ORM

goframeORM

1. Begin*gdb.TX
2. Transaction

Transaction

https://godoc.org/github.com/gogf/gf/database/gdb#TX

Begin/Commit/Rollbackdb.Begin*gdb.Txtx.Committx.Rollback

Commit/RollbackdefergoroutineTransaction

1.

```
if tx, err := db.Begin(); err == nil {
    fmt.Println("")
}
```

db API

2.

```
if tx, err := db.Begin(); err == nil {
    r, err := tx.Save("user", g.Map{
        "id"   :  1,
        "name" : "john",
    })
    if err != nil {
        tx.Rollback()
    }
    fmt.Println(r)
}
```

3.

```
if tx, err := db.Begin(); err == nil {
    r, err := tx.Save("user", g.Map{
        "id"   :  1,
        "name" : "john",
    })
    if err == nil {
       tx.Commit()
    }
    fmt.Println(r)
}
```

4.

tx.Modeldb.Model

Content Menu

```
if tx, err := db.Begin(); err == nil {
    r, err := tx.Table("user").Data(g.Map{"id":1, "name": "john_1"}).Save()
    if err == nil {
       tx.Commit()
    }
    fmt.Println(r)
}
```

ORM()

Transaction

/ORMTransaction

```
func (db DB) Transaction(ctx context.Context, f func(ctx context.Context,
tx *TX) error) (err error)
```

errornilCommitRollbackcontext.Contextgoframe v1.16

panic

```
db.Transaction(context.TODO(), func(ctx context.Context, tx *gdb.TX) error
{
        // user
        result, err := tx.Ctx(ctx).Insert("user", g.Map{
                "passport": "john",
                "password": "12345678",
                "nickname": "JohnGuo",
        })
        if err != nil {
                return err
        }
        // user_detail
        id, err := result.LastInsertId()
        if err != nil {
                return err
        }
        _, err = tx.Ctx(ctx).Insert("user_detail", g.Map{
                "uid":       id,
                "site":      "https://johng.cn",
                "true_name": "GuoQiang",
        })
        if err != nil {
                return err
        }
        return nil
})
```

Transaction

goframev1.16ORMTransaction Save Point

```
// Begin starts a nested transaction procedure.
func (tx *TX) Begin() error

// Commit commits current transaction.
// Note that it releases previous saved transaction point if it's in a
nested transaction procedure,
// or else it commits the hole transaction.
func (tx *TX) Commit() error

// Rollback aborts current transaction.
// Note that it aborts current transaction if it's in a nested transaction
procedure,
// or else it aborts the hole transaction.
func (tx *TX) Rollback() error

// SavePoint performs `SAVEPOINT xxx` SQL statement that saves transaction
at current point.
// The parameter `point` specifies the point name that will be saved to
server.
func (tx *TX) SavePoint(point string) error

// RollbackTo performs `ROLLBACK TO SAVEPOINT xxx` SQL statement that
rollbacks to specified saved transaction.
// The parameter `point` specifies the point name that was saved
previously.
func (tx *TX) RollbackTo(point string) error

// Transaction wraps the transaction logic using function `f`.
// It rollbacks the transaction and returns the error from function `f` if
// it returns non-nil error. It commits the transaction and returns nil if
// function `f` returns nil.
//
// Note that, you should not Commit or Rollback the transaction in
function `f`
// as it is automatically handled by this function.
func (tx *TX) Transaction(ctx context.Context, f func(ctx context.Context,
tx *TX) error) (err error)
```

Transaction

1.

SQLidname

```
CREATE TABLE `user` (
  `id` int(10) unsigned NOT NULL COMMENT 'ID',
  `name` varchar(45) NOT NULL COMMENT '',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

```
tx, err := db.Begin()
if err != nil {
        panic(err)
}
if err = tx.Begin(); err != nil {
        panic(err)
}
_, err = tx.Model(table).Data(g.Map{"id": 1, "name": "john"}).Insert()
if err = tx.Rollback(); err != nil {
        panic(err)
}
_, err = tx.Model(table).Data(g.Map{"id": 2, "name": "smith"}).Insert()
if err = tx.Commit(); err != nil {
        panic(err)
}
```

db.Begintx.Begin

db.Begintx.Begindb.Begintxtxtx.BeginSavePointtransactionNNSAVEPOINT
`transaction1`20

goframeORMSQL

```
2021-05-22 21:12:10.776 [DEBU] [  4 ms] [default] [1] BEGIN
2021-05-22 21:12:10.776 [DEBU] [  0 ms] [default] [1] SAVEPOINT
`transaction0`
2021-05-22 21:12:10.789 [DEBU] [ 13 ms] [default] [1] SHOW FULL COLUMNS
FROM `user`
2021-05-22 21:12:10.790 [DEBU] [  1 ms] [default] [1] INSERT INTO `user`
(`id`,`name`) VALUES(1,'john')
2021-05-22 21:12:10.791 [DEBU] [  1 ms] [default] [1] ROLLBACK TO
SAVEPOINT `transaction0`
2021-05-22 21:12:10.791 [DEBU] [  0 ms] [default] [1] INSERT INTO `user`
(`id`,`name`) VALUES(2,'smith')
2021-05-22 21:12:10.792 [DEBU] [  1 ms] [default] [1] COMMIT
```

[1]ORMIDIDID

```
mysql> select * from `user`;
+----+-------+
| id | name  |
+----+-------+
|  2 | smith |
+----+-------+
1 row in set (0.00 sec)
```

2.

Transaction

```
db.Transaction(ctx, func(ctx context.Context, tx *gdb.TX) error {
        // Nested transaction 1.
        if err := tx.Transaction(ctx, func(ctx context.Context, tx *gdb.
TX) error {
                _, err := tx.Model(table).Ctx(ctx).Data(g.Map{"id": 1,
"name": "john"}).Insert()
                return err
        }); err != nil {
                return err
        }
        // Nested transaction 2, panic.
        if err := tx.Transaction(ctx, func(ctx context.Context, tx *gdb.
TX) error {
                _, err := tx.Model(table).Ctx(ctx).Data(g.Map{"id": 2,
"name": "smith"}).Insert()
                // Create a panic that can make this transaction rollback
automatically.
                panic("error")
                return err
        }); err != nil {
                return err
        }
        return nil
})
```

txdbdaotxctx

```
db.Transaction(ctx, func(ctx context.Context, tx *gdb.TX) error {
        // Nested transaction 1.
        if err := db.Transaction(ctx, func(ctx context.Context, tx *gdb.
TX) error {
                _, err := db.Model(table).Ctx(ctx).Data(g.Map{"id": 1,
"name": "john"}).Insert()
                return err
        }); err != nil {
                return err
        }
        // Nested transaction 2, panic.
        if err := db.Transaction(ctx, func(ctx context.Context, tx *gdb.
TX) error {
                _, err := db.Model(table).Ctx(ctx).Data(g.Map{"id": 2,
"name": "smith"}).Insert()
                // Create a panic that can make this transaction rollback
automatically.
                panic("error")
                return err
        }); err != nil {
                return err
        }
        return nil
})
```

SQL

```
2021-05-22 21:18:46.672 [DEBU] [  2 ms] [default] [1] BEGIN
2021-05-22 21:18:46.672 [DEBU] [  0 ms] [default] [1] SAVEPOINT
`transaction0`
2021-05-22 21:18:46.673 [DEBU] [  0 ms] [default] [1] SHOW FULL COLUMNS
FROM `user`
2021-05-22 21:18:46.674 [DEBU] [  0 ms] [default] [1] INSERT INTO `user`
(`id`,`name`) VALUES(1,'john')
2021-05-22 21:18:46.674 [DEBU] [  0 ms] [default] [1] RELEASE SAVEPOINT
`transaction0`
2021-05-22 21:18:46.675 [DEBU] [  1 ms] [default] [1] SAVEPOINT
`transaction0`
2021-05-22 21:18:46.675 [DEBU] [  0 ms] [default] [1] INSERT INTO `user`
(`name`,`id`) VALUES('smith',2)
2021-05-22 21:18:46.675 [DEBU] [  0 ms] [default] [1] ROLLBACK TO
SAVEPOINT `transaction0`
2021-05-22 21:18:46.676 [DEBU] [  1 ms] [default] [1] ROLLBACK
```

ctx

```
db.Transaction(ctx, func(ctx context.Context, tx *gdb.TX) error {
        // Nested transaction 1.
        if err := db.Transaction(ctx, func(ctx context.Context, tx
*gdb.TX) error {
                _, err := db.Model(table).Ctx(ctx).Data(g.Map
{"id": 1, "name": "john"}).Insert()
                return err
        }); err != nil {
                return err
        }
        // Nested transaction 2, panic.
        if err := db.Transaction(ctx, func(ctx context.Context, tx
*gdb.TX) error {
                _, err := db.Model(table).Data(g.Map{"id": 2,
"name": "smith"}).Insert()
                // Create a panic that can make this transaction
rollback automatically.
                panic("error")
                return err
        }); err != nil {
                return err
        }
        return nil
})
```

SQL

```
2021-05-22 21:29:38.841 [DEBU] [  3 ms] [default] [1] BEGIN
2021-05-22 21:29:38.842 [DEBU] [  1 ms] [default] [1] SAVEPOINT
`transaction0`
2021-05-22 21:29:38.843 [DEBU] [  1 ms] [default] [1] SHOW FULL
COLUMNS FROM `user`
2021-05-22 21:29:38.845 [DEBU] [  2 ms] [default] [1] INSERT INTO
`user`(`id`,`name`) VALUES(1,'john')
2021-05-22 21:29:38.845 [DEBU] [  0 ms] [default] [1] RELEASE
SAVEPOINT `transaction0`
2021-05-22 21:29:38.846 [DEBU] [  1 ms] [default] [1] SAVEPOINT
`transaction0`
2021-05-22 21:29:38.847 [DEBU] [  1 ms] [default] INSERT INTO
`user`(`id`,`name`) VALUES(2,'smith')
2021-05-22 21:29:38.848 [DEBU] [  0 ms] [default] [1] ROLLBACK TO
SAVEPOINT `transaction0`
2021-05-22 21:29:38.848 [DEBU] [  0 ms] [default] [1] ROLLBACK
```

INSERTID

3. SavePoint/RollbackTo

Transaction Save PointSavePointPoint

```
tx, err := db.Begin()
if err != nil {
        panic(err)
}
defer func() {
        if err := recover(); err != nil {
                _ = tx.Rollback()
        }
}()
if _, err = tx.Model(table).Data(g.Map{"id": 1, "name": "john"}).Insert();
err != nil {
        panic(err)
}
if err = tx.SavePoint("MyPoint"); err != nil {
        panic(err)
}
if _, err = tx.Model(table).Data(g.Map{"id": 2, "name": "smith"}).Insert();
err != nil {
        panic(err)
}
if _, err = tx.Model(table).Data(g.Map{"id": 3, "name": "green"}).Insert();
err != nil {
        panic(err)
}
if err = tx.RollbackTo("MyPoint"); err != nil {
        panic(err)
}
if err = tx.Commit(); err != nil {
        panic(err)
}
```

SQL

```
2021-05-22 21:38:51.992 [DEBU] [  3 ms] [default] [1] BEGIN
2021-05-22 21:38:52.002 [DEBU] [  9 ms] [default] [1] SHOW FULL COLUMNS
FROM `user`
2021-05-22 21:38:52.002 [DEBU] [  0 ms] [default] [1] INSERT INTO `user`
(`id`,`name`) VALUES(1,'john')
2021-05-22 21:38:52.003 [DEBU] [  1 ms] [default] [1] SAVEPOINT `MyPoint`
2021-05-22 21:38:52.004 [DEBU] [  1 ms] [default] [1] INSERT INTO `user`
(`id`,`name`) VALUES(2,'smith')
2021-05-22 21:38:52.005 [DEBU] [  1 ms] [default] [1] INSERT INTO `user`
(`id`,`name`) VALUES(3,'green')
2021-05-22 21:38:52.006 [DEBU] [  0 ms] [default] [1] ROLLBACK TO
SAVEPOINT `MyPoint`
2021-05-22 21:38:52.006 [DEBU] [  0 ms] [default] [1] COMMIT
```

```
mysql> select * from `user`;
+----+------+
| id | name |
+----+------+
|  1 | john |
+----+------+
1 row in set (0.00 sec)
```

InsertSavePointMyPointRollbackToInsert

(user)(user_detail)dao

goframeapi-service-dao

api

```
// HTTP
func (*userApi) Signup(r *ghttp.Request) {
        // ....
        service.User.Signup(r.Context(), userServiceSignupReq)
        // ...
}
```

HTTPContext

service

```
//
func (*userService) Signup(ctx context.Context, r *model.
UserServiceSignupReq) {
        // ....
        dao.User.Transaction(ctx, func(ctx context.Context, tx *gdb.TX)
error {
                err := dao.User.Ctx(ctx).Save(r.UserInfo)
                if err != nil {
                        return err
                }
                err := dao.UserDetail.Ctx(ctx).Save(r.UserDetail)
                if err != nil {
                        return err
                }
                return nil
        })
        // ...
}
```

useruser_detailCtxservicectx

```
func (*userService) Signup(ctx context.Context, r *model.
UserServiceSignupReq) {
        // ....
        dao.User.Transaction(ctx, func(ctx context.Context, tx *gdb.TX)
error {
                err := dao.User.Ctx(ctx).Save(r.UserInfo)
                if err != nil {
                        return err
                }
                err := dao.UserDetail.Ctx(ctx).Save(r.UserDetail)
                if err != nil {
                        return err
                }
                err := service.XXXA.Call(ctx, ...)
                if err != nil {
                        return err
                }
                err := service.XXXB.Call(ctx, ...)
                if err != nil {
                        return err
                }
                err := service.XXXB.Call(ctx, ...)
                if err != nil {
                        return err
                }
        // ...
                return nil
        })
        // ...
}
```

dao

daogoframe cli