RELATIONAL ALGEBRA

Dr. Mohammed A. Mohammed
University of Sulaimani | College of science | Computer dept.

Relational Query Languages

- *Query languages:* Allow manipulation and retrieval of data from a database.
- Relational model supports simple, powerful QLs:
  - Strong formal foundation based on logic.
  - Allows for much optimization.
- Query Languages ≠ programming languages!
  - QLs not expected to be "Turing complete".
  - QLs not intended to be used for complex calculations.
  - QLs support easy, efficient access to large data sets.

Formal Relational Query Languages

Two mathematical Query Languages form the basis for "real" languages (e.g. SQL), and for implementation:

- ***Relational Algebra***: More operational, very useful for representing execution plans.
- ***Relational Calculus***: Lets users describe what they want, rather than how to compute it. (Non-operational, *declarative*.)

Preliminaries (Introductions)

A query is applied to *relation instances*, and the result of a query is also a relation instance.

```
SELECT   S.name, E.grade
FROM  Students S, Enrolled E
WHERE  S.sid = E.sid and S.age<20
```

- Each relational query describes a **step-by-step procedure** for computing the desired answer, based on the order in which operators are applied in the query.

- The procedural nature of the algebra allows us to think of an algebra expression as a guidelines, or a plan, for evaluating a query, and relational systems in fact use algebra expressions to represent query evaluation plans.

Notation

Positional vs. named-field notation:

- **Positional notation** – referring to fields by their positions. Easier for formal definitions.
- **Named-field notation** – using field names to refer to fields. Make query more readable.
- Both used in SQL

Sample Schema

- We present a number of sample queries using the following schema:

Sailors(*sid:* integer, *sname:* string, *rating:* integer, *age:* real)

Boats(*bid:* integer, *bname:* string, *color:* string)

Reserves(*sid:* integer, *bid:* integer, *day:* date)

Example Instances

- "Sailors" and "Reserves" relations for our examples.
- We'll use positional or named field notation, assume that names of fields in query results are `inherited' from names of fields in query input relations.

R1

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

S1

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

S2

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

Relational Algebra

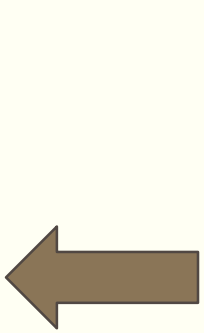
- Basic operations:
  - *Selection* (σ) Selects a subset of **rows** from relation.
  - *Projection* (π) Deletes unwanted **columns** from relation.
  - *Cross-product* ($\times$) Allows us to combine two relations.
  - *Set-difference* ($-$) Tuples in reln. 1, but not in reln. 2.
  - *Union* ($\cup$) Tuples in reln. 1 and in reln. 2.

- Additional operations:
  - Intersection, *join*, division, renaming: Not essential, but (very!) useful.

- Since each operation returns a relation, operations can be *composed*!

Projection

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



$$\pi_{sname,rating}(S2)$$

sname	rating
yuppy	9
lubber	8
guppy	5
rusty	10

Projection

- Deletes attributes that are not in *projection list*.
- *Schema* of result contains exactly the fields in the projection list, with the same names that they had in the (only) input relation.
- Projection operator has to eliminate *duplicates*.
  - Note: real systems typically don't do duplicate elimination unless the user explicitly asks for it.

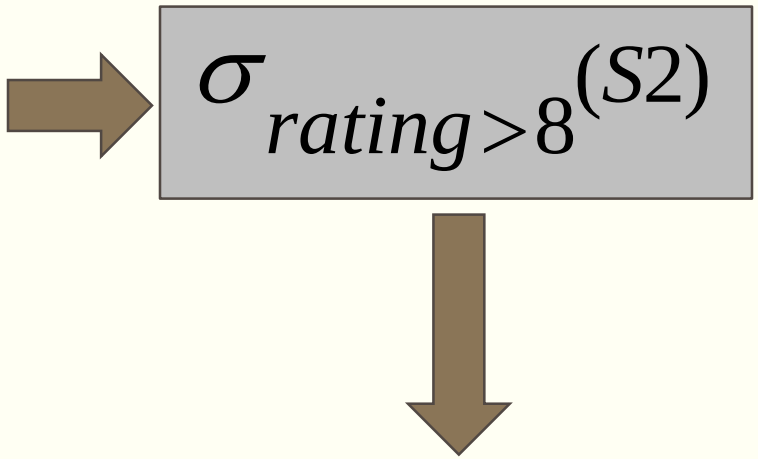
sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

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

age
35.0
55.5

Selection

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

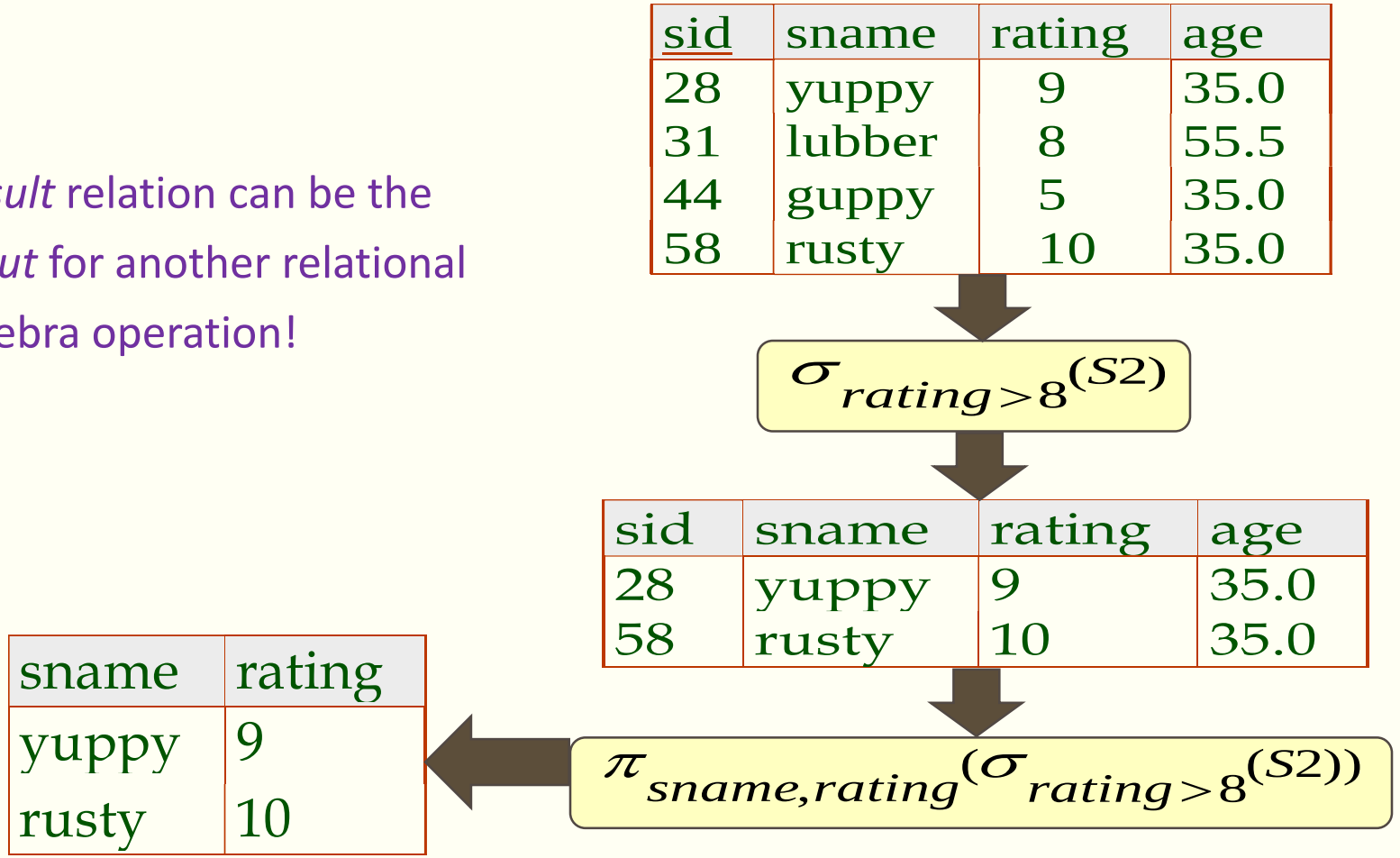


sid	sname	rating	age
28	yuppy	9	35.0
58	rusty	10	35.0

- Selects rows that satisfy *selection condition*.
- *Schema* of result identical to schema of (only) input relation

Composition Operator

Result relation can be the *input* for another relational algebra operation!



Union

S1

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

S2

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

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

$S1 \cup S2$

المتشابة بحيث مرة وحدة بدون تكرار

Intersection

S1

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

S2

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

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

$S1 \cap S2$

Set-Difference

S1

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

S2

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

sid	sname	rating	age
22	dustin	7	45.0

$S1 - S2$

يجيب الموجود في الجدول الاول
وما موجود بالجدول الثاني

Union, Intersection, Set-Difference

All of these operations take two input relations, which must be *union-compatible*:

- Same number of fields.
- `Corresponding' fields have the same type.

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

$S1 \cup S2$

sid	sname	rating	age
22	dustin	7	45.0

$S1 - S2$

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

$S1 \cap S2$

Cross-Product

S1

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

R1

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

R1 × S1

(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

Each row of R1 is paired with each row of S1.

Cross-Product

Result schema has one field per field of S1 and R1, with field names `inherited' if possible.

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

(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

يحتاج في الامتحان يقلك عبر rename للجدول التالي

Renaming operator:

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

C(***sid1***:integer, ***sname****: string,* ***rating****: integer,* ***age****: real,* ***sid2****: integer,* ***bid****: integer,****day****: dates)*

Joins

Reserves

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

Sailors

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

Boats

bid	bname	color
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green
104	Marine	Red

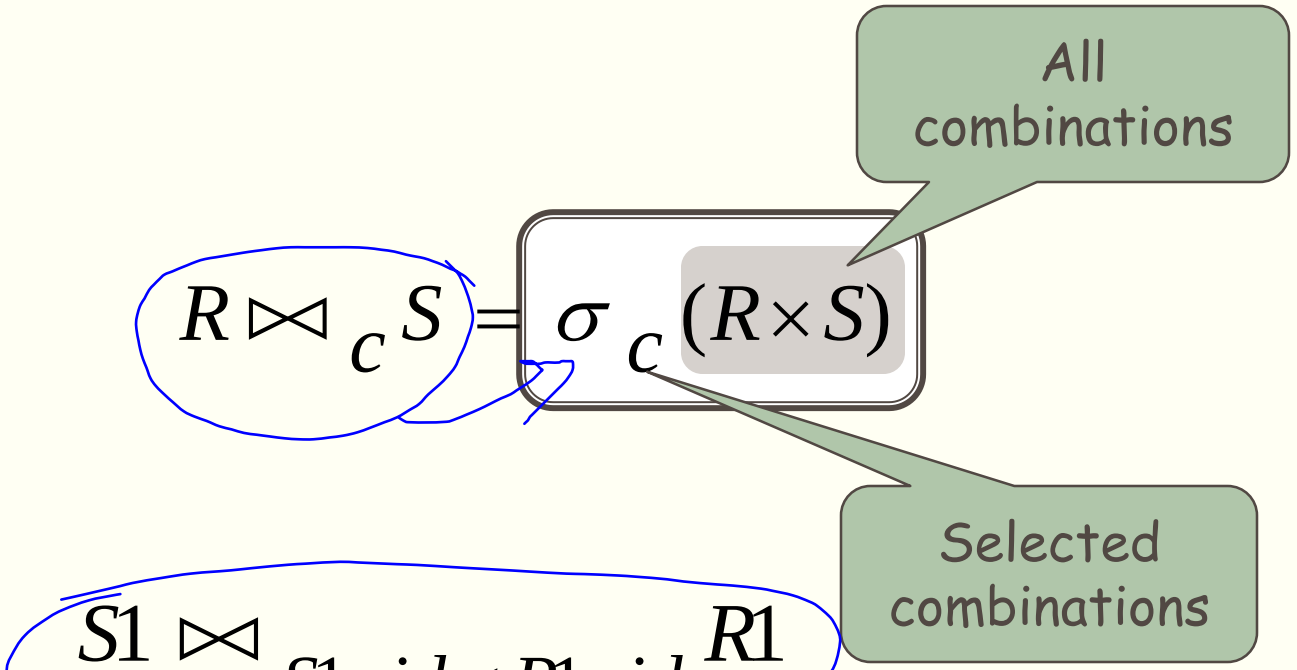
Find names of sailors who've reserved a green boat

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

Joins

Condition Join:

Example:



Joins

$$R \bowtie_c S = \sigma_c (R \times S)$$

Condition Join:

R1

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

S1

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

$$S1 \bowtie_{S1.sid < R1.sid} R1$$

(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

Joins

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

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

R1 × S1

(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

Joins

- *Result schema* same as that of cross-product.
- Fewer tuples than cross-product, might be able to compute more efficiently
- Sometimes called a *theta-join*.

Equi-Join

S1

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

Foreign key

R1

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

Equi-Join: A special case of condition join where the condition *c* contains only ***equalities***.

$S1 \bowtie_{sid} R1$ =

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

Equi-Joins & Natural Join

- *Equi-Join*: A special case of condition join where the condition *c* contains only ***equalities.***

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

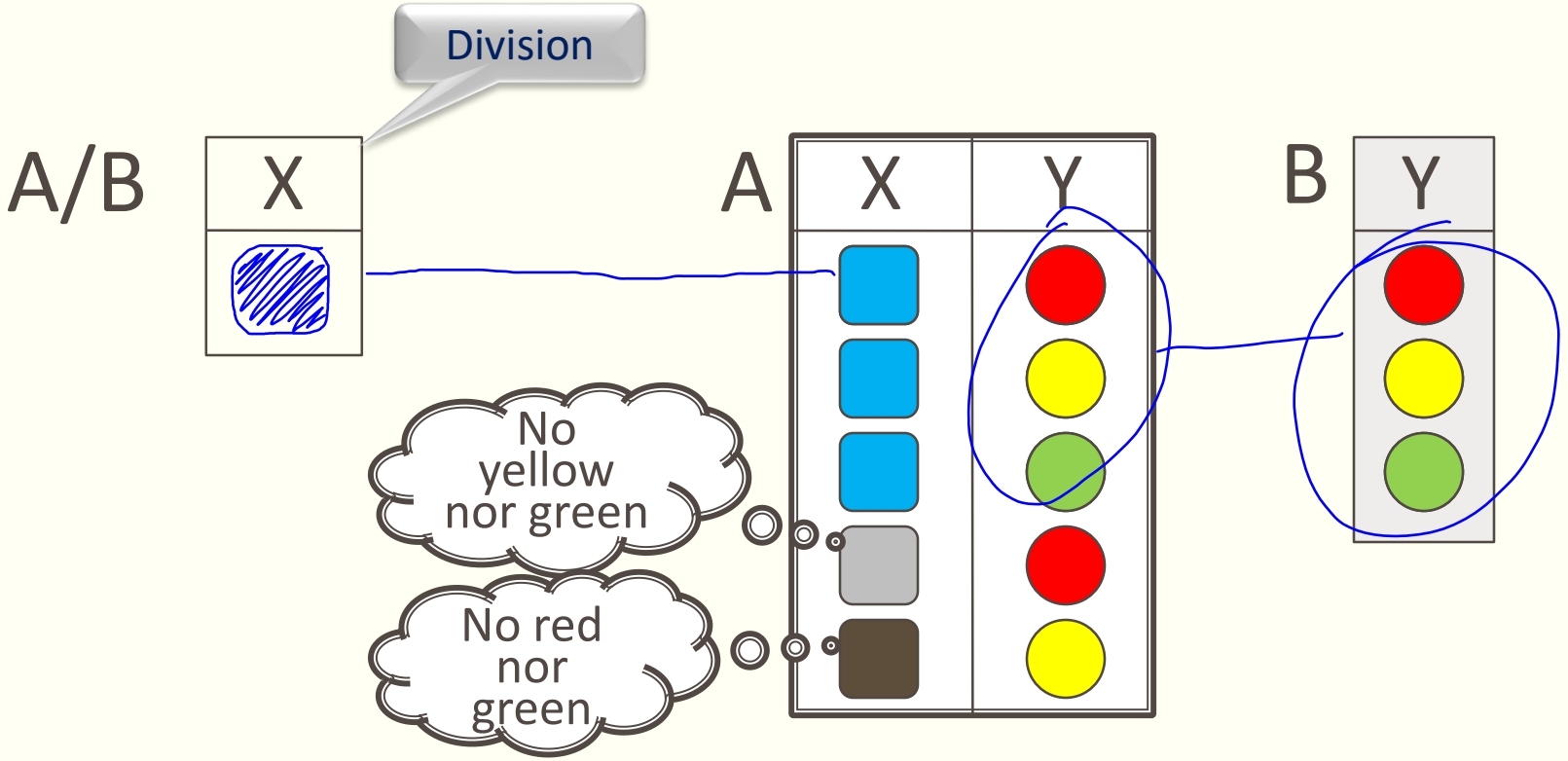
$$S1 \bowtie_{sid} R1$$

- *Result schema* similar to cross-product, but **only one copy of fields** for which equality is specified.
- *Natural Join*: Equijoin on *all* common fields.

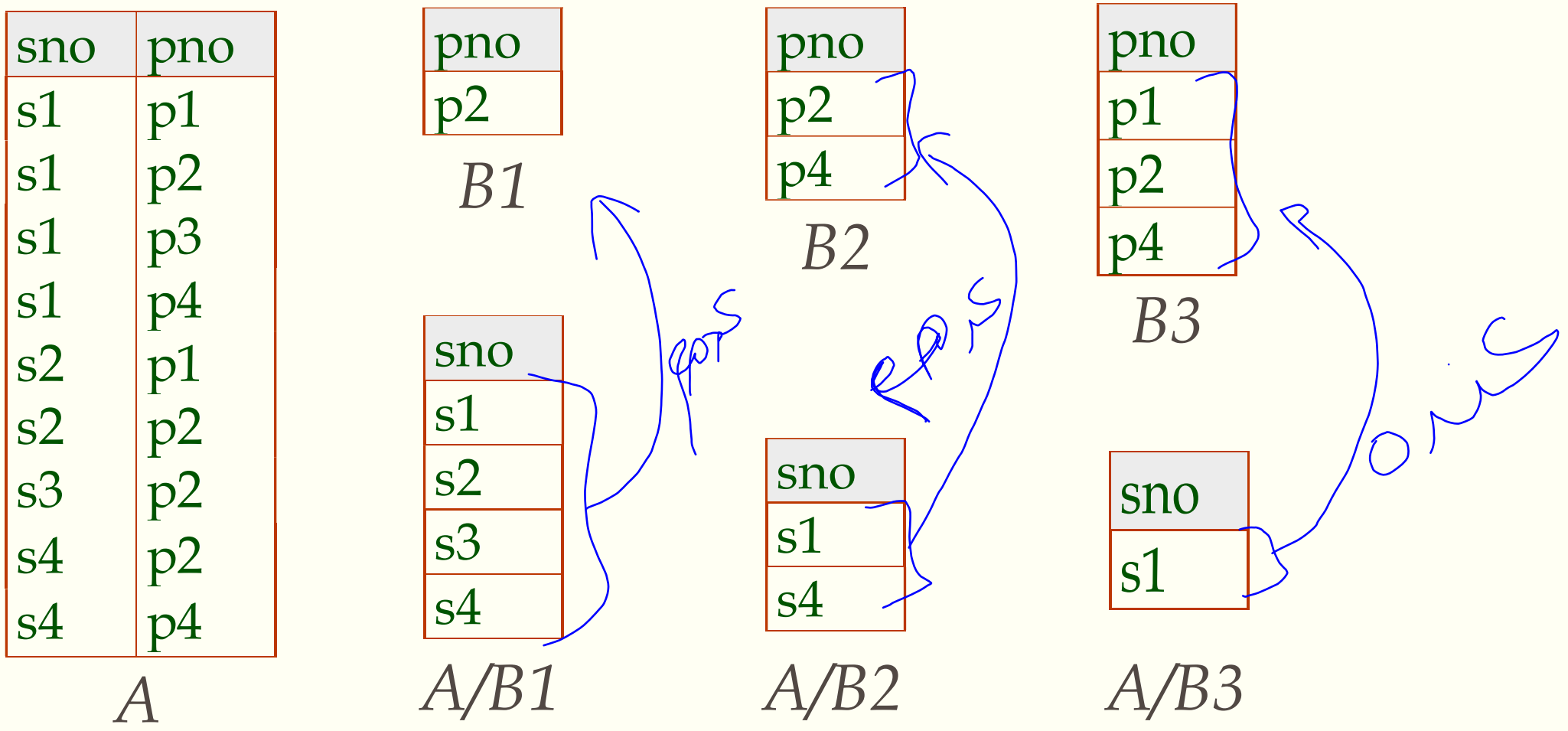
Division

Not supported as a primitive operator, but useful for expressing queries like:

Find sailors who have reserved all boats



Examples of Division A/B



sno	pno
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

A

pno
p2

B1

sno
s1
s2
s3
s4

A/B1

pno
p2
p4

B2

sno
s1
s4

A/B2

pno
p1
p2
p4

B3

sno
s1

A/B3

Find names of sailors who've reserved boat #103

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

(3) Find out the names of those sailors

(1) Find all reservations for boat 103

(2) Follow the foreign-key pointer to identify sailors who made these reservations

Sailors

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

Foreign key

Reserves

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

Find names of sailors who've reserved boat #103

Solution 2:

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

(1) Find all reservations for boat 103

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

(2) Follow the foreign-key pointer to identify sailors who made these reservations

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

(3) Find out the names of those sailors

Sailors

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

Foreign key

Reserves

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

Find names of sailors who've reserved boat #103

Solution 3:

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

(3) Find out the names of sailors who made those sailors

(2) Retain only reservations for boat 103

(1) Find reservations for every sailor

Sailors

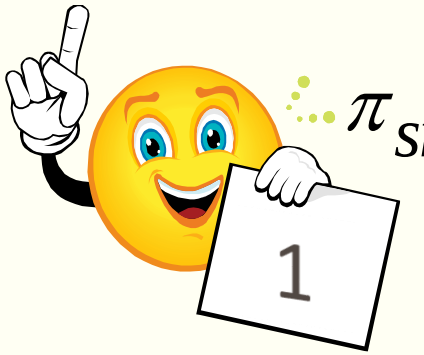
sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

Foreign key

Reserves

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

Which one is better ?



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

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

Find names of sailors who've reserved a red boat

Start with "boat" because we have a predicate about boat

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

1 2

Sailors(sid, sname, rating, age)

Foreign key

Reserves(sid, bid, day)

Foreign key

Boats(bid, bname, color)

2nd join

1st join

Find names of sailors who've reserved a red boat

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

- A more efficient solution:

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

A query optimizer can find this, given the first solution!

Find sailors who've reserved a red or a green boat

- Can identify all red or green boats, then find sailors who've reserved one of these boats:

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

$$\pi_{sname}(Tempboats \bowtie Reserves \bowtie Sailors)$$

❖ Can also define Tempboats using union

$$\rho\left(Tempboats, (\sigma_{color='red'} Boats) \cup \left(\sigma_{color='green'} Boats\right)\right)$$

Find sailors who've reserved a red and a green boat

This won't work!

Empty Relation

$$\rho(\text{Tempboats}, (\sigma_{\text{color='red'} \wedge \text{color='green'}} \text{Boats}))$$

$$\pi_{\text{sname}} (\text{Tempboats} \bowtie \text{Reserves} \bowtie \text{Sailors})$$

Find sailors who've reserved a red and a green boat

- Identify sailors who've reserved red boats

$$\rho\ (Tempred,\ \pi_{sid}((\sigma_{color='red'}\ Boats)\bowtie \mathrm{Re}serves))$$

- Identify sailors who've reserved green boats

$$\rho\ (Tempgreen,\ \pi_{sid}((\sigma_{color='green'}\ Boats)\bowtie \mathrm{Re}serves))$$

- Find the intersection

$$\pi_{sname}((Tempred \cap Tempgreen)\bowtie Sailors)$$

Find the names of sailors who've reserved all boats

- Uses division; schemas of the input relations to division must be carefully chosen:

$$\rho\,(Tempsids, (\pi_{sid,bid} \text{Reserves}) \,/\, (\pi_{bid} Boats))$$

$$\pi_{sname}(Tempsids \bowtie Sailors)$$

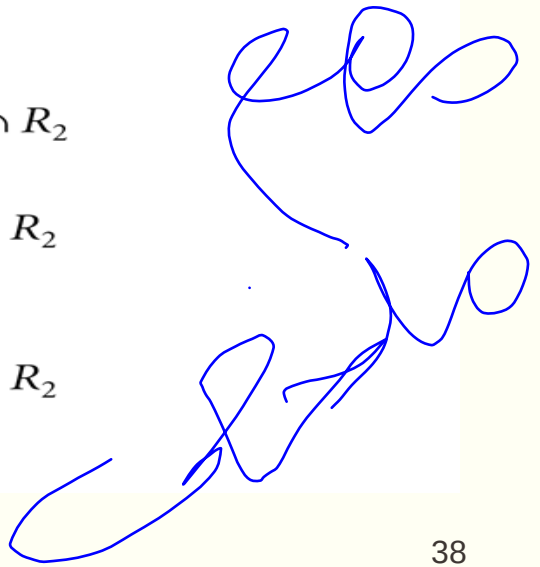
All boats

- To find sailors who've reserved all 'Interlake' boats:

..... $/\pi_{bid}(\sigma_{bname='Interlake'} Boats)$

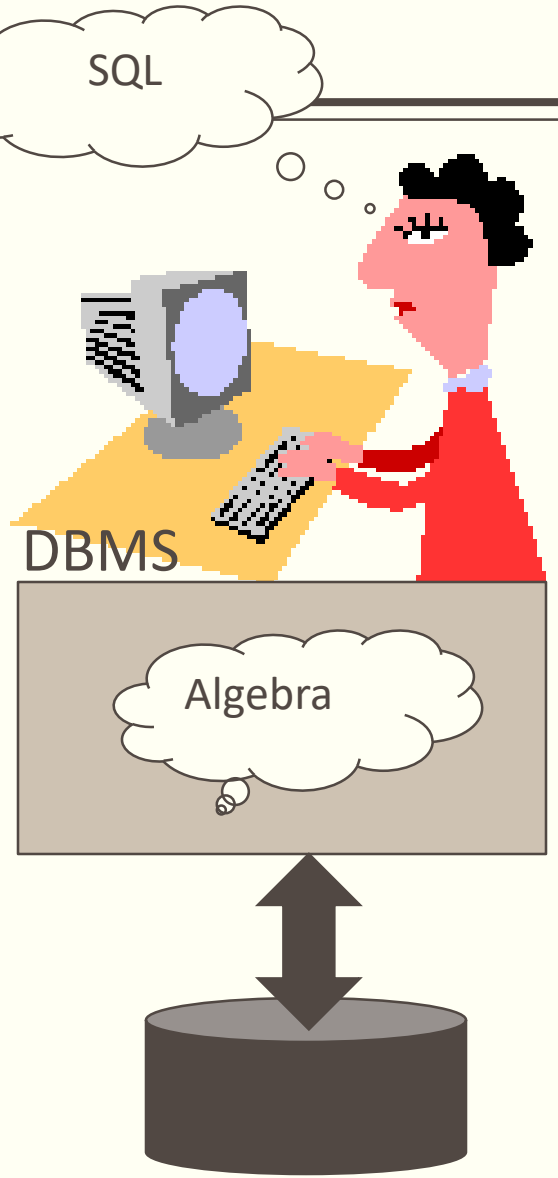
'Interlake' boats

Operation	Purpose	Notation
SELECT	Selects all tuples that satisfy the selection condition from a relation R.	$\sigma_{\text{<selection condition>}}(R)$
PROJECT	Produces a new relation with only some of the attributes of R, and removes duplicate tuples.	$\pi_{\text{<attribute list>}}(R)$
THETA JOIN	Produces all combinations of tuples from R_1 and R_2 that satisfy the join condition.	$R_1 \bowtie_{\text{<join condition>}} R_2$
EQUIJOIN	Produces all the combinations of tuples from R_1 and R_2 that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{\text{<join condition>}} R_2$, OR $R_1 \bowtie_{\text{(<join attributes 1>), (<join attributes 2>)}} R_2$
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of R_2 are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	$R_1 *_{\text{<join condition>}} R_2$, OR $R_1 *_{\text{(<join attributes 1>), (<join attributes 2>)}} R_2$ OR $R_1 * R_2$
UNION	Produces a relation that includes all the tuples in R_1 or R_2 or both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cup R_2$
INTERSECTION	Produces a relation that includes all the tuples in both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cap R_2$
DIFFERENCE	Produces a relation that includes all the tuples in R_1 that are not in R_2; R_1 and R_2 must be union compatible.	$R_1 - R_2$
CARTESIAN PRODUCT	Produces a relation that has the attributes of R_1 and R_2 and includes as tuples all possible combinations of tuples from R_1 and R_2.	$R_1 \times R_2$



Summary

- The relational model has rigorously defined query languages that are simple and powerful.
- Relational algebra is more 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.



Any Question ?