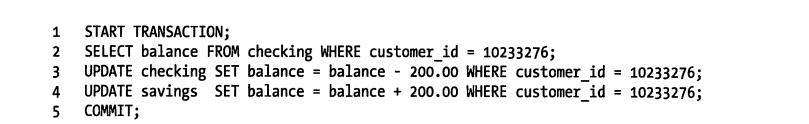
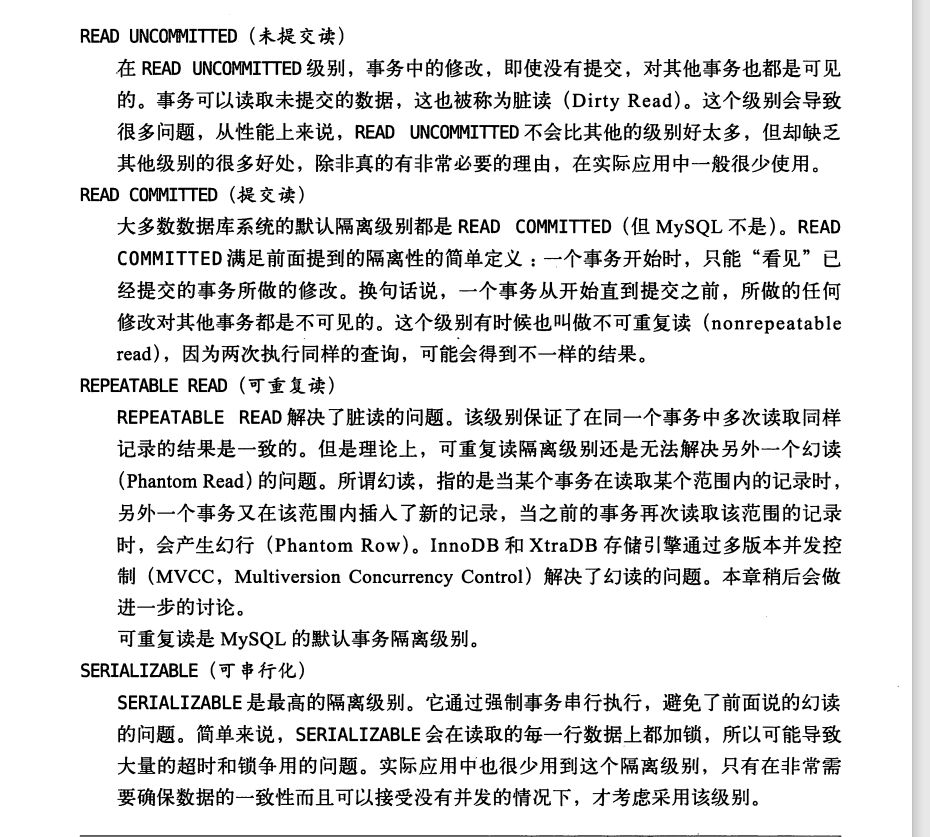
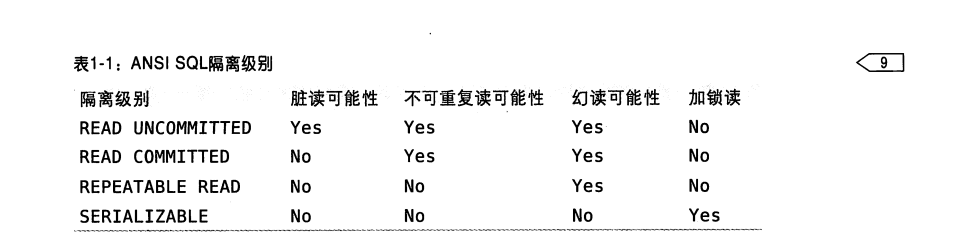


事务

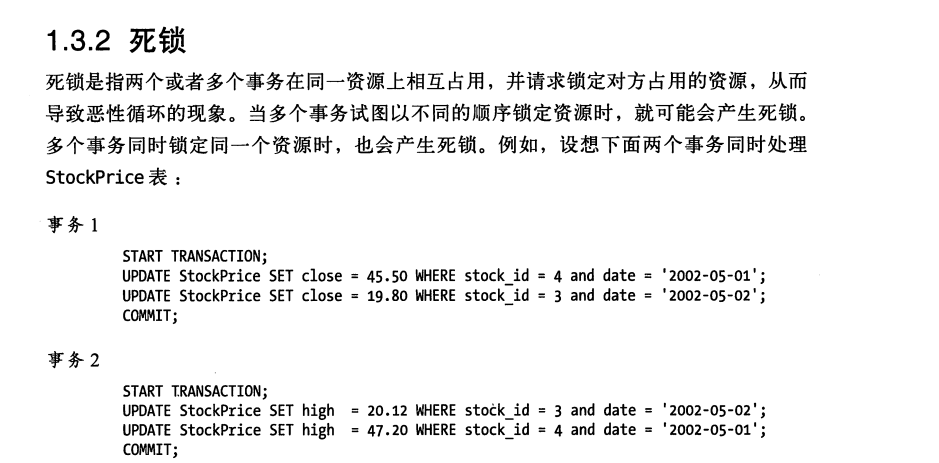


提交级别

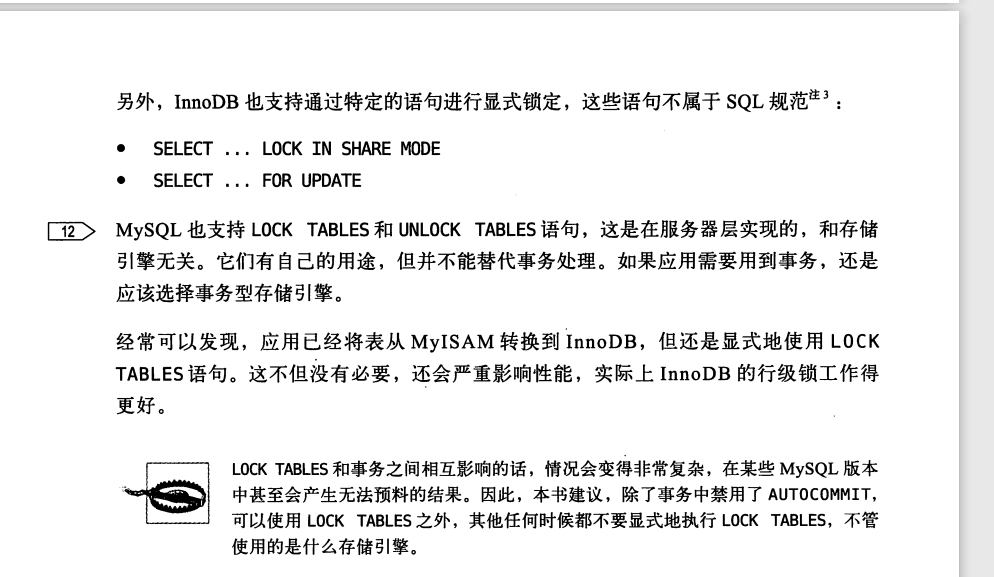




死锁的出现



显式锁



相同点：1.两者都会对并发的操作造成阻塞，等待A操作完成；

这就是共享锁，也叫读锁。就是我们对数据进行读取操作的时候，其实是不会改变数据的值的。

所以我们可以给数据库增加读锁，获得读锁的事务就可以读取数据了。当数据库已经被别人增加了读锁的时候，其他新来的事务也可以读数据，但是不能写。

也就是说，如果事务T对数据A加上共享锁后，则其他事务只能对A再加共享锁，不能加排他锁。获准共享锁的事务只能读数据，不能修改数据。

对于共享锁大家可能很好理解，就是多个事务只能读数据不能改数据，对于排他锁大家的理解可能就有些差别，我当初就犯了一个错误，以为排他锁锁住一行 数据后，其他事务就不能读取和修改该行数据，其实不是这样的。排他锁指的是一个事务在一行数据加上排他锁后，其他事务不能再在其上加其他的锁

mysql InnoDB引擎默认的修改数据语句，update,delete,insert都会自动给涉及到的数据加上排他锁，select语句默认不会加任何锁类 型，如果加排他锁可以使用select ...for update语句，加共享锁可以使用select ... lock in share mode语句。所以加过排他锁的数据行在其他事务种是不能修改数据的，也不能通过for update和lock in share mode锁的方式查询数据，但可以直接通过select ...from...查询数据，因为普通查询没有任何锁机制。

SELECT @@global.tx\_isolation;

　　　　2.查询操作不会造成阻塞(不带for update)

　　　　3.操作阻塞（带for update）

不同点：并发时for update会使B一直阻塞，等待A操作完成后执行B操作；

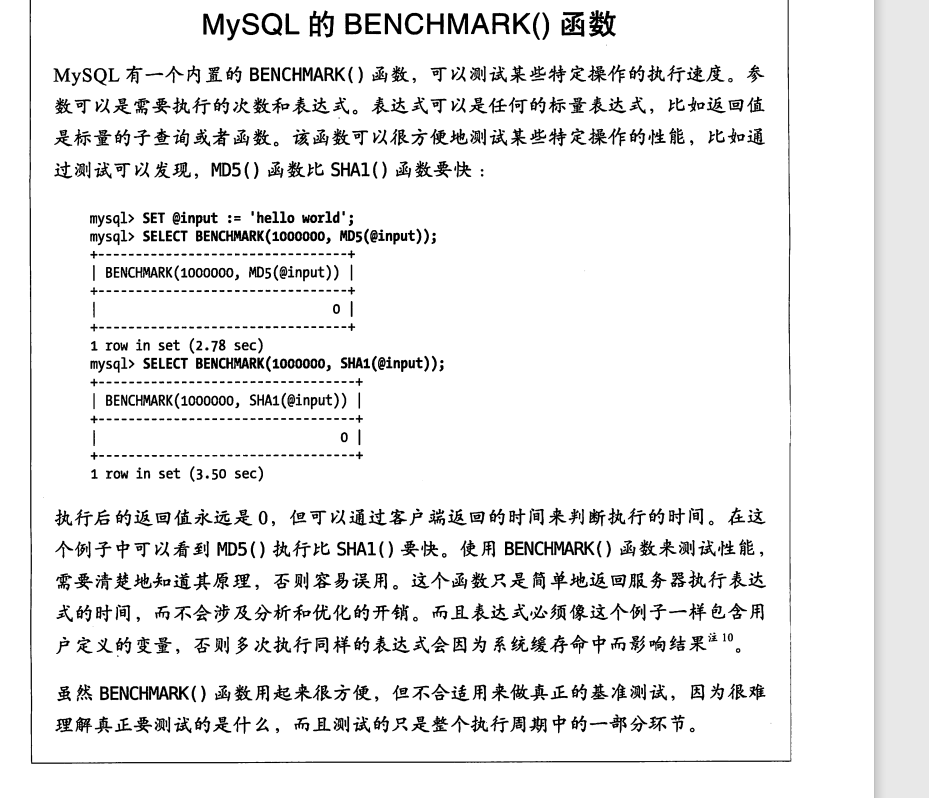
　　　　而在使用lock in share mode下当B阻塞时，如果A继续有修改数据，那么此时B会终止失败

　　lock in share mode意向共享锁（IS）

　　for update意向排它锁（IX）

## **应用场景**

在我看来，SELECT ... LOCK IN SHARE MODE的应用场景适合于两张表存在关系时的写操作，拿mysql官方文档的例子来说，一个表是child表，一个是parent表，假设child表的某一列child\_id映射到parent表的c\_child\_id列，那么从业务角度讲，此时我直接insert一条child\_id=100记录到child表是存在风险的，因为刚insert的时候可能在parent表里删除了这条c\_child\_id=100的记录，那么业务数据就存在不一致的风险。正确的方法是再插入时执行select \* from parent where c\_child\_id=100 lock in share mode,锁定了parent表的这条记录，然后执行insert into child(child\_id) values (100)就ok了。



数据类型优化

