Implementare un ORM di base per PostgreSQL in Go

In questo articolo vedremo come costruire un semplice ORM (Object-Relational Mapping) in Go per interfacciarsi con un database PostgreSQL. L'obiettivo è comprendere i concetti fondamentali di un ORM: mappatura delle strutture Go alle tabelle SQL, generazione dinamica delle query e gestione dei risultati.

Requisiti

- Go 1.18 o superiore
- Database PostgreSQL
- Libreria `database/sql` e driver `lib/pq`

Connessione al database

```
import (
  "database/sql"
  _ "github.com/lib/pq"
  "log"
)

var db *sql.DB

func Connect() {
  var err error
  db, err = sql.Open("postgres", "user=postgres dbname=testdb sslmode=disable")
  if err != nil {
    log.Fatal(err)
  }
}
```

Struttura base del modello

Definiamo una struttura base da cui erediteranno gli altri modelli:

```
type Model interface {
  TableName() string
}

type User struct {
  ID    int
  Name  string
  Email string
}

func (u User) TableName() string {
  return "users"
}
```

Funzione per salvare un record

Una funzione generica per inserire un record nel database:

```
import (
  "fmt"
  "reflect"
  "strings"
)

func Insert(model Model) error {
  v := reflect.ValueOf(model)
  t := reflect.TypeOf(model)

  var columns []string
  var placeholders []string
  var values []interface{}

  for i := 0; i < t.NumField(); i++ {
    field := t.Field(i)
    columns = append(columns, strings.ToLower(field.Name))
    placeholders = append(placeholders, fmt.Sprintf("$%d", i+1))
    values = append(values, v.Field(i).Interface())
  }

  query := fmt.Sprintf("INSERT INTO %s (%s) VALUES (%s)",
    model.TableName(),
    strings.Join(columns, ", "),
    strings.Join(placeholders, ", "),
  )

  _, err := db.Exec(query, values...)
  return err
}
```

Esempio d'uso

```
func main() {
  Connect()

  user := User{
    Name:  "Mario Rossi",
    Email: "mario@example.com",
  }

  err := Insert(user)
  if err != nil {
    log.Fatal(err)
  }
}
```

Creazione della tabella PostgreSQL

Nel database PostgreSQL, assicurati di avere la tabella appropriata:

```
CREATE TABLE users (
  id SERIAL PRIMARY KEY,
  name TEXT,
  email TEXT
);
```

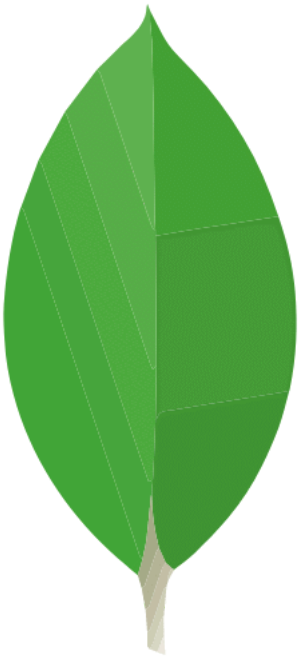
Conclusione

Abbiamo visto come realizzare una versione semplificata di un ORM in Go. Questo approccio può essere esteso per includere funzionalità come aggiornamenti, cancellazioni, query con filtri, validazioni e gestione delle relazioni tra entità. Tuttavia, per progetti reali, è consigliato utilizzare ORM maturi come GORM o SQLBoiler.

Applicazioni Correlate

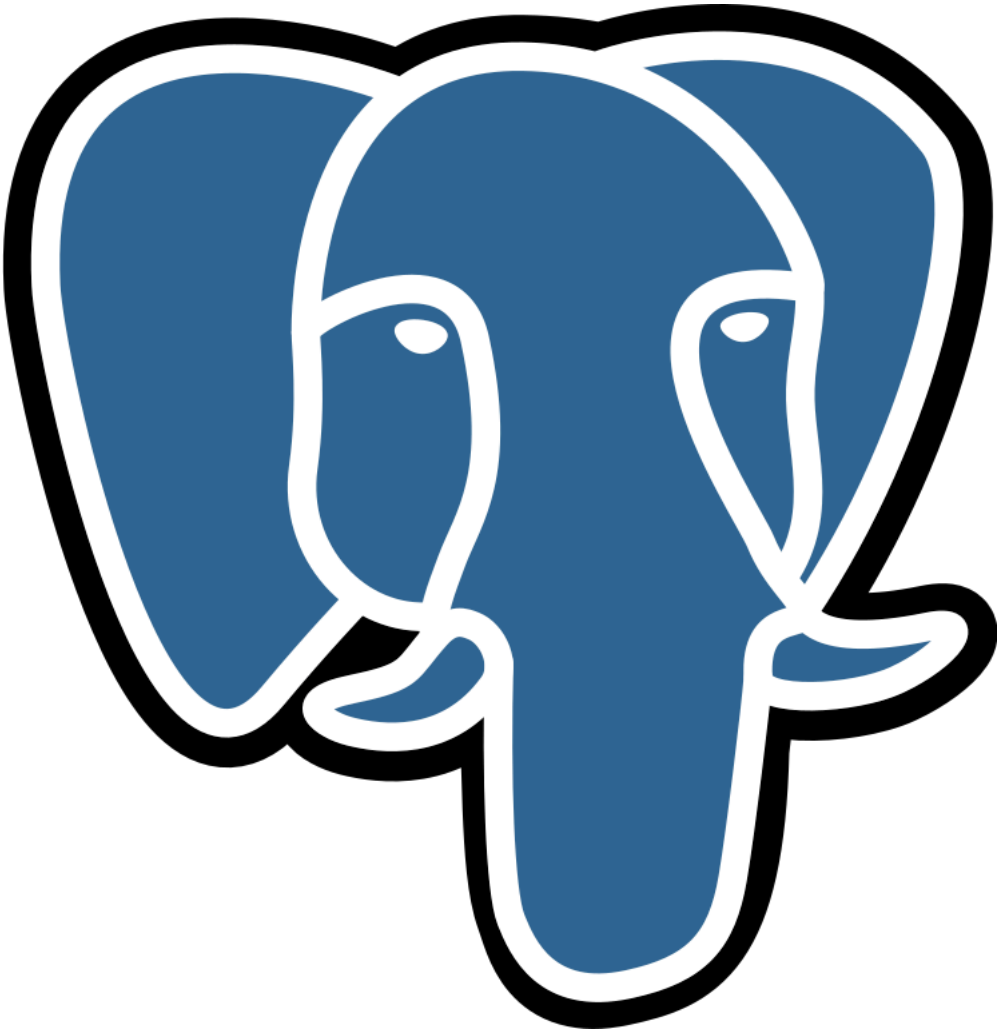
- ### Go Placeholder Image

  Applicazione in Go per la creazione di immagini segnaposto.
  DockerDocker ComposeGoJavaScript



Go MongoDB App

Applicazione basata su MongoDB ed implementata in Go con il driver ufficiale.
DockerDocker ComposeGoMongoDB



Go PostgreSQL App

Applicazione basata su PostgreSQL e sviluppata in Go.
DockerDocker ComposeGoPostgreSQL