

CS 6400 A

Database Systems Concepts and Design

Lecture 5

09/03/25

Announcements

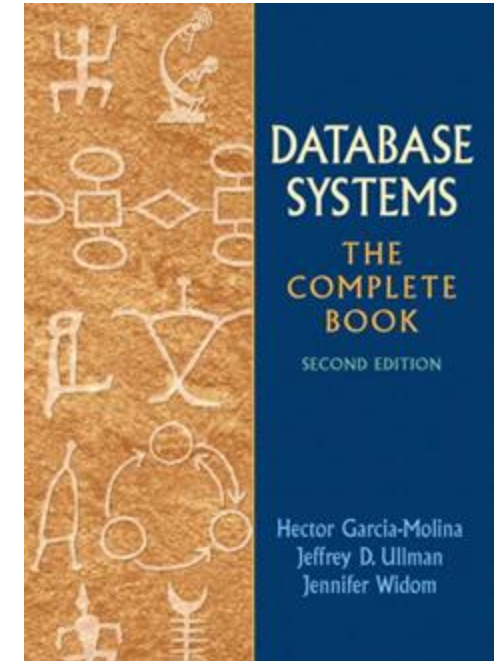
Project proposal released

- Due Oct 1
- Teams of 3-4
- Sign up on Canvas
 - Under the People tab ->Project Groups

Reading Materials

Database Systems: The Complete Book (2nd edition)

- Chapter 2.1: An Overview of Data Models
- Chapter 2.2: Basics of the Relation Model
- Chapter 2.4: An Algebraic Query Language



Acknowledgement: The following slides have been adapted from CS145 (Intro to Big Data Systems) taught by Peter Bailis.

Today's Class

1. Relational Algebra: Basic Operators
2. Relational Algebra Pt. II

Recall: Data model

A notation for describing data or information.

The description generally consists of three parts:

- Structure of the data
- Operations on the data
- Constraints on the data

1st Part of the Model: Structure of the data

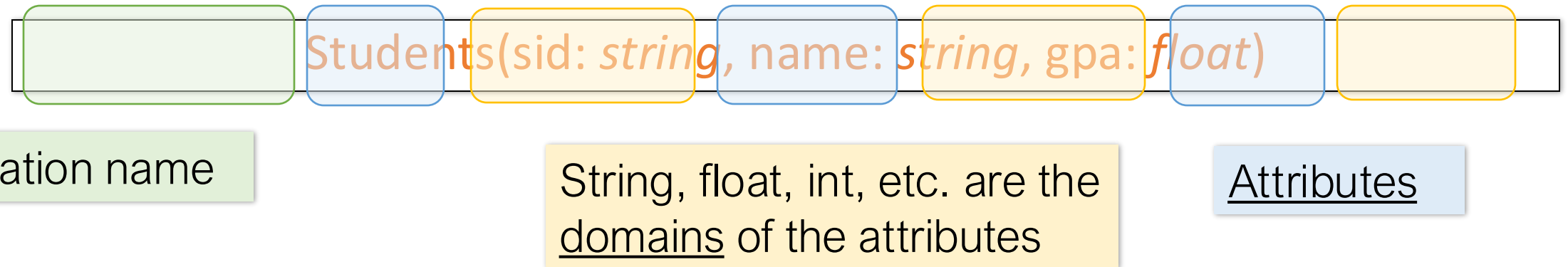
- Referred to as a “conceptual model” of the data
- Higher level than “physical data models” or data structures like arrays and lists

sid	name	gpa
001	Bob	3.2
002	Joe	2.8
003	Mary	3.8
004	Alice	3.5

Example: a relation consists of a schema, attributes, and tuples

The Relational Model: Schema

- Relational Schema:



The Relational Model: Data

An **attribute** (or **column**) is a typed data entry present in each tuple in the relation

Student

sid	name	gpa
001	Bob	3.2
002	Joe	2.8
003	Mary	3.8
004	Alice	3.5

The Relational Model: Data

Student

sid	name	gpa
001	Bob	3.2
002	Joe	2.8
003	Mary	3.8
004	Alice	3.5

A tuple or row (or *record*) is a single entry in the table having the attributes specified by the schema

The Relational Model: Data

Student

sid	name	gpa
001	Bob	3.2
002	Joe	2.8
003	Mary	3.8
004	Alice	3.5

Recall: In practice DBMSs relax the set requirement, and use multisets.

A relational instance is a *set* of tuples all conforming to the same *schema*

Relational schema vs relational instance

- A relational schema describes the data that is contained in a relational instance

Let $R(f_1:\text{Dom}_1, \dots, f_m:\text{Dom}_m)$ be a relational schema then, an instance of R is a subset of $\text{Dom}_1 \times \text{Dom}_2 \times \dots \times \text{Dom}_n$

In this way, a relational schema R is a total function from attribute names to types

A relational database

- A relational database schema is a set of relational schemata, one for each relation
- A relational database instance is a set of relational instances, one for each relation

Two conventions:

1. We call relational database instances as simply **databases**
2. We assume all instances are valid, i.e., satisfy the domain constraints

Example

- *Relation DB Schema*

- Students(sid: *string*, name: *string*, gpa: *float*)
- Courses(cid: *string*, cname: *string*, credits: *int*)
- Enrolled(sid: *string*, cid: *string*, grade: *string*)

Note that the schemas impose effective domain / type constraints, i.e. Gpa can't be "Apple"

Sid	Name	Gpa
101	Bob	3.2
123	Mary	3.8

Students

Relation Instances

sid	cid	Grade
123	564	A

Enrolled

cid	cname	credits
564	564-2	4
308	417	2

Courses

2nd Part of the Model: Operations on the data

Usually a limited set of operations that can be performed

- Queries (operations that retrieve information)
- Modifications (operations that change the database)

This is a strength, not a weakness

- Programmers can describe operations at a very high level
- The DBMS implements them efficiently
- Not easy to do when coding in C

The Relational Model: Operations

“Find names of all students
with GPA > 3.5”

```
SELECT S.name  
FROM Students S  
WHERE S.gpa > 3.5;
```

We specify how or where to get
the data - just what we want,
i.e., Querying is declarative

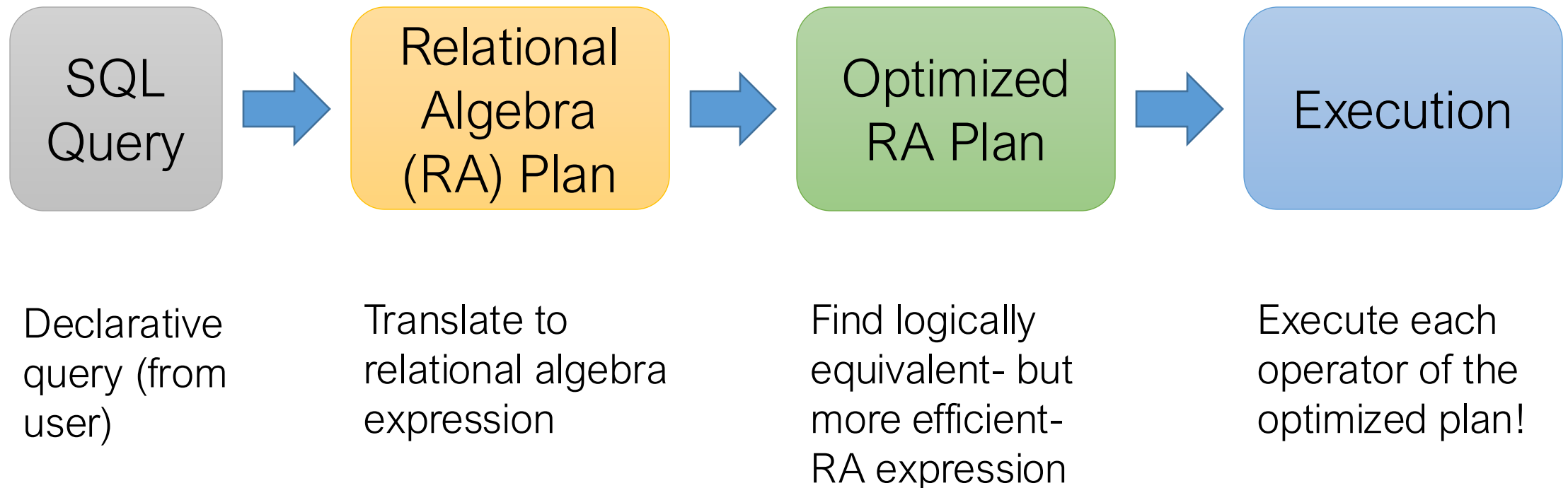
To make this happen, we need to
translate the declarative query
into a series of operators...

The operations normally
associated with the relational
model forms relational algebra

1. Relational Algebra

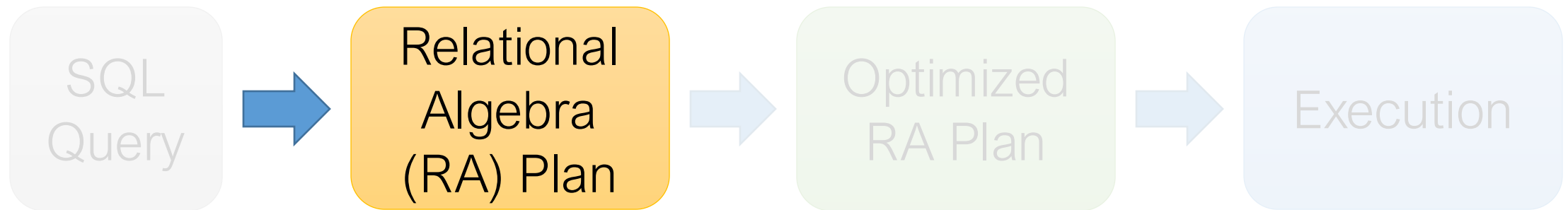
The big picture: RDBMS Architecture

How does a SQL engine work ?



The big picture: RDBMS Architecture

How does a SQL engine work ?



Relational Algebra allows us to translate declarative (SQL) queries into precise and optimizable expressions!

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 first!

- Derived or auxiliary operators:

- Intersection, complement
- Joins (natural, equi-join, theta join, semi-join)
- Renaming: ρ
- Grouping: γ

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

Note: 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...

1. Selection (σ)

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

Students(sid,sname,gpa)

SQL:

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



RA:

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

Another example:

SSN	Name	Salary
1234545	John	200000
5423341	Smith	600000
4352342	Fred	500000

$\sigma_{\text{Salary} > 400000}$ (Employee)



SSN	Name	Salary
5423341	Smith	600000
4352342	Fred	500000

2. Projection (Π)

- Eliminates columns, then removes duplicates
- Notation: $\Pi_{A_1, \dots, A_n}(R)$
- 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?

3. Cross-Product (\times)

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

- 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:

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

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)

- $R_1 \bowtie R_2$: 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:
 - $R_1 \bowtie R_2 = \Pi_{A \cup B}(\sigma_{C=D}(\rho_{C \rightarrow D}(R_1) \times R_2))$
 - 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	216 Rosse

Example: Converting SFW Query -> RA

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

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


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

How do we represent
this query in RA?

In class exercise

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

2. Relational Algebra Pt. II

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

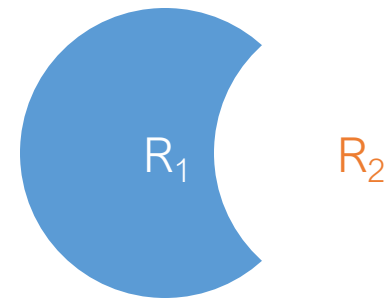
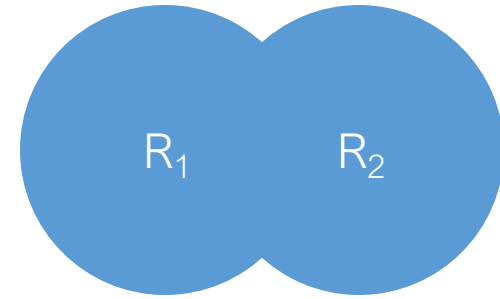
- Derived or auxiliary operators:

- Intersection, complement
- Joins (natural, equi-join, theta join, semi-join)
- Renaming: ρ
- Grouping: γ

And also at some of these derived operators

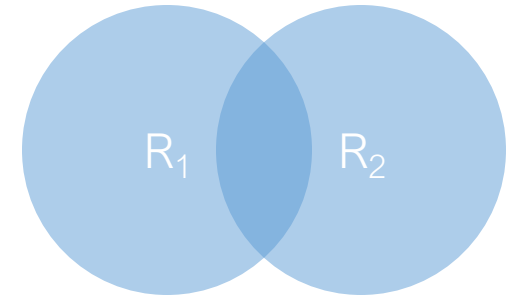
Union (\cup) and 2. Difference ($-$)

- $R_1 \cup R_2$
- 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`



In class exercise

Suppose relations R and S have n and m tuples, respectively

What are the minimum / maximum number of tuples of the following expressions?

1. $R \cup S$

2. $R \bowtie S$

3. $\pi_L(R) - S$, for some list of attributes L

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

How to write $R1 \bowtie_{\theta} R2$ using basic RA operators?

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

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

SQL:

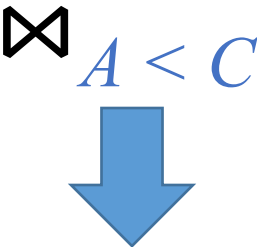
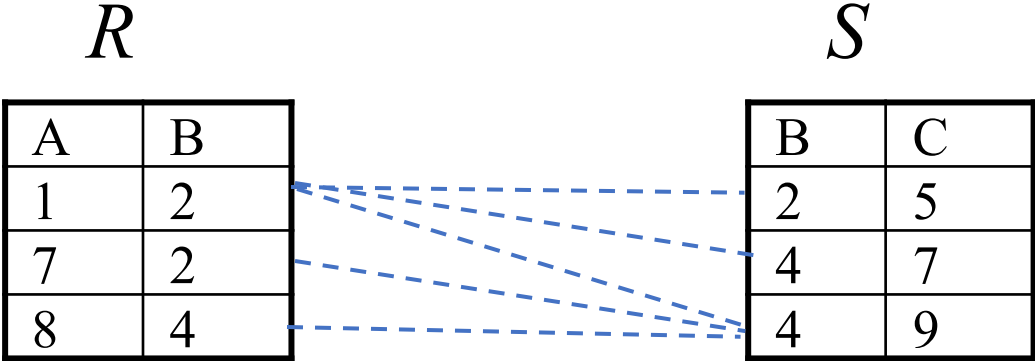
```
SELECT *  
FROM  
  Students,People  
WHERE  $\theta$ ;
```



RA:

Students \bowtie_{θ} *People*

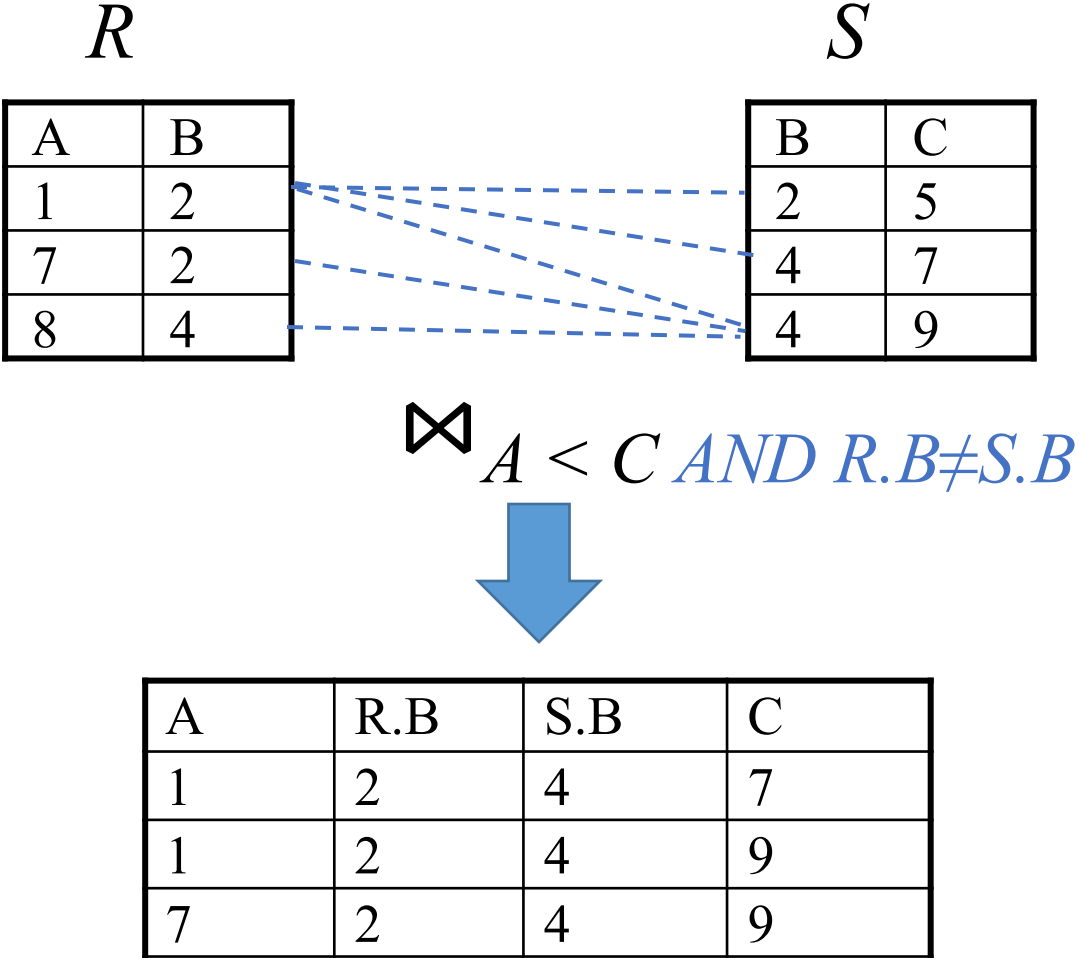
Another example:



$R \bowtie_{A < C} S$

A	R.B	S.B	C
1	2	2	5
1	2	4	7
1	2	4	9
7	2	4	9
8	4	4	9

Another example:



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 (\bowtie)

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

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

SQL:

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

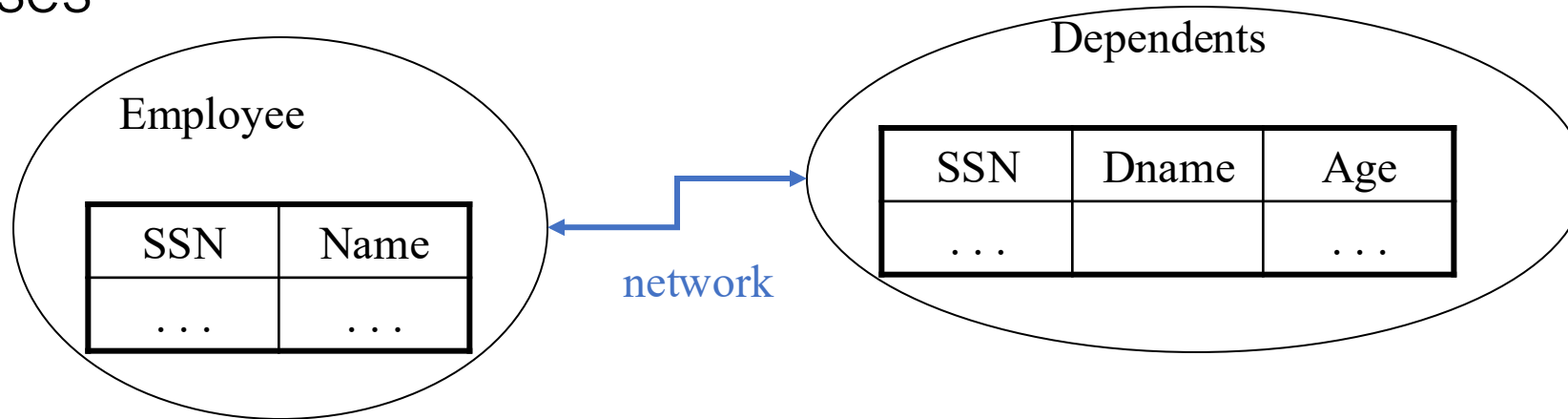


RA:

Students \bowtie People

Semijoins in Distributed Databases

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



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

Send less data
to reduce
network
bandwidth!

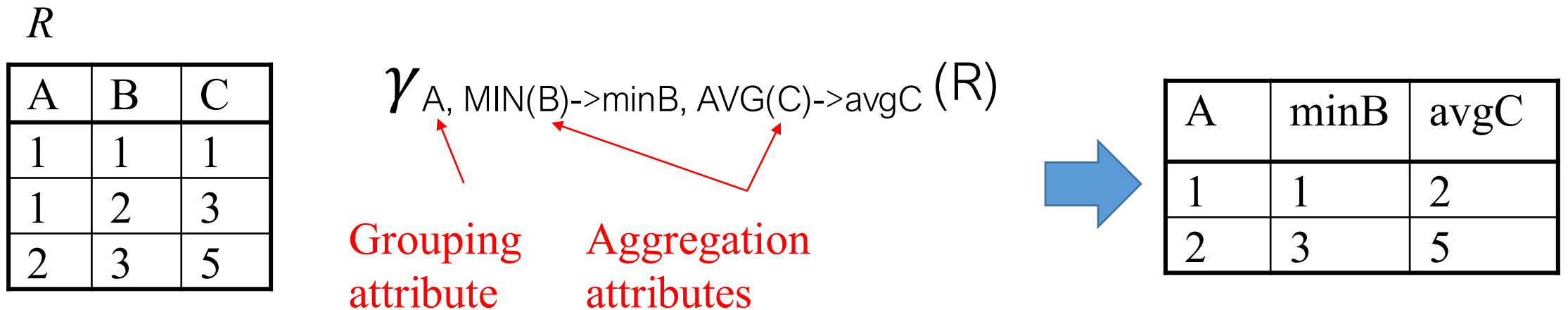
$$R = \text{Employee} \bowtie T$$

Two blue arrows point from the expression $R = \text{Employee} \bowtie T$ to the definitions of T and the final result:

$$T = \Pi_{\text{SSN}} (\sigma_{\text{age}>71} (\text{Dependents}))$$
$$\text{Answer} = R \bowtie \text{Dependents}$$

Grouping (γ)

- The grouping operator γ consists of
 - Grouping attributes: attributes to group by
 - Aggregation attributes: attributes to which aggregation operations are applied
 - SUM, AVG, MIN, MAX, COUNT



Combining operations to form queries

- RA expressions can be arbitrarily complicated by applying operations to other results
- Multiple RA expressions may be equivalent

$$\Pi_{A,D}(\sigma_{A < 10}(T \bowtie (R \bowtie S)))$$

R(A,B) S(B,C) T(C,D)



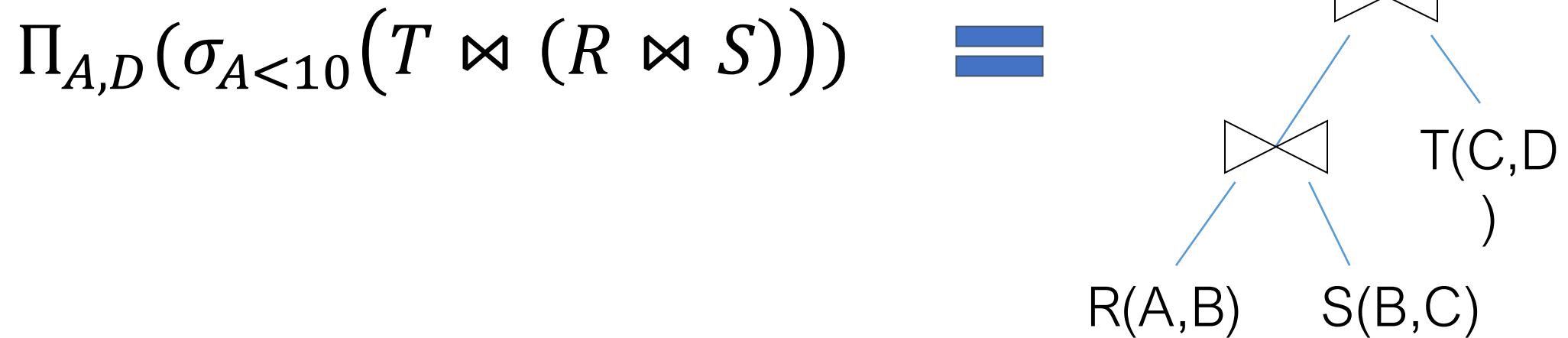
$$\Pi_{A,D}(T \bowtie \Pi_{A,C}(\sigma_{A < 10}(R) \bowtie S))$$

Which version is more efficient?

Logical optimization (will cover later): Find equivalent RA expressions that are more efficient

Expression tree

RA expressions can be represented as expression trees



Bottom-up tree traversal = order of operation execution!

RA Expressions Can Get Complex!

