数据库系统概论
An Introduction to Database System

第十一章 并发控制

中国人民大学信息学院
陈红

第十一章 并发控制

11.2 封锁

- 什么是封锁
- 基本封锁类型
- 锁的相容矩阵

什么是封锁

- 封锁就是事务 T 在对某个数据对象（例如表、记录等）操作之前，先向系统发出请求，对其加锁
- 加锁后事务 T 就对该数据对象有了一定的控制，在事务 T 释放它的锁之前，其它的事务不能更新此数据对象。
- 封锁是实现并发控制的一个非常重要的技术

基本封锁类型

- 一个事务对某个数据对象加锁后究竟拥有什么样的控制由封锁的类型决定。
- 基本封锁类型
  - 排它锁（Exclusive Locks，简记为 X 锁）
  - 共享锁（Share Locks，简记为 S 锁）

排它锁

- 排它锁又称为写锁
- 若事务 T 对数据对象 A 加上 X 锁，则只允许 T 读取和修改 A，其它任何事务都不能再对 A 加任何类型的锁，直到 T 释放 A 上的锁
- 保证其他事务在 T 释放 A 上的锁之前不能再读取和修改 A

共享锁

- 共享锁又称为读锁
- 若事务 T 对数据对象 A 加上 S 锁，则事务 T 可以读 A 但不能修改 A，其它事务只能再对 A 加 S 锁，而不能加 X 锁，直到 T 释放 A 上的 S 锁
- 保证其他事务可以读 A，但在 T 释放 A 上的 S 锁之前不能对 A 做任何修改

锁的相容矩阵

T_1 \ T_2	X	S	-
X	N	N	Y
S	N	Y	Y
-	Y	Y	Y

Y=Yes，相容的请求
N=No，不相容的请求

锁的相容矩阵（续）

在锁的相容矩阵中：

- 最左边一列表示事务 T1 已经获得的数据对象上的锁的类型，其中横线表示没有加锁。
- 最上面一行表示另一事务 T2 对同一数据对象发出的封锁请求。
- T2 的封锁请求能否被满足用矩阵中的 Y 和 N 表示
  - Y 表示事务 T2 的封锁要求与 T1 已持有的锁相容，封锁请求可以满足
  - N 表示 T2 的封锁请求与 T1 已持有的锁冲突， T2 的请求被拒绝

使用封锁机制解决丢失修改问题

例：

	T_1	T_2
①	Xlock A	
②	R(A)=16	
		Xlock A
③	A←A-1	□□
	W(A)=15	□□
	Commit	□□
	Unlock A	□□
④		□□ Xlock A
		R(A)=15
		A←A-1
⑤		W(A)=14
		Commit
		Unlock A

没有丢失修改

- 事务 T1 在读 A 进行修改之前先对 A 加 X 锁
- 当 T2 再请求对 A 加 X 锁时被拒绝
- T2 只能等待 T1 释放 A 上的锁后 T2 获得对 A 的 X 锁
- 这时 T2 读到的 A 已经是 T1 更新过的值 15
- T2 按此新的 A 值进行运算，并将结果值 A=14 送回到磁盘。避免了丢失 T1 的更新。

使用封锁机制解决不可重复读问题

T_1	T_2
① Slock A	
Slock B	
R(A)=50	
R(B)=100	
□□ =150	
②	Xlock B
	□□
③ R(A)=50	□□
R(B)=100	□□
□□ =150	□□
Commit	□□
Unlock A	□□
Unlock B	□□
④	□□ XlockB
	R(B)=100
	B←B*2
⑤	W(B)=200
	Commit
	Unlock B

可重复读

- 事务 T1 在读 A， B 之前，先对 A， B 加 S 锁
- 其他事务只能再对 A， B 加 S 锁，而不能加 X 锁，即其他事务只能读 A， B，而不能修改
- 当 T2 为修改 B 而申请对 B 的 X 锁时被拒绝只能等待 T1 释放 B 上的锁
- T1 为验算再读 A， B，这时读出的 B 仍是 100，求和结果仍为 150，即可重复读
- T1 结束才释放 A， B 上的 S 锁。T2 才获得对 B 的 X 锁

使用封锁机制解决读“脏”数据问题

例

T_1	T_2
① Xlock C	
R(C)=100	
C←C*2	
W(C)=200	
②	Slock C
	等等
③ROLLBACK	等等
(C 等等等 100)	等等
Unlock C	等等
④	等等 Slock C
	R(C)=100
⑤	Commit C
	Unlock C

不读“脏”数据

- 事务 T1 在对 C 进行修改之前，先对 C 加 X 锁，修改其值后写回磁盘
- T2 请求在 C 上加 S 锁，因 T1 已在 C 上加了 X 锁， T2 只能等待
- T1 因某种原因被撤销， C 恢复为原值 100
- T1 释放 C 上的 X 锁后 T2 获得 C 上的 S 锁，读 C=100 。避免了 T2 读“脏”数据