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]

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

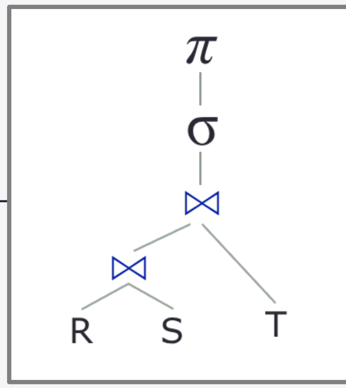
1001010101100
0010111101010
001010100000
1010010100

Computers
("How" to execute)

RDBMS

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

SQL
("What" data to get)



RA
("How" to get the data)

1001010101100
0010111101010
001010100000
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

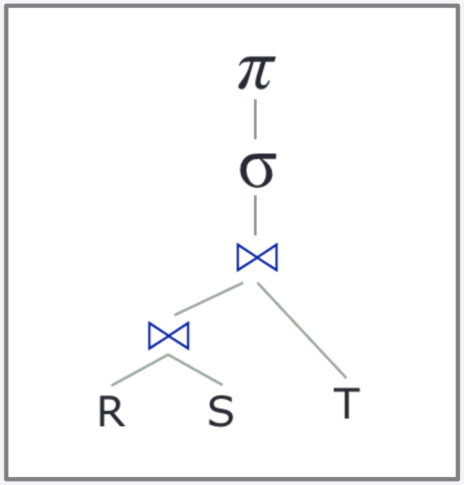
SQL is a declarative language

- 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

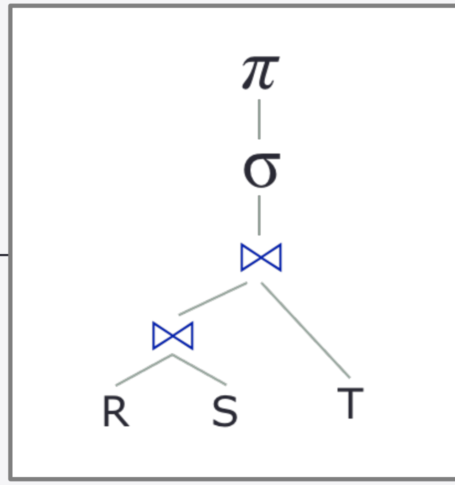
Why? (DBMS and Query Processors)

RDBMS

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

SQL

"What" data to get



RA

"How" to get the data

1001010101100
0010111101010
001010100000
1010010100

Computers

Needs "how" to get data

RA tree

(logical plan)

(physical plan)

Example: Database Internals

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

Define the semantics of a query

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

Every operator outputs a relation

Query output

$\pi_{\textit{S\_id ID, Course}}$

$\sigma_{\textit{Teaching\_assistant <> “Dumpty”}}$

Student_lecture

Data source

RA tree (Query plan)

Tuples flow up the query plan, getting filtered and modified

(read tree from bottom to top)

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

RA Operator Categories

Operations that remove parts of a relation

Unary operations

Take one relation, return a ne… a new relation

Selection $\sigma_p(r)$

Find tuples that satisfy a given condition

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

Slice a relation, return a new relation with certain attributes

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

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

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 …ibutes; compatible domains] […ult: 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 ÷ B
Find "A" for all "B"
[require: there exists B's attributes in A]
[result: A schema]

Addition… Expressi…

Operations that change the schema of a relation

RA Operator Categories

Operations that combine the tuples of two relations

Unary operations

Take one relation, return a new relation

Selection $\sigma_p(r)$

Find tuples that satisfy a given condition

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

Slice a relation, return a new relation with certain attributes

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

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

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 ÷ B
Find "A" for all "B"
[require: there exists B's attributes in A]
[result: A schema]

Extended operation

Additional Expressiveness

Assignment
Aggregate functions

Selection (σ)

- Unary operation – take one operand
- Return all tuples that satisfy a condition (filter tuples)
- Conditions can be =, <, ≤, >, ≥, <> 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

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	Address	Course	TA
1234	57 Hockanum Blvd	Database Systems	Minnie

No "wild card" in RA

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

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

Example: Selection (σ)

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

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	Address	Course	TA
1234	57 Hockanum Blvd	Database Systems	Minnie

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

Data source: *Student_lecture*

RA tree

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

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

Projection (π)

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

$$\pi_{A1,\ A2,\ \ldots,\ Am}\,(R)\quad \textit{where } A_i \textit{ is attribute of a relation } R$$

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

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

$$\pi_{S\_ID,\ Course}(Student\_lecture)$$

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 `TA`s in `Student_lecture` relation

TA
Minnie
Humpty
Dumpty
Mickey

$\pi_{TA}(Student\_lecture)$

$$\pi_{TA}$$

|

Student_lecture

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 B_1, B_2, …, B_n

$\rho_{R'(B1,\ B2,\ \ldots,\ Bn)}(R)$ *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

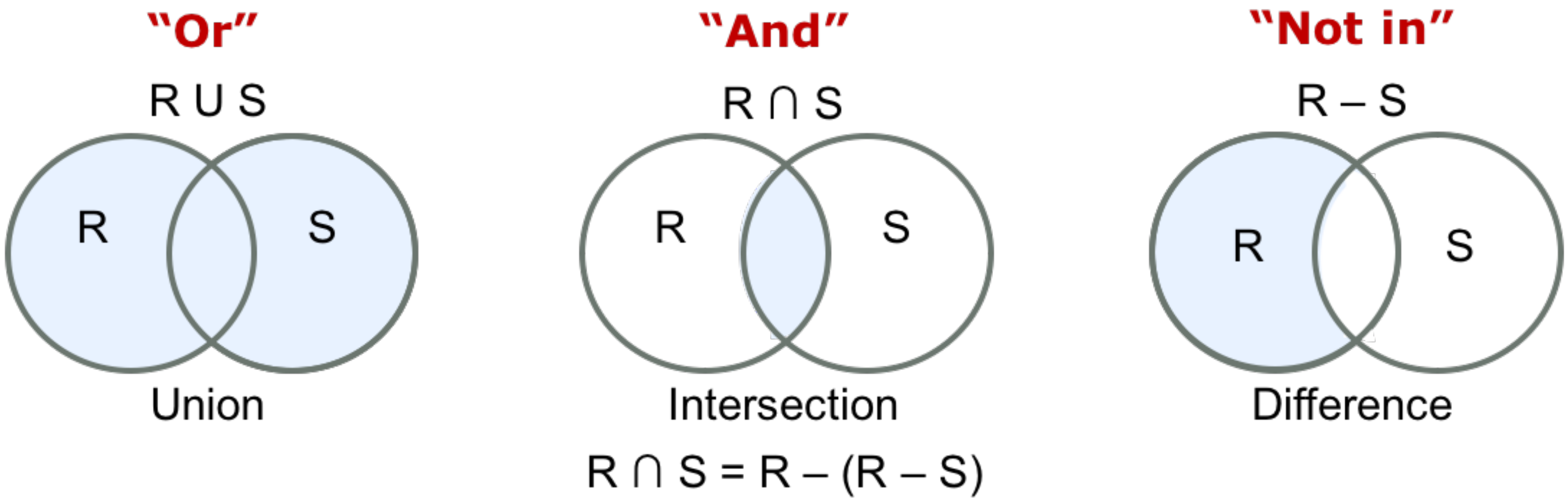
$\rho_{\text{friend\_contact(ID, primary\_email, alternative\_email)}}(\text{contact})$

=

friend_contact

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

Union, Intersection, Difference

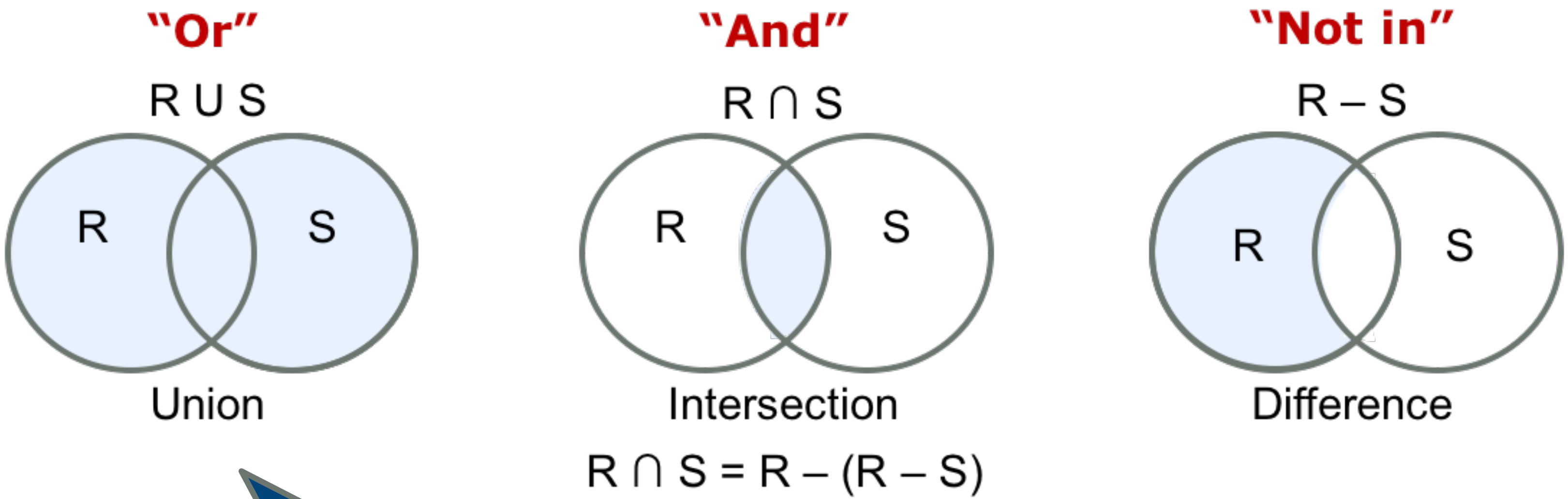


Binary operations – take left and right operands

Requirements:

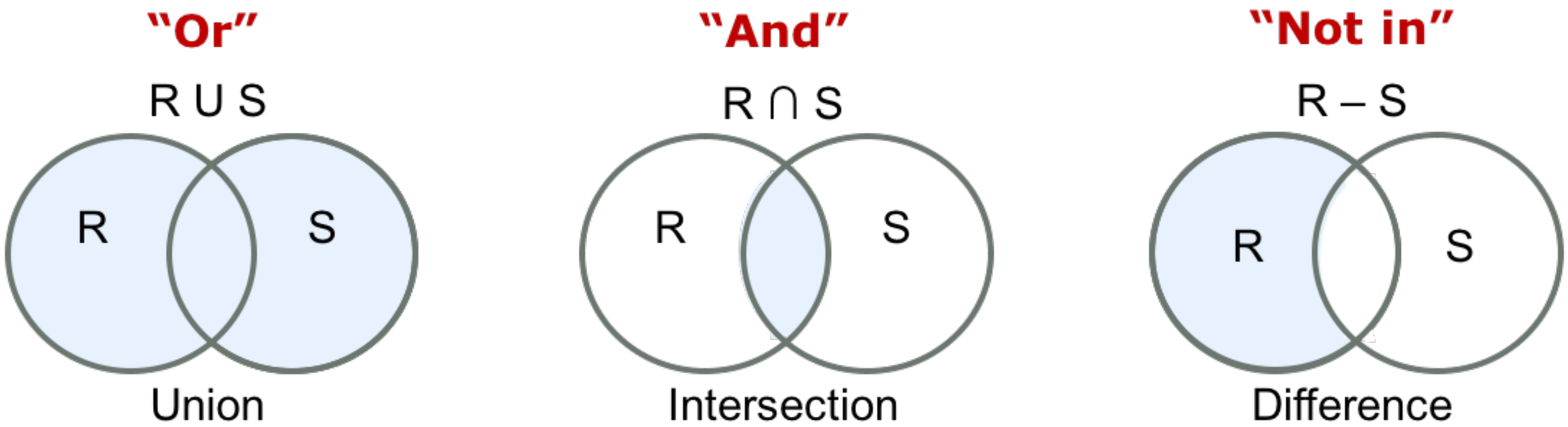
- Schemas of R and S must have same degree (number of attributes)
- Corresponding attributes of R and S must be based on the same domain / compatible data types
- Attributes are in the same order

Union (R U S)



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

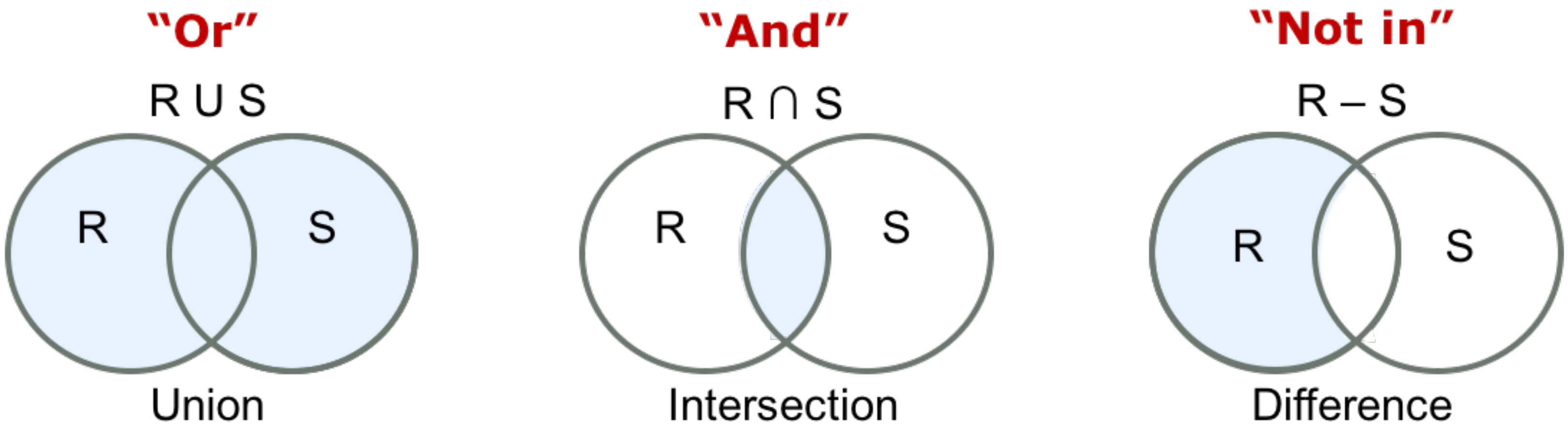
Intersection (R ∩ S)



$R \cap S = R - (R - S)$

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

Difference (R – S)



R ∩ S = R – (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

Attribute names match, Compatible type

ID	name
mi1y	Mickey
mi2e	Minnie
do3d	Donald
da4y	Daisy
do5d	Donald

CS1_students

–

ID	name
go9y	Goofy
~~mi2e~~	~~Minnie~~
~~do3d~~	~~Donald~~
pl4o	Pluto
~~do5d~~	~~Donald~~
pe6e	Pete

CS2_students

=

ID	name
mi1y	Mickey
da4y	Daisy

CS1_students – CS2_students

Same number of attributes

Cartesian Product (R × S)

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

R x S ≠ S x R

R x S   *where* $R(a_1, a_2, \ldots, a_n)$ *and* $S(b_1, b_2, \ldots, b_k)$

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 (×)

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

=

C1 × 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

Usually not meaningful when it is performed alone

More Example: Product (×)

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

=

C1 × 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

Meaningful when it is followed by other operations.

Find all students who have more than one email

(2 steps: cross product, then select tuples)

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*

$$R \bowtie S = \underline{\pi_A} (\underline{\sigma_C} \underline{(R \times S)})$$

1. Cross product

2. Check equality of all common attributes (attributes with same names)

3. Eliminate duplicate common attributes (attributes with same names)

Example: Natural Join (⋈)

R

A	B	C
1	2	3
6	7	8
9	7	8

⋈

S

B	C	D
2	3	4
2	3	5
7	8	10
7	9	9

=

R ⋈ S

A	B	C	D
1	2	3	4
1	2	3	5
6	7	8	10
9	7	8	10

S

B	C	D
2	3	4
2	3	5
7	8	10
7	9	9

⋈

R

A	B	C
1	2	3
6	7	8
9	7	8

=

S ⋈ R

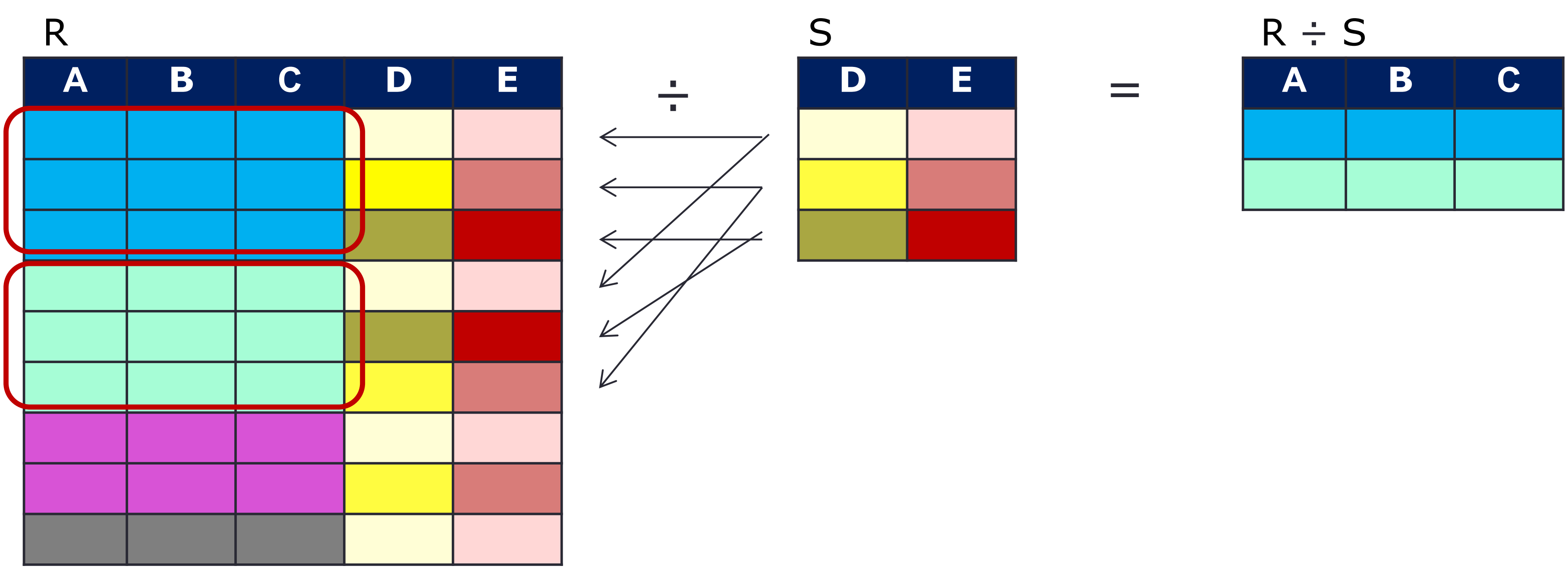
B	C	D	A
2	3	4	1
2	3	5	1
7	8	10	6
7	8	10	9

Division (R ÷ 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 ÷ B = A

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



Short cut: Division (R ÷ 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

=

R ÷ S

A	B
a	3
c	4

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)
```

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

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

$R2 \leftarrow$ Product $\bowtie$ (R1)

π_{maker} (R2)

Aggregate Function ($\mathbf{G}$)

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

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

where

A_1, A_2, …, A_n are attributes of a relation R

G_1, G_2, .., G_m are attributes on which to group;

F_1, F_2, …, 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)
```

$$\pi_{color,\ count(bid)}\ (\ _{color}G_{\ count(bid)}\ (Boats)\ )$$

Result in

color	count(bid)
…	…

$$\pi_{color,\ number\_boats}\ (\ _{color}G_{\ count(bid) \rightarrow number\_boats}\ (Boats)\ )$$

Result in

color	number_boats
…	…

Summary RA Operators

Unary operations

Take one relation, return a new relation

Selection $\sigma_p(r)$

Find tuples that satisfy a given condition

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

Slice a relation, return a new relation with certain attributes

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

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

Additional Expressiveness

- Assignment
- Aggregate functions

Binary operations

Take two relation, return a new relation

Union r ∪ s -- "or"

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

Intersection r ∩ 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 × s

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

Natural join r ⋈ s

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

Division r ÷ s

Similar to AB ÷ 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