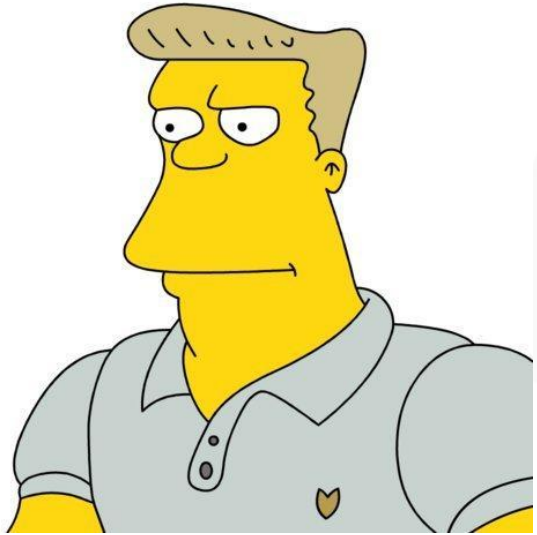
OptionFactory

# Teaching an old dog new tricks

## Extending PostgreSQL capabilities with Rust

# Francesco Degrassi



Enthusiastic yet pragmatic Lean Software Developer.

Uppish and cynical nihilist from time to time.

OptionFactory

# PostgreSQL

- Reliable
  - has been around a while
  - very robust, mature development process
- Supported
  - ubiquitous
  - mature community
- Open source
- Versatile
  - extensive support for schemaless, structured data types
  - **Extensible!!**

Old, but gold

OptionFactory

# Why you should care

- **Improve expressiveness**
  - Use constructs specific of your domain
- **Improve operability**
  - understand how your database is performing
  - expose additional operational metrics
- **Process data more efficiently**
  - leverage specific properties of your data to improve performance
- **Simplify your architecture**
  - just let Postgres do the job
- **Avoid reinventing the wheel**

# Some examples

pg_stat_statements

Track planning and
execution statistics


Spatial database extender
for PostgreSQL

Timescale

Postgres for time-series


Making Postgres and Elasticsearch
work together like it's 2023


The Postgres you love,
at any scale

# Extension points

# User defined functions

```
CREATE FUNCTION increment(x int) returns int AS $$
    SELECT x+1
$$ LANGUAGE SQL;

SELECT increment(41);
```

And also procedures, triggers

# Custom types

```
CREATE TYPE color AS (r int, g int, b int);

CREATE DOMAIN zip_code AS TEXT
       CHECK(VALUE ~ '^\d{5}$');
```

And also

- aggregates
- operators
- operator classes / families

# Procedural languages

```
CREATE TRUSTED LANGUAGE plperl
    HANDLER plperl_call_handler
    INLINE plperl_inline_handler
    VALIDATOR plperl_validator;

CREATE FUNCTION perl_max (integer, integer)
RETURNS integer AS $$
    if ($_[0] > $_[1]) { return $_[0]; }
    return $_[1];
$$ LANGUAGE plperl;
```

# Access methods

## Index

Enhance indexing capabilities by creating your own index type

```
CREATE INDEX idxproducts ON products
    USING zombodb ((products.*))
    WITH (url='http://localhost:9200/');
```

## Table

Change the storage format of tables to improve data locality

```
CREATE TABLE contestant (
    handle TEXT,
    birthdate DATE,
    rating INT,
) USING columnar;
```

# Foreign data wrappers

- Access external data as if it was a postgres table
  - both read and write access
  - parallel scan support

- Anything goes:
  - local files
  - remote files
  - remote databases (SQL or NoSQL)
  - APIs
  - /proc

```
SELECT id, name, size
    FROM ebs_volumes
    WHERE size > 1000;
```

OptionFactory

# … and many more

- background workers
- utility commands
- planner hooks
- …

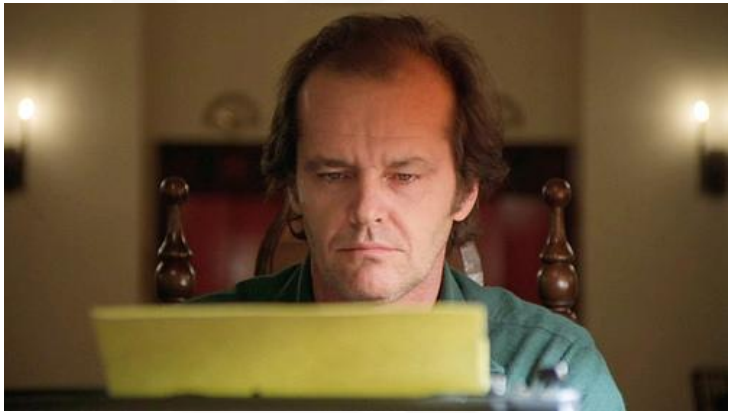
# Packing it all together

## Extensions

- Packaged set of user-defined functionality
  - can consist of a simple SQL script
- Loaded into a specific database
  - CREATE EXTENSION pg_stat_statements;
- Lifecycle support with versioning and migration scripts
  - ALTER EXTENSION pg_stat_statements UPDATE

# Postgres - authoring extensions

- A combination of languages:
  - SQL
  - embedded procedural languages
  - native code, via dynamically loaded libraries



- Native code
  - unavoidable for most advanced use cases
  - “C” calling convention
  - exported functions require metadata (generated via macros in C)
  - free access to postgresql internals and support libraries

# C extension blues

- Segmentation Fault Fest™
- Low abstraction level
- Limited composability
  - no generics or templates
- Limited expressiveness
- Lackluster tooling
  - dependency management
  - build system


OptionFactory

# Red pill #1: Multicorn

## Foreign Data Wrappers in Python

- Just foreign data wrappers
- Limited performance
- It’s Python…
- Not really active anymore

# Red pill #2: AWS TLE

## Trusted Language Extensions

- Develop extensions that can be deployed on managed databases (e.g. RDS)
- Severely limited functionality (by design)
- Provides
  - a mechanism to manage (sub)extensions without access to the filesystem
  - a single ‘passcheck’ hook (as of now)

# Red pill #3: C++

# A wild crab appears

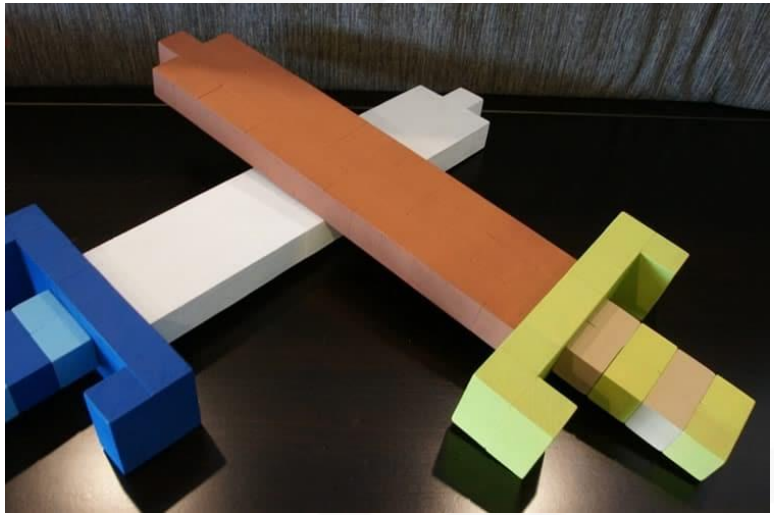
## Rust

- Safe, and yet fast
- High abstraction level
- Composable (generics)
- Expressive, functional
- Top notch tooling (cargo)

# It’s dangerous to go alone!

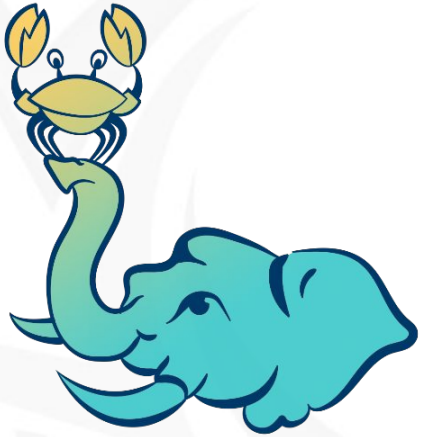
Several obstacles:

- lack of bindings for postgres functions
- lack of equivalents to postgres macros
- memory model mismatch
- unwind (error / exception handling) mismatch



# Take this!

- Rust framework for developing postgres extensions
- Developed and open-sourced by TCDI
- Active, open and friendly community
- Bridges the gap between Rust and PostgreSQL



pgrx

https://github.com/pgcentralfoundation/pgrx

OptionFactory

# On rails (but you can get off)

- Simplifies and abstracts Postgres extension points
  - rust types, rust functions, macros
  - handles incompatibility between rust unwind and postgres forced-unwind

- Still allows low-level access
  - pgrx-sys crate exposes rust bindings for low-level postgres functions

- High and low level access
  - think git porcelain & plumbing

# With you, every step of the way

- Extends build system (cargo) with pgrx-specific commands
- Downloads and compiles multiple postgres major versions
- Generates new project from archetype to start right away
- Includes idiomatic rust test framework
- Builds, deploys and runs the extension locally
- Packages the extension to ship it out

# Demo

- Installation, setup
- User Defined Functions
- Testing
- Running
- Foreign data wrapper
  - Supabase Wrappers

Fork me on github!

OptionFactory

# A sword, not a magic wand

- Unavoidable friction
  - between Rust memory model & ownership and Postgres memory contexts
- Going fast requires quite a bit of unsafe code
- Rust compilation is on the slow side
  - and pgrx is large and monolithic, but it is improving
- Pgrx still has quite a few rough edges
  - but they’re getting fixed

# Case history - project square wheel

- Distributed system, processing large amounts of data, stored in proprietary format
- Proprietary query engine
  - limited feature set
  - expensive to maintain and extend
- **Reimplement query engine as a Foreign Data Wrapper**
  - using pgrx
  - quite advanced: parallel scan, predicate pushdown, bloom filters, range indices
- Excellent results
  - Fast! Over 500M rows scanned per second
  - Easier to access
  - Far cheaper to maintain

# OptionFactory

- Lean Software Development
  - Continuous Delivery - High availability - Scale-up
  - Security sensitive & high uncertainty domains
- Software Architecture Consultancy
  - **Technical Due Diligence**

# Thanks! And get in touch!

@EdMcBane

francesco.degrassi@optionfactory.net

http://www.optionfactory.net