Java, SQL and jOOQ. *Best Practices and Lessons Learned from Writing Awesome Java and SQL Code. Get some hands-on insight on what's behind developing jOOQ.*

August 12, 2014

- **in sql**
- **9 Comments**

The Difference Between ROW_NUMBER(), RANK(), and DENSE_RANK()

One of the best features in SQL are window functions (https://blog.jooq.org/2013/11/03/probably-the-coolest-sql-feature-window-functions/). Dimitri Fontaine (http://tapoueh.org/blog/2013/08/20-Window-Functions) put it bluntly:

> *There was SQL before window functions and SQL after window functions*

If you're lucky enough to be using any of these databases, then you can use window functions yourself:

- CUBRID
- DB2
- Firebird
- Informix
- Oracle
- PostgreSQL
- SQL Server
- Sybase SQL Anywhere
- Teradata

(source here) (http://en.wikipedia.org/wiki/Comparison_of_relational_database_management_systems)

One of the most obvious and useful set of window functions are ranking functions where rows from your result set are ranked according to a certain scheme. There are three ranking functions:

- ROW_NUMBER()
- RANK()
- DENSE_RANK()

The difference is easy to remember. For the examples, let's assume we have this table (using PostgreSQL syntax):

```
1 | CREATE TABLE t(v) AS
2 | SELECT * FROM (
3 |   VALUES('a'),('a'),('a'),('b'),
4 |          ('c'),('c'),('d'),('e')
5 | ) t(v)
```

ROW_NUMBER()

… assigns unique numbers to each row within the `PARTITION` given the `ORDER BY` clause. So you'd get:

```
1 | SELECT v, ROW_NUMBER() OVER()
2 | FROM t
```

Note that some SQL dialects (e.g. SQL Server) require an explicit `ORDER BY` clause in the `OVER()` clause:

```
1 | SELECT v, ROW_NUMBER() OVER(ORDER BY v)
2 | FROM t
```

The above query returns:

```
| V | ROW_NUMBER |
|---|------------|
| a |          1 |
| a |          2 |
| a |          3 |
| b |          4 |
| c |          5 |
| c |          6 |
| d |          7 |
| e |          8 |
```

(see also this SQLFiddle) (http://sqlfiddle.com/#!15/1f6e4/2)

RANK()

… behaves like `ROW_NUMBER()`, except that "equal" rows are ranked the same. If we substitute `RANK()` into our previous query:

```
1 | SELECT v, RANK() OVER(ORDER BY v)
2 | FROM t
```

… then the result we're getting is this:

```
| V | RANK |
|---|------|
| a |    1 |
| a |    1 |
| a |    1 |
| b |    4 |
| c |    5 |
| c |    5 |
| d |    7 |
| e |    8 |
```

(see also this SQLFiddle) (http://sqlfiddle.com/#!15/1f6e4/3)

As you can see, much like in a sports ranking, we have *gaps* between the different ranks. We can avoid those gaps by using

DENSE_RANK()

Trivially, `DENSE_RANK()` is a rank with no gaps, i.e. it is *"dense"*. We can write:

```
1 | SELECT v, DENSE_RANK() OVER(ORDER BY v)
2 | FROM t
```

… to obtain

```
| V | DENSE_RANK |
|---|------------|
| a |          1 |
| a |          1 |
| a |          1 |
| b |          2 |
| c |          3 |
| c |          3 |
| d |          4 |
| e |          5 |
```

(see also this SQLFiddle) (http://sqlfiddle.com/#!15/1f6e4/4)

One interesting aspect of `DENSE_RANK()` is the fact that it "behaves like" `ROW_NUMBER()` when we add the `DISTINCT` keyword (https://blog.jooq.org/2013/10/09/sql-trick-row_number-is-to-select-what-dense_rank-is-to-select-distinct/).

```
1 | SELECT DISTINCT v, DENSE_RANK() OVER(ORDER BY v)
2 | FROM t
```

… to obtain

```
| V | DENSE_RANK |
|---|------------|
| a |          1 |
| b |          2 |
| e |          5 |
| d |          4 |
| c |          3 |
```

(see also this SQLFiddle) (http://sqlfiddle.com/#!15/1f6e4/5)

In fact, `ROW_NUMBER()` prevents you from using `DISTINCT`, because `ROW_NUMBER()` generates unique values across the partition *before* `DISTINCT` is applied:

```
1 SELECT DISTINCT v, ROW_NUMBER() OVER(ORDER BY v)
2 FROM t
3 ORDER BY 1, 2
```

`DISTINCT` has no effect:

```
| V | ROW_NUMBER |
|---|------------|
| a |          1 |
| a |          2 |
| a |          3 |
| b |          4 |
| c |          5 |
| c |          6 |
| d |          7 |
| e |          8 |
```

(see also this SQLFiddle) (http://sqlfiddle.com/#!15/1f6e4/8)

Putting it all together

A good way to understand the three ranking functions is to see them all in action side-by-side. Run this query

```
1 SELECT
2   v,
3   ROW_NUMBER() OVER(ORDER BY v),
4   RANK()       OVER(ORDER BY v),
5   DENSE_RANK() OVER(ORDER BY v)
6 FROM t
7 ORDER BY 1, 2
```

… or this one (using the SQL standard `WINDOW` clause, to reuse window specifications):

```
SELECT
  v,
  ROW_NUMBER() OVER(w),
  RANK()       OVER(w),
  DENSE_RANK() OVER(w)
FROM t
WINDOW w AS (ORDER BY v)
```

… to obtain:

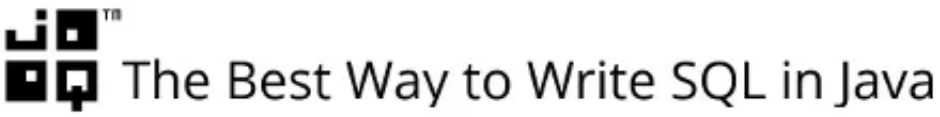
```
| V | ROW_NUMBER | RANK | DENSE_RANK |
|---|------------|------|------------|
| a |          1 |    1 |          1 |
| a |          2 |    1 |          1 |
| a |          3 |    1 |          1 |
| b |          4 |    4 |          2 |
| c |          5 |    5 |          3 |
| c |          6 |    5 |          3 |
| d |          7 |    7 |          4 |
| e |          8 |    8 |          5 |
```

(see also this SQLFiddle) (http://sqlfiddle.com/#!15/1f6e4/11)

Note that unfortunately, the `WINDOW` clause is not supported in all databases.

SQL is awesome

These things can be written very easily using SQL window functions. Once you get a hang of the syntax, you won't want to miss this killer feature in your every day SQL statements any more. Excited?



(http://www.jooq.org)

For further reading, consider:

- The jOOQ manual sections about window functions (http://www.jooq.org/doc/latest4/manual/sql-building/column-expressions/window-functions/)
- Dimitri Fontaine's excellent article "Understanding Window Functions" (http://tapoueh.org/blog/2013/08/20-Window-Functions)
- A real-world use-case: Counting neighboring colours in a stadium choreography

(https://blog.jooq.org/2014/04/15/how-to-do-this-with-sql-of-course/)
- A real-world use-case: Calculating running totals (https://blog.jooq.org/2014/04/29/nosql-no-sql-how-to-calculate-running-totals/) (not only with window functions)
- SQL 101: A Window into the World of Analytic Functions (http://www.oracle.com/technetwork/issue-archive/2013/13-mar/o23sql-1906475.html)

Tags: DENSE_RANK(), RANK, Ranking Functions, ROW_NUMBER(), sql, Window Functions

9 responses to “The Difference Between ROW_NUMBER(), RANK(), and DENSE_RANK()”

vladmihalcea says : August 13, 2014 at 08:41

16

1

i
Rate This

Great explanation. Window functions have many operations and it’s easy to get lost when you start working with them. I always like to remind Java developers that Hibernate query language is no match for window functions, pivot or recursive common table expressions. It’s pretty sad to find out that the vast majority of Java developers have never heard of these SQL features.

REPLY

lukaseder says : August 13, 2014 at 08:45

7

5

i
Rate This

It isn’t sad, it’s a challenge, and good grounds for SQL evangelism!

REPLY

Denis says : November 19, 2015 at 16:18

3

0

i
Rate This

Excellent! Well explained. Really helped me.
Thanks

REPLY

lukaseder says : November 19, 2015 at 17:56

3

0

i
Rate This

Glad it did, thanks for the feedback!

REPLY

DINESH B PATIL says : December 17, 2015 at 06:17

1

2

i
Rate This

Nice Explanation !

REPLY

Eswar says : January 13, 2016 at 07:03

0

0

i
Rate This

Excellent way of explanation,Thanks

REPLY

Manu says : February 8, 2016 at 09:20

1

0

i

Rate This

very well explained. Very much helpful for interview preparation.
Thanks.

REPLY

noname says : February 23, 2016 at 22:39

0

0

i
Rate This

How to use ROW_NUMBER or similar function in Firebird? Need to have current row number in SELECT.

REPLY

lukaseder says : February 24, 2016 at 00:18

0

0

i
Rate This

Use Firebird 3.0!

http://www.firebirdsql.org/file/community/conference-2014/pdf/02_fb.2014.whatsnew.30.en.pdf

REPLY