

# PostgreSQL

quand on est développeur

Dimitri Fontaine `dimitri@2ndQuadrant.fr`

31 Octobre 2012

## 2ndQuadrant France PostgreSQL Major Contributor

- `pgloader`, `prefix`, `skytools`, `debian`, ...
- `CREATE EXTENSION`
- `CREATE EVENT TRIGGER`
- *Bi-Directional Réplication*
- *Partitionnement*



## 2ndQuadrant France PostgreSQL Major Contributor

- pgloader, prefix, skytools, debian, ...
- CREATE EXTENSION
- CREATE EVENT TRIGGER
- *Bi-Directional Réplication*
- *Partitionnement*



## 2ndQuadrant France PostgreSQL Major Contributor

- pgloader, prefix, skytools, debian, ...
- CREATE EXTENSION
- CREATE EVENT TRIGGER
- *Bi-Directional Réplication*
- *Partitionnement*



## 2ndQuadrant France PostgreSQL Major Contributor

- pgloader, prefix, skytools, debian, ...
- CREATE EXTENSION
- CREATE EVENT TRIGGER
- *Bi-Directional Réplication*
- *Partitionnement*



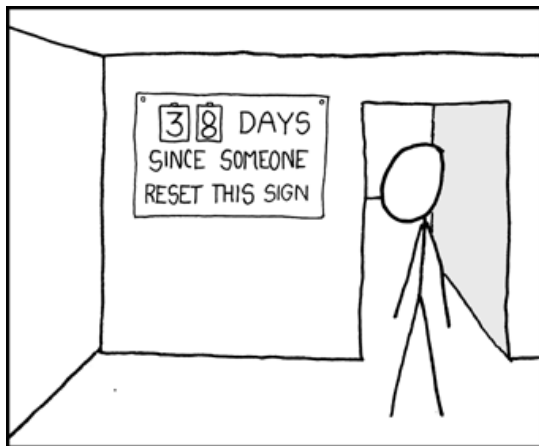
# Outils et langages de développement

PHP est une petite partie de votre métier de développeur web.

- HTML
- Javascript
- *JQuery*
- **SQL**



## Définition d'un projet simple



# Définition du projet

Nous allons avancer à partir d'un exemple

- Gestion d'un compteur avec cycles
- Ajout de nouvelles mesures dans le temps
- Consolidation mensuelle pour facturation
- Analyse du comportement du compteur



# Définition du projet

Nous allons avancer à partir d'un exemple

- Gestion d'un compteur avec cycles
- Ajout de nouvelles mesures dans le temps
- Consolidation mensuelle pour facturation
- Analyse du comportement du compteur

# Définition du projet

Nous allons avancer à partir d'un exemple

- Gestion d'un compteur avec cycles
- Ajout de nouvelles mesures dans le temps
- Consolidation mensuelle pour facturation
- Analyse du comportement du compteur



# Définition du projet

Nous allons avancer à partir d'un exemple

- Gestion d'un compteur avec cycles
- Ajout de nouvelles mesures dans le temps
- Consolidation mensuelle pour facturation
- Analyse du comportement du compteur

# SQL commence avec les DDLs

**Joe Celko:** 80% du boulot est dans les DDL

## Example (DDL)

```
create table mesures(date timestamptz primary key,  
                    mesure integer);
```

```
dim=# \d mesures
```

```
\d mesures
```

Table "public.mesures"

Column	Type	Modifiers
date	timestamp with time zone	not null
mesure	integer	

Indexes:

```
"mesures_pkey" PRIMARY KEY, btree (date)
```

## Simplifions pour la présentation

```
create table measures(tick int, nb int);
```

```
insert into measures  
  values (1, 0), (2, 10), (3, 20), (4, 30), (5, 40),  
         (6, 0), (7, 20), (8, 30), (9, 60);
```



## Les données de test

Mesures issues de notre compteur, avec un démarrage à 0 et un *reset* en milieu de cycle. La consommation globale est ici  $40 + 60 = 100$ .

```
select * from measures;
```

```
tick | nb
```

```
-----+-----
```

```
1 | 0
```

```
2 | 10
```

```
3 | 20
```

```
4 | 30
```

```
5 | 40
```

```
6 | 0
```

```
7 | 20
```

```
8 | 30
```

```
9 | 60
```

```
(9 rows)
```



# PostgreSQL sait travailler avec des tableaux

```
select array_agg(nb) from measures;  
array_agg
```

```
-----  
{0,10,20,30,40,0,20,30,60}  
(1 row)
```

# Trouver les points avant reset

Écrire une requête **SQL** ici

tick	nb	max
1	0	
2	10	
3	20	
4	30	
5	40	40
6	0	
7	20	
8	30	
9	60	60

(9 rows)



## Window Functions: lead() over()

```
select tick,  
       nb,  
       lead(nb) over (order by tick)  
from measures;
```

tick	nb	lead
1	0	10
2	10	20
3	20	30
4	30	40
5	40	0
6	0	20
7	20	30
8	30	60
9	60	

(9 rows)

# Window Functions et CASE

```
select tick, nb,  
       case when lead(nb) over w < nb  
            then nb  
            when lead(nb) over w is null  
            then nb  
            else null  
       end as max  
from measures  
window w as (order by tick);
```

tick	nb	max
1	0	
2	10	
3	20	
4	30	
5	40	40
6	0	
7	20	
8	30	
9	60	60

(9 rows)



## Window Functions et clause WHERE

```
with t(tick, nb, max) as (  
    select tick, nb,  
           case when lead(nb) over w < nb then nb  
                when lead(nb) over w is null then nb  
                else null  
           end as max  
    from measures  
    window w as (order by tick)  
)  
select tick, nb, max from t where max is not null;  
tick | nb | max  
-----+-----+-----  
    5 | 40 | 40  
    9 | 60 | 60  
(2 rows)
```

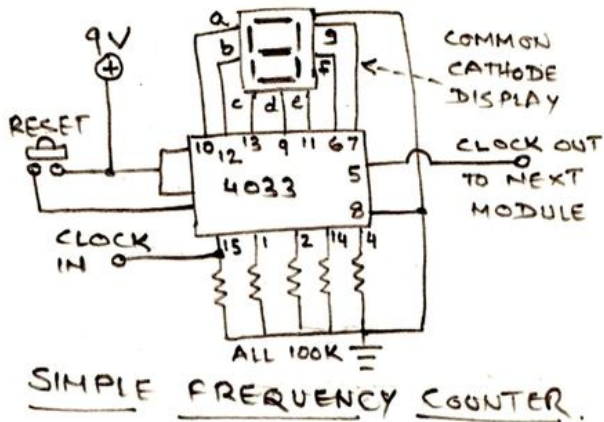


## Common Table Expressions avec WITH

```
with t(tops) as (  
    select case when lead(nb) over w < nb then nb  
              when lead(nb) over w is null then nb  
              else null  
            end as max  
    from measures  
    window w as (order by tick)  
)  
select sum(tops) from t;  
sum  
-----  
100  
(1 row)
```



# Consumation Globale: problème résolu, SQL, 9 lignes



## Création de plusieurs cycles

```
insert into measures
  values (10, 0), (11, 10), (12, 30), (13, 35), (14, 45),
         (15, 25), (16, 50), (17, 100), (18, 110);
```



## Visualisation de plusieurs cycles

```
with t(tick, nb, max) as (  
  select tick, nb,  
         case when lead(nb) over w < nb then nb  
              when lead(nb) over w is null then nb  
              else null  
         end as max  
  from measures  
  window w as (order by tick)  
)
```

```
select tick, nb, max from t where max is not null;
```

```
tick | nb  | max  
-----+-----+-----  
5    | 40  | 40  
9    | 60  | 60  
14   | 45  | 45  
18   | 110 | 110
```

(4 rows)

## Consommation globale avec plusieurs cycles

```
with t(tops) as (  
    select case when lead(nb) over w < nb then nb  
              when lead(nb) over w is null then nb  
              else null  
            end as max  
    from measures  
    window w as (order by tick)  
)  
select sum(tops) from t;  
sum  
-----  
255  
(1 row)
```





## Limiter la période de mesures



## Limiter la période de mesures

```
select tick, nb
  from measures
 where tick >= 4 and tick < 14;
```

tick	nb
4	30
5	40
6	0
7	20
8	30
9	60
10	0
11	10
12	30
13	35



## Limiter la période de mesures, first\_value

```
select nb,
       first_value(nb) over w as first,
       case when lead(nb) over w < nb
            then nb
            when lead(nb) over w is null
            then nb
            else null
       end as max
from measures
where tick >= 4 and tick < 14
window w as (order by tick);
```

nb	first	max
30	30	
40	30	40
0	30	
20	30	
30	30	
60	30	60
0	30	
10	30	
30	30	
35	30	35

(10 rows)



## Consommation globale dans une période limitée

```
with t as (  
  select tick,  
         first_value(nb) over w as first,  
         case when lead(nb) over w < nb then nb  
              when lead(nb) over w is null then nb  
              else null  
         end as max  
  from measures  
  where tick >= 4 and tick < 14  
  window w as (order by tick)  
)  
select sum(max) - min(first) as sum from t;  
sum  
-----  
105  
(1 row)
```

# Comportement du *reset* du compteur

**DC 24V**



**Range :0-99999** SourcingMap

## Partitionnement par *reset*

```
with tops as (  
    select tick, nb,  
           case when lead(nb) over w < nb then nb  
                when lead(nb) over w is null then nb  
                else null  
           end as max  
    from measures  
window w as (order by tick)  
)  
select tick, nb, max,  
       (select tick  
        from tops t2  
        where t2.tick >= t1.tick and max is not null  
        order by t2.tick  
        limit 1) as p  
from tops t1;
```



## Partitionnement par *reset*

tick	nb	max	p
1	0		5
2	10		5
3	20		5
4	30		5
5	40	40	5
6	0		9
7	20		9
8	30		9
9	60	60	9

tick	nb	max	p
10	0		14
11	10		14
12	30		14
13	35		14
14	45	45	14
15	25		18
16	50		18
17	100		18
18	110	110	18



## Consommation globale par range avec PARTITION BY

```
with tops as ( <case lead() over()> ),
      parts as ( <self join limit 1> ),
      ranges as (
select
      first_value(tick) over w as start,
      last_value(tick) over w as end,
      max(max) over w
from parts
window w as (partition by p
              order by tick)
)
select * from ranges
where max is not null;
```

	start	end	max
	1	5	40
	6	9	60
	10	14	45
	15	18	110

(4 rows)





## Consommation globale par range avec int4range()

```
with tops as ( <case lead() over()> ),
     parts as ( <self join limit 1> ),
     ranges as (
select int4range(
    first_value(tick) over w,
    last_value(tick) over w,
    '[]') as range,
    max(max) over w as compteur
from parts
window w as (partition by p
              order by tick)
)
select range, compteur
from ranges
where compteur is not null;
```

range	compteur
[1,6)	40
[6,10)	60
[10,15)	45
[15,19)	110
(4 rows)	

## Consommation globale par range avec @>

```
with tops as ( <case lead() over()> ),
     parts as ( <self join limit 1> ),
     ranges as ( <int4range()
                 over (partition by
                       order by)> )
select range, compteur
   from ranges
  where compteur is not null
        and range @> 11;
```

range	compteur
[10,15)	45

(1 row)

# Conclusion

SQL fait partie de vos outils de développeur, tirez-en le meilleur!

