

Relational Algebra

CS 4750
Database Systems

[A. Silberschatz, H. F. Korth, S. Sudarshan, Database System Concepts, Ch.2.6]

[C.M. Ricardo, S.D. Urban, "Databases Illuminated, Ch.4.5"]

[H. Garcia-Molina, J.D. Ullman, J. Widom, Database Systems: The Complete Book, Ch.2]

Revisit – Basic SQL

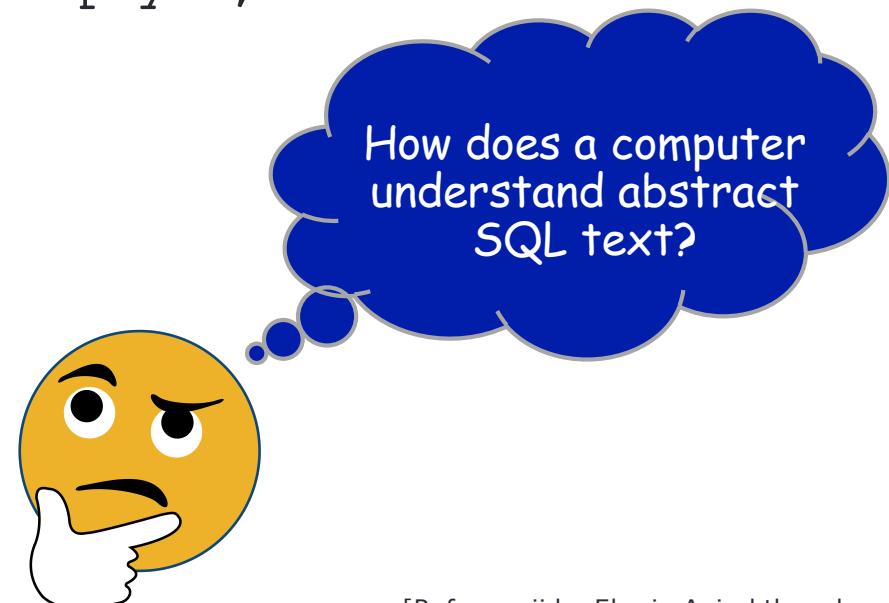
Student_lecture

S_id	Address	Course	Teaching_assistant
1234	57 Hockanum Blvd	Database Systems	Minnie
2345	1400 E. Bellows	Database Systems	Humpty
3456	900 S. Detroit	Cloud Computing	Dumpty
1234	57 Hockanum Blvd	Web Programming Lang.	Mickey
5678	2131 Forest Lake Ln.	Software Analysis	Minnie

```
SELECT S_id as ID, Course as "Course Name"  
FROM Student_lecture  
WHERE Teaching_assistant <> "Dumpty" ;
```



ID	Course Name
1234	Database Systems
1234	Web Programming Lang.
2345	Database Systems
5678	Software Analysis



[Ref: emoji by Ekarin Apirakthanakorn]

Database Internals

High level programming language to machine code

```
public class computeAvg
{
    public static double computeAverage(double[] arr)
    {
        if (arr == null)
            throw new NullPointerException();

        double sum = 0.0;

        for (int i=0; i<=arr.length-1; i++)
        {
            System.out.println("reach in loop");
            sum += arr[i];
        }

        System.out.println("before computing avg");
        double average = sum / arr.length;
        return average;
    }
}
```

Code
(High-level languages)

Low-level languages

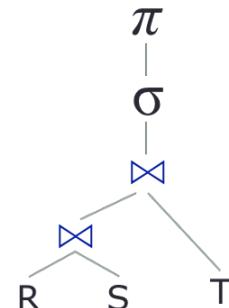
```
1001010101100
0010111101010
0010101000000
1010010100
```

Computers
("How" to execute)

```
SELECT ...
FROM ...
[ WHERE ... ]
[ GROUP BY ... ]
[ HAVING ... ]
[ ORDER BY ... ]
```

SQL
("What" data to get)

RDBMS



RA
("How" to get the data)

```
1001010101100
0010111101010
0010101000000
1010010100
```

Computers
("How" to execute)

Relational Algebra (RA)

- A data model is not just structure
 - Needs a way to query the data
 - Needs a way to modify the data

- Ways to construct new relations from given relations

SQL is a declarative language

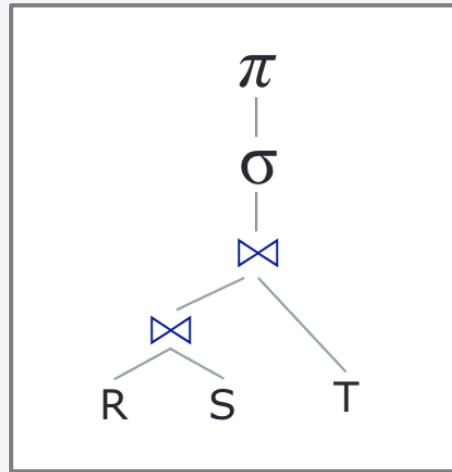
- Relational algebra – “Procedural Query Language”

- Ways to build expressions by applying operators to atomic operands and/or other expressions of the algebra

Atomic operands = Variables that stand for relations or constants

- When a DBMS processes queries, a SQL query is translated into an RA tree
- After some optimizations, the RA tree is converted into instructions

Why? (Query Designers' Perspective)



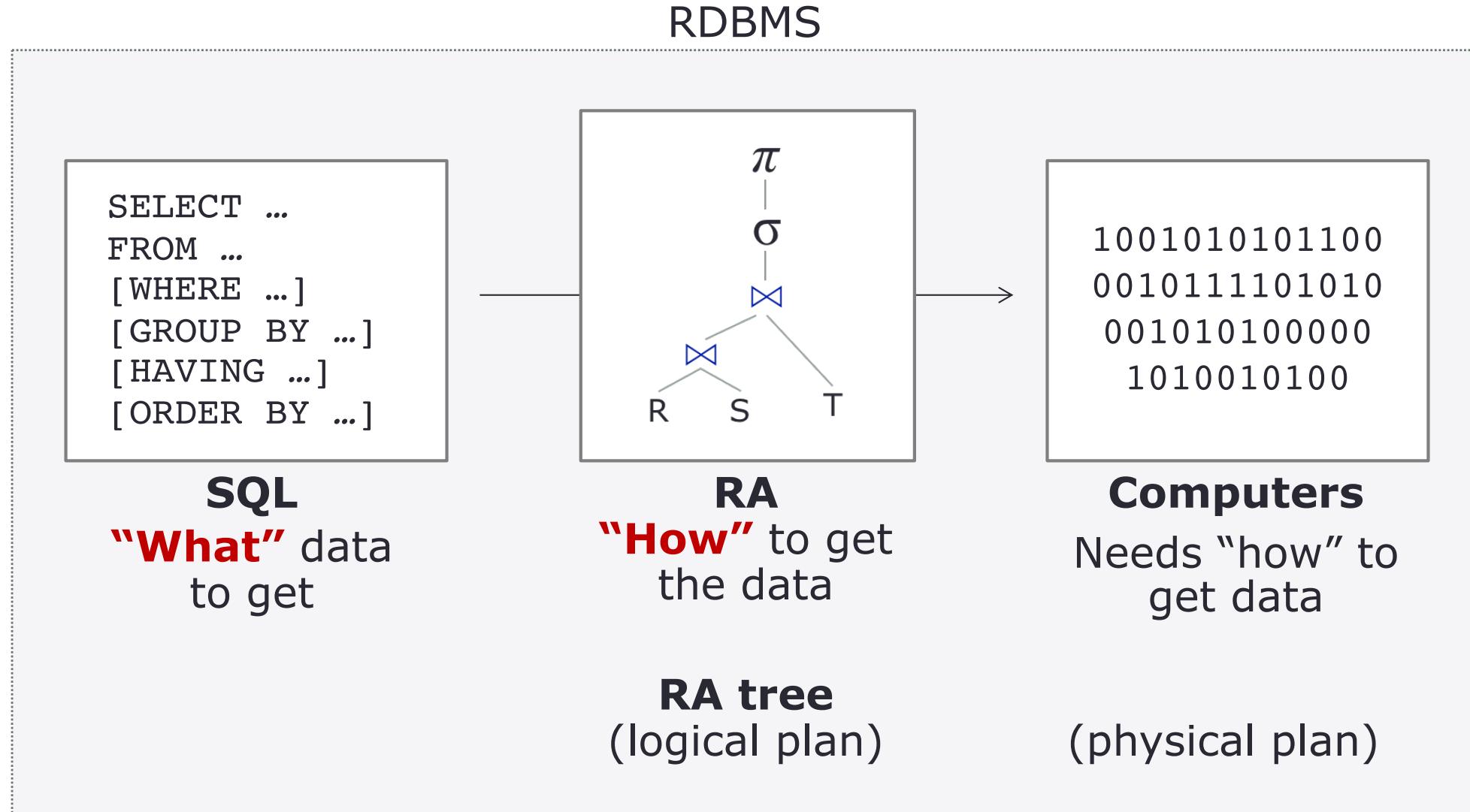
RA
“How” to get
the data



```
SELECT ...
FROM ...
[ WHERE ... ]
[ GROUP BY ... ]
[ HAVING ... ]
[ ORDER BY ... ]
```

SQL
“What” to get
from the data

Why? (DBMS and Query Processors)

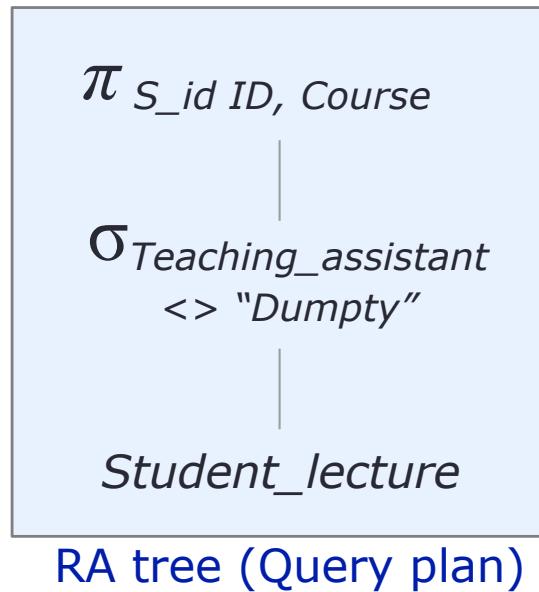


Example: Database Internals

```
SELECT S_id as ID, Course  
FROM Student_lecture  
WHERE Teaching_assistant <> "Dumpty";
```

Query output

Data source



- Define the semantics of a query

- Every operator takes 1 (unary) or 2 (binary) relations as inputs.

- Every operator outputs a relation

```
for each row in Student_lecture:  
  if (Teaching_assistant <> "Dumpty")  
    output (S_id as ID, Course as Course Name)
```

RA Operator Categories

Unary operations

Take one relation, return a new relation

Selection $\sigma_p(r)$

Find tuples that satisfy a given condition

Project $\pi_{A_1, A_2, \dots, A_m}(r)$

Slice a relation, return a new relation with certain attributes

Rename $\rho_{x(A_1, A_2, \dots, A_n)}(E)$

Rename the result of expression E to x; rename resultant attributes to A_1, A_2, \dots

Addition Expressions

Operations that remove parts of a relation

Return a new relation

Union $r \cup s$ -- "or"

Combine tuples from 2 relations
[require: same number of attributes; compatible domains]
[result: same attributes]

Intersection $r \cap s$ -- "and"

Combine tuples from 2 relations
[require: same number of attributes; compatible domains]
[result: same attributes]

Set difference $r - s$

Find tuples that are in one relation but are not in another
[require: same number of attributes; compatible domains]
[result: same attributes]

Cartesian product $r \times s$

Combine 2 relations, all combination
[result: combined attributes]

Natural join $r \bowtie s$

Select tuples that satisfy the matching conditions from combined relations
[result: combined attributes]

Division $r \div s$

Similar to $AB \div B$
Find "A" for all "B"
[require: there exists B's attributes in A]
[result: A schema]

Operations that change the schema of a relation

RA Operator Categories

Unary operations

Take one relation, return a new relation

Selection $\sigma_p(r)$

Find tuples that satisfy a given condition

Project $\pi_{A_1, A_2, \dots, A_m}(r)$

Slice a relation, return a new relation with certain attributes

Rename $\rho_{x(A_1, A_2, \dots, A_n)}(E)$

Rename the result of expression E to x; rename resultant attributes to A_1, A_2, \dots

Additional Expressiveness Assignment
Aggregate functions

Binary operations

Take two relation, return a new relation

Union $r \cup s$ -- "or"

Combine tuples from 2 relations
[require: same number of attributes; compatible domains]
[result: same attributes]

Intersection $r \cap s$ -- "and"

Combine tuples from 2 relations
[require: same number of attributes; compatible domains]
[result: same attributes]

Set difference $r - s$

Find tuples that are in one relation but are not in another
[require: same number of attributes; compatible domains]
[result: same attributes]

Operations that combine the tuples of two relations

Cartesian product $r \times s$

Combine 2 relations, all combination
[result: combined attributes]

Natural join $r \bowtie s$

Select tuples that satisfy the matching conditions from combined relations
[result: combined attributes]

Division $r \div s$

Similar to $AB \div B$
Find "A" for all "B"
[require: there exists B's attributes in A]
[result: A schema]

Extended operation

Selection (σ)

- **Unary** operation – take one operand
- Return all tuples that satisfy a condition (**filter tuples**)
- Conditions can be $=, <, \leq, >, \geq, <>$ and combined with AND, OR, NOT

$\sigma_C(R)$ where C is a set of conditions that involve the attribute(s) of R

Where and Having have the same selection operator σ

Example: Selection (σ)

Find all tuples in `Student_lecture` relation that have a “Database Systems” course and “Minnie” as TA

`Student_lecture`

<code>S_id</code>	<code>Address</code>	<code>Course</code>	<code>TA</code>
1234	57 Hockanum Blvd	Database Systems	Minnie
2345	1400 E. Bellows	Database Systems	Humpty
3456	900 S. Detroit	Cloud Computing	Dumpty
1234	57 Hockanum Blvd	Web Programming Lang.	Mickey
5678	2131 Forest Lake Ln.	Software Analysis	Minnie



<code>S_id</code>	<code>Address</code>	<code>Course</code>	<code>TA</code>
1234	57 Hockanum Blvd	Database Systems	Minnie

No “wild card”
in RA

```
SELECT * FROM Student_lecture
WHERE TA='Minnie' AND Course='Database Systems' ;
```

$\sigma_{\text{TA}='Minnie' \text{ AND } \text{Course}='Database Systems'}$ (`Student_lecture`)

Example: Selection (σ)

Find all tuples in `Student_lecture` relation that have a “Database Systems” course and “Minnie” as TA

`Student_lecture`

<code>S_id</code>	<code>Address</code>	<code>Course</code>	<code>TA</code>
1234	57 Hockanum Blvd	Database Systems	Minnie
2345	1400 E. Bellows	Database Systems	Humpty
3456	900 S. Detroit	Cloud Computing	Dumpty
1234	57 Hockanum Blvd	Web Programming Lang.	Mickey
5678	2131 Forest Lake Ln.	Software Analysis	Minnie



<code>S_id</code>	<code>Address</code>	<code>Course</code>	<code>TA</code>
1234	57 Hockanum Blvd	Database Systems	Minnie

Query output

Data source

$$\sigma_{TA='Minnie' \text{ AND } Course = 'Database Systems'}$$

`Student_lecture`

RA tree

```
SELECT * FROM Student_lecture
WHERE TA='Minnie' AND Course='Database Systems';
```

$$\sigma_{TA='Minnie' \text{ AND } Course='Database Systems'}(\text{Student_lecture})$$

Let's Try: Selection (σ)

Schedule

CNumber	Dept	Course	computingID	DateTime	Location
15783	CS	1111	up3f	MoWe 2:00pm - 3:15pm	Thornton Hall E303
...		
16633	CS	2110	nb3f	MoWeFr 1:00pm - 1:50pm	Mechanical Engr 205
16636	CS	2910	up3f	TuTh 12:30pm - 1:45pm	Rec 240
20294	CS	2910	up3f	Th 11:15am – 12:00pm Fr 3:00pm-3:45pm	
...		
16639	CS	4640	up3f	TuTh 2:00pm - 3:15pm	
16455	CS	4750	nb3f	MoWe 3:30pm- 4:45pm	
...		

$$\sigma_{Dept='CS' \text{ AND } Course = 2910}$$

Schedule

Consider a simplified version of Lou's List. Write RA to find all tuples in the `Schedule` relation that have "cs" and 2910 for Department and Course. Assume course is stored as integer.

$$\sigma_{Dept="CS" \text{ AND } Course=2910} (\text{Schedule})$$

Let's Try: Selection (σ) cont.

Schedule

CNumber	Dept	Course	computingID	DateTime	Location
15783	CS	1111	up3f	MoWe 2:00pm - 3:15pm	Thornton Hall E303
...		
16633	CS	2110	nb3f	MoWeFr 1:00pm - 1:50pm	Mechanical Engr 205
16636	CS	2910	up3f	TuTh 12:30pm - 1:45pm	Rice 340
20294	CS	2910	up3f	Th 11:15am - 12:00pm, Fr 3:00pm-3:45pm	Rice 536, Olsson 001
...		
16639	CS	4640	up3f	TuTh 2:00pm - 3:15pm	Clark Hall 108
16455	CS	4750	nb3f	MoWe 3:30pm- 4:45pm	Mechanical Engr 205
...		

$\sigma_{\text{Dept}=\text{"CS"} \text{ AND } \text{Course}=2910} \text{ (Schedule)}$



CNumber	Dept	Course	computingID	DateTime	Location
16636	CS	2910	up3f	TuTh 12:30pm - 1:45pm	Rice 340
20294	CS	2910	up3f	Th 11:15am - 12:00pm, Fr 3:00pm-3:45pm	Rice 536, Olsson 001

Projection (π)

- Unary operation – take one operand
- Return specified attributes of a relation

$$\pi_{A_1, A_2, \dots, A_m} (R) \quad \text{where } A_i \text{ is attribute of a relation } R$$

Example: Projection (π)

Find all `S_id` and `Course` in `Student_lecture` relation

Student_lecture

<code>S_id</code>	<code>Address</code>	<code>Course</code>	<code>TA</code>
1234	57 Hockanum Blvd	Database Systems	Minnie
2345	1400 E. Bellows	Database Systems	Humpty
3456	900 S. Detroit	Cloud Computing	Dumpty
1234	57 Hockanum Blvd	Web Programming Lang.	Mickey
5678	2131 Forest Lake Ln.	Software Analysis	Minnie



<code>S_id</code>	<code>Course</code>
1234	Database Systems
2345	Database Systems
3456	Cloud Computing
1234	Web Programming Lang.
5678	Software Analysis

```
SELECT S_id, Course  
FROM Student_lecture;
```

$\pi_{S_ID, Course} (\underline{Student_lecture})$

Example: Projection (π)

Find all S_id and Course in $Student_lecture$ relation

Student_lecture

S_id	Address	Course	TA
1234	57 Hockanum Blvd	Database Systems	Minnie
2345	1400 E. Bellows	Database Systems	Humpty
3456	900 S. Detroit	Cloud Computing	Dumpty
1234	57 Hockanum Blvd	Web Programming Lang.	Mickey
5678	2131 Forest Lake Ln.	Software Analysis	Minnie



S_id	Course
1234	Database Systems
2345	Database Systems
3456	Cloud Computing
1234	Web Programming Lang.
5678	Software Analysis

Query output

$\pi_{S_ID, Course}$

Data source

Student_lecture

RA tree

```
SELECT S_id, Course  
FROM Student_lecture;
```

$\pi_{S_ID, Course} (Student_lecture)$

More Example: Projection (π)

- Identical tuples collapse into a single tuple – no duplicates

RA follows “Set” properties

Student_lecture

S_id	Address	Course	TA
1234	57 Hockanum Blvd	Database Systems	Minnie
2345	1400 E. Bellows	Database Systems	Humpty
3456	900 S. Detroit	Cloud Computing	Dumpty
1234	57 Hockanum Blvd	Web Programming Lang.	Mickey
5678	2131 Forest Lake Ln.	Software Analysis	Minnie

Find all TAs in
Student_lecture
relation

↓

TA
Minnie
Humpty
Dumpty
Mickey

$\pi_{TA} (\underline{\text{Student_lecture}})$

π_{TA}
|
Student_lecture

Let's Try: Projection (π)

Schedule

CNumber	Dept	Course	computingID	DateTime
15783	CS	1111	up3f	MoWe 2:00pm - 3:15pm
...	
16633	CS	2110	nb3f	MoWeFr 1:00pm - 2:15pm
16636	CS	2910	up3f	TuTh 12:30pm - 1:45pm
20294	CS	2910	up3f	Th 11:15am - 12:30pm Fr 3:00pm-3:45pm
...	
16639	CS	4640	up3f	TuTh 2:00pm - 3:15pm
16455	CS	4750	nb3f	MoWe 3:30pm- 4:45pm
...	

$\pi_{Dept, Course}$

$\sigma_{computingID='up3f'}$

Schedule

Consider a simplified version of Lou's List (Spring 2019). Write RA to find all Departments and Courses in the Schedule relation that is taught by "up3f"

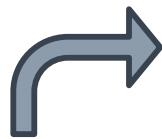
$\pi_{Dept, Course}(\sigma_{computingID="up3f"}(\underline{schedule}))$

Let's Try: Projection (π) cont.

Schedule

CNumber	Dept	Course	computingID	DateTime	Location
15783	CS	1111	up3f	MoWe 2:00pm - 3:15pm	Thornton Hall E303
...		
16633	CS	2110	nb3f	MoWeFr 1:00pm - 1:50pm	Mechanical Engr 205
16636	CS	2910	up3f	TuTh 12:30pm - 1:45pm	Rice 340
20294	CS	2910	up3f	Th 11:15am – 12:00pm, Fr 3:00pm-3:45pm	Rice 536, Olsson 001
...		
16639	CS	4640	up3f	TuTh 2:00pm - 3:15pm	Clark Hall 108
16455	CS	4750	nb3f	MoWe 3:30pm- 4:45pm	Mechanical Engr 205
...		

Notice: no duplicates



Dept	Course
CS	1111
CS	2910
CS	4640

$\pi_{\text{Dept, Course}}(\sigma_{\text{computingID}=\text{"up3f"}}(\text{schedule}))$

Renaming (ρ)

- Unary operation – take one operand
- **Change the schema**, not the instance
- Rename the result of expression E to R'
- Rename the resultant attributes to B₁, B₂, ..., B_n

$\rho_{R'(B_1, B_2, \dots, B_n)}(R)$ where B_i is new attribute name of a relation R'

contact

ID	email1	email2
mi1y	mickey@uva.edu	mi1y@uva.edu
mi1y	mi1y@uva.edu	mickey@uva.edu

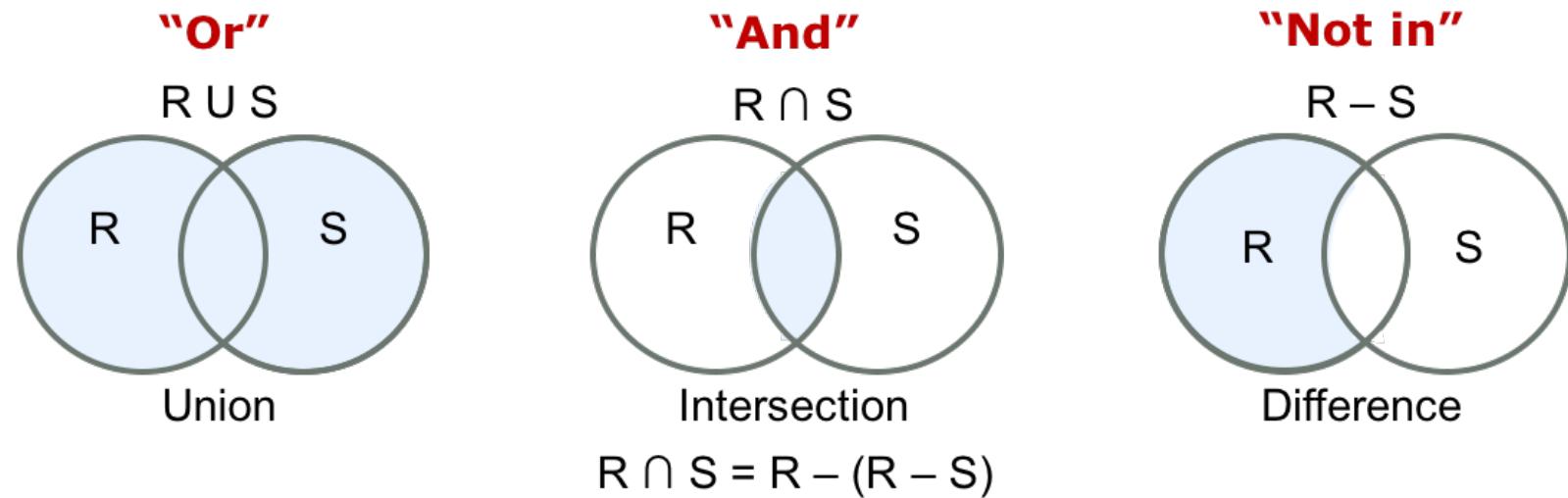
$\rho_{\text{friend_contact}(ID, \text{primary_email}, \text{alternative_email})}(\text{contact})$

The diagram illustrates the renaming process. Three red arrows point from the original attribute names in the contact table to their new names in the friend_contact table. The first arrow points from 'ID' to 'ID'. The second arrow points from 'email1' to 'primary_email'. The third arrow points from 'email2' to 'alternative_email'.

=

ID	primary_email	alternative_email
mi1y	mickey@uva.edu	mi1y@uva.edu
mi1y	mi1y@uva.edu	mickey@uva.edu

Union, Intersection, Difference

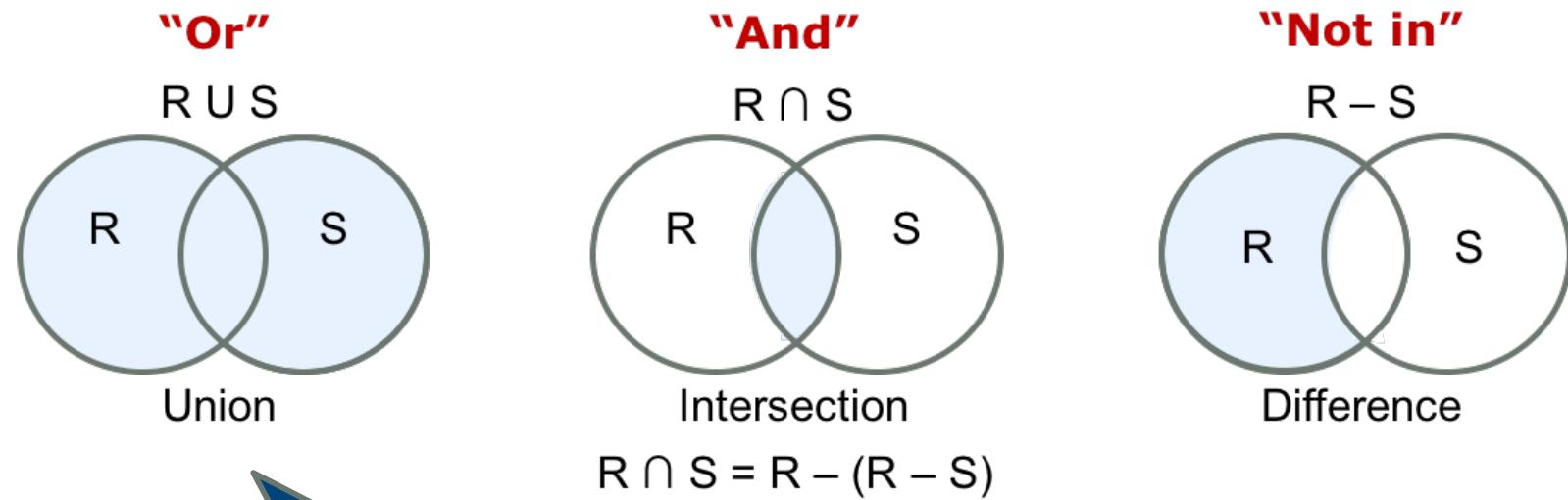


Binary operations – take left and right operands

Requirements:

- Schemas of R and S must have **same degree** (number of attributes)
- Corresponding attributes of R and S must be based on the **same domain / compatible data types**
- Attributes are in the **same order**

Union (R \cup S)

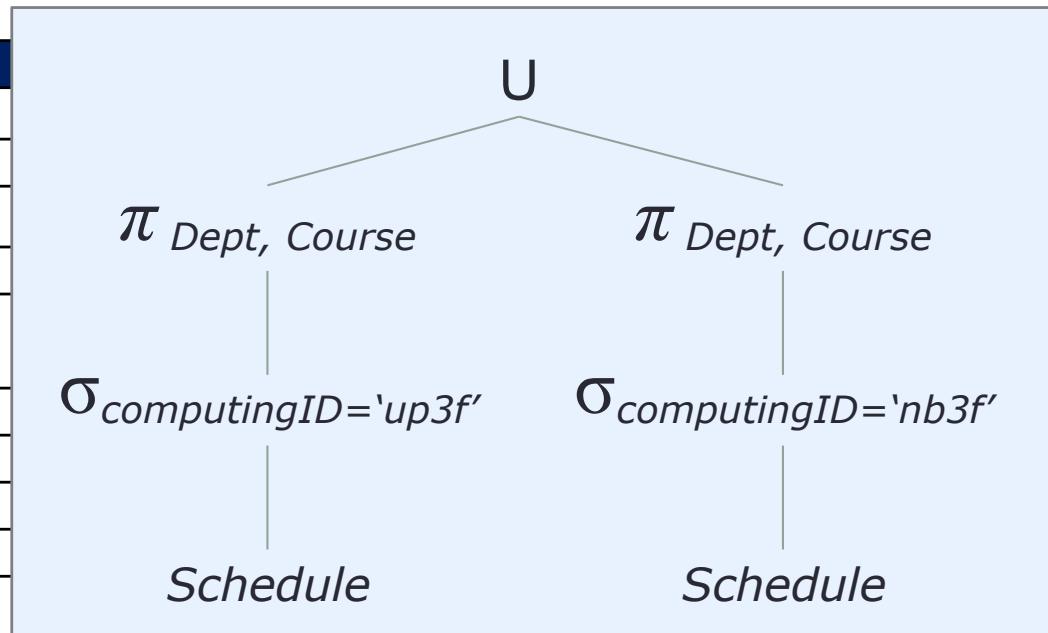


Each tuple of R and each tuple of S is put into the resultant relation

Let's Try: Union (\cup)

Schedule

CNumber	Dept	Course	computingID
15783	CS	1111	up3f
...
16633	CS	2110	nb3f
16636	CS	2910	up3f
20294	CS	2910	up3f
...
16639	CS	4640	up3f
16455	CS	4750	nb3f
...



Consider a simplified version of Lou's List (Spring 2019). Find all Departments and Courses in the Schedule relation that is taught by "up3f", "nb3f", or both.

$$\begin{aligned}
 & \pi_{Dept, Course}(\sigma_{computingID="up3f"}(schedule)) \cup \\
 & \pi_{Dept, Course}(\sigma_{computingID="nb3f"}(schedule))
 \end{aligned}$$

Let's Try: Union (U) cont.

Schedule

CNumber	Dept	Course	computingID	DateTime	Location
15783	CS	1111	up3f	MoWe 2:00pm - 3:15pm	Thornton Hall E303
...		
16633	CS	2110	nb3f	MoWeFr 1:00pm - 1:50pm	Mechanical Engr 205
16636	CS	2910	up3f	TuTh 12:30pm - 1:45pm	Rice 340
20294	CS	2910	up3f	Th 11:15am – 12:00pm, Fr 3:00pm-3:45pm	Rice 536, Olsson 001
...		
16639	CS	4640	up3f	TuTh 2:00pm - 3:15pm	Clark Hall 108
16455	CS	4750	nb3f	MoWe 3:30pm- 4:45pm	Mechanical Engr 205
...		

$$\pi_{\text{Dept, Course}}(\sigma_{\text{computingID}=\text{"up3f"}}(\text{schedule})) \cup$$

$$\pi_{\text{Dept, Course}}(\sigma_{\text{computingID}=\text{"nb3f"}}(\text{schedule}))$$

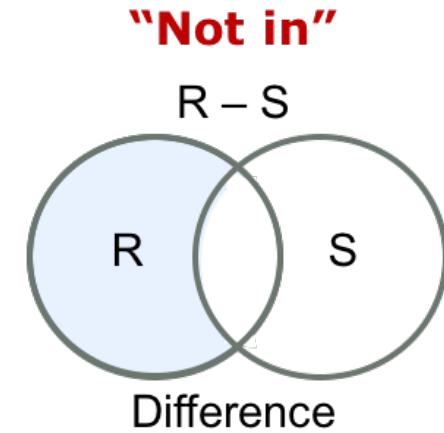
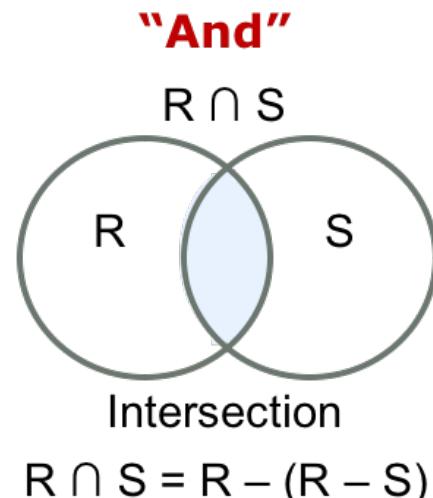
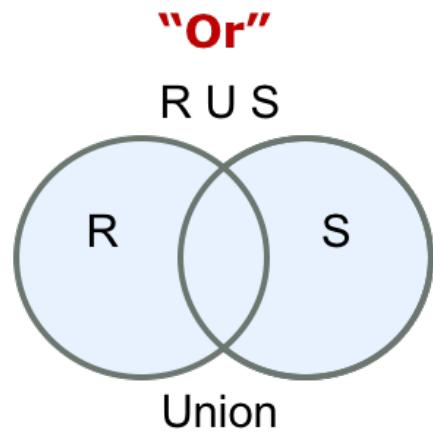
Dept	course
CS	1111
CS	2910
CS	4640

U

Dept	course
CS	2110
CS	4750

Dept	course
CS	1111
CS	2910
CS	4640
CS	2110
CS	4750

Intersection ($R \cap S$)

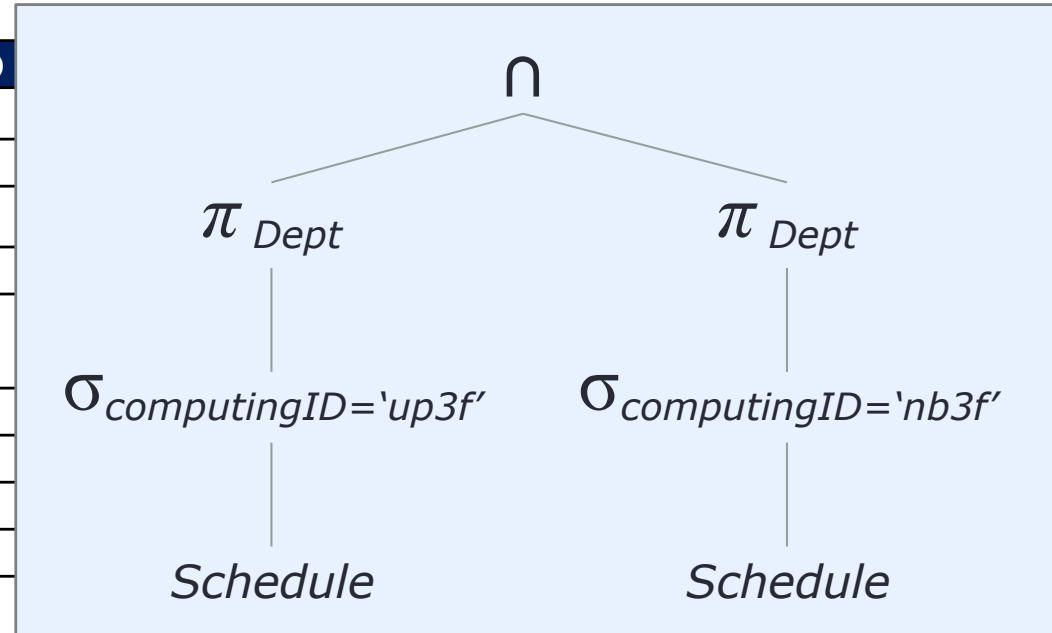


A tuple is in the resultant relation if and only if it is in both R and S

Let's Try: Intersection (\cap)

Schedule

CNumber	Dept	Course	computingID
15783	CS	1111	up3f
...
16633	CS	2110	nb3f
16636	CS	2910	up3f
20294	CS	2910	up3f
...
16639	CS	4640	up3f
16455	CS	4750	nb3f
...



Consider a simplified version of Lou's List (Spring 2019). Find all Departments in the Schedule relation that offer courses taught by "up3f" and "nb3f".

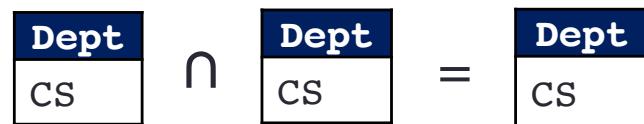
$$\pi_{Dept} (\sigma_{computingID="up3f"}(schedule)) \cap \pi_{Dept} (\sigma_{computingID="nb3f"}(schedule))$$

Let's Try: Intersection (\cap) cont.

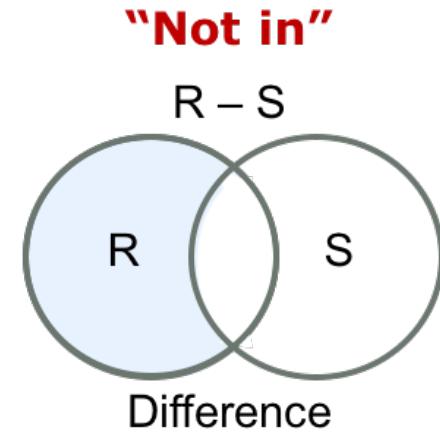
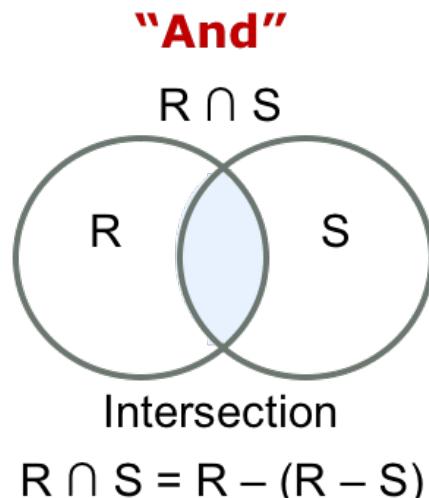
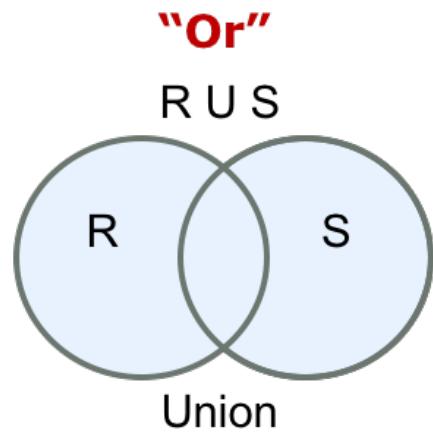
Schedule

CNumber	Dept	Course	computingID	DateTime	Location
15783	CS	1111	up3f	MoWe 2:00pm - 3:15pm	Thornton Hall E303
...		
16633	CS	2110	nb3f	MoWeFr 1:00pm - 1:50pm	Mechanical Engr 205
16636	CS	2910	up3f	TuTh 12:30pm - 1:45pm	Rice 340
20294	CS	2910	up3f	Th 11:15am – 12:00pm, Fr 3:00pm-3:45pm	Rice 536, Olsson 001
...		
16639	CS	4640	up3f	TuTh 2:00pm - 3:15pm	Clark Hall 108
16455	CS	4750	nb3f	MoWe 3:30pm- 4:45pm	Mechanical Engr 205
...		

$$\pi_{\text{Dept}} (\sigma_{\text{computingID}=\text{"up3f"}}(\text{schedule})) \cap$$

$$\pi_{\text{Dept}} (\sigma_{\text{computingID}=\text{"nb3f"}}(\text{schedule}))$$


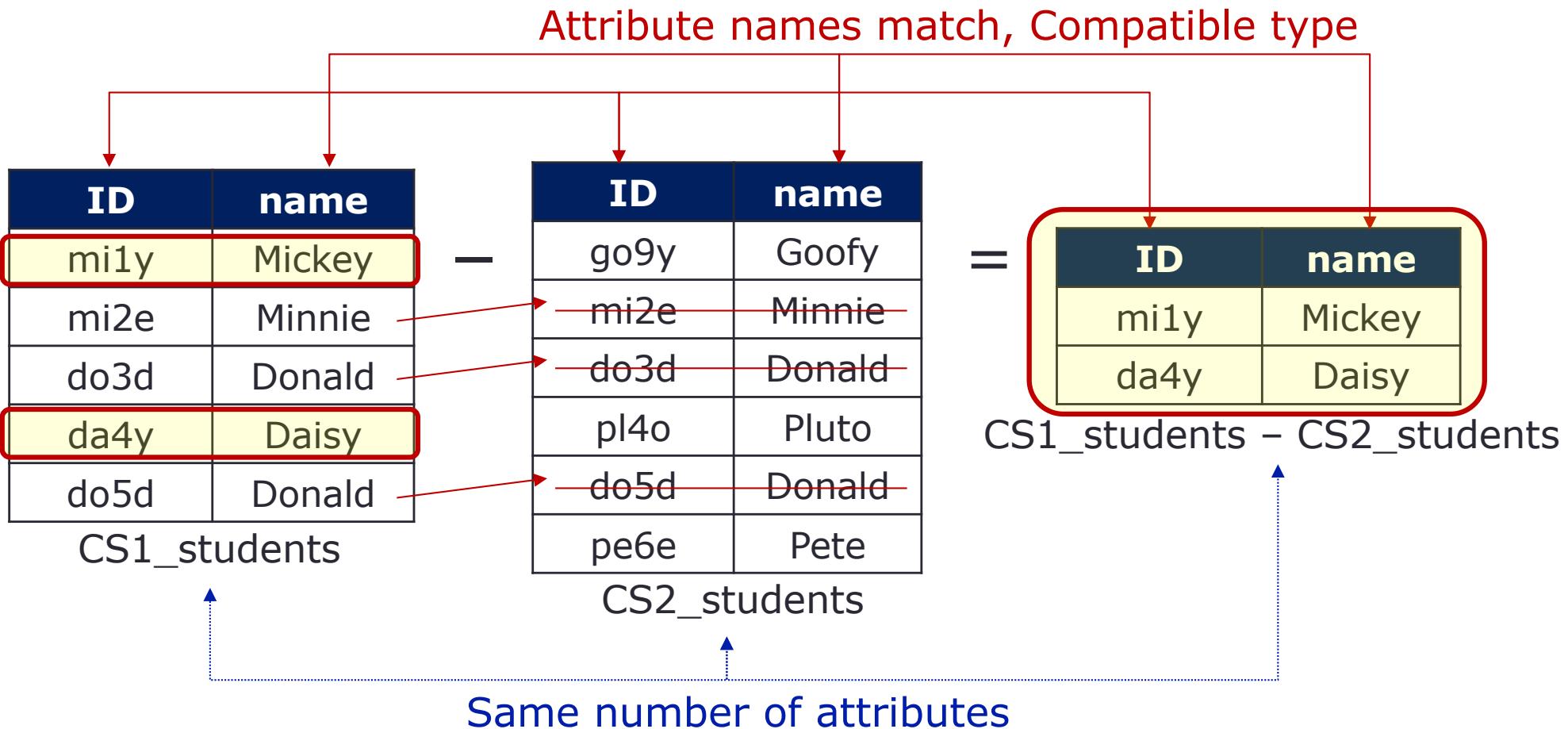
Difference ($R - S$)



A tuple is in the resultant relation if and only if it is in R but not in S

Example: Difference (-)

Find tuples that are in one relation but are not in another



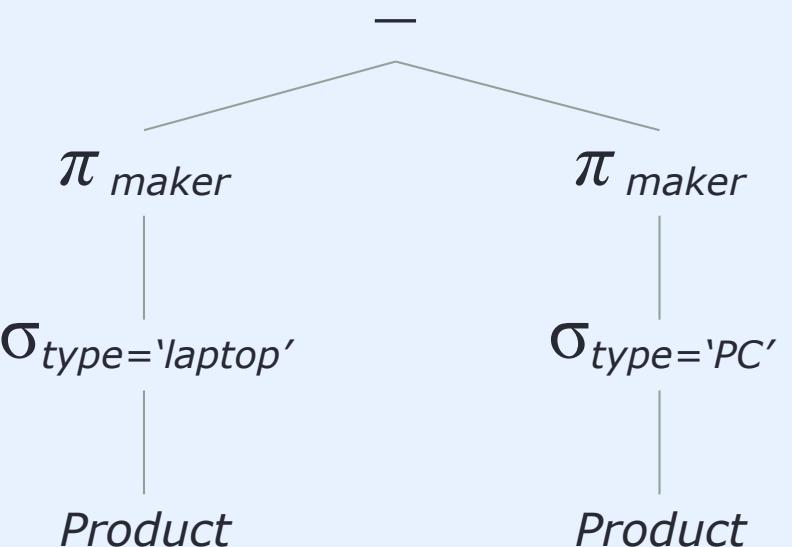
Let's Try: Difference (-)

Consider the following schema statements. Write RA to find manufacturers (makers) that sell Laptops but not PC's

Product (maker, model, type)

-- type is stored as "laptop" or "PC"

```
 $\pi_{maker}(\sigma_{type='laptop'}(Product)) -$   
 $\pi_{maker}(\sigma_{type='PC'}(Product))$ 
```



Cartesian Product ($R \times S$)

- Binary operations – take two operand
- So-called “cross-product” or “product”
- **Combine two relations**
- Usually not meaningful when it is performed alone

$$R \times S \neq S \times R$$

$R \times S$

where $R(a_1, a_2, \dots, a_n)$ and $S(b_1, b_2, \dots, b_k)$

Employee

ID	Name
hm1y	Humpty
dm2y	Dumpty

Department

EID	DeptName
hm1y	CS
dm2y	EE



Employee x Department

ID	Name	EID	DeptName
hm1y	Humpty	hm1y	CS
hm1y	Humpty	dm2y	EE
dm2y	Dumpty	hm1y	CS
dm2y	Dumpty	dm2y	EE

More Example: Product (x)

Contact (Let's call it C1)

$\rho_{C1(ID1, email1)}(\text{contact})$

ID	email
mi1y	mickey@uva.edu
mi2e	minnie@uva.edu
mi1y	mi1y@uva.edu

Contact (Let's call it C2)

$\rho_{C2(ID2, email2)}(\text{contact})$

ID	email
mi1y	mickey@uva.edu
mi2e	minnie@uva.edu
mi1y	mi1y@uva.edu

\times

=

$C1 \times C2$

Usually not meaningful
when it is performed
alone

ID1	email1	ID2	email2
mi1y	mickey@uva.edu	mi1y	mickey@uva.edu
mi1y	mickey@uva.edu	mi2e	minnie@uva.edu
mi1y	mickey@uva.edu	mi1y	mi1y@uva.edu
mi2e	minnie@uva.edu	mi1y	mickey@uva.edu
mi2e	minnie@uva.edu	mi2e	minnie@uva.edu
mi2e	minnie@uva.edu	mi1y	mi1y@uva.edu
mi1y	mi1y@uva.edu	mi1y	mickey@uva.edu
mi1y	mi1y@uva.edu	mi2e	minnie@uva.edu
mi1y	mi1y@uva.edu	mi1y	mi1y@uva.edu

More Example: Product (\times)

Contact (Let's call it C1)

$\rho_{C1(ID1, email1)}(contact)$

ID	email
mi1y	mickey@uva.edu
mi2e	minnie@uva.edu
mi1y	mi1y@uva.edu

Contact (Let's call it C2)

$\rho_{C2(ID2, email2)}(contact)$

ID	email
mi1y	mickey@uva.edu
mi2e	minnie@uva.edu
mi1y	mi1y@uva.edu

\times

$=$

Meaningful when it is followed by other operations.

Find all students who have more than one email

(2 steps: cross product, then select tuples)

$C1 \times C2$

ID1	email1	ID2	email2
mi1y	mickey@uva.edu	mi1y	mickey@uva.edu
mi1y	mickey@uva.edu	mi2e	minnie@uva.edu
mi1y	mickey@uva.edu	mi1y	mi1y@uva.edu
mi2e	minnie@uva.edu	mi1y	mickey@uva.edu
mi2e	minnie@uva.edu	mi2e	minnie@uva.edu
mi2e	minnie@uva.edu	mi1y	mi1y@uva.edu
mi1y	mi1y@uva.edu	mi1y	mickey@uva.edu
mi1y	mi1y@uva.edu	mi2e	minnie@uva.edu
mi1y	mi1y@uva.edu	mi1y	mi1y@uva.edu

Let's Try: Product (\times)

Find all students who have more than one email
(2 steps: cross product, then select tuples)

Write RA to solve this problem

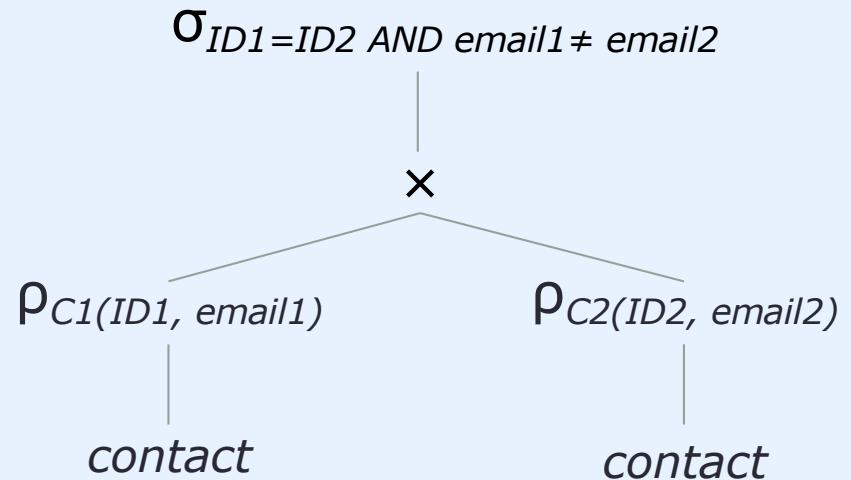
(for more practice, also write SQL to solve this problem)

Contact (Let's call it C1)

ID	email
mi1y	mickey@uva.edu
mi2e	minnie@uva.edu
mi1y	mi1y@uva.edu

Contact (Let's call it C2)

ID	email
mi1y	mickey@uva.edu
mi2e	minnie@uva.edu
mi1y	mi1y@uva.edu

$$\sigma_{ID1=ID2 \text{ AND } email1 \neq email2} (\rho_{C1(ID1, email1)}(\text{contact}) \times \rho_{C2(ID2, email2)}(\text{contact}))$$


Let's Try: Product (\times) cont.

Find all students who have more than one email
(2 steps: cross product, then select tuples)

Write RA to solve this problem

(for more practice, also write SQL to solve this problem)

Contact (Let's call it C1)

ID	email
mi1y	mickey@uva.edu
mi2e	minnie@uva.edu
mi1y	mi1y@uva.edu

Contact (Let's call it C2)

ID	email
mi1y	mickey@uva.edu
mi2e	minnie@uva.edu
mi1y	mi1y@uva.edu

$$\begin{aligned} & \sigma_{ID1=ID2 \text{ AND } email1 \neq email2} (\\ & \rho_{C1(ID1, email1)}(\text{contact}) \times \\ & \rho_{C2(ID2, email2)}(\text{contact})) \end{aligned}$$

```
SELECT * FROM C1, C2
WHERE C1.ID1 = C2.ID2
AND C1.email1 <> C2.email2
```

ID1	email1	ID2	email2
mi1y	mickey@uva.edu	mi1y	mi1y@uva.edu
mi1y	mi1y@uva.edu	mi1y	mickey@uva.edu

Natural Join ($R \bowtie S$)

- Binary operations – take two operand
- Merge relations on the specified condition

$R \bowtie S$

where R and S has a set of attributes that are in common

$$R \bowtie S = \pi_A (\sigma_C (R \times S))$$

3. Eliminate duplicate
common attributes

(attributes with same names)

1. Cross product

2. Check equality of all
common attributes

(attributes with same names)

Example: Natural Join (\bowtie)

R	S	$R \bowtie S$							
A	B	C	B	C	D	A	B	C	D
1	2	3	2	3	4	1	2	3	4
6	7	8	2	3	5	1	2	3	5
9	7	8	7	8	10	6	7	8	10
			7	9	9	9	7	8	10

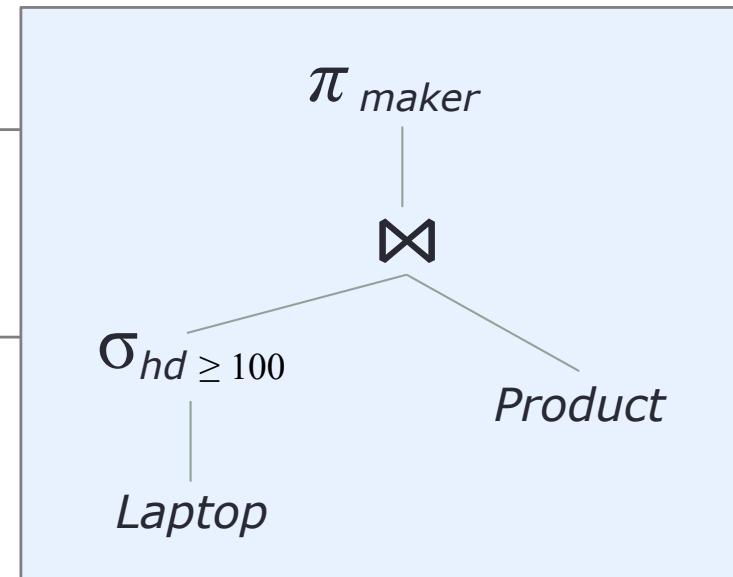
S	R	$S \bowtie R$					
B	C	D	A	B	C	D	A
2	3	4	1	2	3	4	1
2	3	5	6	7	8	5	1
7	8	10	9	7	8	10	6
7	9	9					9

Let's Try: Natural Join (\bowtie)

Consider the following schema statements. Write RA to find manufacturers (makers) that make laptops with a hard disk (hd) of at least 100GB

```
Product (maker, model, type)
    -- type is stored as "laptop" or "PC"
Laptop (model, speed, ram, hd, screen, price)
    -- hd stores size of hard disk
```

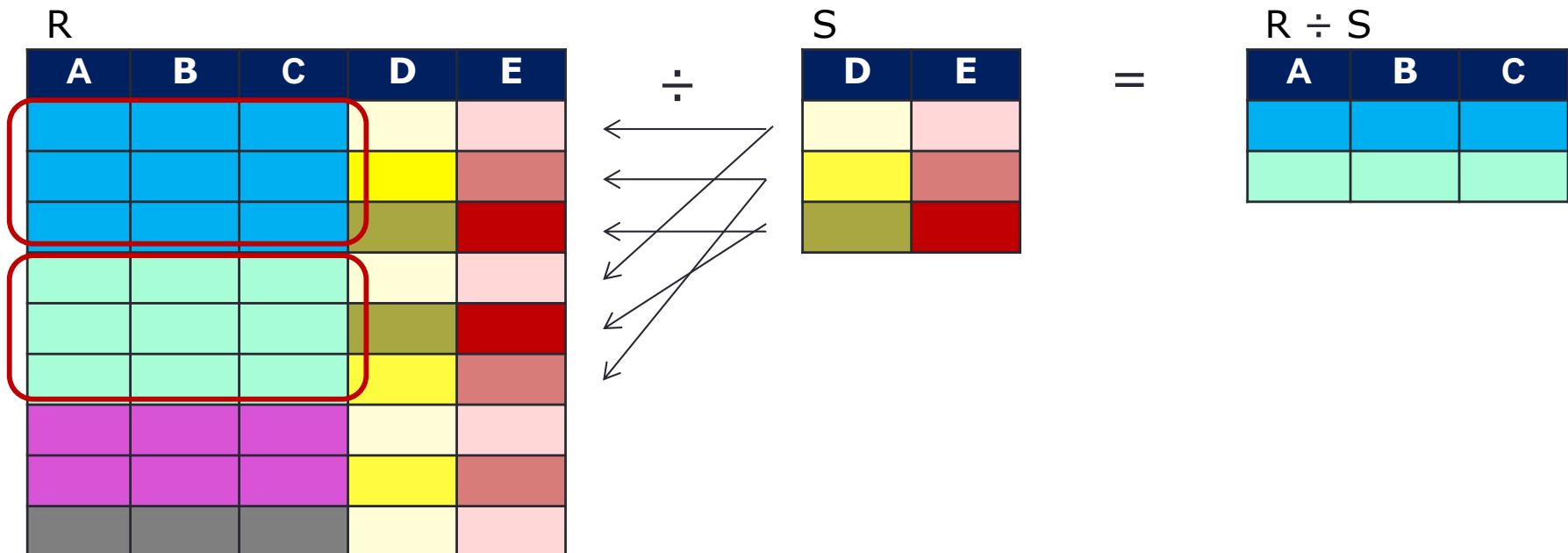
$\pi_{\text{maker}} (\text{Product} \bowtie (\sigma_{\text{hd} \geq 100} (\text{Laptop})))$



Division ($R \div S$)

- Binary operations – take two operand
- Use to find “**for all**” queries
- Find “A” for all “B” where “A” and “B” are sets of attributes
 $AB \div B = A$

$$R \div S = \pi_{A-B}(R) - \pi_{A-B}((\pi_{A-B}(R) \times S) - R)$$



Breakdown: Division ($R \div S$)

$$R \div S = \pi_{A-B}(R) - \pi_{A-B}(\underline{(\pi_{A-B}(R) \times S) - R})$$

where A and B are sets of attributes of R and S

1. Project attributes of R that are not in S , and then product with the divisor S

R	A	B	C	D
a	3	x	m	
a	1	x	o	
a	3	y	o	
b	1	x	m	
c	4	y	o	
b	2	y	n	
c	4	x	m	

S	C	D
y	o	
x	m	

$\pi_{A-B}(R)$	A	B
a	3	
a	1	
b	1	
c	4	
b	2	

S	C	D
y	o	
x	m	

\times

$=$

$\pi_{A-B}(R) \times S$	A	B	C	D
a	3	y	o	
a	3	x	m	
a	1	y	o	
a	1	x	m	
b	1	y	o	
b	1	x	m	
c	4	y	o	
c	4	x	m	
b	2	y	o	
b	2	x	m	

Breakdown: Division ($R \div S$)

$$R \div S = \pi_{A-B}(R) - \pi_{A-B}(\underline{(\pi_{A-B}(R) \times S) - R})$$

where A and B are sets of attributes of R and S

2. Extract tuples from the product that were not in R

R	A	B	C	D
a	3	x	m	
a	1	x	o	
a	3	y	o	
b	1	x	m	
c	4	y	o	
b	2	y	n	
c	4	x	m	

S	C	D
y	o	
x	m	

$\pi_{A-B}(R) \times S$

	A	B	C	D
a	3		y	o
a	3		x	m
a	1		y	o
a	1		x	m
b	1		y	o
b	1		x	m
c	4		y	o
c	4		x	m
b	2		y	o
b	2		x	m

✗

✗

✗

✗

✗

$(\pi_{A-B}(R) \times S) - R$

	A	B	C	D
a	1		y	o
a	1		x	m
b	1		y	o
b	2		y	o
b	2		x	m

Breakdown: Division ($R \div S$)

$$R \div S = \pi_{A-B}(R) - \underline{\pi_{A-B}((\pi_{A-B}(R) \times S) - R)}$$

where A and B are sets of attributes of R and S

3. Project only attributes that are in the original R but not in S

R			
A	B	C	D
a	3	x	m
a	1	x	o
a	3	y	o
b	1	x	m
c	4	y	o
b	2	y	n
c	4	x	m

S	
C	D
y	o
x	m

$$(\pi_{A-B}(R) \times S) - R$$

A	B	C	D
a	1	y	o
a	1	x	m
b	1	y	o
b	2	y	o
b	2	x	m

$$\pi_{A-B}(\pi_{A-B}(R) \times S) - R$$

A	B
a	1
b	1
b	2

Breakdown: Division ($R \div S$)

$$R \div S = \underline{\pi_{A-B}(R) - \pi_{A-B}(\pi_{A-B}(R) \times S) - R}$$

where A and B are sets of attributes of R and S

4. Extract only the tuples that are in the original R

R	A	B	C	D
a	3	x	m	
a	1	x	o	
a	3	y	o	
b	1	x	m	
c	4	y	o	
b	2	y	n	
c	4	x	m	

S	C	D
	y	o
	x	m

$\pi_{A-B}(R)$	A	B
✗	a	3
✗	a	1
✗	b	1
✗	c	4
✗	b	2

$$\pi_{A-B}(\pi_{A-B}(R) \times S) - R$$

—

	A	B
	a	1
	b	1
	b	2

=

	A	B
	a	3
	c	4

Short cut: Division ($R \div S$)

R

A	B	C	D
a	3	x	m
a	1	x	o
a	3	y	o
b	1	x	m
c	4	y	o
b	2	y	n
c	4	x	m

\div

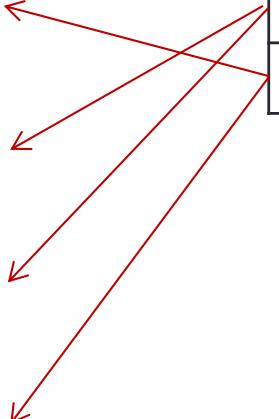
S

C	D
y	o
x	m

$=$

$R \div S$

A	B
a	3
c	4



Let's Try: Division (\div)

Consider the following schema statements. Write RA to find the names of sailors who have reserved **all** the boats

Boats (bid, bname, color)

Sailors (sid, sname, rating, age)

Reserves (sid, bid, day)

$$\pi_{sname, bid} (\text{Sailors} \bowtie \text{Reserves}) \div (\text{Boats})$$

$AB \div B = A$

sname	bid	
...	...	

\div

bid	bname	color
...

$=$

sname
...

The divisor is B.
Don't care the
remaining columns.

Assignment (\leftarrow)

$v \leftarrow E$ where v is a temporary variable representing a relation, E is an expression

Similar to assignment statement in programming

Example: Find manufacturers (makers) that make laptops with a hard disk (hd) of at least 100GB

Product (maker, model, type)

Laptop (model, speed, ram, hd, screen, price)

$\pi_{\text{maker}}(\text{Product} \bowtie (\sigma_{\text{hd} \geq 100}(\text{Laptop})))$

$R1 \leftarrow \sigma_{\text{hd} \geq 100}(\text{Laptop})$

$R2 \leftarrow \text{Product} \bowtie (R1)$

$\pi_{\text{maker}}(R2)$

Aggregate Function (G)

- Not relational operators
- Use **Group by** to help **summarize a column** in some way
- Five standard operators: sum, avg, count, min, and max

$$G_1, G_2, \dots, G_m, \mathbf{G}_{F_1(A_1), F_2(A_2), \dots, F_n(A_n)}(R)$$

where

A_1, A_2, \dots, A_n are attributes of a relation R

G_1, G_2, \dots, G_m are attributes on which to group;

F_1, F_2, \dots, F_n are aggregation functions on an attribute(A_i)

Example: Aggregate Function

Consider the following schema statements. Write RA to find the number of each of the colors of the boats

Boats (bid, bname, color)

Sailors (sid, sname, rating, age)

Reserves (sid, bid, day)

$$\pi_{\text{color}, \text{count}(\text{bid})} (\text{color} \text{ G } \text{count}(\text{bid}) \text{ (Boats) })$$

Result in

color	count(bid)
...	...

$$\pi_{\text{color}, \text{cnt}} (\text{color} \text{ G } \text{count}(\text{bid}) \rightarrow \text{number_boats} \text{ (Boats) })$$

Result in

color	number_boats
...	...

Let's Try: Aggregate Function

Consider the following schema statements. Write RA to find the the average and max loan amount of each customer

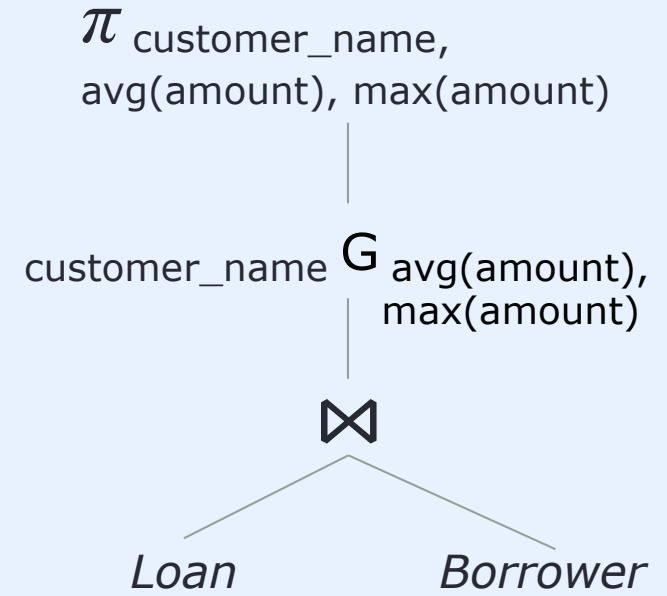
Loan (loan_number, branch_name, amount)

Borrower (customer_name, loan_number)

$$\pi_{\text{customer_name}, \text{avg}(\text{amount}), \text{max}(\text{amount})} (\text{customer_name} \text{ G } \text{avg}(\text{amount}), \text{max}(\text{amount}) \text{ (Loan } \bowtie \text{ Borrower) })$$

Result in

customer_name	avg(amount)	max(amount)
...	...	



Summary RA Operators

Unary operations

Take one relation, return a new relation

Selection $\sigma_p(r)$

Find tuples that satisfy a given condition

Project $\pi_{A_1, A_2, \dots, A_m}(r)$

Slice a relation, return a new relation with certain attributes

Rename $\rho_{x(A_1, A_2, \dots, A_n)}(E)$

Rename the result of expression E to x; rename resultant attributes to A_1, A_2, \dots

Additional Expressiveness Assignment
Aggregate functions

Binary operations

Take two relation, return a new relation

Union $r \cup s$ -- “or”

Combine tuples from 2 relations
[require: same number of attributes; compatible domains]
[result: same attributes]

Intersection $r \cap s$ -- “and”

Combine tuples from 2 relations
[require: same number of attributes; compatible domains]
[result: same attributes]

Set difference $r - s$

Find tuples that are in one relation but are not in another
[require: same number of attributes; compatible domains]
[result: same attributes]

Cartesian product $r \times s$

Combine 2 relations, all combination
[result: combined attributes]

Natural join $r \bowtie s$

Select tuples that satisfy the matching conditions from combined relations
[result: combined attributes]

Division $r \div s$

Similar to $AB \div B$
Find “A” for all “B”
[require: there exists B’s attributes in A]
[result: A schema]

Wrap-Up

Relational operators

- Selection, projection
- Renaming
- Set operations, Cartesian product, Natural join
- Division

Additional operators

- Assignment, aggregate function

What's next?

- Translating between SQL and RA
- RA tree
- Query cost estimation