

POSTGRESQL IS YESQL!

Chti JUG 2018

Dimitri Fontaine @tapoueh

CitusData

22 Mai 2018



Dimitri Fontaine

POSTGRESQL MAJOR CONTRIBUTOR

- **pgloader**
- CREATE EXTENSION
- CREATE EVENT TRIGGER
- *Bi-Directional Réplication*
- apt.postgresql.org

Postgre**SQL**



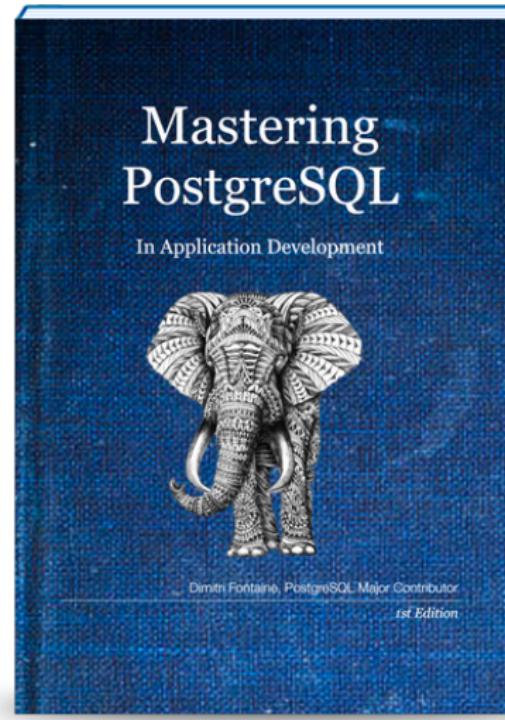


I wrote a book!

PROMO CODE: CHTI-JUG

Mastering PostgreSQL in Application Development teaches SQL to developers: learn to replace thousands of lines of code with simple queries.

<http://MasteringPostgreSQL.com>





Antikythera mechanism

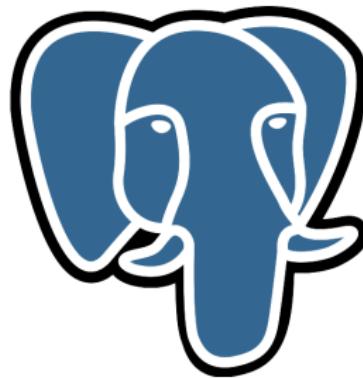


Antikythera mechanism

Relational Database Management System



POSTGRESQL IS YESQL!

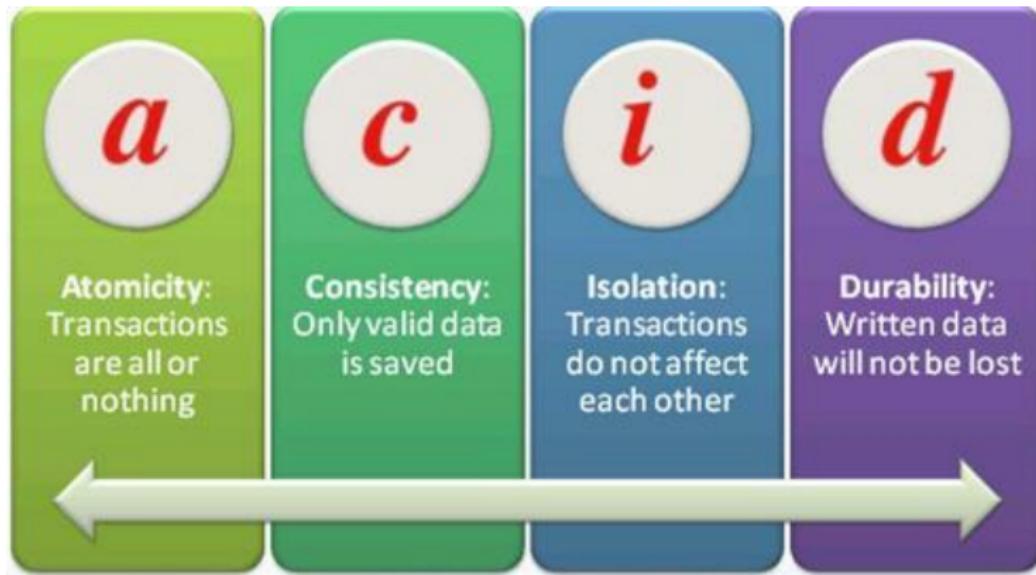


POSTGRESQL IS YESQL!



Not Only SQL







D is for Durability





A is for Atomic, that is,
Transactions

rollback



Transactions

```
BEGIN;  
    DROP TABLE critical_data;  
ROLLBACK;
```



I is for Isolated, that is,
Transactions



You don't want backups, you want data loss recovery!



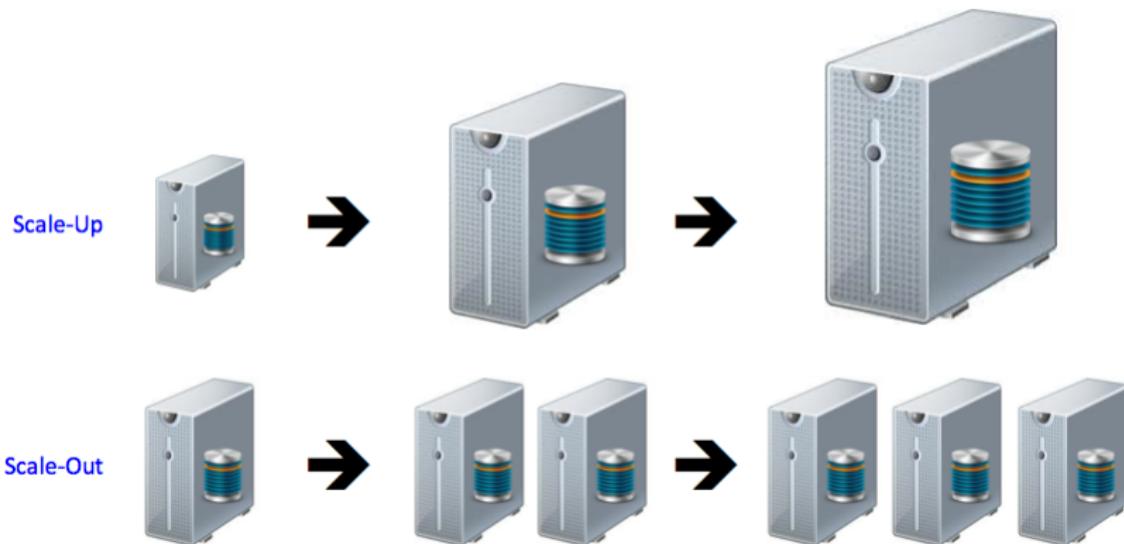


C is for Consistency

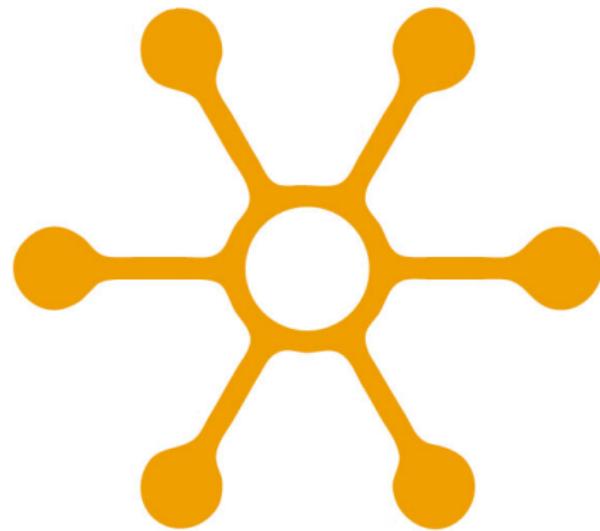




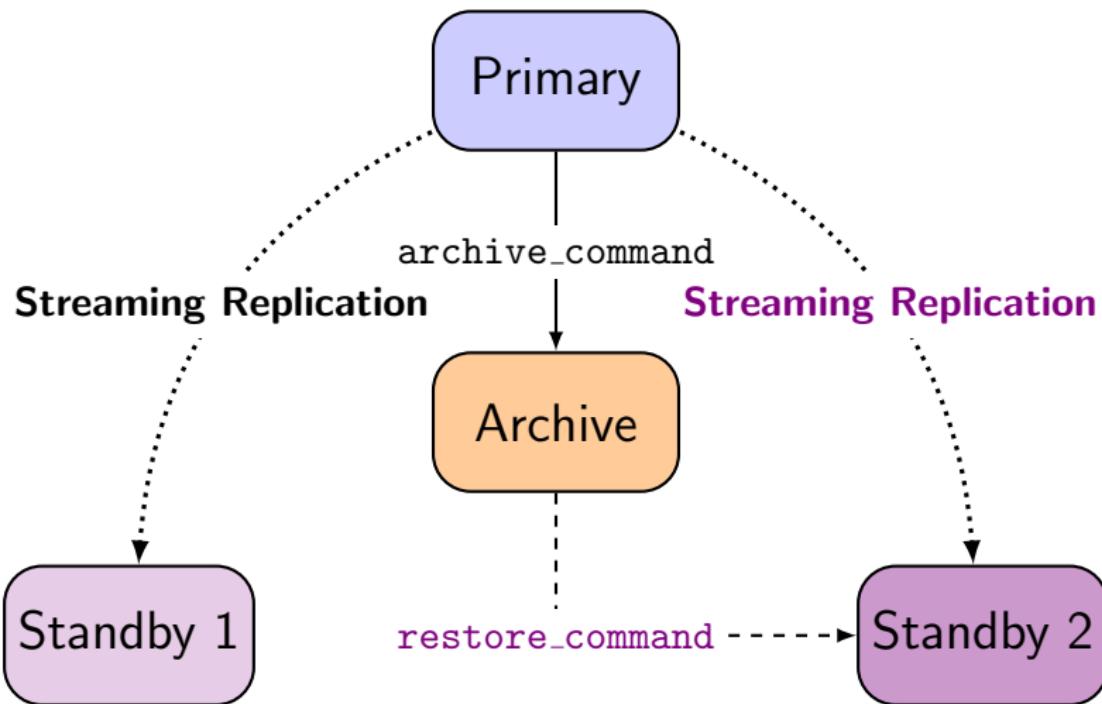
Scaling: up or out?



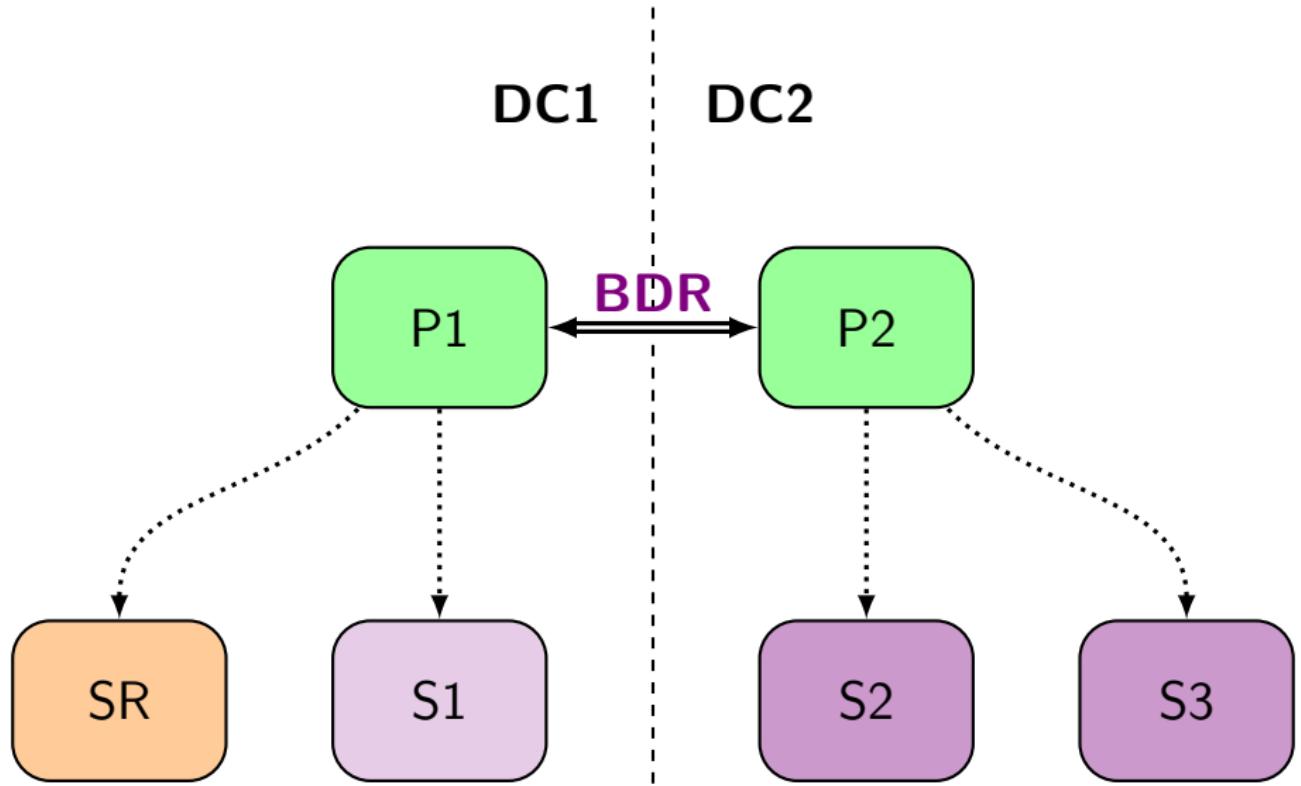
High Availability of Services and Data



High Availability: Consistency and Durability

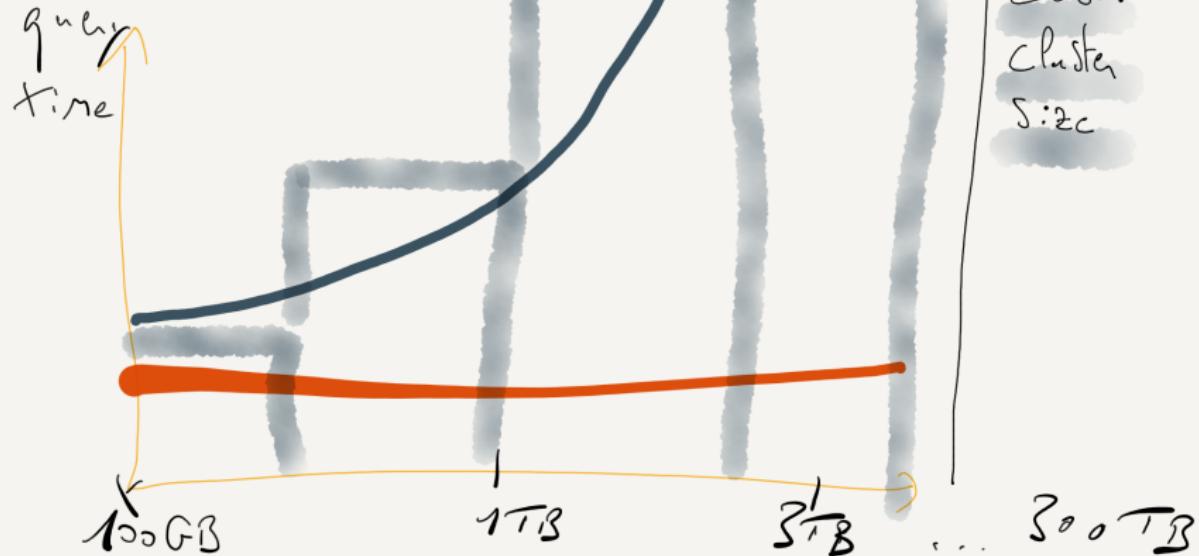


High Availability





Horizontal Scaling: Sharding



Five sharding data models and which is right?



Citus Data blog post at <https://www.citusdata.com/blog/2017/08/28/five-data-models-for-sharding/>

- Sharding by geography
- Sharding by entity id
- Sharding a graph
- Time partitioning
- Depends...



Consistency





Data integrity First Layer: data type

- 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

Advanced data types and constraints



```
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 &&)
);
```

STRUCTURED QUERY LANGUAGE



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()



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)



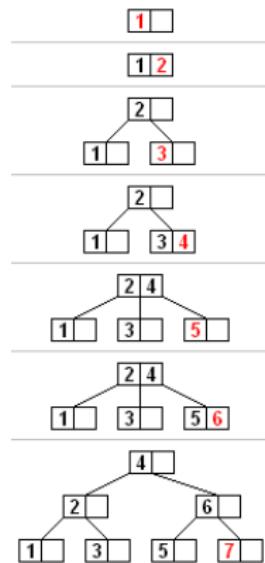
PostgreSQL is *highly extensible*





Balanced Tree, the default index type

- Built for speed
- *unique* concurrency tricks
- Balanced
- support function: `cmp`
- operators: `<= < = > >=`



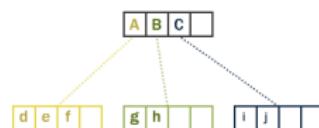
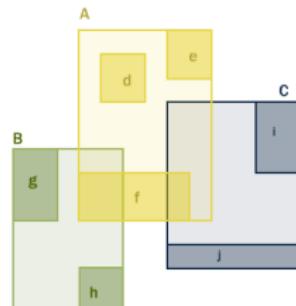


Generalized Index Search Tree

GiST or the Indexing API

- Built for comfort
- Balanced
- API: consistent, same, union
- API: penalty, picksplit
- API: compress, decompress
- operators: @> <@ && @@ = &< &>
<<| ...

R-tree Hierarchy

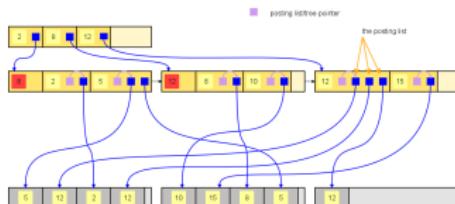


Generalized Inverted iNdex



Indexing several pointers per value, inversed cardinality

- Built for Text Search and Arrays
- Balanced
- API: compare, consistent
- API: extractValue,
extractQuery
- operators: @> <@ && =





PostgreSQL Full Text Search

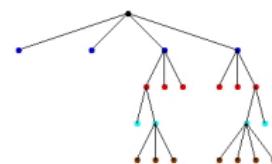
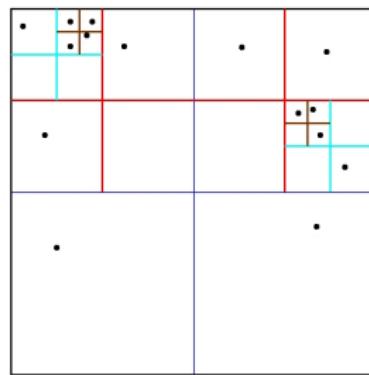




Unbalanced index, used for **GIS**

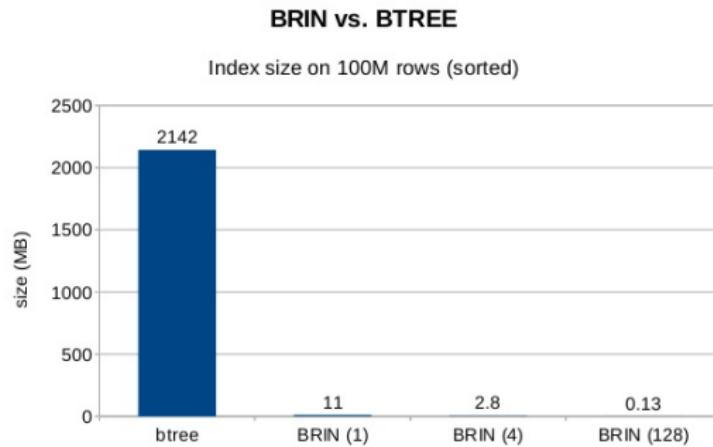


Quadtree





Block Range Index for Big Data



STRUCTURED QUERY LANGUAGE





PostgreSQL allows using SQL and JOINs to match IP4R with geolocation.

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



IP Ranges, ip4r, Geolocation



PostgreSQL allows using SQL and JOINs to match IP4R with geolocation.

```
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

Geolocation: ip4r meets earthdistance



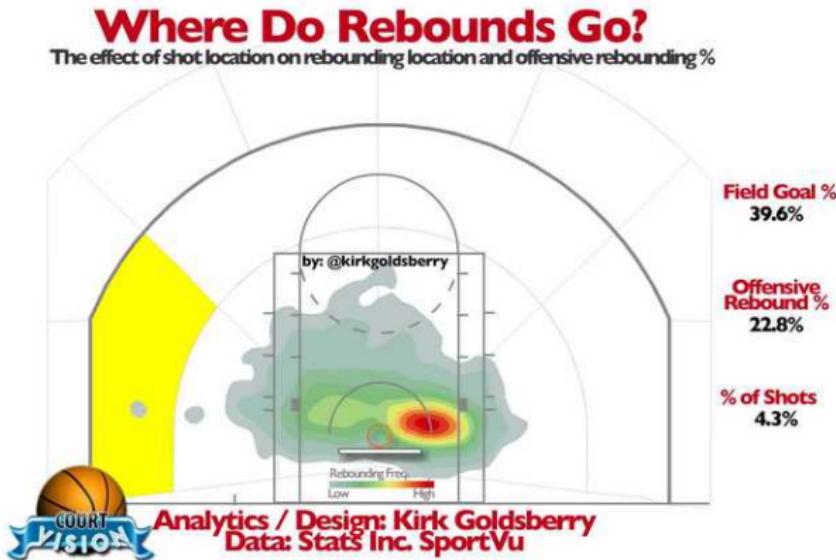


Some pubs nearby... some place...

```
with geoloc as
(
  select location as l
    from location
   join blocks using(locid)
  where iprange
        >>=
      '212.58.251.195'
)
select name,
       pos <@> 1 miles
  from pubnames, geoloc
 order by pos <-> 1
 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
Tottenham Corner	2.007
(10 rows)	

Time: 3.275 ms





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;
```

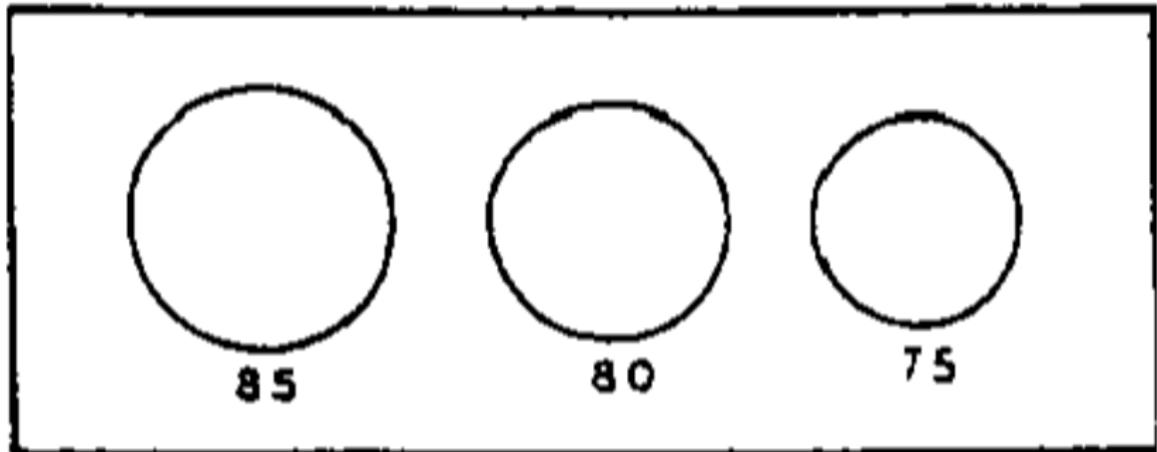


PostgreSQL Data Mining

```
-[ 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  
host      | New York Knicks -- 128  
guest     | Washington Wizards -- 122  
winner_drb | 14
```



Pure SQL Histogram drawing





Pure SQL Histogram drawing

```
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;
```



Pure SQL Histogram drawing

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)

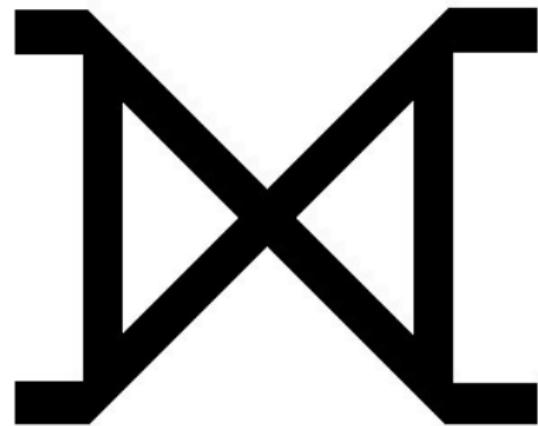
Time: 53.570 ms



Writable CTE

```
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;
```

PostgreSQL Joins





PostgreSQL Joins

```
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 Lateral Joins

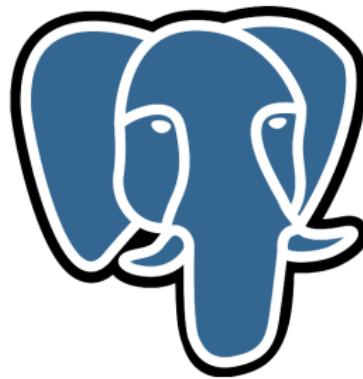


Fetch the 10 most recent news for each topic (**Top-N**)

```
SELECT n.*  
FROM subscriptions s  
JOIN LATERAL (SELECT *  
              FROM news n  
              WHERE n.topic = s.topic  
              ORDER BY n.created DESC  
              LIMIT 10  
            ) top_news ON (true)  
WHERE s.user_id = ?  
ORDER BY n.created DESC  
LIMIT 10
```



POSTGRESQL IS YESQL!



Questions?



Now is the time to ask!

