

넬포유 10201 구분무

ORM

#gin #ServerLessDB

ORM을 쓰는 이유



개발자

```
INSERT INTO users (username, password)  
VALUES (admin,123456);
```

ORM을 쓰는 이유 컬럼추가



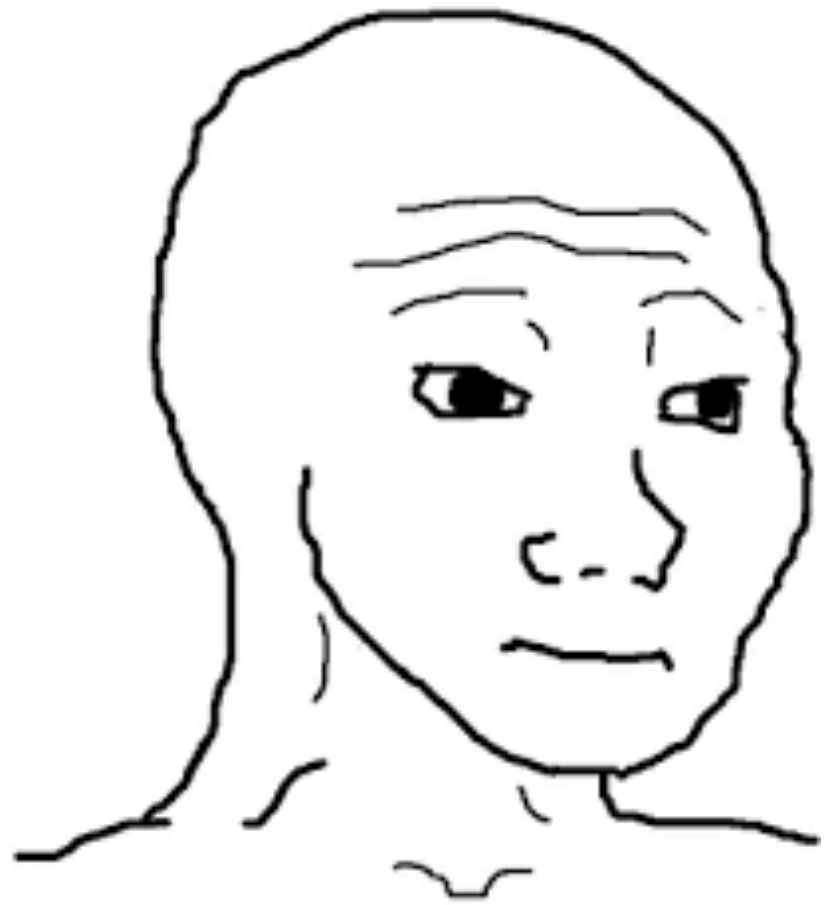
개발자

```
INSERT INTO users (username, password)  
VALUES (admin, 123456);
```

users 테이블에 email 컬럼 추가됐습니다!

ORM을 쓰는 이유 컬럼추가

코드 계속 수정해야함



개발자

```
INSERT INTO users (username, email, password)
VALUES (admin,myemail@com ,123456);
```

users 테이블에 email 컬럼 추가됐습니다!

ORM을 쓰는 이유 노가다

컬럼이 너무 많아(노가다)



개발자

```
INSERT INTO users (username, email, password, age, phone, address)  
VALUES (admin,myemail@com ,123456, 67, 01082714229,eunpyung);
```

ORM을 쓰는 이유 DBMS 변경



개발자

```
SELECT CONCAT(first_name, last_name) FROM users;
```

저희 DBMS mysql에서 포스트그레스로 바꿉니다

ORM을 쓰는 이유 DBMS 변경

문법 갈아엎기 이렇게 100개 더 있으면...



개발자

```
SELECT first_name || last_name FROM users;
```

저희 DBMS mysql에서 포스트그레스로 바꿉니다

ORM을 쓰는 이유 DBMS 변경

문법 갈아엎기 이렇게 100개 더 있으면...



개발자

```
SELECT first_name || last_name FROM users;
```

저희 DBMS mysql에서 포스트그레스로 바꿉니다

ORM으로 해결할수있다!

Object-Relational Mapping

각 언어별로 유명한 ORM들



JPA



Prisma



Django

각 언어별로 유명한 ORM들



JPA

스프링부트전용 ORM



Prisma



Django

각 언어별로 유명한 ORM들



JPA

스프링부트전용 ORM



Prisma

Nodejs 계열 웹프레임워크 사용 가능



Django

각 언어별로 유명한 ORM들



JPA

스프링부트전용 ORM



Prisma

Nodejs 계열 웹프레임워크 사용 가능



Django

프레임워크 내에 내장되어있음

각 언어별로 유명한 ORM들



Go 전용 ORM GORM

진 프레임워크란?



Go 언어 전용 웹 프레임워크

진 프레임워크란?



VS



스프링부트에 비해 내용이 얇고 문법이 간단한편

진 프레임워크란?



VS



스프링부트에 비해 내용이 얇고 문법이 간단한편

Gin (Go)	약 10MB ~ 25MB
Spring Boot (Java)	약 300MB ~ 500MB

진 프레임워크란?

스레드



고루틴

VS



JVM

GORM사용해보기

github.com/go-gorm/gorm

go-gorm / gorm

Type / to search

<> Code

Issues 457

Pull requests 80

Agents

Discussions

Actions

Projects

Wiki

Security and quality

Insights

GORM

gorm

Public

Sponsor

Watch 491

Fork 4.2k

Starred 40k

master 15 Branches 118 Tags

Go to file T

Add file

<> Code

tr1v3r

ci: bump golangci-lint to v2.13 for current Go stable compatibility (#...)

b3d3bf2 · last week

🕒 2,799 Commits

.github	ci: bump golangci-lint to v2.13 for current Go stable comp...	last week
callbacks	Fix potential rows leak on panic by deferring rows.Close() (...)	3 months ago
clause	Add package comments to fix ST1000 warnings (#7708)	6 months ago
internal	Add package comments to fix ST1000 warnings (#7708)	6 months ago
logger	perf: replace fmt.Sprintf with strconv in ExplainSQL numer...	3 months ago
migrator	correct typo and rename fileType to fieldType in AlterColu...	4 months ago
schema	Add package comments to fix ST1000 warnings (#7708)	6 months ago
tests	Fix potential rows leak on panic by deferring rows.Close() (...)	3 months ago
utils	Add package comments to fix ST1000 warnings (#7708)	6 months ago
.gitignore	Adjust ToStringKey use unpack params, fix pass []any as a...	4 years ago
.golangci.yml	use golangci replace reviewdog (#7426)	last year
CODE_OF_CONDUCT.md	Create CODE_OF_CONDUCT.md (#7240)	2 years ago
LICENSE	chore: update copyright year (#7332)	last year
README.md	Update README.md (#7635)	11 months ago

About

The fantastic ORM library for Golang, aims to be developer friendly

[gorm.io](#)

[go](#) [golang](#) [gorm](#) [orm](#) [web](#)

Readme

MIT license

Code of conduct

Activity

Custom properties

40.0k stars

491 watching

4.2k forks

Report repository

Releases 12

Release v1.31.2 Latest

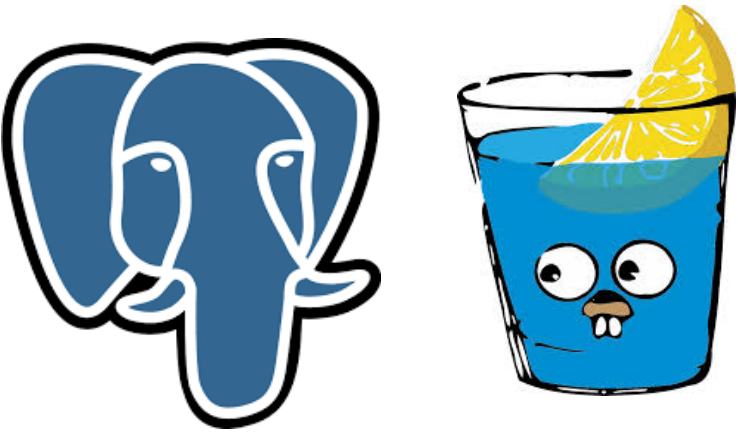
3 months ago

+ 11 releases

Sponsor this project

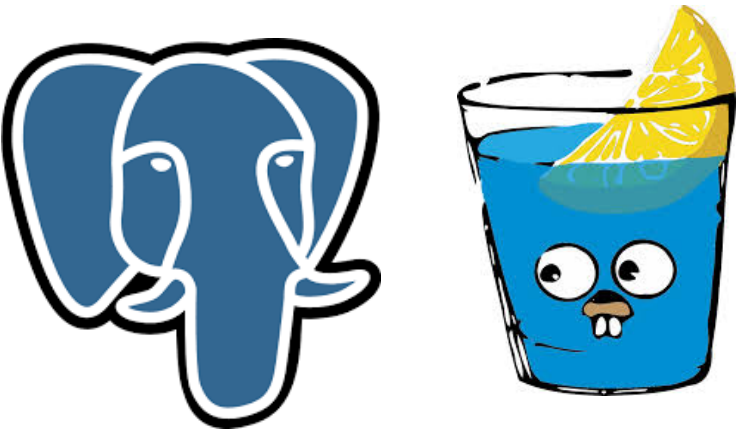
jinzhu Jinzhu

GORM사용해보기



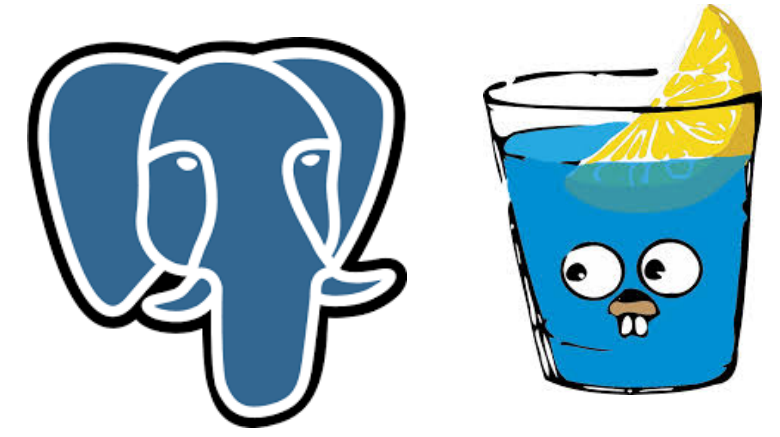
```
gin_study=> SELECT * FROM users;
  id | username | email | create_at
-----+-----+-----+-----
  1 | 구본무 | kubonmo519@gmail.com | 2026-09-14 20:06:21.893633+09
```

GORM사용해보기



```
gin_study=> SELECT * FROM users;
  id | username | email | create_at
-----+-----+-----+-----
  1 | 구본무 | kubonmo519@gmail.com | 2026-09-14 20:06:21.893633+09
```

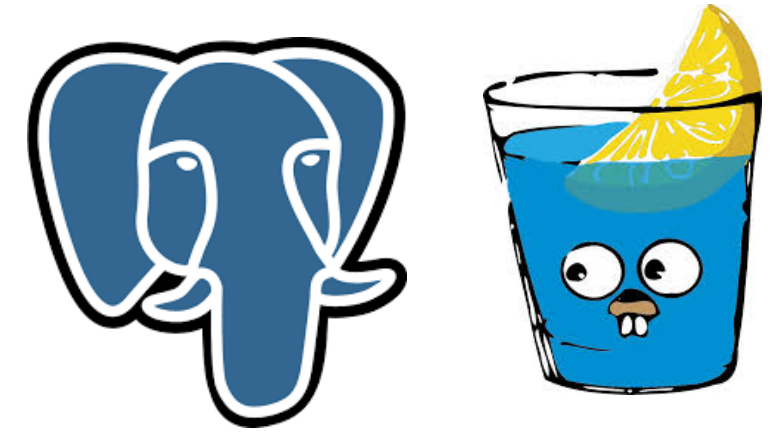
GORM사용해보기



구조체 태그 지정

```
type User struct {  
    ID          int64      `gorm:"primaryKey" json:"id"`  
    Username    string     `gorm:"column:user_name" json:"username"`  
    Email       string     `json:"email"`  
    CreateAt    time.Time  `json:"create_at"`  
}
```

GORM사용해보기

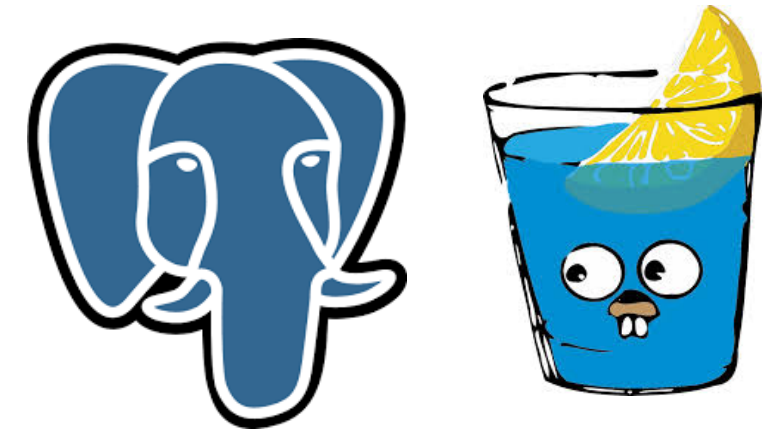


구조체 태그 지정

```
type User struct {  
    ID          int64      `gorm:"primaryKey" json:"id"`  
    Username    string     `gorm:"column:user_name" json:"username"`  
    Email       string     `json:"email"`  
    CreateAt    time.Time  `json:"create_at"`  
}
```

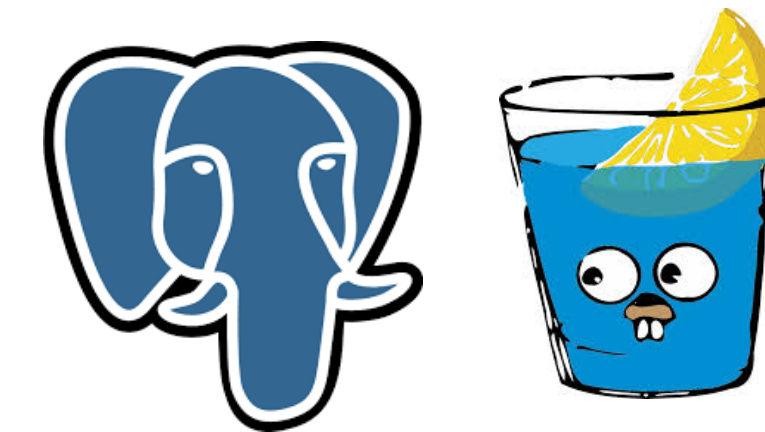
GIN에서 구조체는 많은 역할을 함,

GORM사용해보기



```
func (s *UserStore) List(ctx context.Context) ([]model.User, error) {
    rows, err := s.db.Query(ctx, `SELECT id, username, email, create_at FROM users ORDER BY id`)
    if err != nil {
        return nil, err
    }
    defer rows.Close()
    users := make([]model.User, 0)
    for rows.Next() {
        var u model.User
        if err := rows.Scan(&u.ID, &u.Username, &u.Email, &u.CreateAt); err != nil {
            return nil, err
        }
        users = append(users, u)
    }
    return users, rows.Err()
}
```

GORM사용해보기

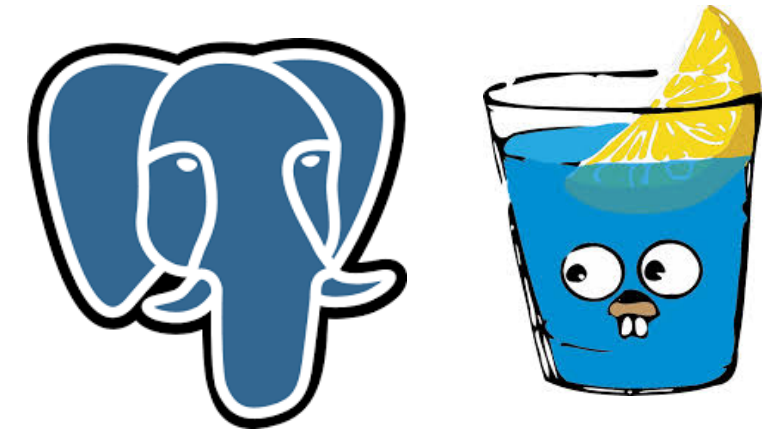


```
func (s *UserStore) List(ctx context.Context) ([]model.User, error) {
    rows, err := s.db.Query(ctx, `SELECT id, username, email, create_at FROM users ORDER BY id`)
    if err != nil {
        return nil, err
    }
    defer rows.Close()
    users := make([]model.User, 0)
    for rows.Next() {
        var u model.User
        if err := rows.Scan(&u.ID, &u.Username, &u.Email, &u.CreateAt); err != nil {
            return nil, err
        }
        users = append(users, u)
    }
    return users, rows.Err()
}
```

너무 길고 복잡해요!

```
func (h *UserHandler) ListUsers(c *gin.Context) {
    users, err := h.service.ListUsers(c.Request.Context())
    if err != nil {
        c.JSON(500, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, users)
}
```

GORM사용해보기

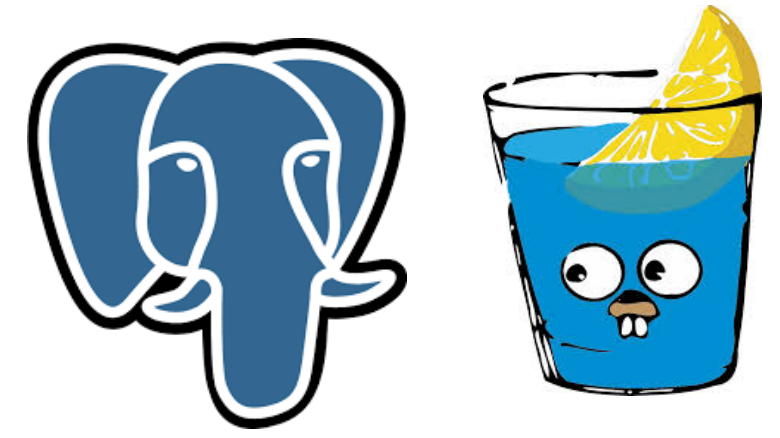


```
func (s *UserStore) List(ctx context.Context) ([]model.User, error) {  
    users := make([]model.User, 0)  
    err := s.db.WithContext(ctx).Order("id").Find(&users).Error  
    return users, err  
}
```

엄청 짧아짐

```
func (h *UserHandler) ListUsers(c *gin.Context) {  
    users, err := h.service.ListUsers(c.Request.Context())  
    if err != nil {  
        c.JSON(500, gin.H{"error": err.Error()})  
        return  
    }  
    c.JSON(http.StatusOK, users)  
}
```

GORM사용해보기(차이점)

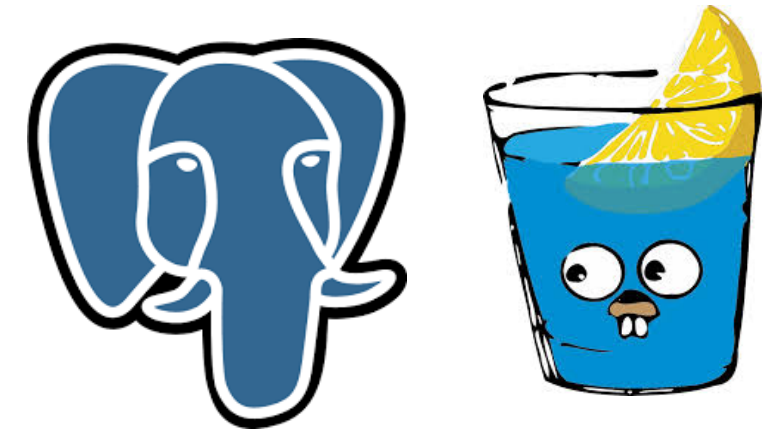


```
func (s *UserStore) List(ctx context.Context) ([]model.User, error) {  
    users := make([]model.User, 0)  
    err := s.db.WithContext(ctx).Order("id").Find(&users).Error  
    return users, err  
}
```

쿼리문이 어디갔지???

```
func (s *UserStore) List(ctx context.Context) ([]model.User, error) {  
    rows, err := s.db.Query(ctx, `SELECT id, username, email, create_at FROM users ORDER BY id`)  
    if err != nil {  
        return nil, err  
    }  
    defer rows.Close()  
    users := make([]model.User, 0)  
    for rows.Next() {  
        var u model.User  
        if err := rows.Scan(&u.ID, &u.Username, &u.Email, &u.CreateAt); err != nil {  
            return nil, err  
        }  
        users = append(users, u)  
    }  
    return users, rows.Err()  
}
```

GORM사용해보기(차이점)

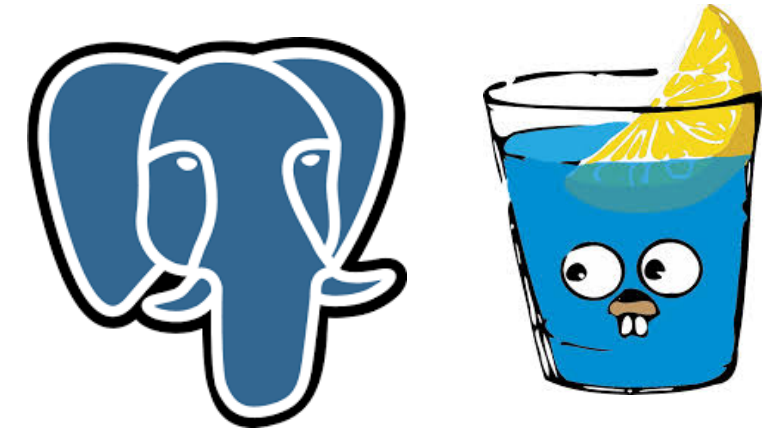


```
func (s *UserStore) List(ctx context.Context) ([]model.User, error) {  
    users := make([]model.User, 0)  
    err := s.db.WithContext(ctx).Order("id").Find(&users).Error  
    return users, err  
}
```

SELECT * FROM users ORDER BY id;

```
func (s *UserStore) List(ctx context.Context) ([]model.User, error) {  
    rows, err := s.db.Query(ctx, `SELECT id, username, email, create_at FROM users ORDER BY id`)  
    if err != nil {  
        return nil, err  
    }  
    defer rows.Close()  
    users := make([]model.User, 0)  
    for rows.Next() {  
        var u model.User  
        if err := rows.Scan(&u.ID, &u.Username, &u.Email, &u.CreateAt); err != nil {  
            return nil, err  
        }  
        users = append(users, u)  
    }  
    return users, rows.Err()  
}
```

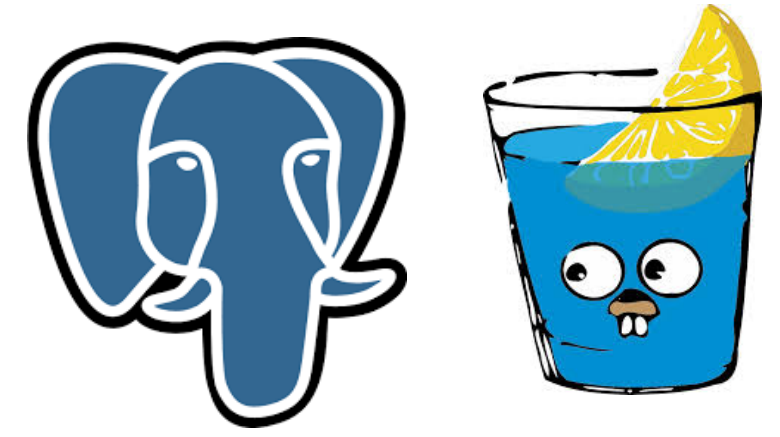
GORM사용해보기(차이점)



```
type User struct {  
    ID          int64      `gorm:"primaryKey" json:"id"`  
    Username    string     `gorm:"column:user_name" json:"username"`  
    Email       string     `json:"email"`  
    CreateAt    time.Time  `json:"create_at"`  
}
```

User 모델의 여러 행을 조회해서 이 슬라이스에 넣어야 하는구나
ORM이 판단 가능

GORM사용해보기



ORM 패키지 함수들 총집합

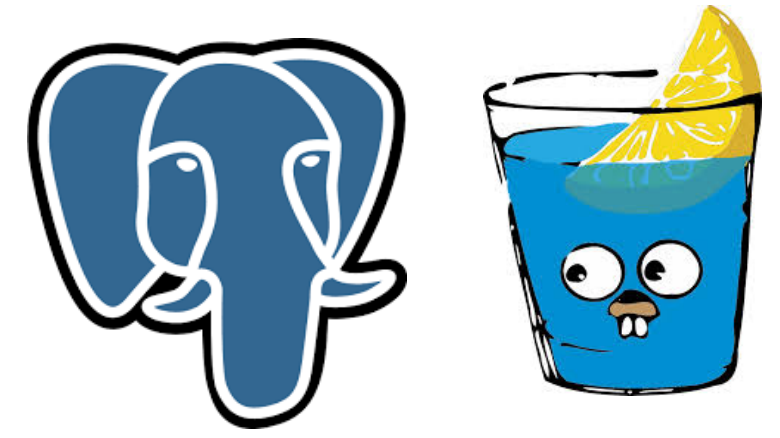
https://gorm.io/ko_KR/docs/query.html

GORM사용해보기 - 메서드들

users는 슬라이스(배열/리스트) 변수

```
db.Find(&users)
```

테이블의 여러 행 조회

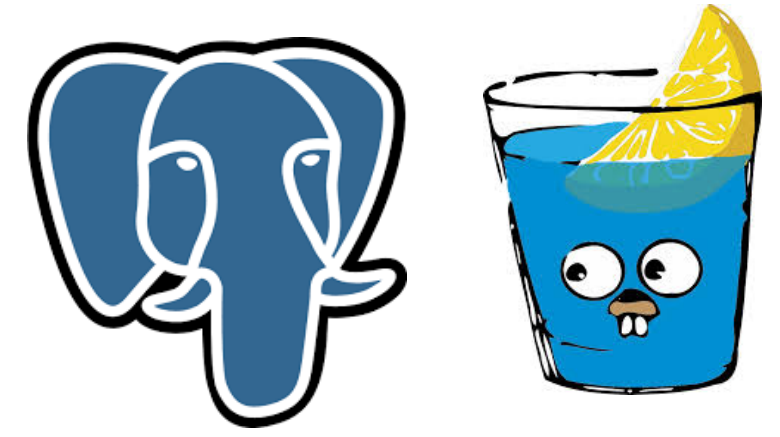


GORM사용해보기 - 메서드들

users는 슬라이스(배열/리스트) 변수

db.Find(&users)
테이블의 여러 행 조회

db.First(&user)
첫번째 행 조회



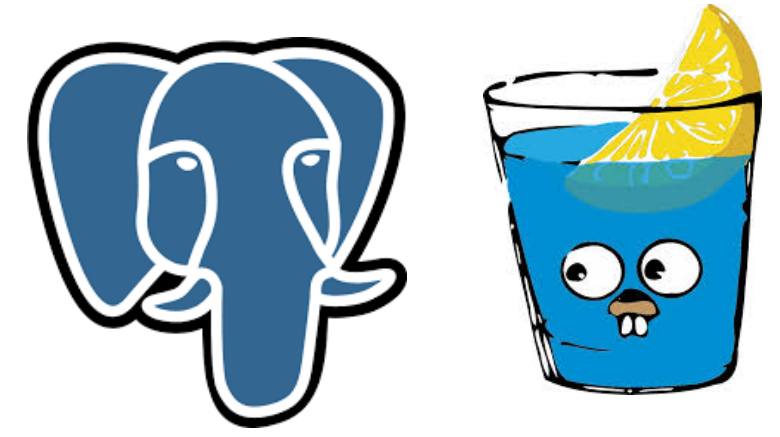
GORM사용해보기 - 메서드들

users는 슬라이스(배열/리스트) 변수

db.Find(&users)
테이블의 여러 행 조회

db.First(&user)
첫번째 행 조회

db.First(&user, id)
ID로 한 행 조회



GORM사용해보기 - 메서드들

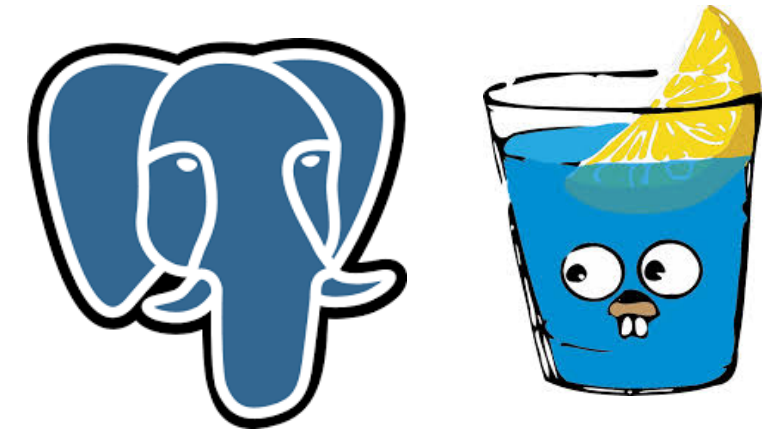
users는 슬라이스(배열/리스트) 변수

db.Find(&users)
테이블의 여러 행 조회

db.First(&user)
첫번째 행 조회

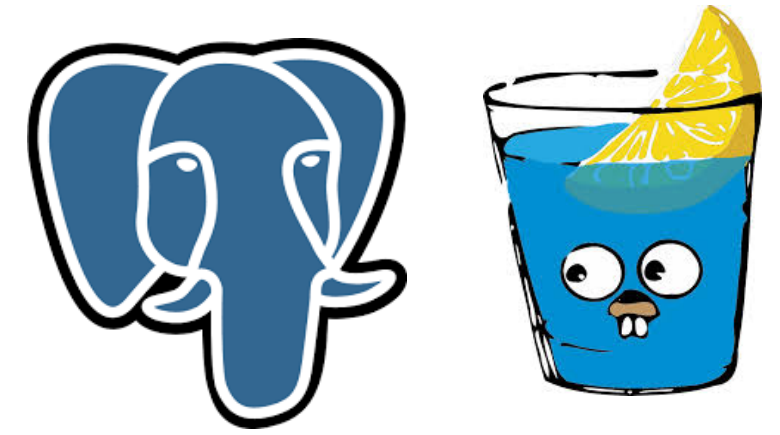
db.First(&user, id)
ID로 한 행 조회

db.Where("email = ?", email).First(&user)
조건에 맞는 첫 행 조회



GORM사용해보기 - 메서드들

users는 슬라이스(배열/리스트) 변수



db.Find(&users)
테이블의 여러 행 조회

db.First(&user)
첫번째 행 조회

db.First(&user, id)
ID로 한 행 조회

db.Where("email = ?", email).First(&user)
조건에 맞는 첫 행 조회

db.Order("id DESC").Find(&users)
정렬해서 조회

사용하다 느낀점

시퀀스를 좀더 배워야겠다
sqld를 따라하나?...

서버리스 NEONDB

서버리스란?

자기가 직접 관리 안해도 된다!
과금 문제 그런거 신경 안써도됨 AWS에서 치면 람다

서버리스 NEONDB

서버리스란?

자기가 직접 관리 안해도 된다!
과금 문제 그런거 신경 안써도됨 AWS에서 치면 람다
관리 안해도 되는 DB가 있다면?

서버리스 NEONDB

서버리스란?

자기가 직접 관리 안해도 된다!
과금 문제 그런거 신경 안써도됨 AWS에서 치면 람다
관리 안해도 되는 DB가 있다면?



서버리스 NEONDB

NEON

myProject

production

Connect

Overview

Integrations

Settings

Monitoring

Postgres database >

Better Auth

Object storage

Functions

AI Gateway

Onboard your agent

Install Neon skills to give your agent safe access to your Neon projects

Copy command

Feedback

Collapse menu

Search...

Ask AI

Upgrade

bernetes

Branch overview

All projects / myProject

Service

Description / Details

Postgres database

Enabled

1 database, 1 compute

Better Auth

Enabled

Get user management via Better Auth

Object storage

Upload files and media to buckets

Functions

Deploy serverless functions next to your data

AI Gateway

Upgrade to access AI models from

Postgres Monitoring

There is no data to display at the moment.

Create child branch

Share

Branch

Nameproduction

IDbr-damp-water-b3u4...

ExpiresNever

Created2026-09-07 09:47:07

Created byBernetes

Project

NamemyProject

IDwithered-hat-618959...

RegionAWS Asia Pacific 1 (Singapore)

PlanFree plan

Default branchproduction

IP restrictionsNone set

VPCNot configured

Usage

Since Sep 7, 2026

Compute0.9 CU-hrs

Storage32.57 MB

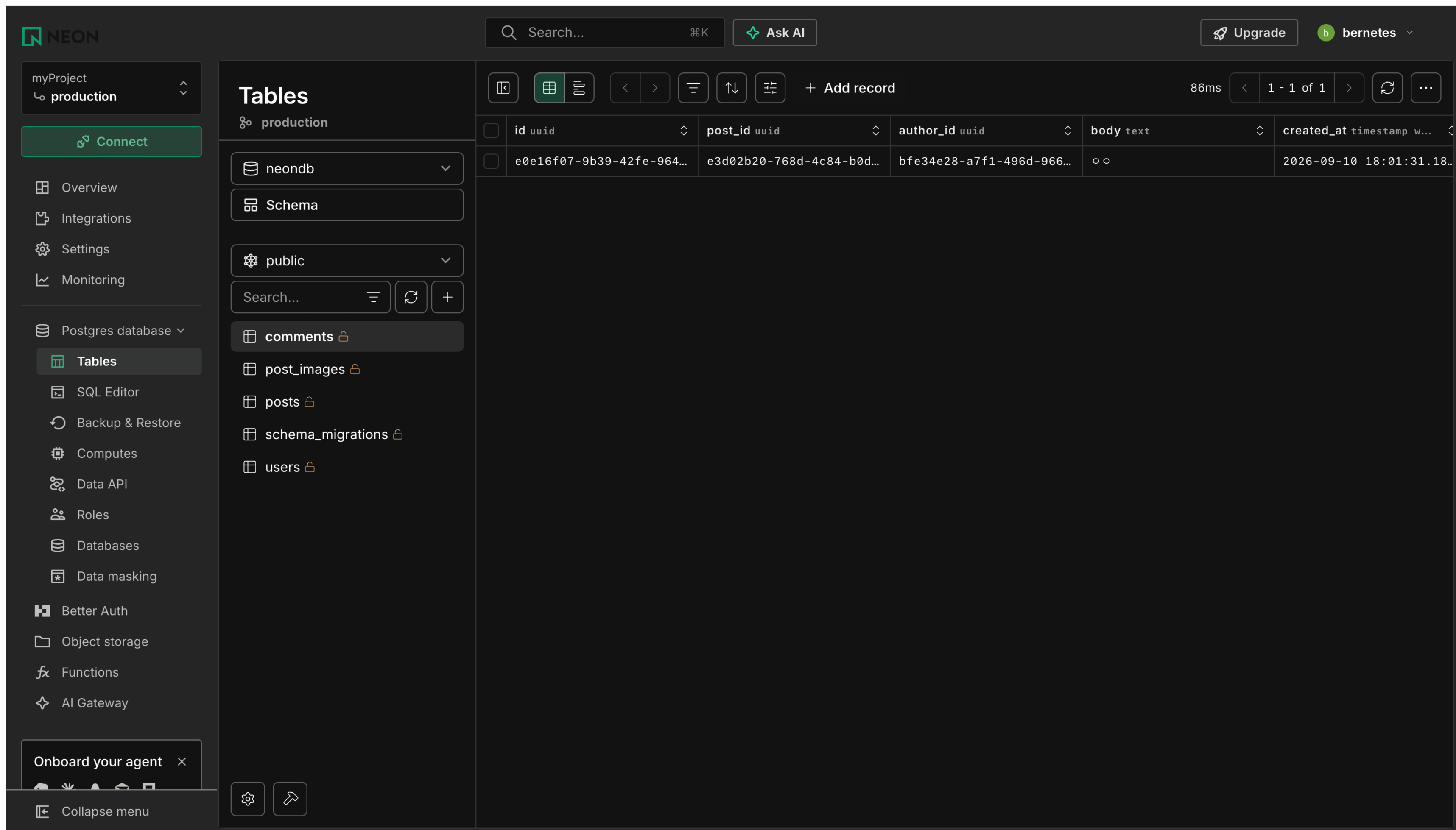
History376 kB

Network transfer328.02 kB

Postgres

Default compute0.25 ↔ 2 CU

서버리스 NEONDB



영알못이 아닌 이상 엄청나게 쉽게 사용 가능 (스키마 화면)

서버리스 NEONDB

```
➤ psql "postgresql://neondb_owner:npg_gr6UdAhRylH2@ep-restless-meadow-b3ez1g0h-pooler.c-4.ap-southeast-1.aws.neon.tech/neondb?sslmode=require&channel_binding=require"
psql (18.6 (Homebrew))
SSL 연결 정보 (프로토콜 : TLSv1.3, 암호화 기법 : TLS_AES_256_GCM_SHA384, 압축 : off, ALPN: postgresql)
도움말을 보려면 "help"를 입력하십시오.

neondb=> SELECT id,display_name
neondb-> FROM users;
      id      | display_name
-----+-----
 158cab4f-5997-4b1f-a70b-bd5208fc9bf1 | Administrator
 bfe34e28-a7f1-496d-9668-33487c87ae2f | 구분무
(2개 행)

neondb=> █
```

터미널에서도 당연히 접근 가능하다.

서버리스 NEONDB

```
➤ psql "postgresql://neondb_owner:npg_gr6UdAhRylH2@ep-restless-meadow-b3ez1g0h-pooler.c-4.ap-southeast-1.aws.neon.tech/neondb?sslmode=require&channel_binding=require"
psql (18.6 (Homebrew))
SSL 연결 정보 (프로토콜 : TLSv1.3, 암호화 기법 : TLS_AES_256_GCM_SHA384, 압축 : off, ALPN: postgresql)
도움말을 보려면 "help"를 입력하십시오.

neondb=> SELECT id,display_name
neondb-> FROM users;
      id      | display_name
-----+-----
 158cab4f-5997-4b1f-a70b-bd5208fc9bf1 | Administrator
 bfe34e28-a7f1-496d-9668-33487c87ae2f | 구분무
(2개 행)

neondb=> █
```

터미널에서도 당연히 접근 가능하다.

서버리스 NEONDB

Provider and region availability

Neon runs on AWS. Lakebase Postgres on Databricks inherits the cloud reach of the Databricks platform, with availability that varies by provider. For the full, current region lists, follow the links below.

Cloud provider	Neon	Databricks
AWS	Yes (Neon regions)	Yes, generally available (AWS regions ↗)
Azure	No, Azure support is being deprecated (Neon regions)	Yes, in beta (Azure regions ↗)
GCP	Not available	Yes, in beta (GCP regions ↗)

AWS환경에서도 있다.

넬포유 10201 구분무

ORM

#gin #ServerLessDB