L4 Relational Algebra

Eugene Wu Fall 2015

Overview

Last time, learned about pre-relational models an informal introduction to relational model an introduction to the SQL query language.

Learn about formal relational query languages
Relational Algebra (algebra: perform operations)
Relational Calculus (logic: are statements true?)

Keys to understanding SQL and query processing

Who Cares?

Query Semantics

Concise domain specific language (DSL)

Rich program analysis

Helps/enables optimization

Opens up rich set of topics

Materialized views

Data lineage/provenance

Query by example

...

Reading

Ramakrishnan

Optional

Helpful References

https://en.wikipedia.org/wiki/Relational_algebra

What's a Query Language?

Allows 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

Scaling to large datasets is a reality
Query languages are a powerful way to
think about data algorithms scale
think about asynchronous/parallel programming

Formal Relational Query Languages

Formal basis for real languages e.g., SQL

Relational Algebra

Operational, used to represent execution plans

Relational Calculus

Logical, describes what data users want (not operational, fully declarative)

Prelims

Query is a function over relation instances

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

Schemas of input and output relations are *fixed* and well defined by the query O.

Positional vs Named field notation Position easier for formal defs

one-indexed (not 0-indexed!!!)

Named more readable Both used in SQL

Prelims

Relation (for this lecture) Instance is a set of tuples

Schema defines field names and types (domains)

Students(sid, name, bday, major, gpa)

How are relations different than generic sets (\mathbb{R})?

Can assume item structure due to schema

Some algebra operations (x) need to be modified

Will use this later

Relational Algebra Overview

Core 5 operations PROJECT (π)

SELECT (σ)

UNION (U)
SET DIFFERENCE (-)
CROSSPRODUCT (x)

Additional operations

RENAME (p)
INTERSECT (∩)
JOIN (⋈)
DIVIDE (/)

Fields in query results are inherited from input relations (unless specified)

Students, Reservations,

Use positional or named

Books

field notation

Instances Used Today: Library

SI

S2

sid rid day
| 101 10/10
| 2 102 11/11

 sid
 name
 gpa
 age

 I
 eugene
 4
 20

 2
 barak
 3
 21

 3
 trump
 2
 88

 sid
 name
 gpa
 age

 4
 aziz
 3.2
 21

 2
 barak
 3
 21

 3
 trump
 2
 88

 5
 rusty
 3.5
 21

Project

$$\pi_{}(R_{in}) = R_{result}$$

Pick out desired attributes (subset of columns)
Schema is subset of input schema in the projection list

 $\pi_{< a,\,b,\,c>}(R_{in})$ has output schema (a,b,c) with types carried over

Project

 sid
 name
 gpa
 age

 4
 aziz
 3.2
 21

 2
 barak
 3
 21

 3
 trump
 2
 88

 5
 rusty
 3.5
 21

 $\pi_{\text{name,age}}(S2) = \begin{bmatrix} name & age \\ aziz & 21 \\ barak & 21 \\ trump & 88 \\ rusty & 21 \end{bmatrix}$

Project

| S2 | sid | name | gpa | age |
|----|-----|-------|-----|-----|
| | 4 | aziz | 3.2 | 21 |
| | 2 | barak | 3 | 21 |
| | 3 | trump | 2 | 88 |
| | 5 | rusty | 3.5 | 21 |

$$\pi_{age}(S2) = \begin{bmatrix} age \\ 21 \\ 88 \end{bmatrix}$$

Where did all the rows go?
Real systems typically don't remove duplicates. Why?

Select

$$\sigma_{\leq p}(R_{in}) = R_{result}$$

Select subset of rows that satisfy condition p Won't have duplicates in result Why? Result schema same as input

Select

$$\sigma_{age < 30} \text{ (S1)} = \begin{bmatrix} \frac{\text{sid}}{\text{name}} & \frac{\text{name}}{\text{gpa}} & \frac{\text{age}}{\text{20}} \\ \frac{\text{I}}{\text{2}} & \frac{\text{eugene}}{\text{barak}} & \frac{4}{\text{3}} & \frac{20}{\text{21}} \end{bmatrix}$$

$$\pi_{name}(\sigma_{age < 30} \text{ (S1)}) = \begin{bmatrix} \frac{\text{name}}{\text{eugene}} & \frac{1}{\text{20}} & \frac{1}{\text$$

Union, Set-Difference

$$R_1 \circ_P R_2 = R_{result}$$

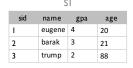
R_1, R_2 must be union-compatible

Same number of fields
Field i in each schema have same type

Result Schema borrowed from first $arg(R_1)$

RI (big int, poppa int) U R2(thug int, life int) =?

Union, Intersect, Set-Difference



| 32 | | | | | | | |
|-----|-------|-----|-----|--|--|--|--|
| sid | name | gpa | age | | | | |
| 4 | aziz | 3.2 | 21 | | | | |
| 2 | barak | 3 | 21 | | | | |
| 3 | trump | 2 | 88 | | | | |
| 5 | rusty | 3.5 | 21 | | | | |

Union, Intersect, Set-Difference



| 32 | | | | | | | |
|-----|-------|-----|-----|--|--|--|--|
| sid | name | gpa | age | | | | |
| 4 | aziz | 3.2 | 21 | | | | |
| 2 | barak | 3 | 21 | | | | |
| 3 | trump | 2 | 88 | | | | |
| 5 | rusty | 3.5 | 21 | | | | |

$$SI-S2 = \begin{bmatrix} sid & name & gpa & age \\ I & eugene & 4 & 20 \end{bmatrix}$$

Note on Set Difference & Performance

Notice that most operators are monotonic increasing size of inputs \Rightarrow outputs grow if $S1 \supseteq S2 \Rightarrow Q(S1,T) \supseteq Q(S2,T)$ can compute incrementally

Set Difference is not monotonic

e.g.,
$$T - SI$$

if $SI \supseteq S2 \rightarrow T - SI \subseteq T - S2$

Set difference is blocking:

For T-S, must wait for all S tuples before any results

Cross-Product

 $R_1(a_1,...,a_n) \times R_2(a_{n+1},...,a_m) = R_{result}(a_1,...,a_m)$

Each row of SI paired with each row of RI

Result schema concats SI and RI's fields, inherit if possible

Conflict SI and RI have sid field

Different than mathematical "X" by flattening results: math $A \times B = \{ (a, b) | a \in A \land b \in B \}$

e.g., {1, 2}
$$\times$$
 {3, 4} = { (1, 3), (1, 4), (2, 3), (2, 4) } what is {1, 2} \times {3, 4} \times {5, 6}?

Cross-Product

| sid | name | gpa | age |
|-----|--------|-----|-----|
| 1 | eugene | 4 | 20 |
| 2 | barak | 3 | 21 |
| 3 | trump | 2 | 88 |

| | RI | |
|-----|-----|-------|
| sid | rid | day |
| I | 101 | 10/10 |
| 2 | 102 | 11/11 |

| | (sid) | name | gpa | age | (sid) | rid | day |
|---------|-------|--------|-----|-----|-------|-----|-------|
| SIx R1= | 1 | eugene | 4 | 20 | 1 | 101 | 10/10 |
| | 2 | barak | 3 | 21 | 1 | 101 | 10/10 |
| | 3 | trump | 2 | 88 | 1 | 101 | 10/10 |
| | 1 | eugene | 4 | 20 | 2 | 102 | 11/11 |
| | 2 | barak | 3 | 21 | 2 | 102 | 11/11 |
| | 2 | trumn | 2 | 00 | 2 | 102 | 11/11 |

Rename

p(<new_name>(<mappings>), Q)

Explicitly defines/changes field names of schema

$$p(C(1 \rightarrow sid1, 5 \rightarrow sid2), S1 \times R1)$$

Intersect

 $R_1 \cap R_2 = R_{result}$

| | sid I | name | gpa | age | sid2 | rid | day |
|-----|-------|--------|-----|-----|------|-----|-------|
| | 1 | eugene | 4 | 20 | 1 | 101 | 10/10 |
| | 2 | barak | 3 | 21 | I | 101 | 10/10 |
| C = | 3 | trump | 2 | 88 | I | 101 | 10/10 |
| | I | eugene | 4 | 20 | 2 | 102 | 11/11 |
| | 2 | barak | 3 | 21 | 2 | 102 | 11/11 |
| | 3 | trump | 2 | 88 | 2 | 102 | 11/11 |

Compound/Convenience Operators

Convenience operations INTERSECT (\cap)

JOIN (⋈) DIVIDE (/)

 R_1, R_2 must be union-compatible

Intersect

| | SI | | | | | | |
|---|----|--------|-----|-----|--|--|--|
| s | id | name | gpa | age | | | |
| 1 | | eugene | 4 | 20 | | | |
| 2 | | barak | 3 | 21 | | | |
| 3 | | trump | 2 | 88 | | | |

| JZ. | | | | | | | |
|-----|-------|-----|-----|--|--|--|--|
| sid | name | gpa | age | | | | |
| 4 | aziz | 3.2 | 21 | | | | |
| 2 | barak | 3 | 21 | | | | |
| 3 | trump | 2 | 88 | | | | |
| 5 | rusty | 3.5 | 21 | | | | |

$$SI \cap S2 = \begin{bmatrix} sid & name & gpa & age \\ 2 & barak & 3 & 21 \\ 3 & trump & 2 & 88 \end{bmatrix}$$

Intersect

 $R_1 \cap R_2 = R_{result}$

 R_1, R_2 must be union-compatible

Can we express using core operators?

$$S \cap T = ?$$

Intersect

 $R_1 \cap R_2 = R_{result}$

R₁, R₂ must be union-compatible

Can we express using core operators?

 $S \cap T = S - ?$ (think venn diagram)

Intersect

 $R_1 \cap R_2 = R_{result}$

 R_1, R_2 must be union-compatible

Can we express using core operators?

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

theta (θ) Join

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

Most general form

Result schema same as cross product

Often far more efficient to compute than cross product

Commutative

 $(A\bowtie_c B)\bowtie_c C = A\bowtie_c (B\bowtie_c C)$

theta (θ) Join

| sid | name | gpa | age |
|-----|--------|-----|-----|
| 1 | eugene | 4 | 20 |
| 2 | barak | 3 | 21 |
| 3 | trump | 2 | 88 |

| | RI | |
|-----|-----|-------|
| sid | rid | day |
| 1 | 101 | 10/10 |
| 2 | 102 | 11/11 |

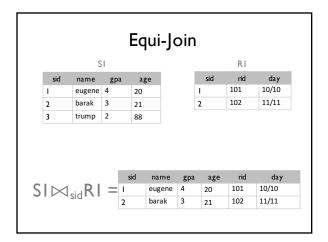
| | (sid) | name | gpa | age | (sid) | rid | day |
|--------------------------------------|-------|--------|-----|-----|-------|-----|-------|
| SI ⋈ _{SI.sid ≤ RI.sid} RI = | 1 | eugene | 4 | 20 | 1 | 101 | 10/10 |
| | I | eugene | 4 | 20 | 2 | 102 | 11/11 |
| | 2 | barak | 3 | 21 | 2 | 102 | 11/11 |

Equi-Join

 $R \bowtie_{attr} S = R \bowtie_{R.attr = S.attr} S$

Special case where the condition is attribute equality Result schema only keeps one copy of equality fields Natural Join (R⋈S):

Equijoin on all shared fields



Division

Let us have relations A(x, y), B(y)

$$A/B = \{ <_X > | \exists <_X, y > \in A \forall A <_Y > \in B \}$$

Good to ponder, not supported in most systems (why?)

Find all students that have reserved all books

Generalization

y can be a list of fields in B

x U y is fields in A

| | | Examp | nes | |
|-----|-----|-------|------|------|
| | A | RI | R2 | R3 |
| sid | rid | rid | rid | rid |
| I | I | 2 | 2 | 1 |
| I | 2 | | 4 | 2 |
| I | 3 | | | 4 |
| I | 4 | sid | | |
| 2 | I | I | | |
| 2 | 2 | 2 | sid | |
| 3 | 2 | 3 | I | sid |
| 4 | 2 | 4 | 4 | I |
| 4 | 4 | A/RI | A/R2 | A/R3 |

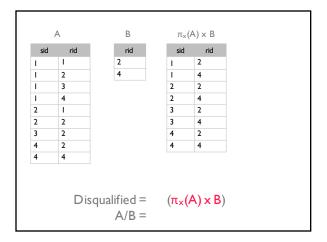
Is A/B a Fundamental Operation?

No. Shorthand like Joins

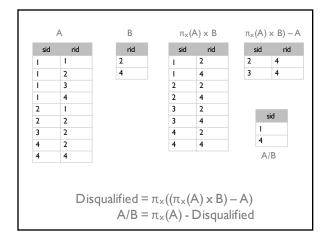
joins so common, it's natively supported

Hint: Find all xs not 'disqualified' by some y in B.
x value is disqualified if by attaching y value from B,
we obtain an xy tuple that is not in A.

| sid rid 2 2 4 4 3 4 4 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 |
|---|
| 2 3 4 1 2 |
| 3 4 1 2 |
| 4 I 2 |
| 1 2 |
| 2 |
| |
| 2 |
| |
| 2 |
| 4 |



| | А | В | π_{x} (| A) x B | $\pi_x(A)\times B)-A$ |
|-----|-----|-----------------------|--------------------|---------|-----------------------|
| sid | rid | rid | sid | rid | sid rid |
| 1 | I | 2 | 1 | 2 | 2 4 |
| 1 | 2 | 4 | I | 4 | 3 4 |
| 1 | 3 | | 2 | 2 | |
| 1 | 4 | | 2 | 4 | |
| 2 | I | | 3 | 2 | |
| 2 | 2 | | 3 | 4 | |
| 3 | 2 | | 4 | 2 | |
| 4 | 2 | | 4 | 4 | |
| 4 | 4 | | | | |
| | Dis | squalified = A/B = | ((π _× (| A) × B) | -A) |



Names of students that reserved book 2 $\pi_{name}(\sigma_{rid=2} \ (R1) \bowtie SI)$ Equivalent $Queries \qquad \begin{array}{l} p(Tmp1, \, \sigma_{rid=2} \ (R1)) \\ p(Tmp2, \, Tmp1 \bowtie SI) \\ \pi_{name}(Tmp2) \end{array}$ $\pi_{name}(\sigma_{rid=2}(R1 \bowtie SI))$

Names of students that reserved db books

Book(bid, type) Reserve(sid, bid) Student(sid)

Need to join with Instructors to access role

 $\pi_{\text{name}}(\sigma_{\text{type='db'}}(\text{Book})\bowtie \text{Reserve}\bowtie \text{Student})$

More efficient query

 $\pi_{name}(\pi_{sid}((\ \pi_{cid}\sigma_{type='db'}\ (Book))\ \bowtie\ Reserve)\ \bowtie\ Student)$

Query optimizer can find the more efficient query!

Students that reserved DB or HCI book

- I. Find all DB or HCl books
- 2. Find students that reserved one of those books

 $p(tmp, (\sigma_{type='DB' \ v \ type='HCI'} (Book)))$ $\pi_{name}(tmp \bowtie Reserve \bowtie Student)$

Alternatives

define tmp using UNION (how?) what if we replaced v with ^ in the query?

Students that reserved a DB and HCI book

Does previous approach work?

 $p(tmp, (\sigma_{type='DB' \land type='HCI'} (Book))$ $\pi_{name}(tmp \bowtie Reserve \bowtie Student)$



Students that reserved a DB and HCI book

Does previous approach work?

- I. Find students that reserved DB books
- 2. Find students that reversed HCI books
- 3. Intersection

$$\begin{split} &p(tmpDB, \pi_{sid}(\sigma_{type='DB'} Book) \bowtie Reserve) \\ &p(tmpHCI, \pi_{sid}(\sigma_{type='HCI'} Book) \bowtie Reserve) \\ &\pi_{namel}(tmpDB \cap tmpHCI) \bowtie Student) \end{split}$$

Is the intersection always allowed? What if it projected book name?

Students that reserved all books

Use division
Be careful with schemas of inputs to /!

 $\begin{array}{c} p(tmp,(\pi_{sid,bid} \ Reserves) \ / \ (\pi_{bid} Books)) \\ \pi_{name}(tmp \bowtie Student) \end{array}$

What if want students that reserved all horror books?

Summary

Relational Algebra (RA) operators

Operators are closed (inputs & outputs are relations)

Multiple Relational Algebra queries can have same semantics (always return same results)

Forms basis for optimizations

Next Time

Relational Calculus