Engineer Software

Databases

24 achievements

Connect the parts of your system

The work collected under “Databases” shows one part of a working system. We can help connect that part to the rest of your operation. Anything that happens in a place — vessels, vehicles, plots, sensors — put on a map you can search, measure and share.

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. Made a geographical data pipeline 50x faster — 50x faster geographical data pipelines

Made a geographical data pipeline 50x faster Get in touch →

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.

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.

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.

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.

Hardened the application with nonce-based CSP

Hardened the application with nonce-based CSP, HSTS, SameSite cookies, least-privilege database roles and server-side entitlement re-checks.

Set the platform's founding technical decisions

Set the platform's founding decisions in the weeks after the repository opened in November 2025 — the layering, database-first data access and the zero-warnings bar — and they still hold nine months on.

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.

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 social layer: posts, feed and groups

Built the platform's social layer — posts, feed, groups, mentions, a follower graph and an occasions digest — on the same function-first data layer as the rest of the product.

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.

Generated 495 pages from a SQLite export

Generated 495 achievement pages across three languages from a read-only SQLite export, with the page address authored as data so that correcting a sentence no longer moved the page and broke the link.

Built a database-driven CV generator

Built a database-driven CV, references, portfolio and cover-letter generator in Python — 41 modules, 10,580 lines — rendering six output formats from one 23-table SQLite source assembled by a 19-step idempotent pipeline.

Held the generator to 981 tests and 92% coverage

Held the generator to 981 test cases at a 92% branch-coverage floor with warnings treated as failures, and asserted idempotence by running the whole build pipeline twice from an empty file and requiring the second pass to change nothing.

Wrote tests for the checkers themselves

Wrote tests for the checkers themselves after establishing that a checker fed only clean input will one day report clean because it read nothing — planting a misspelling to confirm the spell-check finds it, and taking an id range from the database rather than from a number in the test.

Focus the engineering effort

Describe the workflow, the existing interfaces and the users who need the change. We can define a testable delivery and a handover that lets your team maintain it.

See what it costs Get in touch →

Put the subject in your context

Tell us how “Databases” 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://software.engineer.company/categories/databases/