Relational Algebra
(Chapter 4)

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-operational, *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.
  - The schema for the *result* of a given query is fixed! Determined by definition of query language constructs.

Example Instances

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

R1

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

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

Relational Algebra

- Basic operations:
  - 행 - *Selection* (σ) Selects a subset of rows from relation.
  - 열 - *Projection* (π) Deletes unwanted columns from relation.
  - (나) 곱집합 - *Cross-product* (×) Allows us to combine two relations. 가능한 모든 조합
  - 차 - *Set-difference* (−) Tuples in reln. 1, but not in reln. 2.
  - 합 - *Union* (∪) Tuples in reln. 1 or in reln. 2.
- Additional operations:
  - Intersection, *join*, division, renaming: Not essential, but (very!) useful.
- Since each operation returns a relation, operations can be *composed*! (Algebra is "closed".)

Projection (π)

열 추출 → 반환

- Extract attributes that are 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??)

  → 결과테이블의 row 수가 기존 것보다 줄어들 수 있음!

  - 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

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

같은 뜻 표현

$\pi_{\text{sname, rating}}$ Sailors

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

age
35.0
55.5

→ 중복제거결과

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

π_{age} Sailors

Selection (σ)

행추출

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

sid	sname	rating	age
28	yuppy	9	35.0
58	rusty	10	35.0

테이블이름

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

sname	rating
yuppy	9
rusty	10

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

→ projection 후에 중복 record가 생겼는지 확인

녹음없음 스터디에서 확인

∪ ∩ −

Union, Intersection, Set-Difference

column의 이름까지 같을 필요 x (호환가능)
개수와 type은 같아야함 (schema)

- All of these operations take two input relations, which must be *union-compatible*:
  - 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

$S1 - S2$

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	rusty	10	35.0

$S1 \cap S2$

Cross-Product (×)

- Each row of S1 is paired with each row of R1.
- *Result schema* has one field per field of S1 and R1.

$$R1 \times S1$$

$X = \{a, b\}$

$Y = \{1, 2, 4\}$

$\Rightarrow X \times Y = \{(a,1), (a,2), (a,4), (b,1), (b,2), (b,4)\}$

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

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

Cross-Product

- Each row of S1 is paired with each row of R1.
- *Result schema* has one field per field of S1 and R1.

⇒ 모두다나타나야함

column의 개수는
R1+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

순서는상관없음
order by로 지정

Joins

- *Conditional Join*: $R \bowtie_c S = \sigma_c (R \times S)$

→ condition에을 만족하는 것만 select

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

SELECT FROM ~
↳ cross product

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

relation schema 정보가 꼭!

- *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*.

Joins

condition 표현 방법
- 논리곱(AND, ∧)
- 논리합(OR, ∨)
- 부정(NOT, ~)

- *Equi-Join*: A special case of condition join where the condition *c* contains only ***equalities.***

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

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

→ 같은경우만

- *Result schema* similar to cross-product, but only one copy of fields for which equality is specified.

Joins

- *Natural Join*: Equijoin on *all* common fields. (on two tables

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

$$S1 \bowtie R1$$

- *Result schema* similar to cross-product, but only one copy of fields which appear in both relations.

Find names of sailors who've reserved boat #103

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

  얘가 낫다

- 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:

$$\pi_{sname}((\sigma_{color='red'} Boats) \bowtie \text{Reserves} \bowtie Sailors)$$

√ - A more efficient solution:

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

- *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:

$$\pi_{sname}(((\sigma_{color='red' \vee color='green'} Boats) \bowtie Reserves) \bowtie Sailors)$$

- What happens if $\overset{합}{\vee}$ is replaced by $\overset{곱}{\wedge}$ in this query?

→ 불가능!

Boat의 color가 red이면서 green일 수 없음!

Summary

- The relational model has 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.