POSTGRESQL FOR PYTHONISTAS
WHAT DO I DO?

• Working as a senior Python developer for Artirix.
• Building backend systems and services.
• Organiser of Python Glasgow.
AGENDA.

Combining two of my favourite things! To make the most of Python with Postgres, *and* the most of Postgres with Python.

- Very brief history.
- PL/Python.
- Listen/Notify
- Foreign Data Wrappers
HISTORY

- Ingress (1973)
- Postgres
- PostgreSQL (1997)
PL/Python
PL/PYTHON - WHY?

• Use *all* the Python libraries.
• Speak to anything Python can.
• Python is nicer to write than PL/pgSQL.
DIVING IN, A SIMPLE FUNCTION.

CREATE EXTENSION plpythonu;

CREATE OR REPLACE FUNCTION python_pow(a integer, b integer) 
returns integer AS $$
    return pow(a, b)
$$ LANGUAGE plpythonu;
SELECT python_pow(2, 10);

<table>
<thead>
<tr>
<th>python_pow</th>
</tr>
</thead>
<tbody>
<tr>
<td>1024</td>
</tr>
</tbody>
</table>

(1 row)
CREATE OR REPLACE FUNCTION python_pow(a integer, b integer)
returns integer AS $$
    plpy.notice("a to the power of b")
    return pow(a, b)
$$ LANGUAGE plpythonu;
CREATE OR REPLACE FUNCTION python_pow(a integer, b integer)
returns integer AS $$
    if a == 0 and b < 0:
        raise Exception("Zero Division")
    return pow(a, b)
$$ LANGUAGE plpythonu;
SELECT `python_pow`(0, -1);

ERROR: Exception: Zero Division

CONTEXT: Traceback (most recent call last):

PL/Python function "python_pow", line 3, in <module>

    raise Exception("Zero Division")

PL/Python function "python_pow"
CREATE OR REPLACE FUNCTION check_editable()
returns trigger AS $$
    if not TD['old']['is_editable']:
        raise Exception("Not allowed.")
$$ LANGUAGE plpythonu;

CREATE TRIGGER verify_editable BEFORE UPDATE ON events FOR EACH ROW EXECUTE PROCEDURE check_editable();
UPDATE events SET year = "2014" WHERE id = 123;

ERROR: Exception: Not Allowed

CONTEXT: Traceback (most recent call last):

   PL/Python function "check_editable", line 2, in <module>

       raise Exception("Not Allowed")

PL/Python function "check_editable"
WHEN?

• Cache invalidation or warm up with triggers.
• Avoid network round trips.
• Implement logic constraints in the database.
LIMITATIONS

- No virtualenvs and super user privileges. :(
- Debugging is a pain.
- Requires Migrations for your Python functions.
- Use in moderation
LISTEN / NOTIFY
conn = psycopg2.connect(...)  
curs = conn.cursor()  
curs.execute("LISTEN events_notify;")

while 1:
    if select.select([conn], [], [], 5) == ([], [], []):
        continue # after a timeout retry
    else:
        conn.poll()
        while conn.notifies:
            n = conn.notifies.pop()
            print notify.channel, notify.payload
LIMITATIONS

NOTIFY events_notify, 'New event added';

NOTIFY events_json, '{"msg": "New event added", "event": {
"title": "EuroPython"}}';
FOREIGN DATA WRAPPERS
FOREIGN DATA WRAPPERS

• SQL Management of External Data
• Really good support from 9.1
• Many backends: Static files, Twitter, CouchDB, Redis ...
CREATE EXTENSION redis_fdw;

CREATE SERVER redis_server
FOREIGN DATA WRAPPER redis_fdw
OPTIONS (address '127.0.0.1', port '6379');

CREATE USER MAPPING FOR PUBLIC SERVER redis_server
OPTIONS (password 'secret');
CREATE FOREIGN TABLE redis_0(key text, value text) SERVER redis_server OPTIONS (database '0');

SELECT * FROM redis_0 ORDER BY key;

<table>
<thead>
<tr>
<th>key</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>event</td>
<td>EuroPython</td>
</tr>
<tr>
<td>location</td>
<td>Florence</td>
</tr>
</tbody>
</table>

(2 rows)
CREATE FOREIGN TABLE redis_1(key text, value text[]) 
SERVER redis_server 
OPTIONS (tabletype 'hash', tablekeyprefix 'hash', database '1');

SELECT key, hstore(value) FROM redis_1 ORDER BY key;

<table>
<thead>
<tr>
<th>key</th>
<th>hstore</th>
</tr>
</thead>
<tbody>
<tr>
<td>event_1</td>
<td>&quot;name&quot;=&gt;&quot;EuroPython&quot;, &quot;where&quot;=&gt;&quot;Florence&quot;</td>
</tr>
</tbody>
</table>
(1 rows)
WRITING A FDW

So, who here likes C?

Redis - 1.5k SLOC
MongoDB - 8k SLOC
WRITING A FDW WITH PYTHON!

- Multicorn - A FDW that allows you to write other FDW’s in Python.
- Comes with a number built in and makes it trivial to write more.
from multicorn import ForeignDataWrapper
import urllib from icalendar
import Calendar, Event

class ICalFdw(ForeignDataWrapper):
    def __init__(self, options, columns):
        super(ICalFdw, self).__init__(options, columns)
        self.url = options.get('url', None)
        self.columns = columns

    def execute(self, quals, columns):
        ical_file = urllib.urlopen(self.url).read()
        cal = Calendar.from_string(ical_file)

        for v in cal.walk('vevent'):
            e = Event(v)
            line = {}
            for column_name in self.columns:
                line[column_name] = e.decoded(column_name)
            yield line

Tuesday, 9 July 13
CREATE FOREIGN TABLE gmail (  "Message-ID" character varying,  "From" character varying,  "Subject" character varying,  "payload" character varying,  "flags" character varying[],  "To" character varying)  
SERVER multicorn_imap OPTIONS (  host 'imap.gmail.com',  port '465',  payload_column 'payload',  flags_column 'flags',  ssl 'True',  login 'mylogin',  password 'mypassword')  
);

SELECT flags, "Subject", payload FROM gmail  WHERE "Subject" LIKE '%euro%'  OR "Subject" LIKE '%python%' LIMIT 2;
OR, SOMETHING MORE INTERESTING.

- Using Multicorn to write a FDW to use SQLAlchemy to access the database your running in.
OR, SOMETHING MORE INTERESTING
FOREIGN DATA WRAPPER INCEPTION

• Using Multicorn to write a FDW to use SQLAlchemy to access the database you're running in.
FDW CONCLUSION AND LIMITATIONS

- Moving data around.
- For when you really want SQL Joins back.
- Can be slow - really limited query planner performance.
BONUS SLIDE - ASYNC QUERIES

```python
conn = psycopg2.connect(..., async=1)
wait(conn)
acurs = conn.cursor()

acurs.execute("SELECT pg_sleep(10);
                SELECT * from events;")
# Do something useful here
wait(acurs.connection)
print acurs.fetchone()[0]
```
POSTGRES WEEKLY

“A free, once-weekly e-mail round-up of PostgreSQL news and articles”

http://postgresweekly.com/
Questions?

Follow me on Twitter: d0ugal

artirix.com
dougalmatthews.com
speakerdeck.com/d0ugal