Relational Algebra and Calculus

UVic C SC 370

Dr. Daniel M. German

Department of Computer Science

University of Victoria

May 27, 2003 Version: 1.1.1

Overview

- The mathematical foundation of query languages such as SQL
- Relational Algebra and Calculus, and why they are important
- Basic algebra operators

Preliminaries

- The Examples in this chapter will use the following schema

  Sailors (sid: **`integer`**, sname: **`string`**, rating: **`integer`**, age: **`read`**)

  Boats(bid: **`integer`**, bname: **`string`**, color: **`string`**)

  Reserves(sid: **`integer`**, bid: **`integer`**, day: **`date`**)

Instances used

sid	*sname*	*rating*	*age*
22	Dustin	7	45
31	Lubber	8	55.5
58	Rusty	10	35

(a) Instance S1 of Sailors

sid	*sname*	*rating*	*age*
28	yuppy	9	35
31	Lubber	8	55.5
44	guppy	5	35
58	Rusty	10	35

(b) Instance S2 of Sailors

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

(c) Instance R1 of Reserves

bid	*bname*	*color*
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

(d) Instance B1 of Boats

Relational Algebra

- **Relational algebra** (RA) is a **query language** associated with the **relational model**
- Every operator in RA takes one or two relations as parameters and return a relation
- A **relational algebra expression** is recursively defined to be a
  - **relation**,
  - a **unary algebra operator** applied to a **single** expression, or
  - a **binary algebra operator** applied to **two** expressions
- Basic operators:
  - Selection, projection, union, cross-product, and difference
- Procedural

Selection and Projection

- σ: **selects** rows from a table
- Examples:

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

sid	*sname*	*rating*	*age*
28	yuppy	9	35
58	Rusty	10	35

- The selection operation uses a **selection condition**
- It is usually a boolean expression: $<, <=, =, \neq, >=, >, \wedge, \vee$
- Reference to an attribute: by position ($relation.i$, i) or by name ($relation.name$, $name$)

Projection

- Allows us to extract columns from a relation
- Examples:

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

sname	*rating*
yuppy	9
Lubber	8
guppy	5
Rusty	10

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

age
35
55.5

Combining Both

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

sname	*rating*
yuppy	9
Rusty	10

Union

- $R \cup S$ returns a relation which is the set union of R and S
- R and S should be union compatible:
  - They have the same number of fields
  - The corresponding fields have the same **domains**
- For convenience we assume the result inherits names from R (the schema of $R \cup S$ is the schema of R)

$$S1 \cup S2$$

sid	*sname*	*rating*	*age*
22	Dustin	7	45
28	yuppy	9	35
31	Lubber	8	55.5
44	guppy	5	35
58	Rusty	10	35

Intersection

- $R \cap S$ returns a relation which contains the tuples that are both in R and S
- R and S should be **union compatible**

$$S1 \cap S2$$

sid	*sname*	*rating*	*age*
31	Lubber	8	55.5
58	Rusty	10	35

Set Difference

- $R - S$ returns a relation which contains the tuples that are both in R but not in S
- R and S should be **union compatible**

$$S1 - S2$$

sid	*sname*	*rating*	*age*
22	Dustin	7	45

Cross Product

- $R \times S$ returns a relation instance whose schema contains all fields of R (in the same order as in R) followed by all the fields in S (in the same order as in S)
- The result contains one tuple $\langle r, s \rangle$ (concatenation of tuples r and s) for each pair of tuples $r \in R, s \in S$

$$S1 \times R1$$

(sid)	*sname*	*rating*	*age*	*(sid)*	*bid*	*day*
22	Dustin	7	45	22	101	1996-10-10
22	Dustin	7	45	58	103	1996-11-12
31	Lubber	8	55.5	22	101	1996-10-10
31	Lubber	8	55.5	58	103	1996-11-12
58	Rusty	10	35	22	101	1996-10-10
58	Rusty	10	35	58	103	1996-11-12

Renaming

- When operating in more than one table, name conflicts can arise
- **renaming** operator ρ renames the fields of a relation
- $\rho(R(\bar{F})E)$ takes an expression E and returns a new instance relation called R
- R contains same columns as E, but some fields are renamed

Renaming

- R contains same columns as E, but some fields are renamed
- Example: $\rho(C(1 \rightarrow sid1, 5 \rightarrow sid2), S1 \times R1)$ returns:

sid1	*sname*	*rating*	*age*	*sid2*	*bid*	*day*
22	Dustin	7	45	22	101	1996-10-10
22	Dustin	7	45	58	103	1996-11-12
31	Lubber	8	55.5	22	101	1996-10-10
31	Lubber	8	55.5	58	103	1996-11-12
58	Rusty	10	35	22	101	1996-10-10
58	Rusty	10	35	58	103	1996-11-12

- With schema:

  C (sid1: **`integer`**, sname: **`string`**, rating: **`integer`**, age: **`read`**, sid2: **`integer`**, bid: **`integer`**, day: **`date`**)

Other operators

- As in set theory, other operators can be added by combining the current ones
- Even $\cap$ is redundant: $R \cap S = R - (R - S)$

Joins

- The **join** operation is the most useful operation in relational algebra
- It can be defined with cross product and selection, projection
- It is important to do joins without **materializing** the cross product
- There exist several variants of joins:
  - Condition Join
  - Equijoin
  - Natural Join

Condition Joins

- Essentially, a select of a cross product:

$$R \bowtie_c S = \sigma_c(R \times S)$$

- Example: $S1 \bowtie_{S1.sid<R1.sid} R1$

(sid)	*sname*	*rating*	*age*	*(sid)*	*bid*	*day*
22	Dustin	7	45	58	103	1996-11-12
31	Lubber	8	55.5	58	103	1996-11-12

Equijoin

- A special case of join when the join condition consists **only** of a conjunction of **equalities** of the form $R.name1 = S.name2$
- There is redundancy in keeping both attributes in the relation, so $S.name2$ is dropped
- Example: $S1 \bowtie_{S1.sid=R1.sid} R1$

sid	*sname*	*rating*	*age*	*bid*	*day*
22	Dustin	7	45	101	1996-10-10
58	Rusty	10	35	103	1996-11-12

Natural Join

- A **natural join** is a **equijoin** in which equalities are specified on **all** fields that have the same name
- In this case we simply omit the condition
- The result **does not have repeated field names**
- Example: $S1 \bowtie R1 = S1 \bowtie_{S1.sid=R1.sid} R1$

sid	*sname*	*rating*	*age*	*bid*	*day*
22	Dustin	7	45	101	1996-10-10
58	Rusty	10	35	103	1996-11-12

- If the two relations have no common attributes, then, the result is the cross product

Division

- It is a complex operator
- Useful in situations such as: *find the sailors who have reserved all the boats*
- For relations A and B, A/B is the largest relation such that $(A/B) \times B \subseteq A$
- Definition:

$$A/B = \{\langle x \rangle | \forall y \; s.t. \; \langle y \rangle \in B, \exists \langle x, y \rangle \in A\}$$

Examples of Division

- Bi are parts
- A are the suppliers and the parts the supply
- A/Bi are those suppliers who supply **all** parts listed in Bi

A

sno	*pno*
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B1

pno
p2

B2

pno
p2
p4

B3

pno
p1
p2
p4

$A/B1$

sno
s1
s2
s3
s4

$A/B2$

sno
s1
s4

$A/B3$

sno
s1

Relational Calculus

- It is an alternative to relational algebra
- Declarative
- The variant or Relational Calculus here presented is TRC (tuple relational calculus)

Tuple Relational Calculus

- A **tuple variable** takes as a value a set of tuples with the same relation schema
- A TRC query has the form: $\{T|\ p(T)\}$
- T is a tuple variable bound by the predicate $p(T)$
- TRC is a subset of first order logic
- Example: *find all sailors with a ranking above 7*

$$\{S|S \in Sailors \wedge S.Rating > 7\}$$

Formally

- Let Rel be a relation name, R and S tuple variables with attributes R.a, S.b correspondingly
- op denotes an operator in the set $\{<, >, =, \leq, \geq, \neq\}$
- An **atomic formula** is one of the following:
  - $R \in Rel$
  - $R.a$ op $S.b$
  - $R.a$ op $constant$ or $constant$ op $R.a$

Formally...

- A **formula** is recursively defined as one of the following, where p and q are themselves formulae:
  - any atomic formula
  - $\neg p$, $p \wedge q$, $p \vee q$, $p \implies q$
  - $\exists R(p(R))$, where R is a tuple
  - $\forall R(p(R))$, where R is a tuple
- Semantics of TRC
  - The **answer** to a TRC query $\{T | p(T)\}$ is the set of all tuples t for which $p(T)$ is true

Examples

- *Find the sailor name, boat id, and reservation date for each reservation*

$$\{P \mid \exists R \in Reserves \exists S \in Sailors$$
$$(R.sid = S.sid \wedge$$
$$P.bid = R.bid \wedge P.day = R.day \wedge P.sname = S.sname)\}$$

sname	*bid*	*day*
Dustin	101	1996-10-10
Rusty	103	1996-11-12

Examples...

- *Find the names of sailors who have reserved boat 103*

$$\{P | \exists R \in Reserves \exists S \in Sailors$$
$$(R.sid = S.sid \land R.bid = 103 \land$$
$$P.sname = S.sname)\}$$

sname	*bid*	*day*
Dustin	101	1996-10-10
Rusty	103	1996-11-12

Examples...

- *Find the names of sailors who have reserved all boats*

$$\{P | \exists S \in Sailors \quad \forall B \in Boats$$
$$(\exists R \in Reserves(R.sid = S.sid \wedge B.bid = R.bid \wedge$$
$$P.sname = S.sname))\}$$

- "Find sailors S such that for all boats B there is a Reserves tuple showing that sailor S has reserved boat B"
- This query is equivalent to the division operator:

$$\rho(Tempsids, (\pi_{sid,bid} Reserves)/(\pi_{bid} Boats))$$

$$\pi_{sname}(Tempsids \bowtie Sailors)$$

Further Reading

- 4.2.6 Examples of Relational Algebra Queries
- 4.3.1 Examples of TRC queries