Lecture 33: The Relational Model 2

Professor Xiannong Meng

Spring 2018

Lecture and activity contents are based on what Prof Chris Ré of Stanford used in his CS 145 in the fall 2016 term with permission

Relational Algebra (RA)

- Five **basic** operators:
  1. Selection: σ
  2. Projection: Π
  3. Cartesian Product: $\times$
  4. Union: $\cup$
  5. Difference: -
- Derived or auxiliary operators:
  - Intersection, complement
  - Joins (natural,equi-join, theta join, semi-join)
  - Renaming: ρ
  - Division

We'll look at these first!

And also at one example of a derived operator (natural join) and a special operator (renaming)

1. Selection (σ)

- Returns all tuples which satisfy a condition
- Notation: $\sigma_c(R)$
- Examples
  - $\sigma_{Salary > 40000}$ (Employee)
  - $\sigma_{name = \text{"Smith"}}$ (Employee)
- The condition c can be =, <, ≤, >, ≥, <>

Students(sid,sname,gpa)

SQL:

```
SELECT *
FROM Students
WHERE gpa > 3.5;
```

RA:

$\sigma_{gpa > 3.5}(Students)$

2. Projection (Π)

- Eliminates columns, then removes duplicates
- Notation: $\Pi_{A1,\ldots,An}(R)$
- Example: project social-security number and names:
  - $\Pi_{SSN, Name}$ (Employee)
  - Output schema: Answer(SSN, Name)

Students(sid,sname,gpa)

SQL:

```
SELECT DISTINCT
  sname,
  gpa
FROM Students;
```

RA:

$\Pi_{sname,gpa}(Students)$

Note that RA Operators are Compositional!

Students(sid,sname,gpa)

```
SELECT DISTINCT
  sname,
  gpa
FROM Students
WHERE gpa > 3.5;
```

How do we represent this query in RA?

$\Pi_{sname,gpa}(\sigma_{gpa>3.5}(Students))$

$\sigma_{gpa>3.5}(\Pi_{sname,gpa}(Students))$

Are these logically equivalent?

3. Cross-Product ($\times$)

- Each tuple in R1 with each tuple in R2
- Notation: R1 × R2
- Example:
  - Employee × Dependents
- Rare in practice; mainly used to express joins

Students(sid,sname,gpa)
People(ssn,pname,address)

SQL:

```
SELECT *
FROM Students, People;
```

RA:

$Students \times People$

Another example:

People

ssn	pname	address
1234545	John	216 Rosse
5423341	Bob	217 Rosse

×

Students

sid	sname	gpa
001	John	3.4
002	Bob	1.3

$Students \times People$

ssn	pname	address	sid	sname	gpa
1234545	John	216 Rosse	001	John	3.4
5423341	Bob	217 Rosse	001	John	3.4
1234545	John	216 Rosse	002	Bob	1.3
5423341	Bob	216 Rosse	002	Bob	1.3

Renaming (ρ)

Students(sid,sname,gpa)

- Changes the schema, not the instance
- A 'special' operator- neither basic nor derived
- Notation: $\rho_{B1,\ldots,Bn}$ (R)
- **Note: this is shorthand for the proper form (since names, not order matters!):**
  - $\rho_{A1 \rightarrow B1,\ldots,An \rightarrow Bn}$ (R)

SQL:

```
SELECT
  sid AS studId,
  sname AS name,
  gpa AS gradePtAvg
FROM Students;
```

RA:

$\rho_{studId,name,gradePtAvg}(Students)$

We care about this operator *because* we are working in a *named perspective*

Another example:

Students

sid	sname	gpa
001	John	3.4
002	Bob	1.3

$\rho_{studId,name,gradePtAvg}(Students)$

Students

studId	name	gradePtAvg
001	John	3.4
002	Bob	1.3

Natural Join (⋈)

Students(sid,name,gpa)
People(ssn,name,address)

- Notation: $R_1 \bowtie R_2$
- Joins R_1 and R_2 on *equality of all shared attributes*
  - If R_1 has attribute set A, and R_2 has attribute set B, and they share attributes $A \cap B = C$, can also be written: $R_1 \bowtie_C R_2$
- Our first example of a *derived* RA operator:
  - Meaning: $R_1 \bowtie R_2 = \Pi_{A \cup B}(\sigma_{C=D}(\rho_{C \rightarrow D}(R_1) \times R_2))$
  - Where:
    - The rename $\rho_{C \rightarrow D}$ renames the shared attributes in one of the relations
    - The selection $\sigma_{C=D}$ checks equality of the shared attributes
    - The projection $\Pi_{A \cup B}$ eliminates the duplicate common attributes

SQL:

```
SELECT DISTINCT
  ssid, S.name, gpa,
  ssn, address
FROM
  Students S,
  People P
WHERE S.name = P.name;
```

RA:

$Students \bowtie People$

Another example:

Students S

sid	S.name	gpa
001	John	3.4
002	Bob	1.3

⋈

People P

ssn	P.name	address
1234545	John	216 Rosse
5423341	Bob	217 Rosse

$Students \bowtie People$

sid	S.name	gpa	ssn	address
001	John	3.4	1234545	216 Rosse
002	Bob	1.3	5423341	216 Rosse

Natural Join

- Given schemas R(A, B, C, D), S(A, C, E), what is the schema of R ⋈ S ?
- Given R(A, B, C), S(D, E), what is R ⋈ S ?
- Given R(A, B), S(A, B), what is R ⋈ S ?

Example: Converting SFW Query -> RA

Students(sid,sname,gpa)
People(ssn,sname,address)

```
SELECT DISTINCT
  gpa,
  address
FROM Students S,
     People P
WHERE gpa > 3.5 AND
  sname = pname;
```

$\Pi_{gpa,address}(\sigma_{gpa>3.5}(S \bowtie P))$

How do we represent this query in RA?

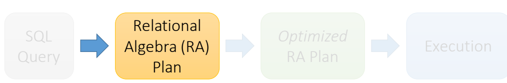
Logical Equivalence of RA Plans

- Given relations R(A,B) and S(B,C):
  - Here, projection & selection commute:
    - $\sigma_{A=5}(\Pi_A(R)) = \Pi_A(\sigma_{A=5}(R))$
  - What about here?
    - $\sigma_{A=5}(\Pi_B(R)) ?= \Pi_B(\sigma_{A=5}(R))$

We'll look at this in more depth later in the lecture…

RDBMS Architecture

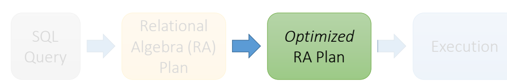
How does a SQL engine work ?



We saw how we can transform declarative SQL queries into precise, compositional RA plans

RDBMS Architecture

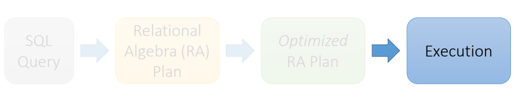
How does a SQL engine work ?



We'll look at how to then optimize these plans later in this lecture

RDBMS Architecture

How is the RA "plan" executed?



We already know how to execute all the basic operators!

RA Plan Execution

- Natural Join / Join:
  - We saw how to use **memory & IO cost considerations to pick the correct algorithm to execute a join with (BNLJ, SMJ, HJ…)!**
- Selection:
  - We saw how to use **indexes to aid selection**
  - Can always fall back on scan / binary search as well
- Projection:
  - The main operation here is finding *distinct* values of the project tuples; we briefly discussed how to do this with e.g. **hashing** or **sorting**

We already know how to execute all the basic operators!

2. Adv. Relational Algebra

What you will learn about in this section

1. Set Operations in RA
2. Fancier RA
3. Extensions & Limitations

Relational Algebra (RA)

- Five **basic** operators:
  1. Selection: σ
  2. Projection: Π
  3. Cartesian Product: $\times$
  4. Union: $\cup$
  5. Difference: -

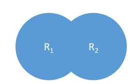
We'll look at these

- Derived or auxiliary operators:
  - Intersection, complement
  - Joins (natural,equi-join, theta join, semi-join)
  - Renaming: ρ
  - Division

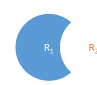
And also at some of these derived operators

1. Union ($\cup$) and 2. Difference (–)

- R1 $\cup$ R2
- Example:
  - ActiveEmployees $\cup$ RetiredEmployees

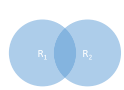


- R1 – R2
- Example:
  - AllEmployees - RetiredEmployees



What about Intersection ($\cap$) ?

- It is a derived operator
- R1 $\cap$ R2 = R1 – (R1 – R2)
- Also expressed as a join!
- Example
  - UnionizedEmployees $\cap$ RetiredEmployees



Fancier RA

Theta Join ($\bowtie_\theta$)

- A join that involves a predicate
- $R1 \bowtie_\theta R2 = \sigma_\theta (R1 \times R2)$
- Here θ can be any condition

Note that natural join is a theta join + a projection.

Students(sid,sname,gpa)
People(ssn,pname,address)

SQL:

```
SELECT *
FROM
  Students,People
WHERE θ;
```

RA:

$$Students \bowtie_\theta People$$

Equi-join ($\bowtie_{A=B}$)

- A theta join where θ is an equality
- $R1 \bowtie_{A=B} R2 = \sigma_{A=B} (R1 \times R2)$
- Example:
  - Employee $\bowtie_{SSN=SSN}$ Dependents

Most common join in practice!

Students(sid,sname,gpa)
People(ssn,pname,address)

SQL:

```
SELECT *
FROM
  Students S,
  People P
WHERE sname = pname;
```

RA:

$$S \bowtie_{sname=pname} P$$

Semijoin ($\ltimes$)

- $R \ltimes S = \Pi_{A1,\ldots,An} (R \bowtie S)$
- Where $A_1, \ldots, A_n$ are the attributes in R
- Example:
  - Employee $\ltimes$ Dependents

Students(sid,sname,gpa)
People(ssn,pname,address)

SQL:

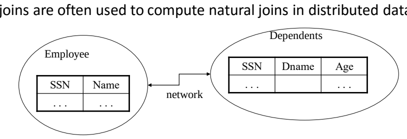
```
SELECT DISTINCT
  sid,sname,gpa
FROM
  Students,People
WHERE
  sname = pname;
```

RA:

$$Students \ltimes People$$

Semijoins in Distributed Databases

- Semijoins are often used to compute natural joins in distributed databases



Send less data to reduce network bandwidth!

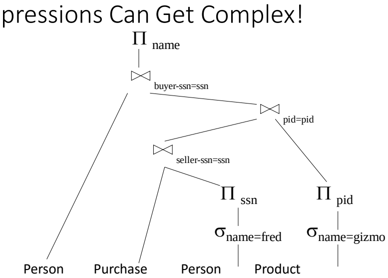
$$Employee \bowtie_{ssn=ssn} (\sigma_{age>71} (Dependents))$$

$T = \Pi_{SSN}\ \sigma_{age>71}$ (Dependents)

$R = Employee \ltimes T$

$Answer = R \bowtie Dependents$

RA Expressions Can Get Complex!



Multisets

Recall that SQL uses Multisets

$\lambda(X)$= "Count of tuple in X"
(Items not listed have implicit count 0)

Multiset X

Tuple
(1, a)
(1, a)
(1, b)
(2, c)
(2, c)
(2, c)
(1, d)
(1, d)

Equivalent Representations of a Multiset

Multiset X

Tuple	$\lambda(X)$
(1, a)	2
(1, b)	1
(2, c)	3
(1, d)	2

Note: In a set all counts are {0,1}.

31

Generalizing Set Operations to Multiset Operations

Multiset X

Tuple	$\lambda(X)$
(1, a)	2
(1, b)	0
(2, c)	3
(1, d)	0

$\cap$

Multiset Y

Tuple	$\lambda(Y)$
(1, a)	5
(1, b)	1
(2, c)	2
(1, d)	2

$=$

Multiset Z

Tuple	$\lambda(Z)$
(1, a)	2
(1, b)	0
(2, c)	2
(1, d)	0

$$\lambda(Z) = min(\lambda(X), \lambda(Y))$$

For sets, this is **intersection**

32

Generalizing Set Operations to Multiset Operations

Multiset X

Tuple	$\lambda(X)$
(1, a)	2
(1, b)	0
(2, c)	3
(1, d)	0

$\cup$

Multiset Y

Tuple	$\lambda(Y)$
(1, a)	5
(1, b)	1
(2, c)	2
(1, d)	2

$=$

Multiset Z

Tuple	$\lambda(Z)$
(1, a)	7
(1, b)	1
(2, c)	5
(1, d)	2

$$\lambda(Z) = \lambda(X) + \lambda(Y)$$

For sets, this is **union**

33

Operations on Multisets

All RA operations need to be defined carefully on bags

- σ_C(R): preserve the number of occurrences
- Π_A(R): no duplicate elimination
- Cross-product, join: no duplicate elimination

This is important- relational engines work on multisets, not sets!

RA has Limitations !

- Cannot compute "transitive closure"

Name1	Name2	Relationship
Fred	Mary	Father
Mary	Joe	Cousin
Mary	Bill	Spouse
Nancy	Lou	Sister

- Find all direct and indirect relatives of Fred
- Cannot express in RA !!!
  - Need to write C program, use a graph engine, or modern SQL…