ORM

gg.DB("")toml/yaml/json/xml/ini/propertiesyaml

v2.2.0link

link

```
type:username:password@protocol(address)[/dbname][?param1=value1&...
&paramN=valueN]
```

```
::@()/?
```

-
- tcp/udp/filetcp
- mysqlhttps://github.com/go-sql-driver/mysql multiStatementsloc

```
database:
  default:
    link:  "mysql:root:12345678@tcp(127.0.0.1:3306)/test"
  user:
    link:  "sqlite::@file(/var/data/db.sqlite3)"
```

link:

	link	extra
mysql	mysql:root:12345678@tcp(127.0.0.1:3306)/test?loc=Local&parseTime=true	mysql
mariadb	mariadb:root:12345678@tcp(127.0.0.1:3306)/test?loc=Local&parseTime=true	mysql
tidb	tidb:root:12345678@tcp(127.0.0.1:3306)/test?loc=Local&parseTime=true	mysql
pgsql	pgsql:root:12345678@tcp(127.0.0.1:5432)/test	pq
mssql	mssql:root:12345678@tcp(127.0.0.1:1433)/test?encrypt=disable	go-mssqldb
sqlite	sqlite::@file(/var/data/db.sqlite3) (: db.sqlite3)	go-sqlite3
oracle	oracle:root:12345678@tcp(127.0.0.1:5432)/test	go-oci8
clickhouse	clickhouse:root:12345678@tcp(127.0.0.1:9000)/test	clickhouse-go
dm	dm:root:12345678@tcp(127.0.0.1:5236)/test	dm

https://github.com/gogf/gf/tree/master/contrib/drivers

config.yaml

```
database:
  :
    host:                   ""
    port:                   ""
    user:                   ""
    pass:                   ""
    name:                   ""
    type:                   "(mariadb/tidb/mysql/pgsql/mssql/sqlite/oracle
/clickhouse/dm)"
    link:                   "()(Host,Port,User,Pass,Name)type"
    extra:                  "()driver"
    role:                   "()(master/slave)master"
    debug:                  "()"
    prefix:                 "()"
    dryRun:                 "()ORM()"
    charset:                "()(: utf8/gbk/gb2312)utf8"
        protocol:               "()TCP"
    weight:                 "()"
    timezone:               "():local"
    namespace:              "()Catalog&SchemaSchemaNameSpaceSchema"
    maxIdle:                "()(10)"
    maxOpen:                "()()"
    maxLifetime:            "()(30)"
        queryTimeout:           "()(ctx)"
        execTimeout:            "()(ctx)"
        tranTimeout:            "()(ctx)"
        prepareTimeout:         "()SQL(ctx)""
    createdAt:              "()"
    updatedAt:              "()"
    deletedAt:              "()"
    timeMaintainDisabled:   "()trueCreatedAt/UpdatedAt/DeletedAt"
```

(YAML)

```
database:
  default:
    host:          "127.0.0.1"
    port:          "3306"
    user:          "root"
    pass:          "12345678"
    name:          "test"
    type:          "mysql"
    extra:         "local=Local&parseTime=true"
    role:          "master"
    debug:         "true"
    dryrun:        0
    weight:        "100"
    prefix:        "gf_"
    charset:       "utf8"
    timezone:      "local"
    maxIdle:       "10"
    maxOpen:       "100"
    maxLifetime:   "30s"
         protocol
```

SQL

gdb

```
database:
  default:
  - link: "mysql:root:12345678@tcp(127.0.0.1:3306)/test"
    role: "master"
  - link: "mysql:root:12345678@tcp(127.0.0.1:3306)/test"
    role: "slave"

  user:
  - link: "mysql:root:12345678@tcp(127.0.0.1:3306)/user"
    role: "master"
  - link: "mysql:root:12345678@tcp(127.0.0.1:3306)/user"
    role: "slave"
  - link: "mysql:root:12345678@tcp(127.0.0.1:3306)/user"
    role: "slave"
```

defaultuserdefaultuserg.DB()g.DB("user")

gdbglog.LoggergdbDEBUGDEBUGdebugtrue

```
database:
  logger:
    path:    "/var/log/gf-app/sql"
    level:   "all"
    stdout:  true
  default:
    link:    "mysql:root:12345678@tcp(127.0.0.1:3306)/user_center"
    debug:   true
```

database.loggergdb -

> ⚠ ORMSQLSQLSQL

()

> ✓

gdb

ConfigNodeConfigGroup()ConfigConfigGroup

1.
2.
3.
4.
5. ConfigNode.Type
6. Master-SlaveConfigNode.Role
7. ConfigNode.Weight

```
type Config      map[string]ConfigGroup //
type ConfigGroup []ConfigNode           //
// ()
type ConfigNode  struct {
    Host             string        //
    Port             string        //
    User             string        //
    Pass             string        //
    Name             string        //
    Type             string        // mysql, sqlite, mssql, pgsql, oracle
        Link             string        // ()(Host,Port,User,Pass,Name)()
    Extra            string        // ()driver
    Role             string        // (master)mastermaster, slave
    Debug            bool          // ()
    Charset          string        // ( utf8) utf8
    Prefix           string        // ()
    Weight           int           // ()
    MaxIdleConnCount int           // ()
    MaxOpenConnCount int           // ()
    MaxConnLifetime  time.Duration // ()
}
```

gdb

gdb

https://pkg.go.dev/github.com/gogf/gf/v2/database/gdb

```
//
func AddConfigNode(group string, node ConfigNode)
// ()
func AddConfigGroup(group string, nodes ConfigGroup)

// (default)
func AddDefaultConfigNode(node ConfigNode)
// (default)
func AddDefaultConfigGroup(nodes ConfigGroup)

//
func SetDefaultGroup(groupName string)

//
func SetConfig(c Config)
```

gdbgdb.NewByGroup()gdbSetConfig

```
gdb.SetConfig(gdb.Config {
    "default" : gdb.ConfigGroup {
        gdb.ConfigNode {
            Host     : "192.168.1.100",
            Port     : "3306",
            User     : "root",
            Pass     : "123456",
            Name     : "test",
            Type     : "mysql",
            Role     : "master",
            Weight   : 100,
        },
        gdb.ConfigNode {
            Host     : "192.168.1.101",
            Port     : "3306",
            User     : "root",
            Pass     : "123456",
            Name     : "test",
            Type     : "mysql",
            Role     : "slave",
            Weight   : 100,
        },
    },
    "user-center" : gdb.ConfigGroup {
        gdb.ConfigNode {
            Host     : "192.168.1.110",
            Port     : "3306",
            User     : "root",
            Pass     : "123456",
            Name     : "test",
            Type     : "mysql",
            Role     : "master",
            Weight   : 100,
        },
    },
})
```

gdb.NewByGroup("")/

DriverDriverORMmysqlDrivermysql driverOpen

```
import (
        "database/sql"

        "github.com/gogf/gf/contrib/drivers/mysql/v2"
        "github.com/gogf/gf/v2/database/gdb"
)

type MyBizDriver struct {
        mysql.Driver
}

// Open creates and returns an underlying sql.DB object for mysql.
// Note that it converts time.Time argument to local timezone in default.
func (d *MyBizDriver) Open(config *gdb.ConfigNode) (db *sql.DB, err error)
{
        config.User = d.decode(config.User)
        config.Pass = d.decode(config.Pass)
        return d.Driver.Open(config)
}

func (d *MyBizDriver) decode(s string) string {
        //
        // ...
        return s
}
```