PostgreSQL is YeSQL!

PostgreSQL Major Contributor

Dimitri Fontaine

- Extensions
- Event Triggers
- pgloader
- pginstall
- prefix, preprepare, pgstaging...

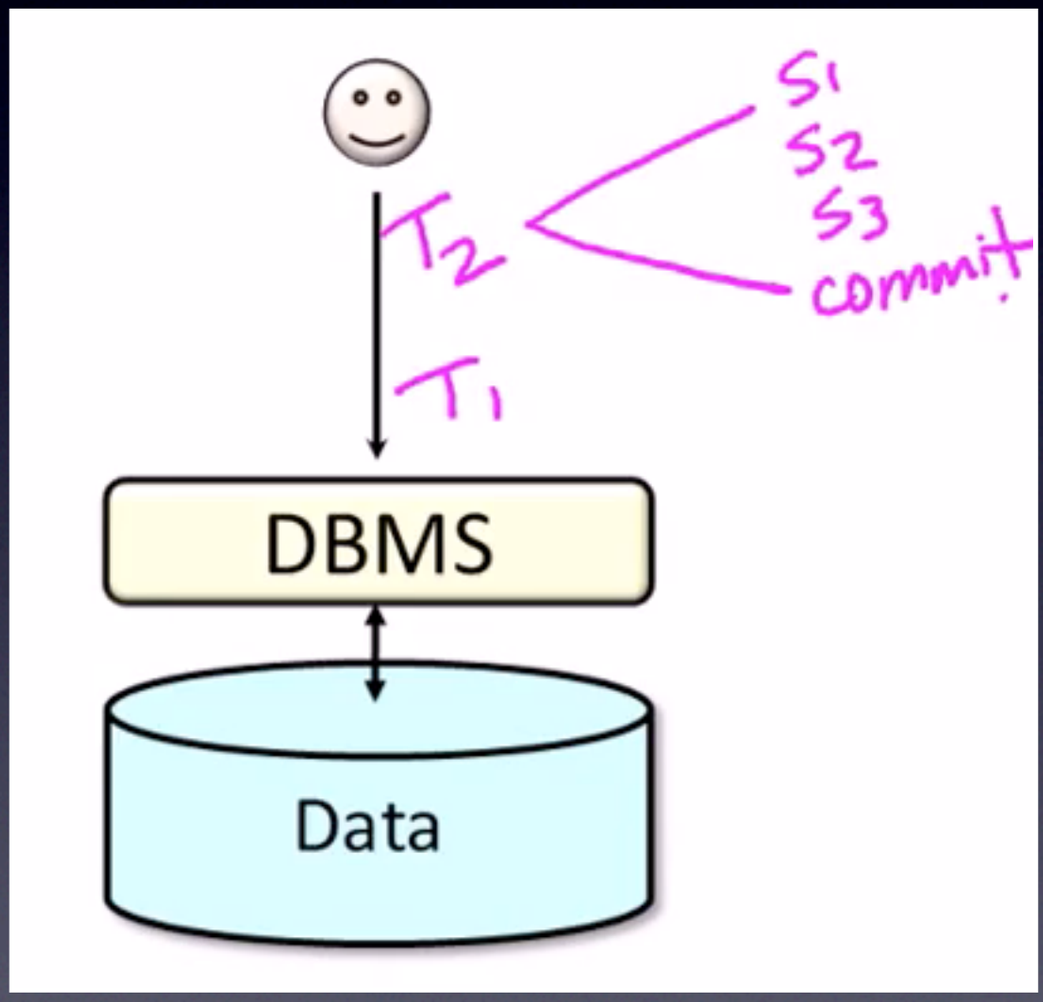
Relational DataBase System

- Data Access Service
- Concurrent Reads and Writes
- Multiple Version Concurrency Control
- “Typed” Protocol
- API, not storage, not serialization


CAUTION
ACID

ACID

- Atomicity
- Consistency
- Isolation
- Durability



Atomicity

- BEGIN; ... COMMIT;
- BEGIN; ... ROLLBACK;
- Includes DDL
- Consistent Backups
- Online Backups
- Physical or Logical

Consistency

- NOT NULL
- CHECK constraints
- CREATE DOMAIN
- Primary Key, Foreign Key
- Unique
- Triggers, Constraint Triggers

Consistency

- Relational Model is Strongly Typed
- Data Type Input Function
  - date/time field value out of range:

    “0000-00-00”

    “0000-03-19”
- Exclusion Constraints

Exclusion Constraints

```
CREATE TABLE reservation
(
  room      text,
  professor text,
  during    period,

  EXCLUDE USING gist
  (   room with =,
    during with &&
  )
);
```

PostgreSQL Data Types

- 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

PostgreSQL Extensions

- cube
- hstore
- intarray
- ltree
- pg_trgm
- seg
- ip4r
- prefix_range
- pgmp, tinyint
- sha-1, sha-2, md5
- hyperloglog
- and much more

Isolation

- SET TRANSACTION *transaction_mode*
- ISOLATION LEVEL
  - serializable
  - repeatable read
  - read committed

Durability

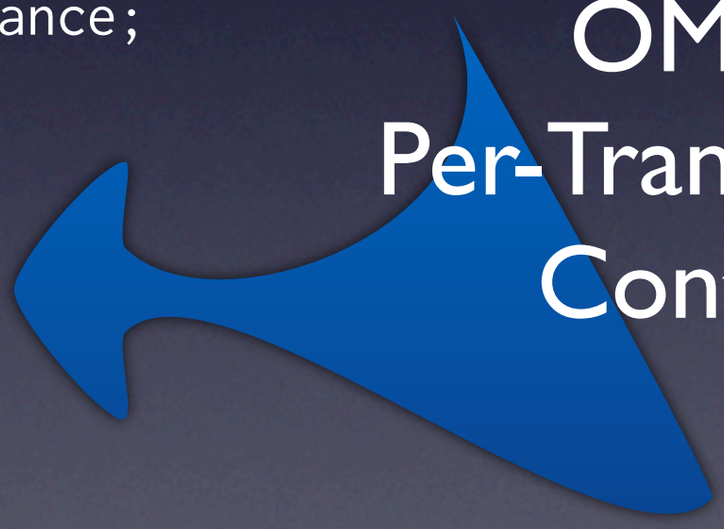
- fsync
- synchronous_commit defaults to on

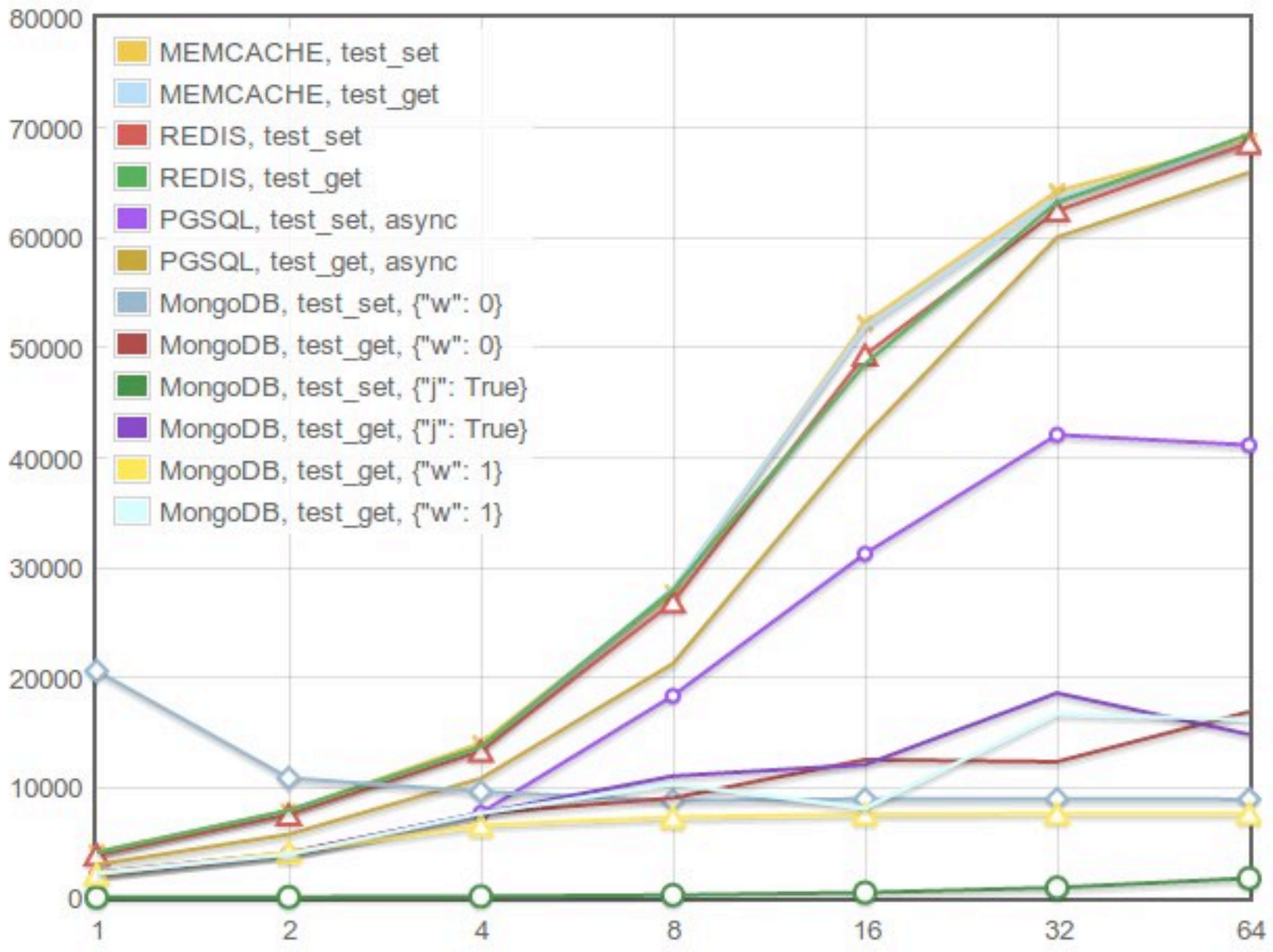
  (off, local, remote_write, on)
- Per-Transaction Control

synchronous_commit

```
SET demo.threshold TO 1000;

CREATE OR REPLACE FUNCTION public.syncrep_important_delta()
  RETURNS TRIGGER
  LANGUAGE PLpgSQL
AS
$$
DECLARE
  threshold integer := current_setting('demo.threshold')::int;
  delta integer := NEW.abalance - OLD.abalance;
BEGIN
  IF delta > threshold
  THEN
    SET LOCAL synchronous_commit TO on;
  END IF;
  RETURN NEW;
END;
$$;
```

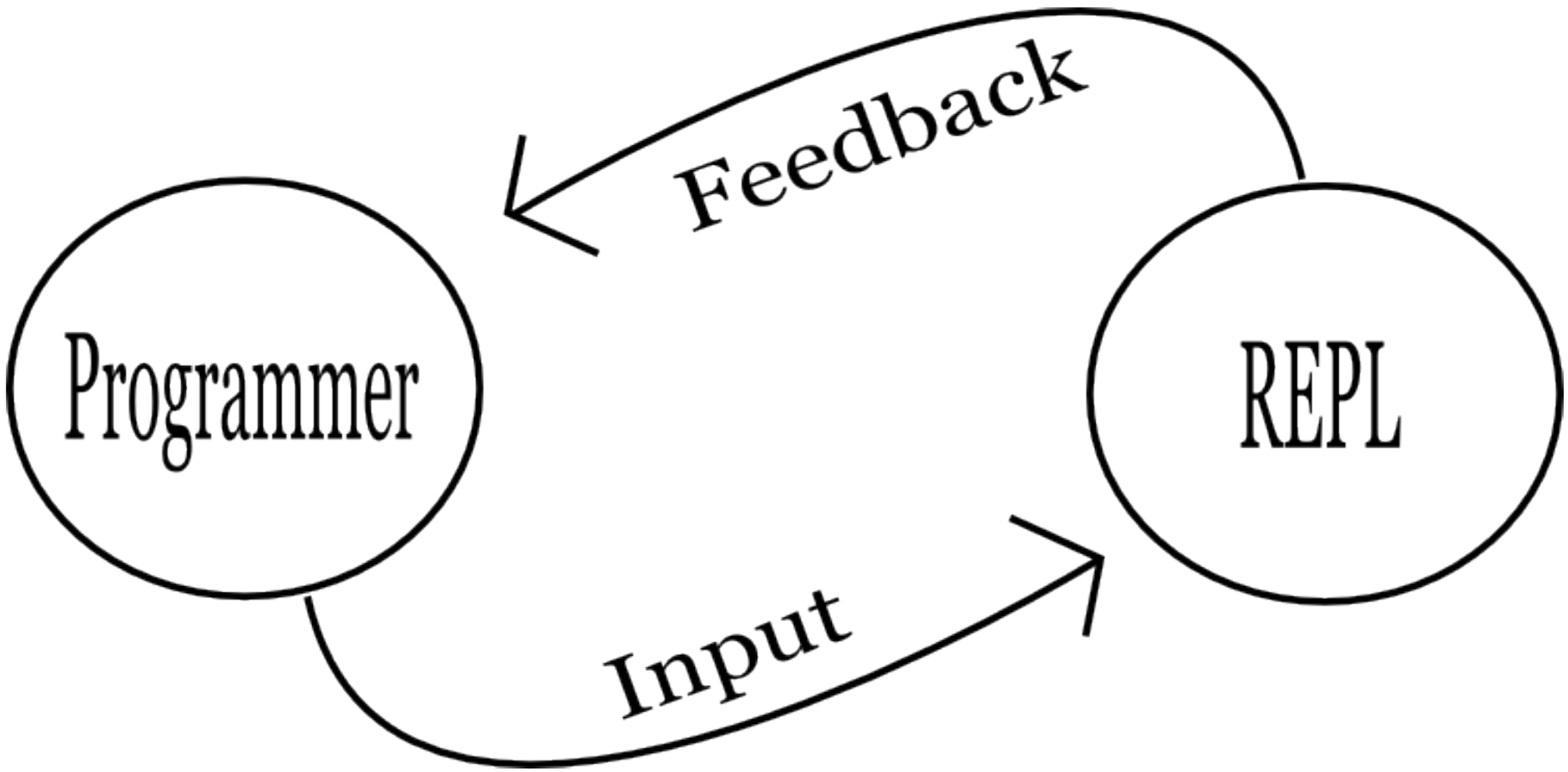




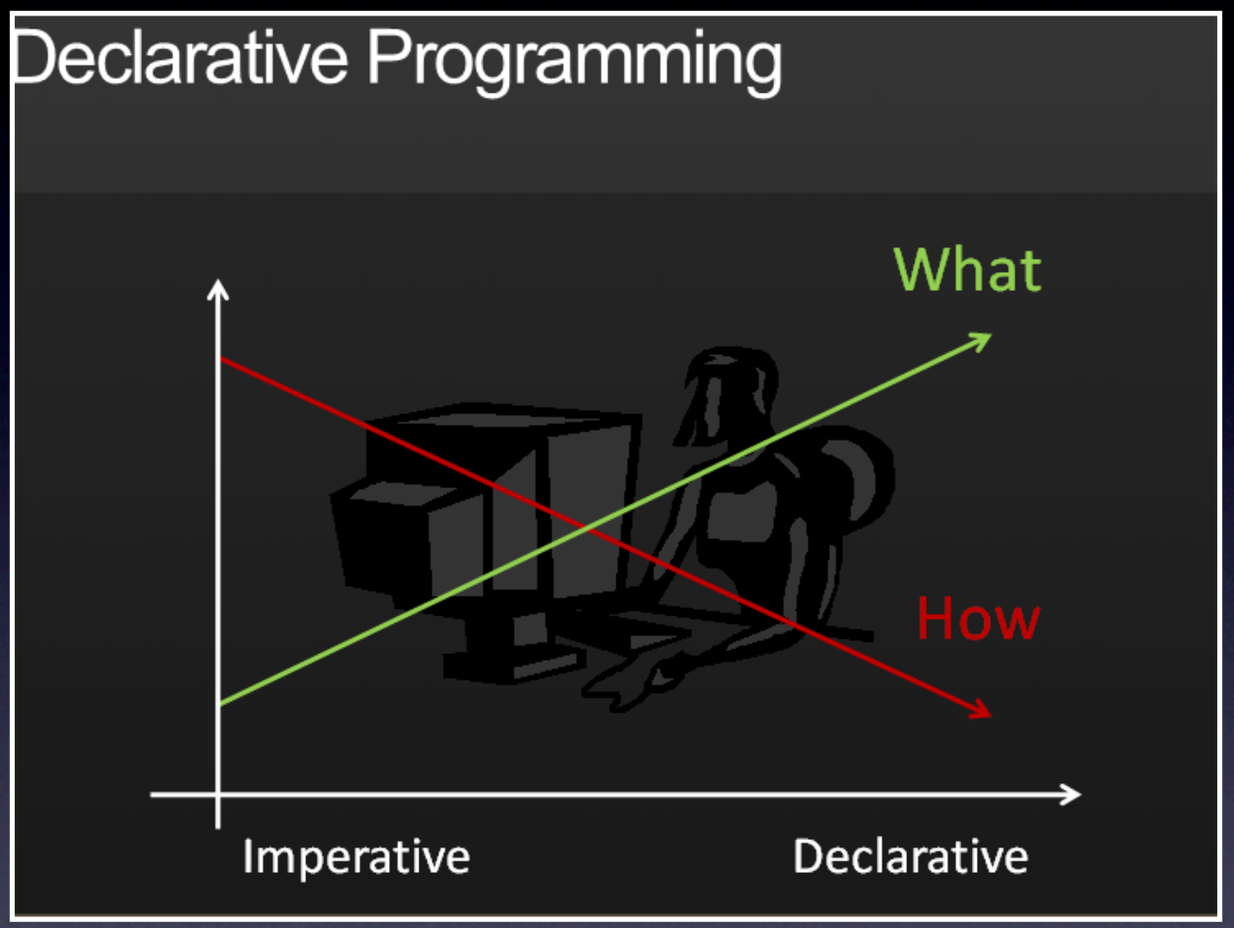
1 Server with 1 to 64 clients, Client(s) and server on separate host
minimum data size: 1188, max size: 2601, average size: 1874

NoSQL GET/SET

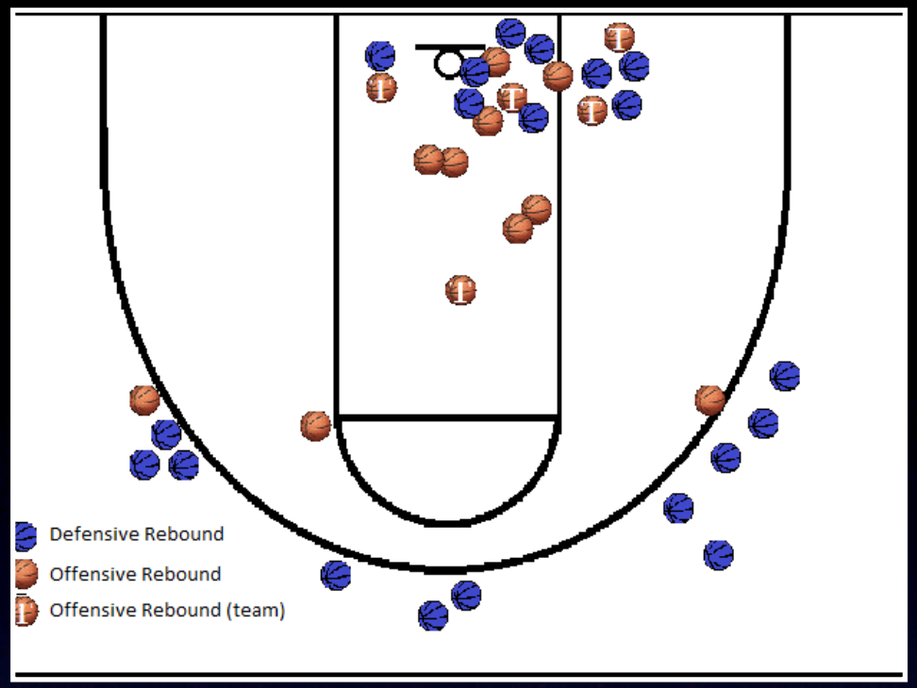

SQL



The Real Power of `psql`



Structured Query



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.

```
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;
```

```
-[ 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
```

PostgreSQL JOINs

- Nested Loop
- Merge Join
- Hash Join
- Semi Join
- Anti Join
- Inner Join
- Outer Join
- Cross Join
- Lateral Join

Window Functions

```
# select x,
          array_agg(x) over (order by x)
    from generate_series(1, 3) as t(x);
```

```
 x | array_agg
---+-----------
 1 | {1}
 2 | {1,2}
 3 | {1,2,3}
(3 rows)
```

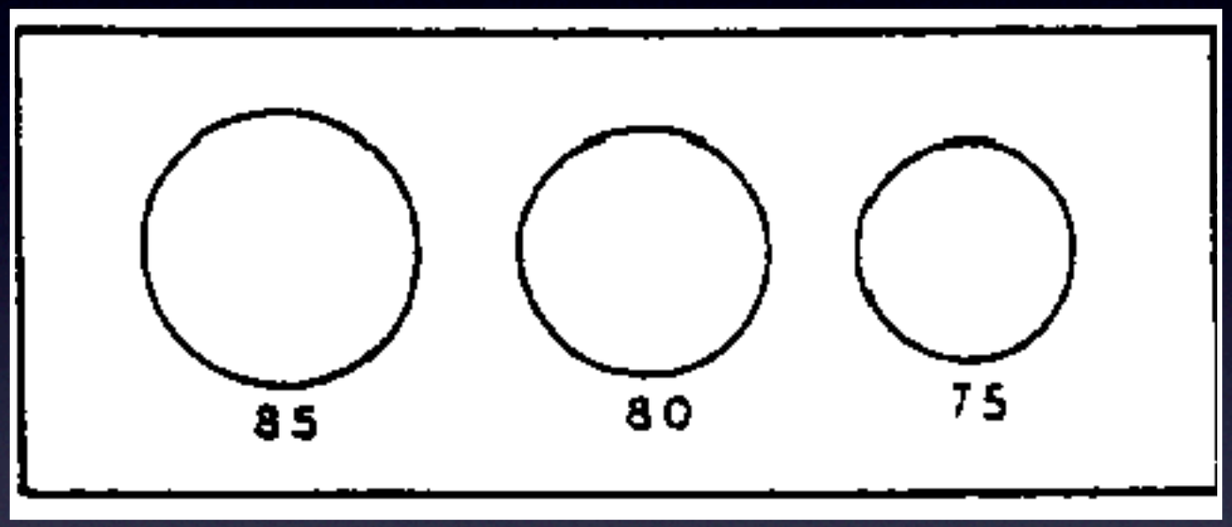
Window Functions

```
# select x,
          array_agg(x) over () as frame,
          sum(x) over () as sum,
          x::float/sum(x) over () as part
     from generate_series(1, 3) as t(x);
 x |   frame   | sum |        part
---+-----------+-----+--------------------
 1 | {1,2,3} |   6 | 0.16666666666666667
 2 | {1,2,3} |   6 | 0.33333333333333333
 3 | {1,2,3} |   6 |                 0.5
(3 rows)
```

Window Functions

```
#  select x,
          row_number() over(),
          ntile(4) over w,
          lag(x, 1) over w,
          lead(x, 1) over w
     from generate_series(1, 15, 2) as t(x)
window w as (order by x);
```

```
 x  | row_number | ntile | lag | lead
----+------------+-------+-----+------
  1 |          1 |     1 |     |    3
  3 |          2 |     1 |   1 |    5
  5 |          3 |     2 |   3 |    7
  7 |          4 |     2 |   5 |    9
  9 |          5 |     3 |   7 |   11
 11 |          6 |     3 |   9 |   13
 13 |          7 |     4 |  11 |   15
 15 |          8 |     4 |  13 |
(8 rows)
```



Histograms

```
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) as bar
   from histogram;
```

```
 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)
```

wCTE Queries

- WITH INSERT INTO ... RETURNING *
- Trick DAOs and ORMs
- Database Access Objects can be cool
- ORM are your enemy

```
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
```

Advanced Indexing

```
#  select id, name, pos,
          round((pos <@> point(-0.12,51.516))::numeric, 3) as miles
     from pubnames
 order by pos <-> point(-0.12,51.516)
    limit 10;
     id     |           name           |            pos             | miles
------------+--------------------------+----------------------------+-------
   21593238 | All Bar One              | (-0.1192746,51.5163499) | 0.039
   26848690 | The Shakespeare's Head | (-0.1194731,51.5167871) | 0.059
  371049718 | The Newton Arms          | (-0.1209811,51.5163032) | 0.047
  438488621 | Marquis Cornwallis       | (-0.1199612,51.5146691) | 0.092
   21593236 | Ship Tavern              | (-0.1192378,51.5172525) | 0.093
  312156665 | The Prince of Wales      | (-0.121732,51.5145794)  | 0.123
  312156722 | O'Neills                 | (-0.1220195,51.5149538) | 0.113
   25508632 | Friend at Hand           | (-0.1224717,51.5148694) | 0.132
  338507304 | The Square Pig           | (-0.1191744,51.5187089) | 0.191
 1975855516 | Holborn Whippet          | (-0.1216925,51.5185189) | 0.189
(10 rows)
```

Joins, Lateral Joins

```
#   select c.name,
           array_to_string(array_agg(distinct(cp.name) order by cp.name), ', '),
           count(*)
      from cities c,
           lateral (select name
                      from pubnames p
                     where (p.pos <@> c.pos) < 5
                    ) as cp
     where c.name <> 'Westminster'
  group by c.name, replace(replace(cp.name, 'The ', ''), 'And', '&')
  order by count(*) desc
     limit 3;

   name    |              array_to_string               | count
-----------+--------------------------------------------+-------
 London    | Prince of Wales, The Prince of Wales       |    15
 London    | All Bar One                                |    12
 London    | The Beehive                                |     8
```

JOIN in DML

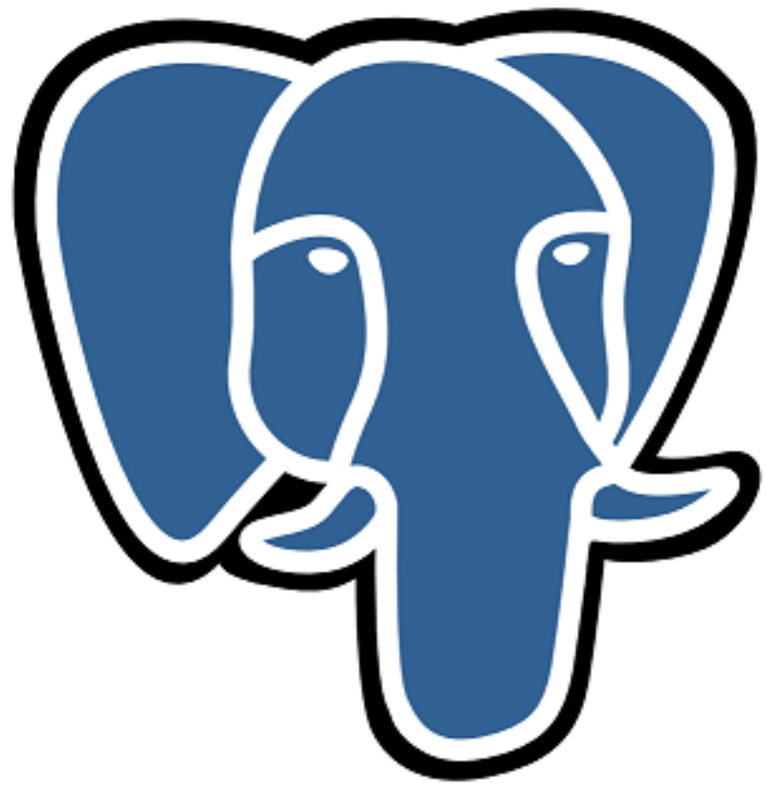
```
WITH upd AS (
 UPDATE target t
        SET counter = t.counter + s.counter,
       FROM source s
      WHERE t.id = s.id
  RETURNING s.id
)
INSERT INTO target(id, counter)
      SELECT id, sum(counter)
        FROM source s LEFT JOIN upd t USING(id)
       WHERE t.id IS NULL
    GROUP BY s.id
  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

Other SQL Features

- COPY: the Streaming Protocol
- LISTEN / NOTIFY
- JSON datatype, JSON result sets
- CREATE FUNCTION
- CREATE AGGREGATE
- Functions, operators, etc

Conclusion

- Tunable ACID
- Data Types
- Functions and Operators
- Extensions
- Advanced Indexing
- Powerful SQL
- Common Table Expressions
- Writeable CTE
- Window Functions
- Aggregates



PostgreSQL

PostgreSQL is YeSQL!