



modern and different

# PostgreSQL

a talk by Armin '@mitsuhiko' Ronacher for DUMP 2014 (Russia)

# That's me.

I do Computers.

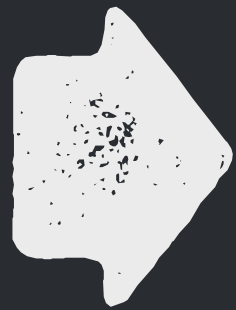
Currently at Fireteam / Splash Damage.

We do Internet for  
Pointy Shooty Games.

Aside from that: lots of Python  
stuff (Flask framework and others)







RELAX

and don't worry

**ANCIENT**

**but good & maintained**

**MODERN**

*many new features*





# Why Mongo?

Document Storage matches us well

Largely Non-Relational Data

Write Heavy

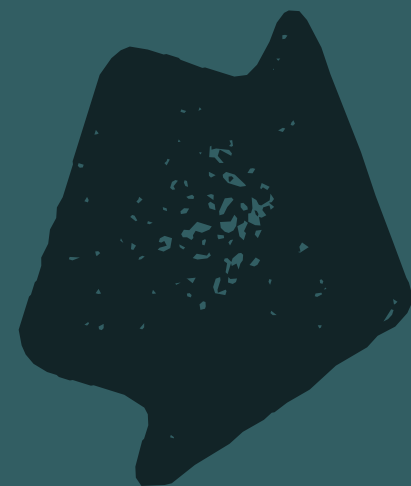
mongos (mongo router) looks interesting

# Mongo's Selling Points

Magic Auto Sharding  
Schemaless  
Automatic Scaling

but Mongo in practice ...

- ... slow
- ... huge Storage Overhead
- ... bad (no) Query Optimizer
- ... not good at using Indexes
- ... very immature



but more than anything

writing reports takes (still) way too much time



BUILD YOUR OWN MONGO

**JSON**

**BUILT IN / SLOW**

hstore  
UNTYPED & FLAT



**ARRAYS**

**get rid of some relations**

STUMBLING BLOCKS

LACK OF

OPSEERT

CAN BE EMULATED

IN THE ABSENCE OF

**hstore2**

YOU NEED TO USE

**JSON**

**SHARDING**

**NEEDS MANUAL HANDLING**

EMULATING MONGO

# emulating upsert

(UNTIL WE GET SUPPORT IN POSTGRES)

## emulating upsert

```
create function upsert_inc(the_id uuid, delta integer) returns void as $$
begin
  loop
    update my_table set value = value + delta where id = the_id;
    if found then
      return;
    end if;
    begin
      insert into my_table (id, value) values (the_id, delta);
      return;
    exception when unique_violation then
    end;
  end loop;
end;
$$ language plpgsql;
```



even better:  
DO IT WITH SAVEPOINTS

# EXCEPTION DIAG

UNDERSTAND YOUR DB EXCEPTIONS

# EXCEPTION DIAG

```
PQresultErrorField(res, PG_DIAG_CONSTRAINT_NAME)  
PQresultErrorField(res, PG_DIAG_COLUMN_NAME)  
PQresultErrorField(res, PG_DIAG_TABLE_NAME)  
PQresultErrorField(res, PG_DIAG_SCHEMA_NAME)
```

NAVCC

IS AWESOME

# TIMING AND INDEXES

# index expressions

index into JSON and other things

## index expressions

```
create index on users ((lower(username)));
```

```
create index on users ((attributes->>'location'));
```

```
create unique index on users (email) where is_active;
```

index expressions

```
set enable_seqscan to 'off';
```

**use indexes when possible for testing - not for production**



pg\_stat\_statements

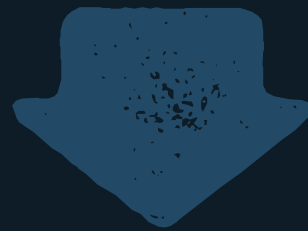
TRACK AND TIME YOUR QUERIES

pg\_stat\_statements

```
create extension pg_stat_statements;
```

pg\_stat\_statements

```
select user_id from users where email = 'foo@bar.invalid';  
select user_id from users where email = 'bar@example.com';
```



```
SELECT user_id FROM users WHERE email = ?;
```

pg\_stat\_statements

```
select (total_time / calls) as avg_time,  
       calls,  
       rows,  
       query  
   from pg_stat_statements  
 order by 1 desc  
  limit 100
```

pg\_stat\_statements

**poll periodically**  
**AND WRITE TO GRAPHITE**

AND FIGURE OUT HOW QUERIES DEGRADE

EXPLAIN ANALYZE

Now With JSON Output

# explain analyze

```
explain (analyze, format json)
select id.display_name, id._id
from instances ii, identities id
where ii.owner = id._id
limit 1;
```

```
QUERY PLAN
[
  {
    "Plan": {
      "Node Type": "Limit",
      "Startup Cost": 1.02,
      "Total Cost": 2.10,
      "Plan Rows": 1,
      "Plan Width": 48,
      "Actual Startup Time": 0.017,
      "Actual Total Time": 0.017,
      "Actual Rows": 1,
      "Actual Loops": 1,
      "Plans": [
        {
          "Node Type": "Hash Join",
          "Parent Relationship": "Outer",
          "Join Type": "Inner",
          "Startup Cost": 1.02,
          "Total Cost": 2.10,
          "Plan Rows": 1,
          "Plan Width": 48,
          "Actual Startup Time": 0.014,
          "Actual Total Time": 0.014,
          "Actual Rows": 1,
          "Actual Loops": 1,
          "Hash Cond": "(id._id = ii.owner)",
          "Plans": [
            {
              "Node Type": "Seq Scan",
              "Parent Relationship": "Outer",
              "Relation Name": "identities",
              "Alias": "id",
              "Startup Cost": 0.00,
              "Total Cost": 1.05,
              "Plan Rows": 5,
              "Plan Width": 48,
              "Actual Startup Time": 0.003,
              "Actual Total Time": 0.003,
              "Actual Rows": 5,
              "Actual Loops": 1
            }
          ]
        }
      ]
    }
  }
]
```

MANAGEMENT & OPS



**STREAMING  
REPLICATION  
AND PITR BACKUPS**

**STREAMING  
REPLICATION**

**REPMGR**

**keep a hot standby  
and fail over quickly**

STREAMING  
REPLICATION

PG\_BASEBACKUP

& WALL-E

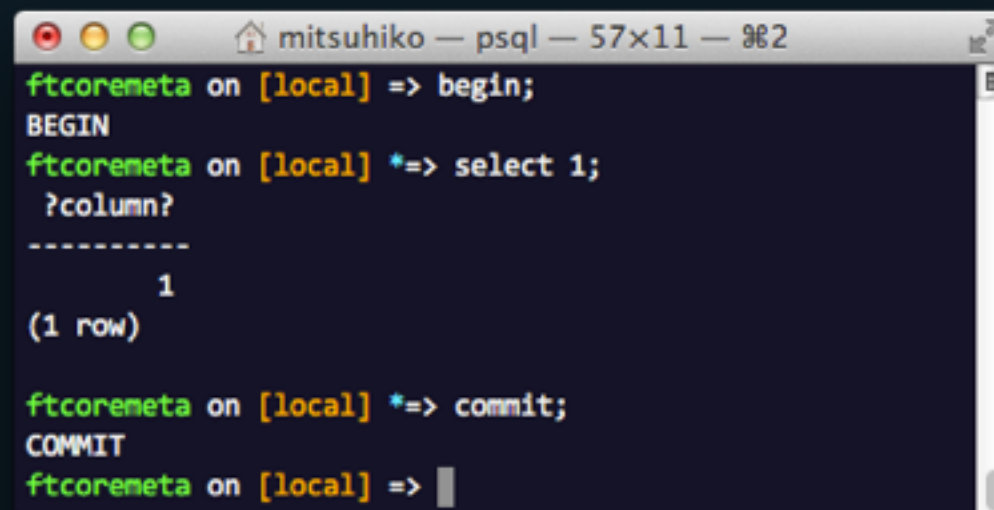
make backups

and restore quickly

pretty prompt  
for more fun when SQLing

pretty prompt

```
\set PROMPT1 '%[%033[0;33;32m%]%/[%033[0m%] on ←  
  [%033[0;33;33m%]M[%033[0m%] ←  
  [%033[0;33;36m%]x[%033[0m%]R> '  
\set PROMPT2 '%R> '
```



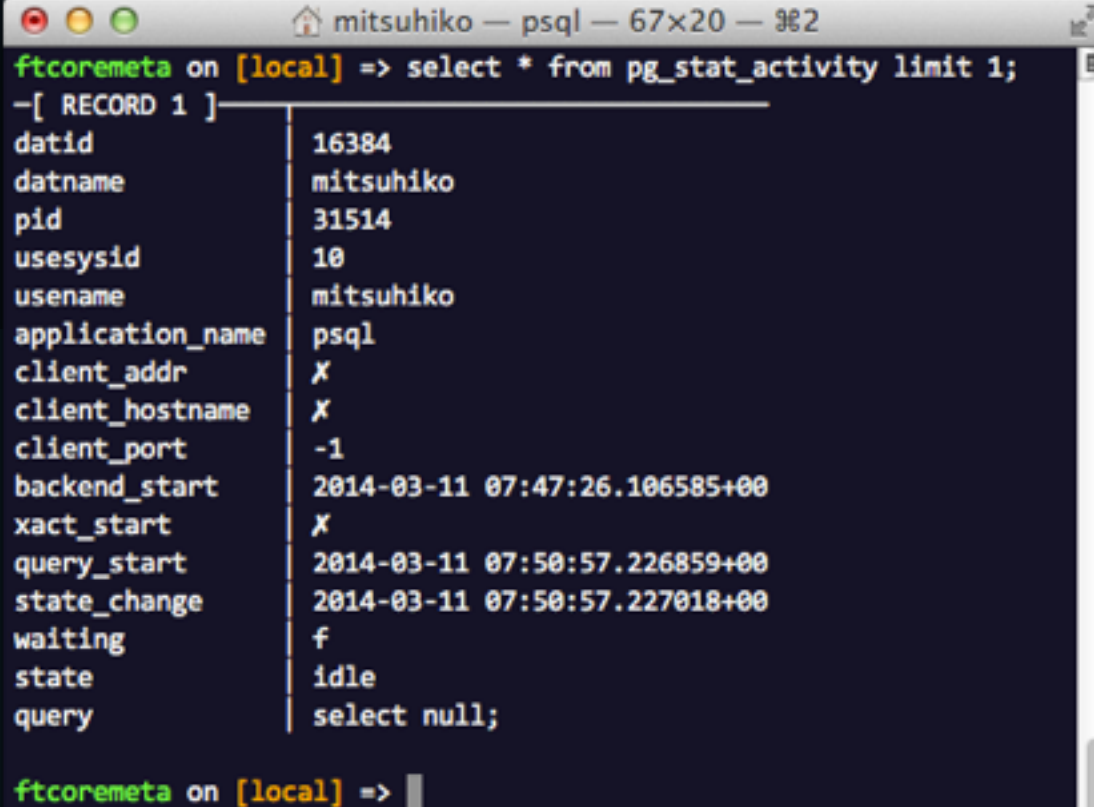
```
mitsuhiro — psql — 57x11 — 982  
ftcoremeta on [local] => begin;  
BEGIN  
ftcoremeta on [local] *=> select 1;  
?column?  
-----  
      1  
(1 row)  
  
ftcoremeta on [local] *=> commit;  
COMMIT  
ftcoremeta on [local] => |
```

# PRETTY RESULTS

Nice NULLs and Unicode

pretty results

```
\pset null 'X'  
\pset linestyle unicode  
\pset pager off  
\x auto
```



```
mitsuhiko — psql — 67x20 — 382  
ftcoremeta on [local] => select * from pg_stat_activity limit 1;  
-[ RECORD 1 ]-----  
datid          | 16384  
datname        | mitsuhiko  
pid            | 31514  
usesysid       | 10  
username       | mitsuhiko  
application_name | psql  
client_addr    | X  
client_hostname | X  
client_port    | -1  
backend_start  | 2014-03-11 07:47:26.106585+00  
xact_start     | X  
query_start    | 2014-03-11 07:50:57.226859+00  
state_change   | 2014-03-11 07:50:57.227018+00  
waiting        | f  
state          | idle  
query         | select null;  
  
ftcoremeta on [local] => |
```

# REPORTS & ANALYTICS



**REPLICATION**  
**IS YOUR FRIEND**

FEDERATED DB  
FDW

That's it.  
Now ask questions.

And add me on Twitter: [@mitsuhiko](https://twitter.com/mitsuhiko)  
Or tip me: [gittip.com/mitsuhiko](https://gittip.com/mitsuhiko)  
Slides at [lucumr.pocoo.org/talks](https://lucumr.pocoo.org/talks)

