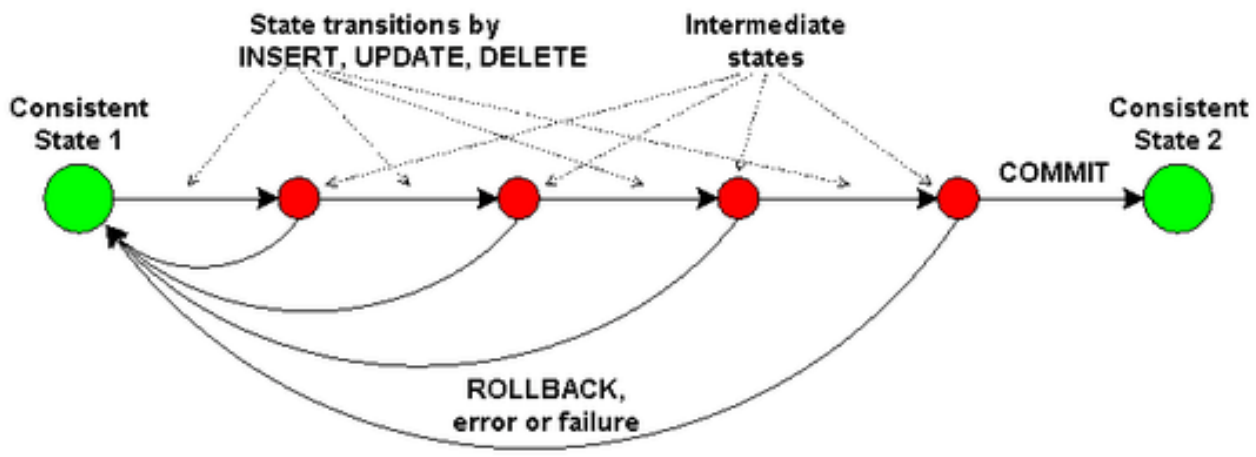

内存数据库的事务怎么做

达达 · 2 年前

上周我在Gopher China的分享内容中有讲到我们游戏服务器上用了一套自己开发的内存数据库，它从MySQL映射数据库结构，并且支持事务。

可能因为当时讲的比较匆忙，加上PPT的篇幅有限，有些朋友对这部分还是有疑问，回来以后比较多朋友问我这个内存数据库事务怎么实现的，所以我趁今天有空门写这一篇文章来细讲一下这个无用技能。

在直接用数据库的时候，我们都体验过事务（没体验过请自行实验），当一个业务需要多步骤的数据非查询操作时，数据库事务机制可以帮我们保证数据的完整性，不至于出现像银行转账操作，钱款已扣除但是对方没收到这样的情况。

而一旦我们脱离了数据库自己来管理数据，数据的完整性就需要自己维护了，最土的办法就是按需来，每种业务都写上各自的错误处理和数据回滚逻辑，但是这样是最不保险的，人是最容易犯错的，所以需要想办法把做一套事务机制。

我当初刚在设计《神仙道》的内存数据库的时候在事务这块卡了好久，这就是科班出身和野路子的最大差别，缺乏理论知识又没做过，所以就不知道怎么做了。但是想清楚了以后，发现原理非常简单。

我们把非查询操作归类一下，无非就是插入、删除、修改，这三类，我们再针对每一类研究回滚方案，插入的数据在回滚时需要删除，删除的数据在回滚时需要插入，修改的数据需要在回滚时修改回旧数据，就这三种情况。

要知道操作类型和操作的数据，就需要有一个列表来记录事务中的操作，于是就有了事务日志这样一个东西。

事务日志中的每一步都有对应的提交和回滚动作，于是就有了这样一个接口：

```
type TransLog interface {
        Commit(*Database)
        Rollback(*Database)
}
```

假设我们的数据库中有一张表叫player_item，内存中的数据库结构映射大概像这样：

```
type Database struct {
        transLogs  []TransLog
        playerItem map[int]*PlayerItem
}

type PlayerItem struct {
        Id     int
        ItemId int
        Num    int
}
```

在对这张表进行增删改操作的时候，我们记录下操作类型和新旧数据，于是我们就有了事务日志。

```
func (db *Database) InsertPlayerItem(playerItem *PlayerItem) {
        db.playerItem[playerItem.Id] = playerItem
```

```
        db.transLogs = append(db.transLogs, &PlayerItemTransLog{
                Type: INSERT, New: playerItem,
        })
}

func (db *Database) DeletePlayerItem(playerItem *PlayerItem) {
        old := db.playerItem[playerItem.Id]
        delete(db.playerItem, playerItem.Id)
        db.transLogs = append(db.transLogs, &PlayerItemTransLog{
                Type: DELETE, Old: old,
        })
}

func (db *Database) UpdatePlayerItem(playerItem *PlayerItem) {
        old := db.playerItem[playerItem.Id]
        db.playerItem[playerItem.Id] = playerItem
        db.transLogs = append(db.transLogs, &PlayerItemTransLog{
                Type: UPDATE, Old: old, New: playerItem,
        })
}
```

因为数据库只管用事务日志接口，不管具体事务日志的实现，所以我们需要实现player_item表的事务日志：

```
type TransType int

const (
        INSERT TransType = iota
        DELETE
        UPDATE
)

type PlayerItemTransLog struct {
        Type TransType
        Old  *PlayerItem
        New  *PlayerItem
}

func (transLog *PlayerItemTransLog) Commit(db *Database) {
        switch transLog.Type {
        case INSERT:
                fmt.Printf(
                        "INSERT INTO player_item (id, item_id, num) VALUES (%d
```

```
                        transLog.New.Id, transLog.New.ItemId, transLog.New.Num
                )
        case DELETE:
                fmt.Printf(
                        "DELETE player_item WHERE id = %d\n",
                        transLog.Old.Id,
                )
        case UPDATE:
                fmt.Printf(
                        "UPDATE player_item SET id = %d, item_id = %d, num = %
                        transLog.New.Id, transLog.New.ItemId, transLog.New.Num
                )
        }
}

func (transLog *PlayerItemTransLog) Rollback(db *Database) {
        switch transLog.Type {
        case INSERT:
                delete(db.playerItem, transLog.New.Id)
        case DELETE:
                db.playerItem[transLog.Old.Id] = transLog.Old
        case UPDATE:
                db.playerItem[transLog.Old.Id] = transLog.Old
        }
}
```

我们把回滚和提交的逻辑包装起来，于是内存数据库就有了事务机制：

```
func (db *Database) Transaction(trans func()) {
        defer func() {
                if err := recover(); err != nil {
                        for i := len(db.transLogs) - 1; i >= 0; i-- {
                                db.transLogs[i].Rollback(db)
                        }
                        panic(err)
                } else {
                        for _, tl := range db.transLogs {
                                tl.Commit(db)
                        }
                }
                db.transLogs = db.transLogs[0:0]
        }()
        trans()
```

```
}
```

因为事务日志是顺序记录的，后一步操作的数据可能由前一步产生，所以回滚的时候需要倒序，从最后一步开始回滚。

因为格式很固定，所以这些代码很容易用代码生成器生成。

完整的代码在：go-labs/labs30.go at master · idada/go-labs · GitHub

现在你就会发现所谓“内存数据库”和“内存数据库事务”完全就是标题党嘛，没什么好神奇的，恭喜你又掌握一个无用技能 ： ）


14 条评论

skoo

内存和数据库是如何保持实时同步的？？？如何发现数据库的变化？

2 年前

达达（作者）回复 **skoo**

查看对话

已经记录下事务日志了，把事务日志发给同步线程，同步线程把内存数据库的事务日志转成真是数据库的操作，在真数据库上执行一遍，数据就同步了。

2 年前

半夜狗叫

刚好需要这方面的经验，get it

1 年前

fleuria

这个技巧比较接近 CQRS/Event Source

1 年前

上一页 1 2

推荐阅读

性能相对论浅说

程序员间最容易引发混战的一些话题：编程语言、框架、算法、操作系统 ... 等，只要一聊到这些话题，难免就是一场混战，各抒己见，谁也别想说… 查看全文 >

达达 · 1 年前 发表于 程序员达达

记 Gopher China 2015

今年很荣幸受 @asta谢 邀请，作为演讲嘉宾参加了第一次全国性的Go语言开发者大会。非常感谢 @asta谢 和七牛商务团队把这次大会组织得这么好，… 查看全文 >

达达 · 2 年前 发表于 程序员达达

“品效合一”这事儿靠谱么？

十八届五中全会以来，数字广告领域开始流传一个词儿，叫“品效合一”，意思是说，在做品牌宣传的同时，也要获得足够的直接客户。打个

比高 在 查看全文 >

北冥乘海生 · 1 个月前 · 编辑精选

满口都是黄段子，想不到你是这样的莎士比亚

一在莎士比亚的戏剧《特洛埃围城记》（又名《特洛埃勒斯与克蕾雪达》，朱生豪译）中，特洛埃勒斯与克蕾雪达好不容易走在一起，春宵一度。“那… 查看全文 >

王辉城 · 5 天前 · 编辑精选 发表于 法科奥夫