

Relational Algebra (RA)

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

Keep in mind: RA operates on sets!

- RDBMSs use *multisets*, however in relational algebra formalism we will consider sets!
- Also: we will consider the ***named perspective***, where every attribute must have a unique name
 - → attribute order does not matter...

Now on to the basic RA operators...

Selection (σ)

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

`Students(sid, sname, gpa)`

SQL:

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



RA:

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

Another example:

SSN	Name	Salary
1234545	John	20000
5423341	Smith	60000
4352342	Fred	50000

$\sigma_{\text{Salary} > 40000}$ (Employee)



SSN	Name	Salary
5423341	Smith	60000
4352342	Fred	50000

Projection (Π)

- Eliminates columns, then removes duplicates
- Notation: $\Pi_{A1, \dots, An}(R)$
- 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:

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

Another example:

SSN	Name	Salary
1234545	John	200000
5423341	John	600000
4352342	John	200000

$\Pi_{\text{Name,Salary}}(\text{Employee})$



Name	Salary
John	200000
John	600000

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?

Cross-Product (\times)

- Each tuple in R1 with each tuple in R2
- Notation: $R1 \times R2$
- Example:
 - Employee \times Departments
- 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 × 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 – *Rho*)

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

`Students(sid, sname, gpa)`

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 (\bowtie)

- 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

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

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 ⋈ People



sid	S.name	gpa	ssn	address
001	John	3.4	1234545	216 Rosse
002	Bob	1.3	5423341	217 Rosse

Natural Join

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

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  
      S.sname = P.sname;
```


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

How do we represent
this query in RA?

Division

- Notation: $r \div s$
- It has nothing to do with arithmetic division.
- Let r and s be relations on schemas R and S respectively where
 - $R = (A_1, \dots, A_m, B_1, \dots, B_n)$
 - $S = (B_1, \dots, B_n)$

The result of $r \div s$ is a relation on schema

$$R - S = (A_1, \dots, A_m)$$

$$r \div s = \{ t \mid t \in \Pi_{R-S}(r) \wedge \forall u \in s (tu \in r) \}$$

Where tu means the concatenation of tuples t and u to produce a single tuple

Division Operation - Example

■ Relations r, s

A	B
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
\in	6
\in	1
β	2

r

B
1
2

s

■ $r \div s$

A
α
β

Division Operation - Example

■ Relations r, s

<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>
α	a	α	a	1
α	a	γ	a	1
α	a	γ	b	1
β	a	γ	a	1
β	a	γ	b	3
γ	a	γ	a	1
γ	a	γ	b	1
γ	a	β	b	1

r

<i>D</i>	<i>E</i>
a	1
b	1

s

■ $r \div s$

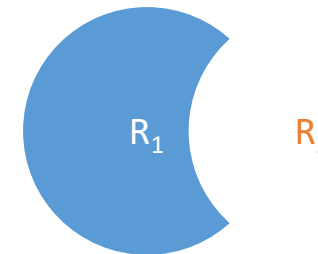
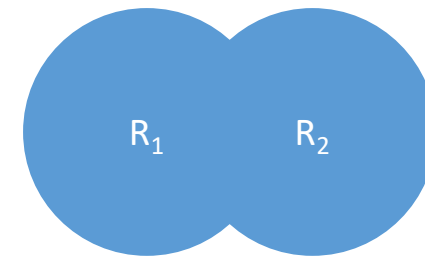
<i>A</i>	<i>B</i>	<i>C</i>
α	a	γ
γ	a	γ

Logical Equivalence of RA Plans

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

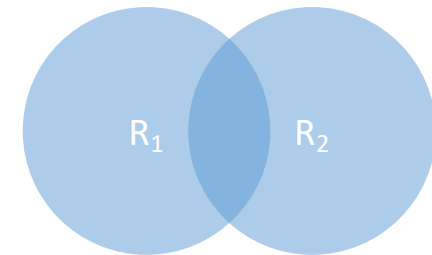
Union (\cup) and Difference ($-$)

- $R_1 \cup R_2$
- Example:
 - $\text{ActiveEmployees} \cup \text{RetiredEmployees}$
- $R_1 - R_2$
- Example:
 - $\text{AllEmployees} - \text{RetiredEmployees}$



What about Intersection (\cap) ?

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



Theta Join (\bowtie_{θ})

- 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  $\theta$ ;
```



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:

$S \bowtie_{sname=pname} P$