# Golang 练习笔记02

#### Golang操作postgres数据库

目标

user数据表

数据库支持的操作

SQL shell

SQL shell 操作命令

pgAdmin

查看table信息

查看table内容

创建数据表

**AUTO INCREMENT** 

插入数据库

向表中部分column插入数据

向表中所有column插入数据

查询

综合使用

参考资料

2020/07/16

cyc/Golang 练习笔记/Golang 练习笔记02

# Golang操作postgres数据库

## 目标

完成如下项目的postgres准备

### 项目一

用Golang实现用户注册登录功能

- 打开localhost:3000/signup, 填写注册信息
  - 用户名(验证要求: 手机号码或邮箱)
  - 密码(拥有大小写字母及数字, 至少8位)
- 打开localhost:3000/signin,显示登录框(可填写用户名和密码),登录框下方有按钮可以调到signup页面注册
  - 若登录成功 -> 跳转localhost:3000/profile,显示用户名及Login Succeeded
  - 若登录失败 -> 停留当前页面,显示Login Failed
- 登录状态保持在前端,未登录状态打开localhost:3000/profile, 跳转至localhost:3000/signin

#### 模块需求

- 任选一种web framework https://github.com/mingrammer/go-web-framework-stars
- 数据库使用postgres
- 登录状态缓存使用redis

### user数据表

建立一个user数据表,表中有如下内容

- id 为自动增长主键
- account 为账户名,是电话或者邮箱,以string形式给出
- password 为用户密码,以string形式给出

### 数据库支持的操作

数据库要支持以下操作

- 插入数据, 当用户注册时使用
- 查询数据, 当用户登陆时使用

(没有关于删除和更新的要求, 所以暂不考虑)

### SQL shell

点击SQL shell,输入信息即可连入数据库 输入信息时如果直接按<Enter>就会使用默认的[]中的值

Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
Password for user postgres:

### SQL shell 操作命令

查看已有数据库	\I
进入数据库	\c 数据库名
查看表格之间的关系	\d
查看表格信息	\d 表格名
查看所有命令	\? (按q退出浏览)

#### [postgres=# \l

#### List of databases

Name	0wner	Encoding	Collate	Ctype	Access privileges
postgres   template0	postgres postgres	•	C   C	C   C	  -c/postgres +
· i	postgres	j i	C	   C	postgres=CTc/postgres   =c/postgres +   postgres=CTc/postgres
(3 rows)		•	l	1	postgres—ere, postgres

[postgres=# \c postgres

You are now connected to database "postgres" as user "postgres". postgres=# \d

postgres=# SELECT \* FROM users;

id	account	password
1	szcyc001@123.com cyc101@300.com	

[postgres=# \q

So long and thanks for all the fish.

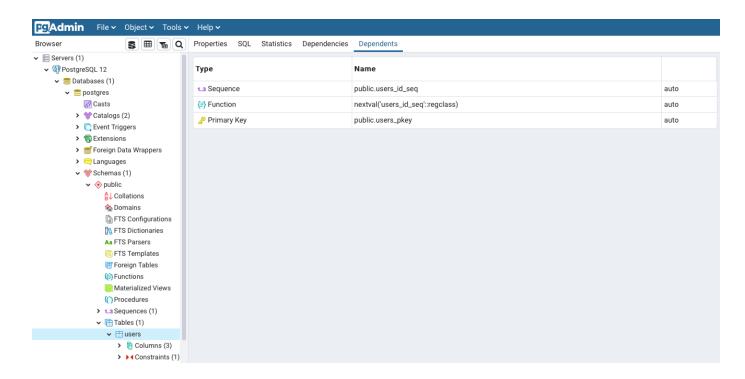
-- Douglas Adams

[Process completed]

## pgAdmin

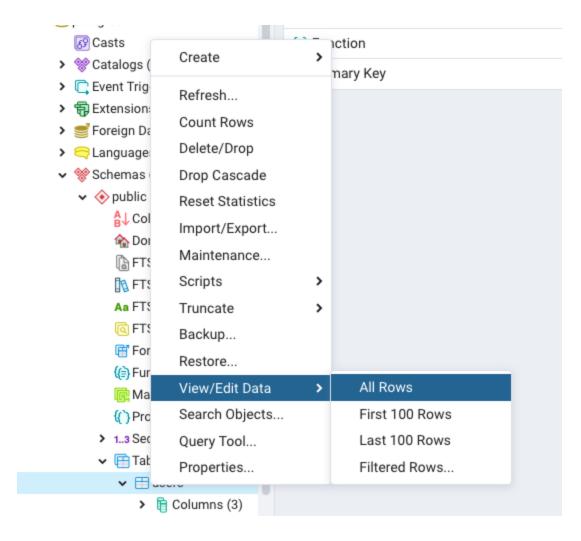
### 查看table信息

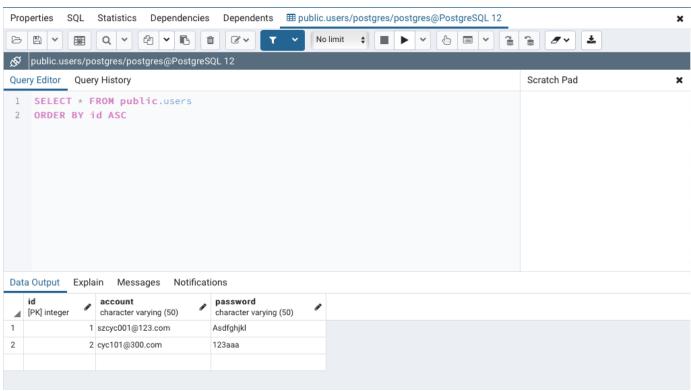
在pgAdmin中可以选择对应的database,选择Schemas中的public即可看到table了选中table可以查看信息



### 查看table内容

选中要查看的table, 右击选择View/Edit data中的All Rows即可





## 创建数据表

```
1 CREATE TABLE table_name(
 column1 datatype,
3 column2 datatype,
    column3 datatype,
    . . . . .
 6 columnN datatype,
 7 PRIMARY KEY( 一个或多个列 )
8);
9
10 --如果主键只有一个
11 CREATE TABLE table_name(
12 column1 datatype PRIMARY KEY,
13 column2 datatype,
    column3 datatype,
    . . . . .
15
16 columnN datatype,
17);
```

#### **AUTO INCREMENT**

使用serial关键字

```
1 CREATE TABLE users
2 (
3    id serial NOT NULL,
4    name text,
5    article text
6 )
```

## 插入数据库

## 向表中部分column插入数据

```
1 INSERT INTO TABLE_NAME (column1, column2, column3,...columnN)
```

```
2 VALUES (value1, value2, value3,...valueN);
```

### 向表中所有column插入数据

```
1 INSERT INTO TABLE_NAME VALUES (value1,value2,value3,...valueN);
```

### 查询

```
1 --读取部分column
 2 SELECT column1, column2,...columnN FROM table_name;
 4 --读取全部数据
5 SELECT * FROM table_name;
 7 --使用where增加条件
8 SELECT column1, column2, columnN
9 FROM table_name
10 WHERE [condition1]
11
12 --条件中可以使用
13 AND
14 OR
15 NOT NULL
16 LIKE
17 IN
18 NOT IN
19 BETWEEN
20 --等关键词来扩充搜索条件
```

## 综合使用

createTable.sql

```
1 DROP TABLE IF EXISTS users;
2 CREATE TABLE users (
3 id SERIAL PRIMARY KEY,
```

```
account VARCHAR(50) NOT NULL,
password VARCHAR(50) NOT NULL
6 );
```

#### connect.go

```
1 package main
2
3 import (
4 "database/sql"
5 "fmt"
6
    "io/ioutil"
7
8
      _ "github.com/lib/pq"
9
10)
11
12 const (
13 host = "localhost"
14
    port
             = 5432
user = "postgres"
password = "123aaa"
17
     dbname = "postgres"
18 )
19
20 var db *sql.DB
21
22 func checkErr(err error) {
23 if err != nil {
24
        panic(err)
25 }
26 }
27
28 func insert(givenAcc string, givenPass string) {
      psqlInfo := fmt.Sprintf("INSERT INTO users(account, password)
  VALUES('%s','%s');", givenAcc, givenPass)
30 _, err := db.Exec(psqlInfo)
31 checkErr(err)
32 }
```

```
33
34 func query(givenAcc string) (hasAccount bool, pass string) {
       hasAccount = false
       pass = ""
36
37
       psqlInfo := fmt.Sprintf("SELECT password FROM users WHERE acc
   ount='%s';", givenAcc)
       info, err := db.Query(psqlInfo)
       checkErr(err)
39
       fmt.Printf("info has type %T\n", info)
40
      for info.Next() {
41
42
           err = info.Scan(&pass)
           checkErr(err)
43
           hasAccount = true
44
       }
45
46
       return
47 } // query
48
49 func main() {
       psqlInfo := fmt.Sprintf("host=%s port=%d user=%s "+
50
51
           "password=%s dbname=%s sslmode=disable",
52
           host, port, user, password, dbname)
53
54
       sqlBytes, err := ioutil.ReadFile("createTable.sql")
       checkErr(err)
55
       sqlCommand := string(sqlBytes)
56
57
58
       var err0penDB error
       db, errOpenDB = sql.Open("postgres", psqlInfo)
59
       fmt.Printf("db has type of %T\n", db)
60
       checkErr(err0penDB)
61
       defer db.Close()
62
63
64
       _, err = db.Exec(sqlCommand)
       checkErr(err)
65
       insert("szcyc001@123.com", "Asdfghjkl")
66
       insert("szcyc003@111.com", "123aaa")
67
       hasAcc, password := query("szcyc001@123.com")
68
       hasAcc2, password2 := query("szcyc001@163.com")
69
       fmt.Printf("%v, %s\n", hasAcc, password)
70
       fmt.Printf("%v, %s\n", hasAcc2, password2)
71
```

```
fmt.Println("Everything done!")
fmt.Println("Everything done!")
fmt.Println("Everything done!")
```

#### 运行结果

```
1 $ go run connect.go
  [ruby-2.6.3p62]
2 db has type of *sql.DB
3 info has type *sql.Rows
4 info has type *sql.Rows
5 true, Asdfghjkl
6 false,
7 Everything done!
```

4	id [PK] integer	account character varying (50)	password character varying (50)
1	1	szcyc001@123.com	Asdfghjkl
2	2	szcyc003@111.com	123aaa

## 参考资料

菜鸟教程——PostgreSQL 教程

https://www.runoob.com/postgresql/postgresql-tutorial.html

Go实战--go语言操作PostgreSQL数据库(github.com/lib/pq)

https://blog.csdn.net/wangshubo1989/article/details/77478838

Go语言使用PostgreSQL数据库

https://www.cnblogs.com/dfsxh/p/10211103.html

go的各种import

https://blog.csdn.net/stpeace/article/details/82901633