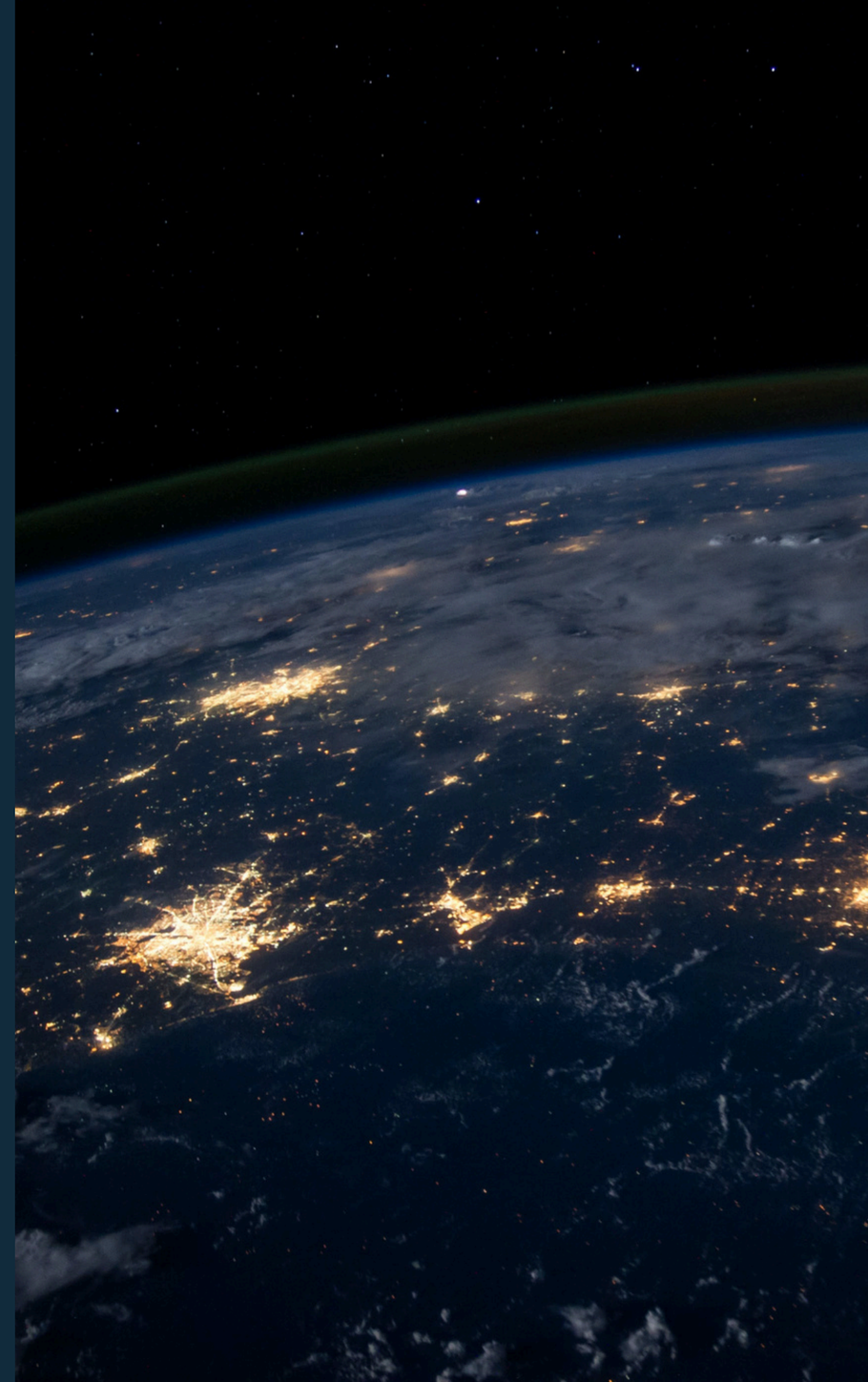
ANALYTICAL FUNCTIONS IN POSTGRESQL: MODERN ALTERNATIVE TO AGGREGATES

P2D2 2025

Jan Suchánek

BAREMON



Jan Suchánek

- More than 25 years of experience in database services, data governance and cloud innovation
- Head of data transformation and co-founder of Baremon

MAIL
jan.suchanek@baremon.eu

X
@jsuchanekcz

WEB
www.baremon.eu

Why did the SQL query love window functions?

Because they always kept things in "order" while staying "partitioned"!

BAREMON

Agenda:

- INTRODUCTION
- HISTORY OF ANALYTICAL FUNCTIONS
- PURPOSE OF ANALYTICAL FUNCTIONS
- WHY ANALYTICAL FUNCTIONS ARE UNDERUTILISED
- THEORETICAL INTRODUCTION TO ANALYTICAL FUNCTIONS
- WHEN TO REPLACE AGGREGATE FUNCTIONS
- PRACTICAL EXAMPLES OF QUERY OPTIMISATION

BAREMON

What are analytical functions?

- Allow calculations across a result set without collapsing rows.
- Enable operations like rankings, running totals, and moving averages.

BAREMON

Why are they important?

- Modern alternative to aggregates.
- Solve complex SQL problems efficiently.
- Key to optimising performance in PostgreSQL.

BAREMON

History of analytical functions

- Oracle 8i in 1999
- SQL-1999 standard (SQL3)

BAREMON

PostgreSQL Milestones

- Version 8.4 (2009): Initial support for window functions.
- Enhanced functionality: Advanced ranking and value-based operations

BAREMON

Purpose of analytical functions

BAREMON

Key use cases:

- **RANKING** (RANK, DENSE_RANK, ROW_NUMBER)
- **COMPARING BY OFFSET** (neighbouring elements and boundaries)
- **AGGREGATION** (sum and average)
- **ROLLING AGGREGATES** (sum and average in dynamics)

BAREMON

Core syntax:

```sql
SELECT
    column1,
    column2,
    window_function(column1) OVER w
FROM
    table_name
WINDOW
    w AS (PARTITION BY column2 ORDER BY column3);
```

BAREMON

Why analytical functions are underutilised

BAREMON

Our goal today:

- Highlight practical examples.
- Show the efficiency of PostgreSQL’s engine.

BAREMON

PostgreSQL Efficiency:

- Engine optimised for window functions.
- Proper indexing further enhances performance.

BAREMON

Theoretical introduction to analytical functions

BAREMON

Key Components of the OVER() Clause:

- PARTITION BY: Group data into subsets.
- ORDER BY: Define sorting within the partition
- Frames: Refine the range of rows for calculations.

Ranking functions:

- row_number() - assigns a unique sequential number to each row
- rank() - assigns a rank to each row with possible gaps
- dense_rank() - assigns a rank to each row
- ntile() - devides rows into n groups and assigns a group number to each row, starting from 1

BAREMON

Offset functions:

- lag(v, n) – value n rows behind
- lead (v, n) – value n rows ahead
- first_value (v) – value from the first frame row
- last_value (v) – value from the last frame row
- nth_value (v, n) – value from the n-th row

BAREMON

Aggregation functions:

- max(v) – maximum partition/frame value
- min(v) – minimum partition/frame value
- avg(v) – average partition/frame value
- count(v) – partition/frame row count
- sum(v) – partition/frame total

BAREMON

Statistics functions:

- cume_dist() - cumulative distributrion
- percent_rank() - relative rank
- percentile_disc(n) - discrete percentile
- percentile_cont (n) - continuous percentile

BAREMON

Frame:

- In general defined as:

```
{ rows | groups | range}
between frame_start and frame_end
[exclude exclusion]
```

- Default frame:

```
range between unbounded preceding and current row
exclude no others
```

Frames:

- Only supported by some functions:
  - offset functions: first_value, last_value, nth_value
  - all aggregation functions
- For other functions - frame = partition

BAREMON

Frame type:

- rows – frames work with individual records
- groups – frames work with groups of records with the same ordering value
- range – frames work with groups of records whose order by column value falls within the specified range

BAREMON

Frame boundaries:

- unbounded preceding – from the partition boundary
- N preceding
- current row – current record
- N following
- unbounded following – to the partition boundary

BAREMON

unbounded preceding/following:

unbounded following

ROWS

name	salary asc
James	67
Michael	72
Anna	76
Tom	84
Mandy	92
Jack	92
Jessica	101
Dan	106
Adam	106
Phil	115

GROUPS

name	salary asc
James	67
Michael	72
Anna	76
Tom	84
Mandy	92
Jack	92
Jessica	101
Dan	106
Adam	106
Phil	115

RANGE

name	salary asc
James	67
Michael	72
Anna	76
Tom	84
Mandy	92
Jack	92
Jessica	101
Dan	106
Adam	106
Phil	115

BAREMON

current row :

current row

ROWS

name	salary asc
James	67
Michael	72
Anna	76
Tom	84
Mandy	92
Jack	92
Jessica	101
Dan	106
Adam	106
Phil	115

GROUPS

name	salary asc
James	67
Michael	72
Anna	76
Tom	84
Mandy	92
Jack	92
Jessica	101
Dan	106
Adam	106
Phil	115

RANGE

name	salary asc
James	67
Michael	72
Anna	76
Tom	84
Mandy	92
Jack	92
Jessica	101
Dan	106
Adam	106
Phil	115

N-preceding/N-following :

2 following
ROWS

name	salary asc
James	67
Michael	72
Anna	76
Tom	84
Mandy	92
Jack	92
Jessica	101
Dan	106
Adam	106
Phil	115

2 following
GROUPS

name	salary asc
James	67
Michael	72
Anna	76
Tom	84
Mandy	92
Jack	92
Jessica	101
Dan	106
Adam	106
Phil	115

15 following
RANGE

name	salary asc
James	67
Michael	72
Anna	76
Tom	84
Mandy	92
Jack	92
Jessica	101
Dan	106
Adam	106
Phil	115

BAREMON

EXCLUDE:

- **exclude no others** - default - don´t exclude anything
- **exclude current row**
- **exclude group** - exclude current record and all equal to it
- **exclude ties** - keep the current record but exclude equal to it

BAREMON

FILTER:

```
func(column) filter (where condition) over window_name
```

- to filter a specific window frame
- alternative: CASE - more flexible

```
func(case when condition then expression else other end) over window_name
```

BAREMON

When to replace aggregate functions

BAREMON

Limitations of aggregate functions:

- Collapse rows, losing granularity.
- Cannot combine summary and detailed data.

BAREMON

Practical examples of query optimisation

BAREMON

Resources:

- https://antonz.org/sql-window-functions-book/

BAREMON

Jan Suchánek

Principal Consultant at Baremon |
Expert in Database Services, ETL, and...

BAREMON

ANALYTICAL FUNCTIONS IN POSTGRESQL: MODERN ALTERNATIVE TO AGGREGATES

BAREMON

Jan Suchánek
Barbora Linhartová


Thank you