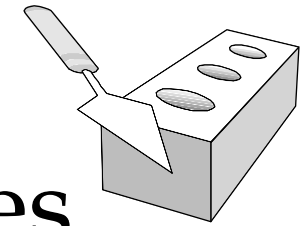
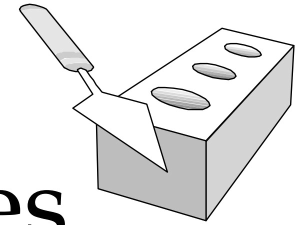
Schema for Example Tables

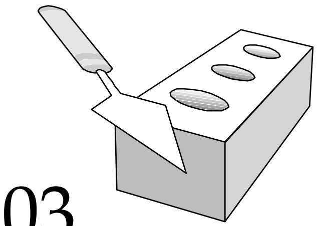


- Sailors: (*<u>sid</u>, sname, rating, age*)
- Boats: (***<u>bid</u>***, *color, bname*)
- Reserves: (*<u>sid, bid, date</u>*)

Schema for Example Tables



- Sailors: (*sid, sname, rating, age*)
- Boats: (***bid***, *color, bname*)
- Reserves: (*sid, bid, date*)



Find names of sailors who've reserved boat #103

- Solution 1: $\pi_{sname}((\sigma_{bid=103} Reserves) \bowtie Sailors)$

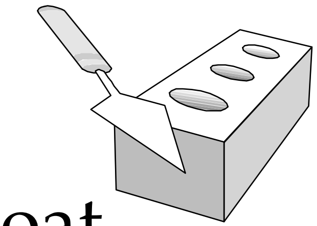
- Same:

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

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

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

- Solution 2: $\pi_{sname}(\sigma_{bid=103}(Reserves \bowtie Sailors))$



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!

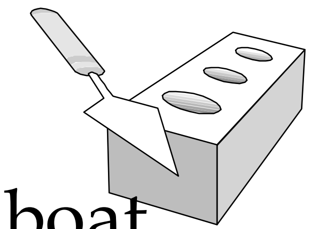
Find sailors who've reserved a red or a green boat

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

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

$$\pi_{sname}(Tempboats \bowtie \mathrm{Re}serves \bowtie Sailors)$$

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



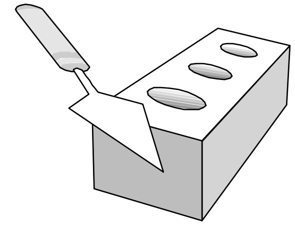
Find sailors who've reserved a red <u>and</u> a green boat

- 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):

$$\rho(Tempred, \pi_{sid}((\sigma_{color=red} Boats) \bowtie \text{Reserves}))$$

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

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



Find the names of sailors who've reserved all boats

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

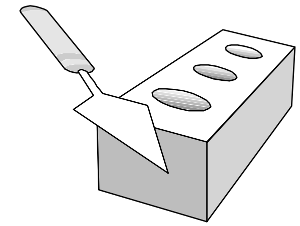
$$\rho(Tempsids, (\pi_{sid,bid} Reserves) / (\pi_{bid} Boats))$$

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

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

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

- What's wrong if $Tempsids = \pi_{sid}(Reserves / \pi_{bid} Boats)$



Banking Example

branch (branch_name, branch_city, assets)

customer (customer_name, customer_street, customer_city)

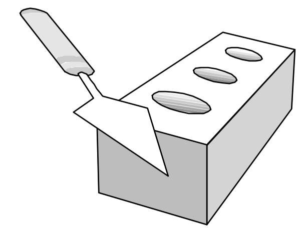
account (account_number, branch_name, balance)

loan (loan_number, branch_name, amount)

depositor (customer_name, account_number)

borrower (customer_name, loan_number)

Example Queries

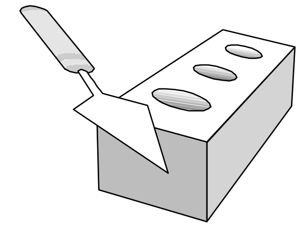


- Find all loans of over $1200

$$\sigma_{amount\ >\ 1200}\ (loan)$$

- Find the loan number for each loan of an amount greater than $1200

$$\Pi_{loan\_number}\ (\sigma_{amount\ >\ 1200}\ (loan))$$



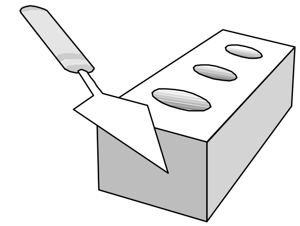
Example Queries

- Find the names of all customers who have a loan, an account, or both, from the bank

$$\Pi_{customer\_name}\,(borrower) \cup \Pi_{customer\_name}\,(depositor)$$

- Find the names of all customers who have a loan and an account at bank.

$$\Pi_{customer\_name}\,(borrower) \cap \Pi_{customer\_name}\,(depositor)$$



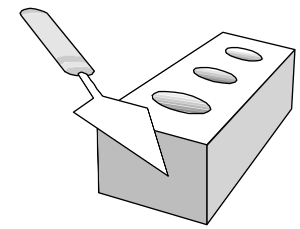
Example Queries

- Find the names of all customers who have a loan at the Perryridge branch.

$$\Pi_{customer\_name}\ (\sigma_{branch\_name=\text{“Perryridge”}}\ (\sigma_{borrower.loan\_number\ =\ loan.loan\_number}(borrower\ x\ loan)))$$

- Find the names of all customers who have a loan at the Perryridge branch but do not have an account at any branch of the bank.

$$\Pi_{customer\_name}\ (\sigma_{branch\_name\ =\ \text{“Perryridge”}}\ (\sigma_{borrower.loan\_number\ =\ loan.loan\_number}(\text{borrower x loan})))\ -\ \Pi_{customer\_name}(\text{depositor})$$



Example Queries

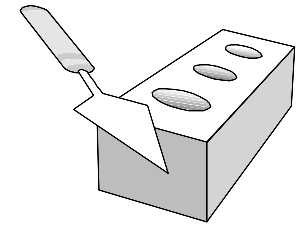
- Find the names of all customers who have a loan at the Perryridge branch.
  - Query 1

$$\Pi_{customer\_name}(\sigma_{branch\_name = \text{“Perryridge”}}(\sigma_{borrower.loan\_number = loan.loan\_number}(borrower \times loan)))$$

  - Query 2

$$\Pi_{customer\_name}(\sigma_{loan.loan\_number = borrower.loan\_number}((\sigma_{branch\_name = \text{“Perryridge”}}(loan)) \times borrower))$$

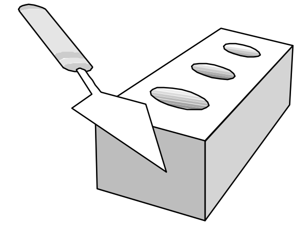
Example Queries



- Find the largest account balance
  - Strategy:
    - Find those balances that are *not* the largest
    - Rename *account* relation as *d* so that we can compare each account balance with all others
    - Use set difference to find those account balances that were *not* found in the earlier step.
  - The query is:

$$\Pi_{balance}(account) - \Pi_{account.balance}(\sigma_{account.balance < d.balance}(account \times \rho_d(account)))$$

Bank Example Queries

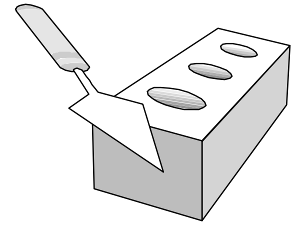


- Find the names of all customers who have a loan and an account at bank.

$$\Pi_{customer\_name}\ (borrower) \cap \Pi_{customer\_name}\ (depositor)$$

- Find the name of all customers who have a loan at the bank and the loan amount

$$\Pi_{customer\text{-}name,\ loan\text{-}number,\ amount}\ (borrower \bowtie loan)$$



Division

- Not supported as a primitive operator, but useful for expressing queries like:
  *Find sailors who have reserved **<u>all</u>** 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 *<u>every</u> 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.
- In general, x and y can be any lists of fields; y is the list of fields in B, and $x \cup y$ is the list of fields of A.

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

pno
p2
p4

B2

pno
p1
p2
p4

B3

sno
s1
s2
s3
s4

A/B1

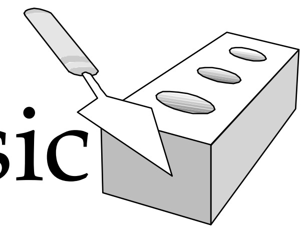
sno
s1
s4

A/B2

sno
s1

A/B3

Expressing A/B Using Basic Operators

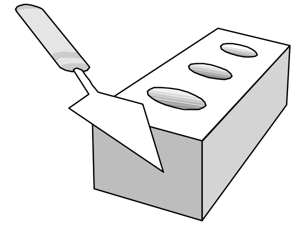


- Division is not essential op; just a useful shorthand.
  - (Also true of joins, but joins are so common that systems implement joins specially.)
- *Idea*: For *A/B*, compute all *x* values that are not \`disqualified' by some *y* value in *B*.
  - *x* value is *disqualified* if by attaching *y* value 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

Bank Example Queries



- Find all customers who have an account from at least the "Downtown" and the Uptown" branches.
  - Query 1

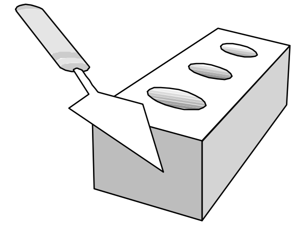
$$\Pi_{customer\_name}(\sigma_{branch\_name = \text{"Downtown"}}(depositor \bowtie account)) \cap$$

$$\Pi_{customer\_name}(\sigma_{branch\_name = \text{"Uptown"}}(depositor \bowtie account))$$

  - Query 2

$$\Pi_{customer\_name,\ branch\_name}(depositor \bowtie account) \div \rho_{temp(branch\_name)}(\{(\text{"Downtown"}), (\text{"Uptown"})\})$$

Note that Query 2 uses a constant relation.



Example Queries

- Find all customers who have an account at all branches located in Brooklyn city.

$$\Pi_{customer\_name,\ branch\_name}(depositor \bowtie account)$$
$$\div\ \Pi_{branch\_name}(\sigma_{branch\_city\ =\ \text{“Brooklyn”}}(branch))$$