
Lecture 3: Relational Algebra and Calculus

Dragan Mirkovic
Department of Computer Science
University of Houston

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Announcements

- Quiz #1 Today
 - Ch. 3.1 – 3.4 and 3.6
 - The scores will be posted on the web (if you have any objections to that please let me know)
- I have Oracle CDs available
- Today:
 - Relational Algebra (Ch. 4 in text)

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

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 are not programming languages!
 - QLs not expected to be “Turing complete”.
 - QLs not intended to be used for complex calculations.
 - QLs should support easy, efficient access to large data sets.

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

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-operational, declarative.)
- Understanding Algebra & Calculus is key to understanding SQL, query processing!

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

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 regardless of instance!)
 - The schema for the result of a given query is also fixed! Determined by definition of query language constructs.
- Positional vs. named-field notation:
 - Positional notation easier for formal definitions,
 - named-field notation more readable, but can be tedious if many intermediate instances require naming
 - Both used in SQL

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Example Instances

- “Sailors” and “Reserves” relations for our examples.
- We’ll use positional or named field notation,
- We assume that names of fields in query results are ‘inherited’ from names of fields in query input relations.
- Positional reference using the order defined by the first listing

R1

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

S1

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

S2

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

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Relational Algebra

- Basic operations:
 - Selection (σ) Selects a subset of rows from relation.
 - Projection (π) Deletes unwanted columns from relation.
 - Cross-product (\times) Allows us to combine two relations.
 - Set-difference ($-$) Tuples in reln. 1, but not in reln. 2.
 - Union (\cup) Tuples in reln. 1 and 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”.)

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

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 (only) input relation.
- Projection operator has to eliminate duplicates! (Why??)
 - 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

$\pi_{sname, rating}(S_2)$

age
35.0
55.5

$\pi_{age}(S_2)$

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Selection

- Selects rows that satisfy selection condition.
- No duplicates in result! (Why?)
- 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

$$\sigma_{rating > 8}(S_2)$$

sname	rating
yuppy	9
rusty	10

$$\pi_{sname, rating}(\sigma_{rating > 8}(S_2))$$

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Union, Intersection, 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 (domain).
- They inherit names from the 1st relation
- What is the schema of result?

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

$$S_1 \cup S_2$$

sid	sname	rating	age
22	dustin	7	45.0

$$S_1 - S_2$$

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

$$S_1 \cap S_2$$

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

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. (listed in parentheses – only corresponding domain is inherited)

(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

$S1 \times R1$

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Renaming Operator

- The results of a relational algebra expression inherits field names from input instances
 - Possible conflicts (e.g., $S1 \times R1$)
 - Solution: renaming
$$\rho (C(1 \rightarrow sid1, 5 \rightarrow sid2), S1 \times R1)$$
 - Resulting schema:
 $C(sid1: integer, sname: string, rating: integer, age: real, sid2: integer, bid: integer, day: dates)$

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Joins

- One of the most useful operations in relational algebra
- It can be defined as a cross-product followed by selection
- Frequent use warrants separate command (shortcut)
- Several variants:
 - Conditional join
 - Equijoin
 - Natural join

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Condition Join

- Definition: $R \bowtie_c S = \sigma_c(R \times S)$

(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

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

- 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.

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Equi-Join, Natural join

- A special case of condition join where the condition c contains only equalities between two fields

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

$$S1 \bowtie_{sid} R1$$

- Result schema similar to cross-product, but only one copy of fields for which equality is specified.
- Natural Join: Equijoin on all common fields.
 - The join condition is omitted

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Division

- Not supported as a primitive operator, but useful for expressing queries like:
 - Find sailors who have reserved all boats.
- Let A have 2 fields, x and y ; B have only field y :
 - $A/B = \{ \langle x \rangle \mid \exists \langle x, y \rangle \in A \ \forall \langle y \rangle \in B \}$
 - 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 $\langle x, y \rangle$ is the list of fields of A .

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Examples of Division A/B

A	sno	pno	pno	pno	pno
	s1	p1	p2	p2	p1
	s1	p2	<i>B1</i>	p4	p2
	s1	p3	<i>B2</i>	<i>B2</i>	p4
	s1	p4	sno	<i>B3</i>	<i>B3</i>
	s2	p1	s1	sno	sno
	s2	p2	s2	s1	s1
	s3	p2	s3	s4	<i>A/B3</i>
	s4	p2	s4	<i>A/B2</i>	<i>A/B2</i>
	s4	p4	<i>A/B1</i>	<i>A/B1</i>	<i>A/B1</i>

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Expressing A/B Using Basic Operators

- Division is not essential op; just a useful shorthand.
 - (Also true of joins, but joins are so common that systems implement joins specially.)
- 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.
- **Disqualified x values:** $\pi_x((\pi_x(A) \times B) - A)$
- **A/B:** $\pi_x(A) -$ all disqualified tuples

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Example

- Find names of sailors who've reserved boat #103

- Solution 1:

$$\pi_{sname}((\sigma_{bid=103} Reserves) \bowtie Sailors)$$

- Solution 2:

$$\rho(Temp1, \sigma_{bid=103} Reserves)$$

$$\rho(Temp2, Temp1 \bowtie Sailors)$$

$$\pi_{sname}(Temp2)$$

- Solution 3:

$$\pi_{sname}(\sigma_{bid=103}(Reserves \bowtie Sailors))$$

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Example

- Find names of sailors who've reserved a red boat

- Information about boat color only available in Boats; so we need an extra join:

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

- A more efficient solution

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

- A query optimizer can find this given the first solution!

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Example

- Find sailors who've reserved a red or a green boat
- We can identify all red or green boats, then find sailors who've reserved one of these boats:

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

- Can identify all red or green boats, then find sailors who've reserved one of these boats:
- Can also define Tempboats using union! (How?)
- What happens if \vee is replaced by \wedge in this query?

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Example

- Find sailors who've reserved a red and a green boat
- Previous approach won't work! Must identify sailors who've reserved red boats, sailors who've reserved green boats, then find the intersection (note that sid is a key for Sailors):

$$\rho (Tempred, \pi_{sid}((\sigma_{color='red'} Boats) \bowtie Reserves))$$
$$\rho (Tempgreen, \pi_{sid}((\sigma_{color='green'} Boats) \bowtie Reserves))$$
$$\pi_{sname}((Tempred \cap Tempgreen) \bowtie Sailors)$$

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Example

- Find the names of sailors who've reserved all boats
- Uses division; schemas of the input relations to / must be carefully chosen:

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

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

- Find to find sailors who've reserved all 'Interlake' boats:

$$\dots / \pi_{bid} (\sigma_{bname='Interlake'} Boats)$$

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Relational Calculus

- We will give only a brief introduction here
 - If you want to know more read Ch. 4.3
- It is an alternative to relational algebra
 - Nonprocedural or declarative
 - We can describe answers without explicitly stating how they should be computed
 - Calculus has *variables, constants, comparison ops, logical connectives* and *quantifiers*.
 - Two variants
 - Tuple relational calculus (TRC): Variables range over (i.e., get bound to) *tuples*.
 - Domain relational calculus (DRC): Variables range over *domain elements* (= field values).
 - Both TRC and DRC are simple subsets of first-order logic.

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Tuple Relational Calculus

- **Tuple variable** = a variable that takes tuples of some schema as values
- TRC query:

$$\{T \mid p(T)\}$$

- Where:
 - T is a tuple variable
 - $P(T)$ is a formula that describes T
- The result
 - the set of all tuples for which $p(T)$ evaluates to **true**

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Example

- Find all sailors with a rating above 7
 $\{S \mid S \in \text{Sailors} \wedge S.\text{rating} > 7\}$
- Using the following instance of sailors

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

S1

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

Result

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Syntax of TRC Queries

- Rel = relation name
- R, S = tuple variables
- a = attribute of R , b attribute of S
- op = operator from $\{<, >, =, \leq, \geq, \neq\}$
- **Atomic formula:** $R \in Rel, R.a \text{ op } S.b$
 $R.a \text{ op } constant, \text{ or } constant \text{ op } R.a$
- **Formula:**
Any atomic formula
 $\neg p, p \wedge q, p \vee q, p \Rightarrow q$
 $\exists R (p(R))$
 $\forall R (p(R))$

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Examples

- Find the names and ages of sailors with a rating above 7
 $\{P \mid \exists S \in Sailors (S.rating > 7 \wedge P.name = S.name \wedge P.age = S.age)\}$
- Find the names of sailors who have reserved a red boat.
 $\left\{ P \mid \exists S \in Sailors \exists R \in Reserves (R.sid = S.sid \wedge P.name = S.name) \right.$
 $\left. \wedge \exists B \in Boats (B.bid = R.bid \wedge B.color = 'red') \right\}$
- Find the names of sailors who have reserved all boats
 $\left\{ P \mid \exists S \in Sailors \forall B \in Boats \right.$
 $\left. (\exists R \in Reserves (R.sid = S.sid \wedge P.name = S.name \wedge B.bid = R.bid)) \right\}$

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

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.
- Algebra and safe calculus have same expressive power, leading to the notion of relational completeness.

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005

Homework

- Read Ch 4.1-4.3
- Page 127, 4.3(1-4), 4.4 (1,2)

D. Mirkovic, COSC 3480: Design of File and Database Systems, Spring 2005