死锁的发生：https://blog.csdn.net/tr1912/article/details/81668423（看一下）

例1::::

T1: begin tran select * from table lock in share mode update table set column1 = 'hello'

T2: begin tran select * from table lock in share mode update table set column2 = 'world'

假设T1和T2同时到达select，T1对table加共享锁，T2对table加共享锁，当T1的select执行完，准备执行update时，根据锁机制，T1的共享锁需要升级到排他性锁才能执行接下来的update。在升级排他锁前，必须等待table上的其他共享锁(T2)释放，同理，T2也在等T1的共享锁释放。于是死锁产生了。

例2:::

T1: begin tran update table set column1 = 'hello' where id = 10

T2: begin tran update table set column2 = 'world' where id = 20

这种语句虽然最常见，很多人觉得它有机会产生死锁，但实际上需要看情况。

|--如果id是主键（默认有主键索引），那么T1会一下子找到该条记录（id = 10的记录），然后对该条记录加排他锁，T2，同样，一下子通过索引定位到记录，

然后对id=20的记录加上排他锁，这样T1和T2各自更新个字的，互不影响。T2也不需要等待。

|--如果id是普通的一列，没有索引。那么当T1对id=10这一行加排他锁后，T2为了找到id=20，需要对全表扫描。但因为T1已经为一条记录加了排他锁，导致T2的全表扫描进行不下去(其实是因为T1加了排他锁，数据库默认会为该表加意向锁，T2要扫描全表，就得等该意向锁释放，也就是T1执行完成)，这样就导致了T2等待。

死锁怎么解决：

T1: begin tran select * from table for update update table set column1 = 'hello'

T2: begin tran select * from table for update update table set column2 = 'world'

这样，当T1的select 执行时，直接对表加上了排他锁，T2在执行select时，就需要等待T1事物完全执行完才能执行。排除了死锁发生，但当第三个user过来想执行一个查询语句的时候，也因为排他锁的存在而不得不等待，第四个、第五个user也会因此而等待。在大并发情况下，让大家等待显得性能太不友好了。所以，有些数据库这里引入了更新锁(如mssql，注意：mysql不存在更新锁)。