Relational Algebra
关系代数

courtesy of Joe Hellerstein for some slides

Guifeng Zheng
SUN YAT-SEN UNIVERSITY

Recap: You are here

- First part of course is done: conceptual foundations
- You now know:
  - E/R Model
  - Relational Model
  - Relational Algebra (a little, project / join)
- You now know how to:
  - Capture part of world as an E/R model
  - Convert E/R models to relational models
  - Convert relational models to good (normal) forms
- Next:
  - Create, update, query tables with R.A/SQL
  - Write SQL/DB-connected applications

3-minute Normalization Review

Q: What’s required for BCNF?

Q: How do we fix a non-BCNF relation?

Q: If As→Bs violates BCNF, what do we do?

Q: Can BCNF decomposition ever be lossy?

Q: How do we combine two relations?

Q: Can BCNF decomp. lose FDs?

Q: Why would you ever use 3NF?

Relational Query Languages

- *Query languages:* manipulation and retrieval of data
- Relational model supports simple, powerful QLs:
  - Strong formal foundation based on logic.
  - Allows for much optimization.
- Query Languages **!=** programming languages!
  - QLs not expected to be "Turing complete".
  - QLs not intended to be used for complex calculations.
  - QLs support easy, efficient access to large data sets.

(Actually, I no longer believe this. But it's the standard viewpoint)

Formal Relational Query Languages

- *Relational Algebra关系代数*:
  - More operational, very useful for representing execution plans.
- *Relational Calculus关系演算*:
  - Describe what you want, rather than how to compute it. (Non-procedural, *declarative*.)

☛ *Understanding Algebra & Calculus is key to understanding SQL, query processing!*

What is relational algebra?

- An algebra for relations

- “High-school” algebra: an algebra for *numbers*
- *Algebra* = formalism for constructing expressions
  - Operations
  - Operands: Variables, Constants, expressions
- Expressions:
  - Vars & constants
  - Operators applied to expressions
  - They evaluate to values

Algebra	Vars/consts	Operators	Eval to
High-school	Numbers	+ * - / etc.	Numbers
Relational	Relations (=sets of tupes)	union, intersection, join, etc.	Relations

Why do we care about relational algebra?

- The exprs are *the form that questions about the data take* （有关数据的问题采用的形式！）
  - The relations these exprs cash out to are *the answers to our questions*（其表示的关系正是我们的问题的答案）
- RA ~ more succinct rep.(简洁表示) of many SQL queries
- DBMS parse SQL into something like RA.

- First proofs of concept for RDBMS/RA:
  - System R at IBM
  - Ingress at Berkeley
- "Modern" implementation of RA: SQL
  - Both state of the art, mid-70s

Preliminaries预备知识

- A query is applied to *relation instances*
- The result of a query is also a relation instance.
  - *Schemas* of input relations for a query are fixed
  - Schema for the *result* of a query is also fixed.
    - determined by the query language constructs
- Positional vs. named-field notation:
  - Positional notation easier for formal definitions
  - Named-field notation more readable.
  - Both used in SQL
    - Though positional notation is discouraged

Relational Algebra: 5 Basic Operations

- ***Selection*** (σ)
  - **Selects a subset of *rows* (horizontal)**
- ***Projection*** (π)
  - **Retains only desired *columns* (vertical)**
- ***Cross-product*** ($\times$)
  - **Allows us to combine two relations.**
- ***Set-difference*** (—)
  - **Tuples in r1, but not in r2.**
- ***Union*** ($\cup$)
  - **Tuples in r1 or in r2.**
- **Since each operation returns a relation, operations can be *composed!* (Algebra is "closed".)**

Example Instances

R1

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

S1

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

Boats

<u>bid</u>	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

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

Projection (π)

- **Example:** $\pi_{sname,rating}(S2)$
- **Retains only attributes that are in the "*projection list*".**
- ***Schema* of result:**
  - the fields in the projection list
  - with the same names that they had in the input relation.
- **Projection operator has to *eliminate duplicates***
  - Note: real systems typically don't do duplicate elimination
  - Unless the user explicitly asks for it.
  - (Why not?)

Projection (π)

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

S2

sname	rating
yuppy	9
lubber	8
guppy	5
rusty	10

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

age
35.0
55.5

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

Selection (σ)

- **Selects rows that satisfy *selection condition*.**
- **Result is a relation.**

  Schema of result is same as that of the input relation.
- **Do we need to do duplicate elimination?**

<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

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

sname	rating
yuppy	9
rusty	10

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

Union, Set-Difference

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

- **For which, if any, is duplicate elimination required?**

Union

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

S1

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

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

$S1 \cup S2$

Set Difference

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

S1

sid	sname	rating	age
22	dustin	7	45.0

$S1 - S2$

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

S2

sid	sname	rating	age
28	yuppy	9	35.0
44	guppy	5	35.0

$S2 - S1$

Cross-Product

- **S1 $\times$ R1:**
  - **Each row of S1 paired with each row of R1.**
- **Q: How many rows in the result?**

- ***Result schema*** **has one field per field of S1 and R1,**
  - Field names `inherited' if possible.

Cross Product Example

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

Naming conflict: S1 and R1 have a field with the same name.

(Can use the *renaming operator*)

Rename op

- Changes the *schema*, not the instance
- Notation: $\rho_{B1,\ldots,Bn}(R)$
- ρ is spelled “rho”, pronounced “row”

$$\rho_{(C(1\text{->}sid1,\ 5\text{->}sid2)}(R1XS1)$$

- Example:
  - Employee(ssn,name)
  - $\rho_{E2(social,\ name)}(Employee)$
  - Or just: $\rho_E(Employee)$

Compound Operator: Intersection

- **On top of 5 basic operators, several additional "Compound Operators"**
  - These add no computational power to the language
  - Useful shorthand
  - Can be expressed solely with the basic operators.
- **Intersection takes two input relations, which must be *union-compatible*.**
- **Q: How to express it using basic operators?**

$$R \cap S = R - (R - S)$$

Intersection

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

S1

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

S2

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

$$S1 \cap S2$$

Compound Operator: Join

- **Involve cross product, selection, and (sometimes) projection.**
- **Most common type of join: "*natural join*"**
  - R $\bowtie$ S conceptually is:
    - Compute R × S
    - Select rows where attributes appearing in both relations have equal values
    - Project all unique attributes and one copy of each of the common ones.
- **Note: Usually done much more efficiently than this.**

Natural Join Example

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

R1

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

S1

S1 $\bowtie$ R1 =

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

- R

A	B
X	Y
X	Z
Y	Z
Z	V

S

B	C
Z	U
V	W
Z	V

- $R \bowtie S = ?$

- Unpaired tuples called *dangling*

Natural Join

- Given the schemas R(A, B, C, D), S(A, C, E), what is the schema of $R \bowtie S$?

- Given R(A, B, C), S(D, E), what is $R \bowtie S$?

- Given R(A, B), S(A, B), what is $R \bowtie S$?

Other Types of Joins

- *Condition Join (or "theta-join")*:

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

(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.**
- **May have fewer tuples than cross-product.**
- *Equi-Join*: Special case: condition *c* contains only conjunction of ***equalities*****.**

Division Operation

- Notation: $r \div s$
- Suited to queries that include the phrase "for all."
- Let r and s be relations over schemas R and S respectively, where

$$R = (A_1, \ldots, A_m, B_1, \ldots, B_n)$$
$$S = (B_1, \ldots, B_n)$$

The result of $r \div s$ is a relation over the schema $(R - S) = (A_1, \ldots, A_m)$

$$r \div s = \{ t \mid (t \in \pi_{R-S}(r)) \wedge (\forall u \in s, tu \in r) \}$$

Division Operation - example

$$r \div s \;=\; \{\, t \mid (t \in \pi_{R-S}(r)) \wedge (\forall u \in s,\ tu \in r) \,\}$$

r

A	B
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
δ	6
ε	1
ε	2

s

B
1
2

The result consists of attribute A only but not all of the 5 values. How to find out?
u = 1, 2   Check if: $\forall u \in s\ (tu \in r)$

$t \in \pi_{R-S}(r)$

A
α
β
γ
δ
ε

Is ⟨α,1⟩ and ⟨α,2⟩ tuples in r?

Is ⟨β,1⟩ and ⟨β,2⟩ tuples in r?

check γ and δ…

Is ⟨ε,1⟩ and ⟨ε,2⟩ tuples in r?

r ÷ s

A
α
ε

Another Division Example

Relations r, s:

A	B	C	D	E
α	A	α	A	1
α	A	γ	A	1
α	A	γ	B	1
β	A	γ	A	1
β	A	γ	B	3
γ	A	γ	A	1
γ	A	γ	B	1
γ	A	β	B	1

D	E
A	1
B	1

r ÷ s

A	B	C
α	A	γ
γ	A	γ

Properties of Division Operation

- Let $q = r \div s$
  Then q is the largest relation satisfying: $q \times s \subseteq r$

Relation r, s:

r

A	B
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
δ	6
ε	1
ε	2

s

B
1
2

q

A
α
ε

Examples

Reserves

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

Sailors

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

Boats

bid	bname	color
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green
104	Marine	Red

Find names of sailors who've reserved boat #103

- Solution 1:

$$\pi_{name}((\sigma_{bid=103}\,\text{Reserves}) \bowtie Sailors)$$

- Solution 2:

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

Find names of sailors who've reserved a red boat

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

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

❖ A more efficient solution:

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

☛ *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:

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

$$\pi_{sname}(Tempboats \bowtie \text{Re}serves \bowtie Sailors)$$

Find sailors who've reserved a red <u>and</u> a green boat

- Cut-and-paste previous slide?

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

$$\pi_{sname}(Tempboats \bowtie \mathrm{Re}serves \bowtie Sailors)$$

Find sailors who've reserved a red <u>and</u> 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):

$$\rho\ (Tempred, \pi_{sid}((\sigma_{color='red'} Boats) \bowtie \mathrm{Re}serves))$$

$$\rho\ (Tempgreen, \pi_{sid}((\sigma_{color='green'} Boats) \bowtie \mathrm{Re}serves))$$

$$\pi_{sname}((Tempred \cap Tempgreen) \bowtie Sailors)$$

Summary

- Relational Algebra: a small set of operators mapping relations to relations
  - Operational, in the sense that you specify the explicit order of operations
  - A *closed* set of operators! Can mix and match.
- Basic ops include: σ, π, $\times$, $\cup$, —,
- Important compound ops: $\cap$, $\bowtie$