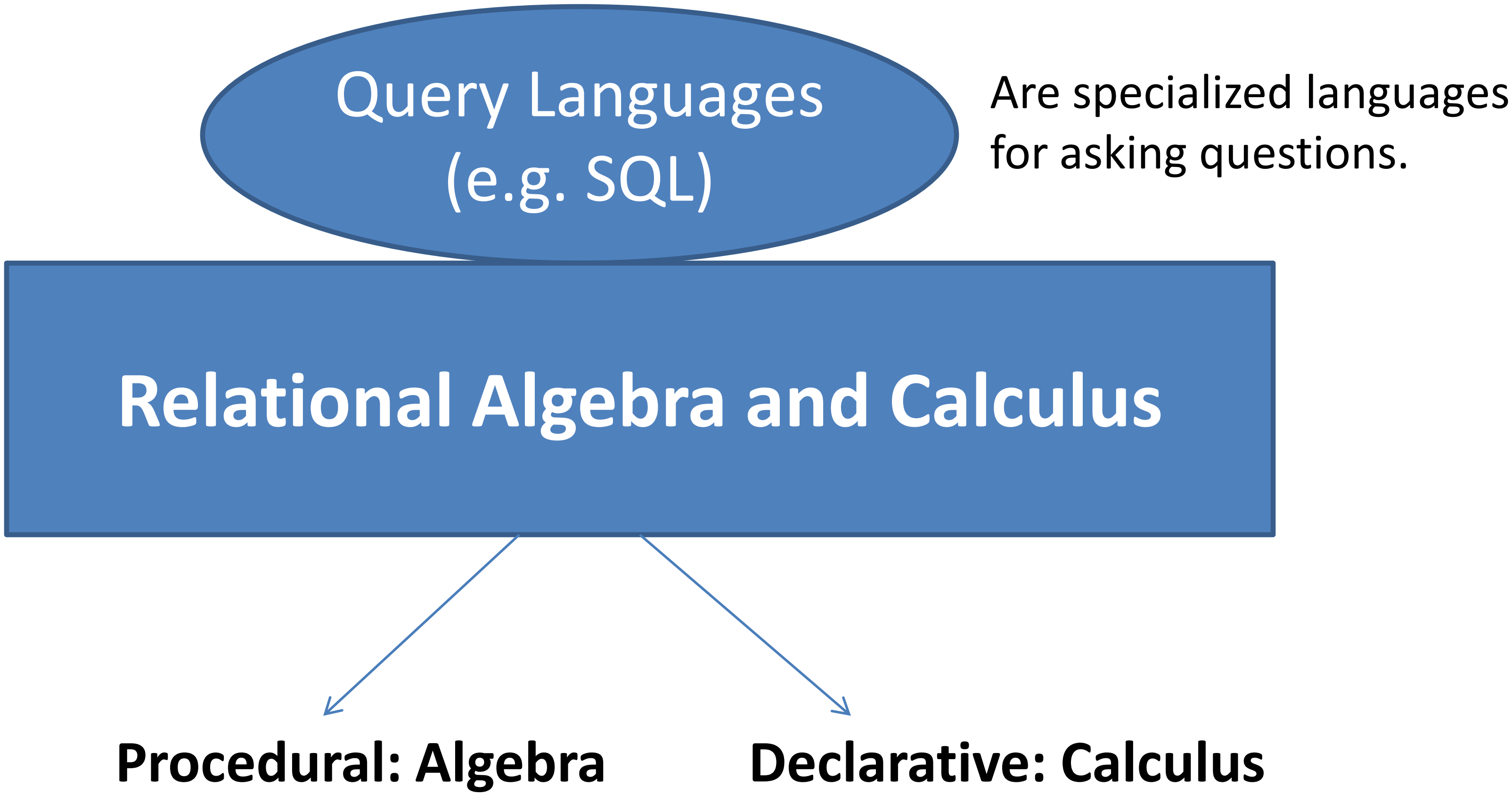
NYU New York University

CSCI-GA.2433-001

Database Systems

Lecture 4: Relational Algebra and Calculus

Mohamed Zahran (aka Z)
mzahran@cs.nyu.edu
http://www.mzahran.com


Query Languages
(e.g. SQL)
Are specialized languages for asking questions.
Relational Algebra and Calculus
Procedural: Algebra
Declarative: Calculus


Instances
of
Relations
Query
Instances
of
A Relation

Relational Algebra

- Queries are composed using a collection of operators.
- Every operator:
  - Accepts one or two relation instances
  - Returns a relation instance.
- Compose relational algebra expression
- Each query describes a step-by-step procedure for computing the desired answer.

Relational Algebra

- Five basic operators
  - Selection
  - Projection
  - Union
  - Cross-product
  - Difference

Selection

$$\sigma_{Selection\_Criteria}(Input)$$

A relation instance

Manipulates data in a single relation

The selection operator specifies the tuples to retain through selection criteria.

A boolean combination (i.e. using V and Λ) of terms

Attribute op constant or attribute1 op attribute2

< , <=, =, ≠, >=, or >

Selection

S2

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

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

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

Projection

$$\pi_{fields}(Input)$$

Allows us to extract columns from a relation

Example:

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

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

age
35.0
55.5

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

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

sname	rating
yuppy	9
rusty	10

Set Operations

- Takes as input two relation instances
- Four standard operations
  - Union
  - Intersection
  - Set-difference
  - Cross-product
- Union, intersection, and difference require the two input set to be union compatible
  - They have the same number of fields
  - corresponding fields, taken in order from left to right, have the same domains

Set Operation: Union

- **R U S** returns relation instance containing all tuples that occur in either relation instance R or S, or both.
- R and S must be union compatible.
- Schema of the result is defined to be that of R.

Set Operation: Union

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

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 U S2

Set Operation: Intersection

- $R \cap S$: returns a relation instance containing all tuples that occur in both R and S.
- R and S must be union compatible.
- Schema of the result is that of R.

Set Operation: Intersection

S1

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

S2

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

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

S1 ∩ S2

Set Operation: Set-Difference

- **R – S**: returns a relation instance containing all tuples that occur in R but not in S.
- R and S must be union-compatible.
- Scheme of the result is the schema of R.

Set Operation: Set-Difference

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

S1 – S2

sid	sname	rating	age
22	dustin	7	45.0

Set Operation: Cross-Product

- R x S: Returns a relation instance whose scheme contains:
  - All the fields of R (in the same order as they appear in R)
  - All the fields os S (in the same order as they appear in S)
- The result contains one tuple <r,s> for each pair with $r \in R$ and $s \in S$
- Basically, it is the Cartesian product.
- Fields of the same name are unnamed.

Set Operation: Cross-Product

S1

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

R1

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

S1 x R1

(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

Renaming

- Name conflict can arise in some situations
- It is convenient to be able to give names to the fields of a relation instance defined by a relational algebra expression.

$$\rho(R(\overline{F}), E)$$

- Take arbitrary relational expression E
- Returns an instance of a new relation R
- R is the result of E except that some fields are renamed
- Renaming list has the form (oldname → newname or position → newname)

Renaming

$$\rho(C(1 \rightarrow sid1, 5 \rightarrow sid2), S1 \times R1)$$

(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

sid1	sname	rating	age	sid2	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

Question: Can you define R ∩ S using other operators?

Other Operators?

- We can define any operation using the operators that we have seen.
- Some other operations appear very frequently.
- So they deserve to have their own operators.
  - Join
  - Division

Join

- Can be defined as cross-product followed by selection and projection.
- We have several variants of join.
  - Condition joins
  - Equijoin
  - Natural join

Condition Join

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

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

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	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

(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

Equijoin

$$R \bowtie_{C} S$$

- Condition consists only of equalities connected by ∧
- Redundancy in retaining both attributes in result
- So, an additional projection is applied to remove the second attribute.

Equijoin

Example:

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

(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

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

Natural Join

- It is an equijoin in which equalities are specified on all fields having the same name in R and S
- We can then omit the join condition.
- Result is guaranteed not to have two fields with the same name.
- If no fields in common, then natural join is simply cross product.

Division

- Suppose A has two groups of fields <x,y>
- y fields are same fields in terms of domain as B
- A/B = <x> such as for every y value in a tuple of B there is <x,y> in A.

Division

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

A

pno
p2

B1

pno
p2
p4

B2

pno
p1
p2
p4

B3

sno
s1
s2
s3
s4

A/B1

sno
s1
s4

A/B2

sno
s1

A/B3

Question: Can we define A/B using the other basic operators?

Disqualified *x* values: $\pi_x((\pi_x(A) \times B) - A)$

A/B: $\pi_x(A)$ − all disqualified tuples

Examples

Sailors

sid	sname	ratin	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Reserves

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Boats

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

Q1. Find the names of sailors who have reserved boat 103

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

Solution 2: $\pi_{sname}(\sigma_{bid=103}(\text{Reserves} \bowtie Sailors))$

Examples

Sailors

sid	sname	ratin	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Reserves

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Boats

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

Q2: Find the names of sailors who have reserved a red boat.

Sol1: $\pi_{sname}((\sigma_{color='red'} Boats) \bowtie Reserves \bowtie Sailors)$

Sol2: $\pi_{sname}(\pi_{sid}((\pi_{bid}\sigma_{color='red'} Boats) \bowtie Res) \bowtie Sailors)$

Examples

Sailors

sid	sname	ratin	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Reserves

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Boats

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

Q3: Find the colors of boats reserved by Lubber.

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

Examples

Sailors

sid	sname	ratin	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Reserves

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Boats

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

Q5. Fine the names of sailors who reserved a red or a green boat.

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

$$\pi_{sname}(Tempboats \bowtie Reserves \bowtie Sailors)$$

Relational Calculus

- An alternative to relational algebra.
- Declarative
  - describe the set of answers
  - without being explicit about how they should be computed
- One variant is called: tuple relational calculus (TRC).
- Another variant: domain relational calculus (DRC)
- Calculus has variables, constants, comparison ops, logical connectives and quantifiers.

Tuple Relational Calculus

- A TRC query has the form {T | p(T)}
  - T is a tuple variable
  - p(T) is a formula that describes T
- Result: set of all tuples t to which p(T) evaluates to true when T = t
- Example: $\{S \mid S \in Sailors \wedge S.rating > 7\}$

Tuple Relational Calculus

Q: Find the names and ages of sailors with a rating above 7

$$\{P \mid \exists S \in Sailors(S.rating > 7 \land Pname = S.sname \land Page = S.age)\}$$

Q: Find the sailor name, boat id, and reservation date for each reservation.

$$\{P \mid \exists R \in Reserves \ \exists S \in Sailors$$
$$(R.sid = 8.sid \land P.bid = R.bid \land P.day = R.day \land P.sname = S.sname)\}$$

Domain Relational Calculus

- *Query* has the form:

$$\left\{ \left\langle x1, x2, \ldots, xn \right\rangle \mid p\left(\left\langle x1, x2, \ldots, xn \right\rangle \right) \right\}$$

- *Answer* includes all tuples $\left\langle x1, x2, \ldots, xn \right\rangle$ that make the *formula* $p\left(\left\langle x1, x2, \ldots, xn \right\rangle \right)$ be *true*.

Example: Find all sailors with a rating above 7

$$\{ \langle I, N, T, A \rangle \mid \langle I, N, T, A \rangle \in Sailors \wedge T > 7 \}$$

Giving each attribute a variable name

Ensures that I, N, T, and A are restricted to be fields of the same tuple

Algebra Vs Calculus

- Every query that can be expressed in relational algebra can also be expressed in relational calculus.
- The other way around is a bit tricky. Think, for example, about:

$$\{S \mid \neg(S \in Sailors)\}$$

Conclusions

- Relational algebra and calculus are the foundation of query languages like SQL.
- Queries are expressed by languages like SQL, and the DBMS translates the query into relational algebra.
  - DBMS tries to look for the cheapest relational expression.
- Section 4.2.6 is very useful, pay close attention to it.
- For the calculus part, we will use slides only.