Working with resolvers and Vercel Postgres

26 May, 2023

 

Jamie Barton Josep Vidal

Resolvers are a powerful way to extend your Grafbase backend. We can use resolvers to compute custom business logic, make network requests, invoke dependencies, and perform various operations.

In this guide, we'll create a new GraphQL API using Grafbase with resolvers and leverage Vercel Postgres to handle OpenAI vector embeddings.

Create a backend with Grafbase

To get started, create a new directory or navigate to an existing project and run the following command:

We'll create a system to store and query vector embeddings generated by the OpenAI API.

Open the file `grafbase/schema.graphql` and replace the contents with the following schema:

```
type Embedding {
  id: ID!
  text: String!
  vector: JSON!
}

extend type Query {
  getEmbedding(id: ID!): Embedding @resolver(name: "get-embedding")
  searchEmbeddings(query: String!): [Embedding!]!
    @resolver(name: "search-embeddings")
}

extend type Mutation {
  createEmbedding(input: EmbeddingInput!): Embedding!
    @resolver(name: "create-embedding")
}

input EmbeddingInput {
  text: String!
}
```

Set up Vercel Postgres

Go to Vercel and create an account if you don't have one. Then, you can follow the quick start guide on Vercel.

Once your database is created, and your `.env` file populated by using `vercel env pull .env.development.local`, you are ready to continue, `@vercel/postgres` automatically retrieves the connection string to connect to the db from the `.env` file.

Set up the database

Before continuing, you must initialize the table that we are going to use to store our data, for that use any Postgres client and run:

```
CREATE TABLE IF NOT EXISTS embeddings (
    id SERIAL PRIMARY KEY,
    text TEXT NOT NULL,
    vector VECTOR(1536) NOT NULL
);
```

Create the resolvers

We'll now create the resolvers for handling the vector embeddings.

First install `@vercel/postgres` by typing `yarn add @vercel/postgress` or `npm install @vercel/postgress`

Create the file `grafbase/resolvers/get-embedding.js` and add the following code:

```
import { db } from '@vercel/postgres'

export default async function Resolver(_, { id }) {
  const client = await db.connect()

  return await client.sql`SELECT * FROM embeddings WHERE id = ${id}`
}
```

Create another file `grafbase/resolvers/search-embeddings.js` and type:

```
import { db } from '@vercel/postgres'

export default async function Resolver(_, { query }) {
  const client = await db.connect()

  return await client.sql`SELECT * FROM embeddings WHERE text ILIKE '%${query}%'`
}
```

Finally, create the file `grafbase/resolvers/create-embedding.js`:

```
import { db } from '@vercel/postgres'

export default async function Resolver(_, { input }) {
  const client = await db.connect()
```

```
  // Generate the vector using OpenAI embeddings API
  const response = await fetch(
    'https://api.openai.com/v1/engines/davinci-codex/completions',
    {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        // You must define OPENAI_API_KEY on your .env file
        Authorization: `Bearer ${process.env.OPENAI_API_KEY}`
      },
      body: JSON.stringify({
        prompt: text,
        max_tokens: 1
      })
    }
  )

  const data = await response.json()
  const vector = data.choices[0].text

  return await client.sql`INSERT INTO embeddings (text, vector) VALUES ('${text}', '$
}
```

Try it out

Start the Grafbase development server by running the following command in your project directory:

```
npx grafbase dev
```

You can now interact with the GraphQL API by visiting http://localhost:4000 in your browser.

Create an embedding

To create an embedding, run the following mutation:

```
mutation {
  createEmbedding(input: { text: "Embedding Example" }) {
    text
  }
```

This will create a new embedding with the provided text and vector.

Get an embedding

To retrieve an embedding by its ID, run the following query:

```
query {
  getEmbedding(id: "EMBEDDING_ID") {
    id
    text
    vector
  }
}
```

Replace `EMBEDDING_ID` with the actual ID of the embedding you want to retrieve.

Search embeddings

To search for embeddings based on a query string, run the following query:

```
query {
  searchEmbeddings(query: "Example") {
    id
    text
    vector
  }
}
```

Replace `"Example"` with your desired query string.

That's it! You now have a Grafbase backend with resolvers connected to Vercel Postgres, allowing you to store and query OpenAI vector embeddings. Feel free to explore and expand upon this setup to suit your needs.

Start building your backend of the future now.

Start Building

Grafbase

Product

Sign Up
Documentation
Templates
CLI
Changelog
Guides
Roadmap

Resources

System Status
Privacy Policy
Terms of Use
Fair Use Policy

Frameworks

Next.js
Nuxt
SvelteKit
Remix
SolidJS
Deno
Fresh

Company

About
Blog
Careers
Community
Contact Us
Pricing
Security

React

Kotlin

Vue