ORM-Where/WhereOr/WhereNot

ORM

```
func (m *Model) Where(where interface{}, args...interface{}) *Model
func (m *Model) Wheref(format string, args ...interface{}) *Model
func (m *Model) WherePri(where interface{}, args ...interface{}) *Model
func (m *Model) WhereBetween(column string, min, max interface{}) *Model
func (m *Model) WhereLike(column string, like interface{}) *Model
func (m *Model) WhereIn(column string, in interface{}) *Model
func (m *Model) WhereNull(columns ...string) *Model
func (m *Model) WhereLT(column string, value interface{}) *Model
func (m *Model) WhereLTE(column string, value interface{}) *Model
func (m *Model) WhereGT(column string, value interface{}) *Model
func (m *Model) WhereGTE(column string, value interface{}) *Model

func (m *Model) WhereNotBetween(column string, min, max interface{}) *Model
func (m *Model) WhereNotLike(column string, like interface{}) *Model
func (m *Model) WhereNotIn(column string, in interface{}) *Model
func (m *Model) WhereNotNull(columns ...string) *Model

func (m *Model) WhereOr(where interface{}, args ...interface{}) *Model
func (m *Model) WhereOrBetween(column string, min, max interface{}) *Model
func (m *Model) WhereOrLike(column string, like interface{}) *Model
func (m *Model) WhereOrIn(column string, in interface{}) *Model
func (m *Model) WhereOrNull(columns ...string) *Model
func (m *Model) WhereOrLT(column string, value interface{}) *Model
func (m *Model) WhereOrLTE(column string, value interface{}) *Model
func (m *Model) WhereOrGT(column string, value interface{}) *Model
func (m *Model) WhereOrGTE(column string, value interface{}) *Model

func (m *Model) WhereOrNotBetween(column string, min, max interface{})
*Model
func (m *Model) WhereOrNotLike(column string, like interface{}) *Model
func (m *Model) WhereOrNotIn(column string, in interface{}) *Model
func (m *Model) WhereOrNotNull(columns ...string) *Model
```

Content Menu

- Where/WhereOr
  -
  -
- Wheref
- WherePri
- WhereBuilder
  -
  -
- 0=1

Where/WhereOr

string/map/slice/struct/*struct

Where?map/struct

WhereAnd gdb()

```
// WHERE `uid`=1
Where("uid=1")
Where("uid", 1)
Where("uid=?", 1)
Where(g.Map{"uid" : 1})
// WHERE `uid` <= 1000 AND `age` >= 18
Where(g.Map{
    "uid <=" : 1000,
    "age >=" : 18,
})

// WHERE (`uid` <= 1000) AND (`age` >= 18)
Where("uid <=?", 1000).Where("age >=?", 18)

// WHERE `level`=1 OR `money`>=1000000
Where("level=? OR money >=?", 1, 1000000)

// WHERE (`level`=1) OR (`money`>=1000000)
Where("level", 1).WhereOr("money >=", 1000000)

// WHERE `uid` IN(1,2,3)
Where("uid IN(?)", g.Slice{1,2,3})
```

structormtagstruct

```
type Condition struct{
    Sex int `orm:"sex"`
    Age int `orm:"age"`
}
Where(Condition{1, 18})
// WHERE `sex`=1 AND `age`=18
```

Where + string

```
// Limit
// SELECT * FROM user WHERE uid>1 LIMIT 0,10
g.Model("user").Where("uid > ?", 1).Limit(0, 10).All()

// Fields
// Fields*
// SELECT uid,name FROM user WHERE uid>1 LIMIT 0,10
g.Model("user").Fields("uid,name").Where("uid > ?", 1).Limit(0, 10).All()

// Where
// SELECT * FROM user WHERE uid=1 LIMIT 1
g.Model("user").Where("uid=1").One()
g.Model("user").Where("uid", 1).One()
g.Model("user").Where("uid=?", 1).One()

// SELECT * FROM user WHERE (uid=1) AND (name='john') LIMIT 1
g.Model("user").Where("uid", 1).Where("name", "john").One()
g.Model("user").Where("uid=?", 1).Where("name=?", "john").One()

// SELECT * FROM user WHERE (uid=1) OR (name='john') LIMIT 1
g.Model("user").Where("uid=?", 1).WhereOr("name=?", "john").One()
```

Where + sliceslice

```
// SELECT * FROM user WHERE age>18 AND name like '%john%'
g.Model("user").Where("age>? AND name like ?", g.Slice{18, "%john%"}).All()

// SELECT * FROM user WHERE status=1
g.Model("user").Where("status=?", g.Slice{1}).All()
```

Where + mapmap

```
// SELECT * FROM user WHERE uid=1 AND name='john' LIMIT 1
g.Model("user").Where(g.Map{"uid" : 1, "name" : "john"}).One()

// SELECT * FROM user WHERE uid=1 AND age>18 LIMIT 1
g.Model("user").Where(g.Map{"uid" : 1, "age>" : 18}).One()
```

Where + struct/*structstruct orm/json

```
type User struct {
    Id       int    `json:"uid"`
    UserName string `orm:"name"`
}
// SELECT * FROM user WHERE uid =1 AND name='john' LIMIT 1
g.Model("user").Where(User{ Id : 1, UserName : "john"}).One()

// SELECT * FROM user WHERE uid =1 LIMIT 1
g.Model("user").Where(&User{ Id : 1}).One()
```

```
condition := g.Map{
    "title like ?"         : "%%",
    "online"               : 1,
    "hits between ? and ?" : g.Slice{1, 10},
    "exp > 0"              : nil,
    "category"             : g.Slice{100, 200},
}
// SELECT * FROM article WHERE title like '%%' AND online=1 AND hits
between 1 and 10 AND exp > 0 AND category IN(100,200)
g.Model("article").Where(condition).All()
```

Wheref

fmt.SprintfWhere+fmt.SprintfWheref

```
// WHERE score > 100 and status in('succeeded','completed')
Wheref(`score > ? and status in (?)`, 100, g.Slice{"succeeded",
"completed"})
```

WherePri

WherePriWhereuseruidWhereWherePri

```
// WHERE `uid`=1
Where("uid", 1)
WherePri(1)

// WHERE `uid` IN(1,2,3)
Where("uid", g.Slice{1,2,3})
WherePri(g.Slice{1,2,3})
```

WherePrislice

WhereBuilder

WhereBuilderWhere

ModelBuilderWhereBuilder

```
// Builder creates and returns a WhereBuilder.
func (m *Model) Builder() *WhereBuilder
```

```
// SELECT * FROM `user` WHERE `id`=1 AND `address`="USA" AND (`status`="
active" OR `status`="pending")
m := g.Model("user")
all, err := m.Where("id", 1).Where("address", "USA").Where(
        m.Builder().Where("status", "active").WhereOr("status", "pending"),
).All()
```

0=1

SQL1

```
m := g.Model("auth")
m.Where("status", g.Slice{"permitted", "inherited"}).Where("uid", 1).All()
// SELECT * FROM `auth` WHERE (`status` IN('permitted','inherited')) AND
(`uid`=1)
```

SQL2

```
m := g.Model("auth")
m.Where("status", g.Slice{}).Where("uid", 1).All()
// SELECT * FROM `auth` WHERE (0=1) AND (`uid`=1)
```

SQL0=1

ORMSQLSQLOmitEmpty/OmitEmptyWhere

```
m := g.Model("auth")
m.Where("status", g.Slice{}).Where("uid", 1).OmitEmpty().All()
// SELECT * FROM `auth` WHERE `uid`=1
```