

The database wasn't slow. It was being asked too much.

An internal back office for an energy retailer, built on Laravel and Filament over Oracle 11g. **Every page of the admin panel ran 928 database queries and took 22 seconds to load.** The cause wasn't the server or the database: it was a handful of patterns in the application, repeated hundreds of times per click. Found by counting queries before guessing, and fixed without a rewrite.

QUERIES PER PAGE	PAGE LOAD	MENU QUERIES	HOTSPOTS
928 → 36 same page, before and after	22.3 s → 2.7 s measured in development	200+ → 1 one cached call	3 menu, record pages, bulk import

THREE FINDINGS

01 A menu that queried the database once per screen

Every admin resource looked up its own navigation override separately: over two hundred queries to draw the menu, on every page. Now one query, cached in Redis and invalidated on change. The same page went from 928 queries to 36, and from 22.3 seconds to 2.7.

02 Lazy loading on the record pages

A status page running 930 queries, loading related records one access at a time. Fixed with eager loading, single-query relations and cached workflow state.

03 A bulk import doing one round trip per row

Per-row existence checks, inserts and lookups replaced with a set-based anti-join, 500-row batch inserts and reference data loaded once. Laravel's OFFSET pagination, which Oracle 11g wraps in ROWNUM, was re-scanning the whole source view on every chunk; a cursor now streams it once.

No new server, no rewrite, no framework change. The ORM makes each query easy to write; it doesn't show what they add up to.

STACK

PHP 8 · Laravel · Filament · Livewire · Oracle 11g · Redis

Richard Trujillo · Desemax. Senior backend and AI-integration engineer. Eleven years of backend engineering, independent since 2015. Based in Milan, working across Europe.

richard@desemax.it · richardtrujillo.desemax.it

Full case study, with each finding and fix: richardtrujillo.desemax.it/case-studies