Note: Slides are posted on the class website, protected by a password written on the board. If you forget it, email me at eoneil@cs.umb.edu for it.

Reading: see class home page www.cs.umb.edu/cs630.

Relational Algebra

CS430/630
Lecture 2

Slides based on "Database Management Systems" 3rd ed, Ramakrishnan and Gehrke

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 aggressive optimization
- Query Languages != programming languages
  - QLs not intended to be used for complex step-by-step calculations
  - QLs support easy, efficient access to large data sets
  - New QL: GraphQL, released by Facebook in 2015, not table-oriented in its syntax, but can use data from an RDBMS.

Formal Relational Query Languages

- Two languages form the basis for SQL:
  - *Relational Algebra*:
    - operational
    - useful for representing execution plans
    - very relevant as it is used by query optimizers!
    - gets us thinking in tables, understanding joins
  - *Relational Calculus*:
    - Lets users describe the result, NOT how to compute it - declarative
    - We will focus on relational algebra

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
  - The schema for the *result* of a given query is determined by operand schemas and operator type
- These relations have no duplicate tuples, i.e., a relation is an (unordered) set of tuples/rows
- Each operation returns a relation
  - operations can be *composed* !
  - Well-formed expression: a relation, or the results of a relational algebra operation on one or two relations

Relational Algebra

- Basic operations:
  - Selection σ Selects a subset of rows from relation
  - Projection π Deletes unwanted columns from relation
  - Cross-product X Allows us to combine several relations
  - Join $\bowtie$ Combines several relations using conditions
  - Division $\div$ A bit more complex, will cover later on
  - Set Operations: Union $\cup$ Intersection $\cap$ Difference -
  - Renaming ρ Helper operator, does not derive new result, just renames relations and fields

$$\rho(R, E)$$

    - here R becomes another name for E

Example Schema, with table contents

Sailors

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

Boats

bid	name	color
101	interlake	red
103	clipper	green

Reserves

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

Schema in abbreviated format

Sailors

<u>sid</u>	sname	rating	age

Boats

<u>bid</u>	name	color

Reserves

<u>sid</u>	<u>bid</u>	<u>day</u>

- No table contents (not part of *schema* anyway)
- No domains shown for columns (string, integer, etc.)
- Just table names, column names, keys of schema
- Compact, and enough for us to understand the database

Example Schema: Reserves Relation

Reserves

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

- Multiple entity ids in a key signals a relationship between those entities, here Sailor and Boat
  - Example: (22, 101, 10/10/96): Sailor 22 reserved boat 101 on 10/10/1996 (ancient example!)
- Note that day is part of the key here too
  - This means (sid, bid) is not a key
  - So multiple rows can have same (sid, bid).
  - Example: (22, 101, 10/10/2016)
  - Sailor 22 can reserve the same boat 101 on different days and the database can hold all of these reservations.

Relation Instances Over Time

Sailors

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

Reserves

R1

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

Projection

- Unary operator (i.e., has only one argument)
- Deletes (projects out) attributes that are not in *projection list*

$$\pi_{attr1, attr2, \ldots} relation$$

- *Result Schema* contains the attributes in the projection list
  - With the same names that they had in the input relation
- Projection operator has to eliminate *duplicates*!
  - Real systems typically do not do so by default
  - Duplicate elimination is expensive! (sorting)
  - In SQL, user must explicitly asks for duplicate eliminations (DISTINCT), but here in RA, it happens automatically

Projection Examples

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

sname	rating
yuppy	9
lubber	8
guppy	5
rusty	10

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

age
35.0
55.5

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

Selection

- Unary Operator
- Selects rows that satisfy *selection condition*

$$\sigma_{condition} relation$$

- Condition contains constants and attributes from relation
  - Evaluated for each **individual** tuple
  - May use logical connectors AND (^), OR (ˇ), NOT (¬)
- No duplicates in result! Why?
- *Result Schema* is identical to schema of the input relation

Selection Example

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
28	yuppy	9	35.0
58	rusty	10	35.0

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

sname	rating
yuppy	9
rusty	10

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

Cross-Product

- Binary Operator

$$R \times S$$

- Each row of relation *R* is paired with each row of *S*
- *Result Schema* has one field per field of R and S
  - Field names `inherited' when 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

C=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

Conflict: Both *R* and *S* have a field called *sid*

Cross-Product + Renaming Example

C

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

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

Condition Join (Theta-join)

$$R \bowtie_{\theta} S = \sigma_{\theta}(R \times S)$$

- *Result Schema* same as that of cross-product

Condition Join (Theta-join) Example

S1* X *R1

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~~

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

sid1	sname	rating	age	sid2	bid	day
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	58	103	11/12/96

Equi-Join

- A special case of condition join where the condition contains only ***equalities***

$$R \bowtie_{R.attr1=S.attr2} S$$

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

Equi-Join Example

S1 X R1

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

$S1 \bowtie_{S1.sid = R1.sid} R1$ or simply $S1 \bowtie R1$

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

- Equijoin on *all* common fields

$$R \bowtie S$$

- Common fields are NOT duplicated in the result

$$S1 \bowtie R1$$

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

Note how it extends each R row to add sailor details

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 domain (type): integer, real, string, date
  - (We will see that SQL has "type compatibility", so char(10) and char(20) can be union'd, for example, to char(20), and float vs. integer, to float, but relational algebra has this simpler rule)

- What is the *schema* of result?

Union Example: common case of same field names

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

$S1 \cup S2$

Union Example: case of different field names

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>	name	color
101	interlake	red
103	clipper	green

sname
dustin
lubber
rusty

$\pi_{sname}(S1)$

name
interlake
clipper

$\pi_{name}(Boats)$

←Union-compatible schema

$$\pi_{sname}(S1) \cup \pi_{name}(Boats) = ?$$

Union Example: case of different field names

sname
dustin
lubber
rusty

$\pi_{sname}(S1)$

name
interlake
clipper

$\pi_{name}(Boats)$

←Union-compatible schema

sname
dustin
lubber
rusty
interlake
clipper

By pg. 104, take field name from left operand.

$$\pi_{sname}(S1) \cup \pi_{name}(Boats)$$

Intersection Example

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
31	lubber	8	55.5
58	rusty	10	35.0

$S1 \cap S2$

Set-Difference Example

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

$S1 - S2$

Example Schema

Sailors

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

Boats

bid	name	color
101	interlake	red
103	clipper	green

Reserves

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

Sample Query 1

Sailors

<u>sid</u>	sname	rating	age

Boats

<u>bid</u>	name	color

Reserves

<u>sid</u>	<u>bid</u>	<u>day</u>

- Find <u>names of sailors</u> who've <u>reserved</u> <u>boat #103</u>

bid

Detail of sailor sid

sid, bid in reserves table

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

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

Example Schema

Sailors

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

Boats

bid	name	color
101	interlake	red
103	clipper	green

Reserves

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

Sample Query 2

Sailors

<u>sid</u>	sname	rating	age

Boats

<u>bid</u>	name	color

Reserves

<u>sid</u>	<u>bid</u>	<u>day</u>

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

Detail of sailor sid   sid, bid …   Detail of boat bid

$$\pi_{sname}(\pi_{sid}((\pi_{bid}(\sigma_{color='red'}B)) \bowtie R) \bowtie S)$$

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