



# 乐观锁



甘干肝

关注

 0.194

2022.03.26 16:12:07

字数 3,194

阅读 8,916

## 一、乐观锁理论基础

所谓的乐观锁，其实主要就是一种思想，因为乐观锁的操作过程中其实没有没有任何锁的参与，乐观锁只是和悲观锁相对，严格的说乐观锁不能称之为锁。所以要了解乐观锁的概念，通常与悲观锁对比起来看才更好理解，下面我们就通过乐观锁与悲观锁的对比来更好的理解乐观锁。

### 1.1乐观锁与悲观锁的概念

- **乐观锁**：总是假设最好的情况，每次去拿数据的时候都认为别人不会修改，所以不会上锁，只在更新的时候会判断一下在此期间别人有没有去更新这个数据。

注意“在此期间”的含义是拿到数据到更新数据的这段时间。因为没有加锁，所以别的线程可能会更改。还有一点那就是乐观锁其实是不加锁的来保证某个变量一系列操作原子性的一种方法。

- **悲观锁**：总是假设最坏的情况，每次去拿数据的时候都认为别人会修改，所以每次在拿数据的时候都会上锁，这样别人想拿这个数据就会阻塞，直到它拿到锁（共享资源每次只给一个线程使用，其它线程阻塞，用完后再把资源转让给其它线程）。传统的关系型数据库里边就用到了很多这种锁机制，比如行锁，表锁等，读锁，写锁等，都是在做操作之前先上锁。  
Java中 `synchronized` 和 `ReentrantLock` 等独占锁就是悲观锁思想的实现。

### 1.2 两种锁的使用场景

从上面对两种锁的介绍，我们知道两种锁各有优缺点，不可认为一种好于另一种，像**乐观锁适用于写比较少的情况下（多读场景）**，即冲突真的很少发生的时候，这样可以省去了锁的开销，加大了系统的整个吞吐量。但如果是多写的情况，一般会经常产生冲突，这就会导致上层应用会不断的进行retry，这样反倒是降低了性能，所以**一般多写的场景下用悲观锁就比较合适**。

### 1.3 乐观锁常见的两种实现方式

乐观锁可以通过版本号机制或者CAS算法实现。

#### 1.3.1 版本号机制

版本号机制实现的方式常用的也有两种：

#### 热门故事

- 超A女霸总在线撩夫
- 我把老板当金库，老板把我当老婆
- 男神学长求放过，你的节操掉了
- 情深见于微

#### 推荐阅读

- 职场内耗  
阅读 335
- 传话筒  
阅读 2,998
- 走在春天里  
阅读 1,702
- 选择适合自己的家庭理财方式  
阅读 642
- 所有失去，都会以另一种方式归来。  
阅读 885



python教学



app开发的公司



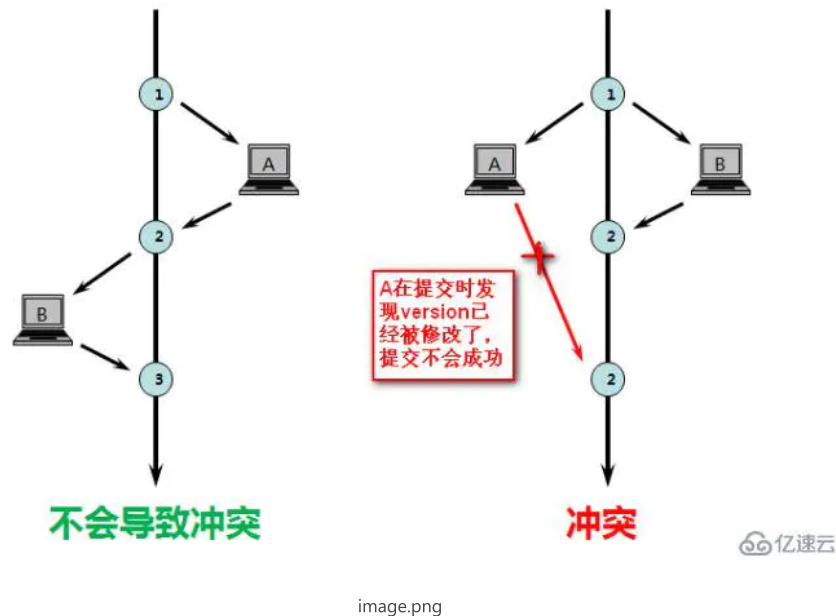
下载阅读软件



宝宝奶粉推荐

段来实现。当读取数据时，将version字段的值一同读出，数据每更新一次，对此version值加一。

当我们提交更新的时候，判断数据库表对应记录的当前版本信息与第一次取出来的version值进行比对，如果数据库表当前版本号与第一次取出来的version值相等，则予以更新，否则认为是过期数据。用下面的一张图来说明：



如上图所示，如果更新操作顺序执行，则数据的版本（version）依次递增，不会产生冲突。但是如果发生有不同的业务操作对同一版本的数据进行修改，那么，先提交的操作（图中B）会把数据version更新为2，当A在B之后提交更新时发现数据的version已经被修改了，那么A的更新操作会失败。

- **使用时间戳（timestamp）**。这种实现方式和第一种差不多，同样是在需要乐观锁控制的table中增加一个字段，名称无所谓，字段类型使用时间戳（timestamp），和上面的version类似，也是在更新提交的时候检查当前数据库中数据的时间戳和自己更新前取到的时间戳进行对比，如果一致则OK，否则就是版本冲突。

1.3.2 CAS算法

即Compare And Swap（比较与交换），是一种有名的无锁算法。无锁编程，即不使用锁的情况下实现多线程之间的变量同步，也就是在没有线程被阻塞的情况下实现变量的同步，所以也叫非阻塞同步（Non-blocking Synchronization）。CAS操作包含三个操作数——内存位置的值（V）、预期原值（A）和新值（B）。执行CAS操作的时候，将内存位置的值与预期原值比较，如果相匹配，那么处理器会自动将该位置值更新为新值，否则，处理器不做任何操作。我们使用一个例子来解释相信你会更加的清楚。

1.在内存地址V当中，存储着值为10的变量。



热门故事

超A女霸总在线撩夫

我把老板当金库，老板把我当老婆

男神学长求放过，你的节操掉了

情深见于微

推荐阅读

职场内耗

阅读 335

传话筒

阅读 2,998

走在春天里

阅读 1,702

选择适合自己的家庭理财方式

阅读 642

所有失去，都会以另一种方式归来。

阅读 885

python教学

app开发的公司

下载阅读软件

宝宝奶粉推荐

2.此时线程1想要把变量的值增加1。对线程1来说，旧的预期值A=10，要修改的新值B=11。



内存地址V

线程1: A = 10    B = 11

image.png

3.在线程1要提交更新之前，另一个线程2抢先一步，把内存地址V中的变量值率先更新成了11。



内存地址V

线程1: A = 10    B = 11

线程2: 把变量值更新为11

image.png

4.线程1开始提交更新，首先进行A和地址V的实际值比较（Compare），发现A不等于V的实际值，提交失败。



内存地址V

线程1: A = 10    B = 11  
A != V的值 ( 10!=11 )  
提交失败！

线程2: 把变量值更新为11

image.png

5.线程1重新获取内存地址V的当前值，并重新计算想要修改的新值。此时对线程1来说，

热门故事

- 超A女霸总在线撩夫
- 我把老板当金库，老板把我当老婆
- 男神学长求放过，你的节操掉了
- 情深见于微

推荐阅读

- 职场内耗  
阅读 335
- 传话筒  
阅读 2,998
- 走在春天里  
阅读 1,702
- 选择适合自己的家庭理财方式  
阅读 642
- 所有失去，都会以另一种方式归来。  
阅读 885



python教学



app开发的公司



下载阅读软件



宝宝奶粉推荐



内存地址V

线程1: A = 11    B = 12

image.png

6.这一次比较幸运，没有其他线程改变地址V的值。线程1进行Compare，发现A和地址V的实际值是相等的。



内存地址V

线程1: A = 11    B = 12  
A == V的值 ( 11 == 11 )

image.png

7.线程1进行SWAP，把地址V的值替换为B，也就是12。



内存地址V

线程1: A = 11    B = 12  
A == V的值 ( 11 == 11 )  
地址V的值更新为12

image.png

注：CAS算法的缺点

【1】循环时间长开销很大：自旋 CAS 如果长时间不成功，会给 CPU 带来非常大的执行开销。

【2】只能保证一个共享变量的原子操作：只能保证一个共享变量的原子操作。当对一个共享变量执行操作时，我们可以使用循环 CAS 的方式来保证原子操作，但是对多个共享变量操作时，循环 CAS 就不能保证操作的原子性。这个时候就可以用锁，或者有一个原子变量。

热门故事

超A女霸总在线撩夫

我把老板当金库，老板把我当老婆

男神学长求放过，你的节操掉了

情深见于微

推荐阅读

职场内耗  
阅读 335

传话筒  
阅读 2,998

走在春天里  
阅读 1,702

选择适合自己的家庭理财方式  
阅读 642

所有失去，都会以另一种方式归来。  
阅读 885



python教学



app开发的公司



下载阅读软件



宝宝奶粉推荐

进行 CAS 操作。

【3】ABA 问题：因为 CAS 需要在操作值的时候检查下值有没有发生变化，如果没有发生变化则更新，但是如果一个值原来是A，变成了B，又变成了A，那么使用 CAS 进行检查时会发现它的值没有发生变化，但是实际上却变化了。ABA 问题的解决思路就是使用版本号。在变量前面追加版本号，每次变量更新的时候把版本号加1，那么A - B - A 就会变成1A-2B-3A。

## 二、乐观锁两种方式的实现实例

### 2.1 利用版本号机制的解决实际问题

#### 2.1.1 实际遇到的问题

使用 MySQL 5.7 做测试，数据库引擎为 InnoDB，数据库隔离级别为可重复读（REPEATABLE-READ），读读共享，读写互斥。在这个隔离级别下，在多事务并发的情况下，还是会出现数据更新的冲突问题。

先分析一下更新冲突的问题是如何产生的。  
假设我们有一张商品表 goods，表结构如下：

字段	数据类型	说明
goods_id	varchar(32)	商品 id
count	int(11)	销量

比如在某一时刻事务 A 和事务 B，在同时操作表 goods\_id = 213214324 的数据，当前销量为 100。

goods_id	count
213214324	100

两个事务的内容一样，都是先读取的数据，count + 100 后更新。

我们这里只讨论乐观锁的实现，为了便于描述，假设项目已经集成 Spring 框架，使用 MyBatis 做 ORM，Service 类的所有方法都使用了事务，事务传播级别使用 PROPAGATION\_REQUIRED，在事务失败会自动回滚。

Service 为 GoodsService，更新数量的方法为 addCount()。

```
1 @Service
2 @Transaction
3 public class GoodsService{
4
5     @Autowired
6     private GoodsDao dao;
7
8     public void addCount(String goodsId, Integer count) {
9         Goods goods = dao.selectByGoodsId(goodsId);
10         if (goodsSale == null) {
11             throw new Exception("数据不存在");
12         }
13         int count = goods.getCount() + count;
14         goods.setCount(count);
15         int count = dao.updateCount(goods);
16         if (count == 0) {
17             throw new Exception("添加数量失败");
18         }
19     }
20 }
```

#### 热门故事

- 超A女霸总在线撩夫
- 我把老板当金库，老板把我当老婆
- 男神学长求放过，你的节操掉了
- 情深见于微

#### 推荐阅读

- 职场内耗 阅读 335
- 传话筒 阅读 2,998
- 走在春天里 阅读 1,702
- 选择适合自己的家庭理财方式 阅读 642
- 所有失去，都会以另一种方式归来。 阅读 885



python教学



app开发的公司



下载阅读软件



宝宝奶粉推荐

```
1 public interface GoodsSaleDao {
2     Goods selectByGoodsId(@Param("goodsId") String goodsId);
3
4     int updateCount(@Param("record") Goods goods);
5 }
```

mapper 文件对应的 sql 操作为:

```
1 <!-- 查询 -->
2 <select id="selectByGoodsId" resultMap="BaseResultMap">
3     select
4     <include refid="Base_Column_List"/>
5     from goods
6     where goods_id = #{goodsId}
7 </select>
8
9 <!-- 更新 -->
10 <update id="updateCount">
11     update
12     goods
13     set count = #{record.count},
14     where goods_id = #{record.goodsId}
15 </update>
```

好了，假设现在有两个线程同时调用了 `GoodsService` 的 `addCount()`，操作同一行数据，会有什么问题？

是否可能会出现两个线程更新时出现了冲突！两次 `addCount(100)`，结果应该是 300，但结果还是 200。因为上面对一个变量的查询与更新这一系列操作并不是原子性的，是有可能出现并发问题的。

### 热门故事

- 超A女霸总在线撩夫
- 我把老板当金库，老板把我当老婆
- 男神学长求放过，你的节操掉了
- 情深见于微

### 推荐阅读

- 职场内耗
- 阅读 335
- 传话筒
- 阅读 2,998
- 走在春天里
- 阅读 1,702
- 选择适合自己的家庭理财方式
- 阅读 642
- 所有失去，都会以另一种方式归来。
- 阅读 885



python教学



app开发的公司



下载阅读软件



宝宝奶粉推荐

image.png

### 2.1.2 如何解决问题

该如何处理上面的问题，有一个简单粗暴的方法，既然这里多线程访问会有线程安全问题，那就上锁，方法加上 `synchronized` 进行互斥。





```
3         if (goodsSale == null) {
4             throw new Exception("数据不存在");
5         }
6         int count = goods.getCount() + count;
7         goods.setCount(count);
8         int count = dao.updateCount(goods);
9         if (count == 0) {
10             throw new Exception("添加数量失败");
11         }
12     }
```

这个方案确实也可以解决问题，但是这种简单互斥的做法，锁的粒度太高，事务排队执行，并发度低，性能低。但如果是分布式应用，还得考虑应用分布式锁，性能就更低了。考虑到这些更新冲突发生的概率其实并不高。这里讨论另一种解决方案，在数据库表中新增版本号字段来实现乐观锁从而保证该变量操作的原子性。

接下来我们来讨论如何实现它。

第一步：数据库表 goods 新增一行 data\_version 来记录数据更新的版本号。新的表结构如下：

字段	数据类型	说明
goods_id	varchar(32)	商品 id
count	int(11)	销量
data_version	int(11)	版本号

第二步：GoodsDao 的 updateCount() 对应的 mapper 的 SQL 语句进行调整，数据更新的时候同时进行 data\_version = data\_version + 1，执行这个 sql 时候已经对数据上行锁了，所以这个 data\_version 加 1 的操作作为原子操作。

```
1 <!-- 乐观锁更新 -->
2 <update id="updateCount">
3     update
4     goods_sale
5     set count = #{record.count}, data_version = data_version + 1
6     where goods_sale_id = #{record.goodsSaleId}
7     and data_version = #{record.dataVersion}
8 </update>
```

Dao 调整之后，事务 A 和事务 B 的变化如下：

### 热门故事

超A女霸总在线撩夫

我把老板当金库，老板把我当老婆

男神学长求放过，你的节操掉了

情深见于微

### 推荐阅读

职场内耗  
阅读 335

传话筒  
阅读 2,998

走在春天里  
阅读 1,702

选择适合自己的家庭理财方式  
阅读 642

所有失去，都会以另一种方式归来。  
阅读 885



python教学



app开发的公司



下载阅读软件



宝宝奶粉推荐

有了发现冲突快速失败的方案，要想让更新成功，可以在 GoodsService 中加入自旋，重新开始事务业务逻辑的执行，直到没有发生冲突，更新成功。自旋的实现有两种，一种是使用循环，一种是使用递归。

• 循环实现：

```
1 public void addCount(String goodsId, Integer count) {
2     while(true) {
3         Goods goods = dao.selectByGoodsId(goodsId);
4         if (goods == null) {
5             throw new Exception("数据不存在");
6         }
7         int count = goods.getCount() + count;
8         goods.setCount(count);
9         int count = dao.updateCount(goods);
10        if (count > 0) {
11            return;
12        }
13    }
14 }
```

• 递归实现：

```
1 public void addCount(String goodsId, Integer count) {
2     Goods goods = dao.selectByGoodsId(goodsId);
3     if (goods == null) {
4         throw new Exception("数据不存在");
5     }
6     int count = goods.getCount() + count;
7     goods.setCount(count);
8     int count = dao.updateCount(goods);
9     if (count == 0) {
10        addCount(goodsId, count)
11    }
12 }
```


通过乐观锁+自旋的方式，解决数据更新的线程安全问题，而且锁粒度比互斥锁低，并发性能好。  
使用时间戳（timestamp），这种解决方式和上面的差不多，这里就不展开介绍了。


## 2.2 利用CAS算法解决实际问题

Java中java.util.concurrent.atomic 并发包下的所有原子类都是基于 CAS 来实现的。  
这里涉及到java的源码分析，这里就不展开了。

关于CAS相关连接推荐：

- CAS底层实现
- 如何实现一个乐观锁

 3人点赞 > 

 java多线程相关 

更多精彩内容，就在简书APP



热门故事

超A女霸总在线撩夫

我把老板当金库，老板把我当老婆

男神学长求放过，你的节操掉了

情深见于微

推荐阅读

职场内耗

阅读 335

传话筒

阅读 2,998

走在春天里

阅读 1,702

选择适合自己的家庭理财方式

阅读 642

所有失去，都会以另一种方式归来。

阅读 885

  
python教学

  
app开发的公司

  
下载阅读软件

  
宝宝奶粉推荐



赞赏支持

还没有人赞赏，支持一下



甘干肝

总资产0.916 共写了8938字 获得8个赞 共2个粉丝

关注

7月25日上线送打金神器，一刀9999，高爆6666！



高爆打金服·效卓

广告

推荐阅读

更多精彩内容>

乐观锁和悲观锁 --转载自JavaGuide

原文地址: <https://snailclimb.top/JavaGuide/#/essential-content...>



了凡\_8504 阅读 231 评论 0 赞 0

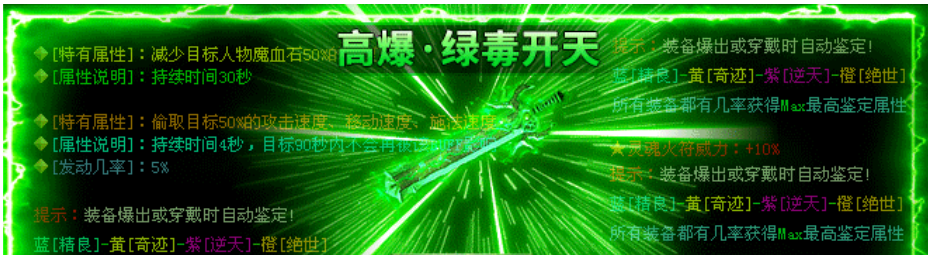
乐观锁 & 悲观锁 CAS

<https://www.cnblogs.com/kismetv/p/10787228.html><https://w...>



peerben 阅读 180 评论 0 赞 0

【限时福利】登录即送神器开天，进服必爆神装，光速升级！



打金传奇·效卓

广告

乐观锁和悲观锁

乐观锁对应于生活中乐观的人总是想着事情往好的方向发展，悲观锁对应于生活中悲观的人总是想着事情往坏的方向发展。所以这...



文茶君 阅读 221 评论 0 赞 0

Java--乐观锁与悲观锁以及乐观锁的一种实现方式-CAS

乐观锁或者悲观锁，都是一种思想，而非是真的锁 悲观锁 悲观锁，就是不管是否发生多线程冲突，只要存在这种可能，就每次...



博弈史密斯 阅读 589 评论 0 赞 2

热门故事

超A女霸总在线撩夫

我把老板当金库，老板把我当老婆

男神学长求放过，你的节操掉了

情深见于微

推荐阅读

职场内耗

阅读 335

电话筒

阅读 2,998

走在春天里

阅读 1,702

选择适合自己的家庭理财方式

阅读 642

所有失去，都会以另一种方式归来。

阅读 885



python教学



app开发的公司



下载阅读软件



宝宝奶粉推荐

乐观锁与悲观锁

写下你的评论...

评论0

赞3

### 一篇文章带你学会面试必备的乐观锁与悲观锁

什么是悲观锁与乐观锁 乐观锁对应于生活中乐观的人总是想着事情往好的方向发展，悲观锁对应于生活中悲观的人总是想着事情...

Java柚子 阅读 106 评论 0 赞 0



### 大数据平台哪个好?



### 面试必备之乐观锁与悲观锁

何谓悲观锁与乐观锁 乐观锁对应于生活中乐观的人总是想着事情往好的方向发展，悲观锁对应于生活中悲观的人总是想着事情往...

徐同学呀 阅读 308 评论 0 赞 1

### 面试必备|乐观锁与悲观锁的大揭秘

先点赞后观看，养成好习惯! 何谓悲观锁与乐观锁 乐观锁对应于生活中乐观的人总是想着事情往好的方向发展，悲观锁对应于...

JAVA炭烧 阅读 541 评论 0 赞 4

### 乐观锁与悲观锁

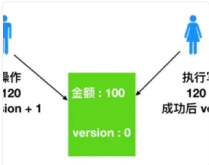
何谓悲观锁与乐观锁 乐观锁对应于生活中乐观的人总是想着事情往好的方向发展，悲观锁对应于生活中悲观的人总是想着事情往...

valor\_wang 阅读 213 评论 0 赞 1

### 看完你就知道的乐观锁和悲观锁

看完你就知道的乐观锁和悲观锁 Java 锁之乐观锁和悲观锁 [TOC] Java 按照锁的实现分为乐观锁和悲观锁，...

程序员will 阅读 268 评论 0 赞 0



### 传奇版本的区别



### 悲观锁与乐观锁

悲观锁 总是假设最坏的情况，每次去拿数据的时候都认为别人会修改，所以每次在拿数据的时候都会上锁，这样别人想拿这个数...

王侦 阅读 311 评论 0 赞 0

### 悲观锁与乐观锁你还记得吗?

何谓悲观锁与乐观锁 乐观锁对应于生活中乐观的人总是想着事情往好的方向发展，悲观锁对应于生活中悲观

### 热门故事

超A女霸总在线撩夫

我把老板当金库，老板把我当老婆

男神学长求放过，你的节操掉了

情深见于微

### 推荐阅读

职场内耗

阅读 335

传话筒

阅读 2,998

走在春天里

阅读 1,702

选择适合自己的家庭理财方式

阅读 642

所有失去，都会以另一种方式归来。

阅读 885



python教学

app开发的公司

PDF

下载阅读软件

宝宝奶粉推荐

何谓悲观锁与乐观锁 乐观锁对应于生活中乐观的人总是想着事情往好的方向发展，悲观锁对应于生活中悲观的人总是想着事情往...

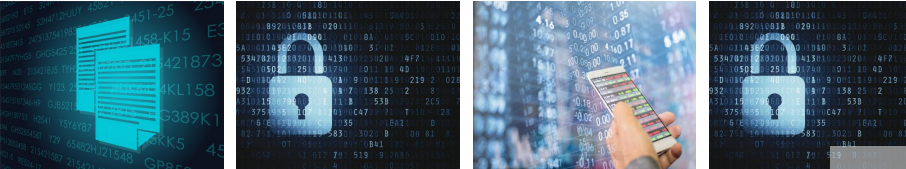
是小猪童鞋啦 阅读 67 评论 0 赞 0

乐观锁与悲观锁小结

乐观锁，就像生活中乐观的人总是想着事情往好的方向发展；悲观锁，就像生活中悲观的人总是想着事情往坏的方向发展。这两种...

DemonJun 阅读 246 评论 0 赞 3

专业数据集测试，快速掌握！



浅谈乐观锁与悲观锁

一、何谓悲观锁与乐观锁 乐观锁对应于生活中乐观的人总是想着事情往好的方向发展，悲观锁对应于生活中悲观的人总是想着事...

爱情小傻蛋 阅读 312 评论 0 赞 0

生活不曾给人希望，却常常让人失望

我印象里，身边的老人们，总喜欢把一句话挂在嘴边：“生活不如意事，十之八九！人嘛，总得，自己学会适应生活！”...

cf96731e55b0 阅读 995 评论 0 赞 4

2021-3-8晨间日记

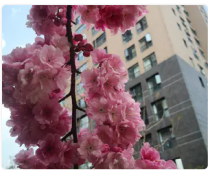
今天是第三周周一，三八女神节祝自己每天开心一点，工作顺心顺意 起床：4：25 就寝：23：56 天气：雨天 心情：...

QXCLZ 阅读 742 评论 1 赞 3

【利平成长日记】第641天

张利平2021.3.6「学习《情绪按钮》第20天收获：[太阳]今天学习内容：第七章《情绪的来源》（五）情绪的来...

张利平专注国学教育139876 阅读 1,167 评论 0 赞 2



意大利语学习班



追梦记2.0

这周的作文题目是：假设你现在的火车上，对面坐着一个漂亮的异性，去构思一段故事，想想接下来会发生什么。那么接下来我的...

2077516 阅读 1,241 评论 0 赞 0

热门故事

超A女霸总在线撩夫  
我把老板当金库，老板把我当老婆  
男神学长求放过，你的节操掉了  
情深见于微

推荐阅读

职场内耗 阅读 335  
传话筒 阅读 2,998  
走在春天里 阅读 1,702  
选择适合自己的家庭理财方式 阅读 642  
所有失去，都会以另一种方式归来。 阅读 885



python教学



app开发的公司



下载阅读软件




宝宝奶粉推荐

写下你的评论...

评论0 赞3

回答道：“并不是我很聪明，只是我和问题...”



世界和平\_众生安康

阅读 1,304

评论 0

赞 6



热门故事

超A女霸总在线撩夫

我把老板当金库，老板把我当老婆

男神学长求放过，你的节操掉了

情深见于微

推荐阅读

职场内耗  
阅读 335

传话筒  
阅读 2,998

走在春天里  
阅读 1,702

选择适合自己的家庭理财方式  
阅读 642

所有失去，都会以另一种方式归来。  
阅读 885



python教学



app开发的公司



下载阅读软件



宝宝奶粉推荐

