Relational Model and Algebra

Introduction to Databases

CompSci 316 Spring 2020

DUKE COMPUTER SCIENCE

1

Announcements (Tue. Jan. 14)

- You should be on Piazza and Gradescope
  - Otherwise, let the instructor know after class
- HW1 will be posted after class, due next Tuesday 11:59 pm
  - Instant feedback, multiple submissions allowed until correct!
  - 5% / hour late submission penalty
  - Use pgweb from course website to try your queries on small Beers dataset
  - If you join the class after Tuesday 01/14, let the instructor know
- Office hours posted on course website
  - There is at least one everyday except Saturday

2

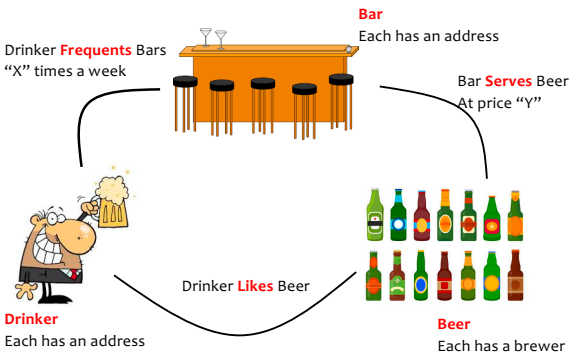
Today's plan

- Revisit relational model
- Revisit simple SQL queries and its semantic
- Start relational algebra

3

The famous "Beers" database

Your database for HW1



4

"Beers" as a Relational Database

See online database for more tuples

Bar

name	address
The Edge	108 Morris Street
Satisfaction	905 W. Main Street

Beer

Name	brewer
Budweiser	Anheuser-Busch Inc.
Corona	Grupo Modelo
Dixie	Dixie Brewing

Drinker

name	address
Amy	100 W. Main Street
Ben	101 W. Main Street
Dan	300 N. Duke Street

Serves

bar	beer	price
The Edge	Budweiser	2.50
The Edge	Corona	3.00
Satisfaction	Budweiser	2.25

Frequents

drinker	bar	times_a_week
Ben	Satisfaction	2
Dan	The Edge	1
Dan	Satisfaction	2

Likes

drinker	beer
Amy	Corona
Dan	Budweiser
Dan	Corona
Ben	Budweiser

What is an example of a

- Relation
- Attribute
- Tuple
- Schema
- Instance

What is

- Set semantic
  - in relational model
- Bag semantic
  - In SQL (why)

5

Basic queries: SFW statement

- SELECT $A_1, A_2, ..., A_n$
  FROM $R_1, R_2, ..., R_m$
  WHERE $condition$

In HW1, you can only use SFW

- SELECT, FROM, WHERE are often referred to as SELECT, FROM, WHERE "clauses"
- Each query must have a SELECT and a FROM
- WHERE is optional

6

Example: reading a table

Serves

bar	beer	price
The Edge	Budweiser	2.50
The Edge	Corona	3.00
Satisfaction	Budweiser	2.25

- SELECT *
  FROM Serves

- Single-table query
- * is a shorthand for "all columns"

7

Example: ORDER BY

Serves

bar	beer	price
The Edge	Budweiser	2.50
The Edge	Corona	3.00
Satisfaction	Budweiser	2.25

- SELECT *
  FROM Serves
  ORDER BY beer

- Equivalent to "ORDER BY beer asc" (asc is default option)
- For descending order, use "desc"
- Can combine multiple orders
- What does this return?
  - ORDER BY beer asc, price desc

8

Example: some columns and DISTINCT

Serves

bar	beer	price
The Edge	Budweiser	2.50
The Edge	Corona	3.00
Satisfaction	Budweiser	2.25

- SELECT beer
  FROM Serves
  Returns a bag

- Only want unique values? Use DISTINCT

- SELECT DISTINCT beer
  FROM Serves
  Returns a set

9

Example: selecting few rows

Serves

bar	beer	price
The Edge	Budweiser	2.50
The Edge	Corona	3.00
Satisfaction	Budweiser	2.25

- SELECT beer AS mybeer
  FROM Serves
  WHERE price < 2.75

- SELECT S.beer
  FROM Serves S
  WHERE bar = 'The Edge'

What does these return?

- SELECT list can contain expressions
  Can also use built-in functions such as SUBSTR, ABS, etc.
- NOT EQUAL TO: Use <>
- LIKE matches a string against a pattern
  % matches any sequence of zero or more characters

10

Example: Join

- Find addresses of all bars that 'Dan' frequents

- SELECT B.address
  FROM Bar B, Frequents F
  WHERE B.name = F.bar
  AND F.drinker = 'Dan'

Bar

name	address
The Edge	108 Morris Street
Satisfaction	905 W. Main Street

drinker	bar	times_a_week
Ben	Satisfaction	2
Dan	The Edge	1
Dan	Satisfaction	2

Frequents

11

Semantics of SFW

- SELECT $E_1, E_2, \ldots, E_n$
  FROM $R_1, R_2, \ldots, R_m$
  WHERE *condition*

- For each t_1 in R_1:
  For each t_2 in R_2: … …
  For each t_m in R_m:

1. Apply "FROM"
Form "cross-product" of R1, .., Rm

If *condition* is true over $t_1, t_2, \ldots, t_m$:

2. Apply "WHERE"
Only consider satisfying rows

Compute and output $E_1, E_2, \ldots, E_n$ as a row

3. Apply "SELECT"
Output the desired columns

12

Step 1: Illustration of Semantics of SFW

- NOTE: This is "NOT HOW" the DBMS outputs the result, but "WHAT" it outputs!

Form a "Cross product" of two relations

- SELECT B.address
  FROM Bar B, Frequents F
  WHERE B.name = F.bar
  AND F.drinker = 'Dan'

Bar

name	address
The Edge	108 Morris Street
Satisfaction	905 W. Main Street

Frequents

drinker	bar	times_a_week
Ben	Satisfaction	2
Dan	The Edge	1
Dan	Satisfaction	2

name	address	drinker	bar	times_a_week
The Edge	108 Morris Street	Ben	Satisfaction	2
The Edge	108 Morris Street	Dan	The Edge	1
The Edge	108 Morris Street	Dan	Satisfaction	2
Satisfaction	905 W. Main Street	Ben	Satisfaction	2
Satisfaction	905 W. Main Street	Dan	The Edge	1
Satisfaction	905 W. Main Street	Dan	Satisfaction	2

13

Step 2: Illustration of Semantics of SFW

- NOTE: This is "NOT HOW" the DBMS outputs the result, but "WHAT" it outputs!

Discard rows that do not satisfy WHERE condition

- SELECT B.address
  FROM Bar B, Frequents F
  WHERE B.name = F.bar
  AND F.drinker = 'Dan'

Bar

name	address
The Edge	108 Morris Street
Satisfaction	905 W. Main Street

Frequents

drinker	bar	times_a_week
Ben	Satisfaction	2
Dan	The Edge	1
Dan	Satisfaction	2

name	address	drinker	bar	times_a_week
~~The Edge~~	~~108 Morris Street~~	~~Ben~~	~~Satisfaction~~	~~2~~
The Edge	108 Morris Street	Dan	The Edge	1
~~The Edge~~	~~108 Morris Street~~	~~Dan~~	~~Satisfaction~~	~~2~~
~~Satisfaction~~	~~905 W. Main Street~~	~~Ben~~	~~Satisfaction~~	~~2~~
~~Satisfaction~~	~~905 W. Main Street~~	~~Dan~~	~~The Edge~~	~~1~~
Satisfaction	905 W. Main Street	Dan	Satisfaction	2

14

Step 3: Illustration of Semantics of SFW

- NOTE: This is "NOT HOW" the DBMS outputs the result, but "WHAT" it outputs!

Output the "address" output of rows that survived

- SELECT B.address
  FROM Bar B, Frequents F
  WHERE B.name = F.bar
  AND F.drinker = 'Dan'

Bar

name	address
The Edge	108 Morris Street
Satisfaction	905 W. Main Street

Frequents

drinker	bar	times_a_week
Ben	Satisfaction	2
Dan	The Edge	1
Dan	Satisfaction	2

name	address	drinker	bar	times_a_week
~~The Edge~~	~~108 Morris Street~~	~~Ben~~	~~Satisfaction~~	~~2~~
The Edge	108 Morris Street	Dan	The Edge	1
~~The Edge~~	~~108 Morris Street~~	~~Dan~~	~~Satisfaction~~	~~2~~
~~Satisfaction~~	~~905 W. Main Street~~	~~Ben~~	~~Satisfaction~~	~~2~~
~~Satisfaction~~	~~905 W. Main Street~~	~~Dan~~	~~The Edge~~	~~1~~
Satisfaction	905 W. Main Street	Dan	Satisfaction	2

15

Final output: Illustration of Semantics of SFW

- NOTE: This is "NOT HOW" the DBMS outputs the result, but "WHAT" it outputs!

Output the "address" output of rows that survived

- SELECT B.address
  FROM Bar B, Frequents F
  WHERE B.name = F.bar
  AND F.drinker = 'Dan'

Bar

name	address
The Edge	108 Morris Street
Satisfaction	905 W. Main Street

Frequents

drinker	bar	times_a_week
Ben	Satisfaction	2
Dan	The Edge	1
Dan	Satisfaction	2

address
108 Morris Street
905 W. Main Street

16

SQL vs. C++, Java, Python…

17

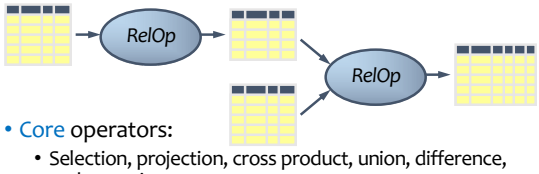
SQL vs. C++, Java, Python…

18

19

Relational algebra

A language for querying relational data based on "operators"



- Core operators:
  - Selection, projection, cross product, union, difference, and renaming
- Additional, derived operators:
  - Join, natural join, intersection, etc.
- Compose operators to make complex queries

19

20

Selection

- Input: a table R
- Notation: $\sigma_p R$
  - p is called a selection condition (or predicate)
- Purpose: filter rows according to some criteria
- Output: same columns as R, but only rows of R that satisfy p (set!)

Example: Find beers with price < 2.75

Serves

bar	beer	price
The Edge	Budweiser	2.50
The Edge	Corona	3.00
Satisfaction	Budweiser	2.25

$\sigma_{price<2.75}$ Serves

bar	beer	price
The Edge	Budweiser	2.50
Satisfaction	Budweiser	2.25

No actual deletion!

Equivalent SQL query?

20

21

More on selection

- Selection condition can include any column of R, constants, comparison ($=, \leq$, etc.) and Boolean connectives ($\wedge$: and, $\vee$: or, $\neg$: not)
  - Example: Serves tuples for "The Edge" or price >= 2.75

$$\sigma_{bar='The\ Edge' \vee price \geq 2.75} Serves$$

- You must be able to evaluate the condition over each single row of the input table!
  - Example: the most expensive beer at any bar

$$\sigma_{price \geq every\ price\ in\ Serves} User$$ **WRONG!**

Serves

bar	beer	price
The Edge	Budweiser	2.50
The Edge	Corona	3.00
Satisfaction	Budweiser	2.25

21

22

Projection

- Input: a table R
- Notation: $\pi_L R$
  - L is a list of columns in R
- Purpose: output chosen columns
- Output: same rows, but only the columns in L $(set!)$

Example: Find all the prices for each beer

Serves

bar	beer	price
The Edge	Budweiser	2.50
The Edge	Corona	3.00
Satisfaction	Budweiser	2.25

$\pi_{beer,price}$ Serves

beer	price
Budweiser	2.50
Corona	3.00
Budweiser	2.25

Output of π_{beer} Serves?

22

23

Cross product

- Input: two tables R and S
- Natation: $R \times S$
- Purpose: pairs rows from two tables
- Output: for each row r in R and each s in S, output a row rs (concatenation of r and s)

Bar

name	address
The Edge	108 Morris Street
Satisfaction	905 W. Main Street

Frequents

drinker	bar	times_a_week
Ben	Satisfaction	2
Dan	The Edge	1
Dan	Satisfaction	2

Bar x Frequents

name	address	drinker	bar	times_a_week
The Edge	108 Morris Street	Ben	Satisfaction	2
The Edge	108 Morris Street	Dan	The Edge	1
The Edge	108 Morris Street	Dan	Satisfaction	2
Satisfaction	905 W. Main Street	Ben	Satisfaction	2
Satisfaction	905 W. Main Street	Dan	The Edge	1
Satisfaction	905 W. Main Street	Dan	Satisfaction	2

Note: ordering of columns does not matter, so R X S = S X R (commutative)

23

24

Derived operator: join

(A.k.a. "theta-join")

One of the most important operations!

- Input: two tables R and S
- Notation: $R \bowtie_p S$
  - p is called a join condition (or predicate)
- Purpose: relate rows from two tables according to some criteria
- Output: for each row r in R and each row s in S, output a row rs if r and s satisfy p
- Shorthand for $\sigma_p(R \times S)$

24

25

Join example

Ambiguous attribute?
Use Bar.name

- Extend Frequents relation with addresses of the bars

$$Frequents \bowtie_{bar=name} Bar$$

Bar

name	address
The Edge	108 Morris Street
Satisfaction	905 W. Main Street

Frequents

drinker	bar	times_a_week
Ben	Satisfaction	2
Dan	The Edge	1
Dan	Satisfaction	2

name	address	drinker	bar	times_a_week
~~The Edge~~	~~108 Morris Street~~	~~Ben~~	~~Satisfaction~~	~~2~~
The Edge	108 Morris Street	Dan	The Edge	1
~~The Edge~~	~~108 Morris Street~~	~~Dan~~	~~Satisfaction~~	~~2~~
Satisfaction	905 W. Main Street	Ben	Satisfaction	2
~~Satisfaction~~	~~905 W. Main Street~~	~~Dan~~	~~The Edge~~	~~1~~
Satisfaction	905 W. Main Street	Dan	Satisfaction	2

25

26

Derived operator: natural join

- Input: two tables R and S
- Notation: $R \bowtie S$
- Purpose: relate rows from two tables, and
  - Enforce equality between identically named columns
  - Eliminate one copy of identically named columns
- Shorthand for $\pi_L(R \bowtie_p S)$, where
  - p equates each pair of columns common to R and S
  - L is the union of column names from R and S (with duplicate columns removed)

26

27

Natural join example

Serves $\bowtie$ *Likes*

$= \pi_?(Serves \bowtie_? Likes)$

$= \pi_{bar,beer,price,drinker}\left(Serves \bowtie_{Serves.beer = Likes.beer} Likes\right)$

Serves

bar	beer	price
The Edge	Budweiser	2.50
The Edge	Corona	3.00
Satisfaction	Budweiser	2.25

Likes

drinker	beer
Amy	Corona
Dan	Budweiser
Dan	Corona
Ben	Budweiser

Serves ⋈ *Likes*

bar	beer	price	drinker
The Edge	Budweiser	2.50	Dan
The Edge	Budweiser	2.50	Ben
The Edge	Corona	3.00	Amy
The Edge	Corona	3.00	Dan
...	....	.....	

Natural Join is on beer

Only one column for beer in the output

What happens if the tables have two or more common columns?

27

28

Union

- Input: two tables R and S
- Notation: $R \cup S$
  - R and S must have identical schema
- Output:
  - Has the same schema as R and S
  - Contains all rows in R and all rows in S (with duplicate rows removed)

Example on board

28

29

Difference

- Input: two tables R and S
- Notation: $R - S$
  - R and S must have identical schema
- Output:
  - Has the same schema as R and S
  - Contains all rows in R that are not in S

Example on board

29

30

Derived operator: intersection

- Input: two tables R and S
- Notation: $R \cap S$
  - R and S must have identical schema
- Output:
  - Has the same schema as R and S
  - Contains all rows that are in both R and S
- How can you write it using other operators?

30

31

Expression tree notation

- Find addresses of all bars that 'Dan' frequents

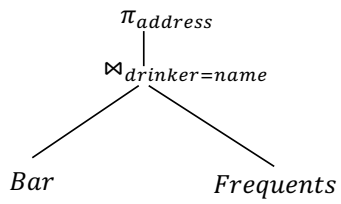
Also called logical Plan tree

Bar

name	address
The Edge	108 Morris Street
Satisfaction	905 W. Main Street

Frequents

drinker	bar	times_a_week
Ben	Satisfaction	2
Dan	The Edge	1
Dan	Satisfaction	2



Equivalent to $\pi_{address}\left(Bar \bowtie_{\substack{drinker=\\ name}} Frequents\right)$

31

32

Using the same relation multiple times

- Find drinkers who frequent both "The Edge" and "Satisfaction"

Frequents

drinker	bar	times_a_week
Ben	Satisfaction	2
Dan	The Edge	1
Dan	Satisfaction	2

32

33

Renaming

- Input: a table R
- Notation: $\rho_S R$, $\rho_{(A_1,A_2,\ldots)}R$, or $\rho_{S(A_1,A_2,\ldots)}R$
- Purpose: "rename" a table and/or its columns
- Output: a table with the same rows as R, but called differently
- Used to
  - Avoid confusion caused by identical column names
  - Create identical column names for natural joins
- As with all other relational operators, it doesn't modify the database
  - Think of the renamed table as a copy of the original

33

34

Summary of core operators

- Selection: $\sigma_p R$
- Projection: $\pi_L R$
- Cross product: $R \times S$
- Union: $R \cup S$
- Difference: $R - S$
- Renaming: $\rho_{S(A_1,A_2,\ldots)}R$
  - Does not really add "processing" power

34

35

Summary of derived operators

- Join: $R \bowtie_p S$
- Natural join: $R \bowtie S$
- Intersection: $R \cap S$

- Many more
  - Semijoin, anti-semijoin, quotient, …

35

36

Exercise

Frequents(drinker, bar, times_of_week)
Bar(name, address)
Drinker(name, address)

- Bars that drinkers in address "300 N. Duke Street" do not frequent

36

A trickier exercise

37

Frequents(drinker, bar, times_of_week)
Bar(name, address)
Drinker(name, address)

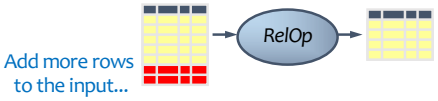
- For each bar, find the drinkers who frequent it max no. times a week

A deeper question:
When (and why) is "−" needed?

37

Monotone operators

38



- If some old output rows may need to be removed
  - Then the operator is non-monotone
- Otherwise the operator is monotone
  - That is, old output rows always remain "correct" when more rows are added to the input
- Formally, for a monotone operator op:
  $R \subseteq R'$ implies $op(R) \subseteq op(R')$ for any R, R'

38

Classification of relational operators

39

• Selection: $\sigma_p R$	Monotone
• Projection: $\pi_L R$	Monotone
• Cross product: $R \times S$	Monotone
• Join: $R \bowtie_p S$	Monotone
• Natural join: $R \bowtie S$	Monotone
• Union: $R \cup S$	Monotone
• Difference: $R - S$	Monotone w.r.t. R; non-monotone w.r.t S
• Intersection: $R \cap S$	Monotone

39

Why is "−" needed for "highest"?

40

- Composition of monotone operators produces a monotone query
  - Old output rows remain "correct" when more rows are added to the input
- Is the "highest" query monotone?
  - No!
  - Current max price is 3.0
  - Add another row with *price 3.01*
  - Old answer is invalidated

☞So it must use difference!

40

Extensions to relational algebra

41

- Duplicate handling ("bag algebra")
- Grouping and aggregation
- "Extension" (or "extended projection") to allow new column values to be computed

41