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
but Mongo in practice...

... slow
... huge Storage Overhead
... bad (no) Query Optimizer
... not good at using Indexes
... very immature
writing reports takes (still) way too much time.
BUILD YOUR OWN MONGO
JSON
BUILT IN / SLOW
hstore
UN TYPED & FLAT
ARRAYS
get rid of some relations
STUMBLING BLOCKS
LACK OF UPSERT CAN BE EMULATED
In the absence of hstore2, you need to use JSON.
SHARDING NEEDS MANUAL HANDLING
emulating upsert
(UNTIL WE GET SUPPORT IN POSTGRES)
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
PQresultErrorField(res, PG_DIAG_CONSTRAINT_NAME)
PQresultErrorField(res, PG_DIAG_COLUMN_NAME)
PQresultErrorField(res, PG_DIAG_TABLE_NAME)
PQresultErrorField(res, PG_DIAG_SCHEMA_NAME)
MVCC is awesome
TIMING AND INDEXES
index expressions
index into JSON and other things
create index on users ((lower(username))); 
create index on users ((attributes->>'location')); 
create unique index on users (email) where is_active;
set enable_seqscan to 'off';

use indexes when possible for testing - not for production
pg_stat_statements

Track and Time Your Queries
create extension 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 = ?;
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, format json)
select id.display_name, id._id
from instances ii, identities id
where ii.owner = id._id
limit 1;
STREAMING REPLICA
TION AND PITR BACKUPS
REPMGR

keep a hot standby
and fail over quickly
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> '
PRETTY RESULT'S

Nice NULLs and Unicode
pretty results

\pset null 'X'
\pset linestyle unicode
\pset pager off
\x auto
REPORTS & ANALYTICS
REPLICATION
IS YOUR FRIEND
Federated DB
That's it.
Now ask questions.

And add me on Twitter: @mitsuhiko
Or tip me: gittip.com/mitsuhiko
Slides at lucumr.pocoo.org/talks