15 并发控制

当多个事务在数据库中并发地执行时，数据的一致性可能无法再维持。系统有必要控制各事务之间的相互作用，这是通过被称为并发控制的机制来实现的。最常用的机制是各种封锁协议、时间戳排序机制、有效性检查技术与多版本机制。

一、封锁协议是一组规则，这些规则阐明了事务何时对数据库中的数据项进行加锁和解锁。

两阶段封锁协议仅在一个事务未曾释放任何数据项上的锁时才允许该事务封锁新的数据项。该协议保证可串行性，但不能避免死锁。在没有关于数据项访问方式信息的情况下，两阶段封锁协议对于保证可串行性即使必要的又是充分的。

严格两阶段封锁协议要求事务持有的所有排他锁必须在事务结束时方可释放，其目的是保证结果调度的可恢复性和无级联性；强两阶段封锁协议要求事务持有的所有锁必须在事务结束时方可释放。

基于图的封锁协议对访问数据项的顺序加以限制，从而不需要使用两阶段封锁还能保证可串行性，而且又能够保证不会产生死锁。

许多种封锁协议都不能防止死锁。一种可以防止死锁的方法是使用数据项的一种顺序，并且按与该顺序一致的次序申请加锁；另一种防止死锁的方法是使用抢占与事务回滚。为控制抢占，我们为每个事务赋予一个唯一时间戳。这些时间戳用于决定事务是等待还是回滚。如果一个事务回滚，它在重启时保持原有时间戳。

如果没有预防死锁的机制，系统必须用死锁检测与恢复机制来处理它们。为此，系统构造了一个等待图，当且仅当等待图包含环时，系统会处于死锁状态。当一个检测算法判定死锁存在时，系统通过回滚一个或多个事务来解除死锁。

某些情况下把多个数据项聚为一组，将它们作为聚集数据项来处理，其效果可能更好，这就涉及到粒度级别的问题。系统允许各种大小的数据项，并定义数据项的层次，其中小数据项、嵌套于大数据项之中。这种层次结构可以图形化地表示为树。封锁按从根结点到叶节点的顺序进行；解锁则按从叶节点到根节点的顺序进行。但这种协议虽然可以保证可串行性，但不能避免死锁。

二、时间戳排序机制通过事先在每对事务之间选择一个顺序来保证可串行性。系统中的每个事务对应一个唯一的时间戳，这个时间戳决定了事务的可串行化顺序。如果事务Ti的时间戳小于事务Tj的时间戳，则该机制保证产生的调度等价于事务Ti出现在事务Tj之前的串行调度。该机制通过回滚违反该次序的食物来保证这一点。

1. 多版本并发控制机制基于在每个事务写数据项时为该数据项创建一个新版本。读操作发出时，系统选择其中的一个版本进行读取。利用时间戳，并发控制机制保证按确保可串行性的方式选择要读取的版本。