

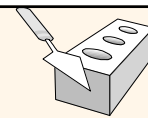
## Joins

❖ Condition Join:  $R \bowtie_c S = \sigma_c(R \times S)$

(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

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

- ❖ *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*.



## Joins (Cont.)

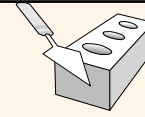
- ❖ Equi-Join: A special case of condition join where the condition  $c$  contains only *equalities*.

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



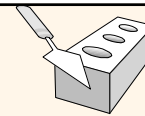
- ❖ Not supported as a primitive operator, but useful for expressing queries like:

*Find sailors who have reserved all boats.*

- ❖ Let  $A$  have 2 fields,  $x$  and  $y$ ;  $B$  have only field  $y$ :

- $A/B = \{ \langle x \rangle \mid \exists \langle x, y \rangle \in A \ \forall \langle y \rangle \in B \}$
- i.e.,  $A/B$  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  $B$ , the  $x$  value is in  $A/B$ .

## 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

$A/B3$

## Expressing $A/B$ Using Basic Operators

- ❖ Division is not essential op; just a useful shorthand.
  - (Also true of joins, but joins are so common that systems implement joins specially.)
- ❖ *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  $A$ .

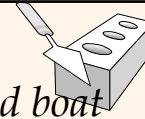
Disqualified  $x$  values:  $\pi_x((\pi_x(A) \times B) - A)$

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

Find names of sailors who've reserved boat #103

- ❖ Solution 1:  $\pi_{sname}((\sigma_{bid=103} Reserves) \bowtie Sailors)$
- ❖ Solution 2:  $\rho(Temp1, \sigma_{bid=103} Reserves)$   
 $\rho(Temp2, Temp1 \bowtie Sailors)$   
 $\pi_{sname}(Temp2)$
- ❖ Solution 3:  $\pi_{sname}(\sigma_{bid=103}(Reserves \bowtie Sailors))$

*Find names of sailors who've reserved a red boat*



❖ Information about boat color only available in Boats; so need an extra join:

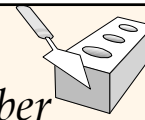
$$\pi_{sname}((\sigma_{color='red'}Boats) \bowtie Reserves \bowtie Sailors)$$

❖ A more efficient solution:

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

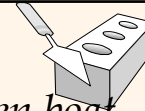
*A query optimizer can find this, given the first solution!*

*Find the color of boats reserved by Lubber*


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

Find the name of sailors who have reserved at least one boat

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

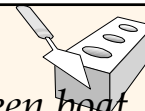


*Find sailors who've reserved a red or a green boat*

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

$$\rho(\text{Tempboats}, (\sigma_{\text{color}='red' \vee \text{color}='green'} \text{Boats}))$$
$$\pi_{\text{sname}}(\text{Tempboats} \bowtie \text{Reserves} \bowtie \text{Sailors})$$

- ❖ Can also define Tempboats using union! (How?)
- ❖ What happens if  $\vee$  is replaced by  $\wedge$  in this query?

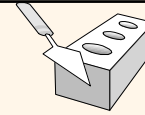


*Find sailors who've reserved a red and a green boat*

- ❖ Previous approach won't work! Must identify sailors who've reserved red boats, sailors who've reserved green boats, then find the intersection (note that *sid* is a key for Sailors):

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

Find the name of sailors who have reserved at least two boats

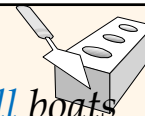


$$\rho(\text{Res}, \pi_{sid, sname, bid}(\text{Sailors} \bowtie \text{Reserves}))$$

$$\rho(\text{Re } spairs \ (1 \rightarrow sid \ 1, 2 \rightarrow sname \ 1, \\ 3 \rightarrow bid \ 1, 4 \rightarrow sid \ 2, 5 \rightarrow sname \ 2, \\ 6 \rightarrow bid \ 2), \text{Re } s \times \text{Re } s)$$

$$\pi_{sname \ 1} \sigma_{(sid \ 1 = sid \ 2) \wedge (bid \ 1 \neq bid \ 2)} \text{Re } spairs$$

Find the names of sailors who've reserved *all* boats



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

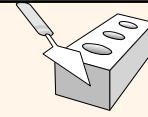
$$\rho(\text{Tempsids}, (\pi_{sid, bid} \text{Reserves}) / (\pi_{bid} \text{Boats}))$$

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

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

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

## *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.