

连接数据库

[下载驱动](#)

[连接代码](#)

[初始化数据库](#)

[连接池的连接数量](#)

[SetMaxOpenConns](#)（控制连接查询的次数）

[SetMaxIdleConns](#)（设置连接池中最大空闲连接数）

[数据库操作](#)

[查询单条数据](#)

[查询多条数据](#)

[插入数据和更新数据](#)

下载驱动

▼ 终端输入

Plain Text |

```
1 go get -u github.com/go-sql-driver/mysql
```

连接代码

连接类型：MySQL

用户名：root

密码：wwy040609

```
1 package main
2
3 import (
4     "database/sql"
5     "fmt"
6     _ "github.com/go-sql-driver/mysql"
7 )
8
9 func main() {
10     //连接数据库
11     dsn := "root:wwy040609@tcp(127.0.0.1:3306)/mydb1"
12     //连接数据库
13     db, err := sql.Open("mysql", dsn)
14     if err != nil {
15         fmt.Printf("Open %s failed,err:%v\n", dsn, err)
16         return //不会校验用户名和密码
17     }
18     err = db.Ping()
19     if err != nil {
20         //用于校验用户名和密码
21         fmt.Printf("Open %s failed,err:%v\n", dsn, err)
22         return
23     }
24     fmt.Println("连接成功")
25 }
```

mysql的Open函数返回

*DB: 一个数据库的连接池

err: 如果连接成功返回nil, 否则返回连接错误类型。

DB是一个数据库（操作）句柄，[代表一个具有零到多个底层连接的连接池。](#)它可以安全的被多个go程同时使用。

sql包会自动创建和释放连接；它也会维护一个闲置连接的连接池。如果数据库具有单连接状态的概念，该状态只有在事务中被观察时才可信。一旦调用了BD.Begin，返回的Tx会绑定到单个连接。当调用事务Tx的Commit或Rollback后，该事务使用的连接会归还到DB的闲置连接池中。连接池的大小可以用SetMaxIdleConns方法控制。

初始化数据库

```

1  var database *sql.DB
2  func initDB() error {
3      //连接数据库
4      dsn := "root:wwy040609@tcp(127.0.0.1:3306)/mydb1"
5      //连接数据库
6      var err error
7      database, err = sql.Open("mysql", dsn)
8      if err != nil {
9          fmt.Printf("Open %s failed,err:%v\n", dsn, err)
10         return err
11     }
12     err = database.Ping()
13     if err != nil {
14         fmt.Printf("Open %s failed,err:%v\n", dsn, err)
15         return err
16     }
17     return nil
18 }

```

与主函数中直接连接数据库不同，如果和主函数中连接相同的话，声明的database作为连接池变量是局部变量，在主函数调用时是空指针。

连接池的连接数量

SetMaxOpenConns（控制连接查询的次数）

```
1 | func (db *DB) SetMaxOpenConns(n int)
```

SetMaxOpenConns 设置与数据库建立连接的最大数目。如果n大于0且小于最大闲置连接数，会将最大闲置连接数减小到匹配最大开启连接数的限制。如果n<=0，不会限制最大开启连接数，默认为0（无限制）。

SetMaxIdleConns（设置连接池中最大空闲连接数）

SetMaxIdleConns

```
1 | func (db *DB) SetMaxIdleConns(n int)
```

SetMaxIdleConns设置连接池中的最大闲置连接数。如果n大于最大开启连接数，则新的最大闲置连接数会减小到匹配最大开启连接数的限制。如果n<=0，不会保留闲置连接。

数据库操作

查询单条数据

```
1 func query() {
2     var ID int
3     fmt.Sprintf("%d", &ID)
4     sqlStr := "select * from stu where id=?"
5     rowObj := database.QueryRow(sqlStr, ID)
6     var stu student
7     rowObj.Scan(&stu.name, &stu.password, &stu.age, &stu.ID)
8     //Scan函数执行结束后会自动关闭数据库
9     fmt.Println(stu)
10 }
```

查询多条数据

```
1 func all_query() {
2     sqlStr := "select * from stu where id > 0"
3     rowObj, err := database.Query(sqlStr)
4     if err != nil {
5         fmt.Printf("Query %s failed,err:%v\n", sqlStr, err)
6         return
7     }
8     defer rowObj.Close()
9     for rowObj.Next() {
10        var stu student
11        err := rowObj.Scan(&stu.name, &stu.password, &stu.age, &stu.ID)
12        if err != nil {
13            fmt.Printf("scan failed,err:%v\n", err)
14            return
15        }
16        fmt.Println(stu)
17    }
18 }
```

插入数据和更新数据

```
1 func insert() {
2     var name string
3     var age int
4     var Id int
5     var password string
6     fmt.Println("请一次输入年龄, 密码, ID号, 姓名")
7     fmt.Scanf("%d", &age)
8     fmt.Scanf("%s", &password)
9     fmt.Scanf("%d", &Id)
10    fmt.Scanf("%s", &name)
11    //1.先写sql语句
12    sqlStr := "insert into stu(password,name,age,ID) values(?,?,?,?)"
13    //2.exec
14    ret, err := database.Exec(sqlStr, password, name, age, Id)
15    if err != nil {
16        fmt.Printf("insert failed,err:%v\n", err)
17        return
18    }
19    //如果是插入数据的操作, 能拿到插入数据的ID值
20    id, err := ret.LastInsertId()
21    if err != nil {
22        fmt.Printf("get id failed,err:%v\n", err)
23    }
24    fmt.Printf("insert success, id:%d\n", id)
25 }
```