Postgres
Extensions Shape the Future

Ryan Lambert
RustProof Labs
Databases over time

- Postgres (2011)
- MS SQL (2008)
- Oracle (2008)
- MySQL (early 2000s)
Agenda

• What are Extensions?
• History of Postgres Extensions
• Tour of Extensions
• How Extensions are Built
This session should have been titled...

What can your database do for you?
“PostgreSQL stores much more information in its catalogs: not only information about tables and columns, but also information about data types, functions, access methods, and so on. These tables can be modified by the user, and since PostgreSQL bases its operation on these tables, this means that PostgreSQL can be extended by users.”

https://www.postgresql.org/docs/current/extend-how.html
What are Postgres Extensions?

- Extensions customize Postgres
- Created in a database
- Some are included by default
- Some need installed (OS)
History of Extensions in Postgres
2000, circa Postgres 6.5.2

- Documentation had section for "Extending PostgreSQL"

5.3) How can I contribute some nifty new types and functions for PostgreSQL?

Send your extensions to the psql-hackers mailing list, and they will eventually end up in the contrib/subdir.

5.4) How do I write a C function to return a tuple?

This requires wizardry so extreme that the authors have never tried it, though in principle it can be done.

5.4) How do I write a C function to return a tuple?

This requires wizardry so extreme that the authors have never tried it, though in principle it can be done.

2001, circa Postgres 7.1

*PostGIS enters the scene*

"Our most sophisticated developer, Dave Blasby, who had actually studied computer science, was unafraid of low-level languages"

-- Paul Ramsey

https://blog.cleverelephant.ca/2021/05/postgis-20-years.html
2008: Postgres 8.3

- Extensions continued to grow in popularity
- 34 modules in Contrib
- Installed through `psql -f extension.sql`

https://www.postgresql.org/docs/8.3/contrib.html
2008: Postgres 8.3

• Extensions included:
  • dblink
  • fuzzystrmatch
  • hstore
  • pgcrypto
2011, Postgres 9.1

New Syntax!

```
CREATE EXTENSION postgis;
```

https://www.postgresql.org/docs/current/sql-createextension.html

https://wiki.postgresql.org/wiki/Extensions
2011-2013

- I migrated all MySQL databases to Postgres
- PostGIS was the catalyst
50 extensions available in Postgres 16

Included in contrib/

- auto_explain
- pg_stat_statements
- pg_prewarm
- file_fdw

https://www.postgresql.org/docs/16/contrib.html
Not all extensions included by default

- Extension has to be installed before you can create it!
- Often available via apt/yum
- May have to download package
- ... or Install from source
- Some have dependencies

```
sudo apt install postgresql-16-postgis-3
```
Tour of Extensions

What can your database do for you today?
Tour of Extensions

• 3rd party
• Not in Contrib
• Need to be Installed
PostGIS

- Data types
- Indexes
- Nearest neighbor
- Spatial analysis

https://postgis.net/

https://blog.rustprooflabs.com/category/postgis
PostGIS is a dependency for other extensions

- pgRouting
- MobilityDB
- h3-pg

https://blog.rustprooflabs.com/2023/08/postgis-mobility-db
https://blog.rustprooflabs.com/2023/05/postgis-h3-v4-refresh
PostGIS and pgRouting
TimescaleDB

- Focus on Timeseries data
- Partitioning (with automated management)
- Compression
- Continuous Aggregates

https://www.timescale.com/
https://blog.rustprooflabs.com/2021/08/timescale-compression-openstreetmap-tags
Foreign data wrappers (FDWs)

- Remote CSV files over internet
- SQLite databases

```
CREATE EXTENSION file_fdw;
CREATE EXTENSION sqlite_fdw;
```

https://blog.rustprooflabs.com/2021/02/postgresql-sqlite-fdw-pihole
https://blog.rustprooflabs.com/2020/03/postgresql-fdw-remote-file
More FDWs
*(It doesn’t matter where your data lives)*

- ogr_fdw
- Supabase’s Wrappers

https://www.crunchydata.com/blog/remote-access-anything-from-postgres
https://supabase.github.io/wrappers/
ZomboDB

● Full text search
● Indexes backed by ElasticSearch

https://www.zombodb.com/
Citus

- Sharding
- Distributed queries
- Columnar compression


https://www.citusdata.com/blog/2016/03/24/citus-unforks-goes-open-source/
How Extensions are Built
Any extension maintainers here?
How Extensions are Built

- Raw SQL
- C
- pgrx
Extensions in Raw SQL

- Not trivial to get started
- Not expert level either
- Lot of boilerplate code
Extensions in Raw SQL

- 2nd iteration of PgDD
- Clone & make install
- Challenging to support multiple Postgres versions

Extensions in C

- Power-extensions were generally in C

However...

- Need to be good at C
- Need to understand Postgres' flavor of C
- Easy to crash Postgres
Extensions in C

"I just want to make stuff work, and I do not want to crash Postgres with my simple extension."

-- Me

https://blog.rustprooflabs.com/2021/10/pgdd-extension-using-pgx-rust
Extensions using pgrx framework
pgrx framework

Use Rust in your database

https://github.com/pgcentralfoundation/pgrx
pgrx

- Automates boilerplate
- Encourages ideation and prototyping
- Feature flags for Postgres versions

```bash
cargo pgrx new
cargo pgrx run pg16
cargo pgrx test
```
**pgrx**

- Easy to make production ready

```bash
cargo pgrx package \
  --pg-config /usr/lib/postgresql/16/bin/pg_config
```

https://tcdioss.tcdi.com/blog/install-pgx-extensions
pgrx in the Wild
PostgresML

“SQL along with the most advanced machine learning algorithms and pretrained models in a high performance database.”

https://postgresml.org/
PostgresML 2.0 built on pgrx

“The more data we're dealing with, the bigger the improvement we see in Rust.”

“... Rust is about 10x faster than native SQL, embedded PL/pgSQL, and pure Python.”

https://postgresml.org/blog/postgresml-is-moving-to-rust-for-our-2.0-release

#PASSDataCommunitySummit
pgrx in the Wild
“PL/Rust include writing natively-compiled functions to achieve the absolute best performance, access to Rust's large development ecosystem, and Rust's compile-time safety guarantees”

https://github.com/tcdi/plrust
PL/Rust

PL/Rust is available on AWS RDS!

“... with performance benefits that are comparable to writing code in C without the risk of unsafe memory access.”

Why PL/Rust on RDS matters

- Limited extensions on RDS
- PL/Rust is built on pgrx
- AWS appears to trust the stack
pgrx in the Wild
pg_subxact_counters

- Extension written in C
- And pgrx!

[Image of a bar chart showing the languages used: C 48.1%, Rust 46.9%, Makefile 3.6%, PLpgSQL 1.4%]

pgrx in the Wild
Supabase pg_graphql

- SQL schema to GraphQL
- One Function (to use)

https://supabase.github.io/pg_graphql/
“Little extensions”

“What I like to see are people using pgrx to solve a problem they have with their application. They need to talk to an S3 bucket so they put together a little pgrx extension for that.” – Eric Ridge, October 2023

https://open.spotify.com/episode/2s1wiM2S1zJVVCCJ4mduUt?si=441d4ea82922401b
My pgrx extensions

- PgDD 2021
- Convert 2022
- QR 2022 (Bad Idea)
- pgFaker 2023
PgDD moved to pgrx
PgDD moved to pgrx

- Started experimenting
- pgx pgrx 0.0.14

"the team's obvious focus on making it easy to create Postgres extensions"

https://blog.rustprooflabs.com/2021/10/pgdd-extension-using-pgx-rust
pgFaker

• Create fake text data
• Built to use in GeoFaker
  • https://github.com/rustprooflabs/geofaker/
• Easy to install

```
3 # Install pgfaker extension
4 RUN wget https://github.com/rustprooflabs/pgfaker/releases/download/v0.2.0/pgfaker.tar.gz -O /tmp/pgfaker.tar.gz \
5           && dpkg -i --force-overwrite /tmp/pgfaker.tar.gz
6```

#PASSDataCommunitySummit
pgFaker

SELECT pgfaker.company(),
    pgfaker.email(), pgfaker.person_full_name(),
    pgfaker.slogan()
FROM generate_series(1, 5);

<table>
<thead>
<tr>
<th>company</th>
<th>email</th>
<th>person_full_name</th>
<th>slogan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Colin and Sons</td>
<td><a href="mailto:abbieorn@mitchell.com">abbieorn@mitchell.com</a></td>
<td>Wyman Williamson PhD</td>
<td>Advanced analyzing paradigms</td>
</tr>
<tr>
<td>Sonny LLC</td>
<td><a href="mailto:preciousauer@champlin.org">preciousauer@champlin.org</a></td>
<td>Norberto Metz DVM</td>
<td>Compatible regional e-commerce</td>
</tr>
<tr>
<td>Cole, Skiles, and Wunsch</td>
<td><a href="mailto:rcartwright35@planda.info">rcartwright35@planda.info</a></td>
<td>Pattie Bosco</td>
<td>Self-enabling client-server content</td>
</tr>
<tr>
<td>Roberts, Upton, and Jakubowski</td>
<td><a href="mailto:ikerluke06@collier.biz">ikerluke06@collier.biz</a></td>
<td>Miss Whitney Johnson</td>
<td>Face to face clear-thinking systems</td>
</tr>
<tr>
<td>Mae LLC</td>
<td><a href="mailto:yasmineerdm@Kling.biz">yasmineerdm@Kling.biz</a></td>
<td>Vivianne Weissnat</td>
<td>Stand-alone leading edge blockchains</td>
</tr>
</tbody>
</table>
Convert extension

• Common Conversions
  • Area - `convert.area_acre_to_mi2()`
  • Distance - `convert.dist_km_to_mi()`
  • Speed - `convert.speed_m_s_to_mph()`
  • Time to Travel - `convert.ttt_meters_m_s()`
  • Power - `convert.power_dbm_to_watts()`
Does it justify an extension?
Does it justify an extension?

- PgDD and Convert could both be implemented in pure SQL code
- Other Faker options exist
- In-DB QR code generation is just a bad idea

- Why make an extension?
QR codes in Postgres

It started with a phone call...
QR codes in Postgres

*It turned into an extension!*

```sql
CREATE EXTENSION qr;
SELECT qr.generate_qr(
    'https://localhost', 'product', '1'
);
```

https://github.com/rustprooflabs/qr
Creating qr extension

- Used `pgrx`
- Took roughly an hour

Compare against 2000: “extreme wizardry”

to return a tuple!
lib.rs

This is the entirety of the custom code!

```rust
use qrcode_generator::QrCodeEcc;

#[pgExtern]
fn generate_qr(base_url: String, obj_name: String, id: String) -> String {
    let input = format!("/{}/{}/{}");
    qrcode_generator::to_svg_to_string(input,
        QrCodeEcc::Low,
        1024,
        None::<&str>).unwrap()
}
```
Does it justify an extension?

If the QR code extension turned out to be "the right solution", it was easy to make it production ready.

https://tcdioss.tcdi.com/blog/install-pgx-extensions
Does it Justify an extension?

- The $qr$ extension is a generally bad idea!
- I did it anyway
- Development cost with $pgrx$ is tiny!
Cost versus Benefit

- We're curious people
- Silly ideas are worth playing with

Some “silly ideas” turn out to be pretty great
Error handling

“If the result is \texttt{Err}, then pgrx will automatically raise a Postgres \texttt{ERROR}. A pgrx \texttt{ErrorReport} can include a specific SQL error code, detail, hint, and context message.”

https://tcdioss.tcdi.com/blog/pgx-0-7-0-spi-changes
What will your database do for you tomorrow?
New in past 2 years

*Powerful new(ish) extensions*

- PL/Rust (2023)
- PostgresML (2022)
- Supabase Wrappers (2022)
- Timescale-DB Toolkit (2021)

What will the next 2 years bring?

What will you build?

- Boilerplate automated
- Built-in safety
- Performance of Compiled
Session evaluation
Your feedback is important to us

Evaluate this session at:
www.PASSDataCommunitySummit.com/evaluation
Thank you

Ryan Lambert

blog.rustprooflabs.com

@rustprooflabs@mastodon.social