Engineer Data

Implemented a catalogue-driven deep-merge

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

2025

stored preferences that survive a schema change

Situation. The platform stores JSON preferences — notification settings and the like — as a user's saved values laid over a set of defaults. The original merge did that at a single level. The problem shows up later: add a new key to the defaults, and rows saved before that key existed simply don't have it. Then some client code reads that field, gets undefined, and falls over — for exactly the users who've been around longest.

Task. Evolving the preference schema had to be safe, so that adding a setting could never break the people who signed up before it existed.

Action. The single-level merge was replaced with a deep-merge driven by the defaults as a catalogue. The defaults are treated as the authoritative list of every key that should exist; the user's stored values are merged recursively on top, so anything in the catalogue is guaranteed to come out present, whether or not it was in the saved blob. Add a key to the defaults and it appears in every existing row's effective preferences automatically, nested keys included. It rolled in through a migration so existing data got the benefit straight away rather than waiting to be rewritten.

Result. Preferences can grow without fear. Adding a setting no longer risks an undefined-field crash in the client, and the frontend stopped needing defensive checks scattered around every place it reads a preference. It also gave the rest of the platform a dependable way to extend any stored-JSON blob — the pattern, not just the one fix.

Bring this experience to your project

Use “Implemented a catalogue-driven deep-merge” as a starting point for a similar challenge in your organisation. The services behind this work are available for your project. A structure that matches how the business actually works, so the questions you need answered later are still possible to ask.

Explore our services Get in touch →

Look behind the result

Read the situation, decisions and result in this example. We can discuss which parts transfer to your constraints and which need another approach. Implemented a catalogue-driven deep-merge — stored preferences that survive a schema change

Explore our services Get in touch →

Need a schema that evolves without breaking?

Talk through your project with the engineer responsible for delivery. We agree the scope, price basis and handover in writing before work begins.

Role context: Database Engineer. It draws on Backend Engineering, Databases, PostgreSQL, Data Governance and Migrations & Modernization. And while you are here, we are rather good at Backend & API Development, Database Design & Modeling and Data Governance & Quality.

Get in touch Explore our services

One piece of **a longer record** — and it is all on this site.

Every entry is written the same way: the situation, the task, what we did and what changed.

See the whole record

Define the improvement you need

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 →

Need a schema that evolves without breaking?

Ask about the decisions behind “Implemented a catalogue-driven deep-merge”. You will speak with the people who can explain the work and assess your own project.

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://data.engineer.company/portfolio/implemented-a-catalogue-driven-deep-merge/