Engineer

PostgreSQL

28 achievements

Connect the parts of your system

The work collected under “PostgreSQL” shows one part of a working system. We can help connect that part to the rest of your operation. Finding the queries and the missing indexes behind a system that has got heavier every year, and fixing the ones that matter.

Explore our services Get in touch →

Find a comparable challenge

Follow an example that resembles your situation. Its review describes the decisions behind the result, beyond the technology label. 30 days of electricity data in 180 ms — 180 ms to aggregate a month of meter readings

30 days of electricity data in 180 ms Get in touch →

30 days of electricity data in 180 ms

Engineered hourly electricity-consumption aggregation variants in Python, SQL and Bash + Jq, with the Python benchmark processing one 30-day JSONL dataset in 180ms.

Made a geographical data pipeline 50x faster

Recorded a 50x geographical data-pipeline improvement in the founder’s Utiligize work through SQL and data-model changes across PostgreSQL, MS SQL and Google Cloud BigQuery.

Improved GIS map performance 10x on PostgreSQL

Improved geographical map application performance by 10x through strategic database transition from MSSQL to PostgreSQL, including the data cutover, optimizing processing and data security.

Resolved 1,000 issues in geographical data

Resolved 1,000 issues in geographical data and time-series data, using GDAL, ArcGIS, PostGIS, Mapbox, QGIS, SQL (PL/pgSQL, Transact-SQL), Bash, ensuring high-quality big data processing.

Designed and ran 6 ETL/ELT pipelines

Designed, implemented, and administered 6 ETL/ELT pipelines, utilizing Google BigQuery, MSSQL, PostgreSQL, Shell scripting, PL/pgSQL, and Transact-SQL, integrating data for efficient Python API processing.

Released 500 electricity and GIS data reports

Released 500 electricity and GIS data analysis reports, utilizing deep research and troubleshooting to ensure accurate geographic and time series big data insights.

Ran 100 PostgreSQL, MS SQL, BigQuery warehouses

Architected, created, and managed 100 PostgreSQL, MS SQL, and Google BigQuery data warehouse databases with primarily GIS and time-series data, optimizing performance and scalability.

Ran 10 PostgreSQL and MS SQL servers on Ubuntu

Designed, deployed, and maintained 10 PostgreSQL and MS SQL servers on Ubuntu Linux VPS, ensuring optimal server performance and reliability.

Accelerated PostgreSQL performance by 10x

Accelerated PostgreSQL performance by 10x via strategic indexing, partitioning, and query optimization, enhancing database efficiency for user, tenant, geospatial, and time-series electrical data.

Automated 100 critical data backups using Barman

Automated 100 critical data backups using Barman, Google Cloud, Bash, and Python, ensuring data integrity across databases.

Directed full-stack GIS map development

Directed full-stack GIS map development, overseeing PostgreSQL, Mapbox, ReactJS, and NodeJS to deliver an integrated solution.

Led 30+ GIS projects from build to support

Led the development, deployment, and support of over 30 GIS projects, demonstrating expertise in PostgreSQL, Bash, Python, JavaScript, GDAL, ArcGIS, PostGIS, and Mapbox technologies.

Automated data processing in Python and PL/pgSQL

Automated data processing tasks using Shell scripting, PL/pgSQL, Python, and Transact-SQL, increasing productivity and efficiency.

Engineered 600 PL/pgSQL ETL/ELT pipelines

Engineered 600 PL/pgSQL-based ETL/ELT pipelines to streamline complex data processing workflows across multiple PostgreSQL development and production environments.

A maritime platform with clear system boundaries

Architected a layered maritime platform separating a Next.js PWA frontend, a Go (Huma/Fiber) API, and a PostgreSQL function layer, keeping all business logic in the database.

Designed a JSON passthrough from Postgres to API

Designed a JSON passthrough architecture where PostgreSQL functions return complete JSON forwarded verbatim by the Go API, eliminating intermediate unmarshalling and decoupling the frontend from schema changes.

Designed a function-first PostgreSQL data layer

Designed a PostgreSQL function-first data layer — 1,275 stored functions across 34 schemas — so every read and write goes through a function the database can grant, rather than through a table.

Time-ordered keys for a growing data model

Adopted UUID v7 time-ordered identifiers (PostgreSQL 18) as entity keys to reduce B-tree index fragmentation and speed up queries.

Implemented a catalogue-driven deep-merge

Implemented a catalogue-driven deep-merge for stored JSON preferences, preventing missing-key crashes as the schema evolves.

Modeled 348 tables across 34 Postgres schemas

Modeled the maritime domain into 348 normalized tables across 34 PostgreSQL schemas — professionals, companies, ships, jobs, reviews and the rest — with SMALLINT lookups and UUID v7 keys.

Implemented backups and disaster recovery as code

Configured database backup retention as infrastructure-as-code, then audited the recovery position and documented the restore procedure — naming the remaining gaps rather than leaving them to be found during an incident.

Built a test suite across Go, SQL and browser

Built a layered automated test suite — 981 Go tests, 543 frontend and browser specs, 494 SQL behavioural tests — with mutation testing, property-based tests and an accessibility gate.

Built Stripe and in-app purchase entitlements

Built the payments and entitlements layer — Stripe alongside Apple and Google in-app purchase — gating the directory, search and export through an 11-table access model checked on the server.

Moved slow work onto a River job queue

Moved slow work off the request path onto a River job queue — 15 worker modules, 8 scheduled tasks and 20 pg_cron jobs — so a request returns while the work behind it carries on.

1,022 migrations behind a schema drift gate

Kept the schema honest across 1,022 migrations with a CI gate that builds the database both ways — a fresh install, and an install plus every migration — and fails when the two disagree.

Built first-party analytics inside PostgreSQL

Built first-party product analytics in PostgreSQL — 47 functions over partitioned event tables that prune themselves — pseudonymised behind a rotating salt and gated on the visitor's consent.

Built a loyalty wallet, leagues and badges

Built the loyalty and reputation system — 67 functions over a 31-table ledger, with leagues, badges and a redemption shop — taking a row lock on the balance to close the double-spend window.

Built the hiring and seafarer career workspaces

Built hiring and seafarer career workspaces; the recorded implementation snapshot contains 151 stored functions across 48 tables for vacancies, applications, certificates, rank progression and sea time.

Focus the engineering effort

Bring a representative dataset, the questions it must answer and the current bottleneck. We can agree how to measure quality, throughput and recovery.

See what it costs Get in touch →

Put the subject in your context

Tell us how “PostgreSQL” appears in your own system. We can help choose the relevant service and define the work around it.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/categories/postgresql/