

SQL databases in your browser

Erik Söhnelt (github.com/hoeck)

twenty years ago



what happened then



today



baby steps

September '22: postgres-wasm

github.com/snaplet made postgres-wasm:



Postgres running in the



browser *on top* of a



linux emulator using a



terminal emulator to interact with it

<https://github.com/snaplet/postgres-wasm>



just another dependency

Feb '24 - pglite

<https://github.com/electric-sql/pglite>

Just PostgreSQL compiled to Web Assembly!

```
$ npm install @electric-sql/pglite
```

hello world

```
import { PGLite } from "@electric-sql/pglite";

const db = new PGLite(); // a single connection
const result = await db.query("select 'Hello world' as msg;");
// {
//   "rows": [
//     {
//       "message": "Hello world"
//     }
//   ],
//   "fields": [
//     {
//       "name": "message",
//       "dataTypeID": 25
//     }
//   ],
//   "affectedRows": 0
// }
```

what cost? size!

```
$ ls -lh react/dist.tar.gz  
-rw-r--r-- 1 erik erik 62K Feb  5 22:01 react/dist.tar.gz
```

VS

```
$ ls -lh react-pglite/dist.tar.gz  
-rw-r--r-- 1 erik erik 4.3M Feb  5 22:01 react-pglite/dist.tar.gz
```


what cost? startup time!

```
const start = Date.now();  
const db = new PGLite();  
const res = await db.query("select 'Hello world' as message");  
  
console.log(Date.now() - start);  
// ~2.5s (on my old thinkpad)
```

case study

cheap orm sql playground

<https://hoeck.github.io/typesafe-query-builder-playground/>

Typesafe Query Builder Playground

RUN

Github

examples

```
1 import { query, table, column as col } from "typesafe-query-builder";
2 import { db, show } from "playground";
3
4 export const Manufacturers = table("classicgames.manufacturers", {
5   id: col("id").integer().primary().default(),
6   name: col("name").string(),
7   country: col("country").string(),
8 });
9
10 export const Systems = table("classicgames.systems", {
11   id: col("id").integer().primary().default(),
12   name: col("name").string(),
13   year: col("year").integer(),
14   manufacturerId: col("manufacturer_id").integer(),
15 });
16
17 const res = await query(Manufacturers).select(Manufacturers.all()).fetch(db);
18
19 show(res);
20
```

Result SQL Log

```
{
  0: {
    "id": 1
    "name": "Sega"
    "country": "Japan"
  }
  1: {
    "id": 2
    "name": "Nintendo"
    "country": "Japan"
  }
  2: {
    "id": 3
    "name": "Atari"
    "country": "USA"
  }
}
```

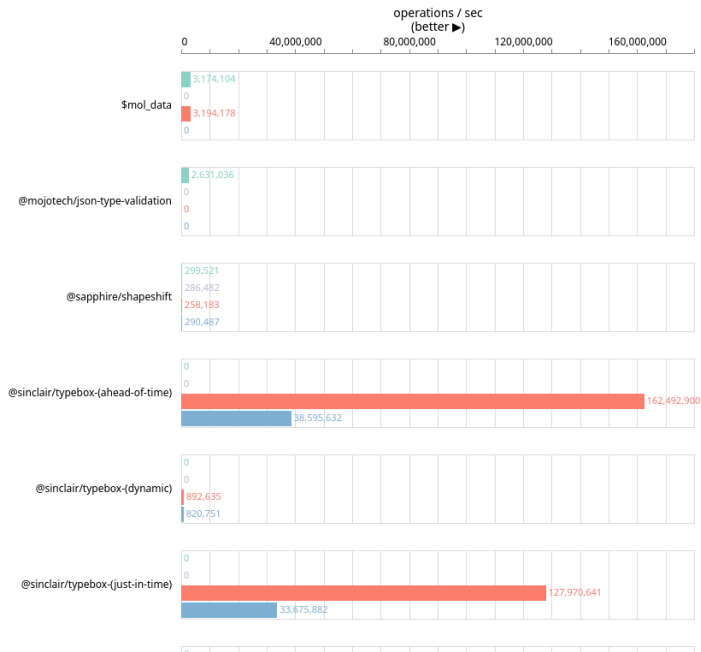
case study

benchmark viewer prototype

uses react + only **sqlite** for state

Benchmark

- ✓ assertLoose
- ✓ assertStrict
- ✓ parseStrict
- ✓ parseSafe



example code (benchmark viewer)

```
const BenchmarkApp() => (  
  <DatabaseContextProvider>  
    <ResultsTable />  
  </DatabaseContextProvider>  
)
```

```
const Graph() => (  
  const results = useDatabase((db) => {  
    return db.findResults();  
  });  
  
  return (  
    <table>  
      {results?.map((r, i) => (...))}  
    </table>  
  )  
)
```

pglite inline SQL

<https://pglite.dev/docs/framework-hooks/react>

```
import { useLiveQuery } from '@electric-sql/pglite-react'

const MyComponent = () => {
  const maxNumber = 100;
  const items = useLiveQuery.sql`
    SELECT *
    FROM my_table
    WHERE number <= ${maxNumber}
    ORDER BY number;
  `

  return ...
};
```

90s shit (digression)

90s shit (digression)

some learnings for me

CHORE: the database needs a schema first

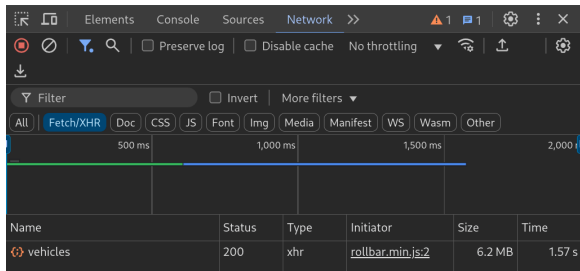
TYPES: without a good orms/query builders, its untyped and that sucks

FUN: listing / sorting stuff is so much easier in SQL

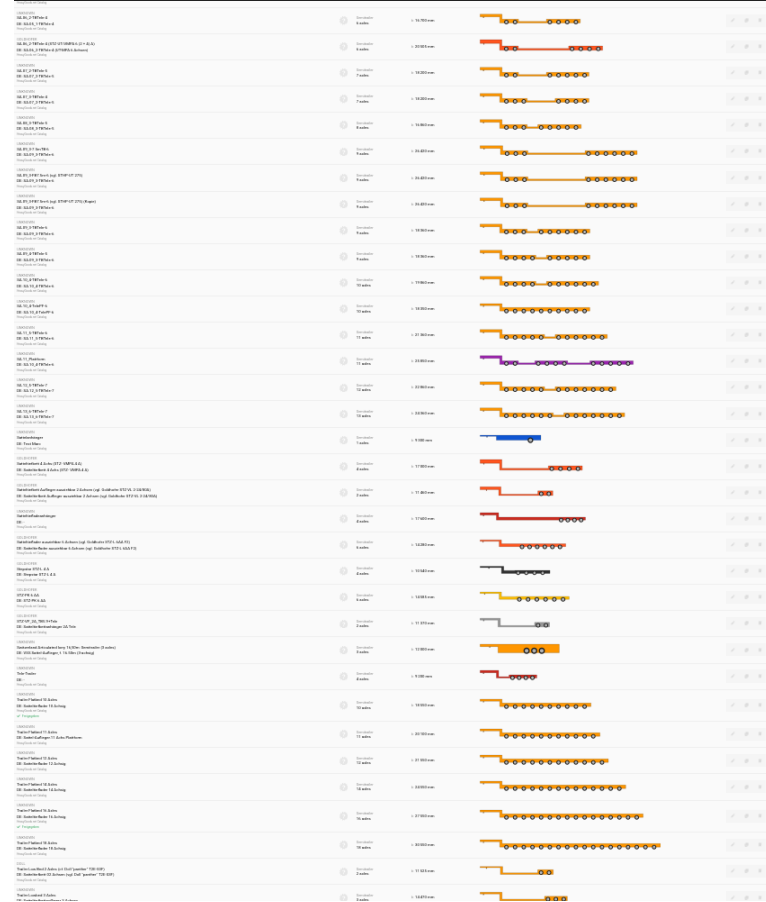
PERF: cannot memoize from SQL don't have object identity

why do I care?

at work, users apparently like the software and use it



data always accumulates 🗄️ 🗄️



recap

easy to install / use / operate SQL databases exist

they have downsides

they might make some things easier than before

welcome a new 🛠️ in your toolbelt

SQL was already old when I started coding, 20yrs later and it looks well alive.

thanks for following along

github.com/hoeck

heavygoods.net/en/news/job-full-stack-developer

dresdenjs.io