Postgres As Your New DevStack

Postgres as a tool for software developers

Susanne Schmidt

PGConf.eu 2018 (Lisbon)

Who am I?

Susanne “Su-Shee” Schmidt

tech lead at SysEleven GmbH (Berlin)

political scientist by education

doing open source since 1995

has cats

likes steak AND icecream!

What am I talking about?

beyond SQL: what can you do with Postgres?

not a DBA: what does Postgres offer you?

why not treat it like any other dev stack?

configure it - augment it - write code in it - make it an API - version it - CD/CI it - devstack it - dockerize it - test it

Start with Docker All The Things!

build a docker container!

```
docker build -t kittydatabase .
```

Postgres has an official docker container

quick and dirty also easy

you can also BUILD a Postgres to your liking!

Configure Postgres

make a nice prompt:

```
\set PROMPT1 ‘%n@%`hostname`[%/]: ‘
```

enable “\timing”:

```
\set timing
```

enable one history per database:

```
\set HISTFILE $HOME/.psql_history-:DBNAME
```

The "print" Equivalents

psql:

```
\echo 'foo something'
```

functions:

```
RAISE NOTICE 'something does not look right'
```

(and other common loglevels: INFO, DEBUG, ERROR..)

```
USING HINT = "check this input!"
```

(and other options: DETAIL, MESSAGE, ERRCODE...)

Start “cheap”: Make Views

hide long SELECTs behind a cosy VIEW

```
CREATE VIEW kitty_by_breed AS...
CREATE VIEW kitty_by_country AS...
CREATE VIEW kitty_by_size AS...
CREATE VIEW kitty_by_age AS …
```

present it as a unified API-like interface

Your View-API a Pseudo-Cache: Materialized Views

```
CREATE MATERIALIZED VIEW kitty_by_breed AS…
CREATE MATERIALIZED VIEW kitty_by_country AS…
CREATE MATERIALIZED VIEW kitty_by_age AS…
```

still a lovely API

now with possible indices and superfast!

Satisfy Your Inner Pedant: Use Constraints

Stronger than types - validate more strict

```
cats.origin VARCHAR(2)
```

```
cats.origin TEXT CHECK (LENGTH(origin) = 2)
```

```
CREATE TYPE origin_country AS ENUM('DE', 'FR', 'GB')
cats.origin origin_country
```

CHECK (some value with another value)

Automate ALL The Things: Triggers

automate away lots of interactions

"events" available:

`INSERT, UPDATE, DELETE, TRUNCATE`

"before" and "after" that.

Developer? Write Code, Write Functions!

functions - the heart of your code

functions - needed to be attached to a trigger

functions - to create new features

Functions to beautify

`show_table_comments()` looks somewhat better than:

```
SELECT tablename, obj_description(tablename::regclass)
       as comment
FROM   pg_catalog.pg_tables
WHERE  schemaname != ‘pg_catalog’
AND    schemaname != ‘information_schema’;
```

* yes, \dd :)

Functions to simplify

lots of people creating a JSON API-alike because:

```
data <@ '{"a":1,"b":2}'::jsonb
```

```
select * from dox.find_one(collection => 'customers',
                           term => '{"name": "Jill"}');
```

https://rob.conery.io/2018/07/05/a-pure-postgresql-document-database-a

```
SELECT json_append('{"a": 1}', '{"b": 2, "c": 3, "a": 4]
```

https://gist.github.com/matheusoliveira/9488951

Functions to fill gaps

helper functions and little niches you need

more statistics or datetime functions

```
DECLARE kitty_age text;
BEGIN
  SELECT age(birth) INTO kitty_age;
  RETURN replace(kitty_age, 'mons', 'months');
END;
```

Example: https://pgxn.org/dist/pgsql_tweaks/0.2.5/

Functions.. You Don’t Like PL/PgSQL?

you can have functions in:

Perl, Tcl, Python from base distribution

additionally from the outside:

Java, Lua, R, Shell, JavaScript (v8)

Be Part of a Processing Chain: Exporters

Export as JSON, XML, CSV, TEXT

psql via CLI

inside the database

COPY ... TO ... stdout/foo.csv

row_to_json etc

query_to_xml etc

Be Part of a Processing Chain: Import

Import CSV, TEXT

`COPY FROM ...`

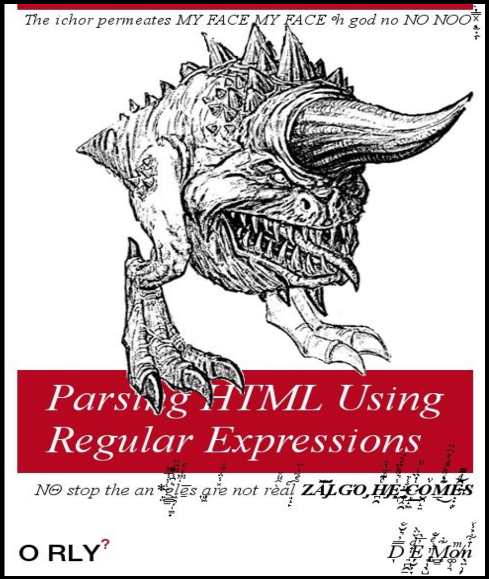
JSON (hmm)

`\set content + temporary table`

`json_populate_recordset`

~~XML~~ Chtulhu

lots of xpath to extract data into table


The ichor permeates MY FACE MY FACE ᵒh god no NO NOO
Parsing HTML Using Regular Expressions
NΘ stop the an*gles are not real ZALGO HE COMES
O RLY?
DE Mon

Be Part of a Processing Chain: psql

as long as you output simple text-based formats:

```
\o | /usr/bin/gnuplot


psql -c ... | gnuplot


$ gnuplot
set style data lines
set datafile separator "|"
....
plot "< psql -c  'select foo from blubb' "
```

<gnuplot dragon syntax here>

Processing Chain? Be a publisher!

Postgres LISTEN/NOTIFY is really simple!

```
LISTEN mychannel;
NOTIFY mychannel, 'my payload';
```

Or, `pg_notify` in function and a trigger plus clients in PG-modules supporting "LISTEN/NOTIFY".

… and LISTEN from somewhere else!

ANY library/module supporting it:

```
client.on('notification', function(msg) {
     console.log(msg);
});
client.query("LISTEN mychannel");
```

Pushed enough data out? Pull Data in!

foreign data wrappers:

for files and other databases

dblink extension:

for other postgreses

FDW FTW: Multicorn

Python module - not C!

makes it easier to write a wrapper

supports lots of wrappers out of the box

https://wiki.postgresql.org/wiki/Foreign_data_wrappers

Use other people’s code: Extensions

modules and libraries are "extensions" in Postgres:

helpful basics to just enable (postgresql-contrib packa

`pgcrypto, uuid, fuzzystrmatch...`

external extensions adding featurezzz:

`postgis`

`pgtap`

Take a look at https://pgxn.org/ !

Write Extensions Yourself!

Write extensions in PLSQL or in C!

set up "control" file (recognize as extension)

write "the extension"

add Makefile to install extension

call `CREATE EXTENSION`

Code Quality: Comments

COMMENT on everything!

```
COMMENT ON TABLE cats
IS ‘my awesome cat table’;

COMMENT ON VIEW kitty_by_breed
IS ‘my superefficient mega kitty select’;
```

Code Quality: Tests

PgTAP – unit tests for postgres

```
BEGIN;
SET search_path TO customers, public;

SELECT * FROM no_plan();

SELECT has_schema('customers');
SELECT has_table('customers');

SELECT * FROM finish();
ROLLBACK;
```

Oh, and SQL works too ;)

Postgres can also totally do this SQL thing!1! :)

Why would you bother with so much work?!

You’re not sure anymore if ORMs are a good thing

You hate cluttered/long/weird code

The concept of APIs and facades appeals to you

You really like Postgres and want to use it with EVERYTHING

Your DB IS your “single point of truth”

You want to be able to change the stack ABOVE the database more easily – but not the database

You think a database is more than just a dumping ground for data

You are suddenly faced with “err.. we need to keep this data for 10 ye

Thank you very much!

Slides: https://gitlab.com/Su-Shee

Code: https://gitlab.com/Su-Shee

New Logo Suggestions :)