e.g. SQL)
 - **Relational Algebra**
 - Queries are composed of operators
 - Each query describes a step-by-step procedure for computing the desired answer
 - Very useful for representing *execution plans*
 - **Relational Calculus**
 - Queries are subsets of first-order logic
 - Queries describe desired answers without specifying how they will be computed
 - A type of *non-procedural* (or *declarative*) formal query language

Formal Relational Query Languages

- There are two mathematical Query Languages which form the basis for commercial languages (e.g. SQL)

- Relational Algebra

- Queries are composed of operators
- Each query describes a procedure for computing the desired answer
- Very useful for representing *execution plans*

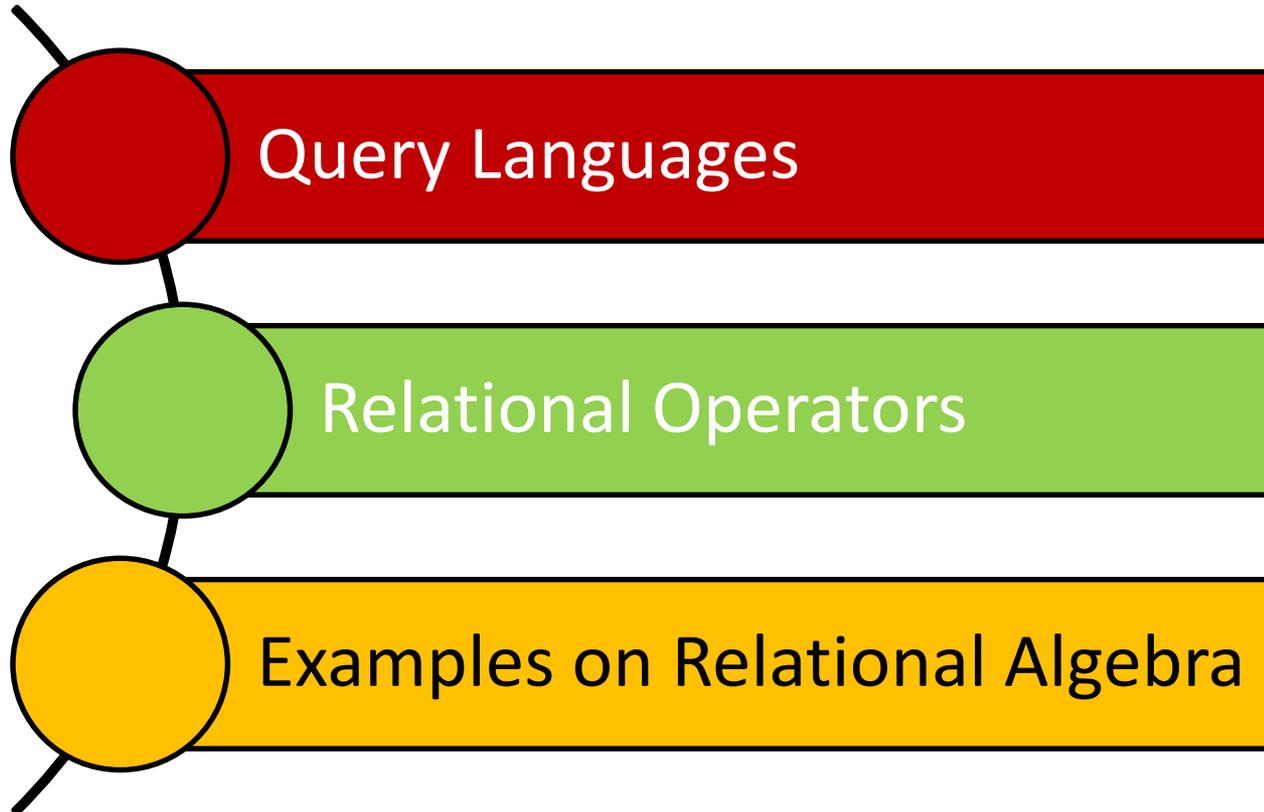
This session's topic

- Relational Calculus

- Queries are subsets of first-order logic
- Queries describe *what* is wanted, not *how* they will be computed
- A type of *non-procedural* (or *declarative*) formal query language

Next session's topic (*very briefly*)

Outline



Relational Algebra

- Operators (with notations):
 1. Selection (σ)
 2. Projection (π)
 3. Cross-product (\times)
 4. Set-difference ($-$)
 5. Union (\cup)
 6. Intersection (\cap)
 7. Join (\bowtie)
 8. Division (\div)
 9. Renaming (ρ)

- Each operation returns a relation, hence, operations can be *composed*! (i.e., Algebra is “closed”)

Relational Algebra

- Operators (with notations):

1.

Selection (σ)

2.

Projection (π)

3.

Cross-product (\times)

Basic

4.

Set-difference ($-$)

5.

Union (\cup)

6.

Intersection (\cap)

7.

Join (\bowtie)

Additional, yet

8.

extremely useful!

9.

Renaming (ρ)

- Each operation returns a relation, hence, operations can be *composed*! (i.e., Algebra is “closed”)

The Projection Operation

- Projection: $\pi_{att-list}(R)$
 - “Project out” attributes that are NOT in *att-list*
 - The schema of the output relation contains ONLY the fields in *att-list*, with the same names that they had in the input relation
- Example 1: $\pi_{sname, rating}(S2)$

Input Relation:

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

Output Relation:

sname	rating
yuppy	9
lubber	8
guppy	5
rusty	10



The Projection Operation

- Example 2: $\pi_{age}(S2)$

Input Relation:

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

Output Relation:

age
35.0
55.5

- The projection operator eliminates *duplicates*!
 - Note: real DBMSs typically do not eliminate duplicates unless explicitly asked for

The Selection Operation

- Selection: $\sigma_{condition}(R)$
 - Selects rows that satisfy the selection *condition*
 - The schema of the output relation is identical to the schema of the input relation

- Example: $\sigma_{rating > 8}(S2)$

Input Relation:

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

Output Relation:

sid	sname	rating	age
28	yuppy	9	35.0
58	rusty	10	35.0



Operator Composition

- *The output relation can be the input for another relational algebra operation!* (*Operator composition*)

- **Example:**

$$\pi_{sname, rating}(\sigma_{rating > 8}(S2))$$

Input Relation:

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

S2

Intermediate Relation:

sid	sname	rating	age
28	yuppy	9	35.0
58	rusty	10	35.0

Final Output Relation:

sname	rating
yuppy	9
rusty	10

The Union Operation

- Union: $R \cup S$
 - The two input relations must be **union-compatible**
 - Same number of fields
 - 'Corresponding' fields have the same type
 - The output relation includes all tuples that occur "in either" R or S "or both"
 - The schema of the output relation is identical to the schema of R

- Example: $S1 \cup S2$

Input Relations:

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



Output Relation:

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

The Intersection Operation

- Intersection: $R \cap S$
 - The two input relations must be *union-compatible*
 - The output relation includes all tuples that occur “in both” R and S
 - The schema of the output relation is identical to the schema of R
- Example: $S1 \cap S2$

Input Relations:

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



Output Relation:

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

The Set-Difference Operation

- Set-Difference: $R - S$
 - The two input relations must be *union-compatible*
 - The output relation includes all tuples that occur in R “but not” in S
 - The schema of the output relation is identical to the schema of R
- Example: $S1 - S2$

Input Relations:

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

Output Relation:



sid	sname	rating	age
22	dustin	7	45.0

The Cross-Product and Renaming Operations

- Cross Product: RXS
 - Each row of R is paired with each row of S
 - The schema of the output relation concatenates S1's and R1's schemas
 - **Conflict:** R and S might have the same field name
 - **Solution:** Rename fields using the “Renaming Operator”
 - Renaming: $\rho(R(\overline{F}), E)$

- **Example:** $S1XR1$

Input Relations:

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

S1

<u>sid</u>	<u>bid</u>	<u>day</u>
22	101	10/10/96
58	103	11/12/96

R1

Output Relation:

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

Conflict: Both S1 and R1 have a field called *sid*

The Cross-Product and Renaming Operations

- Cross Product: $R \times S$
 - Each row of R is paired with each row of S
 - The schema of the output relation concatenates S1's and R1's schemas
 - **Conflict**: R and S might have the same field name
 - **Solution**: Rename fields using the "Renaming Operator"
 - Renaming: $\rho(R(\overline{F}), E)$

- **Example**: $S1 \times R1$

Input Relations:

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

S1

<u>sid</u>	<u>bid</u>	<u>day</u>
22	101	10/10/96
58	103	11/12/96

R1



Output Relation:

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

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

The Join Operation

- (Theta) Join : $R \bowtie_c S = \sigma_c(R \times S)$
 - The schema of the output relation is the same as that of cross-product
 - It usually includes fewer tuples than cross-product

- **Example:** $S1 \bowtie S1.sid < R1.sid R1$

Input Relations:

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	10/10/96
58	103	11/12/96

R1



Output Relation:

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	58	103	11/12/96

Will be redundant "if" the condition is $S1.sid = R1.sid$!

The Join Operation

- Equi-Join: $R \bowtie_c S = \sigma_c(R \times S)$
 - A special case of theta join where the condition c contains only **equalities**
 - The schema of the output relation is the same as that of cross-product, *“but only one copy of the fields for which equality is specified”*
- Natural Join: $R \bowtie S$
 - Equijoin on *“all”* common fields
- Example: $S1 \bowtie_{S1.sid = R1.sid} R1$

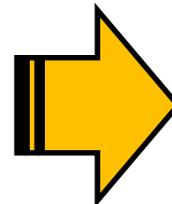
Input Relations:

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

S1

<u>sid</u>	<u>bid</u>	<u>day</u>
22	101	10/10/96
58	103	11/12/96

R1



Output Relation:

sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/96
58	rusty	10	35.0	103	11/12/96

ONLY one sid column!

The Join Operation

- Equi-Join: $R \bowtie_c S = \sigma_c(R \times S)$
 - A special case of theta join where the condition c contains only **equalities**
 - The schema of the output relation is the same as that of cross-product, *“but only one copy of the fields for which equality is specified”*

- Natural Join: $R \bowtie S$
 - Equijoin on *“all”* common fields

- Example: $S1 \bowtie R1$ Natural Join

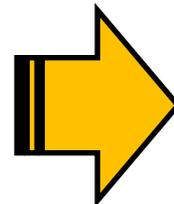
Input Relations:

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	10/10/96
58	103	11/12/96

R1



Output Relation:

sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/96
58	rusty	10	35.0	103	11/12/96

In this case, same as equi-join!

The Division Operation

- Division: $R \div S$
 - Not supported as a primitive operator, but useful for expressing queries like:

Find sailors who have reserved all boats
 - Let A have 2 fields, x and y ; B have only field y :
 - A/B contains all x tuples (sailors) such that for every y tuple (boat) in B , there is an xy tuple in A
 - *Or*: If the set of y values (boats) associated with an x value (sailor) in A contains all y values in B , then x value is in A/B
 - Formally: $A/B = \{ \langle x \rangle \mid \exists \langle x, y \rangle \in A \quad \forall \langle y \rangle \in B \}$
- In general, x and y can be any lists of fields; y is the list of fields in B , and x y is the list of fields in A

Examples of Divisions

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 can be derived from the fundamental operators
- **Idea:** For A/B, compute all x values that are not 'disqualified' by some y value in B
 - x value is disqualified if by attaching y value from B, we obtain an xy tuple that is "not" in A

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

A/B : $\pi_x(A) -$ all disqualified tuples

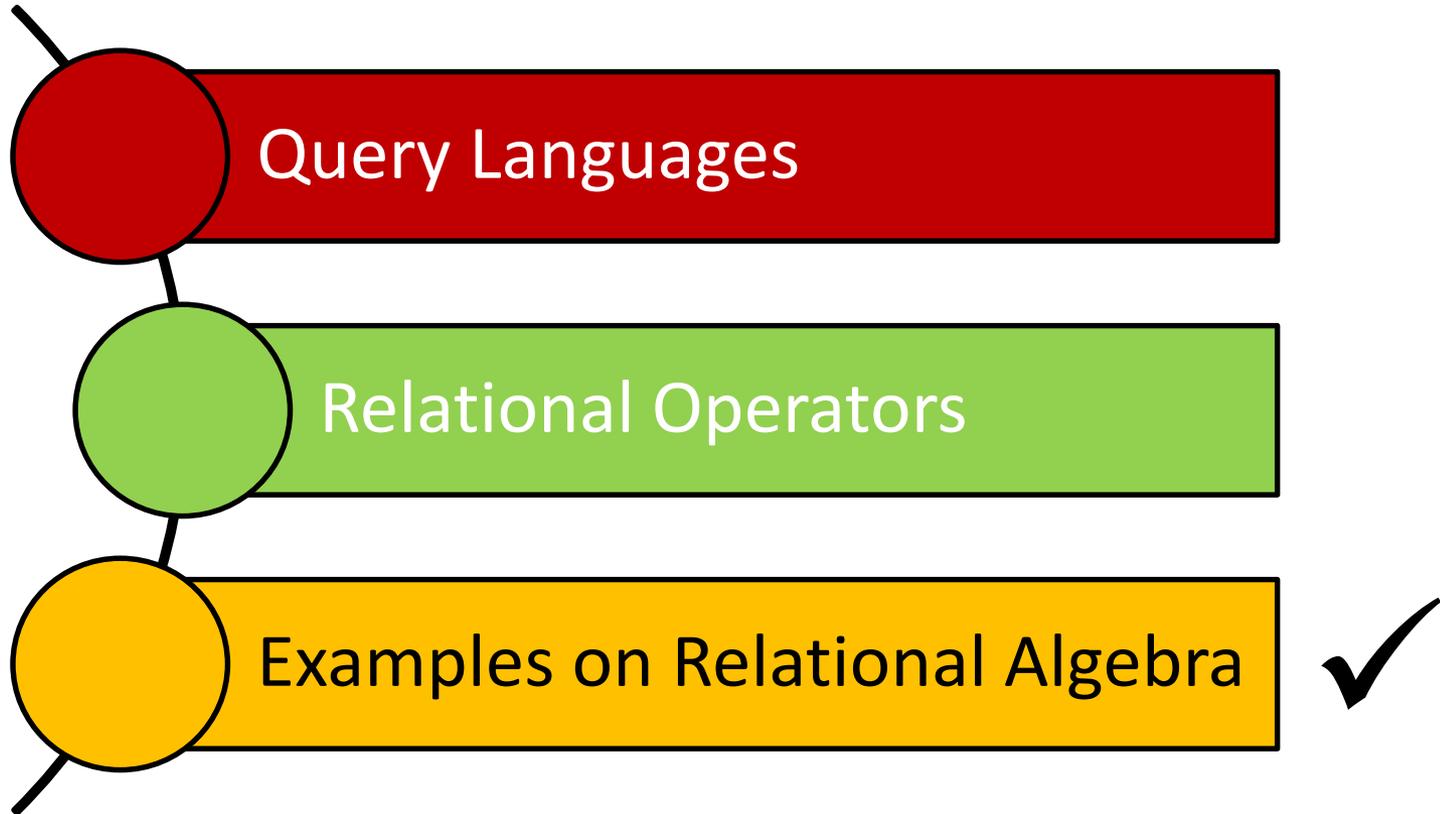
Relational Algebra: Summary

- Operators (with notations):
 1. **Selection** (σ): selects a subset of rows from a relation
 2. **Projection** (π): deletes unwanted columns from a relation
 3. **Cross-product** (\times): allows combining two relations
 4. **Set-difference** ($-$): retains tuples which are in relation 1, “but not” in relation 2
 5. **Union** (\cup): retains tuples which are in “either” relation 1 or relation 2, “or in both”

Relational Algebra: Summary

- Operators (with notations):
 6. **Intersection** (\cap): retains tuples which are in relation 1 “and” in relation 2
 7. **Join** (\bowtie): allows combining two relations according to a specific condition (e.g., *theta*, *equi* and *natural* joins)
 8. **Division** (\div): generates the largest instance Q such that $Q \times B \subseteq A$ when computing A/B
 9. **Renaming** (ρ): returns an instance of a new relation with some fields being potentially “renamed”

Outline



Additional Examples

- Q1: Find names of sailors who've reserved boat #103

Sid	Sname	Rating	Age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

An Instance S3 of Sailors

Sid	Bid	Day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

An Instance R2 of Reserves

Bid	Bname	Color
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green
104	Marine	Red

An Instance B1 of Boats

Additional Examples

- Q1: Find names of sailors who've reserved boat #103

$$\pi_{sname}((\sigma_{bid=103} Reserves) \bowtie Sailors)$$

OR:

$$\pi_{sname}(\sigma_{bid=103}(Reserves \bowtie Sailors))$$

OR:

$$\rho(Temp1, \sigma_{bid=103} Reserves)$$

$$\rho(Temp2, Temp1 \bowtie Sailors)$$

$$\pi_{sname}(Temp2)$$

Which one to choose?

Additional Examples

- Q2: Find names of sailors who've reserved a red boat

Sid	Sname	Rating	Age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

An Instance S3 of Sailors

Sid	Bid	Day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

An Instance R2 of Reserves

Bid	Bname	Color
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green
104	Marine	Red

An Instance B1 of Boats

Additional Examples

- Q2: Find names of sailors who've reserved a red boat

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

OR:

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

A query optimizer can find the second one, given the first solution!

Additional Examples

- Q3: Find sailors who've reserved a red or a green boat

Sid	Sname	Rating	Age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

An Instance S3 of Sailors

Sid	Bid	Day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

An Instance R2 of Reserves

Bid	Bname	Color
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green
104	Marine	Red

An Instance B1 of Boats

Additional Examples

- Q3: Find sailors who've reserved a red or a green boat

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

π_{sname} (*Tempboats* \bowtie *Reserves* \bowtie *Sailors*)

Can we define Tempboats using union?

What happens if \vee is replaced by \wedge ?

Additional Examples

- Q4: Find sailors who've reserved a red and a green boat

Sid	Sname	Rating	Age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

An Instance S3 of Sailors

Sid	Bid	Day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

An Instance R2 of Reserves

Bid	Bname	Color
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green
104	Marine	Red

An Instance B1 of Boats

Additional Examples

- Q4: Find sailors who've reserved a red and a green boat

$$\rho (Tempred, \pi_{sid}((\sigma_{color='red'} Boats) \bowtie Reserves))$$
$$\rho (Tempgreen, \pi_{sid}((\sigma_{color='green'} Boats) \bowtie Reserves))$$
$$\pi_{sname}((Tempred \cap Tempgreen) \bowtie Sailors)$$

Would the previous approach (i.e., using \cap instead of \cup) work?

Additional Examples

- Q5: Find the names of sailors who've reserved all boats

Sid	Sname	Rating	Age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

An Instance S3 of Sailors

Sid	Bid	Day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

An Instance R2 of Reserves

Bid	Bname	Color
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green
104	Marine	Red

An Instance B1 of Boats

Additional Examples

- Q5: Find the names of sailors who've reserved all boats

$$\rho (Temp\ sids, (\pi_{sid, bid} Reserves) / (\pi_{bid} Boats))$$
$$\pi_{sname} (Temp\ sids \bowtie Sailors)$$

How can we find sailors who've reserved all 'Interlake' boats?

Summary

- The relational model has rigorously defined query languages that are simple and powerful
- Relational algebra is operational; useful as internal representation for query evaluation plans
- Several ways of expressing a given query; a query optimizer should choose the most efficient version

Next Class

Relational Calculus