

# Introduction to database systems

Relational algebra

# Relational algebra

- Relational algebra defines a **sequence of operations** that are carried out over a set of relations.
- The **results** and operands are **relations**.

# Groups of relational algebra operations

- **simple** operations:

- projection, selection, renaming

- **set** operations:

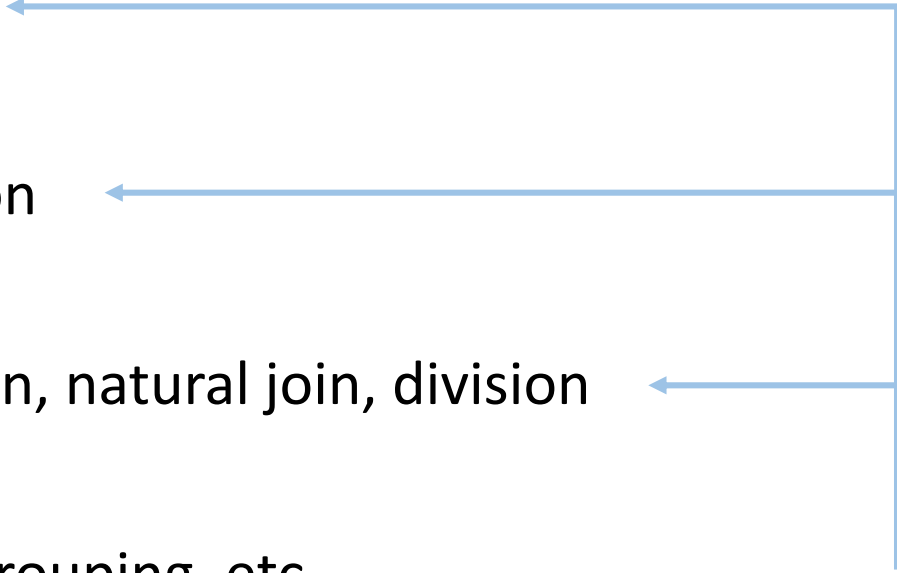
- union, set-difference, intersection

- **product** operations:

- cartesian product,  $\theta$ -join, equijoin, natural join, division

- **other** operations:

- semijoin, outer join, aggregate, grouping, etc.

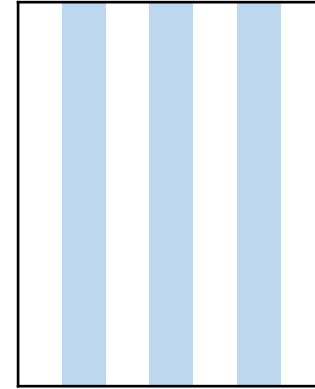


Basic operations with which we can derive others.

# Simple operations

## Projection

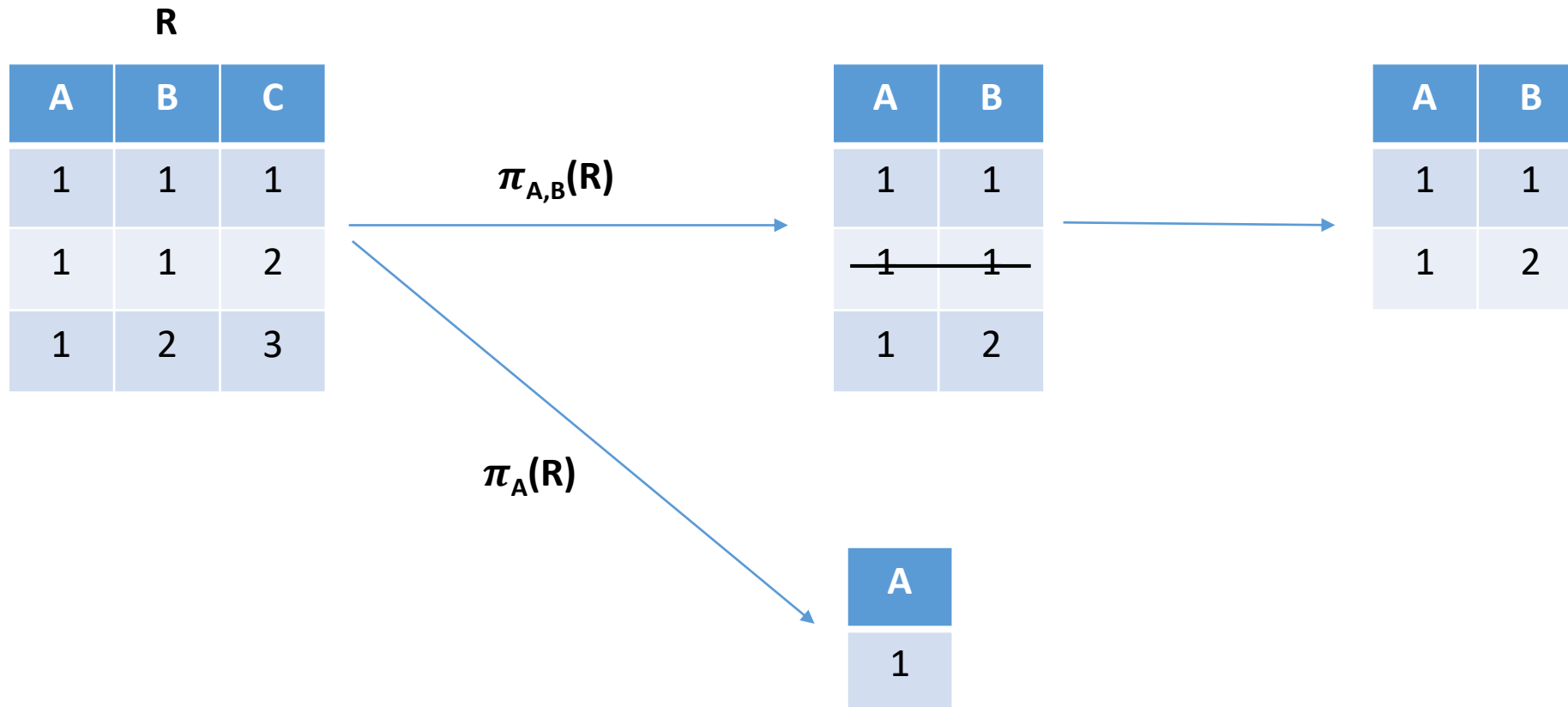
$$\pi_S(R)$$



- Works on a single relation R; returns a relation, which contains only those attributes (columns) that are defined by the list S.
- The projection eliminates duplicates.

# Simple operations

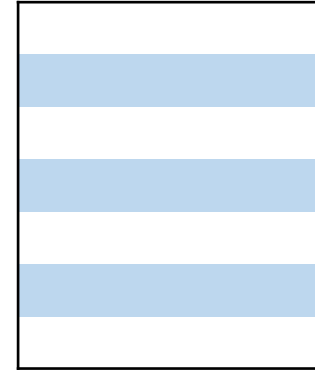
## Projection (example)



# Simple operations

## Selection

$$\sigma_{\text{predicate}}(R)$$



- Works on a single relation R; returns a relation, which contains only those tuples (rows) from the relation R that satisfy a given condition (predicate)

# Simple operations

## Selection (example)

R					
A	B	C			
1	1	1	$\sigma_{A \neq C \wedge B < 2}(R)$	A	B
1	1	2		1	1
1	2	3		2	3

# Simple operations

## Renaming

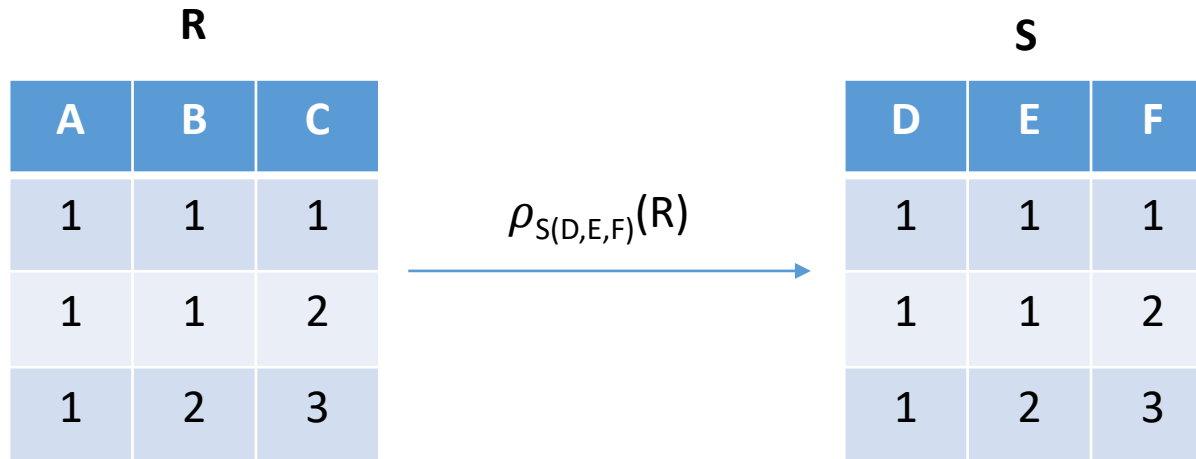
$$\rho_{s(S)}(R)$$

- Renaming of a relation R;
  - s is a new relation name,
  - S is a list of new attribute names.



# Simple operations

## Renaming (example)

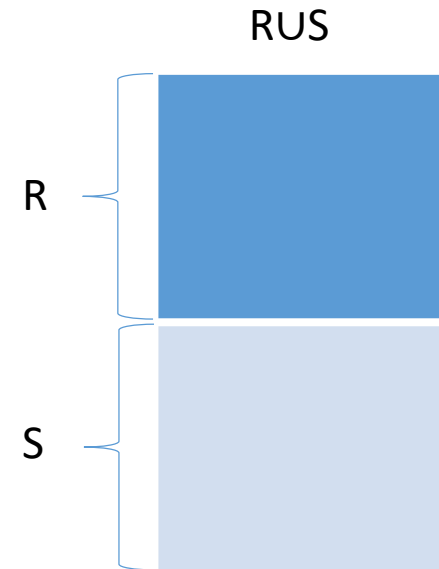


# Set operations

## Union

**RUS**

- Union of the relations R and S
- The condition for the execution of all set operations is that the relations are compatible with one another (the same number of attributes and that the same attributes have the same domains).



# Set operations

## Union (example)

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

S		
A	B	C
1	2	3
2	4	6
3	6	9

RUS

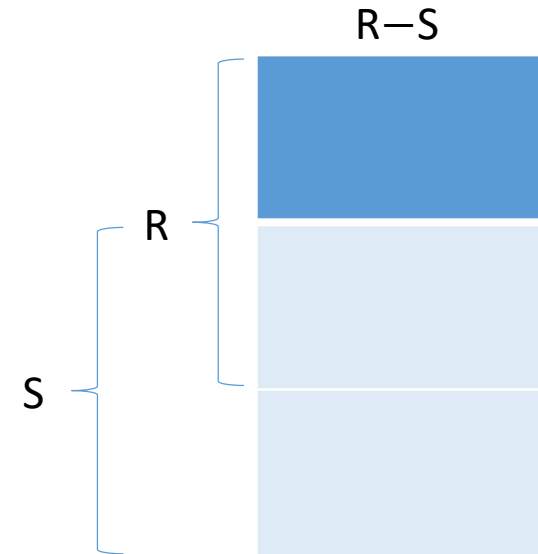
A	B	C
1	2	3
4	5	6
7	8	9
2	4	6
3	6	9

# Set operations

## Set-difference

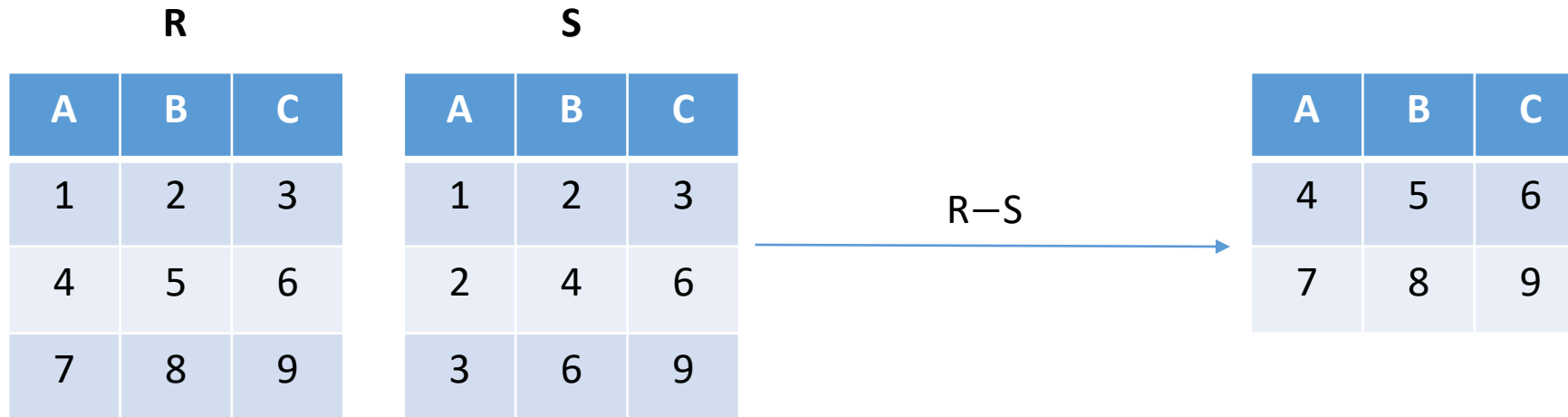
**$R - S$**

- Set-difference of the relations R and S.
- The condition for the execution of all set operations is that the relations are compatible with one another (the same number of attributes and that the same attributes have the same domains).



# Set operations

## Set-difference (example)

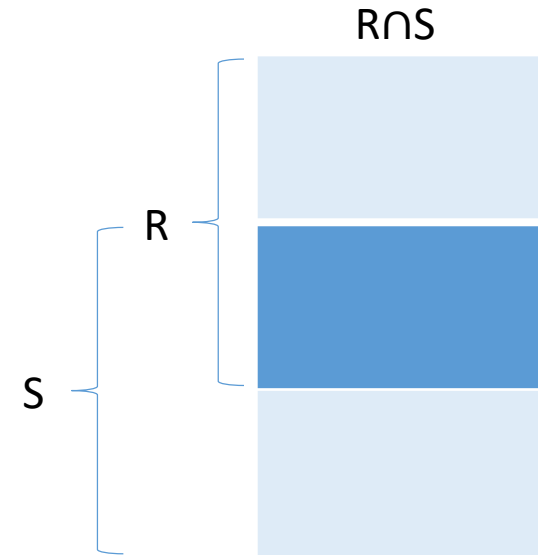


# Set operations

## Intersection

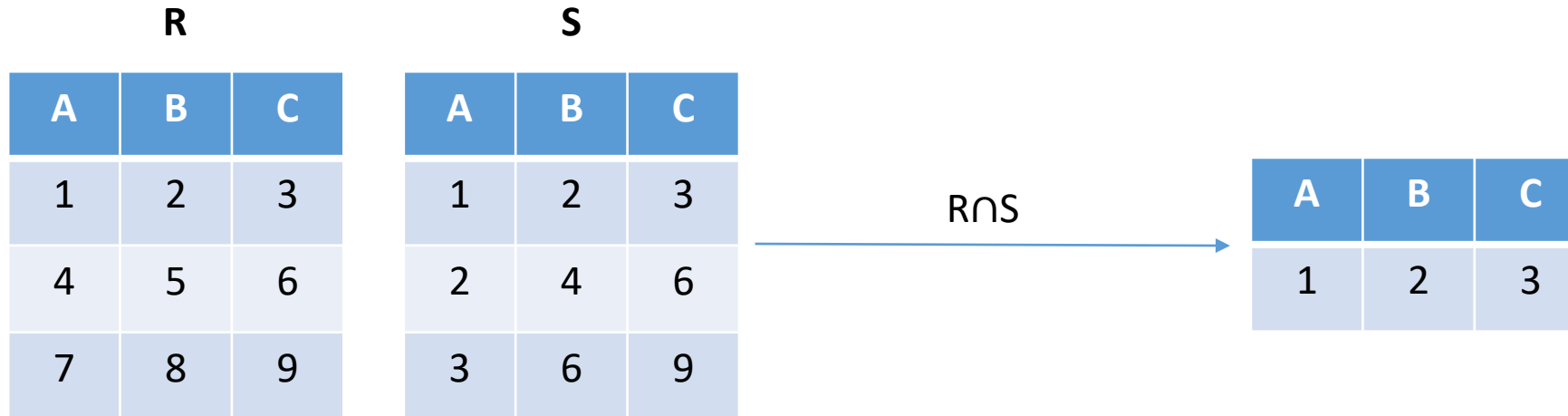
$$R \cap S$$

- Intersection of the relations R and S.
  - Also:  $R \cap S = R - (R - S)$
- The condition for the execution of all set operations is that the relations are compatible with one another (the same number of attributes and that the same attributes have the same domains).



# Set operations

## Intersection (example)



# Product operations

## Cartesian product

**$R \times S$**

R	S		R×S	
a	1	=	a	1
b	2		a	2
	3		a	3
			b	1
			b	2
			b	3

- Cartesian product of the relations R and S is a relation, which contains one tuple for each pair of tuples from the relations R and S.



# Product operations

## Cartesian product (example)

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

S	
D	E
1	2
3	4

$R \times S$

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

# Product operations

## Cartesian product (example)

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

S	
D	E
1	2
3	4

$R \times S$

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

# Product operations

## Cartesian product (example)

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

S	
D	E
1	2
3	4

$R \times S$

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

# Product operations

## Cartesian product (example)

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

S	
D	E
1	2
3	4

$R \times S$

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

# Product operations

## Theta join

$$R \bowtie_{\theta} S$$

- $\theta$ -join of the relations R and S is a cartesian product where we keep only those tuples which satisfy the condition  $\theta$ .

# Product operations

## Theta join (example)

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

S	
D	E
1	2
3	4

$$R \bowtie_{C \geq D \wedge A = E} S$$

A	B	C	D	E
<del>1</del>	<del>2</del>	<del>3</del>	<del>1</del>	<del>2</del>
<del>1</del>	<del>2</del>	<del>3</del>	<del>3</del>	<del>4</del>
<del>4</del>	<del>5</del>	<del>6</del>	<del>1</del>	<del>2</del>
4	5	6	3	4
<del>7</del>	<del>8</del>	<del>9</del>	<del>1</del>	<del>2</del>
<del>7</del>	<del>8</del>	<del>9</del>	<del>3</del>	<del>4</del>

# Product operations

## Equijoin

$$R \bowtie_{\theta=} S$$

- Equijoin of the relations R and S is a  $\theta$ -join, where condition  $\theta_=_$  contains only equalities.

# Product operations

## Equijoin (example)

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

S	
D	E
1	2
3	4

$$R \bowtie_{A=E} S$$

A	B	C	D	E
<del>1</del>	<del>2</del>	<del>3</del>	<del>1</del>	<del>2</del>
<del>1</del>	<del>2</del>	<del>3</del>	<del>3</del>	<del>4</del>
<del>4</del>	<del>5</del>	<del>6</del>	<del>1</del>	<del>2</del>
4	5	6	3	4
<del>7</del>	<del>8</del>	<del>9</del>	<del>1</del>	<del>2</del>
<del>7</del>	<del>8</del>	<del>9</del>	<del>3</del>	<del>4</del>



# Product operations

## Natural join

$$R \bowtie S$$

R		S		R $\bowtie$ S		
A	B	B	C	A	B	C
a	1	1	x	a	1	x
b	2	1	y	a	1	y
		3	z			

- Natural join of the relations R and S is an equijoin over all common attributes, where we keep only one occurrence of the common attributes (no duplicate attributes).
  - If relations R and S do not have common attributes, natural join equals cartesian join.

# Product operations

## Natural join (example)

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

S		
C	D	E
6	1	2
9	3	4

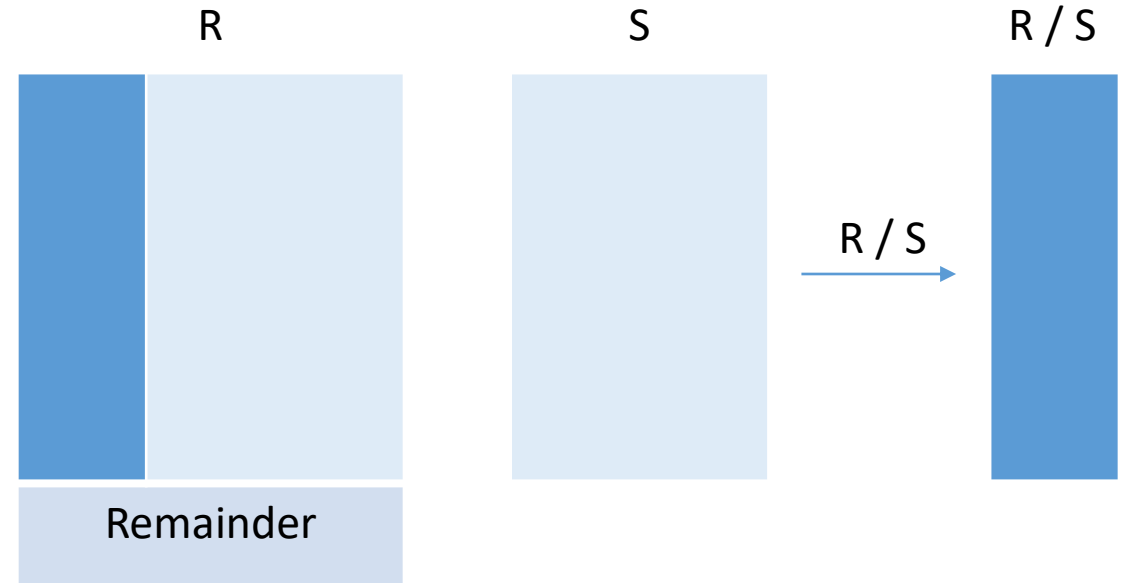
$R \bowtie S$  →

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

# Product operations

## Division

**$R / S$**



- The quotient of the relations R and S includes only those tuples which cover the relation S.
- The new scheme is obtained by subtracting the R and S schemes.

# Product operations

## Division (example)

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

S	
B	C
1	2
3	4

$R / S$  →

A	D
1	1

# Product operations

**Division (example – Who has passed all the exams?)**

passed

Student ID	Subject ID	...
10000	101	...
10001	101	...
10001	102	...
10002	101	...
10003	102	...
...	...	...

exams

Subject ID	...
101	...
102	...

$$\pi_{\text{Student ID, Subject ID}}(\text{passed}) / \pi_{\text{Subject ID}}(\text{exams})$$

Student ID
10001

# Other operations

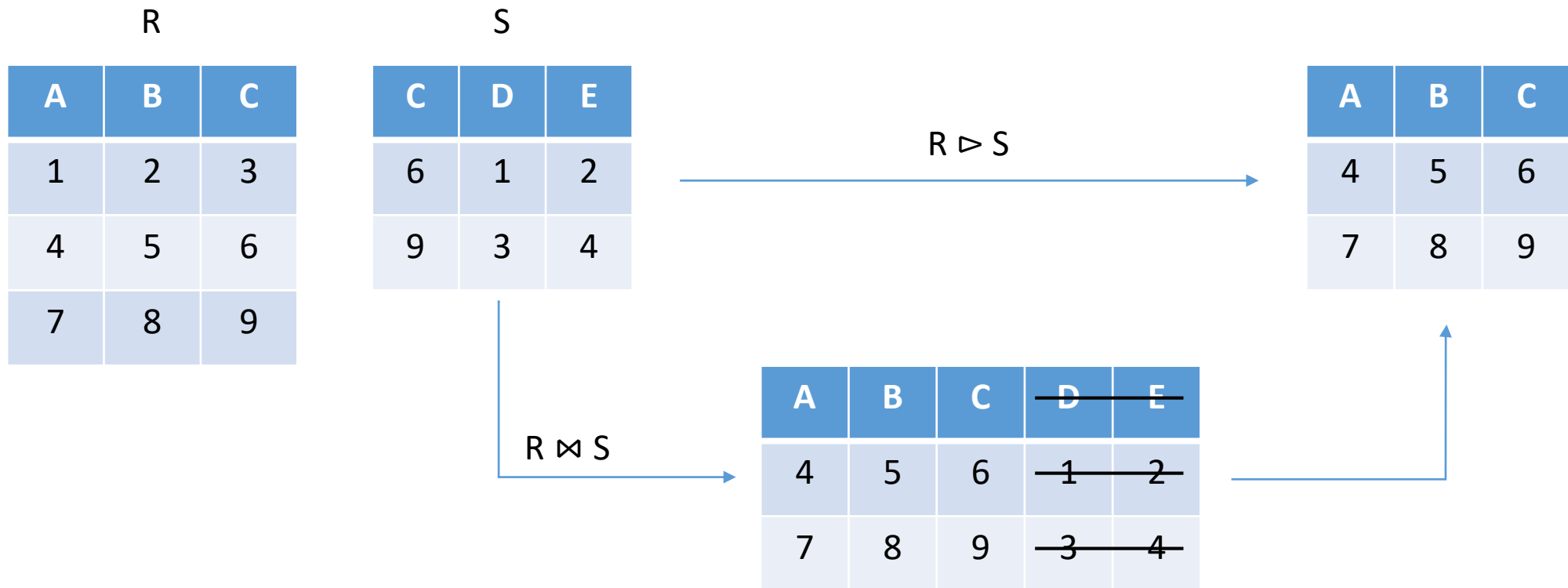
## Semijoin, semi- $\theta$ -join

$$R \triangleright S, \quad R \triangleright_{\theta} S$$

- Semijoin of relations R and S is equal to natural join where we keep only the attributes of the left relation R.

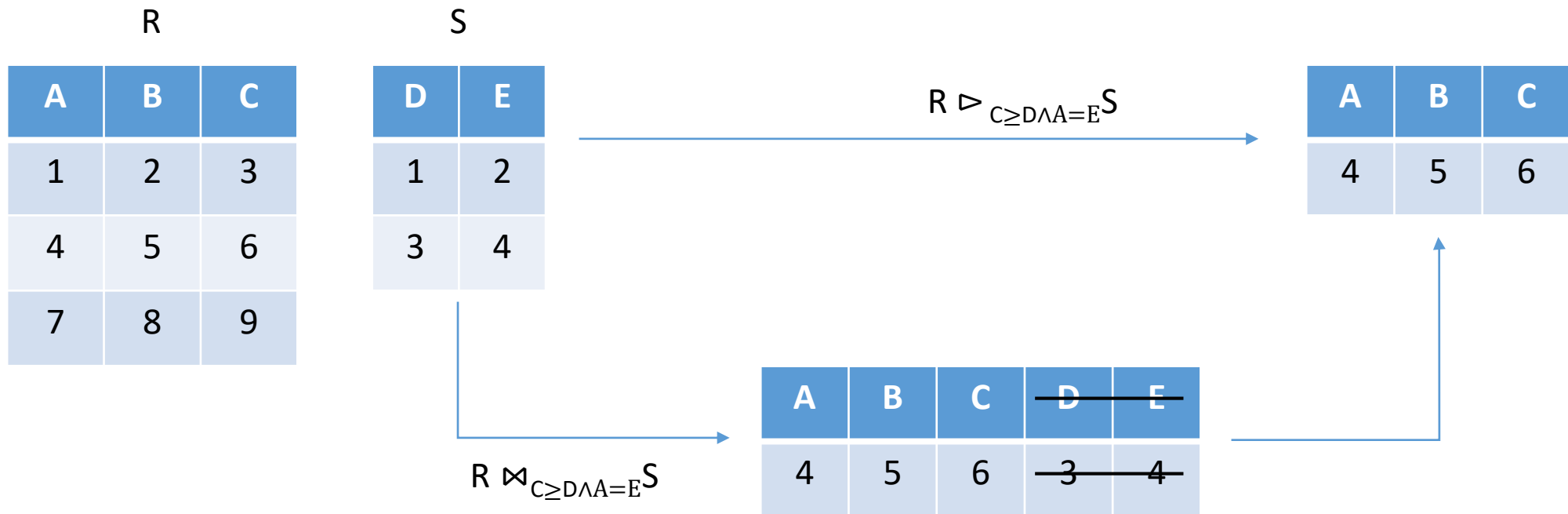
# Other operations

## Semijoin (example)



# Other operations

## Semi- $\theta$ -join (example)





# Other operations

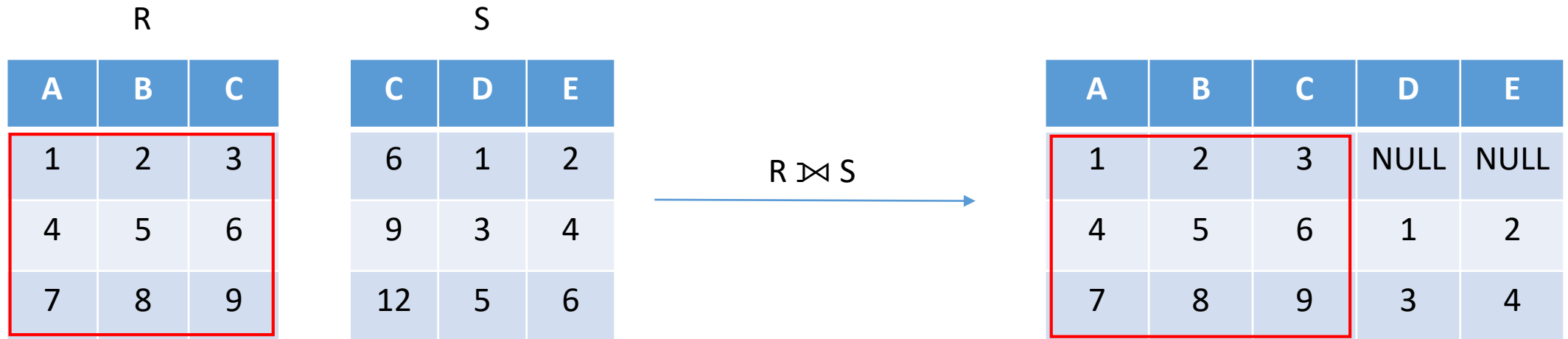
## Outer joins (left, right and full)

$$R \bowtie S, R \ltimes S, R \rtimes S$$

- Left outer join of R and S returns a natural join where tuples of R having no matching values in common attributes of S are also included in the result. Unknown values of the attributes are set to NULL.
  - Similarly goes for right outer join and full outer join.

# Other operations

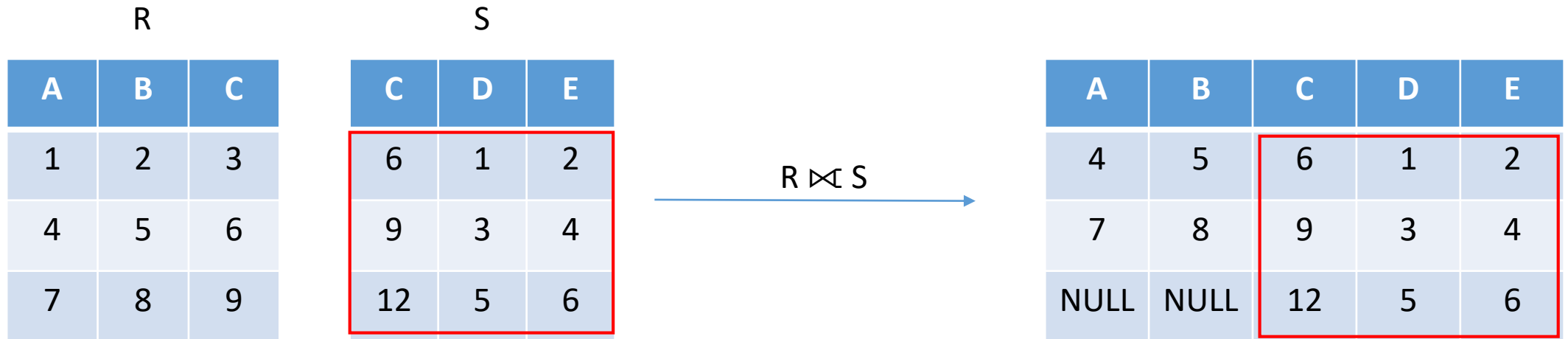
## Left outer join (example)



- All tuples of R are included in the result.

# Other operations

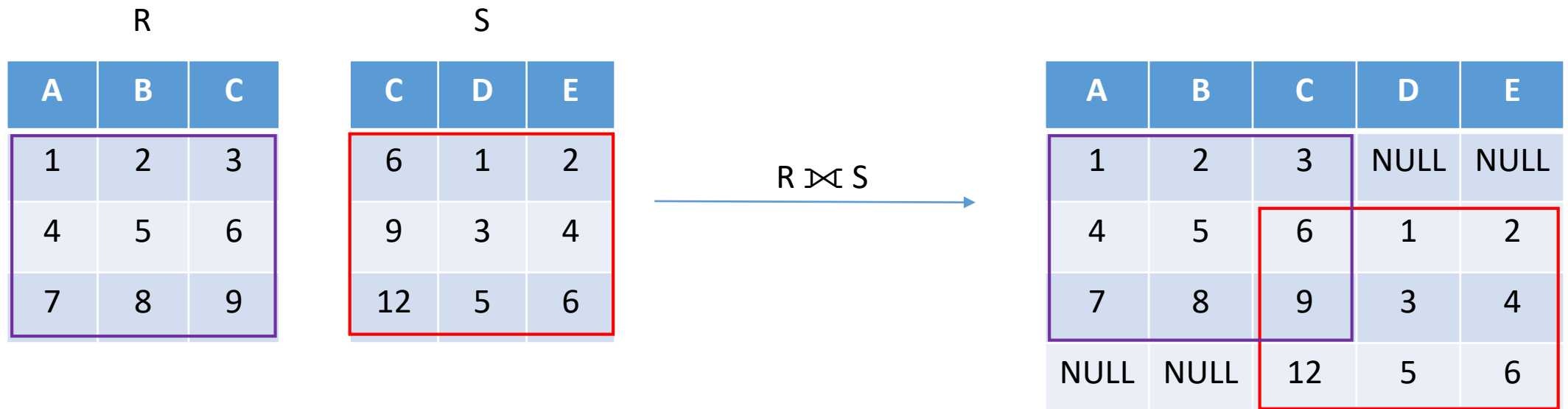
## Right outer join (example)



- All tuples of S are included in the result.

# Other operations

## Full outer join (example)



- All tuples from R and S are included in the result.

# Other operations

## Aggregate

$$\tau_{AS}(R)$$

- Applies aggregate function list AS to the relation R.
- Aggregation functions are:
  - COUNT – counts all non-NULL values
  - SUM – sums the values
  - AVG – computes the average
  - MIN – finds the lowest number
  - MAX – finds the highest number

# Other operations

## Aggregate (example)

R

A	B	C	D
1	a	100	1
1	a	200	1
1	b	200	1
2	a	100	1
2	b	100	1
3	a	100	1
3	NULL	100	1
4	b	100	1
4	b	200	1
4	NULL	200	1

$\tau_{\text{MAX A, COUNT B, SUM C}}(R)$

...	...	...
4	8	1400

# Other operations

## Grouping

$$\mathbf{GA}\tau_{AS}(\mathbf{R})$$

- Aggregation with grouping of relation R uses a list of aggregation functions AS upon groups of relation R which are determined with a list of grouping attributes.

# Other operations

## Grouping (example)

R

A	B	C	D
1	a	100	1
1	a	200	1
1	b	200	1
2	a	100	1
2	b	100	1
3	a	100	1
3	NULL	100	1
4	b	100	1
4	b	200	1
4	NULL	200	1

$A, D \tau_{\text{COUNT } B, \text{SUM } C}(R)$

A	D	...	...
1	1	3	500
2	1	2	200
3	1	1	200
4	1	2	500



# List of operations and their predicates

- Projection =  $\pi_S(R)$
- Selection =  $\sigma_{\text{predicate}}(R)$
- Renaming =  $\rho_{s(S)}(R)$
- Union =  $R \cup S$
- Set-difference =  $R - S$
- Intersection =  $R \cap S$
- Cartesian product =  $R \times S$
- Theta join =  $R \bowtie_{\theta} S$
- Equistick =  $R \bowtie_{\theta=} S$
- Natural join =  $R \bowtie S$
- Division =  $R / S$
- Semijoin =  $R \bowtie_{\supset} S$
- Semi- $\theta$ -join =  $R \bowtie_{\theta \supset} S$
- Left outer join =  $R \bowtie_{\supseteq} S$
- Right outer join =  $R \bowtie_{\supseteq} S$
- Full outer join =  $R \bowtie_{\supseteq} S$
- Aggregate =  $\tau_{AS}(R)$
- Grouping =  $GA\tau_{AS}(R)$

# Priority of operations

## 1. **Simple operations:**

projection, selection, renaming

## 2. **Product operations:**

cartesian product,  $\theta$ -join, natural join, division

## 3. **Set operations:**

intersection, union, set-difference

## 4. **Other operations:**

aggregate (with grouping)

# Relational algebra exercises

# Exercise 1 - Operator

Write the following queries in a relational algebra.

1. Which customers are buying telephones at Telekom?
2. Which operators are selling Janez's preferred phone?
3. Which customers are buying from all operators?  
Assume that all the operators are listed in relation p.
4. Which operators are selling all Janez's preferred telephones? Assume that relation n contains more tuples which correspond to Janez.
5. Which customers are buying from one operator only?

**p (sale)**

Operator	Telephone
Telekom	Nokia
Telekom	Siemens
T2	Nokia
T2	Samsung
A1	Nokia
A1	Siemens
A1	Samsung

**k (buying)**

Customer	Operator
Marko	Telekom
Marko	A1
Meta	Telekom
Meta	T2
Meta	A1
Janez	A1
Petra	Telekom

**n (prefers)**

Customer	Telephone
Marko	Siemens
Meta	Nokia
Janez	Siemens
Petra	Nokia

## Exercise 2 - GSM

Relation	Relation schema
s customer	CUSTOMER ( <u>SID</u> , SName, SSurname, SAge, SCity)
p seller	SELLER ( <u>PID</u> , PName, PSurname, PAge, PDiscount)
g mobile phone	GSM ( <u>GID</u> , GType, GPrice)
k purchase	PURCHASE ( <u>#SID</u> , <u>#PID</u> , <u>#GID</u> , KDate, KPieces)

Write the following queries in a relational algebra.

1. Find names and surnames of customers who are from Kranj and are older than 18 years.
2. Find names and surnames of customers who have bought something.
3. Find names and surnames of customers who have never bought anything. Assume that the customers are stored in a relation CUSTOMER.
4. Find names and surnames of the sellers who have sold Nokia mobile phones.
5. For each type of mobile phone find the total number of sold pieces.

# Exercise 3 - Hotel

Relation	Relation schema
g guest	GUEST ( <u>GNo</u> , GName, GAddress)
h hotel	HOTEL ( <u>HNo</u> , HName, HCity)
r room	ROOM ( <u>RNo</u> , #HNo, RType, RPrice)
b booking	BOOKING ( <u>#HNo</u> , <u>#RNo</u> , <u>#GNo</u> , <u>BFrom</u> , BTo)

Write the following queries in a relational algebra.

1. Find the numbers of all single-bed rooms (Rtype = 1), where the price is lower than 80€ per day.
2. Find the numbers, prices and room types in hotel Bernardin.
3. Find names of the guests which are currently in hotel Bernardin (value today). Show also the prices and types of the rooms they are situated in.
4. For each hotel list its name, number of all rooms and an average room price.
5. List all the room data of hotel Bernardin (RNo, RType and RPrice), including the name of the guest in the room if it is currently occupied (otherwise value NULL).