### 库存

数据库的锁：

共享锁、排他锁、悲观锁、乐观锁，选择哪个？

如何避免死锁？

答：

同顺序  
一次性锁定  
细粒度锁定（行锁）

请问如何解锁？

答：事务提交 或 回滚

注意：

1、

加锁必须跟事务同时使用

2、

数据库的增删改操作默认都会加排他锁，而查询不会加任何锁。

3、

MyISAM 操作数据都是使用的表锁，你更新一条记录就要锁整个表，导致性能较低，并发不高。当然同时它也不会存在死锁问题。

而 InnoDB 与 MyISAM 的最大不同有两点：一是 InnoDB 支持事务；二是 InnoDB 采用了行级锁。也就是你需要修改哪行，就可以只锁定哪行。

在InnoDB中，行级锁并不是直接锁记录，而是锁索引。索引分为主键索引和非主键索引两种，如果一条sql 语句操作了主键索引，Mysql 就会锁定这条主键索引；如果一条语句操作了非主键索引，MySQL会先锁定该非主键索引，再锁定相关的主键索引。

InnoDB 行锁是通过给索引项加锁实现的，如果没有索引，InnoDB 会通过隐藏的聚簇索引来对记录加锁。也就是说：如果不通过索引条件检索数据，那么InnoDB将对表中所有数据加锁，实际效果跟表锁一样。因为没有了索引，找到某一条记录就得扫描全表，要扫描全表，就得锁定表。

4、

乐观锁不是数据库自带的，需要我们自己去实现。

数据库的脏读、幻读、不可重复读：

<https://www.cnblogs.com/cdlyy/p/11067658.html>

级别高低：脏读 < 不可重复读 < 幻读

所以设置了数据库事务隔离级别最高级别的SERIALIZABLE\_READ就不需要设置其他的了，即解决了幻读问题那么脏度和不可重复读自然就都解决了。

### 无锁化编程

**注意：**

1. ABA问题

若线程对同一内存地址进行了两次读操作，而两次读操作得到了相同的值，通过 "值相同" 来判定 "值没变"是不可靠的。因为在这两次读操作的时间间隔之内，另外的线程可能已经多次修改了该值，这样就相当于欺骗了前面的线程，使其认为 "值没变"，实际上值已经被篡改过了。

下面是 ABA 问题发生的过程：

T1 线程从共享的内存地址读取值 A；

T1 线程被抢占，线程 T2 开始运行；

T2 线程将共享的内存地址中的值由 A 改成 B，然后又改成 A；

T1 线程继续执行，读取共享的内存地址中的值 A，认为没有改变，然后继续执行

由于 T1 并不知道两次读取的值 A 已经被 "隐性" 的修改过，所以可能产生无法预期的结果。