

Relational Query Languages

- <u>Query languages</u>: 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.

CS432 Fall 2007

CS432 Fall 2007

Formal Relational Query Languages

- Two mathematical Query Languages form the basis for "real" languages (e.g. SQL), and for implementation:
 - <u>*Relational Algebra*</u>: More operational, very useful for representing execution plans.
 - <u>Relational Calculus</u>: Lets users describe what they want, rather than how to compute it. (Nonoperational, <u>declarative</u>.)

CS432 Fall 2007

Preliminaries

- A query is applied to *relation instances*, and the result of a query is also a relation instance.
 - *Schemas* of input relations for a query are fixed (but query will run regardless of instance!)
 - The schema for the *result* of a given query is also fixed! Determined by definition of query language constructs.
- Positional vs. named-field notation:
 - Positional notation easier for formal definitions, named-field notation more readable.
 - Both used in SQL

CS432 Fall 2007

Example Instances	R1	22	<u>bid</u> 101 103	10/1	<u>ay</u> 10/96 12/96
 "Sailors" and "Reserves" relations for our examples. S1 	<u>sid</u>	sname	e rat	ing	age
 We'll use positional or 	22	dustin	1 7	7	45.0
named field notation,	31	lubbe	r 8	3	55.5
assume that names of fields	58	rusty	1	10	35.0
in query results are `inherited' from names of fields in query input relations.	<u>sid</u> 28 31 44 58	sname yuppy lubbe guppy rusty	r 8	3	age 35.0 55.5 35.0 35.0
CS432 Fall 2007					5

Relational Algebra

✤ Basic operations:

- <u>Selection</u> (σ) Selects a subset of rows from relation.
- <u>Projection</u> (π) Deletes unwanted columns from relation.
- <u>*Cross-product*</u> (X) Allows us to combine two relations.
- <u>Set-difference</u> (—) Tuples in reln. 1, but not in reln. 2.
- <u>Union</u> (\bigcup) Tuples in reln. 1 and in reln. 2.
- * Additional operations:
 - Intersection, <u>join</u>, division, renaming: Not essential, but (very!) useful.
- Since each operation returns a relation, operations can be composed! (Algebra is "closed".) CS432 Fall 2007

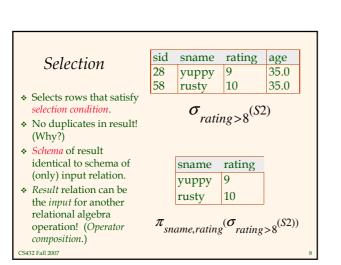
Projection

CS432 Fall 2007

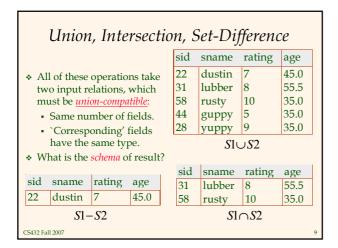
- Deletes attributes that are not in projection list.
- Schema of result contains exactly the fields in the projection list, with the same names that they had in the (only) input relation.
- Projection operator has to eliminate *duplicates*! (Why??)
 - Note: real systems typically don't do duplicate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

sname	rating					
yuppy	9					
lubber	8					
guppy	5					
rusty	10					
$\pi_{sname,rating}(S2)$						

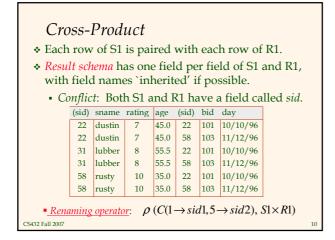
age 35.0 55.5 $\pi_{age}(S2)$







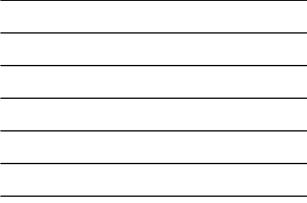






,	Joins * <u>Condition Join</u> : $R \bowtie_c S = \sigma_c (R \times S)$						
(sid)	sname	rating	age	(sid)	bid	day	
22	dustin	7	45.0	58	103	11/12/96	
31	lubber	8	55.5	58	103	11/12/96	
* Pa	$S1 \bowtie_{S1.sid < R1.sid} R1$						
	 <i>Result schema</i> same as that of cross-product. 						
	 Fewer tuples than cross-product, might be able to compute more efficiently 						
✤ Soi 5432 Fall 2007	• Sometimes called a <i>theta-join</i> .						

J	oins							
		-				i join where		
the	sid	tion <i>c</i> co sname	rating	age	bid	dav		
	22	dustin	7	45.0	101	5		
	22 58	rusty	10	45.0 35.0		10/10/96		
	L		$S1 \bowtie_{si}$, <i>R</i> 1				
Sta sta similar to cross-product, but only								
one copy of fields for which equality is specified.								
✤ <u>Natural Join</u> : Equijoin on all common fields.								
CS432 Fall 2	2007		-				12	



Division

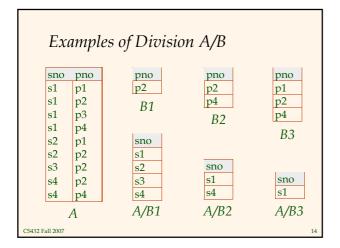
 Not supported as a primitive operator, but useful for expressing queries like:

Find sailors who have reserved <u>all</u> boats.

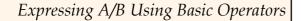
✤ Let A have 2 fields, x and y; B have only field y:

• $A/B = \{ \langle x \rangle \mid \exists \langle x, y \rangle \in A \ \forall \langle y \rangle \in B \}$

- i.e., *A/B* contains all *x* tuples (sailors) such that for *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*.
- * 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*. CS432 Fall2007





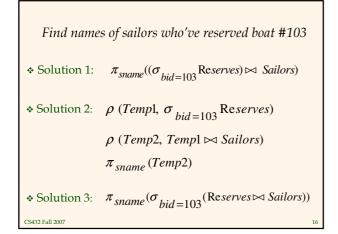


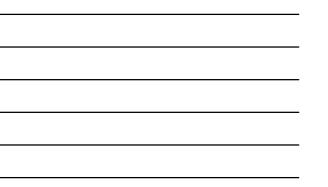
- Division is not essential op; just a useful shorthand.
 (Also true of joins, but joins are so common that systems implement joins specially.)
- *Idea*: For *A/B*, compute all *x* values that are not `disqualified' by some *y* value in *B*.
 - *x* value is *disqualified* if by attaching *y* value from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

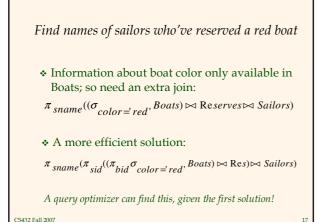
Disqualified *x* values: $\pi_{\chi}((\pi_{\chi}(A) \times B) - A)$

A/B: $\pi_{\chi}(A)$ – all disqualified tuples

CS432 Fall 2007





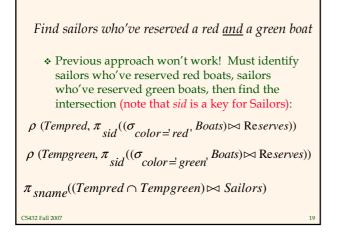


Find sailors who've reserved a red or a green boat

- Can identify all red or green boats, then find sailors who've reserved one of these boats:
 - $\rho \ (\textit{Tempboats}, (\sigma_{\textit{color}='\textit{red'}} \lor \textit{color}='\textit{green'} \ \textit{Boats}))$
 - π_{sname} (Tempboats \bowtie Reserves \bowtie Sailors)
- Can also define Tempboats using union! (How?)

CS432 Fall 2007

* What happens if \vee is replaced by \wedge in this query?



Summary

CS432 Fall 2007

- 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.

CS432 Fall 2007