Utiliser PostgreSQL en 2014

PHP Tour, 2014

Dimitri Fontaine dimitri@2ndQuadrant.fr
@tapoueh

23 juin 2014

Dimitri Fontaine

Principal Consultant at 2ndQuadrant

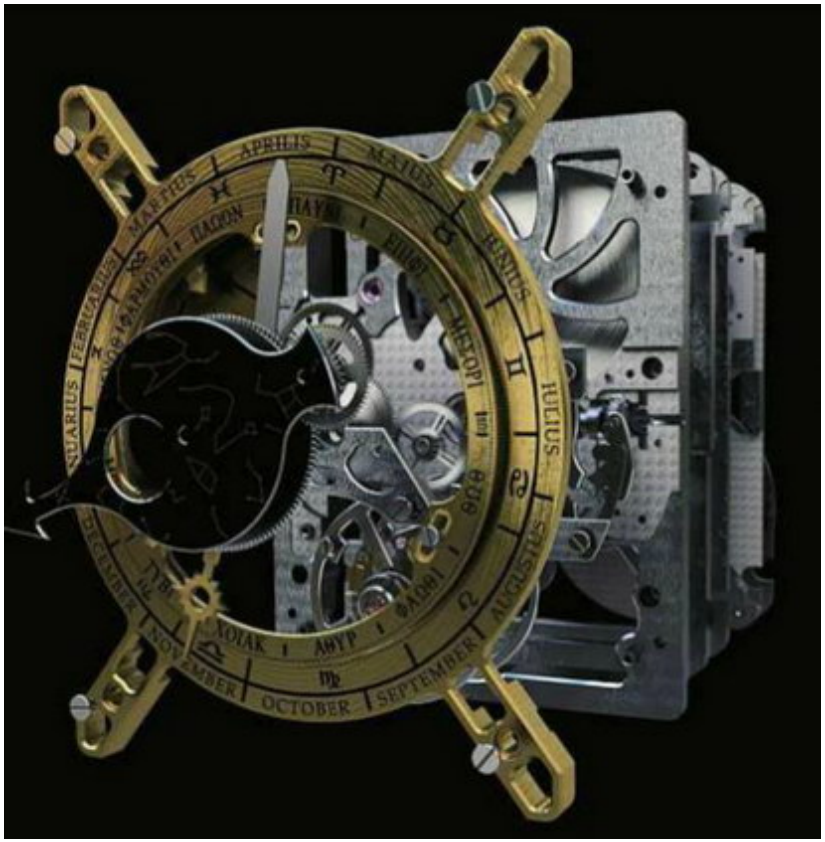
PostgreSQL Major Contributor

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

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;
```

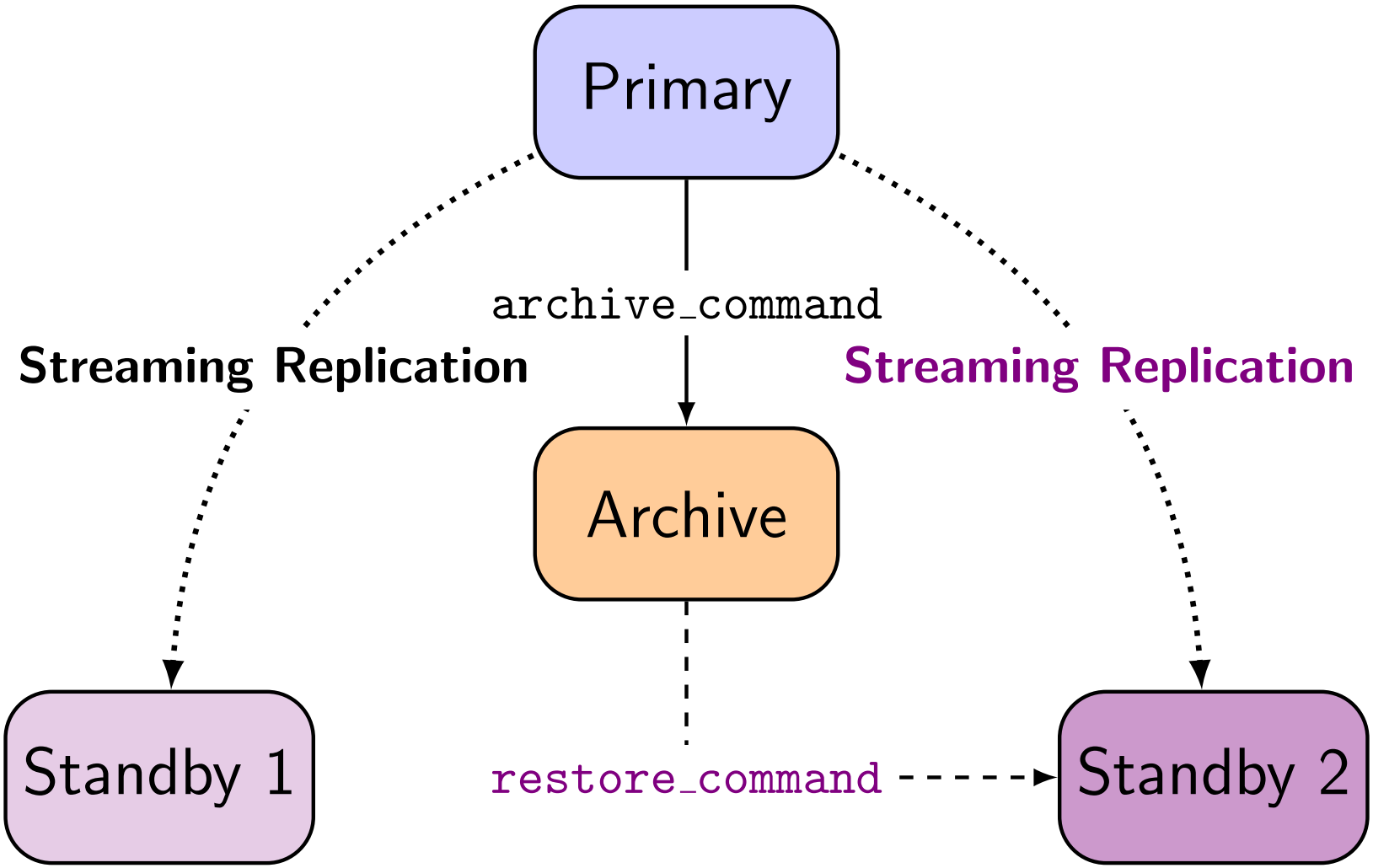
Transactions

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

Backups cohérents à chaud

Disponibilité des services et des données

Disponibilité des services et des données

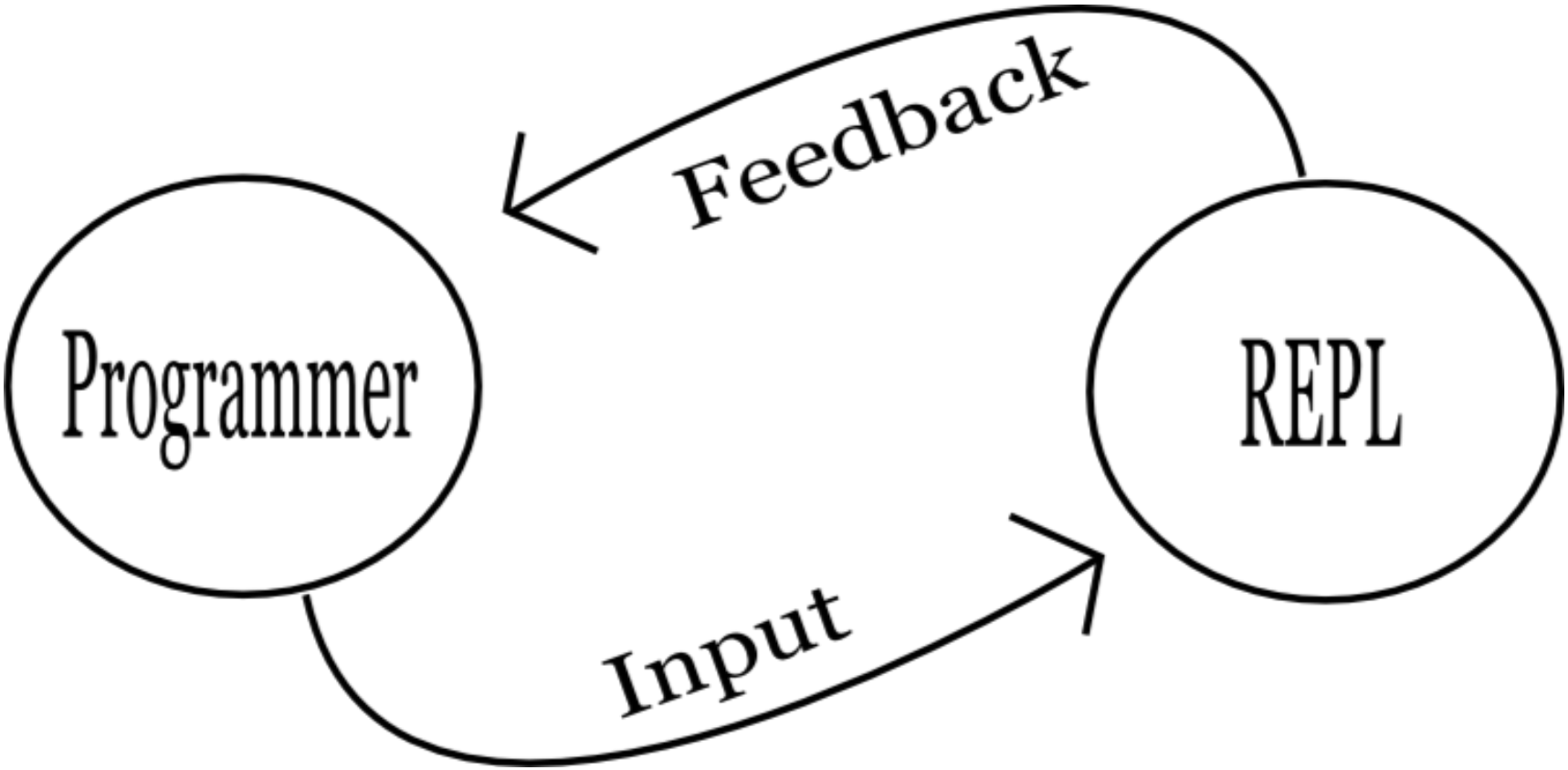


Cohérence des données

Type de données et contraintes

```
CREATE TABLE reservation (
    during tsrange NOT NULL,
    EXCLUDE USING gist (during WITH &&)
);

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

La console interactive psql



Types de données avancés

```
select iprange, locid
  from geolite.blocks
 where iprange >>= '91.121.37.122';
```

```
          iprange           | locid
----------------------------+-------
 91.121.0.0-91.121.159.255 |    75
(1 row)

Time: 1.220 ms
```

IP Ranges, ip4r, Geolocation

Géolocalisation et méta-données obtenues sur une jointure

```
select *
  from geolite.blocks
  join geolite.location
       using(locid)
 where iprange
           >>=
       '74.125.195.147';
```

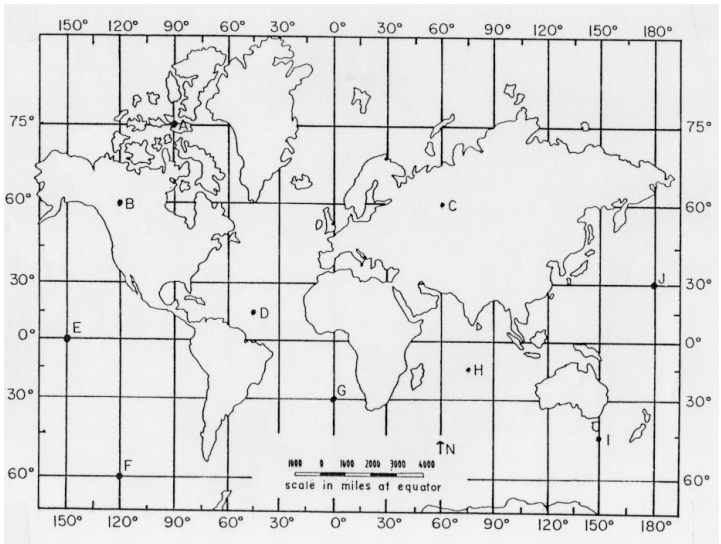
```
-[ RECORD 1 ]----------------------
locid      | 2703
iprange    | 74.125.189.24-74.125.
country    | US
region     | CA
city       | Mountain View
postalcode | 94043
location   | (-122.0574,37.4192)
metrocode  | 807
areacode   | 650

Time: 1.335 ms
```

Structured Query Language

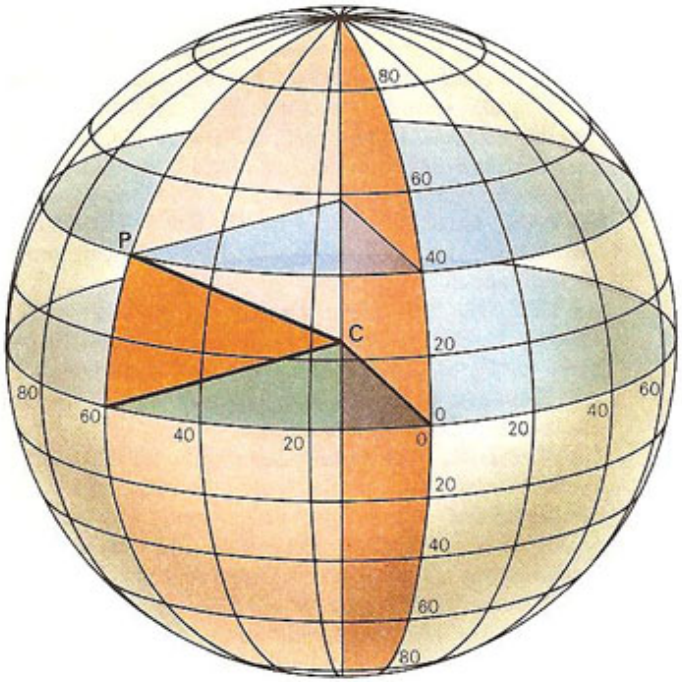
How Far is The Nearest Pub

The `point` datatype is in-core

```
CREATE TABLE pubnames
(
  id   bigint,
  pos  POINT,
  name text
);
```



Eearh Distance



How Far is The Nearest Pub, in Miles please.

```
create extension cube;
create extension earthdistance;
```

```
  select name,
   pos <@> point(-6.25,53.34) miles
    from pubnames
order by pos <-> point(-6.25,53.34)
   limit 3;
```

```
  name   | miles
---------+-------
 Ned's   |  0.06
 Sub Lo  |  0.07
 O'Neil  |  0.12
(3 rows)

Time: 1.335 ms
```

Geolocation

Geolocalisation et meta-données

```
with geoloc as
 (
  select location as l
    from location
    join blocks using(locid)
   where iprange
         >>=
         '212.58.251.195'
 )
  select name,
         pos <@> l miles
    from pubnames, geoloc
order by pos <-> l
   limit 10;
```

```
          name          | miles
------------------------+-------
 Blue Anchor            | 0.299
 Dukes Head             | 0.360
 Blue Ball              | 0.337
 Bell (aka The Rat)     | 0.481
 on the Green           | 0.602
 Fox & Hounds           | 0.549
 Chequers               | 0.712
 Sportsman              | 1.377
 Kingswood Arms         | 1.205
 Tattenham Corner       | 2.007
(10 rows)

Time: 3.275 ms
```

Trigrams

Trigrammes et autocompletion

```
  select actor
    from products
   where actor % 'fran'
order by actor <-> 'fran'
   limit 10;
```

```
     actor
---------------
 FRANK HAWKE
 FRANK BERRY
 FRANK POSEY
 FRANK HAWKE
 FRANCES DEE
 FRANK LEIGH
 FRANCES DAY
 FRANK FOSTER
 FRANK HORNE
 FRANK TOMEI
(10 rows)

Time: 2.960 ms
```

Tags et tableaux

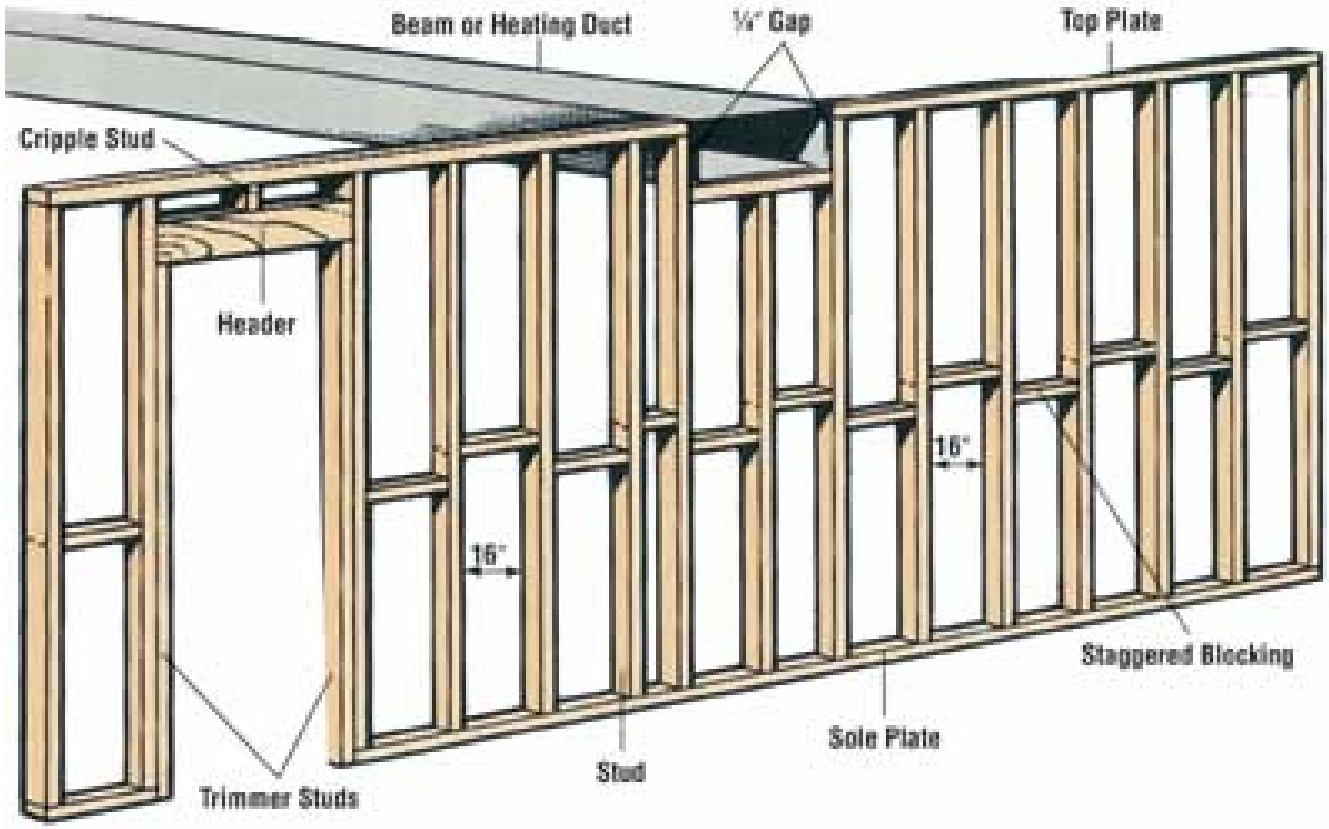
PostgreSQL sait utiliser des tableaux

```
  select tt.tid,
         array_agg(tags.rowid) tags
    from tags
         join tid_tag tt
           on tags.rowid = tt.tag
group by tt.tid
   limit 3;
```

```
 tid |   tags
-----+-----------
   1 | {1,2}
   2 | {3,4}
   3 | {5,6,7,8}
(3 rows)

time: 942.074 ms
```

Types de données composites

SQL en 2014 : les *window functions*



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;
```

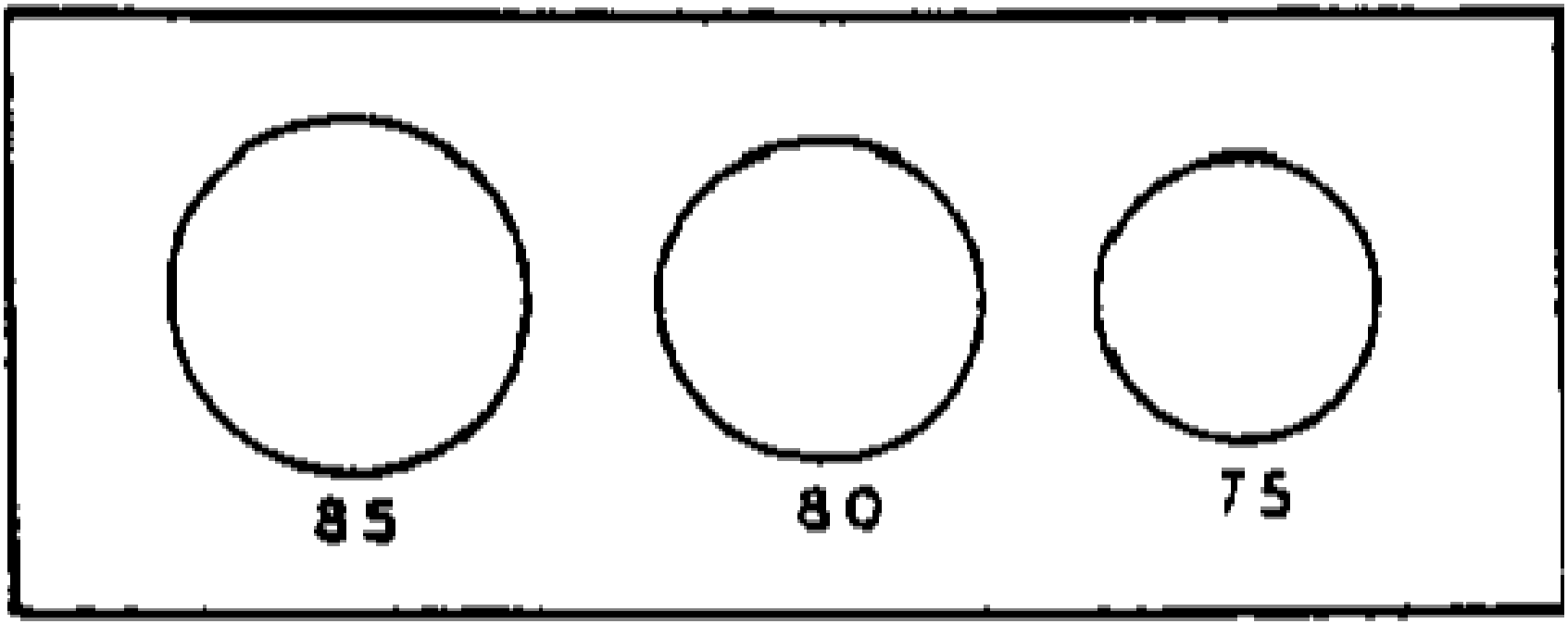
```
 tick | nb | lead
------+----+------
    1 |  0 |   10
    2 | 10 |   20
    3 | 20 |   30
    4 | 30 |   40
    5 | 40 |    0
    6 |  0 |   20
    7 | 20 |   30
    8 | 30 |   60
    9 | 60 |
(9 rows)
```

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)
```

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;
```

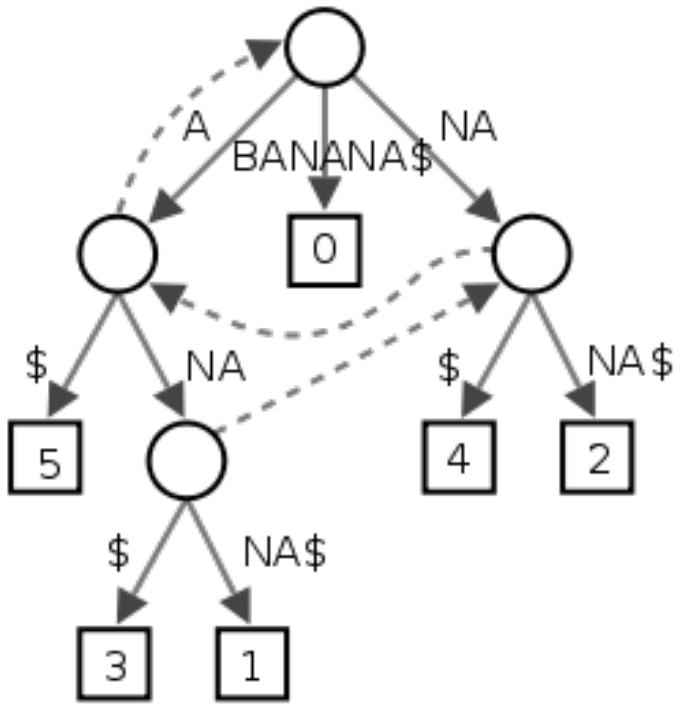
Histogramme tout en SQL

```
 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
```

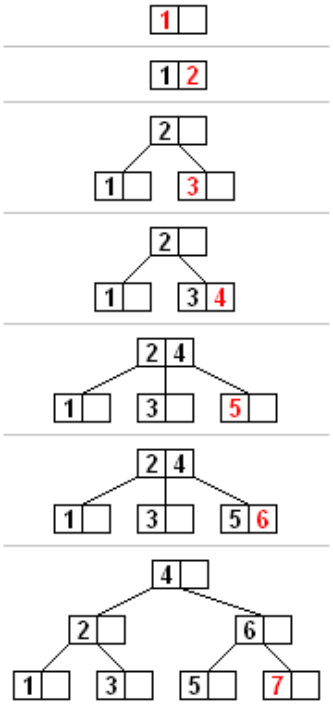
Savoir contourner les ORMs avec les wCTE

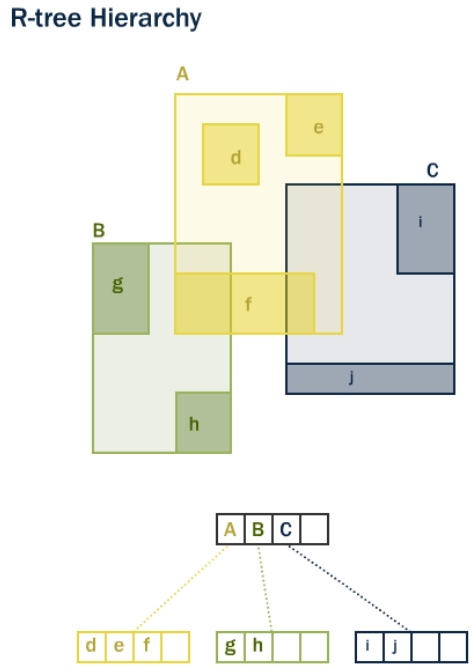
```
with queue as (
   insert into queue (extension)
        select id
          from extension
         where shortname = $1
  returning id, extension
)
select q.id, e.id as ext_id,
       e.fullname, e.uri, e.description
  from      queue q
       join extension e on q.extension = e.id;
```

Les Indexes dans PostgreSQL

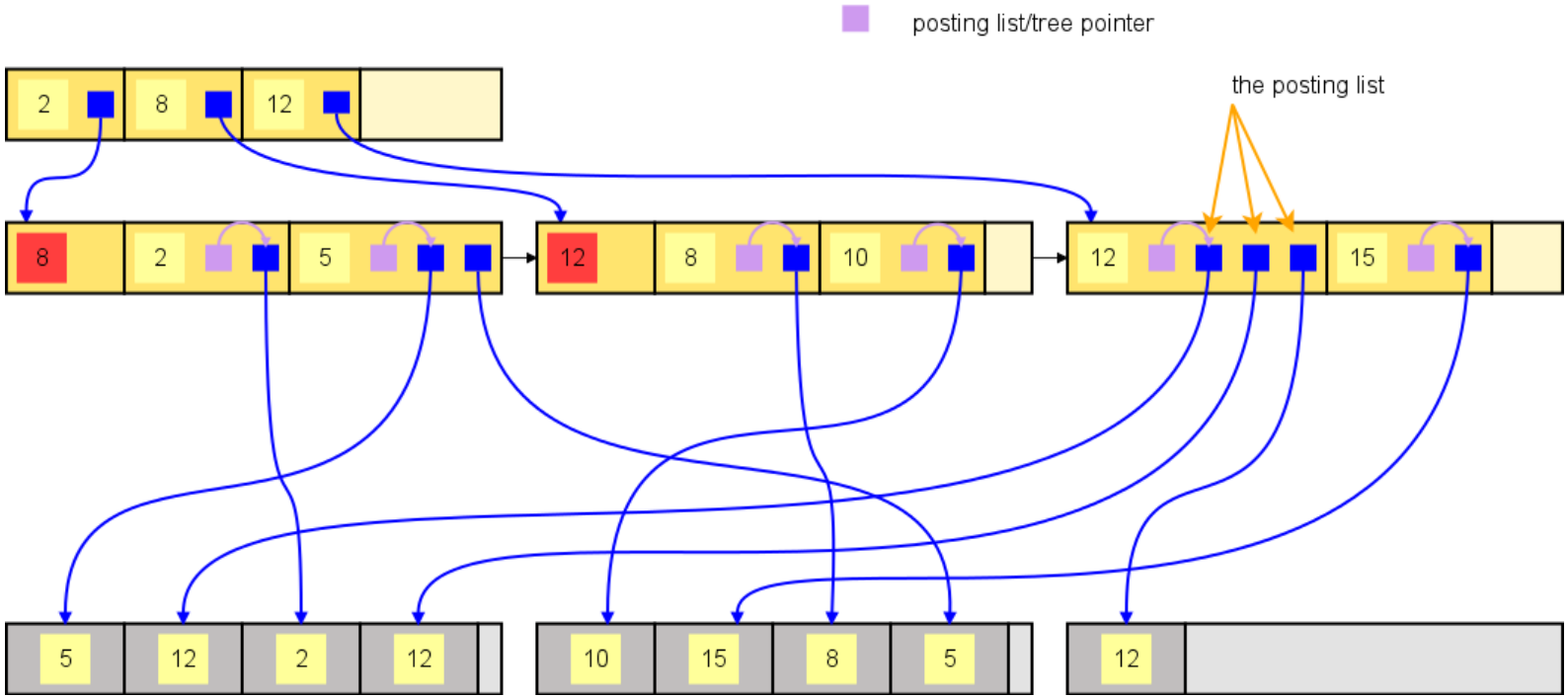


Les Indexes dans PostgreSQL

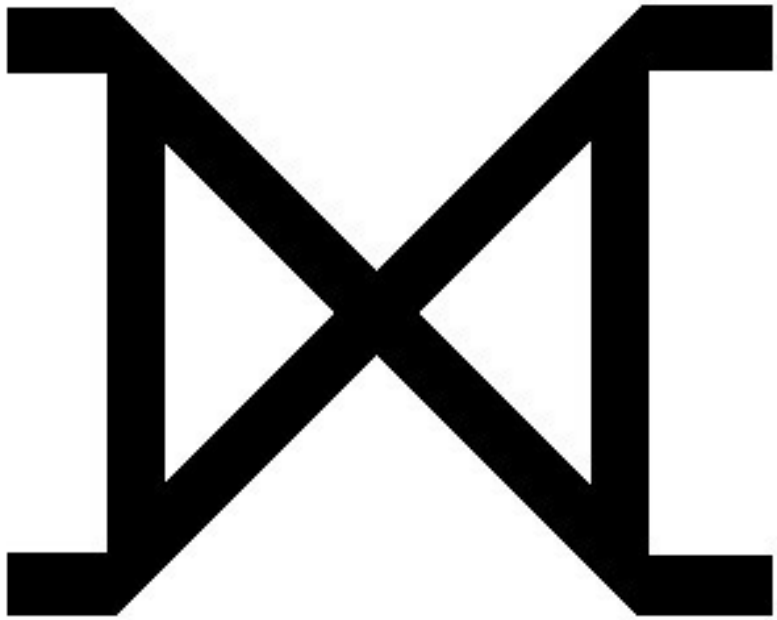




Les Indexes dans PostgreSQL



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!