SQL

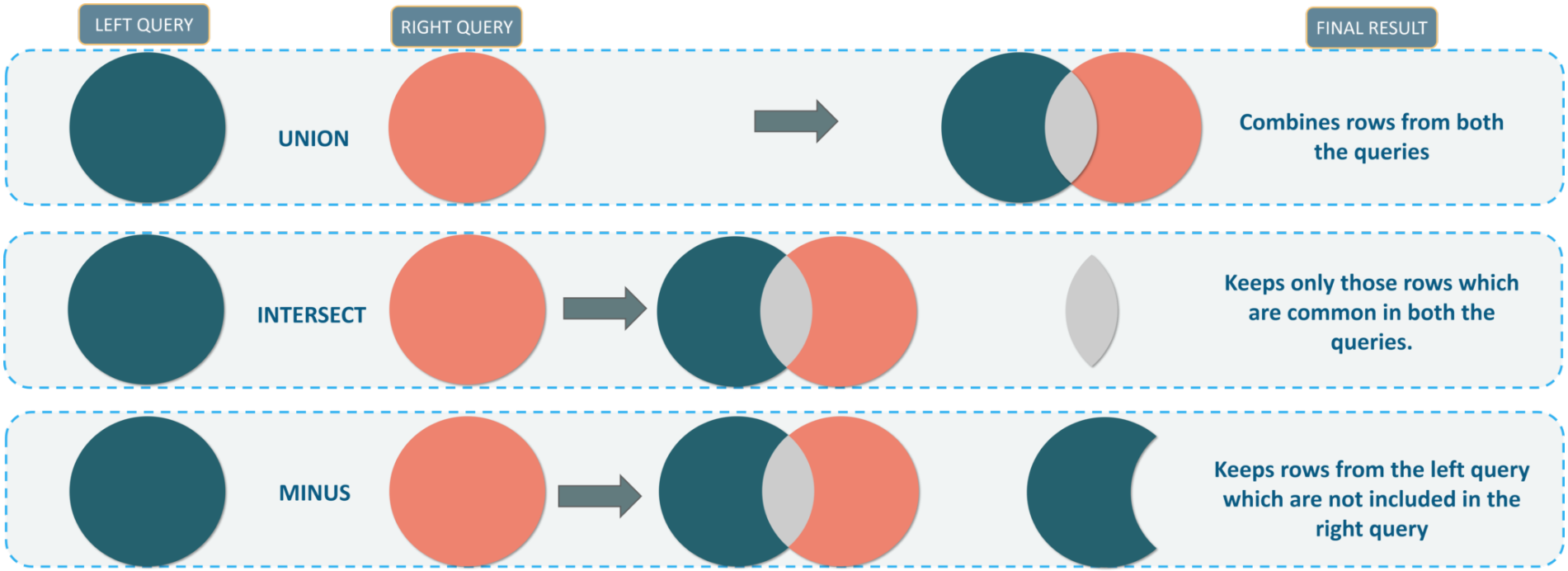
Sets and aggregate functions

dr Szymon Murawski

Comarch SA

October 22, 2019

Set operations



Union

Actors

actor_id	first_name	last_name
1	Sam	Elliot
2	Will	Smith
3	Jodie	Foster

Clients

client_id	first_name	last_name
1	John	Smith
2	Maurice	Turnbull
3	Sam	Davis

```
SELECT first_name FROM actors
UNION
SELECT first_name FROM clients;

first_name
---------------
Sam
Will
Jodie
John
Maurice
(5 rows)
```

Union

Actors

actor_id	first_name	last_name
1	Sam	Elliot
2	Will	Smith
3	Jodie	Foster

Movies

movie_id	title	year	price
1	Anchorman	2004	10
2	Ghostbusters	1984	5.50
3	Terminator	1984	8.50

```
SELECT first_name FROM actors
UNION
SELECT title FROM movies;

first_name
--------------
Sam
Will
Jodie
Anchorman
Ghostbusters
Terminator
(6 rows)
```

- Name of columns in two selects does not matter
- Type of columns must be the same
- Number of columns in each select must be the same

Union All

Actors

actor_id	first_name	last_name
1	Sam	Elliot
2	Will	Smith
3	Jodie	Foster

Clients

client_id	first_name	last_name
1	John	Smith
2	Maurice	Turnbull
3	Sam	Davis

```
SELECT first_name FROM actors
UNION ALL
SELECT first_name FROM clients;

first_name
---------------
Sam
Will
Jodie
John
Maurice
Sam
(6 rows)
```

- Adding keyword ALL prevents removing duplicates

Intersect

Actors

actor_id	first_name	last_name
1	Sam	Elliot
2	Will	Smith
3	Jodie	Foster

Clients

client_id	first_name	last_name
1	John	Smith
2	Maurice	Turnbull
3	Sam	Davis

```
SELECT first_name FROM actors
EXCEPT
SELECT first_name FROM clients;

first_name
--------------
Sam
(1 row)
```

Except

Actors

actor_id	first_name	last_name
1	Sam	Elliot
2	Will	Smith
3	Jodie	Foster

Clients

client_id	first_name	last_name
1	John	Smith
2	Maurice	Turnbull
3	Sam	Davis

```
SELECT first_name FROM actors
EXCEPT
SELECT first_name FROM clients;

first_name
---------------
Will
Jodie
(2 rows)
```

Aggregate functions

- SQL provides several aggregation functions, that allow for performing calculations on a set of rows, returning a single value
- Aggregate functions:
  - **AVG()** - returns average
  - **COUNT()** - returns count of rows
  - **MIN()** - returns minimum value from the set
  - **MAX()** - returns maximum value from the set
  - **SUM()** - returns sum of values

Simple aggregate function

Employees

employee_id	first_name	last_name	city	salary
1	John	Smith	New York	150
2	Ben	Johnson	New York	250
3	Louis	Armstrong	New Orleans	75
4	John	Lennon	London	300
5	Peter	Gabriel	London	100

```
SELECT AVG(salary)
FROM employees;

avg
---------------
175
(1 row)
```

Selection in aggregate functions

Employees

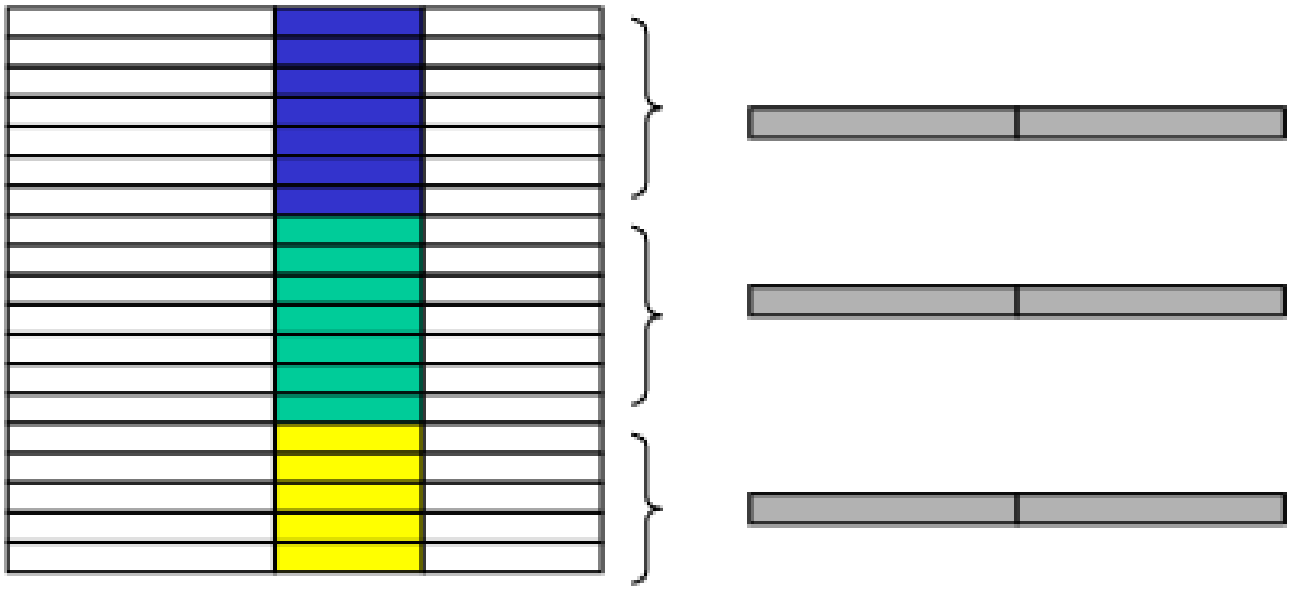
employee_id	first_name	last_name	city	salary
1	John	Smith	New York	150
2	Ben	Johnson	New York	250
3	Louis	Armstrong	New Orleans	75
4	John	Lennon	London	300
5	Peter	Gabriel	London	100

```
SELECT SUM(salary)
FROM employees
WHERE first_name = 'John';

sum
---------------
450
(1 row)
```

Grouping

- `GROUP BY` clause is used to group rows returned by `SELECT` statement
- On this groups aggregate functions can be performed
- In the result there are as many rows as there are groups
- `HAVING` is like where, but for groups



Simple GROUP BY

Employees

employee_id	first_name	last_name	city	salary
1	John	Smith	New York	150
2	Ben	Johnson	New York	250
3	Louis	Armstrong	New Orleans	75
4	John	Lennon	London	300
5	Peter	Gabriel	London	100

```
SELECT city, AVG(salary)
FROM employees
GROUP BY city;

city          | avg
--------------+----
New York      | 200
New Orleans   | 75
London        | 200
(3 rows)
```

HAVING clause

Employees

employee_id	first_name	last_name	city	salary
1	John	Smith	New York	150
2	Ben	Johnson	New York	250
3	Louis	Armstrong	New Orleans	75
4	John	Lennon	London	300
5	Peter	Gabriel	London	100

```
SELECT city, AVG(salary)
FROM employees
GROUP BY city
HAVING AVG(salary) > 100;

city          | avg
--------------+----
New York      | 200
London        | 200
(2 rows)
```

HAVING and WHERE combined

Employees

employee_id	first_name	last_name	city	salary
1	John	Smith	New York	150
2	Ben	Johnson	New York	250
3	Louis	Armstrong	New Orleans	75
4	John	Lennon	London	300
5	Peter	Gabriel	London	100

```
SELECT city, AVG(salary)
FROM employees
WHERE first_name != 'John'
GROUP BY city
HAVING AVG(salary) > 100;

city          | avg
--------------+----
New York      | 250
(1 row)
```