第十一章 并发控制

11.1 并发控制概述
11.2 封锁
11.3 封锁协议
11.4 活锁和死锁
11.5 并发调度的可串行性
11.6 两段锁协议
11.7 封锁的粒度
*11.8 其他并发控制机制
11.9 小结

11.6 两段锁协议

- 数据库管理系统普遍采用两段锁协议的方法实现并发调度的可串行性，从而保证调度的正确性

- 两段锁协议

  指所有事务必须分两个阶段对数据项加锁和解锁

  - 在对任何数据进行读、写操作之前，事务首先要获得对该数据的封锁
  - 在释放一个封锁之后，事务不再申请和获得任何其他封锁

两段锁协议（续）

- “两段”锁的含义

  事务分为两个阶段

  - 第一阶段是获得封锁，也称为扩展阶段
    - 事务可以申请获得任何数据项上的任何类型的锁，但是不能释放任何锁
  - 第二阶段是释放封锁，也称为收缩阶段
    - 事务可以释放任何数据项上的任何类型的锁，但是不能再申请任何锁

两段锁协议（续）

例

事务T_i遵守两段锁协议，其封锁序列是：

Slock A　Slock B　Xlock C　Unlock B　Unlock A　Unlock C；

|←　　扩展阶段　　→|

事务T_j不遵守两段锁协议，其封锁序列是：

Slock A　Unlock A　Slock B　Xlock C　Unlock C　Unlock B；

两段锁协议（续）

例

事务T_i遵守两段锁协议，其封锁序列是：

Slock A　Slock B　Xlock C　Unlock B　Unlock A　Unlock C；

|←　　收缩阶段　　→|

事务T_j不遵守两段锁协议，其封锁序列是：

Slock A　Unlock A　Slock B　Xlock C　Unlock C　Unlock B；

两段锁协议（续）

例

事务T_i遵守两段锁协议，其封锁序列是：

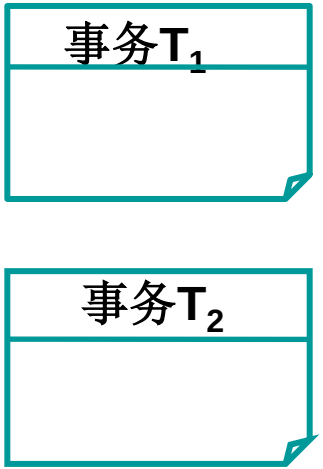
Slock A　Slock B　Xlock C　Unlock B　Unlock A　Unlock C；

事务T_j不遵守两段锁协议，其封锁序列是：

Slock A　Unlock A　Slock B　Xlock C　Unlock C　Unlock B；

两段锁协议（续）

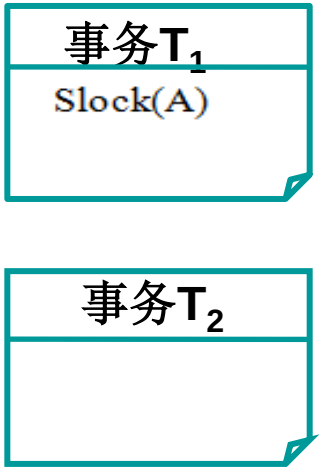
事务T_1	事务T_2



两段锁协议（续）

事务T_1	事务T_2
Slock(A)	
R(A)=260	

扩展阶段

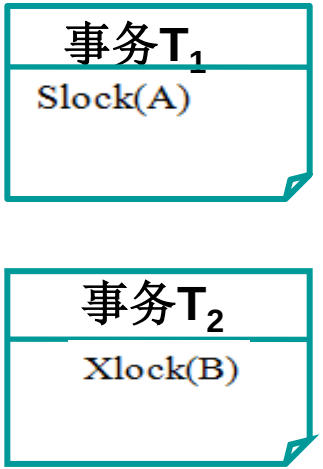


两段锁协议（续）

事务T_1	事务T_2
Slock(A)	
R(A)=260	
	Xlock(B)
	W(B)=300

扩展阶段

扩展阶段



两段锁协议（续）

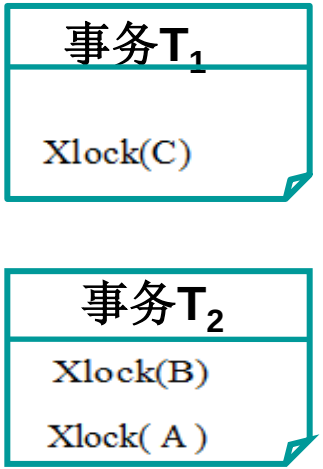
事务T_1	事务T_2
Slock(A)	
R(A)=260	
	Xlock(B)
	W(B)=300
扩展阶段 Xlock(C)	
W(C)=160	
	Xlock(A)
	等待

两段锁协议（续）

事务T_1	事务T_2
Slock(A)	
R(A)=260	
	Xlock(B)
	W(B)=300
Xlock(C)	
W(C)=160	
	Xlock(A)
	等待
Unlock(A)	
	等待
	W(A)=250

扩展阶段

收缩阶段



扩展阶段

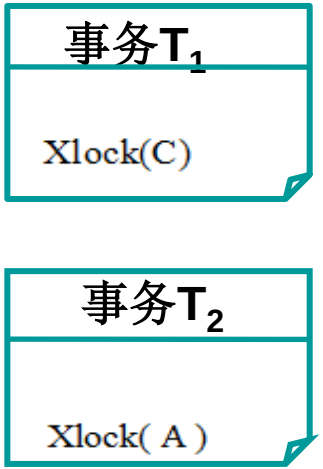
收缩阶段

两段锁协议（续）

事务T_1	事务T_2
Slock(A)	
R(A)=260	
	Xlock(B)
	W(B)=300
扩展阶段 Xlock(C)	
W(C)=160	
	Xlock(A)
	等待
Unlock(A)	等待
	W(A)=250
收缩阶段	Unlock(B)

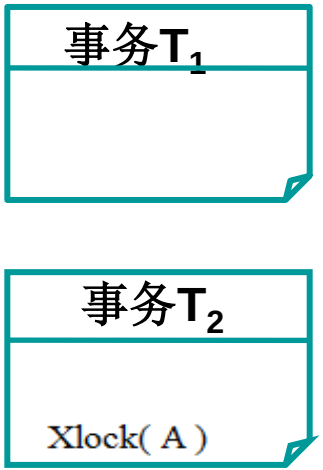
扩展阶段

收缩阶段



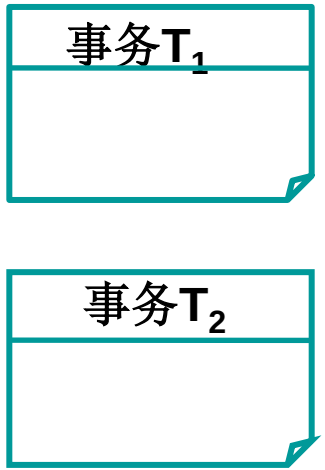
两段锁协议（续）

事务T_1	事务T_2
Slock(A)	
R(A)=260	
	Xlock(B)
	W(B)=300
扩展阶段 Xlock(C)	
W(C)=160	
	Xlock(A)
	等待
Unlock(A)	等待
	W(A)=250
收缩阶段	Unlock(B)
Unlock(C)	



两段锁协议（续）

	事务T_1	事务T_2
扩展阶段	Slock(A)	
	R(A)=260	
		Xlock(B)
		W(B)=300
	Xlock(C)	
	W(C)=160	
		Xlock(A)
		等待
收缩阶段	Unlock(A)	等待
		W(A)=250
		Unlock(B)
	Unlock(C)	
		Unlock(A)



- 遵守两段锁协议，是一个可串行化调度。
- 如何验证？

两段锁协议（续）

事务T_1	事务T_2
Slock(A)	
R(A)=260	
	Xlock(B)
	W(B)=300
Xlock(C)	
W(C)=160	
	Xlock(A)
	等待
Unlock(A)	等待
	W(A)=250
	Unlock(B)
Unlock(C)	
	Unlock(A)

$L_1=R_1(A)W_2(B)W_1(C)W_2(A)$

$L_S=R_1(A)\ W_1(C)\ W_2(B)W_2(A)$

两段锁协议（续）

- 事务遵守两段锁协议是可串行化调度的充分条件，而不是必要条件。
- 若并发事务都遵守两段锁协议，则对这些事务的任何并发调度策略都是可串行化的
- 若并发事务的一个调度是可串行化的，不一定所有事务都符合两段锁协议

两段锁协议（续）

T_1	T_2
Slock B	
Y=R(B)=2	
Unlock B	
Xlock A	
	Slock A
A=Y+1=3	等待
W(A)	等待
Unlock A	等待
	X=R(A)=3
	Unlock A
	Xlock B
	B=X+1=4
	W(B)
	Unlock B

- 对**T1**和**T2**的调度没有遵守两段锁协议
- 但是它是可串行化的

$$L_1=R_1(B)W_1(A)R_2(A)W_2(B)$$

两段锁协议（续）

T_1	T_2
Slock B	
Y=R(B)=2	
	Slock A
	X=R(A)=2
Unlock B	
	Unlock A
Xlock A	
A=Y+1=3	
W(A)	
	Xlock B
	B=X+1=3
	W(B)
Unlock A	
	Unlock B

- 对T1和T2的调度没有遵守两段锁协议
- 它不是可串行化的

- 并发事务遵守两段锁协议，对这些事务的任何并发调度策略都是可串行化的；
- 不遵守两段锁协议，对这些事务的并发调度策略可能是可串行化的，也可能不是可串行化的。

两段锁协议（续）

- 两段锁协议与防止死锁的一次封锁法
  - 一次封锁法要求每个事务必须一次将所有要使用的数据全部加锁，否则就不能继续执行，因此一次封锁法遵守两段锁协议
  - 但是两段锁协议并不要求事务必须一次将所有要使用的数据全部加锁，因此遵守两段锁协议的事务可能发生死锁

两段锁协议（续）

[例] 遵守两段锁协议的事务可能发生死锁

事务T_1	事务T_2
Slock B	
R(B)=2	
	Slock A
	R(A)=2
Xlock A	
等待	Xlock B
等待	等待

死锁

小结

- 两段锁协议
  - 第一阶段是获得封锁，也称为扩展阶段
    - 事务可以申请获得任何数据项上的任何类型的锁，但是不能释放任何锁
  - 第二阶段是释放封锁，也称为收缩阶段
    - 事务可以释放任何数据项上的任何类型的锁，但是不能再申请任何锁

思考题

1. 遵守两段锁协议是否遵守三级封锁协议？
2. 遵守三级封锁协议是否遵守两段锁协议？