Relational Algebra: Part 1

Relational Algebra and Calculus

- Two mathematical Query Languages are the basis for SQL and it's implementation
- Relational Algebra: More operational, useful for
  - representing execution plans, a foundation for SQL execution
- Relational Calculus: Lets users describe what they want, rather than how to compute it. (Non-operational, declarative.)
  - SQL is the usable result from Relational Calculus

Preliminaries

- A query is applied to relation instances (i.e. current "value" of the relation), and the result of a query is also a relation instance.
  - Schemas of input relations for a query are fixed
  - The schema for the result of a given query is also pre-determined by the query definition

Example Instances

R1

sid	bid	day
22	101	1996-10-10
58	103	1996-11-12

S1

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55.5
58	rusty	10	35

S2

sid	sname	rating	age
31	lubber	8	55.5
58	rusty	10	35
28	yuppy	9	35
44	guppy	5	35

Algebraic Operations

- Basic operations:
  - Selection (σ) Selects a subset of rows from relation.
  - Projection (π) Deletes unwanted columns from relation.
  - Cross-product (x) Allows us to combine two relations.
  - Set-difference (-) Tuples in relation 1, but not in relation 2.
  - Union (∪) Tuples in relation1 and in relation 2
- Additional operations:
  - Intersection(∩), join(⋈), division(/), renaming: not essential
- Since each operation returns a relation operations can be composed

Projection

- 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 input relation
- Projection operator has to eliminate duplicates.
  - In practice, systems don't do duplicate elimination unless asked

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

sname	rating
lubber	8
rusty	10
yuppy	9
guppy	5

$\pi_{\text{age}}(S2)$

age
55.5
35

sid	sname	rating	age
31	lubber	8	55.5
58	rusty	10	35
28	yuppy	9	35
44	guppy	5	35

Selection

- Selects rows that satisfy selection condition.
- *Schema* of result identical to schema of input relation.
- *Result* relation can be the *input* for another relational algebra operation

sid	sname	rating	age
31	lubber	8	55.5
58	rusty	10	35
28	yuppy	9	35
44	guppy	5	35

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

sid	sname	rating	age
58	rusty	10	35
28	yuppy	9	35

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

sname	rating
rusty	10
yuppy	9

Union/Intersection/Minus

- All of these operations take two input relations, which must be union-compatible:
  - Same number of fields.
  - Corresponding fields have the same type.

S1∪ S2

sid	sname	rating	age
31	lubber	8	55.5
22	dustin	7	45
58	rusty	10	35
44	guppy	5	35
28	yuppy	9	35

S1∩ S2

sid	sname	rating	age
31	lubber	8	55.5
58	rusty	10	35

S1- S2

sid	sname	rating	age
22	dustin	7	45

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.
  - Renaming operation helps here: ρ(C(1->sid1, 5->sid2),S1 x R1)

sid	sname	rating	age	sid	bid	day
22	dustin	7	45	22	101	1996-10-10
31	lubber	8	55.5	22	101	1996-10-10
58	rusty	10	35	22	101	1996-10-10
22	dustin	7	45	58	103	1996-11-12
31	lubber	8	55.5	58	103	1996-11-12
58	rusty	10	35	58	103	1996-11-12

Joins

- Condition (θ) Join: $R \bowtie_c S = \sigma_c (R \times S)$
- Result schema is same as that of cross product

$S1 \bowtie_{(s1.sid < R1.sid)} R1$

sid	sname	rating	age	sid	bid	day
~~22~~	~~dustin~~	~~7~~	~~45~~	~~22~~	~~101~~	~~1996-10-10~~
~~31~~	~~lubber~~	~~8~~	~~55.5~~	~~22~~	~~101~~	~~1996-10-10~~
~~58~~	~~rusty~~	~~10~~	~~35~~	~~22~~	~~101~~	~~1996-10-10~~
22	dustin	7	45	58	103	1996-11-12
31	lubber	8	55.5	58	103	1996-11-12
~~58~~	~~rusty~~	~~10~~	~~35~~	~~58~~	~~103~~	~~1996-11-12~~

Joins

- Equi-join: special case where condition *c* contains only equality conjuncts ("and")
  - Expressed as S1 $\bowtie_{sid}$ R1. Implies: S1.sid = R1.sid
- Result schema: same as join but only one copy of *sid*
- Natural Join: Equi-join on all *common fields*

S1 $\bowtie_{sid}$ R1

sid	sname	rating	age	bid	day
22	dustin	7	45	101	1996-10-10
58	rusty	10	35	103	1996-11-12

“Division”

- Find sailors who have reserved all boats
- Let A have 2 fields, x and y; B have only field y:
- $A/B = \{x \mid \exists\, x, y \in A\ \forall\, y \in B\}$
  - 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.

Thinking about A/B

- 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
  - for an $x \in \pi_x(A)$ to qualify we need $\{x\} \times B$ to be in A
  - So if any $\{x\} \times B$ is not in A then that x value gets disqualified
  - Thus all disqualified x values: $\pi_x(\pi_x(A) \times B - A)$
  - A/B : $\pi_x(A) - \pi_x(\pi_x(A) \times B - A)$

Why “Division”?

- Integers:
  - a/b = c means something like “c is the largest integer such that b * c <= a
- Relations:
  - A/B = C means something like “C is the largest relation such that B X C <= A”
    - (here for relations, “<=“ means set containment.)

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.