Utiliser PostgreSQL en 2015

Java User Group, Montpellier

Dimitri Fontaine dimitri@2ndQuadrant.fr
@tapoueh

20 Mai 2015

Dimitri Fontaine

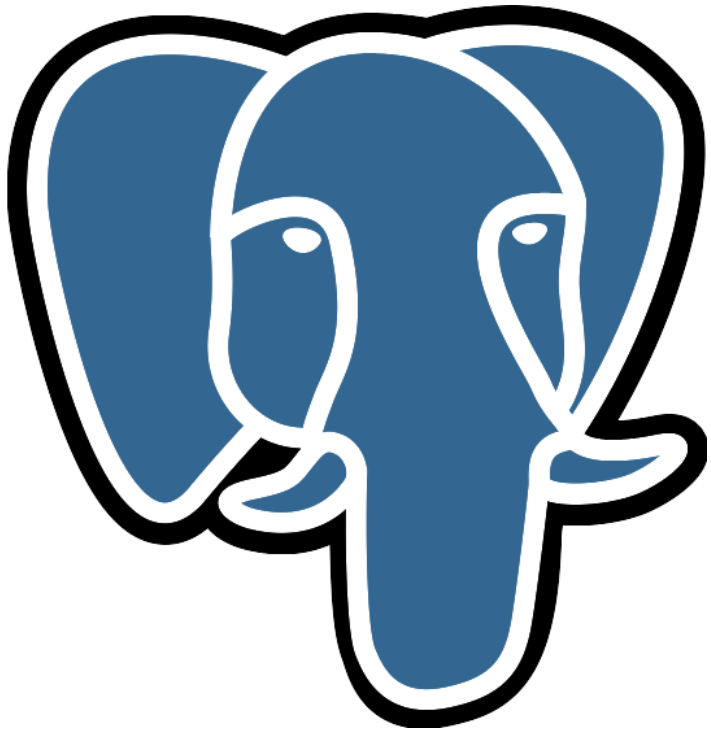
Principal Consultant at 2ndQuadrant

2ndQuadrant

Professional PostgreSQL

PostgreSQL Major Contributor

- `CREATE EXTENSION`
- `CREATE EVENT TRIGGER`
- *Bi-Directional Réplication*
- `pgloader`
- `prefix`, `skytools`
- `apt.postgresql.org`
- `pginstall`
- `pgcharts`



pgloader

http://pgloader.io/

Migrer de MySQL à PostgreSQL

Example de migration des données, *complet*

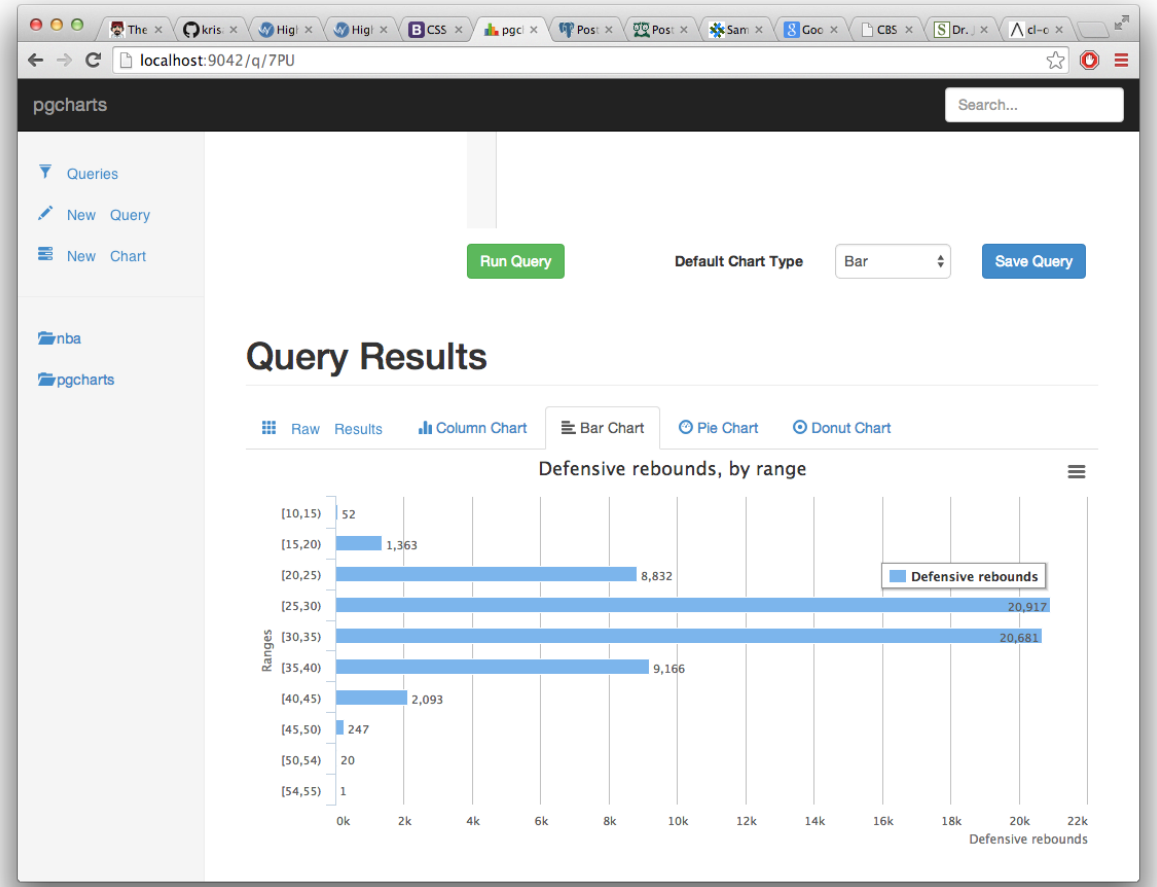
```
$ createdb pagila
$ pgloader mysql://user@localhost/sakila
           pgsql:///pagila
```

pgloader mysql://root@localhost/sakila pgsql:///pagila

```
./build/bin/pgloader mysql://root@localhost/sakila pgsql:///pagila
2015-01-30T12:49:09.030000+01:00 LOG Main logs in '/private/tmp/pgloader/pgloader.log'
2015-01-30T12:49:09.034000+01:00 LOG Data errors in '/private/tmp/pgloader/'
                    table name       read   imported     errors            time
-----------------------------  ---------  ---------  ---------  --------------
                fetch meta data         43         43          0          0.358s
                   create, drop          0         36          0          0.263s
-----------------------------  ---------  ---------  ---------  --------------
                         actor        200        200          0          0.151s
                       address        603        603          0          0.067s
                      category         16         16          0          0.025s
                          city        600        600          0          0.036s
                       country        109        109          0          0.019s
                      customer        599        599          0          0.040s
                          film       1000       1000          0          0.106s
                     film_actor       5462       5462          0          0.171s
                  film_category       1000       1000          0          0.068s
                      film_text       1000       1000          0          0.070s
                     inventory       4581       4581          0          0.172s
                      language          6          6          0          0.078s
                       payment      16049      16049          0          0.470s
                        rental      16044      16044          0          0.458s
                         staff          2          2          0          0.103s
                         store          2          2          0          0.035s
           Index Build Completion          0          0          0          0.001s
-----------------------------  ---------  ---------  ---------  --------------
                  Create Indexes         41         41          0          0.291s
                 Reset Sequences          0         13          0          0.045s
                    Primary Keys         16         16          0          0.014s
                    Foreign Keys         22         22          0          0.079s
                        Comments          0          0          0          0.000s
-----------------------------  ---------  ---------  ---------  --------------
                Total import time      47273      47273          0          2.829s
```

Turn your SQL queries into nice looking charts.

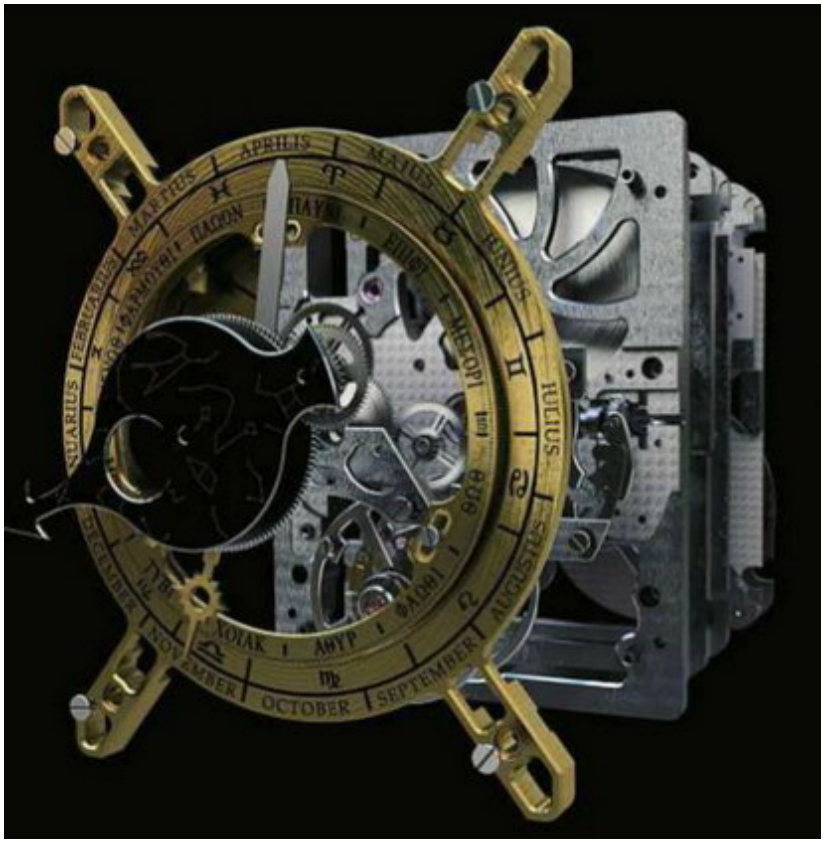
https://github.com/dimitri/pgcharts/



Système de Gestion de Bases de données Relationnelles

La Fabuleuse machine d'Anticythère

Système de Gestion de Bases de données Relationnelles



La Fabuleuse machine d'Anticythère

Système de Gestion de Bases de données Relationnelles

Système de Gestion de Bases de données Relationnelles

Transactions

Transactions

```
BEGIN;
  DROP TABLE donnees_critiques;
ROLLBACK;
```

Transactions

```
START TRANSACTION ISOLATION LEVEL SERIALIZABLE;
  DROP TABLE donnees_critiques;
ROLLBACK;
```

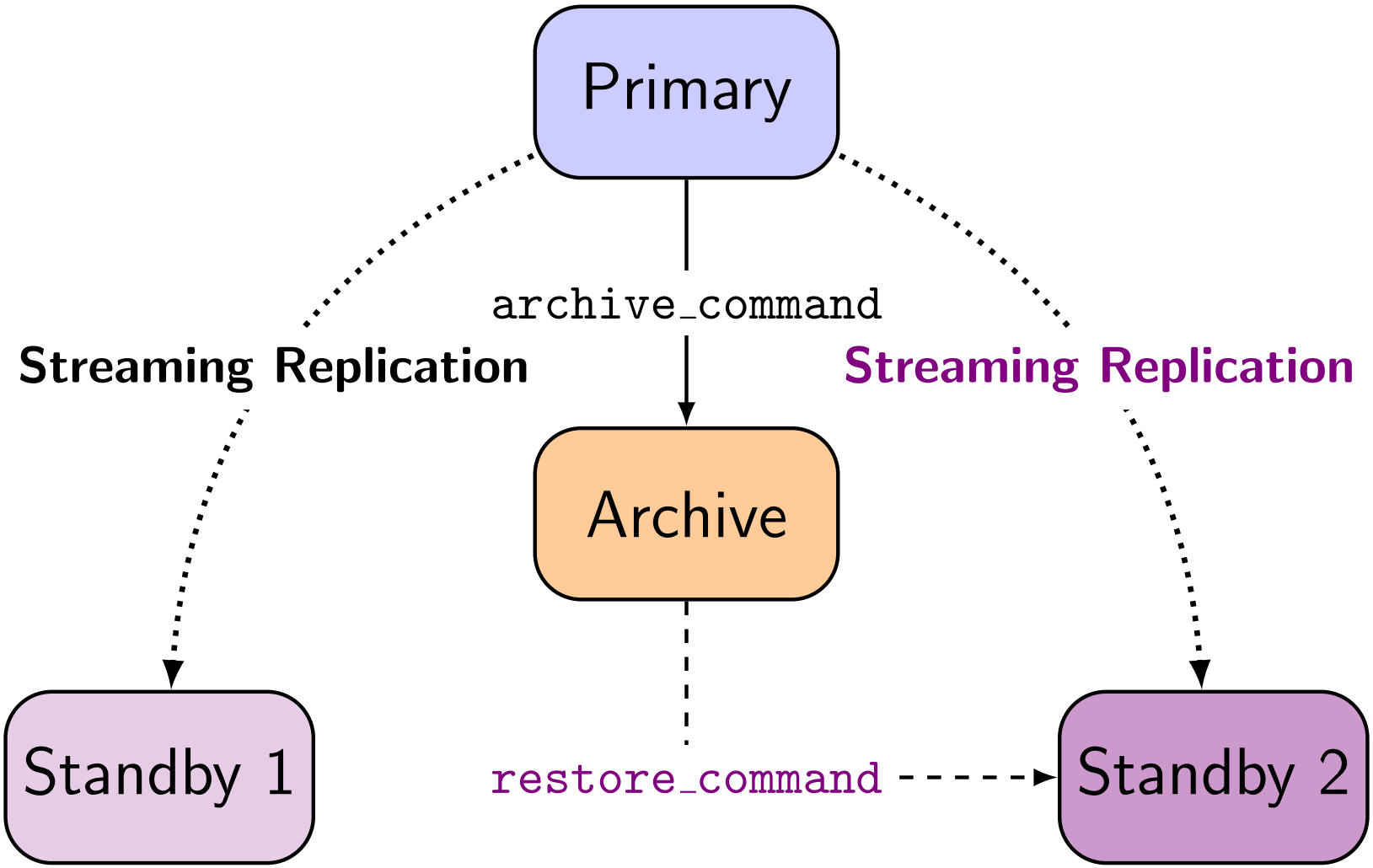
Transactions

```
BEGIN;
  CREATE INDEX ... ON ...;
  EXPLAIN ANALYZE SELECT ...;
ROLLBACK;
```

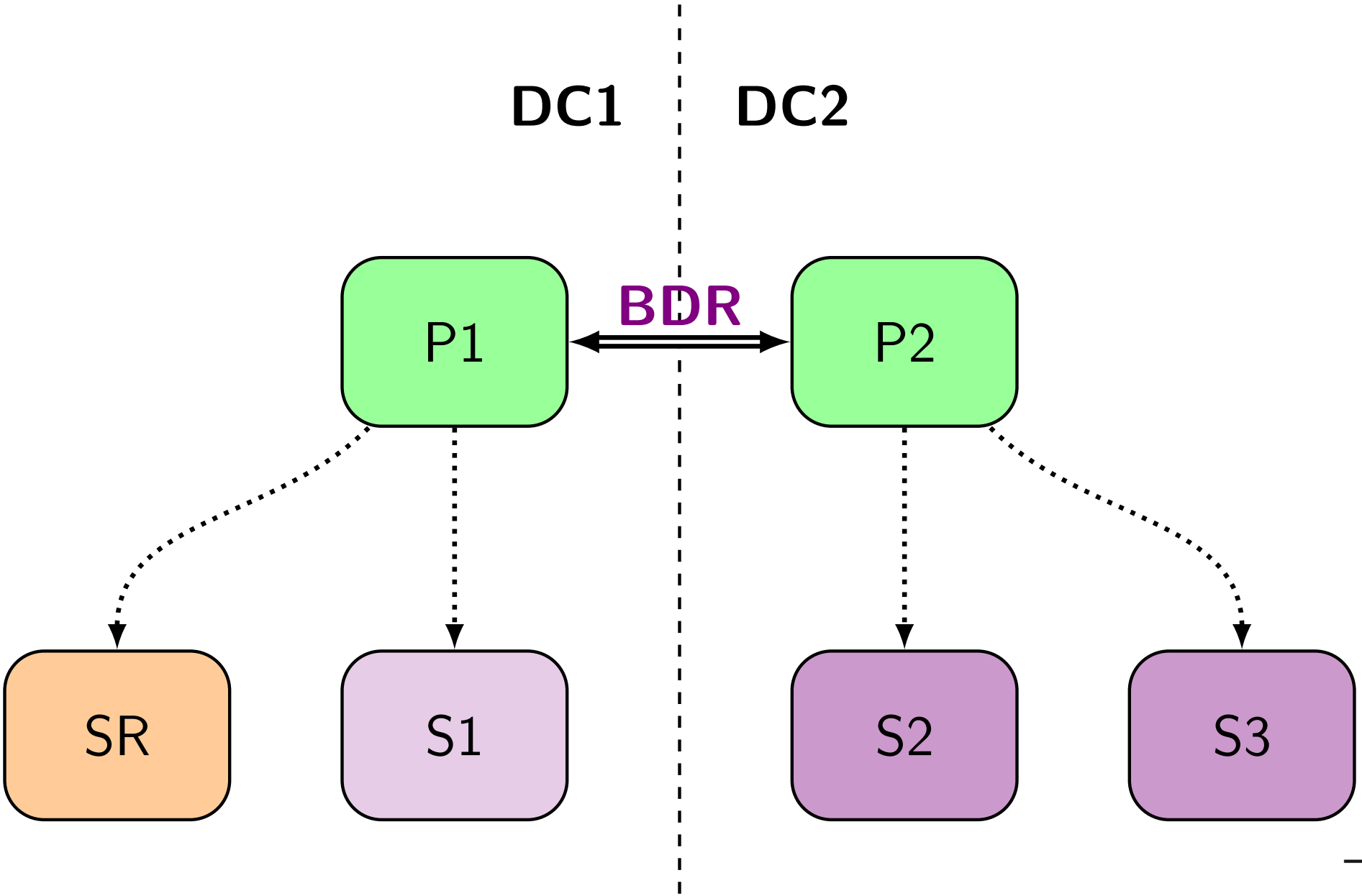
Backups cohérents à chaud

Disponibilité des services et des données

Disponibilité des services et des données



Disponibilité des services et des données



Cohérence des données

Type de données et contraintes

Types et Intégrité des données

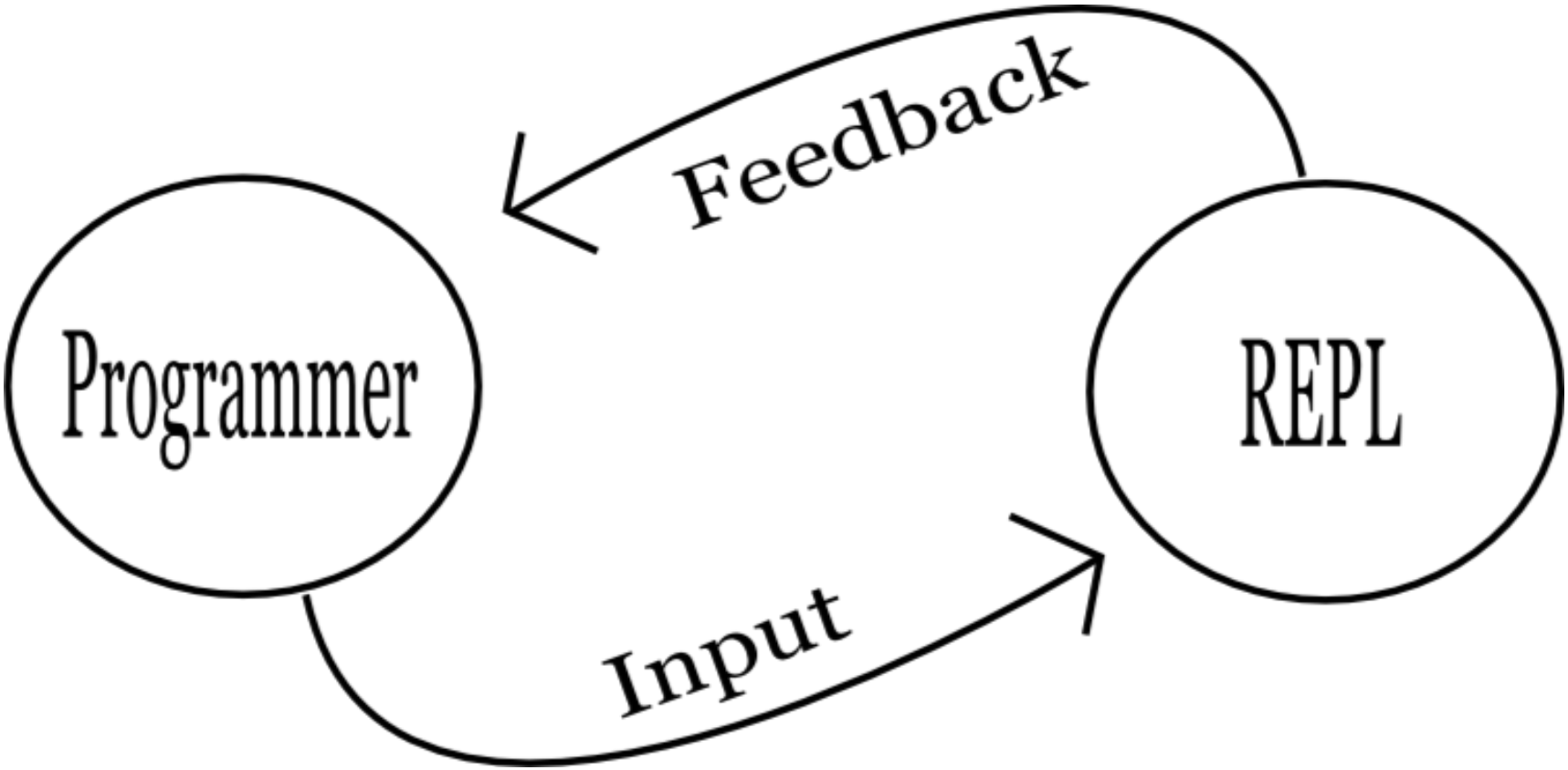
- Integer
- Arbitrary precision numbers, UUID
- Floating point
- Character, Text
- Bytea, bitstring
- Date/Time, Time Zones
- Boolean
- Enum, Arrays, Composite Types, Range Types
- Point, Line Segments, Boxes
- Paths, Polygons, Circles
- Inet, CIDR, Macaddr
- JSON, XML

Type de données et contraintes

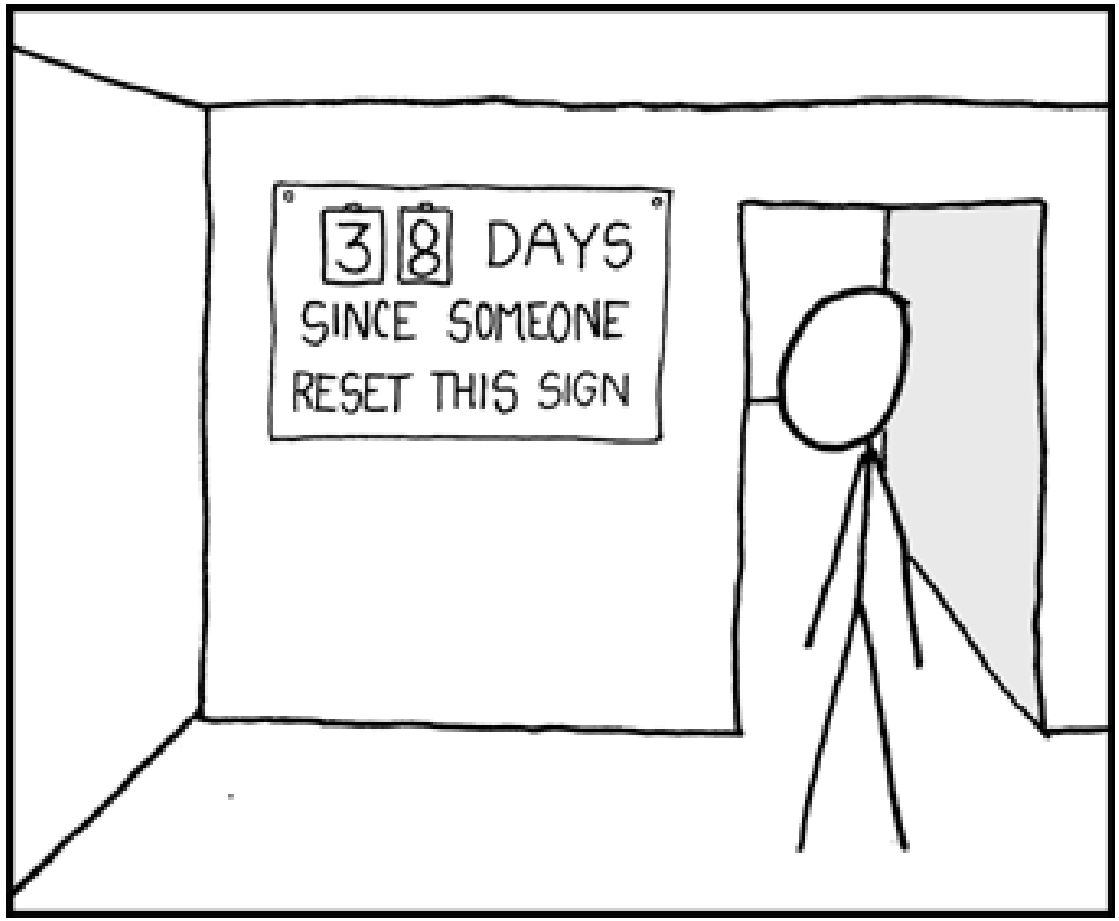
```
CREATE TABLE reservation
(
  room      text,
  professor text,
  during    period,
  EXCLUDE USING gist
  (   room with =,
    during with &&
    )
);

CREATE TABLE circles (
  c circle,
  EXCLUDE USING gist (c WITH &&)
);
```

La console interactive psql



A simple project



SQL: we start with DDLs

Joe Celko: 80% of the job is to define the schema

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)
```

We take a very simple model for the presentation

```
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);
```

Testing data

Let's take some measures as if they came out of our counter, starting at 0, and with a *reset* in there. In that example, the global usage measured is $40 + 60 = 100$.

```
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)
```

Aside: PostgreSQL knows about arrays

```
select array_agg(nb) from measures;
         array_agg
-----------------------------
 {0,10,20,30,40,0,20,30,60}
(1 row)
```

Finding the last counter value before *reset*

Write some SQL here

```
 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 and 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)
```

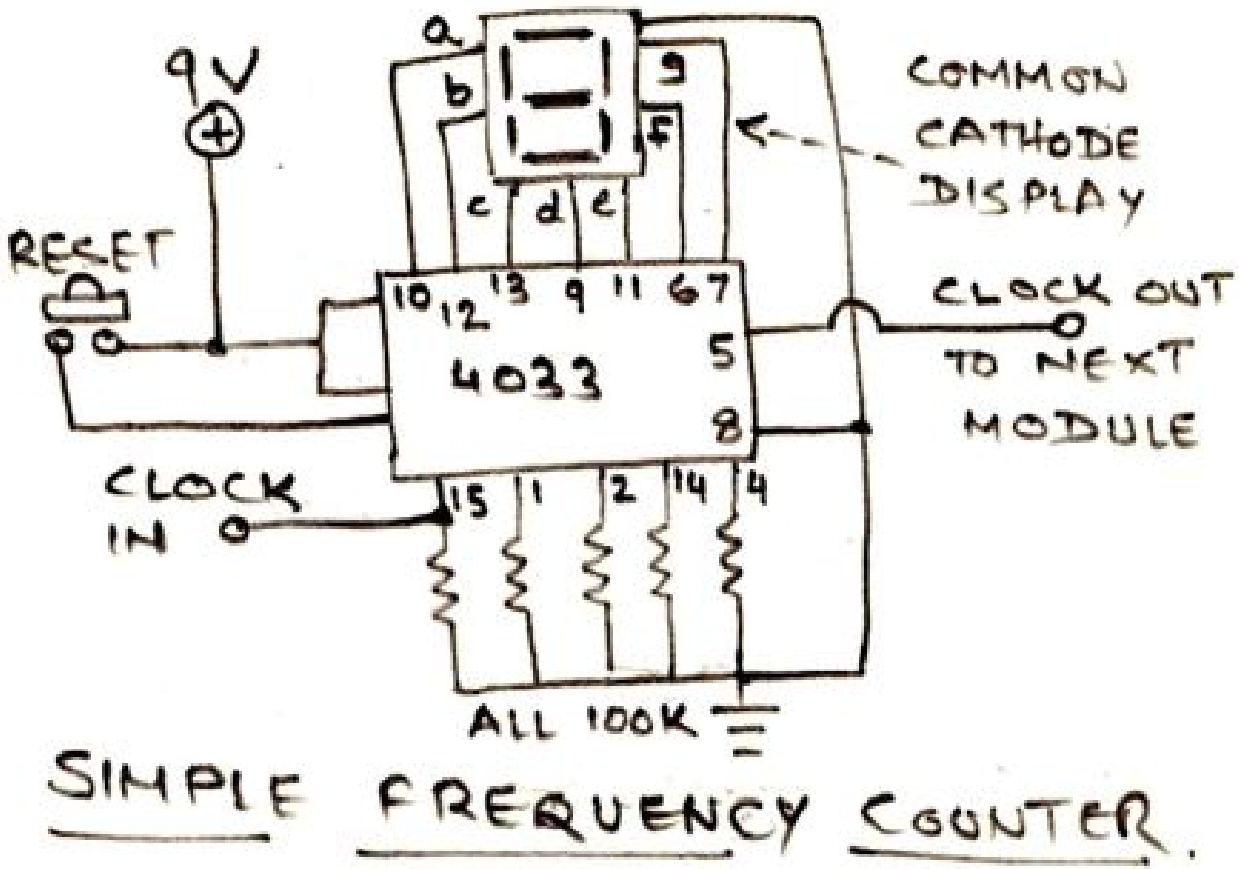
Window Functions and WHERE clause

```
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 to complement WITH

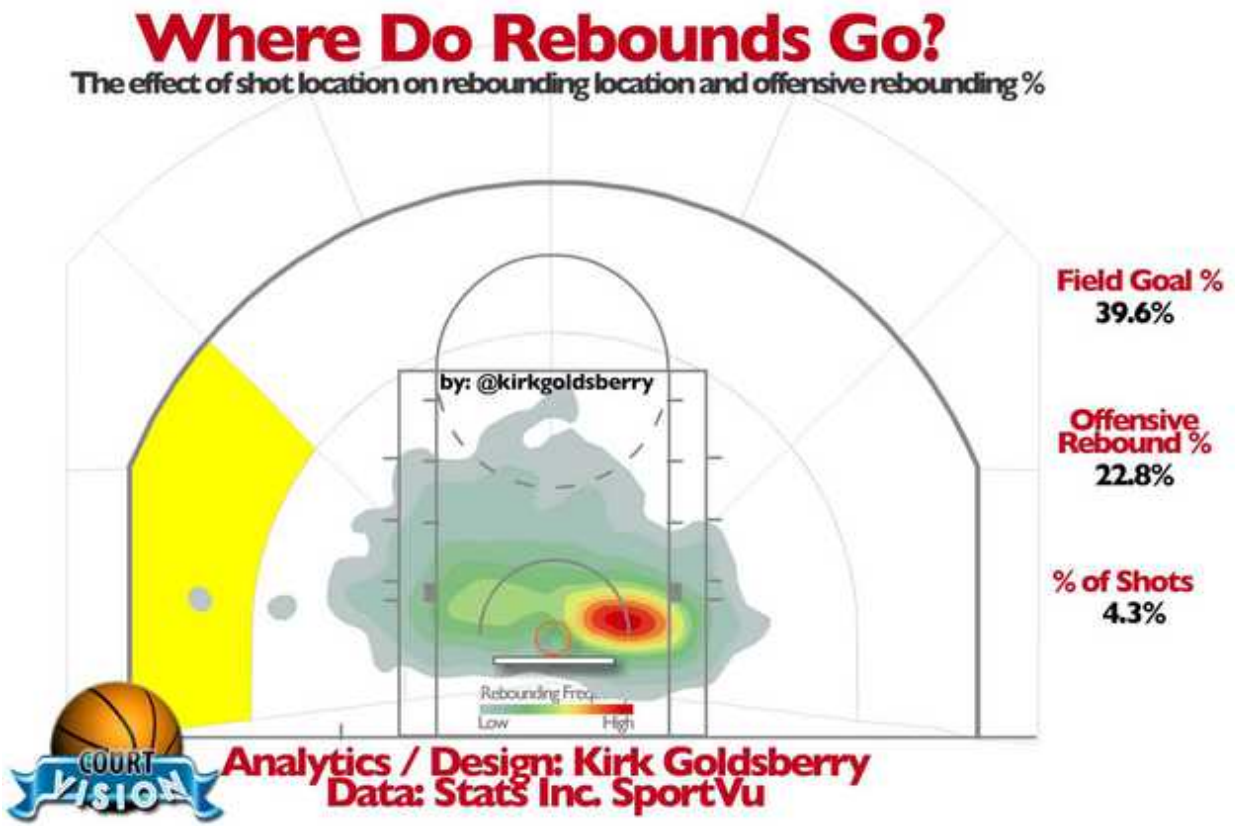
```
with t(tops) as (
  select case when lead(nb) over w < nb then nb
              when lead(nb) over w is null then nb
              else null
          end as max
    from measures
  window w as (order by tick)
)
select sum(tops) from t;
 sum
-----
 100
(1 row)
```

Getting usage from the counter: done. SQL. 9 lines.



Structured Query Language

NBA Games Statistics



NBA Games Statistics

An interesting factoid: the team that recorded the fewest defensive rebounds in a win was the 1995-96 Toronto Raptors, who beat the Milwaukee Bucks 93-87 on 12/26/1995 despite recording only 14 defensive rebounds.

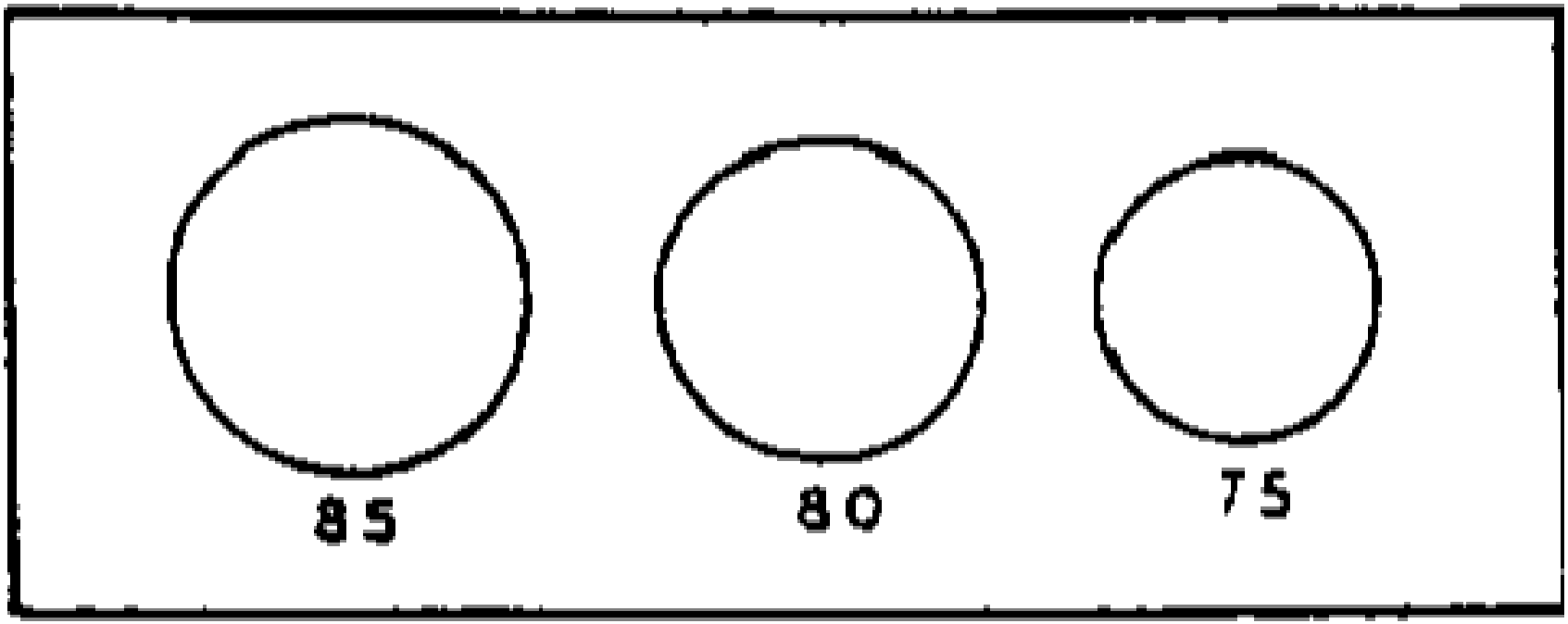
PostgreSQL Data Mining

```
with stats(game, team, drb, min) as (
    select ts.game, ts.team, drb, min(drb) over ()
      from team_stats ts
           join winners w on w.id = ts.game
                         and w.winner = ts.team
)
select game.date::date,
       host.name || ' -- ' || host_score as host,
       guest.name || ' -- ' || guest_score as guest,
       stats.drb as winner_drb
  from stats
       join game on game.id = stats.game
       join team host on host.id = game.host
       join team guest on guest.id = game.guest
 where drb = min;
```

PostgreSQL Data Mining

```
-[ RECORD 1 ]-------------------------
date       | 1995-12-26
host       | Toronto Raptors -- 93
guest      | Milwaukee Bucks -- 87
winner_drb | 14
-[ RECORD 2 ]-------------------------
date       | 1996-02-02
host       | Golden State Warriors -- 114
guest      | Toronto Raptors -- 111
winner_drb | 14
-[ RECORD 3 ]-------------------------
date       | 1998-03-31
host       | Vancouver Grizzlies -- 101
guest      | Dallas Mavericks -- 104
winner_drb | 14
-[ RECORD 4 ]-------------------------
date       | 2009-01-14
```

Un histogramme tout en SQL



Histogramme tout en SQL

```
with drb_stats as (
    select min(drb) as min,
           max(drb) as max
      from team_stats
),
     histogram as (
   select width_bucket(drb, min, max, 9) as bucket,
          int4range(min(drb), max(drb), '[]') as range,
          count(*) as freq
     from team_stats, drb_stats
 group by bucket
 order by bucket
)
 select bucket, range, freq,
        repeat('*', (freq::float / max(freq) over() * 30)::int
   from histogram;
```

Histogramme tout en SQL

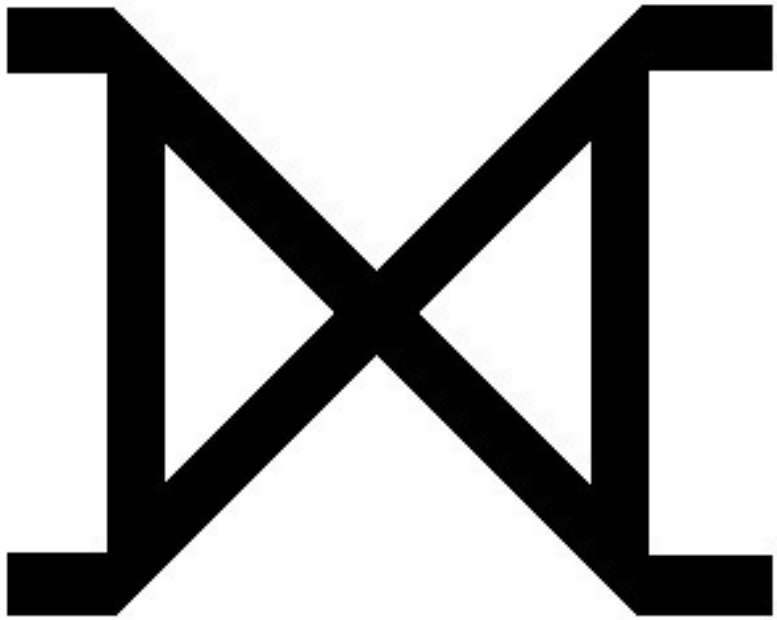
```
 bucket |   range   | freq  |                bar
--------+-----------+-------+-----------------------------------
      1 | [10,15)   |    52 |
      2 | [15,20)   |  1363 | **
      3 | [20,25)   |  8832 | **********
      4 | [25,30)   | 20917 | ************************************
      5 | [30,35)   | 20681 | ************************************
      6 | [35,40)   |  9166 | **********
      7 | [40,45)   |  2093 | ***
      8 | [45,50)   |   247 |
      9 | [50,54)   |    20 |
     10 | [54,55)   |     1 |
(10 rows)

Time: 53.570 ms
```

Savoir contourner les ORMs avec les wCTE

```
with queue as (
   insert into queue (extension)
        select id
          from extension
         where shortname = $1
  returning id, extension
)
select q.id, e.id as ext_id,
       e.fullname, e.uri, e.description
  from      queue q
       join extension e on q.extension = e.id;
```

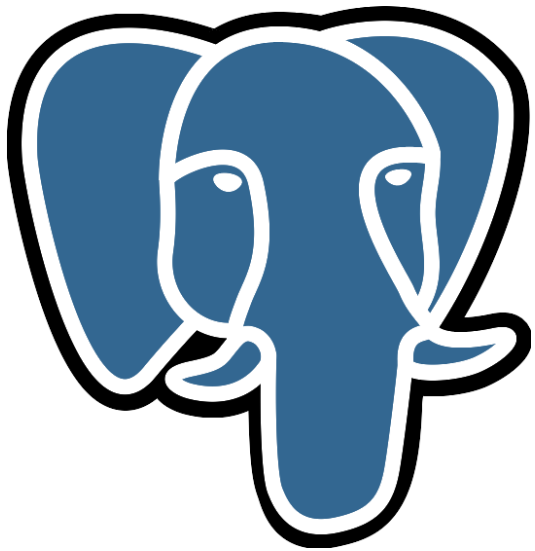
Les jointures avec PostgreSQL



Les jointures avec PostgreSQL

```
WITH upd AS (
    UPDATE target t
       SET counter = t.counter + s.counter,
      FROM source s
     WHERE t.id = s.id
 RETURNING s.id
)
INSERT INTO target(id, counter)
     SELECT id, sum(counter)
       FROM source s LEFT JOIN upd t USING(id)
      WHERE t.id IS NULL
   GROUP BY s.id
   RETURNING t.id
```

PostgreSQL is YeSQL!



Questions?

Now is the time to ask!