**乐观锁**

乐观锁不是数据库自带的，需要我们程序实现，如EF的并发控制，没次更新时总是检查并发字段有没有被修改过，并更新并发字段

**悲观锁**

悲观锁由数据库提供

**共享锁**

又叫S锁或者读锁，加了共享锁的数据对象可以被其他事务读取，但不能修改， 通常是该数据对象被读取完毕，锁立即被释放

多个共享锁可共存

示例：

select \* from table

当我们读取table表时，数据库会自动加上共享锁

在事务中手动增加共享锁

begin tran

select \* from table (holdlock) (holdlock意思是加共享锁，直到事物结束才释放)

update table set column1='hello'

**更新锁**

又称U锁，为解决死锁，引入更新锁，一个对象只能同意时刻只能有一个更新锁，更新锁可以和共享锁共存，即一般表被上了更新锁，但我还是可以读取他

示例：

T1:

begin tran

select \* from table(updlock) (加更新锁)

update table set column1='hello'

T2:

begin tran

select \* from table(updlock)

update table set column1='world'

T1先执行，T1加了更新锁，T2无法添加更新锁，所以T2等T1执行完后才执行

**排他锁**

又称X锁，在执行update时，数据库自动添加排他锁，同一时刻只能有一个排他锁，排他锁与共享锁不相容

示例：

update table set column1='hello' where id<1000

手动添加排它锁

begin tran

select \* from table(xlock) (xlock意思是直接对表加排他锁)

update table set column1='hello'

**死锁的产生**

T1:

begin tran

select \* from table (holdlock) (holdlock意思是加共享锁，直到事物结束才释放)

update table set column1='hello'

T2:

begin tran

select \* from table(holdlock)

update table set column1='world'

T1，T2同时执行，T1，T2都对表table 添加了共享锁，但都没有释放共享锁而后又申请排他锁

**何时使用悲观锁**

大多情况下数据库会更加隔离级别自动选择加锁，我们无需自己加锁