# ウィンドウ関数とplv8, PL/R

2015年12月12日　PostgreSQLアンカンファレンス

国府田 諭　Satoshi Koda　satkouda@gmail.com

本スライドの掲載場所　*http://kenpg.bitbucket.org/blog/201512/14.html*

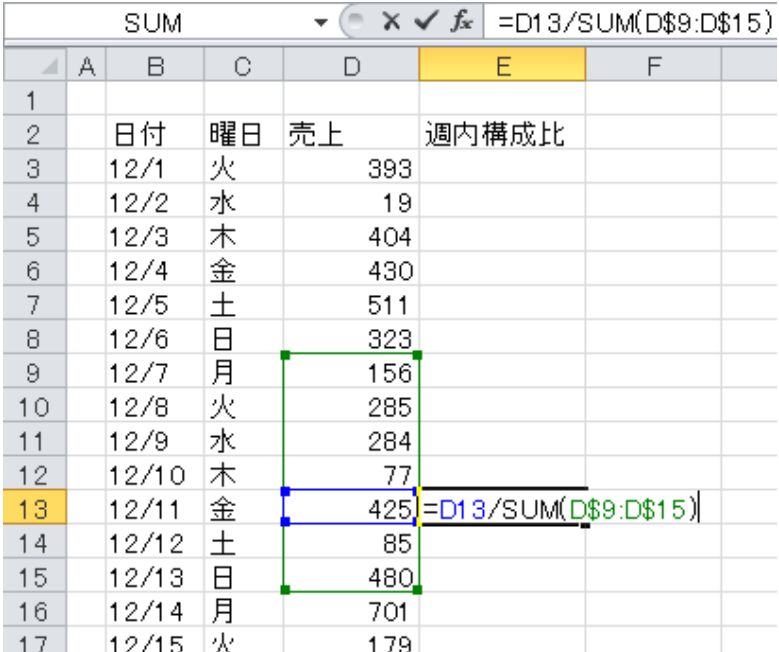
# 内容

- PostgreSQLのウィンドウ関数「超入門」
- plv8でウィンドウ関数を自作してみた
- PL/Rでもウィンドウ関数を作れた、けど微妙…

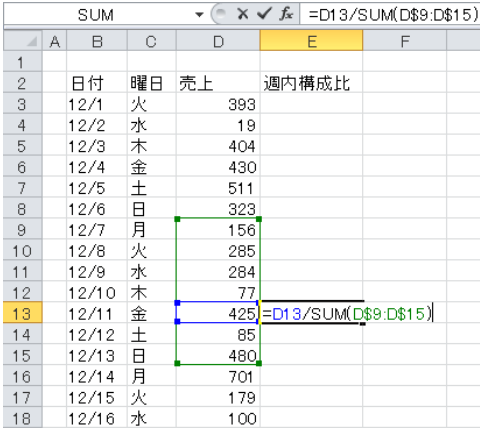
- 今回の実行環境：Windows 7 x64
  + PostgreSQL 9.4.5
  + plv8 1.4.2
  + PL/R 8.3.0.16

# ウィンドウ関数って何

- 簡単なイメージ ↓　Excelで時々やりませんか？



# 1. 集計結果を各行に出す・使う

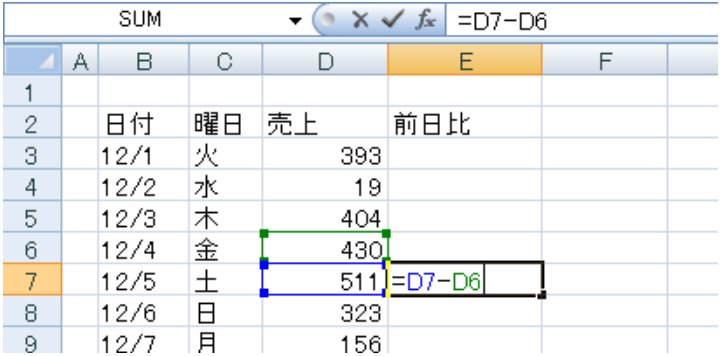



| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | | | | | | |
| 2 | | 日付 | 曜日 | 売上 | 週内構成比 | |
| 3 | | 12/1 | 火 | 393 | | |
| 4 | | 12/2 | 水 | 19 | | |
| 5 | | 12/3 | 木 | 404 | | |
| 6 | | 12/4 | 金 | 430 | | |
| 7 | | 12/5 | 土 | 511 | | |
| 8 | | 12/6 | 日 | 323 | | |
| 9 | | 12/7 | 月 | 156 | | |
| 10 | | 12/8 | 火 | 285 | | |
| 11 | | 12/9 | 水 | 284 | | |
| 12 | | 12/10 | 木 | 77 | | |
| 13 | | 12/11 | 金 | 425 | =D13/SUM(D$9:D$15) | |
| 14 | | 12/12 | 土 | 85 | | |
| 15 | | 12/13 | 日 | 480 | | |
| 16 | | 12/14 | 月 | 701 | | |
| 17 | | 12/15 | 火 | 179 | | |
| 18 | | 12/16 | 水 | 100 | | |

```
-- 上と同じことをウィンドウ関数で行うと

select ymd, val,
    val / (sum(val) over(partition by week))
from mytable,
    extract(week from ymd) as week;
```

# 2. 行の並びを決め、前後の行の値を持ってくる

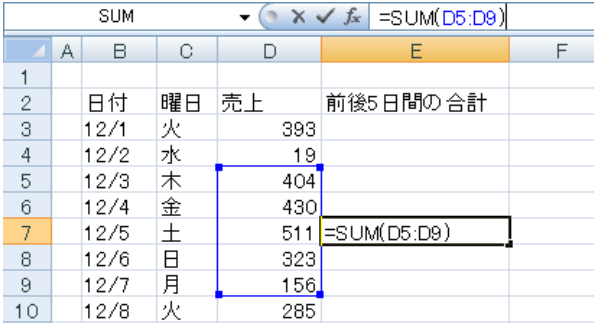



```
-- 上と同じことをウィンドウ関数で行うと

select ymd, val, val - lag(val) over w
from dummy
window w as (order by ymd);

-- 関数 lag() : カレントより前
-- 関数 lead() : カレントより後　ともにオフセット指定も可
```

# 3. 前後の「持ってくる対象」を指定する「フレーム」




| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | | | | | | |
| 2 | | 日付 | 曜日 | 売上 | 前後5日間の合計 | |
| 3 | | 12/1 | 火 | 393 | | |
| 4 | | 12/2 | 水 | 19 | | |
| 5 | | 12/3 | 木 | 404 | | |
| 6 | | 12/4 | 金 | 430 | | |
| 7 | | 12/5 | 土 | 511 | =SUM(D5:D9) | |
| 8 | | 12/6 | 日 | 323 | | |
| 9 | | 12/7 | 月 | 156 | | |
| 10 | | 12/8 | 火 | 285 | | |

```
-- 上と同じことをウィンドウ関数で行うと

select ymd, val, sum(val) over w
from dummy
window w as (order by ymd
    rows between 2 preceding and 2 following);

-- lag() など関数で「持ってくる対象」を指定せず、
-- フレームを作って丸ごと集約関数に渡す
```

# ドキュメントはちょっと分かりにくい

- 関連する記述が複数箇所に分散

  第3章 高度な諸機能 : ウィンドウ関数
  *http://www.postgresql.jp/document/9.4/html/tutorial-window.html*

  第4章 SQLの構文 : ウィンドウ関数呼び出し
  *http://www.postgresql.jp/document/9.4/html/sql-expressions.html*

  第9章 関数と演算子 : ウィンドウ関数
  *http://www.postgresql.jp/document/9.4/html/functions-window.html*

  VI. リファレンス : SELECT : WINDOW句
  *http://www.postgresql.jp/document/9.4/html/sql-select.html*

- フレームのサンプルがない

# 悩み 1. 最初と最後の「半端」を除くのが面倒

```
-- 前後5日間の集計値を単純に出すと、
-- ぱっと見て、最初と最後が「半端な数の集計」とは分からない

select ymd, val, sum(val) over w
from mytable
window w as (order by ymd
    rows between 2 preceding and 2 following);

    ymd     | val | sum
------------+-----+-----
 2015-12-01 |  20 | 154  -- 3日間のみ
 2015-12-02 |  54 | 207  -- 4日間のみ
 2015-12-03 |  80 | 223  -- これ以降が5日間
 2015-12-04 |  53 | 289
 2015-12-05 |  16 | 299
 2015-12-06 |  86 | 270
 2015-12-07 |  64 | 266
 2015-12-08 |  51 | 292
...
```

# 悩み 1. 最初と最後の「半端」を除くのが面倒（続）

```
-- 前後5日間かどうかは count() で分かる
-- 下記をサブクエリにし、上位で where 句で絞ればいいけど、
-- 前後■日が変わると面倒

select ymd, val, count(val) over w
from mytable
window w as (order by ymd
    rows between 2 preceding and 2 following);

    ymd     | val | count
------------+-----+-------
 2015-12-01 |  20 |     3  -- 3日間のみ
 2015-12-02 |  54 |     4  -- 4日間のみ
 2015-12-03 |  80 |     5
 2015-12-04 |  53 |     5
 2015-12-05 |  16 |     5
 2015-12-06 |  86 |     5
 2015-12-07 |  64 |     5
 2015-12-08 |  51 |     5
...
```

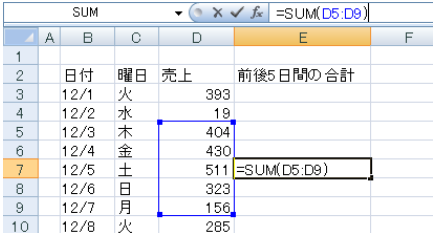
# 悩み 1. 最初と最後の「半端」を除くのが面倒（続）

```
-- サブクエリなしで、ぱっと見で最初・最後の半端な日数分を
-- 分かるように NULL にする例
-- case句で分岐してるけど、これも面倒…

select ymd, val,
    case when count(*) over w = 5 then sum(val) over w
    end as sum5days
from mytable
window w as (order by ymd
    rows between 2 preceding and 2 following);

    ymd     | val | sum5days
------------+-----+----------
 2015-12-01 |  20 |
 2015-12-02 |  54 |
 2015-12-03 |  80 |      223
 2015-12-04 |  53 |      289
 2015-12-05 |  16 |      299
 2015-12-06 |  86 |      270
...
```

# 悩み 2. フレームの「前後の幅」を変数化できない




| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | | | | | | |
| 2 | | 日付 | 曜日 | 売上 | 前後5日間の合計 | |
| 3 | | 12/1 | 火 | 393 | | |
| 4 | | 12/2 | 水 | 19 | | |
| 5 | | 12/3 | 木 | 404 | | |
| 6 | | 12/4 | 金 | 430 | | |
| 7 | | 12/5 | 土 | 511 | =SUM(D5:D9) | |
| 8 | | 12/6 | 日 | 323 | | |
| 9 | | 12/7 | 月 | 156 | | |
| 10 | | 12/8 | 火 | 285 | | |

```
-- Excelの「参照範囲の貼り替え」より楽に、SQLで「前後■日」を
-- 可変にしたくても、現状（PostgreSQL 9.5まで）はできない

with x (days) as (
    values (2)  -- ここを変えるだけで済めばいいなぁ…
)
select ymd, val, sum(val) over w
from x, dummy
window w as (order by ymd
    rows between x.days preceding and x.days following);

-- ERROR: argument of ROWS must not contain variables
```

## 何とかしたい → plv8でウィンドウ関数が作れる！

# plv8：自作関数をJavaScriptで書ける

- PostgreSQL Extension Network → *http://pgxn.org/dist/plv8/*

- CentOS、OS X、Windowsでのインストールまとめと、簡単なウィンドウ関数作成例（実用性はないけど…）を下記に書きました

  *http://kenpg.bitbucket.org/blog/201509/26.html*


# 自作ウィンドウ関数について

- 文法：CREATE FUNCTION 文にパラメータ WINDOW を加えるだけ

- ただしドキュメントいわく「WINDOWは、この関数が普通の関数ではなくウィンドウ関数であることを示します。 現在これは C言語で作成した関数のみに使用することができます。」
  *http://www.postgresql.jp/document/9.4/html/sql-createfunction.html*

- でも、組み込み言語plv8とPL/Rでも作れるらしい。これは便利^o^

  plv8（1.3.0以降）
  *http://pgxn.org/dist/plv8/doc/plv8.html*
  *https://github.com/plv8/plv8/blob/master/Changes*

  PL/R User's Guide - Window Functions
  *http://www.joeconway.com/plr/doc/plr-window-funcs.html*

# 早速plv8で、「前後■日」を可変にする関数を作ろう！

# 対象とするテーブル（日付と値だけ）

```
select ymd, val from mytable;

    ymd     | val
------------+-----
 2015-12-01 |  20
 2015-12-02 |  54
 2015-12-03 |  80
 2015-12-04 |  53
 2015-12-05 |  16
 2015-12-06 |  86
 2015-12-07 |  64
 2015-12-08 |  51
 2015-12-09 |  49
 2015-12-10 |  42
 2015-12-11 |  28
...
```

# 最初に単純なテスト：各行の値を配列に足す

```
create or replace function plv8_winf_ex1(val float)
returns float[] language plv8 immutable window as
$$
    var w = plv8.get_window_object(),
        obj = w.get_partition_local();

    if (w.get_current_position() === 0) obj = {"ary": []};
    obj.ary.push(val);
    w.set_partition_local(obj);
    return obj.ary;
$$;
```

- plv8.get_window_object() でウィンドウ関数オブジェクトを準備
- get_partition_local() で直前までの状態を取得、
  set_partition_local() で現状の状態を保存、らしい

# 実行すると

```
select ymd, val, plv8_winf_ex1(val) over(order by ymd)
from mytable;

    ymd     | val |                          plv8_winf_ex1
------------+-----+-----------------------------------------
 2015-12-01 |  20 | {20}
 2015-12-02 |  54 | {20,54}
 2015-12-03 |  80 | {20,54,80}
 2015-12-04 |  53 | {20,54,80,53}
 2015-12-05 |  16 | {20,54,80,53,16}
 2015-12-06 |  86 | {20,54,80,53,16,86}
 2015-12-07 |  64 | {20,54,80,53,16,86,64}
 2015-12-08 |  51 | {20,54,80,53,16,86,64,51}
 2015-12-09 |  49 | {20,54,80,53,16,86,64,51,49}
 2015-12-10 |  42 | {20,54,80,53,16,86,64,51,49,42}
 2015-12-11 |  28 | {20,54,80,53,16,86,64,51,49,42,28}
 2015-12-12 |  18 | {20,54,80,53,16,86,64,51,49,42,28,18}
...
```

# テスト 2：「過去■日」を可変にしてみる（引数化）

```
create or replace function plv8_winf_ex2
(val float, pre int)
returns float[] language plv8 immutable window as
$$
    var w = plv8.get_window_object(),
        obj = w.get_partition_local();

    if (w.get_current_position() === 0) obj = {"ary": []};
    obj.ary.push(val);
    obj.ary = obj.ary.slice(-1 * pre); // ■日より前は捨てる
    w.set_partition_local(obj);
    return obj.ary;
$$;
```

- 最初のテストに、「■日より前を捨てる」一行を追加しただけ
- 2番目の引数で、対象とする「直近■日間」を指定

# 実行すると

```
select ymd, val,
    plv8_winf_ex2(val, 5) over(order by ymd)
from mytable;

    ymd     | val |  plv8_winf_ex2
------------+-----+-----------------
 2015-12-01 |  20 | {20}
 2015-12-02 |  54 | {20,54}
 2015-12-03 |  80 | {20,54,80}
 2015-12-04 |  53 | {20,54,80,53}
 2015-12-05 |  16 | {20,54,80,53,16}
 2015-12-06 |  86 | {54,80,53,16,86}
 2015-12-07 |  64 | {80,53,16,86,64}
 2015-12-08 |  51 | {53,16,86,64,51}
 2015-12-09 |  49 | {16,86,64,51,49}
 2015-12-10 |  42 | {86,64,51,49,42}
 2015-12-11 |  28 | {64,51,49,42,28}
...
```

# もう少し便利に：最初の半端な日数を NULL にする

```
create or replace function plv8_winf_ex3(val float, pre int)
returns float[] language plv8 immutable window as
$$
    var w = plv8.get_window_object(),
        obj = w.get_partition_local();

    if (w.get_current_position() === 0) obj = {"ary": []};
    obj.ary.push(val);
    obj.ary = obj.ary.slice(-1 * pre);
    w.set_partition_local(obj);

    if (obj.ary.length === pre) return obj.ary;
    // ここで絞る
$$;
```

- 最後の出力時に、半端な日数の行を除く処理を入れただけ。簡単！

# 実行すると

```
select ymd, val, plv8_winf_ex3(val, 5) over(order by ymd)
from mytable;

    ymd     | val |  plv8_winf_ex3
------------+-----+-----------------
 2015-12-01 |  20 |
 2015-12-02 |  54 |
 2015-12-03 |  80 |
 2015-12-04 |  53 |
 2015-12-05 |  16 | {20,54,80,53,16}
 2015-12-06 |  86 | {54,80,53,16,86}
 2015-12-07 |  64 | {80,53,16,86,64}
 2015-12-08 |  51 | {53,16,86,64,51}
 2015-12-09 |  49 | {16,86,64,51,49}
 2015-12-10 |  42 | {86,64,51,49,42}
 2015-12-11 |  28 | {64,51,49,42,28}
 2015-12-12 |  18 | {51,49,42,28,18}
...
```

# 対象とする「過去■日」が、引数で可変になった

```
select ymd, val, plv8_winf_ex3(val, 3) over(order by ymd)
from mytable;

    ymd     | val | plv8_winf_ex3
------------+-----+---------------
 2015-12-01 |  20 |
 2015-12-02 |  54 |
 2015-12-03 |  80 | {20,54,80}
 2015-12-04 |  53 | {54,80,53}
 2015-12-05 |  16 | {80,53,16}
 2015-12-06 |  86 | {53,16,86}
 2015-12-07 |  64 | {16,86,64}
 2015-12-08 |  51 | {86,64,51}
 2015-12-09 |  49 | {64,51,49}
...
```

- 配列からの集計・演算は、実際の必要に応じて関数内に書いてもいいし、結果取得後のクエリで自由に設定するのもできます。

# 同じことが、通常のウィンドウ関数のクエリだと

```
select ymd, val,
    case when count(*) over w = 3 then
        array_agg(val) over w
    end as agg3days
from mytable
window w as (order by ymd rows 2 preceding);

    ymd     | val |  agg3days
------------+-----+------------
 2015-12-01 |  20 |
 2015-12-02 |  54 |
 2015-12-03 |  80 | {20,54,80}
 2015-12-04 |  53 | {54,80,53}
 2015-12-05 |  16 | {80,53,16}
 2015-12-06 |  86 | {53,16,86}
...
```

- フレーム指定の「2」と、NULLにするための「3」を柔軟に変えられない

# 今日の本命：「前後■日間」を引数で指定

```
create or replace function plv8_winf_ex4
(val float, pre int, follow int)
returns float[] language plv8 immutable window as
$$
    var w = plv8.get_window_object(),
        ary = [],
        len = follow - pre + 1;
    for(var i = pre; i <= follow; i++) {
        var num = w.get_func_arg_in_partition(
            0, i, w.SEEK_CURRENT, false);
        if (num) ary.push(num);
    }
    if(ary.length == len) return ary;
$$;
```

- 使うメソッドは get_func_arg_in_partition() のみ。
  引数の詳細はドキュメント http://pgxn.org/dist/plv8/ を参照

※ 最後の引数は、今回特に関係なかった模様

# 実行すると

```
select ymd, val,
    plv8_winf_ex4(val,-2, 2) over(order by ymd)
from mytable;

    ymd     | val |  plv8_winf_ex4
------------+-----+-----------------
 2015-12-01 |  20 |
 2015-12-02 |  54 |
 2015-12-03 |  80 | {20,54,80,53,16}
 2015-12-04 |  53 | {54,80,53,16,86}
 2015-12-05 |  16 | {80,53,16,86,64}
 2015-12-06 |  86 | {53,16,86,64,51}
 2015-12-07 |  64 | {16,86,64,51,49}
 2015-12-08 |  51 | {86,64,51,49,42}
 2015-12-09 |  49 | {64,51,49,42,28}
 2015-12-10 |  42 | {51,49,42,28,18}
 2015-12-11 |  28 | {49,42,28,18,41}
 2015-12-12 |  18 | {42,28,18,41,28}
...
```

# 実行すると（出力の最後）

```
select ymd, val,
    plv8_winf_ex4(val, -2, 2) over(order by ymd)
from mytable;
...
 2015-12-19 |  89 | {31,98,89,6,83}
 2015-12-20 |   6 | {98,89,6,83,42}
 2015-12-21 |  83 | {89,6,83,42,44}
 2015-12-22 |  42 | {6,83,42,44,43}
 2015-12-23 |  44 | {83,42,44,43,90}
 2015-12-24 |  43 | {42,44,43,90,30}
 2015-12-25 |  90 | {44,43,90,30,6}
 2015-12-26 |  30 | {43,90,30,6,15}
 2015-12-27 |   6 | {90,30,6,15,19}
 2015-12-28 |  15 | {30,6,15,19,22}
 2015-12-29 |  19 | {6,15,19,22,46}
 2015-12-30 |  22 |
 2015-12-31 |  46 |
(31 rows)
```

# 通常のクエリだと

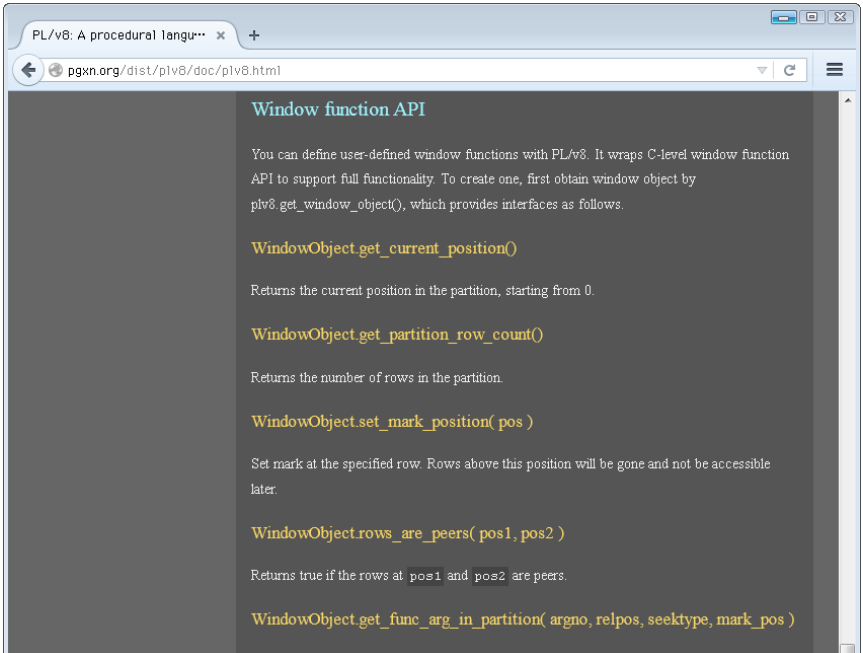
```
select ymd, val,
    case when count(*) over w = 5 then
        array_agg(val) over w
    end as agg5days
from mytable
window w as (order by ymd
    rows between 2 preceding and 2 following);

    ymd     | val |     agg5days
------------+-----+------------------
 2015-12-01 |  20 |
 2015-12-02 |  54 |
 2015-12-03 |  80 | {20,54,80,53,16}
 2015-12-04 |  53 | {54,80,53,16,86}
 2015-12-05 |  16 | {80,53,16,86,64}
 2015-12-06 |  86 | {53,16,86,64,51}
...
```

- フレーム指定の「2, 2」と、NULLにするための「5」を変えるのは面倒

# 各種メソッドを理解すれば、もっと便利に使えそう！

*http://pgxn.org/dist/plv8/doc/plv8.html*



PL/Rのドキュメントに「ウィンドウ関数」を

作るセクションがあったが、

実際作ってみたら結果が微妙だった件

# とりあえず関数定義は普通にできる

```
create or replace function plr_winf_ex2(float, int)
returns float[] language plr immutable window as
$$
    len1 = length(farg1)
    len2 = farg2
    if (len1 >= len2) {
        return(farg1[(len1 - len2 + 1) : len1])
    }
$$;
```

- plv8と同じウィンドウ関数（過去■件を配列化）を定義 ↑
- ウィンドウ関数用のオブジェクトは特になく、fargNという変数を使う
  （引数が一つならfarg1、二つならfarg1とfarg2、のように）
- fargNがウィンドウオブジェクト（当該の列の全件）になる模様

# 確かに動くように見えるが…

```
select gs, plr_winf_ex2(gs, 5) over(order by gs)
from generate_series(1, 20) as gs;

 gs |   plr_winf_ex2
----+------------------
  1 |
  2 |
  3 |
  4 |
  5 | {1,2,3,4,5}
  6 | {2,3,4,5,6}
  7 | {3,4,5,6,7}
  8 | {4,5,6,7,8}
  9 | {5,6,7,8,9}
 10 | {6,7,8,9,10}
 11 | {7,8,9,10,11}
 12 | {8,9,10,11,12}
...
```

# 値がソートされてないと、現在行と次行しか見ない orz

```
select gs, val,
    plr_winf_ex2(gs, 5) over w as agg_gs,
    plr_winf_ex2(val, 5) over w as agg_val
from (
    select gs, floor(random() * 100) as val
    from generate_series(1, 20) as gs
) foo
window w as (order by gs);

 gs | val |      agg_gs      |     agg_val
----+-----+------------------+------------------
  1 |  39 |                  |
  2 |  50 |                  |
  3 |  65 |                  |
  4 |  77 |                  |
  5 |  74 | {1,2,3,4,5}      | {74,21,74,21,74}
  6 |  21 | {2,3,4,5,6}      | {21,27,21,27,21}
  7 |  27 | {3,4,5,6,7}      | {27,81,27,81,27}
...
```

# まとめ

- PostgreSQLのウィンドウ関数、使い方を覚えると結構はかどる！
- 不便だな～と思う点は、plv8で気軽に自作して補える！
- PL/Rの挙動は残念。でも実行環境によっては違うかも？　今後調べます
- 以上、前回のアンカンファレンスに続き、PostgreSQL＆周辺にある便利な集計・分析環境についてでした！