

--With

goframegoframegoframe ORM BelongsTo , HasOne , HasMany , ManyToMany ScanListWith --
ScanList

ScanListWithWithScanList

aries

 Withgoframe v1.15.7

WithScanList

1

```
#
CREATE TABLE `user` (
  id int(10) unsigned NOT NULL AUTO_INCREMENT,
  name varchar(45) NOT NULL,
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
CREATE TABLE `user_detail` (
  uid int(10) unsigned NOT NULL AUTO_INCREMENT,
  address varchar(45) NOT NULL,
  PRIMARY KEY (uid)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
CREATE TABLE `user_scores` (
  id int(10) unsigned NOT NULL AUTO_INCREMENT,
  uid int(10) unsigned NOT NULL,
  score int(10) unsigned NOT NULL,
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

2

- 1. 1:1
- 2. 1:N
- 3. N:N1:N1:N

Golang

Content Menu

-
- - 1
 - 2
 - 3
 - 4
 - 5
 - 6
- - 1gmeta
 - 2
 - 3With/WithAll
 - 1
 - 2
 - 3
 - 4
- - 1
 - 2
-
- - 1
 - 2
 - 3meta
-

```
//
type UserDetails struct {
    gmeta.Meta `orm:"table:user_detail"`
    Uid        int    `json:"uid"`
    Address    string `json:"address"`
}
//
type UserScores struct {
    gmeta.Meta `orm:"table:user_scores"`
    Id         int    `json:"id"`
    Uid        int    `json:"uid"`
    Score      int    `json:"score"`
}
//
type User struct {
    gmeta.Meta `orm:"table:user"`
    Id         int    `json:"id"`
    Name       string `json:"name"`
    UserDetails *UserDetails `orm:"with:uid=id"`
    UserScores []*UserScores `orm:"with:uid=id"`
}
```

3

5

- id1-5name1name_5
- 5addressaddress_1address_5
- 51-5

```
g.DB().Transaction(ctx, func(ctx context.Context, tx gdb.TX) error {
    for i := 1; i <= 5; i++ {
        // User.
        user := User{
            Name: fmt.Sprintf(`name_%d`, i),
        }
        lastInsertId, err := g.Model(user).Data(user).OmitEmpty().
InsertAndGetId()
        if err != nil {
            return err
        }
        // Detail.
        userDetails := UserDetails{
            Uid:      int(lastInsertId),
            Address:  fmt.Sprintf(`address_%d`, lastInsertId),
        }
        _, err = g.Model(userDetails).Data(userDetails).OmitEmpty().
Insert()
        if err != nil {
            return err
        }
        // Scores.
        for j := 1; j <= 5; j++ {
            userScore := UserScores{
                Uid:      int(lastInsertId),
                Score: j,
            }
            _, err = g.Model(userScore).Data(userScore).
OmitEmpty().Insert()
            if err != nil {
                return err
            }
        }
    }
    return nil
})
```

```
mysql> show tables;
+-----+
| Tables_in_test |
+-----+
| user            |
| user_detail     |
| user_score      |
+-----+
3 rows in set (0.01 sec)
```

```
mysql> select * from `user`;
+----+-----+
| id | name  |
+----+-----+
| 1  | name_1 |
| 2  | name_2 |
| 3  | name_3 |
| 4  | name_4 |
| 5  | name_5 |
+----+-----+
5 rows in set (0.01 sec)
```

```
mysql> select * from `user_detail`;
+----+-----+
| uid | address |
+----+-----+
| 1   | address_1 |
| 2   | address_2 |
| 3   | address_3 |
| 4   | address_4 |
| 5   | address_5 |
+----+-----+
5 rows in set (0.00 sec)
```

```
mysql> select * from `user_score`;
+----+-----+-----+
| id | uid | score |
+----+-----+-----+
| 1  | 1   | 1     |
| 2  | 1   | 2     |
| 3  | 1   | 3     |
| 4  | 1   | 4     |
| 5  | 1   | 5     |
| 6  | 2   | 1     |
| 7  | 2   | 2     |
| 8  | 2   | 3     |
| 9  | 2   | 4     |
| 10 | 2   | 5     |
| 11 | 3   | 1     |
| 12 | 3   | 2     |
| 13 | 3   | 3     |
| 14 | 3   | 4     |
| 15 | 3   | 5     |
| 16 | 4   | 1     |
| 17 | 4   | 2     |
| 18 | 4   | 3     |
| 19 | 4   | 4     |
| 20 | 4   | 5     |
| 21 | 5   | 1     |
| 22 | 5   | 2     |
| 23 | 5   | 3     |
| 24 | 5   | 4     |
| 25 | 5   | 5     |
+----+-----+-----+
25 rows in set (0.00 sec)
```

4

With

```
var user *User
g.Model(tableUser).WithAll().Where("id", 3).Scan(&user)
```

ID3SQL

```
2021-05-02 22:29:52.634 [DEBU] [ 2 ms] [default] SHOW FULL COLUMNS FROM
`user`
2021-05-02 22:29:52.635 [DEBU] [ 1 ms] [default] SELECT * FROM `user`
WHERE `id`=3 LIMIT 1
2021-05-02 22:29:52.636 [DEBU] [ 1 ms] [default] SHOW FULL COLUMNS FROM
`user_detail`
2021-05-02 22:29:52.637 [DEBU] [ 1 ms] [default] SELECT `uid`,`address`
FROM `user_detail` WHERE `uid`=3 LIMIT 1
2021-05-02 22:29:52.643 [DEBU] [ 6 ms] [default] SHOW FULL COLUMNS FROM
`user_score`
2021-05-02 22:29:52.644 [DEBU] [ 0 ms] [default] SELECT `id`,`uid`,
`score` FROM `user_score` WHERE `uid`=3
```

g.Dump(user)

```
{
  Id:      3,
  Name:    "name_3",
  UserDetail: {
    Uid:    3,
    Address: "address_3",
  },
  UserScores: [
    {
      Id:    11,
      Uid:    3,
      Score: 1,
    },
    {
      Id:    12,
      Uid:    3,
      Score: 2,
    },
    {
      Id:    13,
      Uid:    3,
      Score: 3,
    },
    {
      Id:    14,
      Uid:    3,
      Score: 4,
    },
    {
      Id:    15,
      Uid:    3,
      Score: 5,
    },
  ],
}
```

5

With

```
var users []*User
g.Model(users).With(UserDetail{}).Where("id>?", 3).Scan(&users)
```

g.Dump(users)

```
[
  {
    Id:      4,
    Name:    "name_4",
    UserDetail: {
      Uid:    4,
      Address: "address_4",
    },
    UserScores: [],
  },
  {
    Id:      5,
    Name:    "name_5",
    UserDetail: {
      Uid:    5,
      Address: "address_5",
    },
    UserScores: [],
  },
]
```

6

With

```
type User struct {
    gmeta.Meta `orm:"table:user"`
    Id         int         `json:"id"`
    Name       string        `json:"name"`
    UserDetail *UserDetail `orm:"with:uid=id, where:uid > 3"`
    UserScores []*UserScores `orm:"with:uid=id, where:score>1 and
score<5, order:score desc"`
}
```

ormwhereorder

gmetaWithAllormwithModelstruct

1gmeta

embedgmeta.Meta

```
type UserDetail struct {
    gmeta.Meta `orm:"table:user_detail"`
    Uid        int        `json:"uid"`
    Address    string    `json:"address"`
}
```

GoFramegmetagmetagmeta.Meta`orm:"table:user\_detail"`-gmeta

gmeta.Meta

2

```

type User struct {
    gmeta.Meta `orm:"table:user"`
    Id         int         `json:"id"`
    Name       string       `json:"name"`
    UserDetail *UserDetail `orm:"with:uid=id"`
    UserScores []*UserScore `orm:"with:uid=id"`
}

```

ormormwithwith

```
with:=
```

```

with:UID=ID
with:Uid=Id
with:U_ID=id

```

```
with:uid
```

UserDetailuser_detailUserScoresuser_scoreUseruseruiduserid

3With/WithAll

1

ormwithORMWithWith/WithAll

- With
- WithAllwith

WithAllUserormwithWithWithWithAll

```

var user *User
g.Model(tableUser).With(UserDetail{}, UserScore{}).Where("id", 3).Scan(
&user)

```

```

var user *User
g.Model(tableUser).With(User{}.UserDetail, User{}.UserScore).Where("id",
3).Scan(&user)

```

2

With

```

var user *User
g.Model(tableUser).With(UserDetail{}).Where("id", 3).Scan(&user)

```

```
var user *User
g.Model(tableUser).With(User{}.UserDetail).Where("id", 3).Scan(&user)
```

g.Dump(user)

```
{
  "id": 3,
  "name": "name_3",
  "UserDetail": {
    "uid": 3,
    "address": "address_3"
  },
  "UserScores": null
}
```

3

```
var user *User
g.Model(tableUser).With(UserScore{}).Where("id", 3).Scan(&user)
```

```
var user *User
g.Model(tableUser).With(User{}.UserScore).Where("id", 3).Scan(&user)
```

g.Dump(user)

```
{
  "id": 3,
  "name": "name_3",
  "UserDetail": null,
  "UserScores": [
    {
      "id": 11,
      "uid": 3,
      "score": 1
    },
    {
      "id": 12,
      "uid": 3,
      "score": 2
    },
    {
      "id": 13,
      "uid": 3,
      "score": 3
    },
    {
      "id": 14,
      "uid": 3,
      "score": 4
    },
    {
      "id": 15,
      "uid": 3,
      "score": 5
    }
  ]
}
```

4

```
var user *User
g.Model(tableUser).Where("id", 3).Scan(&user)
```

g.Dump(user)

```
{
  "id": 3,
  "name": "name_3",
  "UserDetail": null,
  "UserScores": null
}
```

1

SQLSELECT *With

Withmodel

ContentCMS

```

type Content struct {
    Id      uint      `orm:"id,primary"      json:"id"
    // ID
    Key      string     `orm:"key"             json:"key"
    //
    Type      string     `orm:"type"            json:"type"
    // : topic, ask, article
    CategoryId uint     `orm:"category_id"     json:"category_id"
    // ID
    UserId    uint     `orm:"user_id"         json:"user_id"
    // ID
    Title     string   `orm:"title"           json:"title"
    //
    Content    string   `orm:"content"         json:"content"
    //
    Sort       uint     `orm:"sort"            json:"sort"
    //
    Brief      string   `orm:"brief"           json:"brief"
    //
    Thumb      string   `orm:"thumb"           json:"thumb"
    //
    Tags       string   `orm:"tags"            json:"tags"
    // JSON
    Referer    string   `orm:"referer"         json:"referer"
    // github/gitee
    Status     uint     `orm:"status"          json:"status"
    // 0: , 1:
    ReplyCount uint     `orm:"reply_count"     json:"reply_count"
    //
    ViewCount  uint     `orm:"view_count"      json:"view_count"
    //
    ZanCount   uint     `orm:"zan_count"       json:"zan_count"
    //
    CaiCount   uint     `orm:"cai_count"       json:"cai_count"
    //
    CreatedAt   *gtime.Time `orm:"created_at"      json:"created_at"
    //
    UpdatedAt   *gtime.Time `orm:"updated_at"      json:"updated_at"
    //
}

```

Content

```

type ContentListItem struct {
    Id      uint      `json:"id"` // ID
    CategoryId uint    `json:"category_id"` // ID
    UserId   uint    `json:"user_id"` // ID
    Title    string   `json:"title"` //
    CreatedAt *gtime.Time `json:"created_at"` //
    UpdatedAt *gtime.Time `json:"updated_at"` //
}

```

2

withSQLwith

withWith

```

//
type UserDetails struct {
    gmeta.Meta `orm:"table:user_detail"`
    Uid        int    `json:"uid"`
    Address    string `json:"address"`
}

// -
type UserScoresRequired struct {
    gmeta.Meta `orm:"table:user_scores"`
    Id         int    `json:"id"`
    Uid        int    `json:"uid"`
    Score      int    `json:"score"`
}

// -
type UserScoresOptional struct {
    gmeta.Meta `orm:"table:user_scores"`
    Id         int    `json:"id"`
    Uid        int    `json:"uid"`
    Score      int    `json:"score"`
}

//
type UserScores struct {
    gmeta.Meta `orm:"table:user_scores"`
    Id         int    `json:"id"`
    Uid        int    `json:"uid"`
    Required   []UserScoresRequired `orm:"with:id, where:type=1"`
    Optional   []UserScoresOptional `orm:"with:id, where:type=2"`
}

//
type User struct {
    gmeta.Meta `orm:"table:user"`
    Id         int    `json:"id"`
    Name       string `json:"name"`
    UserDetails *UserDetails `orm:"with:uid=id"`
    UserScores []UserScores `orm:"with:uid=id"`
}

```

```

type UserDetails struct {
    gmeta.Meta `orm:"table:user_detail"`
    Uid        int    `json:"uid"`
    Address     string `json:"address"`
}

type UserScores struct {
    gmeta.Meta `orm:"table:user_scores"`
    Id         int    `json:"id"`
    Uid        int    `json:"uid"`
    Score      int    `json:"score"`
}

type User struct {
    gmeta.Meta `orm:"table:user"`
    *UserDetails `orm:"with:uid=id"`
    Id          int    `json:"id"`
    Name        string `json:"name"`
    UserScores  []*UserScores `orm:"with:uid=id"`
}

```

```

type UserDetails struct {
    Uid    int    `json:"uid"`
    Address string `json:"address"`
}

type UserDetailsEmbedded struct {
    UserDetails
}

type UserScores struct {
    Id    int    `json:"id"`
    Uid   int    `json:"uid"`
    Score int    `json:"score"`
}

type User struct {
    *UserDetailsEmbedded `orm:"with:uid=id"`
    Id                  int    `json:"id"`
    Name                string `json:"name"`
    UserScores          []*UserScores `orm:"with:uid=id"`
}

```

```

type UserDetails struct {
    gmeta.Meta `orm:"table:user_detail"`
    Uid      int    `json:"uid"`
    Address  string `json:"address"`
}

type UserScores struct {
    gmeta.Meta `orm:"table:user_scores"`
    Id         int    `json:"id"`
    Uid        int    `json:"uid"`
    Score      int    `json:"score"`
}

type UserEmbedded struct {
    Id   int    `json:"id"`
    Name string `json:"name"`
}

type User struct {
    gmeta.Meta `orm:"table:user"`
    UserEmbedded
    *UserDetails `orm:"with:uid=id"`
    UserScores []*UserScores `orm:"with:uid=id"`
}

```

3meta

metametaCaseSnakeUserDetailuser_detailUserScoresuser_scores

```

type UserDetails struct {
    Uid      int    `json:"uid"`
    Address  string `json:"address"`
}

type UserScores struct {
    Id   int    `json:"id"`
    Uid  int    `json:"uid"`
    Score int    `json:"score"`
}

type User struct {
    *UserDetails `orm:"with:uid=id"`
    Id           int           `json:"id"`
    Name         string        `json:"name"`
    UserScores []*UserScores `orm:"with:uid=id"`
}

```

- With

