COMP 543: Tools & Models for Data Science

Relational Algebra

Chris Jermaine & Risa Myers

Rice University

RICE

Relational Calculus vs. Algebra

- In Relational Calculus
  - You say what you want
  - And not how to compute it
- But obviously...
  - This needs to be compiled into an actual computational plan
  - And in relational DBs, the plan is expressed in relational algebra
- RA is the “abstract machine” of relational databases

What Is An Algebra?

- Many Definitions!
  - Simplest: it is a set (domain) with a number of operations
  - The domain is closed under those operations
- In RA...
  - The domain is the set of all valid relations
  - The set of operations includes $\pi, \sigma, \times, \bowtie, \cup, \cap, -$
- Now let's go through the operations!

Projection

- Projection removes attributes
- $\pi_A(R)$...
  - A is a set of attributes of relation R
  - This simply removes all attributes not in A from R
  - Note: cardinality of output can differ from R
  - Output is a relation

FREQUENTS(DRINKER, CAFE)

FREQUENTS

DRINKER	CAFE
Risa	JL
Risa	BH
Chris	BH
Chris	DT

$\pi_{\text{DRINKER}}(\text{FREQUENTS})$

DRINKER
Risa
Chris

Selection

- Selection removes tuples
- $\sigma_B(R)$...
  - B is a boolean predicate that can be applied to a single tuple from R
  - This simply removes all tuples not accepted by B
  - Again: output is a relation

FREQUENTS

DRINKER	CAFE
Risa	JL
Risa	BH
Chris	BH
Chris	DT

$\sigma_{\text{DRINKER='Risa'}}(\text{FREQUENTS})$

DRINKER	CAFE
Risa	JL
Risa	BH

Selection/Projection Example

LIKES (DRINKER, COFFEE)
FREQUENTS (DRINKER, CAFE)
SERVES (CAFE, COFFEE)

? Query: Who likes ‘Cold Brew’ coffee?

Selection/Projection Example

LIKES (DRINKER, COFFEE)
FREQUENTS (DRINKER, CAFE)
SERVES (CAFE, COFFEE)

- Query: Who likes ‘Cold Brew’ coffee?
  - $\pi_{\text{DRINKER}}(\sigma_{\text{COFFEE='Cold Brew'}}(\text{LIKES}))$

Join: Cartesian Product

- Join combines tuples
- Simplest join is Cartesian product (aka: cross product)
- $R \times S$...
  - Returns $r \bullet s$ for all $r \in R, s \in S$
  - ? What is the output cardinality?

Join: Theta Join

LIKES (DRINKER, COFFEE)
FREQUENTS (DRINKER, CAFE)
SERVES (CAFE, COFFEE)

- Often you want $\sigma_B(R \times S)$
- Shorthand for this is $R \bowtie_B S$
- ? Query: Who likes a coffee that 'Risa' likes?

Join: Theta Join

LIKES (DRINKER, COFFEE)
FREQUENTS (DRINKER, CAFE)
SERVES (CAFE, COFFEE)

- Often you want $\sigma_B(R \times S)$
- Shorthand for this is $R \bowtie_B S$
- Query: Who likes a coffee that ‘Risa’ likes?
  - TEMP(d_1, c_1, d_2, c_2)
    $\leftarrow \text{LIKES} \bowtie_{\text{COFFEE}=\text{COFFEE}} (\sigma_{\text{DRINKER}=\text{'Risa'}}(\text{LIKES}))$
  - π_{d_1}(TEMP)

Join: Natural Join

LIKES (DRINKER, COFFEE)
FREQUENTS (DRINKER, CAFE)
SERVES (CAFE, COFFEE)

- Often you want to join two relations
  - Using an equality check on all attributes having the same name
  - Then project away redundant attributes
- Shorthand for this is $R * S$

? Query: Who goes to a cafe serving a coffee that they like?

Join: Natural Join

LIKES (DRINKER, COFFEE)
FREQUENTS (DRINKER, CAFE)
SERVES (CAFE, COFFEE)

- Often you want to join two relations
  - Using an equality check on all attributes having the same name
  - Then project away redundant attributes
- Shorthand for this is $R * S$
- Query: Who goes to a cafe serving a coffee that they like?
  - $\pi_{\text{DRINKER}}(\text{LIKES} * \text{FREQUENTS} * \text{SERVES})$

Set-Based Operations

LIKES (DRINKER, COFFEE)
FREQUENTS (DRINKER, CAFE)
SERVES (CAFE, COFFEE)

- Can use standard set operations as well: $\cup, \cap, -$
  - To use, types and numbers of input attributes must match
  - By convention, attribute names come from LHS
  - $R \cup S$: all tuples in R or in S
  - $R \cap S$: all tuples in R and in S
  - $R - S$: all tuples in R and not in S

? Query: Who does not like ‘Cold Brew’ coffee?

Set-Based Operations

LIKES (DRINKER, COFFEE)
FREQUENTS (DRINKER, CAFE)
SERVES (CAFE, COFFEE)

- Can use standard set operations as well: $\cup, \cap, -$
  - To use, types and numbers of input attributes must match
  - By convention, attribute names come from LHS
  - $R \cup S$: all tuples in R or in S
  - $R \cap S$: all tuples in R and in S
  - $R - S$: all tuples in R and not in S
- Query: Who does not like ‘Cold Brew’ coffee?
  - $\text{COLDBREWGOOD} \leftarrow \pi_{\text{DRINKER}}(\sigma_{\text{COFFEE}=\text{'Cold Brew'}}(\text{LIKES}))$
  - $(\pi_{\text{DRINKER}}(\text{FREQUENTS})) - \text{COLDBREWGOOD}$

Complicated Set-Based Example

- Query: Who does not like 'Cold Brew' coffee?
  - $\text{COLDBREWGOOD} \leftarrow \pi_{\text{DRINKER}}(\sigma_{\text{COFFEE}=\text{'Cold Brew'}}(\text{LIKES}))$
  - $(\pi_{\text{DRINKER}}(\text{FREQUENTS})) - \text{COLDBREWGOOD}$

? Why use FREQUENTS instead of LIKES?

Complicated Set-Based Example

LIKES (DRINKER, COFFEE)
FREQUENTS (DRINKER, CAFE)
SERVES (CAFE, COFFEE)

? Who only goes to cafes where they can get a coffee they like?

Complicated Set-Based Example

LIKES (DRINKER, COFFEE)
FREQUENTS (DRINKER, CAFE)
SERVES (CAFE, COFFEE)

- Who only goes to cafes where they can get a coffee they like?
  - Use 'all people' – 'those who go to a cafe where they can't get a coffee they like'
  - $\text{ALLPEEPS} \leftarrow \pi_{\text{DRINKER}}(\text{FREQUENTS})$
  - How about 'those who go to a cafe where they can't get a coffee they like'?
  - Use FREQUENTS – DRINKER, CAFE 'combos where the person can get a coffee they like'
  - $\text{GOODCOFFEE} \leftarrow \pi_{\text{DRINKER,CAFE}}(\text{LIKES} * \text{SERVES})$
- Then the answer is
  - $\text{ALLPEEPS} - \pi_{\text{DRINKER}}(\text{FREQUENTS} - \text{GOODCOFFEE})$

Questions?