本小节将展示如何使用pg驱动或GORM框架进行编程交互。用户需要事先安装、配置CockroachDB和Golang运行环境。

使用pg驱动

Step 1: 安装pg驱动

go get -u github.com/lib/pq

Step 2: 启动节点

以非安全模式：

cockroach start --insecure --store=hello-1 --host=localhost

Step 3: 创建用户

执行[cockroach user](http://doc.cockroachchina.baidu.com/#deploy/access-management/manage-users/)命令创建maxroach用户：

cockroach user set maxroach --insecure

Step 4: 创建数据库并授权

以root用户启动[内置的SQL客户端](http://doc.cockroachchina.baidu.com/#quick-start/learn-cockroachdb-sql/use-the-built-in-sql-client/)，创建bank数据库并授权maxroach用户。

cockroach sql --insecure -e 'CREATE DATABASE bank'

cockroach sql --insecure -e 'GRANT ALL ON DATABASE bank TO maxroach'

Step 5: 编写执行Go代码

基本操作

建表、插值、查询，Go代码：

**package** main

**import** (

"database/sql"

"fmt"

"log"

\_ "github.com/lib/pq"

)

**func** main() {

*// Connect to the "bank" database.*

*//非安全模式*

db, err := sql.Open("postgres", "postgresql://maxroach@localhost:26257/bank?sslmode=disable")

*//安全模式*

*//db, err := sql.Open("postgres", "postgresql://maxroach@localhost:26257/bank?ssl=true&sslmode=require&sslrootcert=certs/ca.crt&sslkey=certs/client.maxroach.key&sslcert=certs/client.maxroach.crt")*

**if** err != nil {

log.Fatal("error connecting to the database: ", err)

}

*// Create the "accounts" table.*

**if** \_, err := db.Exec(

"CREATE TABLE IF NOT EXISTS accounts (id INT PRIMARY KEY, balance INT)"); err != nil {

log.Fatal(err)

}

*// Insert two rows into the "accounts" table.*

**if** \_, err := db.Exec(

"INSERT INTO accounts (id, balance) VALUES (1, 1000), (2, 250)"); err != nil {

log.Fatal(err)

}

*// Print out the balances.*

rows, err := db.Query("SELECT id, balance FROM accounts")

**if** err != nil {

log.Fatal(err)

}

**defer** rows.Close()

fmt.Println("Initial balances:")

**for** rows.Next() {

**var** id, balance int

**if** err := rows.Scan(&id, &balance); err != nil {

log.Fatal(err)

}

fmt.Printf("%d %d\n", id, balance)

}

}

执行：

go run basic-sample.go

输出：

Initial balances:

1 1000

2 250

创建事务（带重试逻辑）

Go代码：

**package** main

**import** (

"context"

"database/sql"

"fmt"

"log"

"github.com/cockroachdb/cockroach-go/crdb"

)

**func** transferFunds(tx \*sql.Tx, from int, to int, amount int) error {

*// Read the balance.*

**var** fromBalance int

**if** err := tx.QueryRow(

"SELECT balance FROM accounts WHERE id = $1", from).Scan(&fromBalance); err != nil {

**return** err

}

**if** fromBalance < amount {

**return** fmt.Errorf("insufficient funds")

}

*// Perform the transfer.*

**if** \_, err := tx.Exec(

"UPDATE accounts SET balance = balance - $1 WHERE id = $2", amount, from); err != nil {

**return** err

}

**if** \_, err := tx.Exec(

"UPDATE accounts SET balance = balance + $1 WHERE id = $2", amount, to); err != nil {

**return** err

}

**return** nil

}

**func** main() {

*//非安全模式*

db, err := sql.Open("postgres", "postgresql://maxroach@localhost:26257/bank?sslmode=disable")

*//安全模式*

*//db, err := sql.Open("postgres", "postgresql://maxroach@localhost:26257/bank?ssl=true&sslmode=require&sslrootcert=certs/ca.crt&sslkey=certs/client.maxroach.key&sslcert=certs/client.maxroach.crt")*

**if** err != nil {

log.Fatal("error connecting to the database: ", err)

}

*// Run a transfer in a transaction.*

err = crdb.ExecuteTx(context.Background(), db, nil, **func**(tx \*sql.Tx) error {

**return** transferFunds(tx, 1 */\* from acct# \*/*, 2 */\* to acct# \*/*, 100 */\* amount \*/*)

})

**if** err == nil {

fmt.Println("Success")

} **else** {

log.Fatal("error: ", err)

}

}

在默认的隔离级别SERIALIZABLE下，因读写冲突导致事务执行失败，需要用户主动地重新提交事务。用户也实现通用的retry函数在事务内部重新执行事务。

在Golang环境下，CockroachDB的retry函数位于CockroachDB Go客户端的crdb包中，用户需要将相关库克隆到$GOPATH当中：

mkdir -p $GOPATH/src/github.com/cockroachdb

cd $GOPATH/src/github.com/cockroachdb

git clone git@github.com:cockroachdb/cockroach-go.git

执行：

go run txn-sample.go

输出：

**Success**

执行查询：

cockroach sql *--insecure -e 'SELECT id, balance FROM accounts' --database=bank*

+*----+---------+*

| id | balance |

+*----+---------+*

| 1 | 900 |

| 2 | 350 |

+*----+---------+*

(2 rows)

使用GORM框架

**NOTE:** 更多细节可查看[examples-orms](https://github.com/cockroachdb/examples-orms)项目。

Step 1: 安装GORM

go get -u github.com/lib/pq *# dependency*

go get -u github.com/jinzhu/gorm

Step 2: 启动节点

以非安全模式启动节点：

cockroach start --insecure --store=hello-1 --host=localhost

Step 3: 创建用户

执行[cockroach user](http://doc.cockroachchina.baidu.com/#deploy/access-management/manage-users/)命令创建maxroach用户：

cockroach user set maxroach --insecure

Step 4: 创建数据库并授权

以root用户启动[内置的SQL客户端](http://doc.cockroachchina.baidu.com/#quick-start/learn-cockroachdb-sql/use-the-built-in-sql-client/)，创建bank数据库并授权maxroach用户。

cockroach sql --insecure -e 'CREATE DATABASE bank'

cockroach sql --insecure -e 'GRANT ALL ON DATABASE bank TO maxroach'

Step 5: 编写执行Go代码

下述Go代码中，代码db.AutoMigrate(&Account{}将根据Account模型创建accounts表：

**package** main

**import** (

"fmt"

"log"

*// Import GORM-related packages.*

"github.com/jinzhu/gorm"

\_ "github.com/jinzhu/gorm/dialects/postgres"

)

*// Account is our model, which corresponds to the "accounts" database table.*

**type** Account **struct** {

ID int `gorm:"primary\_key"`

Balance int

}

**func** main() {

*// Connect to the "bank" database as the "maxroach" user.*

*//非安全模式*

**const** addr = "postgresql://maxroach@localhost:26257/bank?sslmode=disable"

*//安全模式*

*//const addr = "postgresql://maxroach@localhost:26257/bank?ssl=true&sslmode=require&sslrootcert=certs/ca.crt&sslkey=certs/client.maxroach.key&sslcert=certs/client.maxroach.crt"*

db, err := gorm.Open("postgres", addr)

**if** err != nil {

log.Fatal(err)

}

**defer** db.Close()

*// Automatically create the "accounts" table based on the Account model.*

db.AutoMigrate(&Account{})

*// Insert two rows into the "accounts" table.*

db.Create(&Account{ID: 1, Balance: 1000})

db.Create(&Account{ID: 2, Balance: 250})

*// Print out the balances.*

**var** accounts []Account

db.Find(&accounts)

fmt.Println("Initial balances:")

**for** \_, account := **range** accounts {

fmt.Printf("%d %d\n", account.ID, account.Balance)

}

}

执行：

go run gorm-basic-sample.go

输出：

Initial balances:

1 1000

2 250

执行查询：

cockroach sql *--insecure -e 'SELECT id, balance FROM accounts' --database=bank*

+*----+---------+*

| id | balance |

+*----+---------+*

| 1 | 1000 |

| 2 | 250 |

+*----+---------+*

(2 rows)