

## Relational Algebra

R & G, Chapter 4

By relieving the brain of all unnecessary work, a good notation sets it free to concentrate on more advanced problems, and, in effect, *increases the mental power of the race.*

-- Alfred North Whitehead (1861 - 1947)



## 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  $\neq$  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-procedural, declarative.)

- Understanding Algebra & Calculus is key to understanding SQL, query processing!

## 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 over any legal instance)
  - The schema for the result of a given query is also fixed. It is determined by the definitions of the query language constructs.
- Positional vs. named-field notation:
  - Positional notation easier for formal definitions, named-field notation more readable.
  - Both used in SQL
    - Though positional notation is not encouraged

## Relational Algebra: 5 Basic Operations

- **Selection** ( $\sigma$ ) Selects a subset of rows from relation (horizontal).
- **Projection** ( $\pi$ ) Retains only wanted columns from relation (vertical).
- **Cross-product** ( $\times$ ) Allows us to combine two relations.
- **Set-difference** ( $-$ ) Tuples in r1, but not in r2.
- **Union** ( $\cup$ ) Tuples in r1 and/or in r2.

Since each operation returns a relation, operations can be composed! (Algebra is "closed".)

## Example Instances

			r1		
bid	bname	color	sid	bid	day
101	Interlake	blue	22	101	10/10/96
102	Interlake	red	58	103	11/12/96
103	Clipper	green			
104	Marine	red			

Boats

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

## Projection

- Examples:  $\pi_{age}(S2)$ ;  $\pi_{sname, rating}(S2)$
- Retains only attributes that are in the "projection list".
- Schema of result:
  - exactly the fields in the projection list, with the same names that they had in the input relation.
- Projection operator has to **eliminate duplicates** (How do they arise? Why remove them?)
  - Note: real systems typically don't do duplicate elimination unless the user explicitly asks for it. (Why not?)

## 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

S2

sname	rating
yuppy	9
lubber	8
guppy	5
rusty	10

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

age
35.0
55.5

$\pi_{age}(S2)$

## Selection ( $\sigma$ )

- Selects rows that satisfy **selection condition**.
- Result is a relation.
  - Schema of result is same as that of the input relation.
- Do we need to do duplicate elimination?

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

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

sname	rating
yuppy	9
rusty	10

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

## Union and 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.
- For which, if any, is duplicate elimination required?

## Union

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

S1

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
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S2

## Set Difference

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

S1

sid	sname	rating	age
22	dustin	7	45.0

$S1 - S2$

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

S2

sid	sname	rating	age
28	yuppy	9	35.0
44	guppy	5	35.0

$S2 - S1$

## Cross-Product

- $S1 \times R1$ : Each row of  $S1$  paired with each row of  $R1$ .
- Q: How many rows in the result?
- **Result schema** has one field per field of  $S1$  and  $R1$ , with field names "inherited" if possible.
  - May have a naming conflict: Both  $S1$  and  $R1$  have a field with the same name.
  - In this case, can use the renaming operator:  $\rho(C1 \rightarrow sid1, S \rightarrow sid2), S1 \times R1$

## Cross Product Example

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

R1

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

S1

$R1 \times S1 =$

(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

## Compound Operator: Intersection

- In addition to the 5 basic operators, there are several additional "Compound Operators"
  - These add no computational power to the language, but are useful shorthands.
  - Can be expressed solely with the basic ops.
- Intersection takes two input relations, which must be **union-compatible**.
- Q: How to express it using basic operators?
 
$$R \cap S = R - (R - S)$$

## Intersection

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

S1

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

S2

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

$S1 \cap S2$

## Compound Operator: Join

- Joins are compound operators involving cross product, selection, and (sometimes) projection.
- Most common type of join is a "**natural join**" (often just called "join").  $R \bowtie S$  conceptually is:
  - Compute  $R \times S$
  - Select rows where attributes that appear in both relations have equal values
  - Project all unique attributes and one copy of each of the common ones.
- Note: Usually done much more efficiently than this.
- Useful for putting "normalized" relations back together.

## Natural Join Example

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

R1

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

S1

$R1 \bowtie S1 =$

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