Database

Theory

Relational Algebra

Preliminary

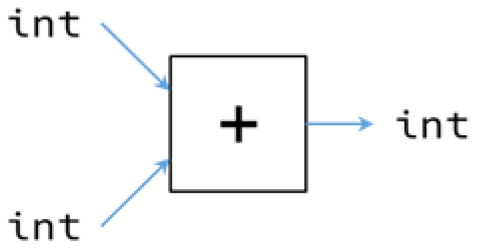
Algebra

Definition

Mathematical Algebra

An **algebra** is a mathematical system consisting of

- Operands   variables or values from which new values can be constructed.
- Operators   symbols denoting procedures that construct new values from the given values.



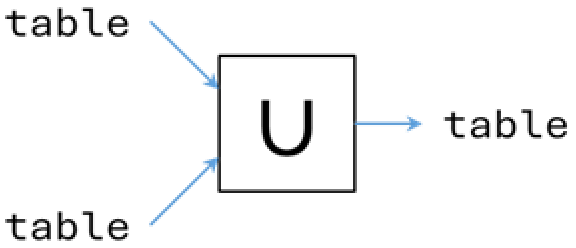
Algebra

Relational

Relational Algebra

A **relational algebra** is an algebra system for SQL queries consisting of

- Operands   relations/tables.
- Operators   transformations from one or more input relations into one output relations.



Note

Relational algebra is an **imperative** query language. It forms the mathematical foundation of relational database engines and are used to specify how data can be retrieved.

Relational algebra is essential to understanding how database queries are **procesed** and **optimized**.

Algebra

Operators

Relational Algebra Operators

The main **algebraic operators** are the following.

Unary Operation	Symbol	Set Operation	Symbol	Binary Operation	Symbol
Selection	σ	Union	$\cup$	Cross Product	$\times$
Projection	π	Intersection	$\cap$	Inner/Natural Join	$\bowtie$
Renaming	ρ	Set Difference	$-$	Outer Join	⟗ ⟕ ⟖

Other algebraic system adds more operators but we will not use them.

Relational algebra is based on relations. In turn, relation is based on set. So, there is no duplicate row in relational algebra. This is ***slightly different*** from SQL.

Preliminary

Operations

Logical

Propositional Logic

The model semantics of **propositional logic** is defined by the truth tables of connectives: conjunction ($\land$), disjunction ($\lor$), and negation ($\neg$)*.

p	q	$\neg p$	$\neg q$	$p \land q$	$p \lor q$	$p \land \neg q$	$\neg(p \land \neg q)$	$\neg p \lor q$	$p \rightarrow q$
True	True	False	False	True	True	False	True	True	True
False	True	True	False	False	True	False	True	True	True
True	False	False	True	False	True	True	False	False	False
False	False	True	True	False	False	False	True	True	True

Preliminary

Operations

Logical

Propositional Logic

The model semantics of **propositional logic** is defined by the truth tables of connectives: conjunction ($\wedge$), disjunction ($\vee$), and negation ($\neg$)*.

p	q	$\neg p$	$\neg q$	$p \wedge q$	$p \vee q$	$p \wedge \neg q$	$\neg(p \wedge \neg q)$	$\neg p \vee q$	$p \rightarrow q$
True	True	False	False	True	True	False	True	True	True
False	True	True	False	False	True	False	True	True	True
True	False	False	True	False	True	True	False	False	False
False	False	True	True	False	False	False	True	True	True

Equivalence

Two propositional formulae are equivalent if and only if they have the same truth table.

Preliminary

Operations

Relational

Relational Operator

We use the standard relational operators: equal to (=), not equal to (≠), less/greater than (</>), and less/greater than or equal to (≤/≥)*.

Note

The meaning of the operation follows the usual meaning. For **TEXT**, the S1 < S2 means that S1 is *lexicographically* smaller than S2.

To avoid issues with precedence, add parentheses as necessary.

*Without NULL value, we do not have to worry about the problem with = and ≠ operators.

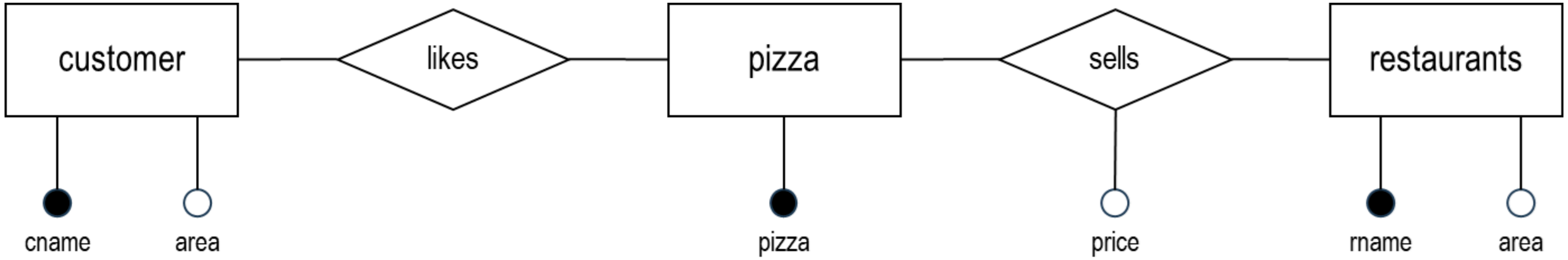
Preliminary

Sample Data

Schema

Pizza

pizza(__pizza: TEXT__)
customer(__cname: TEXT__, area: TEXT)
restaurant(__rname: TEXT__, area: TEXT)
likes(__cname: TEXT, pizza: TEXT__)
sells(__rname: TEXT, pizza: TEXT__, price: INTEGER)



Relational Algebra

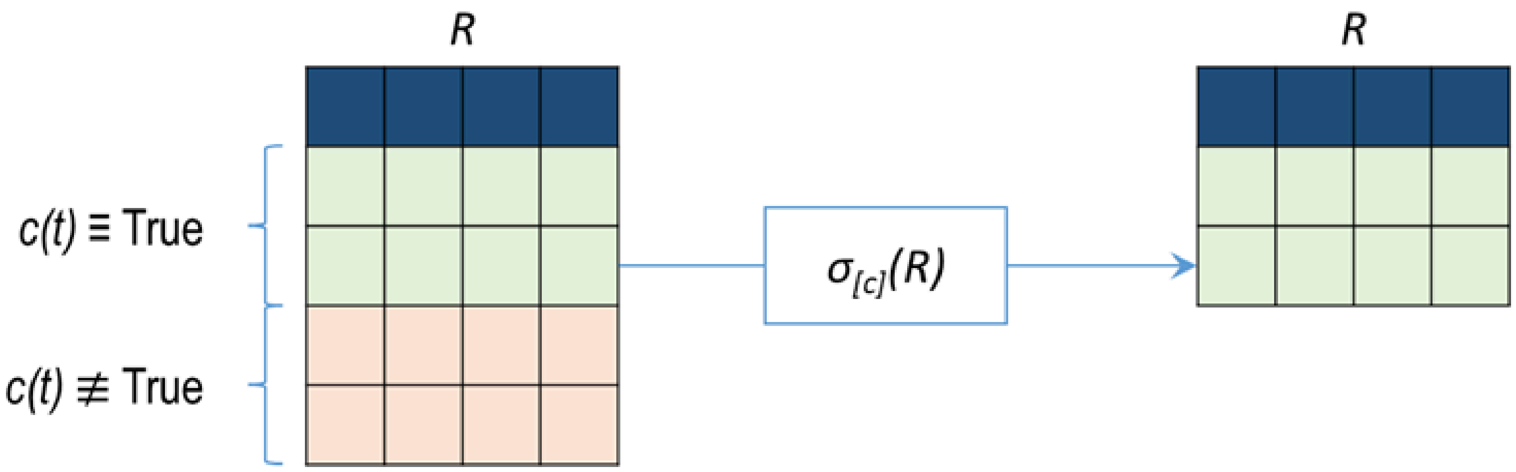
Selection

Operator

Selection Operator

$\sigma_{[c]}(R)$ **selects** all rows from the relation R that satisfies the selection condition ***c****.

Visualization



*This is similar to `WHERE` clause in SQL.

Selection

Examples

Example #1

Find the name (*rname*) and area of the different restaurants in London.

Code

```
σ[area = 'London'](restaurant)
```

```
SELECT *
FROM restaurant r
WHERE r.area = 'London';
```

Result

rname	area
Spice Palace	London
London Seafood Shack	London
Thames River Tavern	London

3 rows

Relational Algebra

Selection

Examples

Example #2

Find the name (*rname*), pizza, and price of the different restaurants that (a) sells Veggie cheaper than 14 or (b) is named Sizzle Grill.

Code

```
σ[(pizza = 'Veggie' ∧ price < 14)
  ∨ (rname = 'Sizzle Grill')](sells)
```

```sql
SELECT *
FROM sells s
WHERE (s.pizza = 'Veggie'
       AND s.price < 14)
   OR (s.rname = 'Sizzle Grill')
```

Result

rname	pizza	area
Bella Italia	Veggie	11
Spice Palace	Veggie	13
Sizzle Grill	BBQ Chicken	13
...		

6 rows

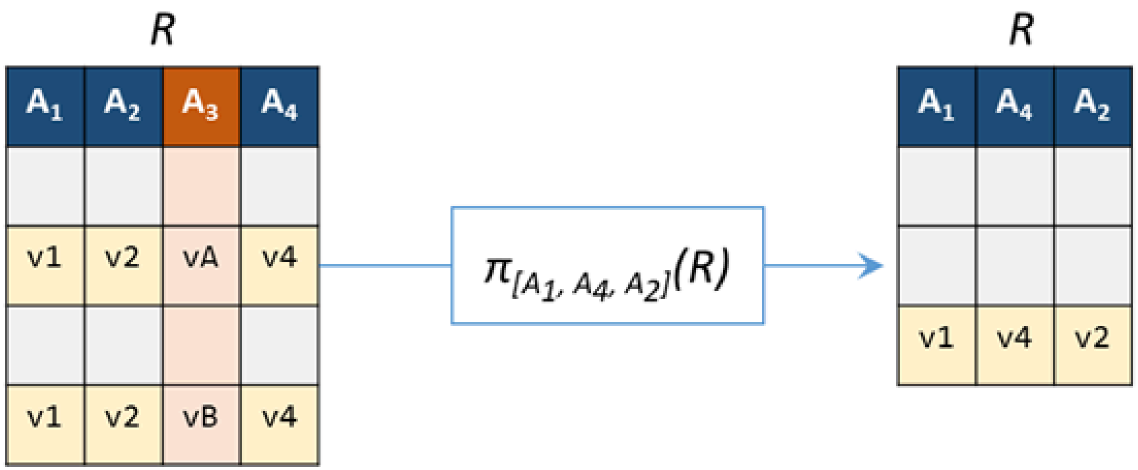
Projection

Operator

Projection Operator

$\pi_{[l]}(R)$ **keeps** only the columns specified in the ordered list $\boldsymbol{l}$ and in the same order*.

Visualization



*This is similar to SELECT clause in SQL.

Projection

Examples

Example #1

Find the different name (*cname*) of customers that likes at least one pizza.

Code

```
π[cname](likes)
```

```sql
SELECT DISTINCT l.cname
FROM likes l;
```

Result

cname
Alice
Bob
Emily
...

10 rows

Projection

Examples

Example #2

Find the name (*rname*) of the different restaurants that (a) sells Veggie cheaper than 14 or (b) is named Sizzle Grill.

Code

```
π[rname](
  σ[(pizza = 'Veggie' ∧ price < 14)
    ∨ (rname = 'Sizzle Grill')](sells))
```

```
SELECT DISTINCT s.rname
FROM sells s
WHERE (s.pizza = 'Veggie'
       AND s.price < 14)
   OR (s.rname = 'Sizzle Grill')
```

Result

rname
Bella Italia
Spice Palace
Sizzle Grill

3 rows

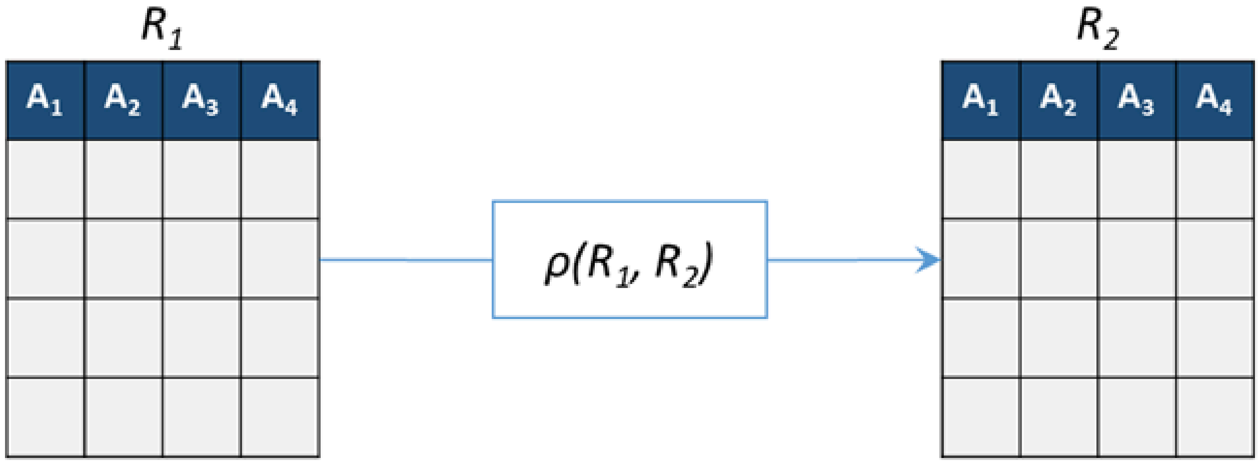
Renaming

Operator

Renaming Operator

$\rho(R_1, R_2)$ can be used to **rename** the relation*.

Visualization



*This do not create a new relation in the hard disk, but we can simply refer to this table as R_2.

Relational Algebra

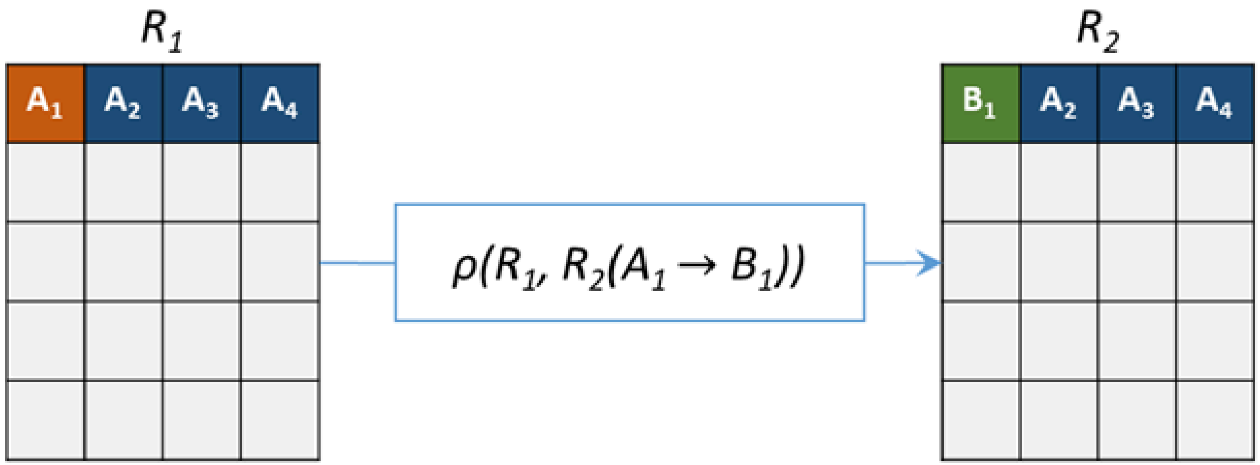
Renaming

Operator

Renaming Operator

$\rho(R_1, R_2(A_1 \rightarrow A_2))$ can be used to **rename** the relation and some of its attributes*.

Visualization



*$A_i \rightarrow B_i$ renames attribute A_i into B_i similar to **AS** keyword.

Renaming

Why Renaming?

SELECT-FROM-WHERE + Dot Notation

For good practice, we require all tables to always be renamed in SQL. This notation is now also available in relational algebra.

We also allow dot notation `r.attr` to simplify the writing.

```
SELECT DISTINCT r.attr
FROM rel r
WHERE c;
```

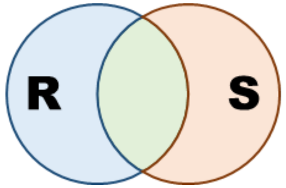
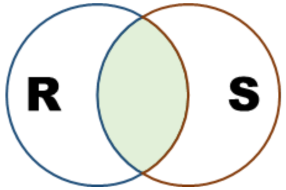
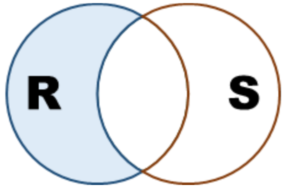
```
π[r.attr](
  σ[c](
    ρ(rel, r)))
```

Note

Other source of relational algebra *(even from past semesters)* may use different convention and/or notation. Our notation is chosen to simplify reading and writing by adhering closer to good SQL notation.

Set

Operators

Operation	Visualization	SQL
$R \cup S$		SELECT * FROM R UNION SELECT * FROM S
$R \cap S$		SELECT * FROM R INTERSECT SELECT * FROM S
$R - S$		SELECT * FROM R EXCEPT SELECT * FROM S

Note

The two relations must be **union-compatible** *(basically, they must have the same column types)*.

Relational Algebra

Set

Examples

Note

We also add := similar to an **assignment** to break up complex queries to simpler queries. The **temporary** relation can be used for subsequent algebraic operation.

Example #1

Find the different pizza sold by both Bella Italia and Desert Diner.

Code

```
Q1 := π[pizza](σ[rname = 'Bella Italia'](sells))
Q2 := π[pizza](σ[rname = 'Desert Diner'](sells))
Q1 ∩ Q2
```

```sql
SELECT s.pizza FROM sells s
WHERE s.rname = 'Bella Italia'
INTERSECT
SELECT s.pizza FROM sells s
WHERE s.rname = 'Desert Diner';
```

Result

pizza
Margherita
Hawaiian
BBQ Chicken
Mushroom

4 rows

Set

Examples

Note

Recap that UNION, INTERSECT, and EXCEPT automatically removes duplicates. So there is no need for the DISTINCT keyword.

Example #2

Find the different pizza sold by Bella Italia but not Desert Diner.

Code

```
Q1 := π[pizza](σ[rname = 'Bella Italia'](sells))
Q2 := π[pizza](σ[rname = 'Desert Diner'](sells))
Q1 – Q2
```

```
SELECT s.pizza FROM sells s
WHERE s.rname = 'Bella Italia'
EXCEPT
SELECT s.pizza FROM sells s
WHERE s.rname = 'Desert Diner';
```

Result

pizza
Veggie
Pepperoni
Four Cheese

3 rows

Product

Operator

Cross Product

$R_1 \times R_2$ combine each row of R_1 with each row of R_2 and keep the n columns of R_1 and the m columns of R_2.

R_1

a	b
1	2
3	4

R_2

c	d	e
A	B	C
D	E	F
G	H	I

$R_1 \times R_2$

a	b	c	d	e
1	2	A	B	C
1	2	D	E	F
1	2	G	H	I
3	4	A	B	C
3	4	D	E	F
3	4	G	H	I

Product

Examples

Example #1

Find all the different pairs of customer name and restaurant name such that they are in the same area.

SQL

```
SELECT c.cname, r.rname
FROM customer c, restaurant r
WHERE c.area = r.area;
```

Question

Is there a need for the `DISTINCT` keyword here?

Result

cname	rname
Alice	Bella Italia
Alice	Big Apple Bistro
Alice	Down Under Delights
...	

28 rows

Product

Examples

Example #1

Find all the different pairs of customer name and restaurant name such that they are in the same area.

SQL

```
SELECT c.cname, r.rname
FROM customer c, restaurant r
WHERE c.area = r.area;
```

```
π[c.cname, r.rname](
  σ[c.area = r.area](
    ρ(customer, c) x ρ(restaurant, r)
))
```

Result

cname	rname
Alice	Bella Italia
Alice	Big Apple Bistro
Alice	Down Under Delights
...	

28 rows

Relational Algebra

Operation	Symbol	Operation	Symbol	Operation	Symbol	Operation	Symbol
Selection	σ	Projection	π	Renaming	ρ	Cross Product	×
Conjunction	∧	Disjunction	∨	Negation	¬		
Union	∪	Intersection	∩	Difference	−		

Product

Examples

Example #2

Find all the different restaurant name (*rname*), pizza, and the price of the pizza sold by restaurants in London.

Relational Algebra

```
SELECT r.rname, s.pizza, s.price
FROM restaurant r, sells s
WHERE r.rname = s.rname
  AND r.area = 'London';
```

Result

rname	pizza	price
Spice Palace	Veggie	13
Spice Palace	Mushroom	14
Spice Palace	Supreme	16
...		

12 rows

Join

Operator

Note

The two versions are ***almost equivalent***. The only exception is when the condition `c1` uses attributes from `r3`.

Join

$R_1 \bowtie_{[c]} R_2$ is simply **defined** as $\sigma_{[c]}(R_1 \times R_2)$. In other words, we include only tuples that **satisfies the condition** $\boldsymbol{c}$ after the cross product.

SELECT-FROM-WHERE

```
SELECT DISTINCT a1, a2, a3, ...
FROM r1, r2, r3, ...
WHERE c; -- c = c1 AND c2
```

```
π[a1, a2, a3, ...](
  σ[c](r1 x r2 x r3 x ...)
) # c = c1 ∧ c2
```

Inner Join

```
SELECT DISTINCT a1, a2, a3, ...
FROM r1 JOIN r2 ON c1
        JOIN r3 ON c2;
```

```
π[a1, a2, a3, ...](
  r1 ⋈[c1] r2 ⋈[c2] r3
) # (r1 ⋈[c1] r2) ⋈[c2] r3
```

Join

Operator

Join

$R_1 \bowtie_{[c]} R_2$ is simply **defined** as $\sigma_{[c]}(R_1 \times R_2)$. In other words, we include only tuples that **satisfies the condition *c*** after the cross product.

Variants

Natural Join

If we exclude the condition ***c***, the operator becomes the **natural join operator** *(i.e., $\bowtie$)*. For example, $R_1 \bowtie R_2$.

Outer Join

We also have **left** ($⟕_{[c]}$), **right** ($⟖_{[c]}$), and **full** ($⟗_{[c]}$) **outer join** variants that depends on the condition ***c***.

Natural

There is also **natural** variant by omitting ***c***.

Join

Examples

Example #1

Find all the different pairs of customer name and restaurant name such that they are in the same area.

Relational Algebra

```
π[c.cname, r.rname](
  ρ(customer, c)
    ⋈[c.area = r.area]
  ρ(restaurant, r)
)
```

Result

cname	rname
Alice	Bella Italia
Alice	Big Apple Bistro
Alice	Down Under Delights
...	

28 rows

Relational Algebra

Operation	Symbol	Operation	Symbol	Operation	Symbol	Operation	Symbol
Selection	σ	Projection	π	Renaming	ρ	Cross Product	$\times$
Conjunction	$\wedge$	Disjunction	$\vee$	Negation	$\neg$	Join	$\bowtie$
Union	$\cup$	Intersection	$\cap$	Difference	$-$	Outer Joins	⟗ ⟕ ⟖

Join

Examples

Example #2

Find all the different restaurant name (*rname*), pizza, and the price of the pizza sold by restaurants in London.

Relational Algebra

```
SELECT r.rname, s.pizza, s.price
FROM restaurant r, sells s
WHERE r.rname = s.rname
  AND r.area = 'London';
```

Result

rname	pizza	price
Spice Palace	Veggie	13
Spice Palace	Mushroom	14
Spice Palace	Supreme	16
...		

12 rows

Break

Practical Algebra

Operation	Symbol	Operation	Symbol	Operation	Symbol	Operation	Symbol
Selection	σ	Projection	π	Renaming	ρ	Cross Product	$\times$
Conjunction	$\wedge$	Disjunction	$\vee$	Negation	$\neg$	Join	$\bowtie$
Union	$\cup$	Intersection	$\cap$	Difference	$-$	Outer Joins	⟗ ⟕ ⟖

Complex

Simplification

Question

Find all the different pairs of customer name (*cname*) and pizza that the customer likes such that the price of the pizza is more than 15.

SQL

Relational Algebra

Practical Algebra

Operation	Symbol	Operation	Symbol	Operation	Symbol	Operation	Symbol
Selection	σ	Projection	π	Renaming	ρ	Cross Product	$\times$
Conjunction	$\wedge$	Disjunction	$\vee$	Negation	$\neg$	Join	$\bowtie$
Union	$\cup$	Intersection	$\cap$	Difference	$-$	Outer Joins	⟗ ⟖ ⟕

- **Complex**
  - *Simplification*
  - *Chaining*
  - *Universal*

Complex

Chaining

Question

Find the different pair of customer name (*cname*) and pizza the customer like such that the customer is in London and the pizza cost is less than 20.

SQL

Relational Algebra

Practical Algebra

Operation	Symbol	Operation	Symbol	Operation	Symbol	Operation	Symbol
Selection	σ	Projection	π	Renaming	ρ	Cross Product	$\times$
Conjunction	$\wedge$	Disjunction	$\vee$	Negation	$\neg$	Join	$\bowtie$
Union	$\cup$	Intersection	$\cap$	Difference	$-$	Outer Joins	⟕ ⟗ ⟖

Complex

Universal

Question

Find the different customers that likes all the pizza. Include pizza that is not sold by any restaurant.

SQL

Relational Algebra

```
postgres=# exit
Press any key to continue . . .
```