Databázové systémy

SQL – Window functions

Scores

- Tabuľka s bodmi pre jednotlivých študentov
  - id, name, score
- Chceme ku každému doplniť rozdiel voči priemeru

Demo data

SELECT * FROM scores

ORDER BY score DESC;

	id integer	name character varying(50)	score integer
1	3	Clarence Mcdonald	76
2	4	Gary Gardner	71
3	10	Eugene Gardner	69
4	2	Laura Ryan	63
5	1	Sara Alvarez	62
6	7	William Peterson	62
7	5	Aaron Williamson	61
8	6	Roger Martin	60
9	9	Ashley Watkins	58
10	8	Beverly Hamilton	57

Obvious riešenie

```
SELECT s.name, s.score, s.score - (SELECT avg(score) FROM scores)

FROM scores s

ORDER BY 2 DESC;
```

	name character varying(50)	score integer	?column? numeric
1	Clarence Mcdonald	76	12.100000000000000
2	Gary Gardner	71	7.100000000000000
3	Eugene Gardner	69	5.100000000000000
4	Laura Ryan	63	-0.900000000000000
5	Sara Alvarez	62	-1.900000000000000
6	William Peterson	62	-1.900000000000000
7	Aaron Williamson	61	-2.900000000000000
8	Roger Martin	60	-3.900000000000000
9	Ashley Watkins	58	-5.900000000000000
10	Beverly Hamilton	57	-6.900000000000000

Riešenie cez Window Functions

```
SELECT s.name, s.score,
s.score - avg(score) OVER ()
FROM scores s
ORDER BY 2 DESC;
```

Scores 2

- Tabuľka s bodmi pre jednotlivých študentov
  - id, name, study_programme, score
- Chceme ich zoradiť a ku každému dopísať jeho poradie
  - Ak majú dvaja rovnaký počet bodov, tak nech majú rovnakú pozíciu

Riešenie so subselect

```
SELECT s1.name, s1.score as score,
(
  SELECT count(DISTINCT s2.score)
  FROM scores s2 WHERE s2.score >= s1.score
) AS rank
FROM scores s1
ORDER BY 2 DESC, 1
```

Riešenie cez Window Functions

```
SELECT s.name, s.score,
DENSE_RANK() OVER (ORDER BY s.score
DESC)

FROM scores s

ORDER BY 2 DESC,1
```

	name character varying(50)	study_programme character varying(30)	score integer	dense_rank bigint
1	Clarence Mcdonald	it	76	1
2	Gary Gardner	history	71	2
3	Eugene Gardner	history	69	3
4	Laura Ryan	history	63	4
5	Sara Alvarez	it	62	5
6	William Peterson	it	62	5
7	Aaron Williamson	it	61	6
8	Roger Martin	history	60	7
9	Ashley Watkins	it	58	8
10	Beverly Hamilton	history	57	9

A chceme to po študijných programoch

```
SELECT s.name, s.study_programme, s.score,
DENSE_RANK() OVER (PARTITION BY stu
dy_programme ORDER BY score DESC)
FROM scores s
ORDER BY 2, 3 DESC,1
```

	name character varying(50)	study_programme character varying(30)	score integer	dense_rank bigint
1	Gary Gardner	history	71	1
2	Eugene Gardner	history	69	2
3	Laura Ryan	history	63	3
4	Roger Martin	history	60	4
5	Beverly Hamilton	history	57	5
6	Clarence Mcdonald	it	76	1
7	Sara Alvarez	it	62	2
8	William Peterson	it	62	2
9	Aaron Williamson	it	61	3
10	Ashley Watkins	it	58	4

Príklad “employees”

Napíšte SELECT, ktorý vráti zamestnancov, ktorý poberajú tri najvyššie platy v každom oddelení. Ak poberá jeden z troch top platov oddelenia viacero zamestnancov, nech sú vo výpise všetci. Výpis nech obsahuje (v tomto poradí) názov oddelenia, meno zamestnanca a jeho plat a nech je zoradený podľa mena oddelenia vzostupne, výšky platu zostupne a mena zamestnanca vzostupne.

Riešenie...

```
SELECT d.name as department, e3.name as
employee, e3.salary FROM employees e3
JOIN (SELECT DISTINCT e.department_id, e.salary
FROM employees e WHERE (SELECT COUNT(*)
FROM (SELECT DISTINCT department_id, salary
FROM employees) e2 WHERE e2.department_id =
e.department_id AND e2.salary > e.salary) < 3) tmp ON
tmp.department_id = e3.department_id AND tmp.salary
= e3.salary
JOIN departments d ON e3.department_id = d.id
ORDER BY 1, 3 DESC, 2 ASC
```

Riešenie...

```
SELECT d.name as department, e3.name as employee,
e3.salary FROM employees e3
JOIN (SELECT DISTINCT e.department_id, e.salary
FROM employees e WHERE (SELECT COUNT(*)
FROM (SELECT DISTINCT department_id, salary
FROM employees) e2 WHERE e2.department_id =
e.department_id AND e2.salary > e.salary) < 3) tmp
ON tmp.department_id = e3.department_id AND
tmp.salary = e3.salary
JOIN departments d ON e3.department_id = d.id
ORDER BY 1, 3 DESC, 2 ASC
```

Riešenie...

JOIN(
SELECT DISTINCT e.department_id, e.salary
FROM employees e
WHERE (
SELECT COUNT(*) FROM (SELECT DISTINCT department_id, salary FROM employees) e2
WHERE e2.department_id = e.department_id
AND e2.salary > e.salary) < 3

) tmp ON tmp.department_id = e3.department_id
AND tmp.salary = e3.salary

Riešenie...

```
SELECT DISTINCT e.department_id, e.salary
FROM employees e
WHERE (
SELECT COUNT(*) FROM (
    SELECT DISTINCT department_id, salary
    FROM employees) e2 WHERE
    e2.department_id = e.department_id AND
    e2.salary > e.salary) < 3)
```

Existuje aj riešenie pomocou
ORDER BY + DISTINCT ON komba

“Employees” pomocou window functions

```
SELECT tmp.name, tmp.empl, tmp.salary
FROM (
SELECT d.name, e.name empl, e.salary, DENSE_RANK()
OVER (PARTITION BY d.id ORDER BY e.salary DESC) as
rank
FROM departments d
JOIN employees e ON e.department_id = d.id
) tmp
WHERE tmp.rank < 4
ORDER BY 1,3 DESC, 2 ASC;
```

Window functions

- Výpočet nad sadou riadkov, ktoré súvisia s aktuálnym riadkom
- Agregácia, ktorá vám nezruší spracovanie po riadkoch, neurobí GROUP BY do jednej hodnoty
- Viete si určiť okno (partíciu) a frame okolo aktuálneho riadku
  - A zistiť napr. pozíciu aktuálneho riadku v okne

Syntax

function_name ([expression [, expression ...]]) [FILTER (WHERE filter_clause)] OVER (window_definition)

http://www.postgresql.org/docs/current/static/sql-expressions.html#SYNTAX-WINDOW-FUNCTIONS

Syntax

function_name ([expression [, expression ...]])
[FILTER (WHERE filter_clause)] OVER
(**window_definition**)

http://www.postgresql.org/docs/current/static/sql-expressions.html#SYNTAX-WINDOW-FUNCTIONS

Window definition

[existing_window_name]

[PARTITION BY expression [, ...]]

[ORDER BY expression [ASC | DESC | USING operator]
[NULLS { FIRST | LAST }] [, ...]]

[frame_clause]

http://www.postgresql.org/docs/current/static/sql-expressions.html#SYNTAX-WINDOW-FUNCTIONS

Frame clause

{ RANGE | ROWS } frame_start

{ RANGE | ROWS } BETWEEN frame_start AND frame_end

frame_start a frame_end:

UNBOUNDED PRECEDING

value PRECEDING

CURRENT ROW

value FOLLOWING

UNBOUNDED FOLLOWING

Default frame

The default framing option is
RANGE UNBOUNDED PRECEDING, which is the same as
RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW.

With ORDER BY, this sets the frame to be all rows from the partition start up through the current row's last ORDER BY peer. Without ORDER BY, all rows of the partition are included in the window frame, since all rows become peers of the current row.

Default frame

The default framing option is
RANGE UNBOUNDED PRECEDING, which is the same as
RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW.

With ORDER BY, this sets the frame to be all rows from the partition start up through the current row's last ORDER BY peer. Without ORDER BY, all rows of the partition are included in the window frame, since all rows become peers of the current row.

expressions

- Ľubovoľná agregácia alebo
- Window function podľa

  http://www.postgresql.org/docs/current/static/functions-window.html

Demo

Zhrnutie

- Window functions nám dávajú kontext práve spracovaného riadka
- Running sums, ranking..
- Často sa dá problém vyriešiť aj bez nich
  - Ale s nimi to môže byť efektívnejšie, elegantnejšie, čitatelnejšie