ORM-

gdb

```
type CacheOption struct {
        // Duration is the TTL for the cache.
        // If the parameter `Duration` < 0, which means it clear the cache
with given `Name`.
        // If the parameter `Duration` = 0, which means it never expires.
        // If the parameter `Duration` > 0, which means it expires after
`Duration`.
        Duration time.Duration

        // Name is an optional unique name for the cache.
        // The Name is used to bind a name to the cache, which means you
can later control the cache
        // like changing the `duration` or clearing the cache with
specified Name.
        Name string

        // Force caches the query result whatever the result is nil or not.
        // It is used to avoid Cache Penetration.
        Force bool
}

// Cache sets the cache feature for the model. It caches the result of the
sql, which means
// if there's another same sql request, it just reads and returns the
result from cache, it
// but not committed and executed into the database.
//
// Note that, the cache feature is disabled if the model is performing
select statement
// on a transaction.
func (m *Model) Cache(option CacheOption) *Model
```

ORM*gcache.Cache*gcache.CacheGetCache() *gcache.Cache g.DB().GetCache().Keys()

Redis

ORM*gcache.CacheRedis*gcache.CacheRedis

```
redisCache := gcache.NewAdapterRedis(g.Redis())
g.DB().GetCache().SetAdapter(redisCache)
```

-Redis

v2.2.0

Content Menu

```
// ClearCache removes cached sql result of certain table.
func (c *Core) ClearCache(ctx context.Context, table string) (err error)

// ClearCacheAll removes all cached sql result from cache
func (c *Core) ClearCacheAll(ctx context.Context) (err error)
```

CoreCoreDBCore

```
g.DB().GetCore()
```

```
CREATE TABLE `user` (
  `uid` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(30) NOT NULL DEFAULT '' COMMENT '',
  `site` varchar(255) NOT NULL DEFAULT '' COMMENT '',
  PRIMARY KEY (`uid`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
```

```
package main

import (
        "time"

        "github.com/gogf/gf/v2/database/gdb"
        "github.com/gogf/gf/v2/frame/g"
        "github.com/gogf/gf/v2/os/gctx"
)

func main() {
        var (
                db  = g.DB()
                ctx = gctx.New()
        )

        // SQL
        db.SetDebug(true)

        //
        _, err := g.Model("user").Ctx(ctx).Data(g.Map{
                "name": "john",
                "site": "https://goframe.org",
        }).Insert()

        // 21()
        for i := 0; i < 2; i++ {
                r, _ := g.Model("user").Ctx(ctx).Cache(gdb.CacheOption{
                        Duration: time.Hour,
                        Name:     "vip-user",
                        Force:    false,
                }).Where("uid", 1).One()
                g.Log().Debug(ctx, r.Map())
        }

        //
        _, err = g.Model("user").Ctx(ctx).Cache(gdb.CacheOption{
                Duration: -1,
                Name:     "vip-user",
                Force:    false,
        }).Data(gdb.Map{"name": "smith"}).Where("uid", 1).Update()
        if err != nil {
                g.Log().Fatal(ctx, err)
        }

        //
        r, _ := g.Model("user").Ctx(ctx).Cache(gdb.CacheOption{
                Duration: time.Hour,
                Name:     "vip-user",
                Force:    false,
        }).Where("uid", 1).One()
        g.Log().Debug(ctx, r.Map())
}
```

```
2022-02-08 17:36:19.817 [DEBU] {c0424c75f1c5d116d0df0f7197379412} {"name":"
john","site":"https://goframe.org","uid":1}
2022-02-08 17:36:19.817 [DEBU] {c0424c75f1c5d116d0df0f7197379412} {"name":"
john","site":"https://goframe.org","uid":1}
2022-02-08 17:36:19.817 [DEBU] {c0424c75f1c5d116d0df0f7197379412} [  0 ms]
[default] [rows:1  ] UPDATE `user` SET `name`='smith' WHERE `uid`=1
2022-02-08 17:36:19.818 [DEBU] {c0424c75f1c5d116d0df0f7197379412} [  1 ms]
[default] [rows:1  ] SELECT * FROM `user` WHERE `uid`=1 LIMIT 1
2022-02-08 17:36:19.818 [DEBU] {c0424c75f1c5d116d0df0f7197379412} {"name":"
smith","site":"https://goframe.org","uid":1}
```

1. debugSQL
2. OneSQLSQLSQL
3. vip-user
4. Update
5. One