

Relational Algebra

Chapter 4, Part A

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke

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.

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke

2

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>.)

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke



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

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke

Example Instances

				6
R1	sid	bid	day	
	22	101	10/10/96	
	58	103	11/12/96	

- "Sailors" and "Reserves" relations for our examples. S1
- 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.

	36 10)3	11/1	2/90
sid	sname	rat	ing	age
22	dustin	7	7	45.0
31	lubber	8	3	55.5
58	rusty	1	0	35.0

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

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke

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> (∪) 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".)

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke



- 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?)

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke

	J.	6
sname	rating	TY.
yuppy	9	$\overline{}$
lubber	8	
guppy	5	
rusty	10	

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

age 35.0 55.5

 $\pi_{age}(S2)$

Selection

- Selects rows that satisfy selection condition.
- No duplicates in result! (Why?)
- Schema of result identical to schema of (only) input relation.
- Result relation can be the input for another relational algebra operation! (Operator composition.)

			7	S
sid	sname	rating	age	1
28	yuppy	9	35.0	7
58	rusty	10	35.0	

 $\sigma_{rating>8} (S2)$

sname	rating
yuppy	9
rustv	10

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

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke

Union, Intersection, Set-Difference

- All of these operations take two input relations, which must be <u>union-compatible</u>:
 - Same number of fields.
 - `Corresponding' fields
- have the same type.
- * What is the *schema* of result?

sid	sname	rating	age
22	dustin	7	45.0

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

 $S1 \cup S2$

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

 $S1 \cap S2$

S1-S2

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke

Cross-Product

- ❖ Each row of S1 is paired with each row of R1.
- * 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.

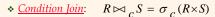
(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rustv	10	35.0	58	103	11/12/96

• Renaming operator: $\rho(C(1 \rightarrow sid1, 5 \rightarrow sid2), S1 \times R1)$

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke

10

Joins



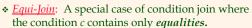
(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	58		11/12/96
31	lubber	8	55.5	58	103	11/12/96

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

- * 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*.

11

Joins



sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/96
58	rustv	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.
- * <u>Natural Join</u>: Equijoin on *all* common fields.

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke







 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*:
 - $A/B = \{\langle x \rangle | \forall \langle y \rangle \in B \exists \langle x, y \rangle \in A \}$

Division

- 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.

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke

13

Examples of Division A/B pno pno pno sno pno p2 s1p2 p1 p1 p4 s1 p2 p2 B1 s1 p3 p4 B2 s1 p4 В3 s2 p1 sno p2 s2 s1 s3p2 s2 sno p2 sno s4 s3s1 p4 A/B1 *A/B2* A/B3Α Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke

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:

A/B:

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke

Find names of sailors who've reserved boat #103

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

* Solution 2:
$$\rho$$
 (Templ, $\sigma_{bid=103}$ Reserves)

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

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

* Solution 3:
$$\pi_{sname}(\sigma_{bid=103}(\text{Reserves}\bowtie Sailors))$$

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke

Find names of sailors who've reserved a red boat

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

$$\pi_{sname}((\sigma_{color='red}, Boats) \bowtie \mathsf{Re}\, serves \bowtie Sailors)$$

* A more efficient solution:

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

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

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke

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

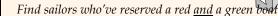
Can identify all red or green boats, then find sailors who've reserved one of these boats:

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

 π_{sname} (Temphoats \bowtie Reserves \bowtie Sailors)

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

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke



- 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):
- $\rho~(Tempred,~\pi_{sid}((\sigma_{color='red'}^{}Boats)\bowtie Reserves))$
- $\rho \; (\textit{Tempgreen}, \pi_{\textit{sid}} ((\sigma_{\textit{color} = \textit{green}}, \textit{Boats}) \bowtie \mathsf{Reserves}))$
- $\pi_{sname}((Tempred \cap Tempgreen) \bowtie Sailors)$

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke

19

Find the names of sailors who've reserved all boats

- Uses division; schemas of the input relations to / must be carefully chosen:
 - $\rho~(Tempsids, (\pi_{sid,bid}^{\rm Re} serves) \, / \, (\pi_{bid}^{\rm Boats}))$
 - π_{sname} (Tempsids \bowtie Sailors)
- * To find sailors who've reserved all 'Interlake' boats:

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

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke

20

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.

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke