Chichi’s Learning Materials
SQL Querying with Go

Karol Moroz

December 8, 2024

Abstract

The goal of these materials is to familiarize the student with SQL database querying techniques in Go. To this end, we are going to write a very similar thing over and over again.

1 Initial setup

In this section, we are going to set up a PostgreSQL instance in a Docker container and populate it with data from the Northwind Traders database, which is an example database created by Microsoft.

See this link for a diagram of the database schema.

Download this file to a local directory, e. g. `~/working/northwind/northwind.sql`. In the same directory, create a file named `docker-compose.yml` with the following contents:

```yaml
services:
  db:
    image: "postgres:17"
    environment:
      POSTGRES_DB: northwind
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
    volumes:
      - "pg_data:/var/lib/postgres/data"
      - "./northwind.sql:/docker-entrypoint-initdb.d/northwind.sql"
      - "./files:/files"

    ports:
      - "5433:5432"

volumes:
  pg_data: {}
```

In a terminal window, `cd` into the directory containing `docker-compose.yml` and `northwind.sql` and run:

```
docker compose up
```

If everything goes correct, you should be able to connect to the database using `psql` or o
GUI, like TablePlus or DBeaver on port 5433:

```
$ psql "postgres://postgres:postgres@localhost:5433/northwind"
psql (17.2 (Debian 17.2-1.pgdg120+1))
Type "help" for help.

northwind=#
```

2 Service pattern: querying for a single record

In a previous assignment, we have learned to build “service” types to encapsulate logic related to database queries.

For this assignment, create a new Go project. Within this project, we are going to create two packages: a `types` package where we define data structures for the data we will be fetching from the database, like so:

```go
// northwind/types/supplier.go
package types

type Supplier struct {
  SupplierID   int
  CompanyName  string
  ContactName  string
  ContactTitle string
  Address      string
  City         string
  Region       string
  PostalCode   string
  Country      string
  Phone        string
  Fax          string
  Homepage     string
}
```

Another package will be `services`, where we define our service types, like so:

```go
// northwind/services/supplier_service.go
package services

import (
  "database/sql"
  "github.com/moroz/go-teaching/types"
)

type SupplierService struct {
  db *sql.DB
}

func NewSupplierService(db *sql.DB) SupplierService {
  return SupplierService{db}
}

const getSupplierByIDQuery = `select supplier_id, company_name,
contact_name, contact_title, address, city, region, postal_code,
country, phone, fax, homepage from suppliers where supplier_id = $1`

func (s *SupplierService) GetSupplierByID(id int) (*types.Supplier, error) {
  var i types.Supplier
  err := s.db.QueryRow(getSupplierByIDQuery, id).Scan(
    &i.SupplierID,
    &i.CompanyName,
    &i.ContactName,
    &i.ContactTitle,
    &i.Address,
    &i.City,
    &i.Region,
    &i.PostalCode,
    &i.Country,
    &i.Phone,
    &i.Fax,
    &i.Homepage,
  )
  return &i, err
}
```

Based on the example above and the Northwind Traders database, define a data structure for each of the following tables: `suppliers`, `products`, `categories`, `customers`, `employees`, `us_states`, `orders`, `shippers`. The name of a structure should be the singular form of the table name, e. g. `table products` => `type Product struct ...`

Each of the types should be defined in a separate source file, e. g. `type Product` should be defined in `types/product.go`.

For each of these tables, define a service type with a single method that simply fetches a record by its database ID, e. g. `GetProductByID`, with logic similar to the provided code samples. Please ensure that all column and table names are the same as in the actual database.