CMPT 321
Fall 2017

Relational algebra

Lecture 02.01

By *Marina Barsky*

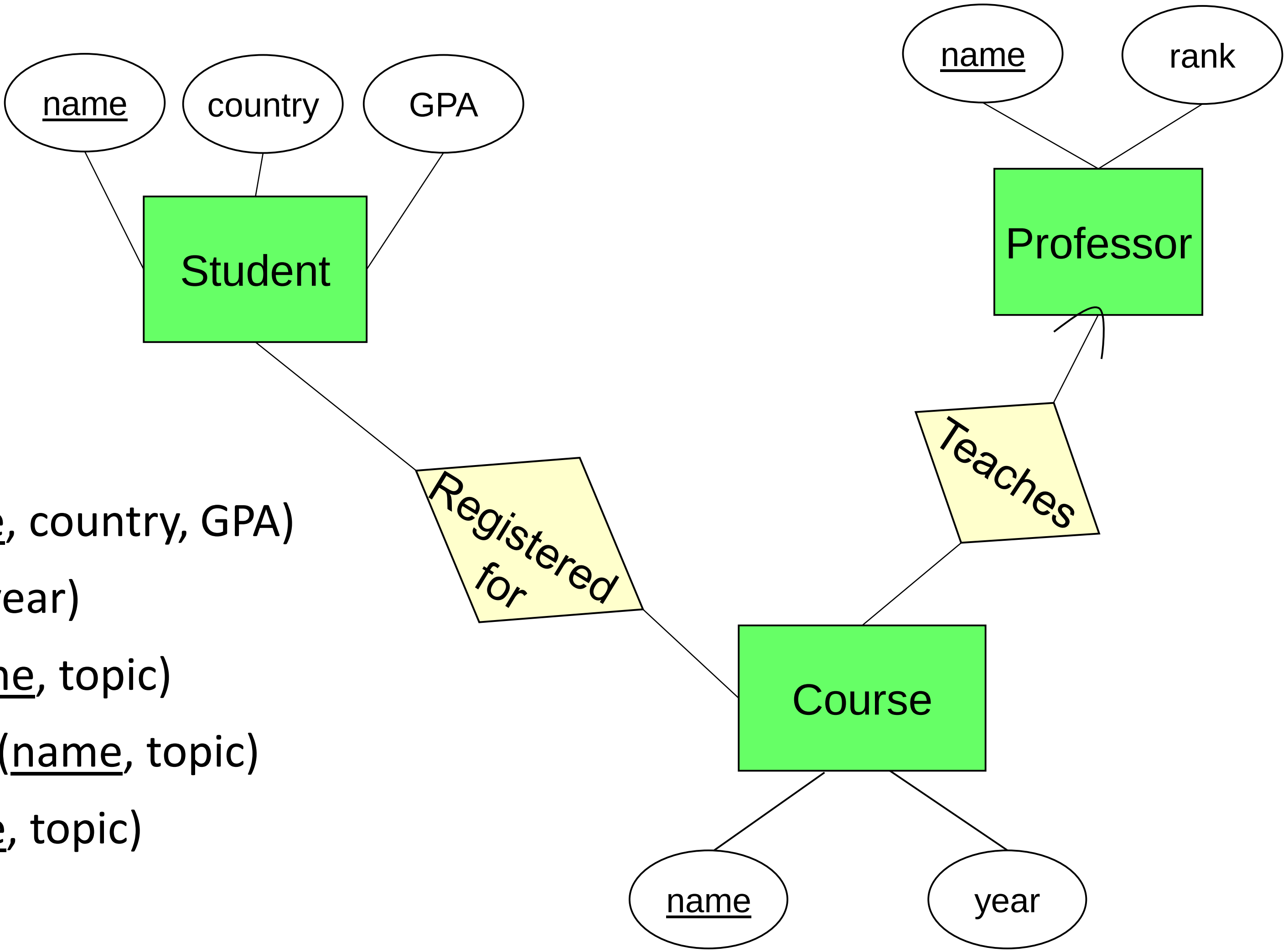
Relations: what are they?

- *Relations* are records of related facts or properties for each entity in the entity set
- How the facts are related is defined through the list of attributes
- The facts themselves are represented as tuples of values – one value for each attribute

Facts required to be different – relation is a SET

- There are no two completely identical tuples in a given relations
- Each relation is a **set** of tuples – no duplicates

Consider an example



Student (<u>name</u>, country, GPA)

Couse (<u>topic</u>, year)

Professor (<u>name</u>, topic)

RegisteredFor (<u>name</u>, topic)

Teaches (<u>name</u>, topic)

Sample instances for each relation

Student		
Name	Country	GPA
Bob	Canada	3
John	Britain	3
Tom	Canada	3.5
Maria	Mexico	4

Course	
Topic	Year
Algorithms	2
Python	2
Databases	3
GUI	3

RegisteredFor	
Name	Topic
Bob	Algorithms
John	Algorithms
Tom	Algorithms
Bob	Python
Tom	Python
Bob	Databases
John	Databases
Maria	Databases
John	GUI
Maria	GUI

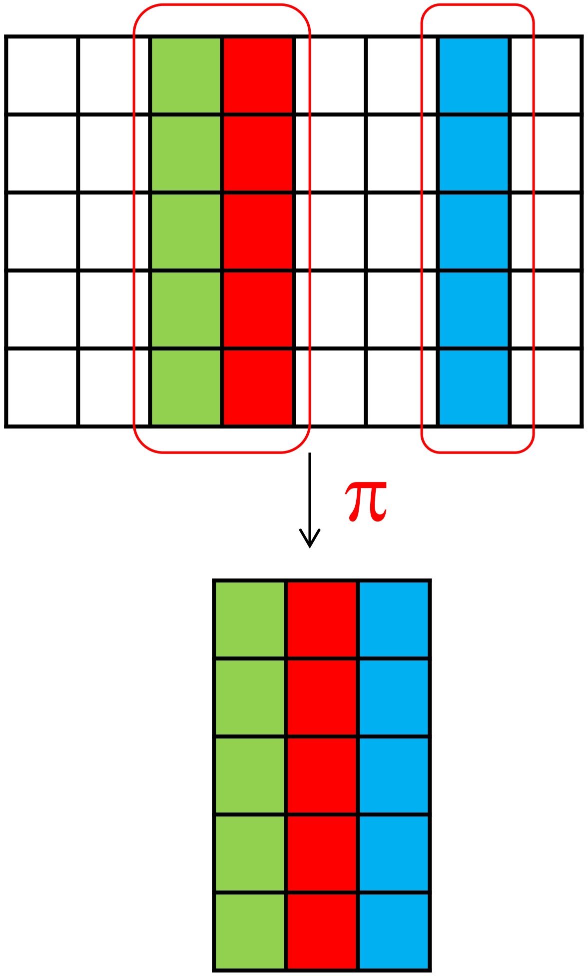
Professor	
Name	Rank
Dr. Monk	Professor
Dr. Pooh	Associate Professor
Dr. Patel	Assistant Professor

Teaches	
Name	Topic
Dr. Monk	Algorithms
Dr. Pooh	Python
Dr. Patel	Databases
Dr. Patel	GUI

Core operators of *relational* algebra

Slice operators: Projection

Produces from relation **R** a new relation that has only the A_1, …, A_n columns of **R**.



$$S=\pi_{\text{attribute list}}(R)$$

Projection: example

Query: list names of students

Student			
SIN	Name	GPA	Country
111	Bob	3	Canada
222	John	3	Britain
333	Tom	3.5	Canada
444	Maria	4	Mexico

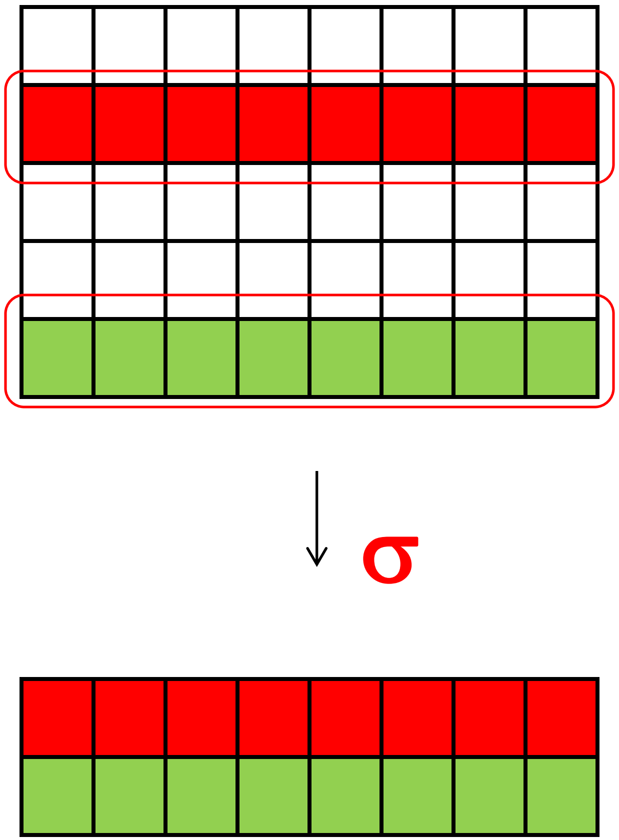
S
Name
Bob
John
Tom
Maria

$$S = \pi_{Name}(Student)$$

Slice operators: Selection

Produces a new relation with those tuples of **R** which satisfy condition **C**.

$$S=\sigma_{condition}(R)$$



Selection example.

Query: list students with GPA >3

Student		
Name	GPA	Country
Bob	3	Canada
John	3	Britain
Tom	3.5	Canada
Maria	4	Mexico

S		
Name	GPA	Country
Tom	3.5	Canada
Maria	4	Mexico

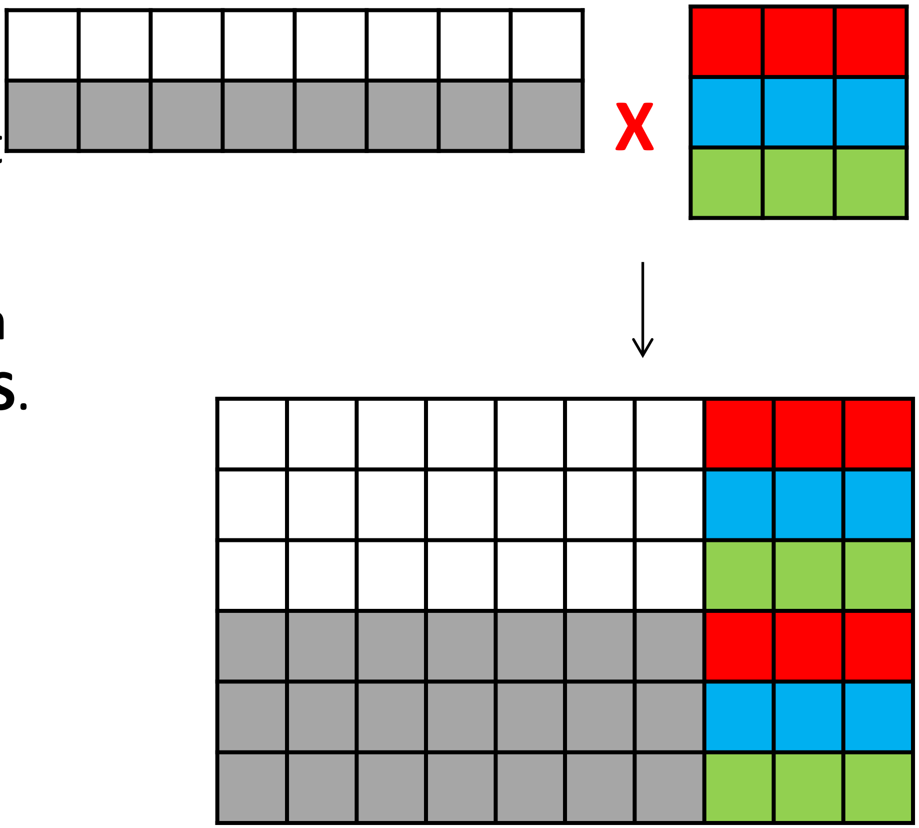
$S = \sigma_{gpa>3}(\text{Student})$

Join operation: Cartesian product (Cross-product)

1. Set of tuples ***rs*** that are formed by choosing the first part (***r***) to be any tuple of **R** and the second part (***s***) to be any tuple of **S**.

2.Schema for the resulting relation is the union of schemas for **R** and **S**.

3.If **R** and **S** happen to have some attributes in common, then prefix those attributes by the relation name.



T=R x **S**

Cartesian product example

T=Course x Professor

Course	
Topic	Year
Algorithms	2
Python	2
Databases	3
GUI	3

Professor	
Name	Rank
Dr. Monk	Professor
Dr. Pooh	Associate Professor
Dr. Patel	Assistant Professor

Cartesian product output

Algorithms	2
Python	2
Databases	3
GUI	3

Dr. Monk	Dr. Pooh	Dr. Patel
Professor	Associate Professor	Assistant Professor

Topic	Y	Name	Rank
Algorithms	2	Dr. Monk	Professor
Algorithms	2	Dr. Pooh	Assoc. Professor
Algorithms	2	Dr. Patel	Assist. Professor
Python	2	Dr. Monk	Professor
Python	2	Dr. Pooh	Assoc. Professor
Python	2	Dr. Patel	Assist. Professor
Databases	3	Dr. Monk	Professor
Databases	3	Dr. Pooh	Assoc. Professor
Databases	3	Dr. Patel	Assist. Professor
GUI	3	Dr. Monk	Professor
GUI	3	Dr. Pooh	Assoc. Professor
GUI	3	Dr. Patel	Assist. Professor

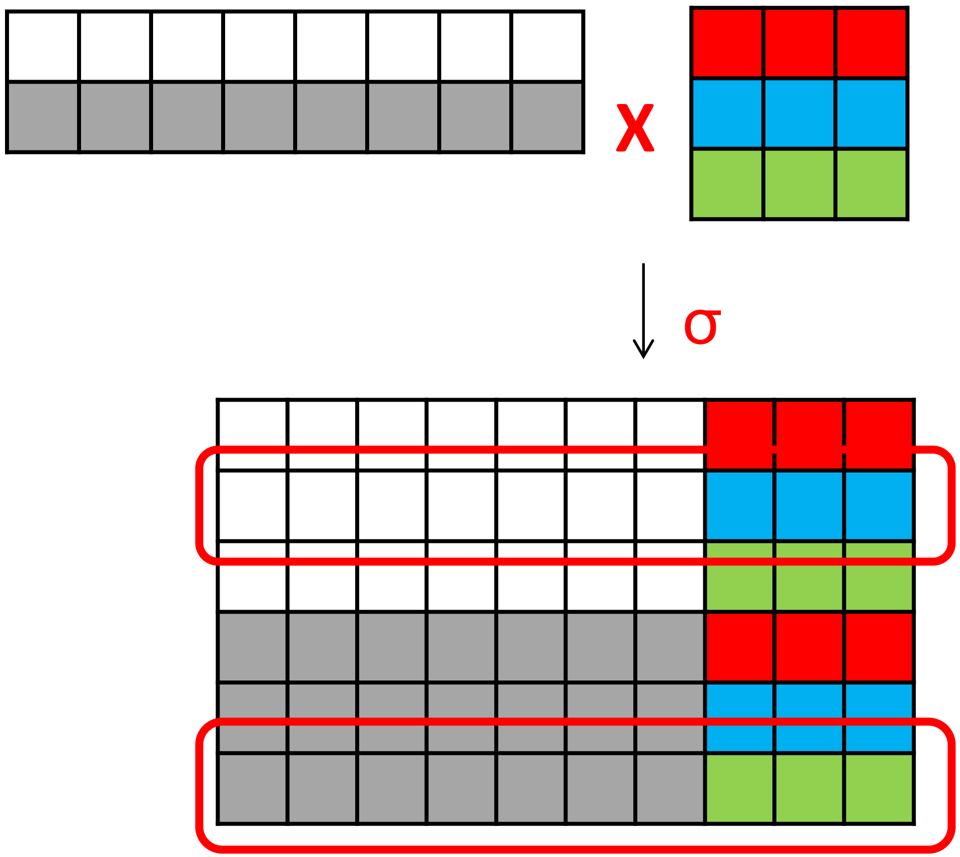
Combining Cross-product with selection

1. The result is constructed as follows:

    a) Take the Cartesian product of **R** and **S**.

    b) Select from the product only those tuples that satisfy the condition **C**.

2. Schema for the result is the union of the schema of **R** and **S**, with **"R"** or **"S"** prefix as necessary.



$$\mathbf{T=\sigma_{condition}\ (R \times S)}$$

Example.

Query: Dr. Monk wonders whether he has to teach a multi-cultural group of students

Student

Name	Country	GPA
Bob	Canada	3
John	Britain	3
Tom	Canada	3.5
Maria	Mexico	4

Teaches

Name	Topic
Dr. Monk	Algorithms
Dr. Pooh	Python
Dr. Patel	Databases
Dr. Patel	GUI

RegisteredFor

Name	Topic
Bob	Algorithms
John	Algorithms
Tom	Algorithms
Bob	Python
Tom	Python
Bob	Databases
John	Databases
Maria	Databases
John	GUI
Maria	GUI

Multi-cultural class

Student		
Name	Country	GPA
Bob	Canada	3
John	Britain	3
Tom	Canada	3.5
Maria	Mexico	4

AlgoList	
Name	Topic
Bob	Algorithms
John	Algorithms
Tom	Algorithms

AlgoList $= \sigma_{Topic=Algorithms}$ (RegisteredFor)

Multi-cultural class

Student		
Name	Country	GPA
Bob	Canada	3
John	Britain	3
Tom	Canada	3.5
Maria	Mexico	4

AlgoList	
Name	Topic
Bob	Algorithms
John	Algorithms
Tom	Algorithms

ClassInfo		
Name	Country	GPA
Bob	Canada	3
John	Britain	3
Tom	Canada	3.5

AlgoList $=\sigma_{\text{Topic=Algorithms}}$ (RegisteredFor)

ClassInfo= $\sigma_{\text{Student.name=AlgoList.name}}$ AlgoList x Student

Multi-cultural class

Student		
Name	Country	GPA
Bob	Canada	3
John	Britain	3
Tom	Canada	3.5
Maria	Mexico	4

AlgoList	
Name	Topic
Bob	Algorithms
John	Algorithms
Tom	Algorithms

ClassInfo		
Name	Country	GPA
Bob	Canada	3
John	Britain	3
Tom	Canada	3.5

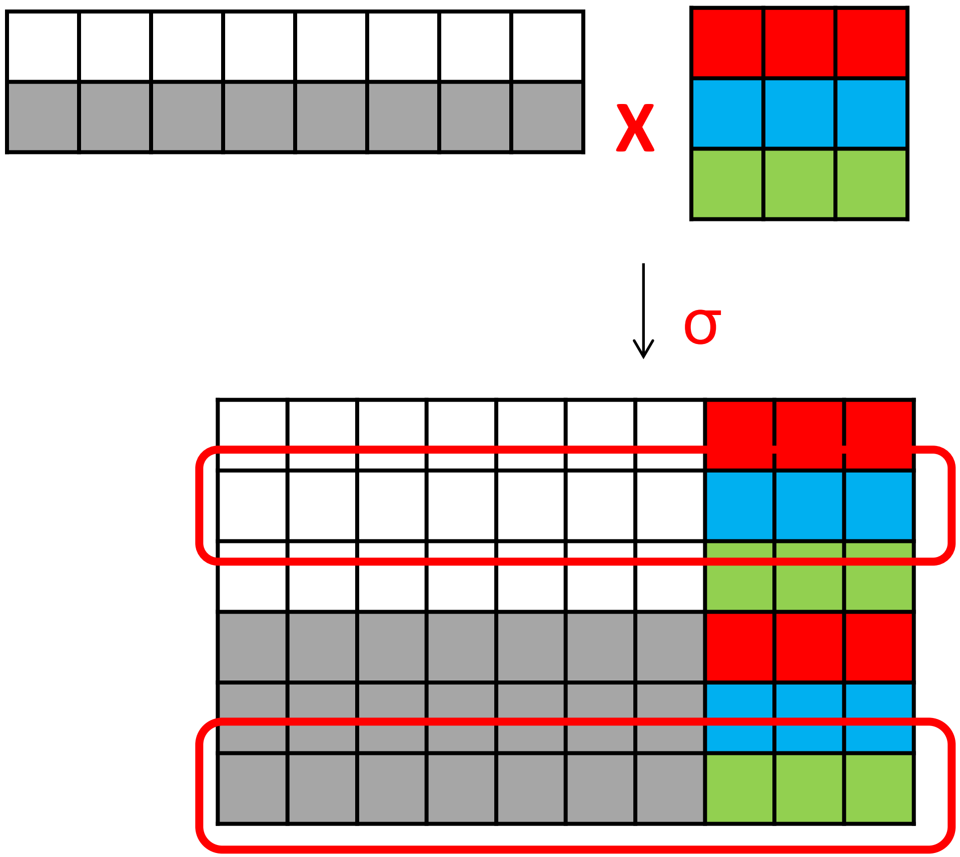
Countries
Country
Canada
Britain

AlgoList $=\sigma_{\text{Topic=Algorithms}}$ (RegisteredFor)

ClassInfo= $\sigma_{\text{Student.name=AlgoList.name}}$ AlgoList x Student

Countries=π_{country} (ClassInfo)

Cross-product with selection



$T=\sigma_{condition}\ (R \times S)$

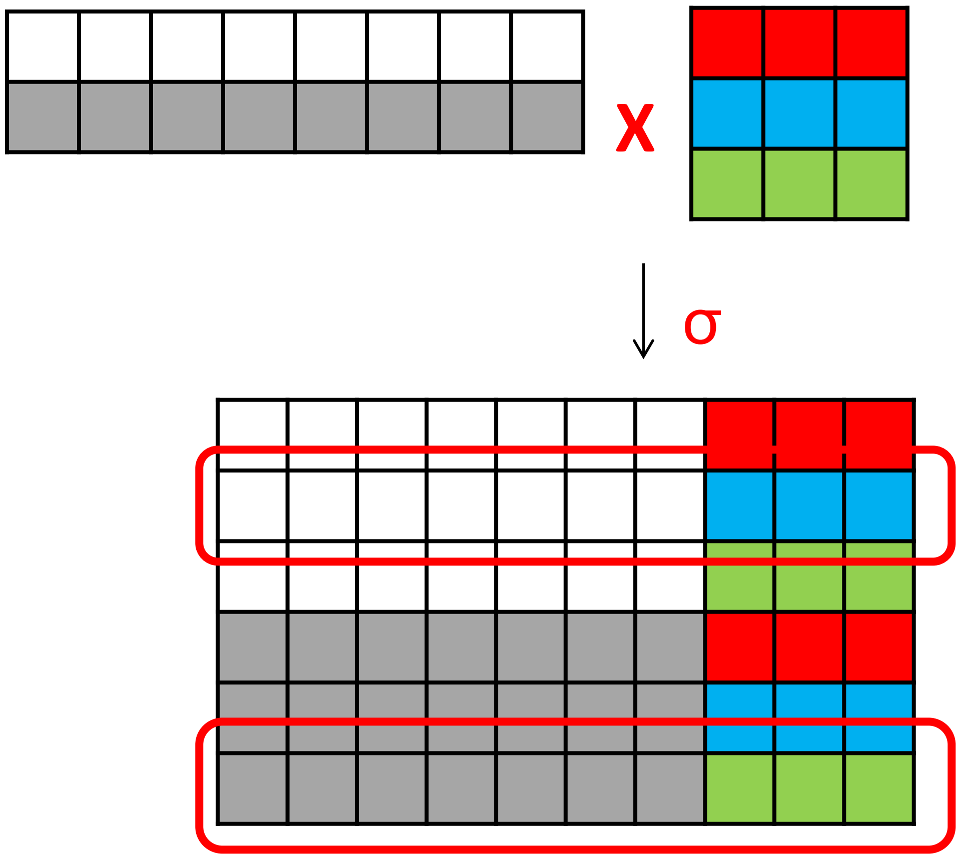
Shortcut: Theta-join

1. The result of this operation is constructed as follows:

    a) Take the Cartesian product of **R** and **S**.

    b) Select from the product only those tuples that satisfy the condition **C**.

2. Schema for the result is the union of the schema of **R** and **S,** with **"R"** or **"S"** prefix as necessary.



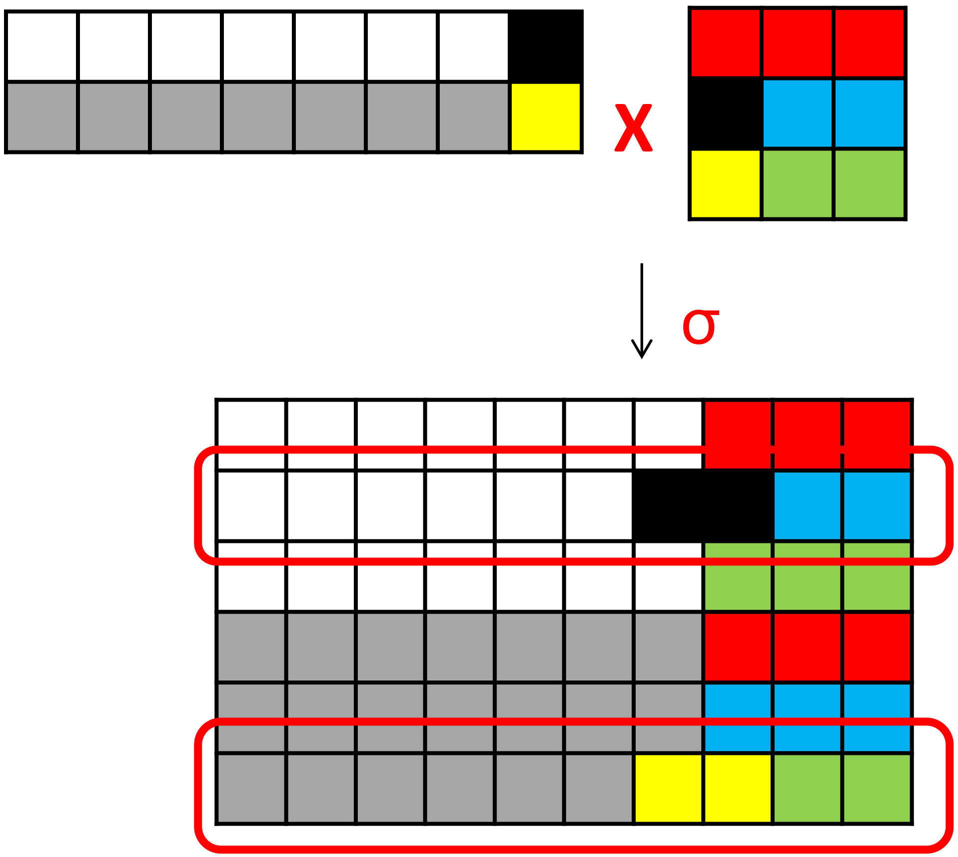
$T = R \bowtie_{condition} S$

Shortcut for

$T = \sigma_{condition} (R \times S)$

Subtype of theta-join: Equijoin

1.Equijoin is a subset of theta-joins where the join condition is equality



T= R $\bowtie_{R.A = S.B}$ S

Shortcut for

T=$\sigma_{R.A = S.B}$ (R x S)

Special case of equijoin: Natural Join

$R \bowtie S$

Let $\mathbf{A_1, A_2, \ldots, A_n}$ be the attributes in both the schema of **R** and the schema of **S**.

Then a tuple ***r*** from **R** and a tuple ***s*** from **S** are successfully paired if and only if ***r*** and ***s*** agree on each of their common attributes $\mathbf{A_1, A_2, \ldots, A_n}$.

Still the same meaning as:

$T = \sigma_{R.A = S.A} (R \times S)$,

but common attributes are not duplicated as in Cartesian Product

Set Operations on Relations

$R \cup S$, the **union** of **R** and **S**, is the set of tuples that are in **R** or **S** or both.

$R - S$, the **difference** of **R** and **S**, is the set of tuples that are in **R** but not in **S**.

Note that **$R - S$** is different from **$S - R$**.

$R \cap S$, the **intersection** of **R** and **S**, is the set of tuples that are in both **R** and **S**.

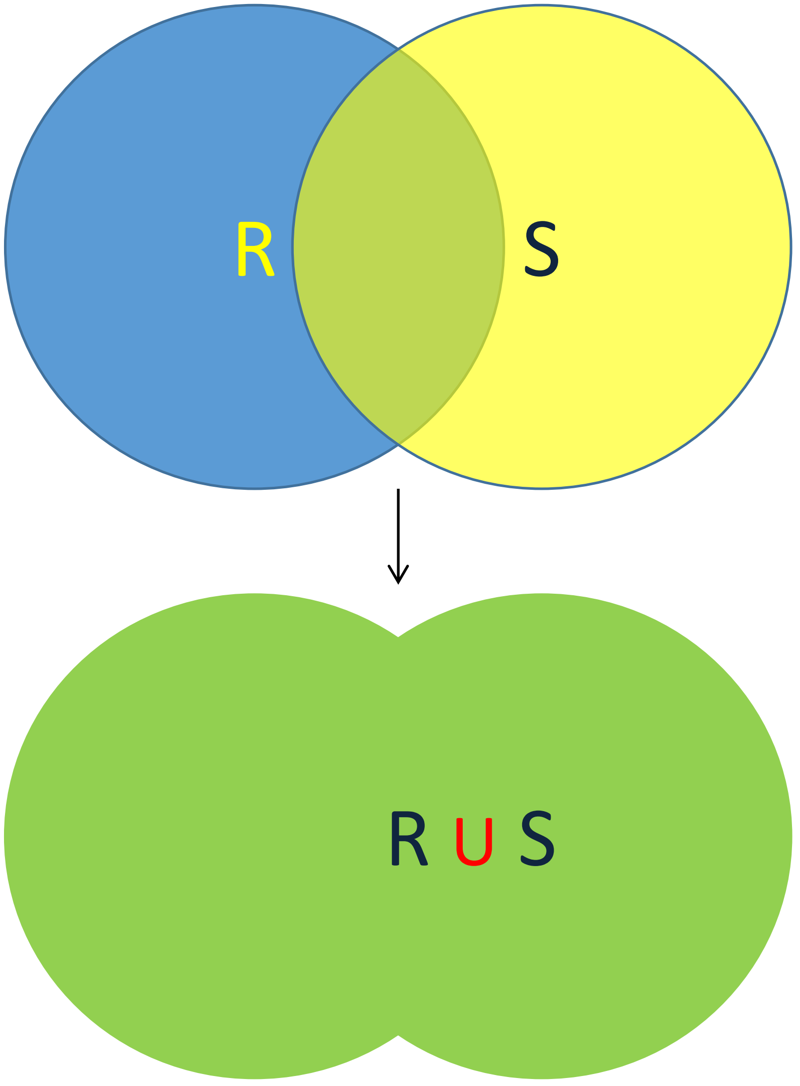
Condition for set operators

Set operators can operate only on two union-compatible relations

Two relations are **union-compatible** if they have the same number of attributes and each attribute must be from the same domain

Union

$T = R \cup S$



Union example.

Query: list names of all people in the department

Student		
Name	Country	GPA
Bob	Canada	3
John	Britain	3
Tom	Canada	3.5
Maria	Mexico	4

Professor	
Name	Rank
Dr. Monk	Professor
Dr. Pooh	Associate Professor
Dr. Patel	Assistant Professor

Can we do ?

T=Student $\cup$ Professor

Union example.

Query: list names of all people in the department

Student
Name
Bob
John
Tom
Maria

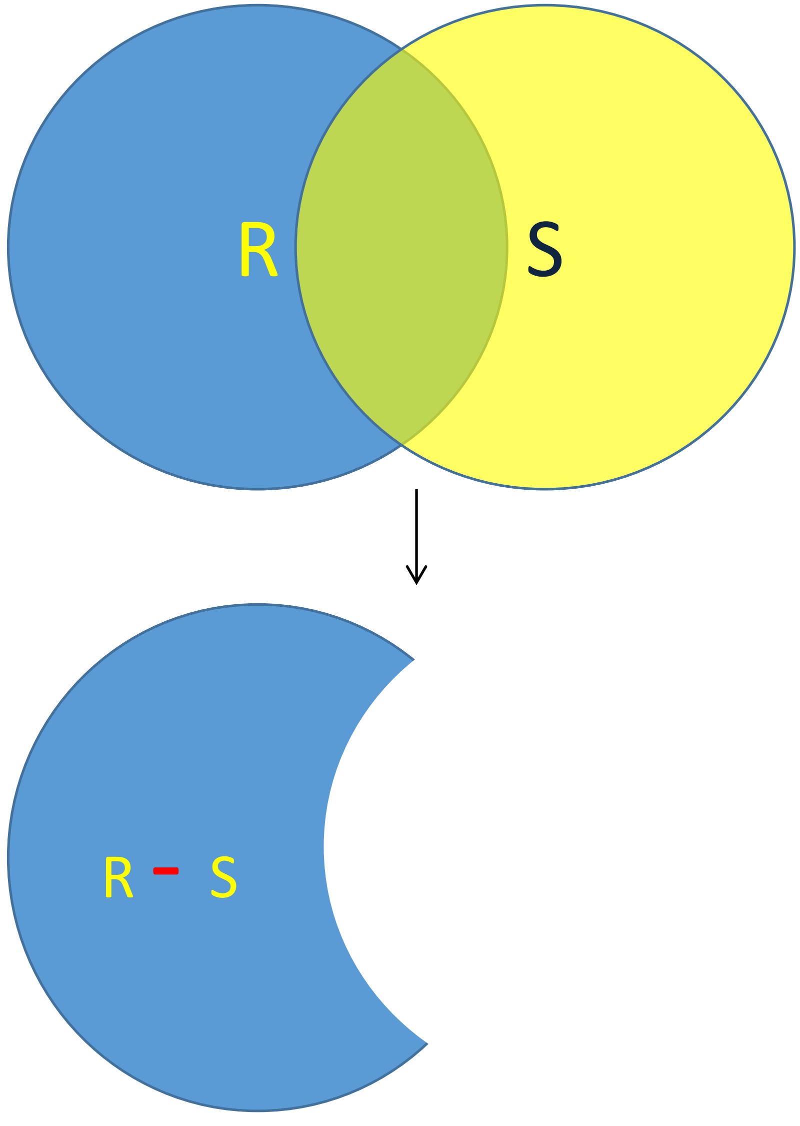
Professor
Name
Dr. Monk
Dr. Pooh
Dr. Patel

$$T = \pi_{name}(Student) \cup \pi_{name}(Professor)$$

Note: if attributes in 2 operands have different names, the names of the left relation are used in the union (PostgreSQL)

Difference

R – S



Difference example.

Query: Who is registered in the Database course but not in the Algorithms?

RegisteredFor	
Name	**Topic**
Bob	Algorithms
John	Algorithms
Tom	Algorithms
Bob	Python
Tom	Python
Bob	Databases
John	Databases
Maria	Databases
John	GUI
Maria	GUI

First do some selections:

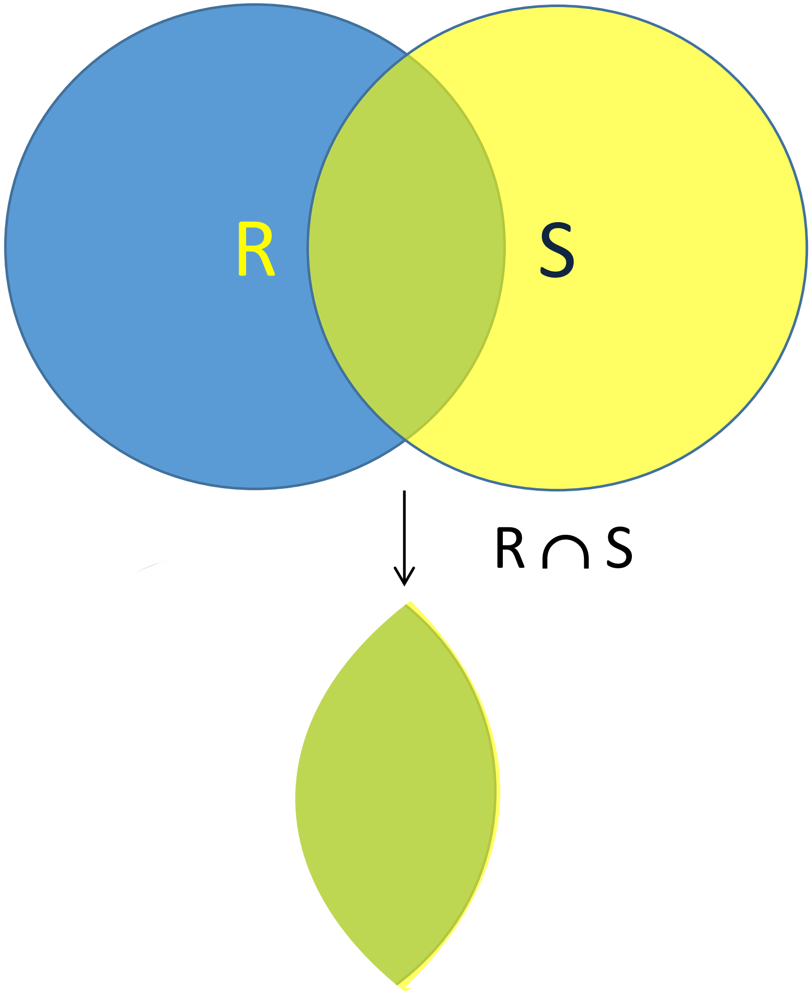
$A = \sigma_{\text{topic=algorithms}}(\text{RegisteredFor})$

$D = \sigma_{\text{topic=databases}}(\text{RegisteredFor})$

Then take $D - A$

Intersection

T=R ∩ S



Intersection example.

Query: Which courses are taught at both Universities?

Alright University

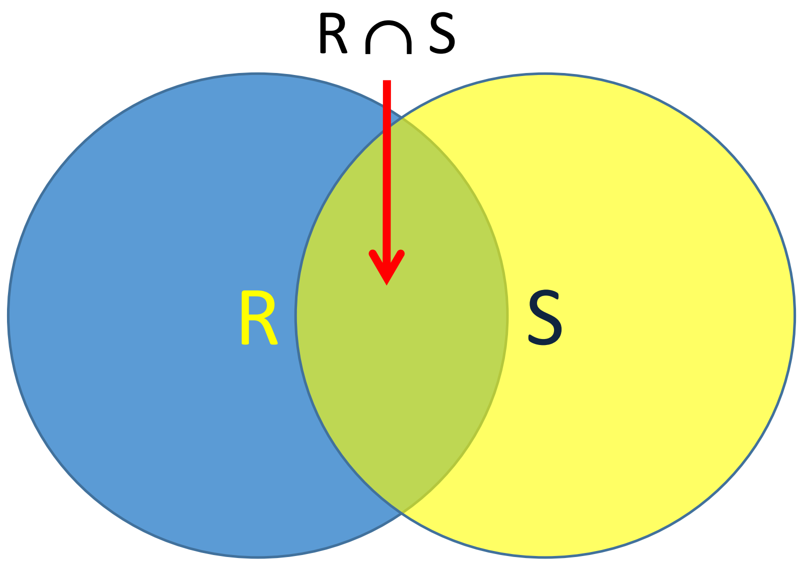
Course
Topic
Algorithms
Python
Databases
GUI

EvenBetter University

Course
Topic
Algorithms
Java
Databases
Networks
Human-Computer Interaction

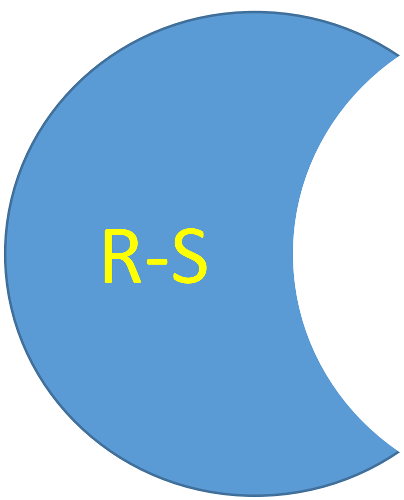
$$T = \pi_{\text{topic}}(\text{A.course}) \cap \pi_{\text{topic}}(\text{B.course})$$

Intersection is a shortcut for R – (R – S)

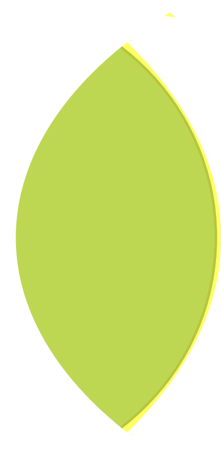


R ∩ S can be derived using 2 difference operators

R – (R – S)



R - S (are in R but not in S)



R - (R-S)

Renaming Operator

$$\rho_{S(A1,A2,\ldots,An)}\ (R)$$

1. Resulting relation has exactly the same tuples as **R**, but the name of the relation is **S**.
2. Moreover, the attributes of the result relation **S** can be re-named A_1, A_2, …, A_n, in order from the left.
3. If not all attributes are renamed, can specify renamed attributes:

$$\rho_{S,\ a \rightarrow a1,\ b \rightarrow b1}\ (R)$$

Renaming: example

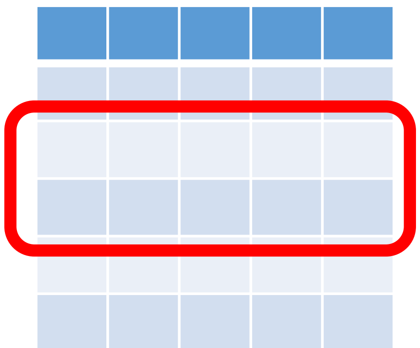
T (uid1, uid2)

A → B
B → A
B → C
A → C
C → B

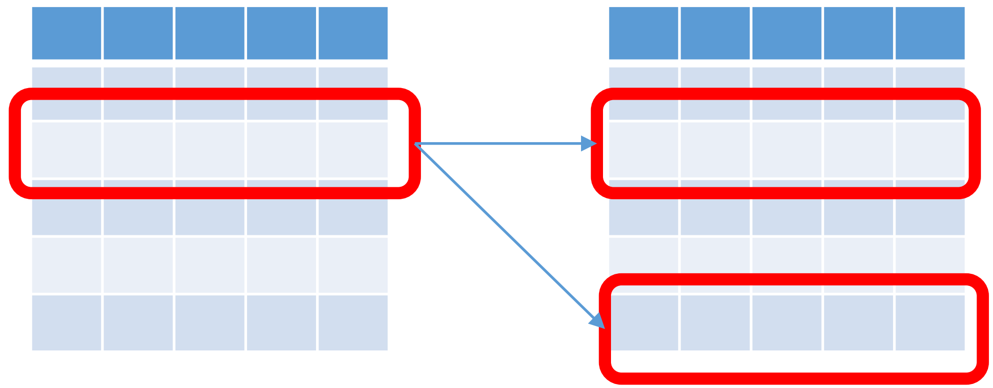
- Find all true friends in twitter dataset
- By renaming T we created two identical relations R and S, and we now extract all tuples where for each pair X → Y in R there is a pair Y → X in S

$$\pi_{R.uid1,\ R.uid2}\ \sigma_{R.uid1=S.uid2\ AND\ R.uid2\ =\ S.uid1}(\rho_R\ (T)\ x\ \rho_S\ (T))$$

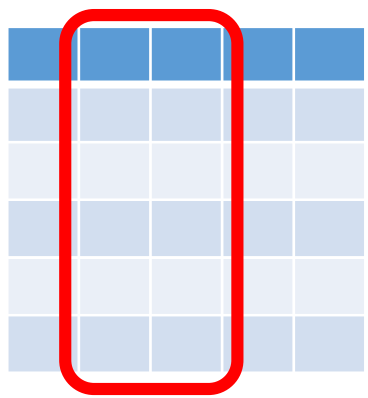
Core operators of relational algebra



Selection **σ**



Cross-product **x**



Projection **π**

Union **U**

Difference **–**

Renaming **ρ**

Core operators – sufficient to express any query in relational model

Edgar "Ted" Codd, a mathematician at IBM in 1970, proved that any query can be expressed using these core operators:
σ, π, x, U, –, ρ

> A Relational Model of Data for Large Shared Data Banks". *Communications of the ACM* **13** (6): 377–387

The Relational model is **precise**, **implementable**, and we can operate on it (combine, optimize)

Relational algebra: closure

In regular algebra the result of every operator is another number, and we can compose complex expressions using basic operators +,-,x,/:

$$a^2 - b^2 = (a-b)x(a+b)$$

The same applies to relational algebra: any RA operator returns a relation, so we can compose complex queries by operating on these intermediate results:

$$\pi_{name,gpa}(\sigma_{gpa>3.5}(\text{Student}))$$

Are these logically equivalent?

$$\sigma_{gpa>3.5}(\pi_{name,gpa}(\text{Student}))$$

Relational algebra equivalences

- Commutative: $R \bowtie S = S \bowtie R$
- Associative: $(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$
- Splitting: $\sigma_{C \cap D}(R) = \sigma_C(\sigma_D(R))$
- Pushing selections: $\sigma_C(R \bowtie_D S) = \sigma_C(R) \bowtie_D (S)$, if condition C applies only to R
- …

Example of a valid RA transformation

- Consider ***R*(*A*,*B*)** and ***S*(*B*,*C*)** and the expression below:

  $\boldsymbol{\sigma}_{A=1 \cap B<C}(R \bowtie S)$

1. Splitting **AND** $\boldsymbol{\sigma}_{A=1}(\boldsymbol{\sigma}_{B<C}(R \bowtie S))$
2. Push **σ** to *S* $\boldsymbol{\sigma}_{A=1}(R \bowtie \boldsymbol{\sigma}_{B<C}(S))$
3. Push **σ** to *R* $\boldsymbol{\sigma}_{A=1}(R) \bowtie \boldsymbol{\sigma}_{B<C}(S)$

Intermediate variables

As in traditional algebra,

$x^2 + 2x + 1 = 0$

$D = 4 – 4 = 0$

$x = -2 \underline{+} \sqrt{D} = -2$

we can use *temporary variables* to store the results of intermediate queries. These temporary variables hold results of what is called a ***subquery***

$T_1 = \sigma_{A=1}(R)$

$T_2 = \sigma_{B < C}(S)$

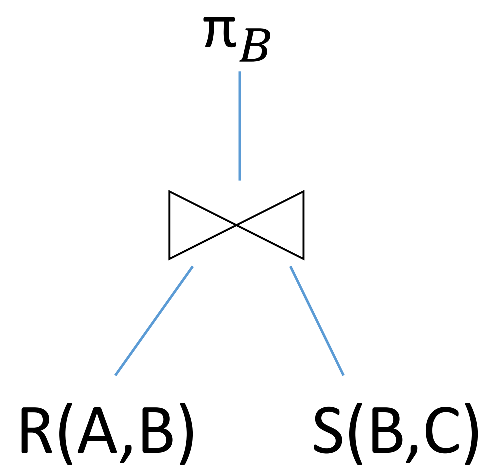
$\text{Result} = T_1 \bowtie T_2$

We can visualize an RA expression as a tree

Linear notation

$$\pi_B(R(A,B) \bowtie S(B,C))$$

Tree notation



Bottom-up tree traversal = order of operation execution!

Why do we care about relational algebra?

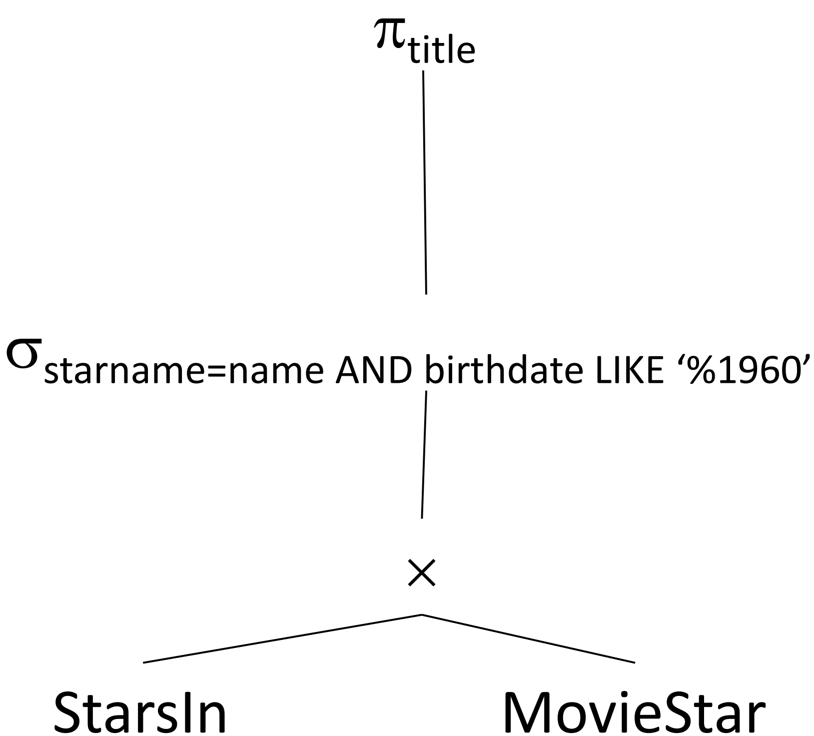
Why not learn just SQL?

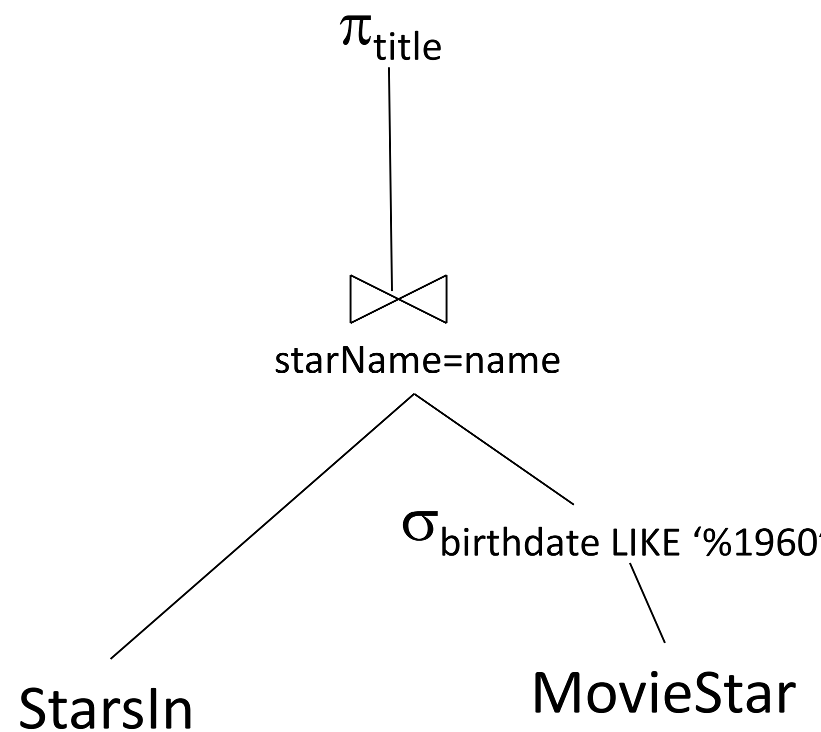
SQL is a query language that implements Relational Algebra

Why not learn how to solve quadratic equations looking only at a java implementation?

```
double discriminant = Math.pow(b,2) - 4*a*c;
double x1 = (-b + Math.sqrt(discriminant))/(2*a);
double x2 = (-b - Math.sqrt(discriminant))/(2*a);
double i=Math.sqrt(-1);
double x3 = (-b + (Math.sqrt(discriminant))*i)/(2*a);
double x4 = (-b + (Math.sqrt(discriminatn))*i)/(2*a);
if (discriminat > 0 ){
    System.out.println("there are two solutions:" +x1+"and"+x2);
}
```

RA is a basis for logical query optimization





Which query is more efficient?

Extended operators of Relational Algebra

can be derived from core operators

Outer join

Motivation

- Suppose we join $R \bowtie S$.
- A tuple of R which doesn't join with any tuple of S is said to be *dangling*.
  - Similarly for a tuple of S.
  - **Problem**: We loose dangling tuples.

Outerjoin

- Preserves dangling tuples by padding them with a special **NULL** symbol in the result.

Types of outer join

- $R ⟗_C S$ – This is the **full outerjoin**: Pad dangling tuples from both tables.

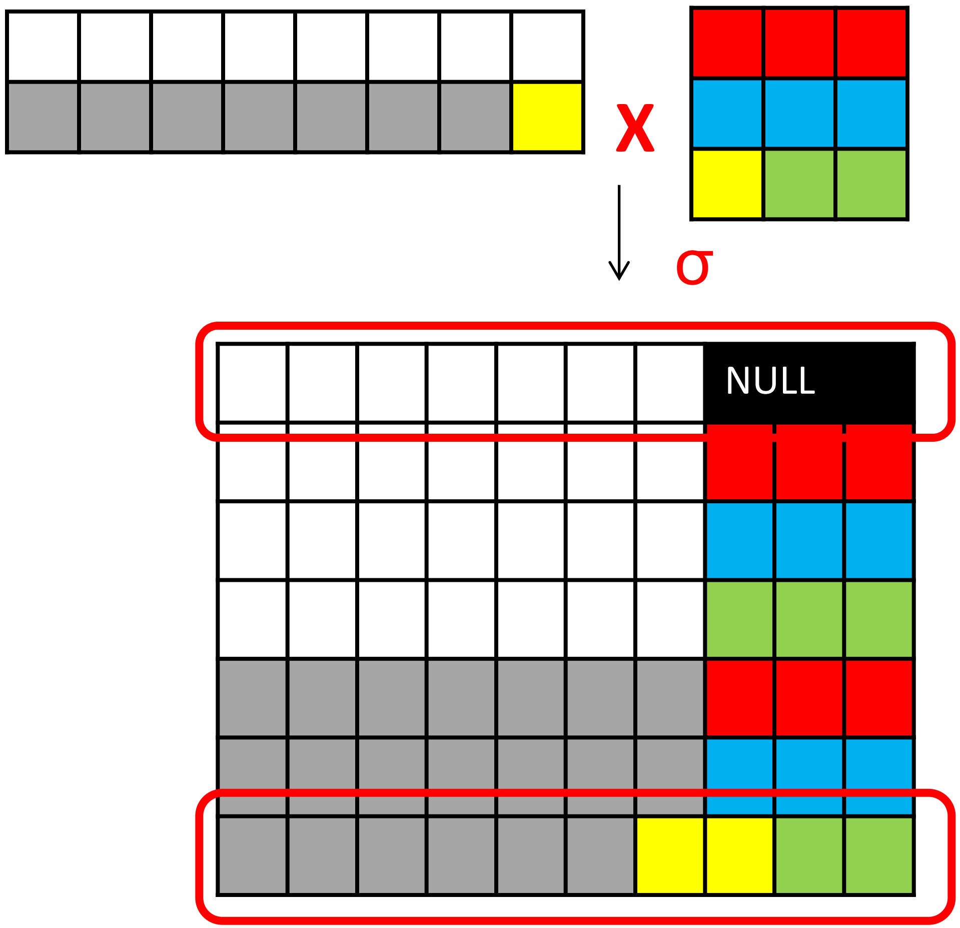
- $R ⟕_C S$ – **left outerjoin**: Only pad dangling tuples from the left table.

- $R ⟖_C S$ – **right outerjoin**: Only pad dangling tuples from the right table.

Left outer join

1. For each tuple in R, include all tuples in S which satisfy join condition, but include also tuples of R that do not have matches in S

2. For this case, pair tuples of R with NULL



$T = R ⟕_{\text{condition}} S$

Outer join: example

Anonymous patient P

age	zip	disease
54	99999	heart
20	44444	flue
33	66666	lung

Anonymous occupation O

age	zip	job
54	99999	lawyer
20	44444	cashier

T= P ⟕ O

age	zip	disease	job
54	99999	heart	lawyer
20	44444	flue	cashier
33	66666	lung	**NULL**

Estimating size of resulting relations

Size estimation example 1

Given relation R with N tuples and relation S with M tuples, what is the maximum and minimum size of the output to the following queries:

$\sigma_c (R)$

- Min: 0 (no tuples satisfy the condition)
- Max: N

$\pi_A (R)$

- Min: 1
- Max: N

What if A is a key?

- Min: N
- Max: N

Size estimation example 2

Given relation R (A,B) with N tuples and relation S(B,C) with M tuples, tell what is the maximum and minimum size of the output to the following queries

R x S

- Min: NM
- Max: NM

R ⋈ S

- Min: 0 (no tuples to join)
- Max: NM (all tuples of S join with all tuples of R on their common attribute – equal values of B in both relations)

Sample test question

If I have a relation R with 100 tuples and a relation S with exactly 1 tuple, how many tuples will be in the result of

$R \ltimes S$?

A. At least 100, but could be more
B. Could be any number between 0 and 100 inclusive
C. 0
D. 1
E. 100