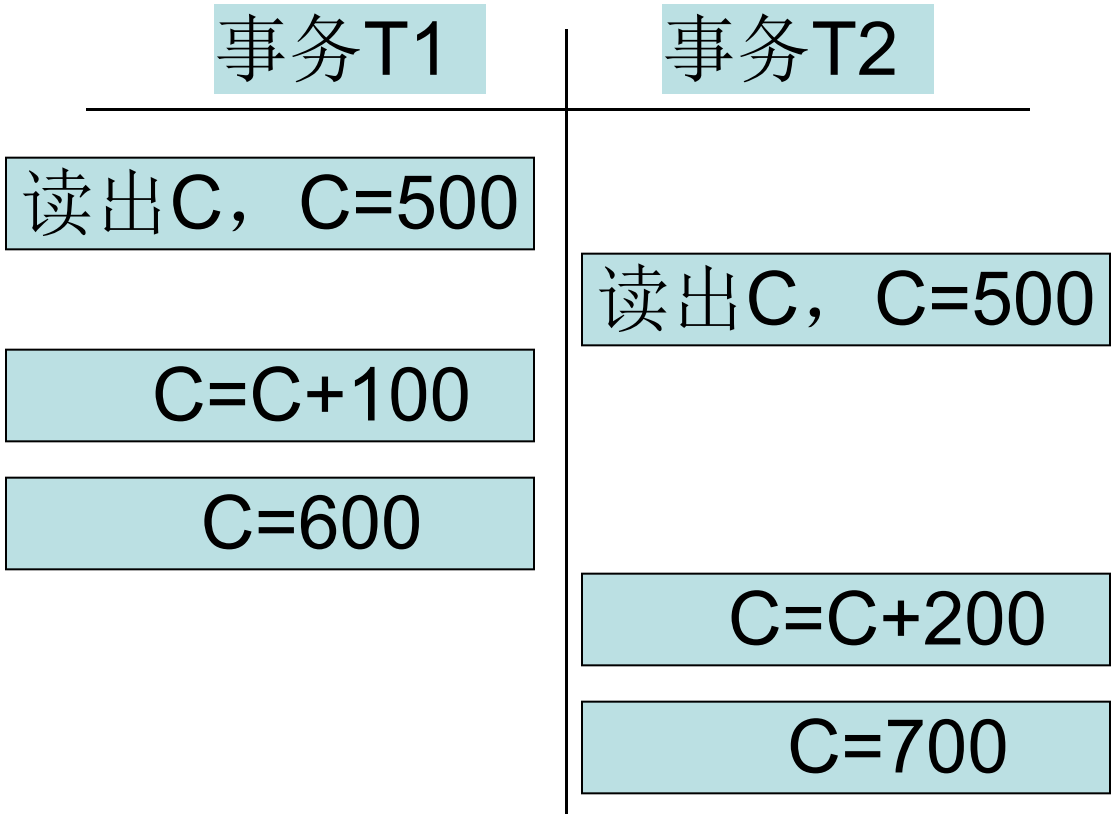
并发控制的任务

1. 对并发操作进行正确调度
2. 保证事务的隔离性
3. 保证事务的一致性

思考：事务的原子性和持久性由DBMS的哪个子系统进行维护？

6.3.1 并发操作引发的问题

丢失修改



事务T1	事务T2
读出A=50	
读出B=100	
求和=150	
	读出B=100
	B=B*2
	WRITE（B）
读出A=50	
读出B=200	
求和=250	
（验算不对）	

不可重复读

读脏数据

事务T1	事务T2
读出 C=100	
C=C*2	
WRITE（C）	
	读出 C=200
ROLLBACK	
C恢复为100	

6.3.2、调度及其可串行化

1、事务的表示方法：

$R_i(X)$表示事务T_i的读X操作；

$W_i(X)$表示事务T_i的写X操作。

例：

事务T1(Read(B)； A=B+1； write(A))，

事务T2(Read(A)； B=A+1； write(B))

可以表示成：

T_1： $R_1(B)\,W_1(A)$　　T_2： $R_2(A)\,W_2(B)$

2、冲突操作

定义：如果两个操作来自不同的事务，它们对同一数据单位进行操作，并且其中至少有一个是写操作，则称这两个操作是相互冲突的或冲突操作。

例：事务T_0：$W_0(X)W_0(Y)W_0(Z)$

事务T_1：$R_1(X)R_1(Z)W_1(X)$

则在这两个事务中有冲突操作：

$R_1(X)$与$W_0(X)$　$W_1(X)$与$W_0(X)$　$R_1(Z)$与$W_0(Z)$

3、调度： 事务执行的次序。

设τ={T_1，T_2，…T_n}是一事务集，τ的一个调度S是一拟序集（∑,<s)

其中：1) ∑说明S执行的操作正是T_1，T_2，…T_n的操作。

2) <s 说明调度S遵守每个事务的操作的内部执行次序

3) 每对冲突操作的执行次序由S决定。

4 串行调度

定义：如果调度S中的任意两个事务Ti和Tj，如果Ti的所有操作都先于Tj的所有操作，或者相反，则称S为串行调度。

串行调度事务执行的结果总是正确的
串行调度不能够充分利用系统资源

5、并发调度：

如果在一个调度中，各个事务交叉地执行，这个调度称为并发调度。

6、可串行化的调度：

如果一个事务集的并发调度与某一串行调度是等价的，则称该并发调度是可串行化的 。

可串行化是作为并发调度正确与否的判定准则。

A=10
B=10

串行调度1

T1	T2
Read（A）	
A:=A-5	
Write（A）	
	A=5 B=10
Read（B）	
B:=B+5	
	Read(B)
	B:=B-5
Write(B)	
	Write(B)

串行调度2

T1	T2
	Read(B)
	B:=B-5
	Write(B)
Read（A）	
A:=A-5	
Write（A）	
	A=5 B=10
Read（B）	
B:=B+5	
Write(B)	

并发调度1

T1	T2
Read(A)	
	Read(B)
A:=A-5	
	B:=B-5
Write(A)	
	Write(B)
	A=5 B=10
Read(B)	
B:=B+5	
Write(B)	

并发调度2

T1	T2
Read(A)	
A:=A-5	
	A=5 B=15
Write(A)	
	Read(B)
Read(B)	
	B:=B-5
B:=B+5	
	Write(B)
Write(B)	

7、冲突可串行化定理

一个调度Sc在保证冲突操作的次序不变得情况下，通过交换两个事务不冲突操作的次序得到另一个调度Sc’，如果Sc’是串行的称调度Sc为冲突可串行化调度。

一个调度是冲突可串行化的，一定是可串行化调度，反之则不成立！

例题

调度Sc1=r1(A)w1(A)r2(A)w2(A)r1(B)w1(B)r2(B)w2(B)

判断Sc1是否是冲突可串行化的调度。

把r2(A) w2(A)和r1(B)w1(B)交换，得到

Sc2=r1(A)w1(A) r1(B)w1(B)r2(A)w2(A)r2(B)w2(B)

Sc2等价于串行调度T1T2，所以Sc1是冲突可串行化的

冲突可串行化的判定方法：前驱图

设S是若干事务{T1,T2,…,Tn}的一个调度，S的前驱图G（V，E）是一个有向图，其构成规则如下：

1）V是由所有参加调度的事务构成的节点

2）E是图中的一条有向边，如果O_i和O_j是冲突操作，且O_i先于O_j执行，则在图中有一条边$T_i \rightarrow T_j$。

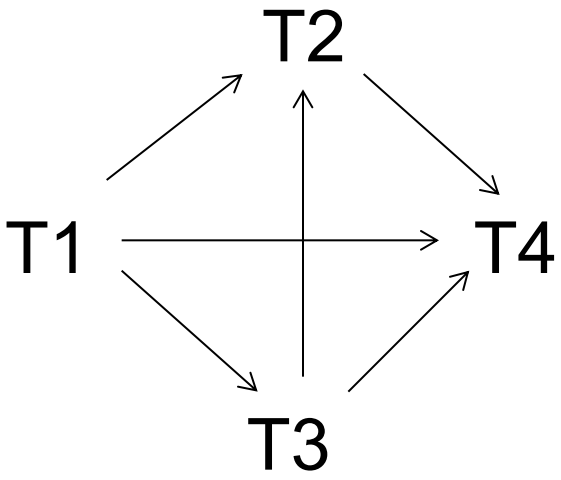
- 事务Ti读x在事务Tj写x之前
- 事务Ti写x在事务Tj读x之前
- 事务Ti写x在事务Tj写x之前

无环的前趋图的调度是调度可串行化的

例题

T1	T2	T3	T4
Read(x)		Write(y)	
	Read(y)	Write(x)	
	Write(x)	Write(z)	
			Read(z)
			Write(x)

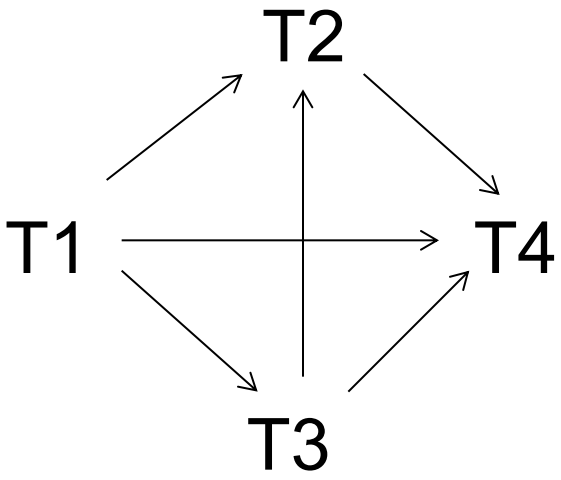
一个并发调度**S**



获得调度S的一个等价的串行调度

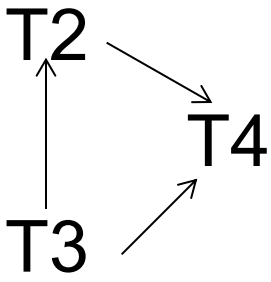
由于图中无回路，必有一个节点无入弧，将这个节点及其相连的弧删去，并把该节点存入先进先出的队列中。对剩下的图做同样的处理，直至所有节点移入队列中。按照队列中次序串行安排各事务的执行，就可以得到一个等价的串行调度

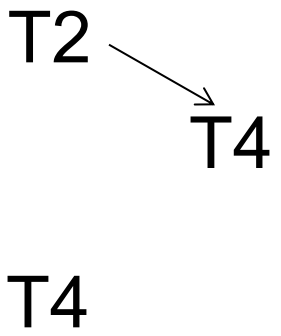
T1	T2	T3	T4
Read(x)		Write(y)	
	Read(y)	Write(x)	
	Write(x)	Write(z)	Read(z)
			Write(x)



一个并发调度**S**

T1 → T3 → T2 → T4





习题1-采用可串行化调度判断方法

今有3个事务的一个调度

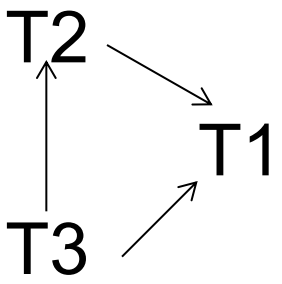
r3(B) r1(A)w3(B)r2(B)r2(A)w2(B)r1(B)w1(A)

该调度是冲突可串行化的么？

（1）找出冲突操作

r3(B)和w2(B)	T3→T2
w3(B)和r2(B)	T3→T2
w3(B)和w2(B)	T3→T2
w3(B)和r1(B)	T3→T1
r2(A)和w1(A)	T2→T1
w2(B)和r1(B)	T2→T1

（2）画出前驱图



（3）给出等价的串行调度

T3 ⟶T2⟶T1

习题1-采用冲突可串行化方法

今有3个事务的一个调度

R3(B) r1(A)w3(B)r2(B)r2(A)w2(B)r1(B)w1(A)

该调度是冲突可串行化的么？

交换r1(A)和w3(B)r2(B)r2(A)w2(B)

R3(B)w3(B)r2(B)r2(A)w2(B) r1(A) r1(B)w1(A)

等价于串行调度T3T2T1

习题2

设T1，T2是如下的三个事务

T1：A：=A*B+2+D

T2：B：=A*2+D

设A的初值为2，B的初值为4：

（1）若这两个事务允许并发执行，讨论他们可能实施的调度，请一一列举并求每种调度的结果

（2）试给出一个可串行化调度，并给出执行结果

（2）可串行化调度

T1：A：=A*B+2+D

T2：B：=A*2+D

T1：R1(A) R1(B) R1(D) W1(A)

T2：R2(A) R2(D) W2(B)

R1(A) R1(B) R1(D) W1(A) R2(A) R2(D) W2(B) T1T2

R2(A) R2(D) W2(B) R1(A) R1(B) R1(D) W1(A) T2T1

R2(A) R2(D) R1(A) W2(B) R1(B) R1(D) W1(A)

6.3.4、封锁技术

封锁：是实现并发控制的一种机制。所谓封锁，就是事务T在对某个数据对象操作之前，先对其加锁。

1. 排它锁（X锁，写锁）

若事务T对数据对象A加上排它锁，则只允许T读取和修改A，不允许其他任何事务在对A加锁，直到T释放A上的X锁。

2. 共享锁（S锁，读锁）

若事务T对数据对象A加上共享锁，则事务T可以读A但不能修改A，其他事务只能对A加S锁，而不能加X锁，直到T释放A上的S锁。。

3. 封锁协议

封锁级别	加锁	放锁
一级	事务T在修改数据A之前必须先对其加X锁	事务结束才释放X锁
二级	一级封锁协议加上事务T在读取数据A之前必须对其加S锁	读完后即可释放S锁
三级	一级封锁协议加上事务T在读取数据A之前必须对其加S锁	事务结束才释放S锁

（1）一级封锁协议

T1	T2
Xlock A	
读A=16	
	Xlock A
	等待
	等待
	等待
A ←A-1	等待
写回A=15	等待
commit	Xlock A
Unlock A	读A=15
	A ←A-1
	写回A=14
	commit
	Unlock A

T1	T2
读A=16	
	读A=16
A ←A-1	
	A ←A-1
写回A=15	
	写回A=15

（a）丢失修改

T1	T2
读C=100	
C ←C*2	
写回C=200	
	读C=200
	D=C+5
	写回D=205
ROOLBACK	

（c）读脏数据

T1	T2
读A=50	
读B=100	
求和=150	
	读B=100
	B=B*2
	写回B=200
读A=50	
读B=200	
求和=250	

（b）不可重复读

解决丢失修改问题

（2）二级封锁协议

T1	T2
Xlock C	
读C=100	
C ←C*2	Slock C
写回C=200	等待
	等待
ROOLBACK	等待
Unlock C	等待
	Slock C
	读C=200
	D=C+5
	写回D=205
	commit
	Unlock C

（a）不可重复读

T1	T2
读C=100	
C ←C*2	
写回C=200	读C=200
	D=C+5
ROOLBACK	写回D=205

（c）读脏数据

T1	T2
读A=50	
读B=100	
求和=150	读B=100
	B=B*2
读A=50	写回B=200
读B=200	
求和=250	

（b）不可重复读

解决丢失修改和读脏数据问题

（3）三级封锁协议

T1	T2
Slock A	
Slock B	
读A=50	Xlock B
读B=100	等待
求和=150	等待
	等待
	等待
读A=50	等待
读B=100	等待
求和=150	等待
Ulock B	Xlock B
Ulock A	读B=100
	B=B*2
	写回B=200
	commit
	Unlock B

T1	T2
读A=50	
读B=100	
求和=150	读B=100
	B=B*2
	写回B=200
读A=50	
读B=200	
求和=250	

（b）不可重复读

解决三种并发操作引起的问题

级别 \ 操作	X 锁		S锁		一致性保证		
	操作结束释放	事务结束释放	操作结束释放	事务结束释放	不丢失修改	不读脏数据	可重复读
一级封锁协议		√			√		
二级封锁协议		√	√		√	√	
三级封锁协议		√		√	√	√	√

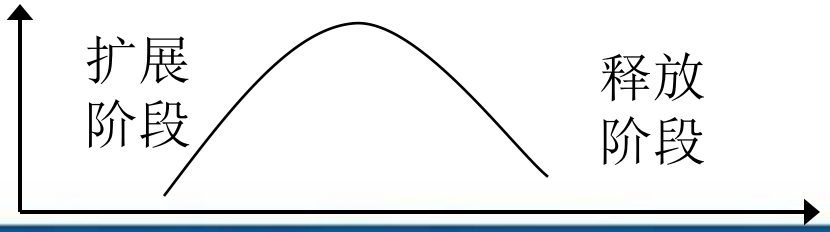
(4) 两阶段锁协议

两阶段协议（Two- Phase Locking Protocol，2PL协议）

- 某一事务在对数据进行读、写之前，先要申请并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它封锁。

T1： Lock(A) Lock(B) Lock(C) Unlock(B)Ulock(C) Ulock (A) √

T1： Lock(A) ULock(A) Lock(B) lock(C)Ulock(C) Ulock (B) ×



任何一个遵从2PL协议的调度都是可串行化的。

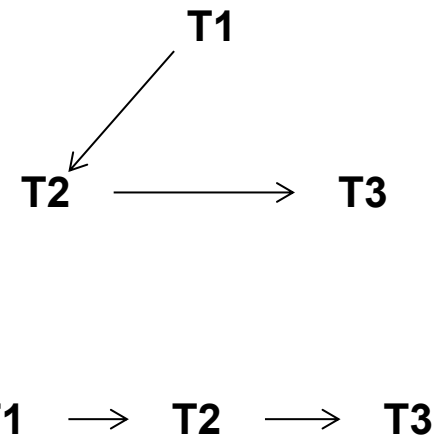
说明：事务遵守2PL协议是可串行化调度的充分条件，而不是必要条件。

T1：Slock(X) ,Read(X), Ulock(X)

T2: Slock(Y),Read(Y),Unlock(Y),Xlock(X),Write(X),Unlock(X)

T3: Xlock(Y),Write(Y),Unlock(Y)

T1	T2	T3
Slock(X)	Slock(Y)	
Read(X)	Read(Y)	
Ulock(X)	Ulock(Y)	
		Slock(Y)
		Read(Y)
		Ulock(Y)
	Slock(X)	
	Read(X)	
	Ulock(X)	



不遵守2PL协议的可串行化调度

习题2

T1：A：=A*B+2
T2：B：=A*2
设A的初值为2，B的初值为4

（1）所有的串行调度

T1T2 ⟶ A=10，B=20

T2T1 ⟶ A=10，B=4

运用2PL协议生成一个可串行化的调度

6.3.5 死锁与活锁

1. **死锁** ：事务T1已经封锁A，而又想申请封锁B，而此时事务T2已经封锁B，而又想申请封锁A，这样，T1等待T2释放B，而T2等待T1释放A，使得T1、T2均无法继续执行下去，这种情况称为死锁。

LOCK A …	LOCK B …
LOCK B …	LOCK A …
等待	等待

- *判断并解除死锁*
  * 超时法：事务的等待超过了规定的时限
  * 等待图法：检测等待图中是否有回路存在。

2. **活锁：**事务T1，T2申请数据对象A，T1先给A加锁，T1释放A上的锁后，事务T3又给A加锁，T2等待，这样，A始终被其他事务封锁，事务T2可能长时间得不到A，这种情况称为活锁。

避免活锁的方法：
采用先来先服务的原则。