第 16 讲:两相锁定

15-445/645 数据库系统(2022 年秋季)

https://15445.courses.cs.cmu.edu/fall2022/

卡内基梅隆大学安迪·帕夫洛

1 交易锁

DBMS 使用*锁*动态地为可序列化的事务生成执行计划，而无需提前知道每个事务的读/写设置。当有多个读和写时，这些锁在并发访问期间保护数据库对象。DBMS 包含一个集中的*锁管理器*，它决定事务是否可以获取锁。它还提供了一个关于系统内部正在发生什么的全局视图。

锁有两种基本类型:

- **共享锁(S-LOCK):**一种允许多个事务同时读取同一个对象的共享锁。如果一个事务持有一个共享锁，那么另一个事务也可以获得相同的共享锁。
- **排他锁(Exclusive Lock, X-LOCK):**排他锁允许事务修改对象。该锁防止其他事务获取该对象上的任何其他锁(S-LOCK 或 X-LOCK)。一次只有一个事务可以持有排他锁。

事务必须从锁管理器请求锁(或升级)。锁管理器根据其他事务当前持有的锁授予或阻塞请求。当事务不再需要锁来释放对象时，必须释放锁。锁管理器用哪些事务持有哪些锁以及哪些事务正在等待获取锁的信息更新其内部锁表。

DBMS 的锁表不需要是持久的，因为当 DBMS 崩溃时，任何活动(即仍在运行)的事务都会自动中止。

仅仅使用锁并不能自动解决与并发事务相关的所有问题。锁需要由并发控制协议来补充。

2 两相锁定

两相锁定(Two-Phase locking, 2PL)是一种悲观并发控制协议，它使用锁来确定是否允许事务动态访问数据库中的对象。该协议不需要提前知道事务将执行的所有查询。

阶段#1—增长:在增长阶段，每个事务向 DBMS 的锁管理器请求它所需的锁。锁管理器授予/拒绝这些锁请求。

阶段 2 -收缩:事务在释放第一个锁后立即进入收缩阶段。在收缩阶段，事务只允许释放锁。不允许他们获取新的锁。

2PL 本身就足以保证冲突的可序列化性。它生成的调度的优先级图是无环的。但是它很容易受到*级联中止*的影响，即当一个事务中止时，现在必须回滚另一个事务，这导致了工作的浪费。

2PL 仍然可能有脏读，也可能导致死锁。还有一些可能的调度是可序列化的，但不被 2PL 所允许(锁可以限制并发性)。

强严格的两阶段锁

如果一个事务写入的任何值永远不会被另一个事务读取或覆盖，直到第一个事务提交，则调度是*严格*的。*强严格 2PL*(也称为*严格 2PL*)是 2PL 的一种变体，其中事务仅在提交时释放锁。

这种方法的优点是 DBMS 不会导致级联中止。DBMS 还可以通过恢复修改元组的原始值来逆转中断的事务的更改。然而，严格的 2PL 会产生更谨慎/悲观的调度，从而限制并发性。

时间表的宇宙

系列化时间表⊂StrongStrict2PL 冲突系列化时间表⊂V viewserializable 时间表⊂所有时间表

3 死锁处理

*死锁*是一个事务等待锁被彼此释放的循环。在 2PL 中有两种处理死锁的方法:检测和预防。

方法#1:死锁检测

为了检测死锁，DBMS 创建一个*等待*图，其中事务是节点，如果事务 Ti 正在等待事务 Tj 释放锁，则存在从 Ti 到 Tj 的有向边。系统将定期检查等待图中的循环(通常带有后台线程)，然后做出如何打破它的决定。构造图时不需要锁存器，因为如果 DBMS 在一次传递中错过了死锁，它将在随后的传递中找到它。请注意，在死锁检查的频率(使用 cpu 周期)和死锁被打破的等待时间之间有一个折衷。

当 DBMS 检测到死锁时，它将选择一个“受害者”事务来中止以打破这个循环。受害者事务将重新启动或中止，这取决于应用程序如何调用它。

当选择一个受害者来打破死锁时，DBMS 可以考虑多个事务属性:

1.按年龄(最新或最旧的时间戳)。

2.按进度(最少/最多执行的查询)。

3.根据已经锁定的物品的#。

4.通过需要回滚的事务#。

5.#过去重新启动事务的次数(以避免饥饿)。

没有一种选择比其他选择更好。许多系统结合使用这些因素。

在选择要中止的受害事务之后，DBMS 还可以决定回滚事务更改的程度。它既可以回滚整个事务，也可以回滚足以打破死锁的查询。

方法#2:预防死锁

死锁预防 2PL 不是让事务尝试获取它们需要的任何锁，然后在之后处理死锁，而是在事务发生之前阻止它们造成死锁。当一个事务试图获取另一个事务持有的锁(这可能导致死锁)时，DBMS 会杀死其中一个事务。为了实现这一点，事务根据时间戳被分配优先级(旧的事务有

更高的优先级)。这些方案保证没有死锁，因为在等待锁时只允许一种类型的方向。当事务重新启动时，DBMS 重用相同的时间戳。

在防止死锁的情况下，有两种方法杀死事务。

- **Wait-Die(“Old Waits for Young”):**如果请求事务的优先级高于持有事务，则等待。否则，它会中止。
- **Wound-Wait(“年轻的等待年老的”):**如果请求事务的优先级高于持有事务，持有事务将终止并释放锁。否则，请求交易等待。

4 .锁粒度

如果一个事务想要更新 10 亿个元组，它必须向 DBMS 的锁管理器请求 10 亿个锁。这将是缓慢的，因为事务在获取/释放锁时必须在锁管理器的内部锁表数据结构中进行锁存。

为了避免这种开销，DBMS 可以使用一种锁层次结构，这种结构允许事务在系统中使用更粗粒度的锁。例如，它可以用 10 亿个元组来获取表上的单个锁，而不是 10 亿个单独的锁。当一个事务在这个层次结构中获取了一个对象的锁时，它隐式地获取了其所有子对象的锁。

数据库锁层次结构:

1.数据库级别(略罕见)

2.表级(非常常见)

3.页面级别(普通)

4.元组级别(非常常见)

5.属性等级(稀有)

意图锁允许在共享模式或独占模式下锁定更高级别的节点，而无需检查所有后代节点。如果一个节点处于意图模式，那么显式的锁定是在树的较低级别上进行的。

- **IS (Intention-Shared):**表示在较低级别的显式锁定与共享锁。
- **Intention-Exclusive (IX):**表示在较低级别显式锁定，具有独占或共享锁。
- **共享+Intention-Exclusive(SIX):**在该节点上的子树在共享模式下显式锁定，在较低级别上使用排他模式锁进行显式锁定。