CMPT 354: Database System I

Lecture 5. Relational Algebra

What have we learned

- Lec 1. Database History
- Lec 2. Relational Model
- Lec 3-4. SQL

Why Relational Algebra matter?

- An essential topic to understand how query processing and optimization work
  - What happened when an SQL is issued to a database?

- Help you master the kills to quickly learn a new query language
  - How to quickly learn XML QL and MangoDB QL?

Relational Query Languages

- Query languages allow the manipulation and retrieval of data from a database

- Traditionally: QL != programming language
  - Doesn’t need to be Turing complete
  - Not designed for computation
  - Supports easy, efficient access to large databases

- Recent Years:
  - Everything interesting involves a large data set
  - QLs are quite powerful for expressing algorithms at scale

Formal Query Languages

- Relational Algebra
  - Procedural query language
  - used to represent execution plans

- Relational Calculus
  - Non-procedural (declarative) query language
  - Describe **what** you want, rather than **how** to compute it
  - Foundation for SQL

Results of a Query

- Query is a function over relations

$$Q(R_1,\ldots,R_n) = R_{result}$$

- The schema of the result relation is determined by the input relation and the query
- Because the result of a query is a relation, it can be used as input to another query

Sets v.s. Bags

- Sets: {a, b, c}, {a, d, e, f}, {}, …
- Bags: {a, a, b, c}, {b, b, b, b, b}, …

- Relational Algebra has two flavors:
  - Set semantics = standard Relational Algebra
  - Bag semantics = extended Relational Algebra

- DB systems implement bag semantics (Why?)

Sets v.s. Bags

- Sets: {a, b, c}, {a, d, e, f}, {}, ...
- Bags: {a, a, b, c}, {b, b, b, b, b}, ...

- Relational Algebra has two flavors:
  - **Set semantics = standard Relational Algebra**
  - Bag semantics = extended Relational Algebra

- DB systems implement bag semantics (Why?)

Relational Algebra Operators

- Core 5 operators
  - Selection (σ)
  - Projection (π)
  - Union ($\cup$)
  - Set Difference (-)
  - Cross product (X)

- Additional operators
  - Rename (ρ)
  - join ($\bowtie$)
  - Intersect ($\cap$)

Selection

- The selection operator, σ (sigma), specifies the *rows* to be retained from the input relation
- A selection has the form: $\sigma_{condition}$(*relation*), where *condition* is a Boolean expression
  - Terms in the condition are comparisons between two fields (or a field and a constant)
  - Using one of the comparison operators: $<, \leq, =, \neq, \geq, >$
  - Terms may be connected by $\wedge$ (and), or $\vee$ (or),
  - Terms may be negated using $\neg$ (not)

Selection Example

Customer

sin	firstName	lastName	birth
111	Buffy	Summers	1981
222	Xander	Harris	1981
333	Cordelia	Chase	1980
444	Rupert	Giles	1955
555	Dawn	Summers	1984

$\sigma_{birth\ <\ 1981}$(Customer)

sin	firstName	lastName	birth
333	Cordelia	Chase	1980
444	Rupert	Giles	1955

$\sigma_{lastName\ =\ "Summers"}$(Customer)

sin	firstName	lastName	birth
111	Buffy	Summers	1981
555	Dawn	Summers	1984

Projection

- The projection operator, π (pi), specifies the columns to be retained from the input relation
- A selection has the form: $\pi_{columns}(relation)$
  - Where *columns* is a comma separated list of column names
  - The list contains the names of the columns to be retained in the result relation

Projection Example

Customer

sin	firstName	lastName	birth
111	Buffy	Summers	1981
222	Xander	Harris	1981
333	Cordelia	Chase	1980
444	Rupert	Giles	1955
555	Dawn	Summers	1984

$\pi_{firstName,lastName}$(Customer)

firstName	lastName
Buffy	Summers
Xander	Harris
Cordelia	Chase
Rupert	Giles
Dawn	Summers

π_{birth}(Customer)

birth
1981
1980
1955
1984

Selection and Projection Notes

- Selection and projection eliminate duplicates
  - Since relations are sets
- Both operations require one input relation
- The schema of the result of a selection is *the same as* the schema of the input relation
- The schema of the result of a projection contains just those attributes in the projection list

Composing Selection and Projection

$\pi_{sin,\ firstName}(\sigma_{birth\ <\ 1982\ \wedge\ lastName\ =\ "Summers"}(\text{Customer}))$

Customer

sin	firstName	lastName	birth
111	Buffy	Summers	1981
222	Xander	Harris	1981
333	Cordelia	Chase	1980
444	Rupert	Giles	1955
555	Dawn	Summers	1984

intermediate relation

sin	firstName	lastName	birth
111	Buffy	Summers	1981

sin	firstName
111	Buffy

Composing Selection and Projection

$\pi_{birth}(\sigma_{birth < 1981}(Customer))$

birth
1980
1955

Customer

sin	firstName	lastName	birth
111	Buffy	Summers	1981
222	Xander	Harris	1981
333	Cordelia	Chase	1980
444	Rupert	Giles	1955
555	Dawn	Summers	1984

$\sigma_{birth < 1981}(\pi_{birth}(Customer))$

birth
1980
1955

Commutative property

- For example:
  - x + y = y + x
  - x * y = y * x

- Does it hold for projection and selection?

$$\pi_{columns}(\sigma_{condition}(R)) = \pi_{condition}(\sigma_{columns}(R)) ?$$

- What about

$$\pi_{firstName}(\sigma_{birth < 1981}(\text{Customer}))?$$

Commutative property

$\pi_{firstName}(\sigma_{birth < 1981}(Customer))$

firstName
Cordelia
Rupert

Customer

sin	firstName	lastName	birth
111	Buffy	Summers	1981
222	Xander	Harris	1981
333	Cordelia	Chase	1980
444	Rupert	Giles	1955
555	Dawn	Summers	1984

firstName
Cordelia
Rupert

$\pi_{firstName}(\sigma_{birth < 1981}(\pi_{firstName, birth}(Customer)))$

Set Operations Review

A = {1, 3, 6} **B = {1, 2, 5, 6}**

Union ($\cup$)	$A \cup B \equiv B \cup A$	$A \cup B = \{1, 2, 3, 5, 6\}$	
Intersection($\cap$)	$A \cap B \equiv B \cap A$	$A \cap B = \{1, 6\}$	
Set Difference($-$)	$A - B \neq B - A$	$A - B = \{3\}$	$B - A = \{2, 5\}$

Union Compatible Relations

$$A \text{ op } B = R_{result}$$

- where op = $\cup$, $\cap$, or -
- *A* and *B* must be union compatible
  - Same number of fields
  - Field i in each schema have the same type

Union Compatible Relations

Intersection of the Employee and Customer relations

Customer

sin	firstName	lastName	birth
111	Buffy	Summers	1981
222	Xander	Harris	1981
333	Cordelia	Chase	1980
444	Rupert	Giles	1955
555	Dawn	Summers	1984

Employee

sin	firstName	lastName	salary
208	Clark	Kent	80000.55
111	Buffy	Summers	22000.78
412	Carol	Danvers	64000.00

The two relations are not union compatible as birth is a DATE and salary is a REAL

We can carry out preliminary operations to make the relations union compatible

$\pi_{sin,\ firstName,\ lastName}(\text{Customer}) \cap \pi_{sin,\ firstName,\ lastName}(\text{Employee})$

Union Compatible Relations

$$A \text{ op } B = R_{result}$$

- where op = ∪, ∩, or -
- *A* and *B* must be union compatible
  - Same number of fields
  - Field I in each schema have the same type

- Result schema borrowed from A

$$\mathbf{A}(\text{age int}) \cup \mathbf{B}(\text{num int}) = \mathbf{R_{result}}(\text{age int})$$

Union

A

sin	firstName	lastName
111	Buffy	Summers
222	Xander	Harris
333	Cordelia	Chase
444	Rupert	Giles
555	Dawn	Summers

B

sin	firstName	lastName
208	Clark	Kent
111	Buffy	Summers
412	Carol	Danvers

$A \cup B$

sin	firstName	lastName
111	Buffy	Summers
222	Xander	Harris
333	Cordelia	Chase
444	Rupert	Giles
555	Dawn	Summers
208	Clark	Kent
412	Carol	Danvers

Set Difference

A

sin	firstName	lastName
111	Buffy	Summers
222	Xander	Harris
333	Cordelia	Chase
444	Rupert	Giles
555	Dawn	Summers

A – B

sin	firstName	lastName
222	Xander	Harris
333	Cordelia	Chase
444	Rupert	Giles
555	Dawn	Summers

B

sin	firstName	lastName
208	Clark	Kent
111	Buffy	Summers
412	Carol	Danvers

B – A

sin	firstName	lastName
208	Clark	Kent
412	Carol	Danvers

Note on Set Difference

- Notice that most operators are monotonic
  - Increasing size of inputs → outputs grow
- Set Difference is non-monotonic
  - Example: A – B
  - Increasing the size of B could decrease output size
- Set difference is blocking:
  - For A – B, must wait for all B tuples before any results

Intersection

A

sin	firstName	lastName
111	Buffy	Summers
222	Xander	Harris
333	Cordelia	Chase
444	Rupert	Giles
555	Dawn	Summers

B

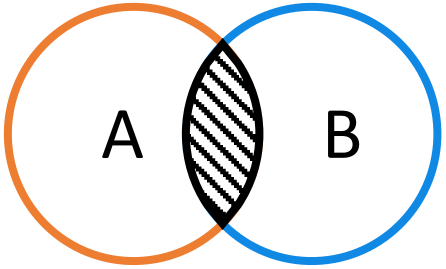
sin	firstName	lastName
208	Clark	Kent
111	Buffy	Summers
412	Carol	Danvers

$A \cap B$

sin	firstName	lastName
111	Buffy	Summers

Note on Intersect

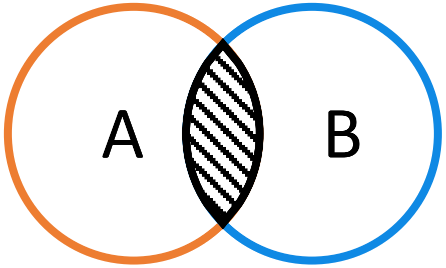
- $A \cap B = R_{result}$



- Can we express using other operators?
  - $A \cap B = ?$

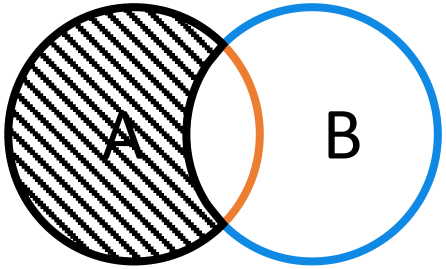
Note on Intersect

- $A \cap B = R_{result}$



- Can we express using other operators?
  - $A \cap B = A - (A - B)$

$A - B =$



Cartesian Product

$$A(a_1, \ldots, a_n) \times B(a_{n+1}, \ldots, a_m) = R_{result}(a_1, \ldots, a_m)$$

- Each row of A paired with each row of B
  - Result schema concats A and B’s fields
  - Names are inherited if possible (i.e. if not duplicated)
    - If two field names are the same (i.e., a *naming conflict* occurs) and the affected columns are referred to by position
  - If *R* contains *m* records, and *S* contains *n* records, the result relation will contain *m* * *n* records

Cartesian Product Example

$\sigma_{lastName = "Summers"}$(Customer)

sin	firstName	lastName	birth
111	Buffy	Summers	1981
555	Dawn	Summers	1984

Account

acc	type	balance	sin
01	CHQ	2101.76	111
02	SAV	11300.03	333
03	CHQ	20621.00	444

$\sigma_{lastName = "Summers"}$(Customer) × Account

1	firstName	lastName	birth	acc	type	balance	8
111	Buffy	Summers	1981	01	CHQ	2101.76	111
111	Buffy	Summers	1981	02	SAV	11300.03	333
111	Buffy	Summers	1981	03	CHQ	20621.00	444
555	Dawn	Summers	1984	01	CHQ	2101.76	111
555	Dawn	Summers	1984	02	SAV	11300.03	333
555	Dawn	Summers	1984	03	CHQ	20621.00	444

Renaming

- It is sometimes useful to assign names to the results of a relational algebra query
- The rename operator, ρ (rho)
  - $\rho_S(R)$ renames a relation
  - $\rho_{S(a1,a2,\ldots,an)}(R)$ renames a relation and its attributes
  - $\rho_{new/old}(R)$ renames specified attributes

R

sid1	firstName	lastName	birth	acc	type	balance	sid2
111	Buffy	Summers	1981	01	CHQ	2101.76	111
111	Buffy	Summers	1981	02	SAV	11300.03	333
111	Buffy	Summers	1981	03	CHQ	20621.00	444
555	Dawn	Summers	1984	01	CHQ	2101.76	111
555	Dawn	Summers	1984	02	SAV	11300.03	333
555	Dawn	Summers	1984	03	CHQ	20621.00	444

$$\rho_{sid1/1,\ sid2/8}(R)$$

Account = {accNumber, type, balance, branchName}

Largest Balance

- Find the account with the largest balance; return *accNumber*
  1. Find accounts which are less than some other account

$$\sigma_{account.balance < d.balance}(Account \times \rho_d(Account))$$

  2. Use set difference to find the account with the largest balance

$$\pi_{accNumber}(Account) - \pi_{account.accNumber}(\sigma_{account.balance < d.balance}(Account \times \rho_d(Account)))$$

Relational Algebra Operators

- Core 5 operations
  - Selection (σ)
  - Projection (π)
  - Union ($\cup$)
  - Set Difference (-)
  - Cross product (X)

- Additional operations
  - Rename (ρ)
  - Intersect ($\cap$)
  - **Join ($\bowtie$)**

Relational Algebra Exercises

- **Student** (<u>sID</u>, lastName, firstName, cgpa)
  - 101, Jordan, Michael, 3.8
- **Offering** (<u>oID</u>, dept, cNum, term, instructor)
  - abc, CMPT, 354, Fall 2018, Jiannan
- **Took** (<u>sID, oID</u>, grade)
  - 101, abc, 95

1. sID of all students who have earned some grade over 80 and some grade below 50.

$$\pi_{sID}(\sigma_{grade > 80}(Took)) \cap \pi_{sID}(\sigma_{grade < 50}(Took))$$

Relational Algebra Exercises

- **Student** (<u>sID</u>, lastName, firstName, cgpa)
  - 101, Jordan, Michael, 3.8
- **Offering** (<u>oID</u>, dept, cNum, term, instructor)
  - abc, CMPT, 354, Fall 2018, Jiannan
- **Took** (<u>sID, oID</u>, grade)
  - 101, abc, 95

2. Student number of all students who have taken CMPT 354

$$\pi_{sID}\left(\sigma_{Offering.oID = Took.oID \wedge dept = \text{'CMPT'} \wedge cNum = 354}\left(Offering \times Took\right)\right)$$

(Inner) Joins

- **Motivation**
  - **Simplify some queries that require a Cartesian product**

- **Natural Join:** $R \bowtie S = \pi_A(\sigma_\theta(R \times S))$

- **Theta Join:** $R \bowtie_\theta S = \sigma_\theta(R \times S)$

- **Equijoin:** $R \bowtie_\theta S = \sigma_\theta(R \times S)$
  - Join condition θ consists only of equalities

Natural Join

- There is often a natural way to join two relations
  - Join based on common attributes
  - Eliminate duplicate common attributes from the result

Customer

sin	firstName	lastName	birth
111	Buffy	Summers	1981
222	Xander	Harris	1981
333	Cordelia	Chase	1980
444	Rupert	Giles	1955

Employee

sin	firstName	lastName	salary
208	Clark	Kent	80000.55
111	Buffy	Summers	22000.78
396	Dawn	Allen	41000.21

Customer ⋈ Employee

sin	firstName	lastName	birth	salary
111	Buffy	Summers	1981	22000.78

Natural Join

$$\mathbf{R \bowtie S}$$

- Meaning: $R \bowtie S = \pi_A(\sigma_\theta\ (R \times S))$
- Where:
  - Selection σ_θ checks equality of all common attributes (i.e., attributes with same names)
  - Projection π_A eliminates duplicate common attributes
- The natural join of two tables with *no fields in common is the Cartesian product*
  - Not the empty set

Natural Join Example

R

A	B	C	D
111	Buffy	Summers	1981
222	Xander	Harris	1981
333	Cordelia	Chase	1980
444	Rupert	Giles	1955

S

A	B	C	E
208	Clark	Kent	80000.55
111	Buffy	Summers	22000.78
396	Dawn	Allen	41000.21

$$R \bowtie S = \pi_{A,B,C,D,E}(\sigma_{R.A=S.A \wedge R.B=S.B \wedge R.C=S.C}(R \times S))$$

A	B	C	D	E
111	Buffy	Summers	1981	22000.78

Theta Join

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

- Most general form
  - θ can be any condition
- No projection in this case!
  - Result schema same as cross product

Theta Join Example

Customer

sin	firstName	lastName	birth
111	Buffy	Summers	1981
222	Xander	Harris	1981
333	Cordelia	Chase	1980
444	Rupert	Giles	1955
555	Dawn	Summers	1984

Employee

sin	firstName	lastName	salary
208	Clark	Kent	80000.55
111	Buffy	Summers	22000.78
412	Carol	Danvers	64000.00

Customer $\bowtie_{\text{Customer.sin} < \text{Employee.sin}}$ **Employee**

1	2	3	birth	5	6	7	salary
111	Buffy	Summers	1981	208	Clark	Kent	80000.55
111	Buffy	Summers	1981	412	Carol	Danvers	64000.00
222	Xander	Harris	1981	412	Carol	Danvers	64000.00
333	Cordelia	Chase	1980	412	Carol	Danvers	64000.00

Equi-Joins

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

- A theta join where θ is an equality predicate

Customer

sin	firstName	lastName	birth
111	Buffy	Summers	1981
222	Xander	Harris	1981
333	Cordelia	Chase	1980
444	Rupert	Giles	1955

Employee

sin	firstName	lastName	salary
208	Clark	Kent	80000.55
111	Buffy	Summers	22000.78
396	Dawn	Allen	41000.21

Customer $\bowtie_{Customer.sin = Employee.sin}$ Employee

1	2	3	birth	5	6	7	salary
111	Buffy	Summers	1981	111	Buffy	Summers	22000.78

(Inner) Joins Summary

- **Natural Join:** $R \bowtie S = \pi_A(\sigma_\theta(R \times S))$
  - Equality on all fields with same name in R and in S
  - Projection π_A drops all redundant attributes
- **Theta Join:** $R \bowtie_\theta S = \sigma_\theta(R \times S)$
  - Join of R and S with a join condition θ
  - Cross-product followed by selection θ
  - No projection
- **Equijoin:** $R \bowtie_\theta S = \sigma_\theta(R \times S)$
  - Join condition θ consists only of equalities
  - No projection

Relational Algebra Exercises

- **Student** (<u>sID</u>, lastName, firstName, cgpa)
  - 101, Jordan, Michael, 3.8
- **Course** (<u>dept, cNum</u>, name, breadth)
  - CMPT, 354, DB, True
- **Offering** (<u>oID</u>, dept, cNum, term, instructor)
  - abc, CMPT, 354, Fall 2018, Jiannan
- **Took** (<u>sID, oID</u>, grade)
  - 101, abc, 95

The names of all students who have passed a breadth course (grade >= 60 and breadth = True) with Martin

$$\pi_{lastName,\ firstName}(\sigma_{breadth\ =\ True\ \wedge\ grade\ >\ 60\ \wedge\ instructor\ =\ 'Martin'}(\text{Student} \bowtie \text{Took} \bowtie \text{Offering} \bowtie \text{Course}))$$

Different Plans, Same Results

- Semantic equivalence: results are *always* the same

$$\pi_{name}(\sigma_{cNum=354}(R \bowtie S))$$

$$\pi_{name}(\sigma_{cNum=354}(R) \bowtie S)$$

- Are they equivalent?
- Which one is more efficient?
- Can you make it even more efficient?

Other Operators

- There are additional relational algebra operators
  - Usually used in the context of query optimization
- Duplicate elimination – δ
  - Used to turn a bag into a set
- Aggregation operators
  - e.g. sum, average
- Grouping – γ
  - Used to partition tuples into groups
    - Typically used with aggregation

Summary

- Relational Algebra (RA) operators
  - Five core operators: selection, projection, cross-product, union and set difference
  - Additional operators are defined in terms of the core operators: rename, intersection, join
- Theorem: SQL and RA can express exactly the same class of queries
- Multiple RA queries can be equivalent
  - Same semantics but difference performance
  - Form basis for optimizations

RDBMS translate SQL → RA, then optimize RA

Acknowledge

- Some lecture slides were copied from or inspired by the following course materials
  - “W4111: Introduction to databases” by Eugene Wu at Columbia University
  - “CSE344: Introduction to Data Management” by Dan Suciu at University of Washington
  - “CMPT354: Database System I” by John Edgar at Simon Fraser University
  - “CS186: Introduction to Database Systems” by Joe Hellerstein at UC Berkeley
  - “CS145: Introduction to Databases” by Peter Bailis at Stanford
  - “CS 348: Introduction to Database Management” by Grant Weddell at University of Waterloo