Go 语言从入门到项目实战

第12章 数据库操作

12.1 MySQL

- MySQL是目前最流行的关系数据库，被广泛应用于各行业的Web应用，是较为传统的数据库软件之一；
- MySQL有两类版本可供选择——免费的社区版及收费的企业版。

12.1 MySQL

- Go 语言操作 MySQL 包：mysql
- 获取 mysql 包：
  - go env -w GOPROXY=https://goproxy.cn,direct
  - go get -u github.com/go-sql-driver/mysql

12.1 MySQL

- 连接示例：

```
db, err := sql.Open("mysql", "root:123456@/go_test")
if err != nil {
        panic(err)
}
db.SetConnMaxLifetime(time.Minute * 3)
db.SetMaxOpenConns(10)
db.SetMaxIdleConns(10)
fmt.Println("连接成功！！")
```

12.1 MySQL

- 执行 SQL 语句示例：

```
_, err = db.Exec("CREATE TABLE `test` (" +
        "`id` bigint(20) NOT NULL AUTO_INCREMENT," +
        "`name` varchar(45) DEFAULT ''," +
        "`age` int(11) NOT NULL DEFAULT '0'," +
        "`gender` tinyint(3) NOT NULL DEFAULT '0'," +
        "PRIMARY KEY (`id`)" +
        ") ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;")
if err != nil {
        panic(err)
}
```

12.1 MySQL

- 插入新数据示例：

```
stmtInsert, err := db.Prepare("INSERT INTO test SET name=?,age=?,gender=?")
if err != nil {
        fmt.Println(err)
        return
}
res, err := stmtInsert.Exec("三酷猫", "18", "0")
id, err := res.LastInsertId()
if err != nil {
        panic(err)
}
```

12.1 MySQL

- 删除新数据示例：

```
stmtDelete, err := db.Prepare("DELETE FROM test WHERE id=?")
if err != nil {
        fmt.Println(err)
        return
}
res, err := stmtDelete.Exec("1")
id, err := res.RowsAffected()
if err != nil {
        panic(err)
}
```

12.1 MySQL

- 修改数据示例：

```
stmtUpdate, err := db.Prepare("UPDATE test SET age=? WHERE id=2")
if err != nil {
        fmt.Println(err)
        return
}
res, err := stmtUpdate.Exec("24")
id, err := res.RowsAffected()
if err != nil {
        panic(err)
}
```

12.1 MySQL

- 单条查询数据示例：

```
type User struct {
        Name string `db:"name"`
        Id   int    `db:"id"`
        Age  int    `db:"age"`
        Sex  int    `db:"sex"`
}
var user Usererr = db.QueryRow("SELECT * FROM test WHERE
id=5").Scan(&user.Id, &user.Name, &user.Age, &user.Sex)
```

12.1 MySQL

- 结果集查询数据示例：

```
rows, e := db.Query("SELECT * FROM test WHERE age = 18")
if e != nil { panic(e) }
for rows.Next() {
        e := rows.Scan(&user.Id, &user.Name, &user.Age, &user.Sex)
        if e != nil { panic(e) }
        fmt.Println(user)
}
err = rows.Close()
if err != nil { panic(err) }
```

12.1 MySQL

- 事务操作示例：

```
tx, err := db.Begin()  // 开启事务
if err != nil { panic(err) }
i := 0
for i < 5000 {
        stmtTransaction, err := db.Prepare("INSERT INTO test SET name=?,age=?,gender=?")
        if err != nil { panic(err) }
        _, err = stmtTransaction.Exec("三酷猫", "18", "0")
        if err != nil { panic(err) }
        i++
}
err = tx.Commit()     // 提交事务
if err != nil { panic(err) }
```

12.2 Redis

- Redis是一个内存型数据库，最大的特点为数据的读/写过程运行在内存中，同时可以将数据持久化到硬盘上，因此具有十分优异的性能；
- Redis具有字符串（STRING）、列表（LIST）、集合（SET）、有序集合（ZSET）和散列表（HASH）5种数据类型。

12.2 Redis

- 示例：

```
options := redis.DialPassword(pwd)                                    //为Redis的拨号连接传入密码pwd
conn, err := redis.Dial(protocol, ip+":"+strconv.Itoa(port), options)      //对Redis进行拨号，获取Redis连接对象
defer func() {
    if err = conn.Close(); err != nil {                               //在使用完成后，关闭Redis连接
        log.Println("close error:", err)
    }
}()
if err != nil { return }
```

12.2 Redis

- 示例：

```
//切换到需要操作的Redis数据集，获取响应结果
reply, err := conn.Do("select", userInfoIndex)
if err != nil {
    log.Println("select error:", err)
    return
}
```

12.2 Redis

- 示例：

```
//为字符串型变量bar设置值为“3”
reply, err = redis.String(conn.Do("set", "bar", "3"))
if err != nil {
    log.Println("set error:", err)
    return
}
```

12.2 Redis

- 示例：

```
//获取bar的值为“3”
reply, err = redis.String(conn.Do("get", "bar"))
if err != nil {
    log.Println("get error:", err)
    return
}
```

12.2 Redis

- 示例：

```
//为列表变量list1从右端添加“0”这个元素
reply, err = redis.Int64(conn.Do("rpush", "list1", "777"))
if err != nil {
    log.Println("rpush error:", err)
    return
}
```

12.2 Redis

- 示例：

```
//获取list1的第1个元素
reply, err = redis.String(conn.Do("lindex", "list1", "0"))
if err != nil {
    log.Println("lrange error:", err)
    return
}
```

12.2 Redis

- 示例：

```
//为集合变量set1添加 "tom"
reply, err = redis.Int64(conn.Do("sadd", "set1", "tom"))
if err != nil {
    log.Println("sadd error:", err)
    return
}
```

12.2 Redis

- 示例：

```
//获取set1中所有元素对应的字节切片数组
replyBytes, err := redis.ByteSlices(conn.Do("smembers", "set1"))
if err != nil {
        log.Println("smembers error:", err)
        return
}
//将set1中所有元素对应的字节切片数组中的第1个元素转换为字符串
fmt.Println("smembers reply=", string(replyBytes[0]))
```

12.2 Redis

- 示例：

```
//为有序集合变量zset1添加分值为500的元素ele1
reply, err = redis.Int64(conn.Do("zadd", "zset1", "500", "ele1"))
if err != nil {
    log.Println("zadd error:", err)
    return
}
```

12.2 Redis

- 示例：

```
//获取zset1中的所有元素和分值
replyBytes, err = redis.ByteSlices(conn.Do("zrange", "zset1", "0","-1", "withscores"))
if err != nil {
        log.Println("zrange error:", err)
        return
}
//将zset1中所有元素对应的字节切片数组中的第1个元素转换为字符串
fmt.Println("zrange 操作的元素名称为：", string(replyBytes[0]),"\nzrange 操作的元素的分值为：
",string(replyBytes[1]))
```

12.2 Redis

- 示例：

```
//为散列表变量hash1添加name字段，值为Tom
reply, err = redis.Int64(conn.Do("hset", "hash1", "name","Tom"))
if err != nil {
    log.Println("hset error:", err)
    return
}
```

12.2 Redis

- 示例：

```
//获取hash1中的第1个字段和值
replyBytes, err = redis.ByteSlices(conn.Do("hgetall", "hash1"))
if err != nil {
    log.Println("hset error:", err)
    return
}
```

12.2 Redis

- 示例：

```
db.RedisConnect("tcp", "127.0.0.1", "gopher2020", 6379, 0)
```