**事务的概念**

事务是访问并可能更新各种数据项的一个程序执行单元。事务有四个基本性质：

1、原子性：事务中的全部操作要么全部做完要么全部不做

2、隔离性：多个事务并发执行时，系统保证对于任意一对事务a和b，在a看来，b或者在a开始执行之前就已经完成了，或者在a执行之后开始执行。也就是说每个事务都觉得只有自己在执行

3、一致性：在没有其他事务并发执行的情况下，保持数据库的一致性

4、持久性：一个事务完成后，他对数据库的改变必须是永久的，即使系统崩溃

**事务的几种状态：**

1、活跃的：初始状态，事务开始执行时处于这个状态

2、部分提交的：最后一条语句执行后

3、失败的：发现正常的执行不能继续后

4、中止的：事务回滚并且数据库恢复到事务执行之前的状态后

5、提交的：成功执行后

事务进入中止状态后，系统这时候有两种选择：

1、重启事务：当且仅当中止是由硬件错误造成的或者不是事物本身逻辑所产生的错误时。重启的事务会被看成一个新的事务

2、杀死事务：如果中止是由事务内部逻辑错误导致的那么事务就会被杀死

**事务并发控制机制**

采用并发的原因：

1、提高吞吐量和资源利用率，一个事务由多个步骤组成，一些涉及io活动，一些涉及到CPU活动。两者是可以同时工作的，也就是说当一个事务在一个磁盘上进行io活动时，另一个事务可以进行CPU活动，第三个事务可以在另一个磁盘上进行io活动，这样一来就提高了资源利用率和吞吐量

2、减少等待时间，事务有长有短，如果事务串行执行，那么短事务必须等到它前面的长事务执行完成后才能执行，这样一来总的等待时间就会很长，但是如果采用并发，短事务和长事务之间可以共享CPU和磁盘，这样以来延迟就会大大减少

**可串行化**

如果两个事务他们顺序执行和并发执行的结果是一样的那么我们就称他们是可串行化的。例如有两个事务T1和T2，我们先执行T1再执行T2，如下：

表格

描述已自动生成

表格

描述已自动生成

我们假设I和J是不同事务的上的两个指令，只有当I和J都是读操作时，两条指令的顺序无关紧要；如果I和J是在相同数据库的操作，并且至少其中一个是写操作，那么这两条指令就是冲突的。非冲突指令可以相互交换，但是冲突指令不可以，如果一个调度可以经过非冲突指令交换转换成另一个调度，则称这两个事务是冲突等价的。若一个调度与一个串行调度冲突等价，则称调度室冲突可串行化的。例如：

表格

描述已自动生成 表格

描述已自动生成

显然调度1和调度2的执行结果是一样的。

可恢复调度：对于事务T1和T2，如果T2读取了之前由T1所写的数据，那么T1必须先于T2提交。

**事务隔离性级别**

SQL标准规定的隔离性级别如下：

**可串行化**：保证可串行化调度

**可重复读**：只允许读取已经提交的数据，而且一个事物两次读取一个数据项期间，其他事务不得更新数据。但不要求可串行化。

**已提交读：**只允许读已经提交的数据，不要求可重复读。

**未提交读：**允许读取没有提交的数据。

**并发控制**

并发控制有多种机制，每一种都各有利弊，最常用的有两阶段封锁和快照隔离。

基于锁的协议

为了保证对数据的互斥操作，人们引入了锁机制。事务在对数据进行操作之前必须先获得锁，操作完之后在释放锁。有两种常用的锁：

**共享锁：**如果某个事务获得了一个数据项上的共享锁（记为S），则其他事务只能对这个数据项进行读操作

**排它锁：**如果一个事务获得了一个数据项上的排它锁（记为X），则其他事务不能对这个数据项进行任何操作

表格

描述已自动生成

事务T1拥有B的排它锁，想要获取A的排它锁，但是事务T2现在正在占据着A的排它锁，并且它想获得B的共享锁，如此一来它们就形成了一个环路，导致谁也无法运行下去。解决死锁的办法就是回滚Rollback其中一个事务。

饥饿，就是说一个事务可能永远也无法获得一个数据项的锁。例如：事务T1占据了A的共享锁，另一个事务T2想要获得A的排它锁，由于两个锁不兼容，所以T2必须等待T1释放共享锁之后才能得到A的排它锁，但是这时候如果有一个T3想要获得A的共享锁，那么问题就来了，由于共享锁是互相兼容的，所以T3直接就获得了A的共享锁，T2再想获得A的排它锁就必须等T3也完成。如果再有其他的事务想要得到A的共享锁那么T2就必须等到他们都执行完，也就是说T2可能永远也得不到锁。为了解决这个问题在赋予事务锁的时候加了一些条件：

1、不存在锁冲突

2、在这个事务前面没有其他事务也想获得数据项的锁

**两阶段封锁协议**

为了保证可串行化，人们引入了两阶段封锁协议，这个协议要求事务分成两个阶段提出加锁和解锁申请：

1、增长阶段：事务可以获得锁，但是不能释放锁

2、缩减阶段：事务可以释放锁，但是不能申请新的锁

两阶段封锁协议可以保证没有死锁，但是可能会发生级联回滚，为了解决级联回滚，我们将引入严格两阶段封锁协议，他要求：所有排它锁必须在事务提交后才能释放。另一个变体是强两阶段封锁协议，他要求：事务提交之前不可以释放任何锁，这样一来就保证了可串行化。