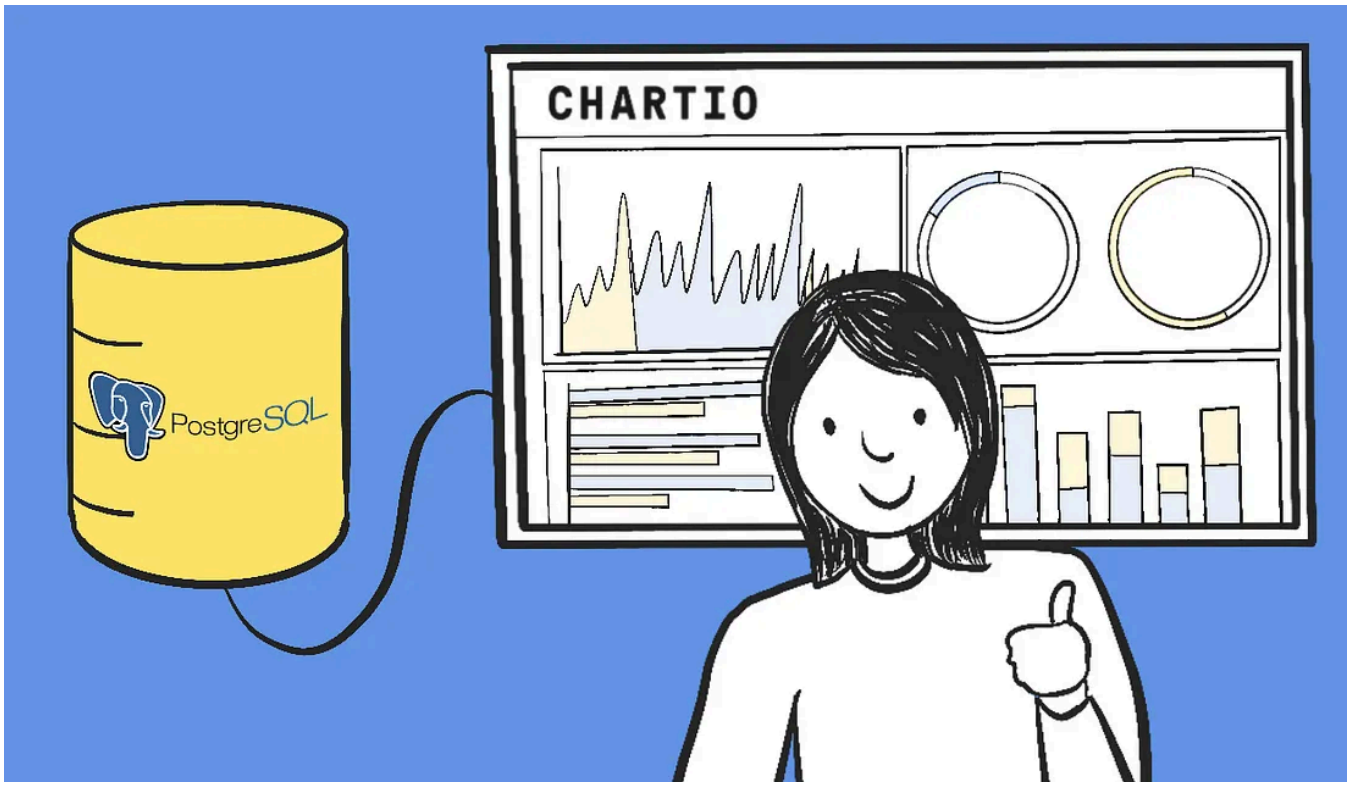
Member-only story

Historic Workload Reports For postgres

Oz · Following
3 min read · Mar 18, 2024

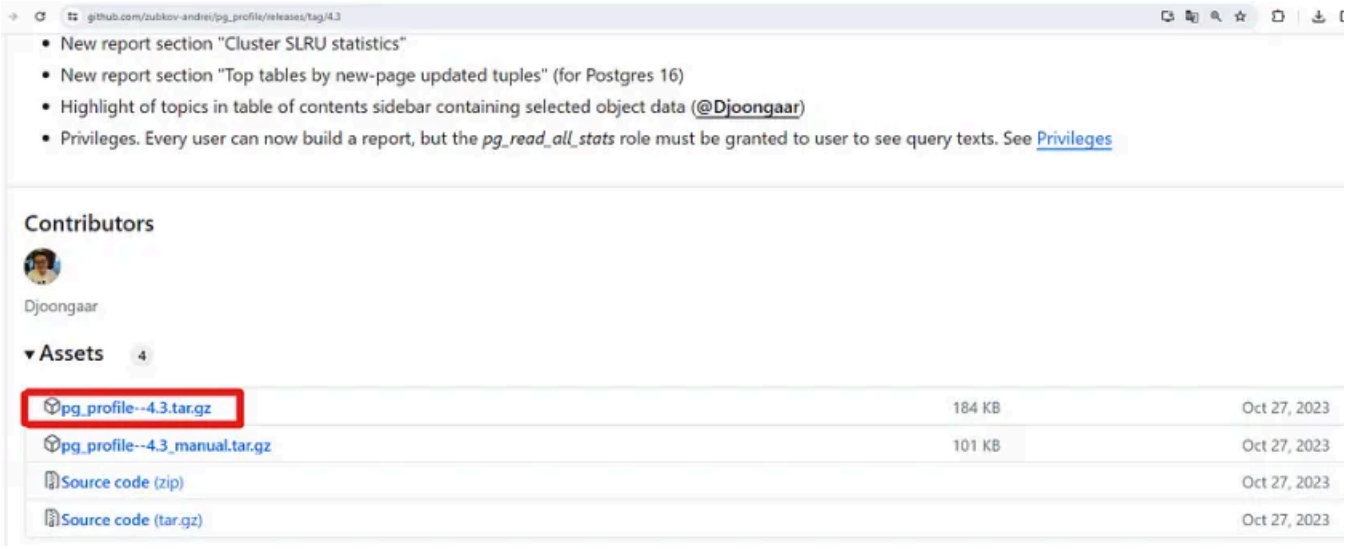
Listen Share More

Historic Workload Reports offer insights into past database activities, helping to analyze performance, identify issues, and optimize PostgreSQL databases. Using tools like pg_profile, these reports capture and present historical data on query performance, resource usage, and more, aiding in informed decision-making for database maintenance and tuning.



Download and Installation:

- The first step is to download the pg_profile extension from its GitHub releases page as shown below figure. This extension provides tools for monitoring and analyzing workload in PostgreSQL databases.



https://github.com/zubkov-andrei/pg_profile/releases/download/4.3/pg_profile--4.3.tar.gz

- You can use the `wget` command to download the `.tar.gz` file from the specified release.

```
wget https://github.com/zubkov-andrei/pg_profile/releases/download/4.3/pg_profi
```

- Once downloaded, you need to extract the contents of the archive to the extension directory of your PostgreSQL installation. The `tar` command is used for this purpose.

```
tar -xzf pg_profile - 4.3.tar.gz -C /usr/pgsql-14/share/extension/
```

```
ls -ltr /usr/pgsql-14/share/extension/ | grep pg_profile
```

Configuration:

- Next, you need to configure PostgreSQL to load the `pg_stat_statements` module and enable various tracking parameters. These settings are essential for collecting workload information.
- Edit the `postgresql.conf` file (or Patroni configuration if you're using Patroni for managing PostgreSQL) and add/modify the following settings:

```
vi /var/lib/pgsql/14/data/postgresql.conf

# You can use patroni edit configuration if you use
patronictl -c /etc/patroni/patroni.yml edit-config
```

```
# for Postgresql.conf

shared_preload_libraries = 'pg_stat_statements'
track_activities = on
track_counts = on
track_io_timing = on
track_functions = all
# Snapshot age parameters
# Adjust the parameters related to snapshot age and top-N queries according to
pg_profile.max_sample_age = 7
pg_profile.topn = 20


# for patroni edit-config
shared_preload_libraries: pg_stat_statements
 track_activities: true
 track_counts: true
 track_functions: all
 track_io_timing: true
 pg_profile.max_sample_age: 30
 pg_profile.topn: 10
```

Extension and Schema Setup:

- Create the required extensions and schema for pg_profile:

```
CREATE EXTENSION dblink;
CREATE EXTENSION pg_stat_statements;
CREATE SCHEMA profile;
CREATE EXTENSION pg_profile SCHEMA profile;
```

Usage:

- You can now interact with the pg_profile extension to gather workload information and generate reports.
- Commands like `profile.show_servers()`, `profile.take_sample()`, `profile.snapshot()`, and `profile.show_samples()` are available to manage and view collected data.

```
select * from profile.show_servers();
select * from profile.take_sample();
select * from profile.snapshot();
select * from profile.show_samples();
```

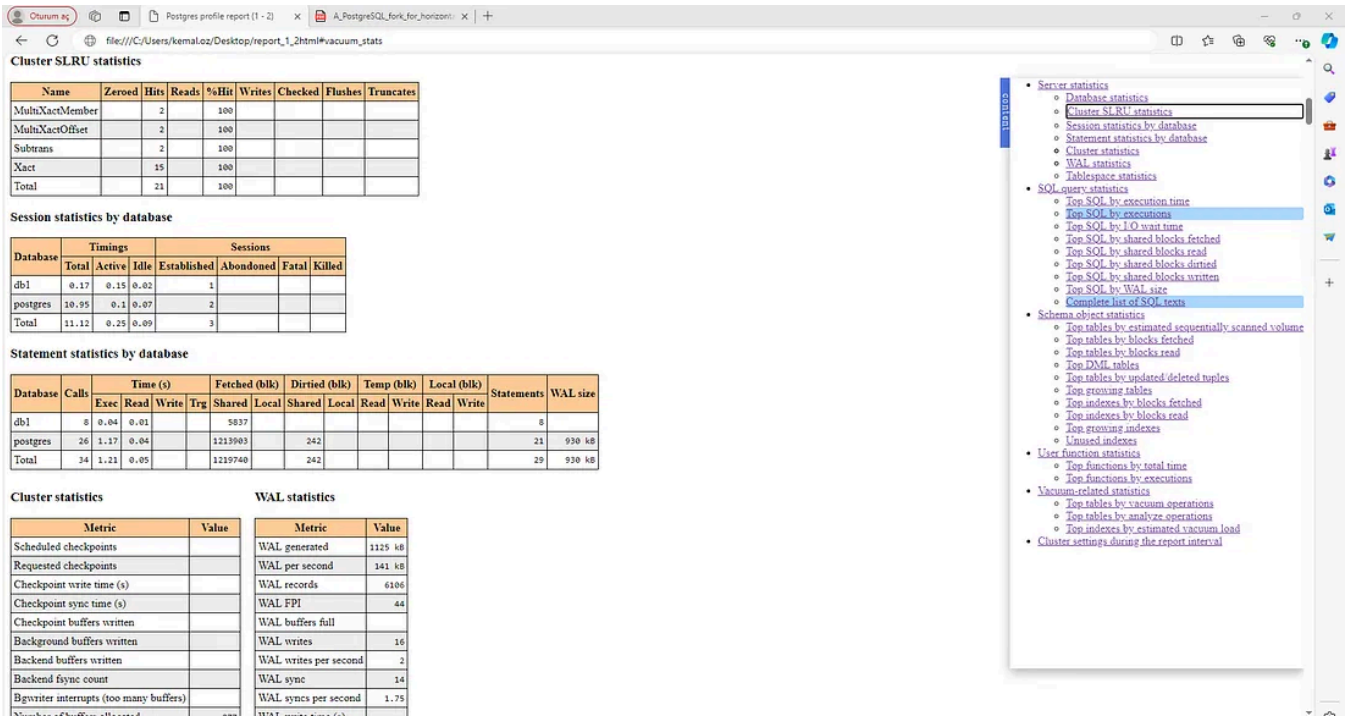
```
db=# select * from profile.show_servers() ;
 server_name |       connstr       | enabled | max_sample_age | description
-------------+---------------------+---------+----------------+-------------
 local       | dbname=db port=5432 | t       |                |
(1 row)

db=# select * from profile.take_sample();

 server | result |   elapsed
--------+--------+-------------
 local  | OK     | 00:00:01.26
(1 row)

db=#
db=# select * from profile.snapshot();
 server | result |   elapsed
--------+--------+-------------
 local  | OK     | 00:00:01.22
(1 row)

db=# select * from profile.show_samples();
 sample |      sample_time       | sizes_collected | dbstats_reset | clustats_reset | archstats_reset
--------+------------------------+-----------------+---------------+----------------+-----------------
      1 | 2024-03-18 09:39:58+03 | t               |               |                |
      2 | 2024-03-18 09:40:06+03 | t               |               |                |
(2 rows)
```

- Finally, you can generate a report using `psql` with the `profile.get_report()` function, specifying the desired snapshot IDs. For example:

```
psql -Aqtc "SELECT profile.get_report(1,2)" -o report_1_2.html
```

- Also, you can take a profile sample each 30 minutes

```
# Scheduling Snapshot
*/30 * * * * psql -U postgres -d postgres -c 'SELECT profile.take_sample()' > /
```



Example HTML report that based on the snapshots with IDs 1 and 2

For more detailed and technical articles like this, keep following our blog on Medium. If you have any questions or need further assistance, feel free to reach out in the comments below and directly.

Following

Written by Oz

149 Followers · 13 Following

Database Administrator 🐘

No responses yet

Gvadakte

What are your thoughts?

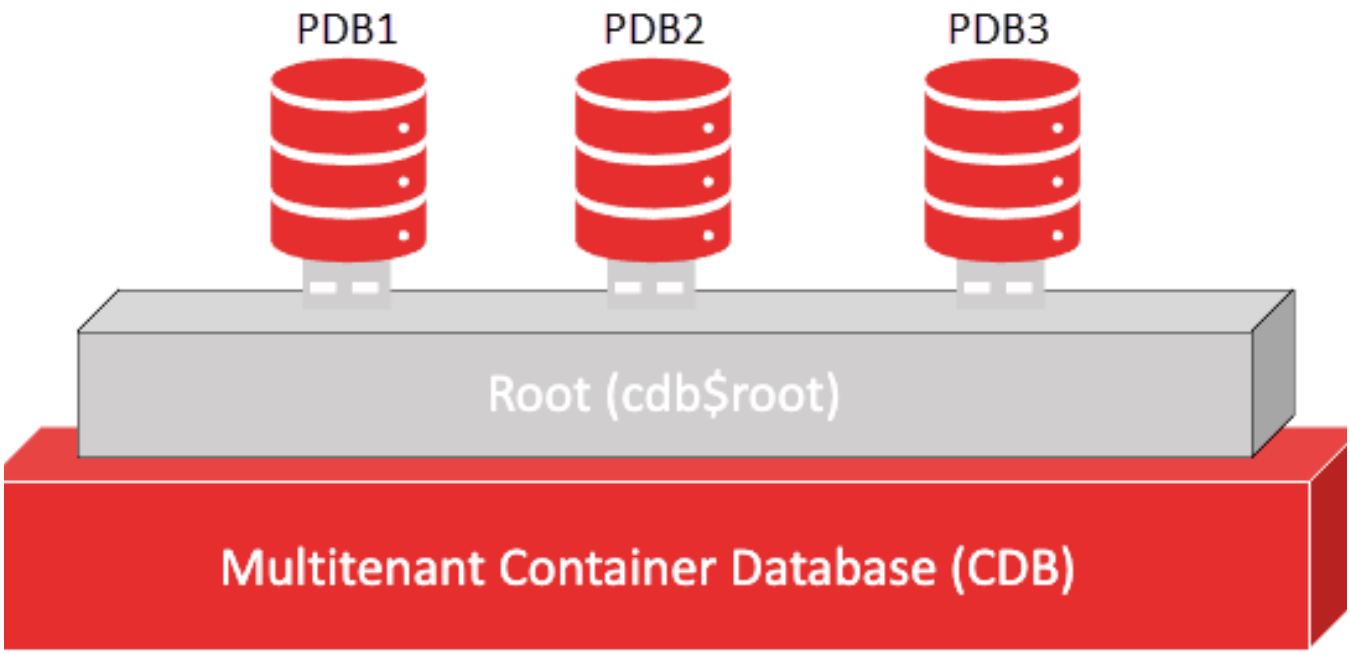
More from Oz

Oz

Installing Percona Monitoring & Management (PMM) with Postgres

Introduction:

Sep 26, 2024 54 1

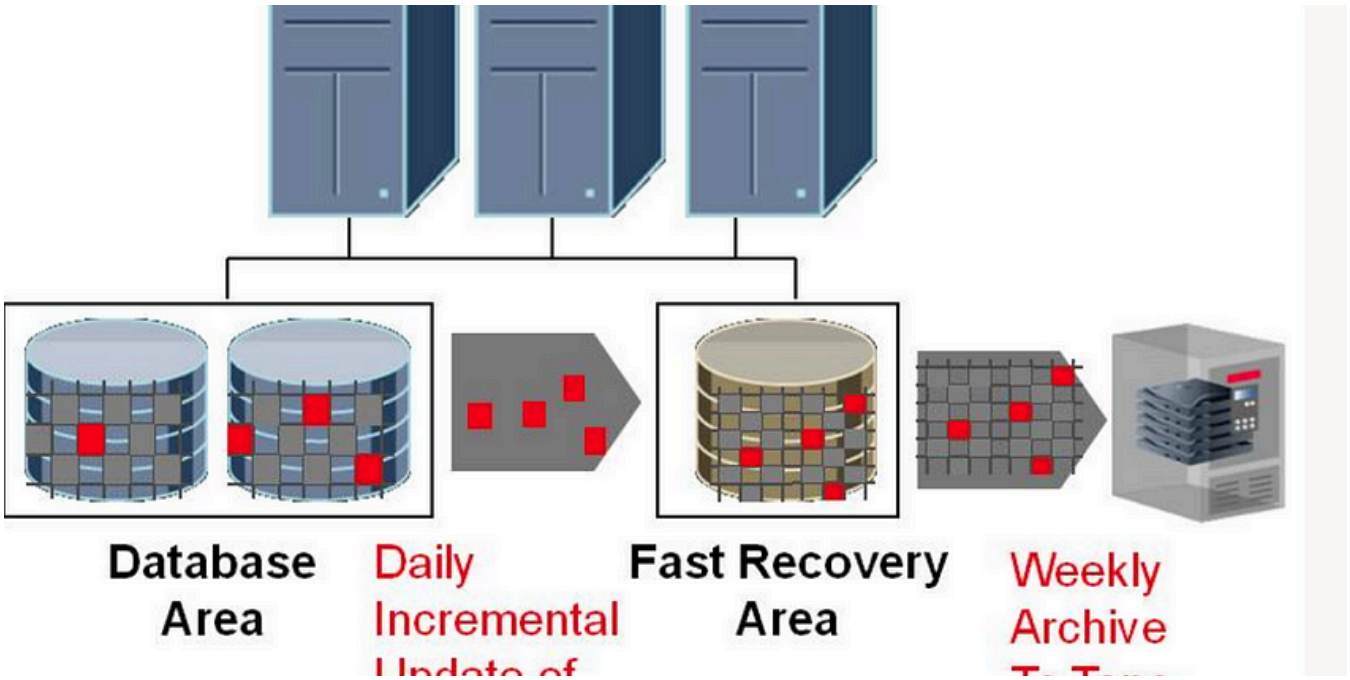


Oz

Pluggable Database Command

———————————— -———————————————— - create pluggable database pdb1 admin user root identified by test123; alter pluggable database...

May 12, 2023



Oz

RMAN Backup Basic Commands

rman target / rman target sys/password@YDKTST; backup database; backup database format ‘/backup/path/%d_%t_%s.rman’; backup tablespace...

May 11, 2023 1

Oz

delete jobs

May 8, 2023

See all from Oz

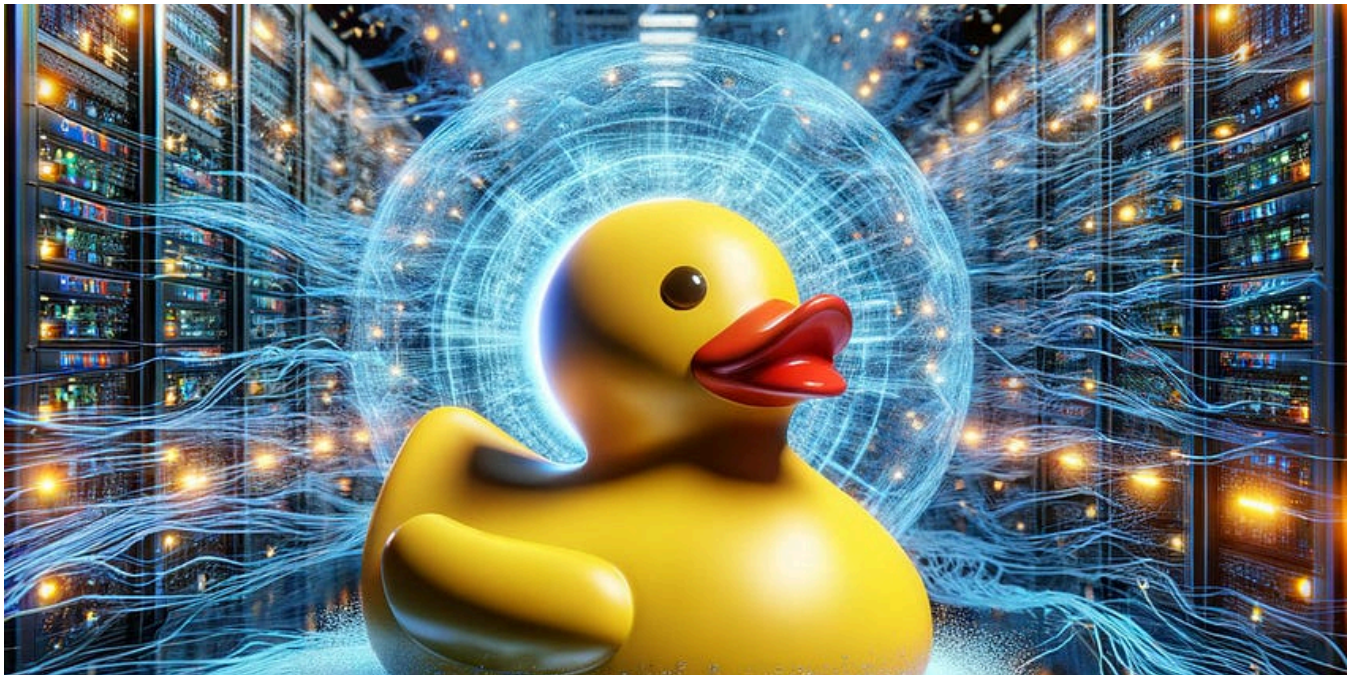
Recommended from Medium

mehmetozanguven

Running PostgreSQL with Podman

Instead of running PostgresSQL locally, we can easily run with Podman. Here are the basic steps you should follow.

Mar 28 2

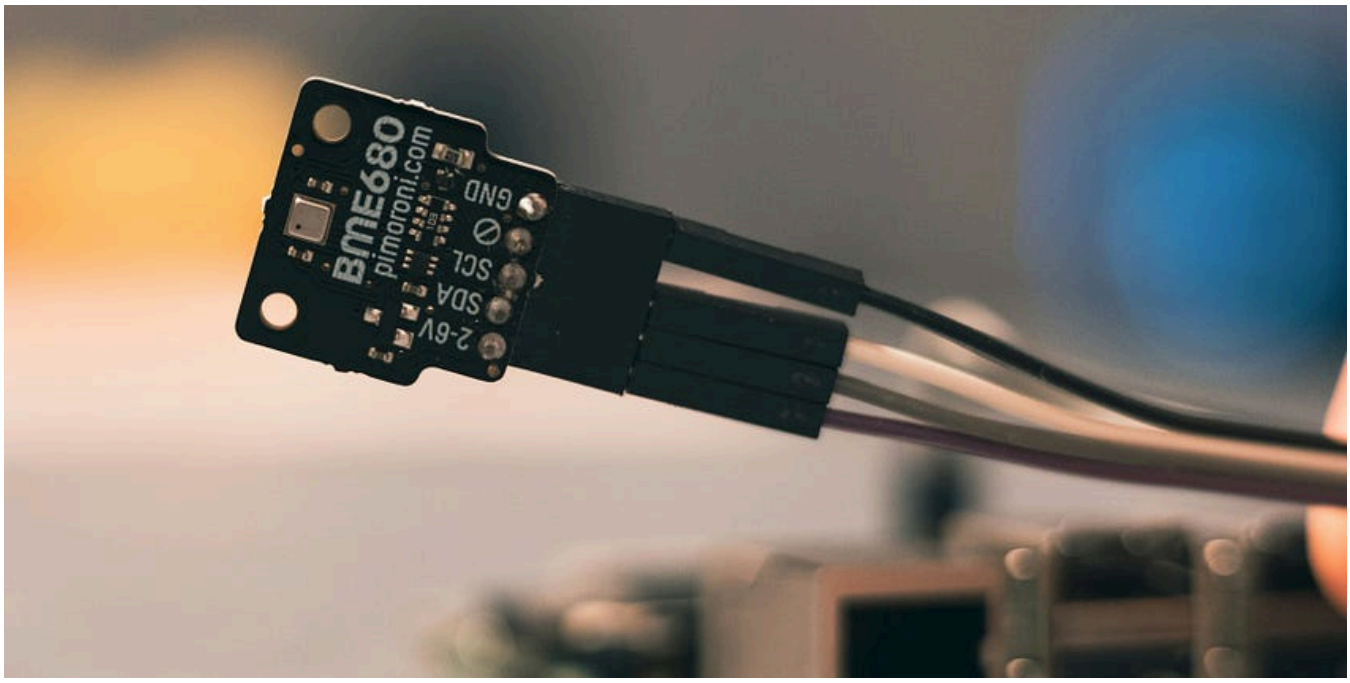


Josef Machytka

Quick and Easy Data Exports to Parquet Format Using DuckDB

The Parquet format has become almost an industry standard for Data Lakes and Data Lakehouses, thanks to its efficiency and compact storage...

Dec 5, 2024 17 1



Tihomir Manushev

Time-Series Data with TimescaleDB and PostgreSQL

Supercharge PostgreSQL for time-series workloads

Mar 10 7

In Towards Dev by Nakul Mitra

PostgreSQL Performance Optimization—Cleaning Dead Tuples & Reindexing

Performance optimization is crucial in PostgreSQL to ensure efficient query execution and minimal resource consumption.

Mar 28 1

In Timescale by Team Timescale

State of PostgreSQL 2024: PostgreSQL and AI

A sneak peek into how PostgreSQL developers are using AI, courtesy of the 2024 State of PostgreSQL survey.

Dec 26, 2024

In Databases by Sergey Egorenkov

Making SQL query 40x faster for 10 million rows table

Make your SQL query really fast using this approach

Mar 17 8