并发操作的基本概念

并发操作的基本概念

总述

数据库系统是一个多用户系统，允许用户并发操作数据库。本知识点学习数据库并发操作的基本概念。

并发操作的基本概念

一、并发操作

二、并发控制

并发操作的基本概念

一、并发操作

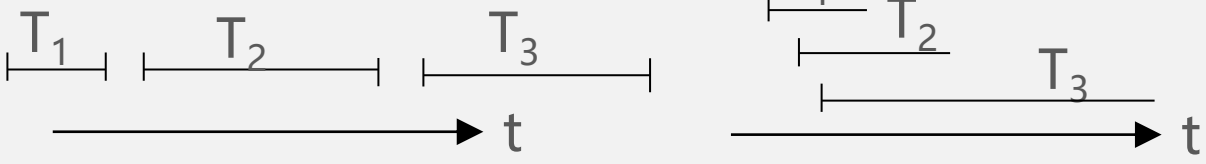
数据库是共享资源，允许多个用户使用。在多用户共享系统中，如果多个用户同时对同一数据进行操作，称为并发操作。

并发是指多个事务利用分时的技术，在同一CPU上分段执行。

并发操作的基本概念

一、并发操作

并发操作可以改善系统的资源利用率和吞吐率，改善短事务的响应时间。



串行访问

并发访问

并发操作的基本概念

一、并发操作

当用户并发操作数据库时，如果不加以控制可能会带来数据的不一致问题。

例如，飞机订票系统中的一个活动序列：

（1）甲售票点读出某航班的机票余额A，设A＝16；

（2）乙售票点读出同一航班的机票余额A，也为16；

（3）甲售票点卖出一张机票，修改余额A←A-1， A为15，把A写回数据库；

（4）乙售票点也卖出一张机票，修改余额A←A-1，A为15，把A写回数据库。

结果，卖出两张机票而余额只减少1。

二、并发控制

为避免并发操作造成数据的不一致性，DBMS就要用正确的方式调度并发操作，使一个用户事务的执行不受其它事务的干扰，称为并发控制。

数据库并发控制的基本单位是事务。事务具有原子性、一致性、隔离性和永久性。

DBMS并发控制子系统用于保证事务的隔离性。

二、并发控制

在大多数商用DBMS中，并发控制的主要方法是采用封锁机制。

不同的DBMS提供的封锁类型、封锁协议、达到的系统一致性级别不尽相同，但是其依据的基本原理和技术是共同的。

并发操作的基本概念

练习

为了防止一个用户的工作不适当地影响另一个用户，应该采取（　　）。

A. 完整性控制

B. 访问控制

C. 安全性控制

D. 并发控制

并发操作的基本概念

解答

为了防止一个用户的工作不适当地影响另一个用户，应该采取（ D ）。

A. 完整性控制

B. 访问控制

C. 安全性控制

D. 并发控制

小结

数据库管理系统必须提供并发控制机制来协调并发用户的并发操作，以保证并发事务的隔离性和一致性，进而保证数据库的一致性。

数据并发程度取决于数据库中的并发控制机制。并发控制机制是衡量一个数据库管理系统性能的重要标志之一。


谢谢！


并发操作带来的不一致问题

并发操作带来的不一致问题

总述

数据库系统是一个多用户系统，允许用户并发操作。本知识点学习并发操作可能带来的数据不一致问题，了解并发控制的必要性。

并发操作带来的不一致问题

一、丢失更新

二、不一致分析

三、读“脏”数据

并发操作带来的不一致问题

一、丢失更新

两个事务T_1和T_2读入同一数据并修改，T_2提交的结果破坏了T_1提交的结果，T_1的修改被丢失。

例如，飞机订票系统中的一个活动序列：

（1）甲售票点读出某航班的机票余额A，设A＝16；

（2）乙售票点读出同一航班的机票余额A，也为16；

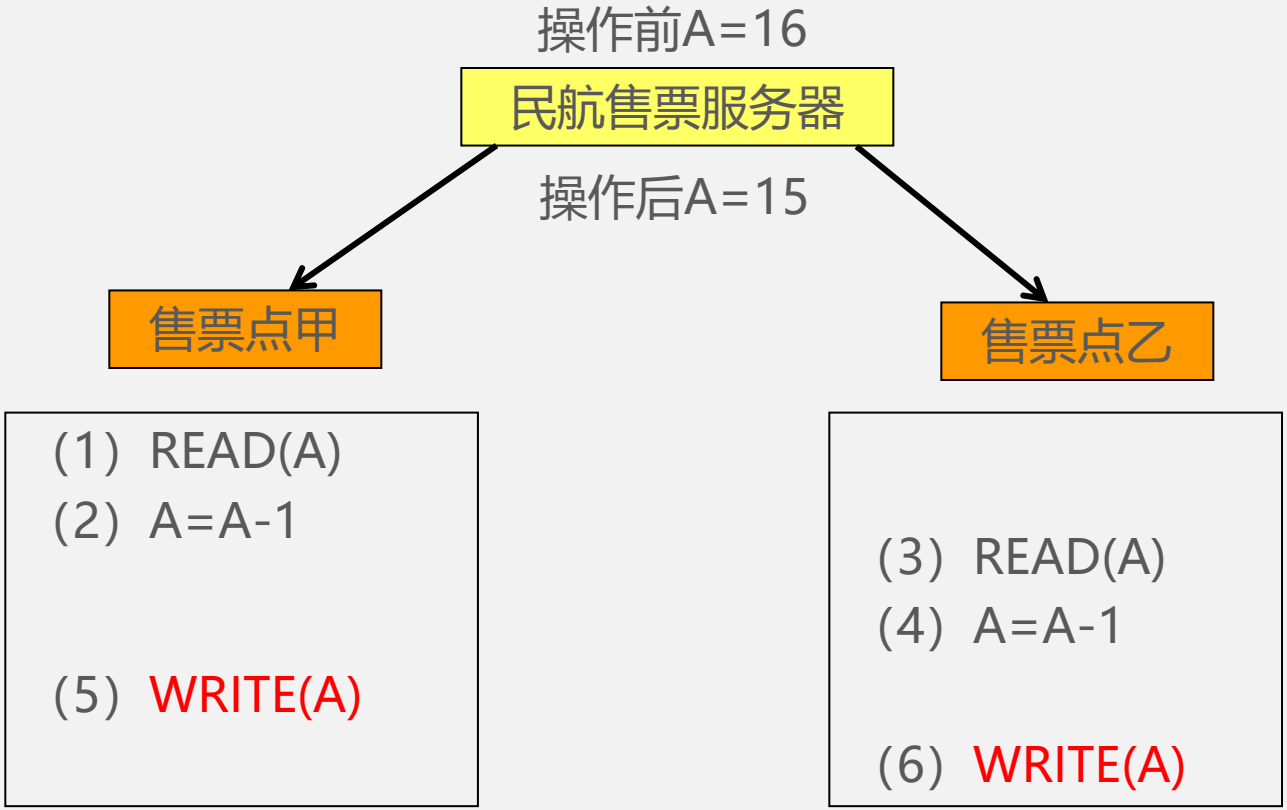
（3）甲售票点卖出一张机票，修改余额A←A-1， A为15，把A写回数据库；

（4）乙售票点也卖出一张机票，修改余额A←A-1，A为15，把A写回数据库。

结果，卖出两张机票而余额只减少1。

并发操作带来的不一致问题

一、丢失更新



并发操作带来的不一致问题

一、丢失更新

A = 16

甲	乙
READ(A)(A = 16) A=A-1	
	READ(A)(A = 16) A=A-1
WRITE(A)	
	WRITE(A)

A = 15，丢失了更新

二、不一致分析(不可重复读)

事务T_1读取某一数据，事务T_2读取并修改了同一数据，T_1为了对所读取的数据进行校对再读此数据，得到了不同的结果。

例如T_1读取B = 100，T_2读取B并把B改为200，T_1再读B得200，与第一次读取的数值不一致，T_1读了过时的数据。

二、不一致分析

例如： B = 100

T_1	T_2
READ(B) B = 100	
	B = 200 WRITE(B)
READ(B) B = 200	

T_1读了过时的数据

三、读“脏”数据

事务T_1修改某一数据，事务T_2读取同一数据，而T_1由于某种原因被ROLLBACK撤消，则T_2读到的数据就为“脏”数据，即不正确的数据。

例如T_1把C由100改为200，T_2读到C为200，而由于事务T_1被撤消，其修改宣布无效，C恢复为原值100，而T_2却读到了C为200，与数据库内容不一致，读了“脏”数据。

并发操作带来的不一致问题

三、读“脏”数据

例如： C = 100

T_1	T_2
C = 200 WRITE(C)	
	READ(C) C = 200
ROLLBACK	

T_2读了“脏”数据

并发操作带来的不一致问题

练习

说明以下三种并发操作带来的数据不一致问题各是什么。

（1） T_1	T_2
READ(A) A=A-1	
	READ(A) A=A-1
WRITE(A)	
	WRITE(A)

（2） T_1	T_2
READ(B) B＝100	
	B＝200 WRITE(B)
READ(B) B＝200	

（3） T_1	T_2
C＝200 WRITE(C)	
	READ(C) C＝200
ROLLBACK	

并发操作带来的不一致问题

解答

丢失了T1的更新

（1） T_1	T_2
READ(A) A=A-1	
	READ(A) A=A-1
WRITE(A)	
	WRITE(A)

T_1读了过时的数据

（2） T_1	T_2
READ(B) B＝100	
	B＝200 WRITE(B)
READ(B) B＝200	

（3） T_1	T_2
C＝200 WRITE(C)	
	READ(C) C＝200
ROLLBACK	

T_2读了“脏”数据

并发操作带来的不一致问题

小结

数据库系统是一个多用户系统，允许用户并发操作。对并发操作不加以控制的话，可能带来丢失更新、不一致分析和读“脏”数据的问题。


谢谢！

封锁的基本概念

封锁的基本概念

总述

封锁是实现数据库并发控制最常用的方法。本知识点学习封锁技术的基本理论。

封锁的基本概念

一、封锁的概念

二、锁的种类

三、相容矩阵

封锁的基本概念

一、封锁的概念

封锁是指事务对数据库操作之前，先对数据加锁获得这个数据对象的一定控制，其他事务不能更新此数据直到该事务解锁为止。

在飞机订票例子中，若甲事务要修改A时，在读出A前先封锁A，这时其他事务就不能读取和修改A，直到甲修改完并写回A后，解除了对A的封锁为止，这样就不会丢失甲事务对A的修改。

封锁的基本概念

一、封锁的概念

时间	事务T_1	数据库中A的值	事务T_2
1	LOCK X(A)	16	
2	READ(A)		
3			LOCK X(A)
4	A=A-1		WAIT
5	WRITE(A)		WAIT
6	UNLOCK(A)	15	WAIT
7			LOCK X(A)
8			READ(A)
9			A=A-1
10		14	WRITE(A)
11			UNLOCK(A)

二、锁的种类

（1）排他锁-X锁（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，其他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。

二、锁的种类

（2）共享锁-S锁（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，其他事务只能再对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。

三、相容矩阵

一个事务是否能加上锁由相容矩阵决定。

T_1 \ T_2	x型封锁	s型封锁	—
x型封锁	N	N	Y
s型封锁	N	Y	Y
—	Y	Y	Y

封锁的基本概念

练习

封锁机制中，若事务T对数据对象W加了X锁，下面（　　）的描述是正确的。

A. T只能读取W，不能修改W

B. 其他事务也能读取W

C. T只能修改W，不能读取W

D. T可以读取和修改W

封锁的基本概念

解答

封锁机制中，若事务T对数据对象W加了X锁，下面（ D ）的描述是正确的。

A. T只能读取W，不能修改W

B. 其他事务也能读取W

C. T只能修改W，不能读取W

D. T可以读取和修改W

小结

并发控制的主要方法是采用封锁机制。封锁是DBMS实现并发控制的一个非常重要的技术。

常用的锁有两种，即排他锁（写锁，X锁）和共享锁（读锁，S锁）。

对各种锁的使用还需遵循一些约定的规则。


谢谢！

封锁协议

封锁协议

总述

封锁是实现数据库并发控制最常用的方法，锁的控制规则称为封锁协议。本知识点学习三级封锁协议，了解使用X锁和S锁封锁数据对象的规则。

封锁协议

一、一级封锁协议

二、二级封锁协议

三、三级封锁协议

封锁协议

一、一级封锁协议

加锁后如何使用锁或如何控制，把加锁的控制规则称为协议，加锁必须遵守一定的协议，才能保证调度的正确性，从而保证数据的一致性。

一级封锁协议是X封锁的规则。

封锁协议

一、一级封锁协议

一级封锁协议的内容为：

任何企图更新记录R的事务必须先执行LOCK X(R)操作，对它取得X封锁， X封锁必须保留到事务终点(COMMIT或ROLLBACK) 。如果未获得X封锁，那么这个事务进入等待状态，一直到获准X封锁，事务继续进行。

简记为：修改数据加X锁，直到事务结束释放X锁。取不到，就等待。

一、一级封锁协议

一级封锁协议解决丢失更新的问题。

时间	更新事务T_1	数据库中A的值	更新事务T_2
1	LOCK X(A)	16	
2	READ(A)		
3			LOCK X(A)
4	A=A-1		WAIT
5	WRITE(A)		WAIT
6	COMMIT	15	WAIT
7	UNLOCK(A)		WAIT
8			LOCK X(A)
9			READ(A)
10			A=A-1
11			WRITE(A)
12		14	COMMIT
13			UNLOCK(A)

二、二级封锁协议

一级封锁协议对要读的数据不加锁，不能保证可重复读和不读“脏”数据。

二级封锁协议是在一级封锁协议基础上增加的对S封锁的规则。

二、二级封锁协议

时间	T_1 A+B+C	A,B,C的值	T_2 C-10, A+10
1	READ(A)	40,50,30	
2	SUM=A READ(B)		
3	SUM=SUM+B		
4			LOCK X(C)
5			READ(C) C=C-10
6			WRITE(C)
13	SUM=SUM+C（110）		

先执行T_1后执行T_2，T_1求得的结果是120

先执行T_2后执行T_1，T_1求得的结果也是120

T_1和T_2并发执行，T_1求得的结果是110

对读的数据不加锁，事务T_1得到“脏”数据

二、二级封锁协议

二级封锁协议的内容为：

任何要读取记录R的事务必须先执行LOCK S(R)操作，如果未获准S封锁，那么这个事务进入等待状态，一直到获准S封锁，事务才继续进行下去。当事务获准对记录R的S封锁后，在记录R修改前必须把S封锁升级为X封锁。

简记为：读数据加S锁，再执行，锁不到，就等待，读完后就释放S锁。

二、二级封锁协议

时间	T_1 A+B+C	A,B,C的值	T_2 C-10, A+10
1	LOCK S(A)	40,50,30	
2	READ(A)		
3	SUM=A		
4	LOCK S(B)		
5	READ(B)		
6	SUM=SUM+B		LOCK X(C)
9			WRITE(C)
10		40,50,20	LOCK X(A)
11			WAIT
12	LOCK S(C)		WAIT
13	WAIT		

二级封锁协议可避免读“脏”数据的问题

二、二级封锁协议

时间	T_1读两次A	A的值	T_2修改A
1	LOCK S(A)	50	
2	READ(A)		
3	UNLOCK(A)		
4			LOCK X(A)
5			READ(A)
7			WRITE(A)
8		40	COMMIT
9	LOCK S(A)		
10	READ(A)	40	
11	UNLOCK(A)		

二级封锁协议没有解决“不可重复读”的问题

三、三级封锁协议

在二级封锁协议中，对要读的数据加锁，读完可以释放锁，没有将S封锁保持到最后，不能保证可重复读。

三级封锁协议是在二级封锁协议基础上增加了一条规则“将S封锁保持到事务终点”。

三、三级封锁协议

三级封锁协议可以解决“不可重复读”问题

时间	T_1读两次A	A的值	T_2修改A
1	LOCK S(A)	50	
2	READ(A)		
3			LOCK X(A)
4			WAIT
5	READ(A)		WAIT
6	UNLOCK(A)	50	WAIT
7			LOCK X(A)
8			READ(A)
9			A=A-10
10			WRITE(A)

三、三级封锁协议

不同级别的封锁协议达到的系统一致性级别不同，封锁协议级别越高，一致性程度越好。

	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		√		√	√	√	√

练习

在数据库管理系统的三级封锁协议中，一级封锁协议能够解决的问题是（　）。

A. 丢失修改

B. 不可重复读

C. 读脏数据

D. 死锁

解答

在数据库管理系统的三级封锁协议中，一级封锁协议能够解决的问题是（A）。

A. 丢失修改

B. 不可重复读

C. 读脏数据

D. 死锁

小结

遵守封锁协议“修改数据加X锁，直到事务结束释放。”可以解决“丢失更新”的问题。

遵守封锁协议“读数据加S锁，直到事务结束释放。” 可以解决“读脏数据”和“不可重复读”的问题。


谢谢！

活锁与死锁

活锁与死锁

总述

封锁是实现数据库并发控制最常用的方法，封锁的方法可能引起活锁与死锁问题。本知识点学习活锁与死锁的基本概念。

活锁与死锁

一、活锁

二、死锁

活锁与死锁

一、活锁

某一事务的请求可能永远得不到，该事务一直处于等待状态，称为活锁。

时间	事务T_1	事务T_2	事务T_3	事务T_4
1	LOCK S(A)	…	…	…
2	…	LOCK X(A)	…	…
3	…	WAIT	LOCK S(A)	…
4	UNLOCK(A)	WAIT	WAIT	LOCK S(A)
5	…	WAIT	LOCK S(A)	WAIT
6		WAIT	…	WAIT
7		WAIT	UNLOCK(A)	WAIT
8		…	…	LOCK S(A)

活锁与死锁

一、活锁

某一事务的请求可能永远得不到，该事务一直处于等待状态，称为活锁。

避免活锁的简单方法是采用先来先服务的策略。

当多个事务请求封锁同一数据对象时，按先后次序对事务排队。数据对象上的锁一旦释放就批准队列中的第一个事务获得锁。

二、死锁

有两个或以上的事务处于等待状态，每个事务都在等待另一个事务解除封锁，它才能继续执行下去，结果任何一个事务都无法执行，这种现象就是死锁。

时间	事务T_1	事务T_2
1	LOCK X(A)	
2		LOCK X(B)
3	READ(A)	
4		READ(B)
5	LOCK X(B)	
6	WAIT	LOCK X(A)
7	WAIT	WAIT
8	WAIT	WAIT

二、死锁

DBMS如何解决死锁?

(1) 预防死锁

(2) 诊断解除

二、死锁

（1）预防死锁

- 一次封锁法
  - 一次将要使用的数据全部加锁(要么都加上，要么都释放)

A	B	C
√	√	√
×	×	×

  - 加大封锁范围，降低并发度

二、死锁

（1）预防死锁

- 顺序封锁法
  - 所有事务按封锁顺序实行封锁

A	B	C
1	2	3
√	√	√
×	×	√

  - 维护封锁对象的顺序成本高

预防死锁的策略适合OS，不适合DB。

二、死锁

（2）诊断与解除死锁

是由DBMS中的“死锁测试程序”来检查，如发现死锁则牺牲一个事务，并做回退操作，解除它的所有封锁。

确定死锁发生一般采用两种方法。

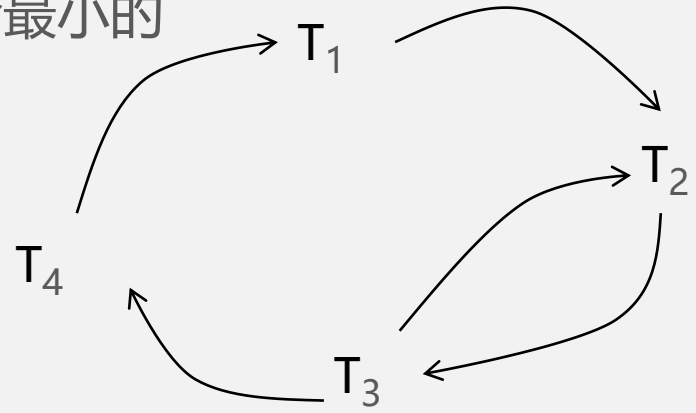
- 超时法：事务等待时间超过时限，认为死锁。
- 等待图法：等待有向图中出现回路，出现死锁。

二、死锁

（2）诊断与解除死锁

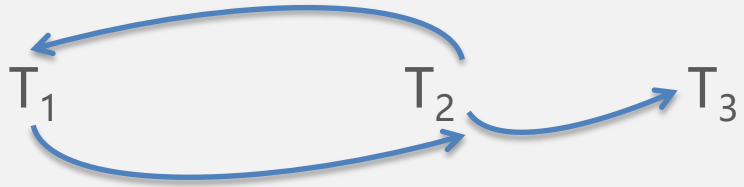
牺牲哪个事务?

- 最晚交付的
- 获得锁最少的
- 卷回代价最小的



练习

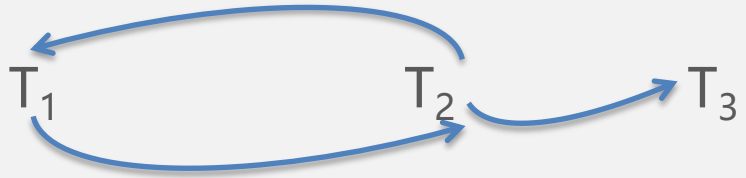
已知事务等待图如下图所示，请判断是否产生死锁。



解答

事务等待图是一个有向图，T_2指向T_1的有向边表示T_2等待T_1，T_1指向T_2的有向边表示T_1等待T_2，T_2指向T_3的有向边表示T_2等待T_3。

图中可知，T_1等待T_2，T_2又等待T_1，因此产生死锁。



小结

对数据对象施加封锁，解决并发可能引起的不一致，又可能产生新的问题，即活锁和死锁问题。

DBMS避免活锁一般采用先来先服务的策略。

DBMS对死锁的处理办法一般不是预防死锁发生而是采用发现死锁并处理的方法。


谢谢！

可串行化调度

可串行化调度

总述

数据库管理系统对并发事务不同的调度可能会产生不同的结果，其中执行结果等价于串行调度的调度是正确的。本知识点学习并发调度的可串行性。

可串行化调度

一、串行调度和并发调度

二、可串行化调度

可串行化调度

一、串行调度和并发调度

事务的执行次序称为调度，对多个事务处理有两种调度方法：

（1）串行调度：事务的依次执行称为串行调度。

（2）并发调度：利用分时的方法，同时处理多个事务，称为事务的并发调度。

可串行化调度

一、串行调度和并发调度

这是一个先T_1 ,后T_2的串行调度。

时间	事务T_1 A=B+1	数据库中A,B的值	事务T_2, B=A+1
1	LOCK S(B)	10,20	
2	READ(B)		
3	UNLOCK(B)		
4	LOCK X(A)		
5	A=B+1		
6	WRITE(A)		
7	UNLOCK(A)	21,20	
8			LOCK S(A)
9			READ(A)
10			UNLOCK(A)
11			LOCK X(B)
12			B=A+1
13			WRITE(B)
14		21,22	UNLOCK(B)

可串行化调度

一、串行调度和并发调度

这是一个先T_2 ,后T_1的串行调度。

时间	事务T_1 A=B+1	数据库中A,B的值	事务T_2, B=A+1
1		10,20	LOCK S(A)
2			READ(A)
3			UNLOCK(A)
4			LOCK X(B)
5			B=A+1
6			WRITE(B)
7		10,11	UNLOCK(B)
8	LOCK S(B)		
9	READ(B)		
10	UNLOCK(B)		
11	LOCK X(A)		
12 、13	A=B+1 WRITE(A)		
14	UNLOCK(A)	12,11	

可串行化调度

一、串行调度和并发调度

这是一个并发调度。

时间	事务T_1 A=B+1	数据库中A,B的值	事务T_2, B=A+1
1	LOCK S(B)	10,20	
2	READ(B)		
3	UNLOCK(B)		
4			LOCK S(A)
5			READ(A)
6			UNLOCK(A)
7	LOCK X(A)		
8	A=B+1	21,20	
9、10			LOCK X(B) B=A+1
11	WRITE(A)		
12	UNLOCK(A)		
13			WRITE(B)
14		21,11	UNLOCK(B)

二、可串行化调度

（1）可串行化调度：对于事务集(T_1,T_2,...,T_n)， 如果一个并发调度的结果与某一个串行调度的执行结果等价，则称此并发调度是可串行化的调度。

（2）不可串行化调度：对于某事务集的一个并发调度结果如果与任一串行调度均不等价，则该调度是不可串行化调度。

二、可串行化调度

先T_1后T_2，A、B的值为21,22

先T_2后T_1，A、B的值为12,11

时间	事务T_1 A=B+1	数据库中A,B的值	事务T_2, B=A+1
1	LOCK S(B)	10,20	
2	READ(B)		
3	UNLOCK(B)		
4			LOCK S(A)
5			READ(A)
6			UNLOCK(A)
7	LOCK X(A)		
8	A=B+1	21,20	
9、10			LOCK X(B) B=A+1
11	WRITE(A)		
12	UNLOCK(A)		
13			WRITE(B)
14		21,11	UNLOCK(B)

二、可串行化调度

并发调度的结果，A、B的值为21,11

这是一个不可串行化的并发调度。

时间	事务T_1 A=B+1	数据库中A,B的值	事务T_2, B=A+1
1	LOCK S(B)	10,20	
2	READ(B)		
3	UNLOCK(B)		
4			LOCK S(A)
5			READ(A)
6			UNLOCK(A)
7	LOCK X(A)		
8	A=B+1	21,20	
9、10			LOCK X(B) B=A+1
11	WRITE(A)		
12	UNLOCK(A)		
13			WRITE(B)
14		21,11	UNLOCK(B)

二、可串行化调度

只有可串行化的并发操作是正确的，而不可串行化的调度将破坏数据的一致性。

可串行化是并发事务正确性的准则，调度是可串行的，并发操作一定是正确的。

可串行化调度

练习

判断右图所示的调度是否是可串行化调度。

事务T_1是将B加1写回A，
事务T_2是将A加1写回B，
先T_1后T_2，B=4,A=3
先T_2后T_1，B=4,A=5
并发调度结果：

B=4,A=3

事务T_1 A=3，B=2	事务T_2 A=3，B=2
LOCK S(B)	
Y=R (B)=2	
UNLOCK(B)	
LOCK X(A)	
	LOCK S(A)
A=Y+1=3	WAIT
W(A)	WAIT
UNLOCK(A)	WAIT
	X=R (A)=3
	UNLOCK(A)
	LOCK X(B)
	B=X+1=4
	W(B)
	UNLOCK(B)

可串行化调度

解答

是可串行化调度。

事务T_1	事务T_2
LOCK S(B)	
Y=R (B)=2	
UNLOCK(B)	
LOCK X(A)	
	LOCK S(A)
A=Y+1=3	WAIT
W(A)	WAIT
UNLOCK(A)	WAIT
	X=R (A)=3
	UNLOCK(A)
	LOCK X(B)
	B=X+1=4
	W(B)
	UNLOCK(B)

小结

并发调度有很多方案。并发控制机制调度并发事务操作是否正确的判别准则是可串行性。

一个给定的并发调度，当且仅当它是可串行化的，才认为是正确调度。


谢谢！

冲突可串行化调度

冲突可串行化调度

总述

可串行化是并发事务调度正确性的判别准则，冲突可串行化是可串行化调度的充分条件。本知识点学习冲突可串行化调度的基本概念。

冲突可串行化调度

一、问题的提出

二、冲突操作

三、冲突可串行化调度

冲突可串行化调度

一、问题的提出

如何判断一个并发调度是可串行调度?

事务T_1	事务T_2
LOCK S(B)	
Y=R (B)=2	
UNLOCK(B)	
LOCK X(A)	
	LOCK S(A)
A=Y+1=3	WAIT
W(A)	WAIT
UNLOCK(A)	WAIT
	X=R (A)=3
	UNLOCK(A)
	LOCK X(B)
	B=X+1=4
	W(B)
	UNLOCK(B)

二、冲突操作

- 冲突操作是指不同的事务对同一个数据的读写操作和写写操作
  - $R_i(x)$与$W_j(x)$　　/* 事务T_i读x，T_j写x*/
  - $W_i(x)$与$W_j(x)$　　/* 事务T_i写x，T_j写x*/
- 其他操作是不冲突操作
  - $R_i(x)$与$R_j(x)$ 、 $R_i(x)$与$W_j(y)$ 、 $W_i(x)$与$W_j(y)$等

三、冲突可串行化调度

一个调度Sc在保证冲突操作的次序不变的情况下，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为冲突可串行化的调度。

不同事务的冲突操作（例如$W_i(x)$与$W_j(x)$）和同一事务的两个操作（例如$R_i(x)$与$W_i(y)$）不能交换(Swap)。

三、冲突可串行化调度

例1：今有调度

$Sc_1=r_1(A)w_1(A)r_2(A)w_2(A)r_1(B)w_1(B)r_2(B)w_2(B)$，判断该调度是不是冲突可串行化的调度。

- 把$w_2(A)$与$r_1(B)w_1(B)$交换，得到：
  $r_1(A)w_1(A)r_2(A)r_1(B)w_1(B)w_2(A)r_2(B)w_2(B)$；
- 再把$r_2(A)$与$r_1(B)w_1(B)$交换：

$Sc_2=r_1(A)w_1(A)r_1(B)w_1(B)r_2(A)w_2(A)r_2(B)w_2(B)$；

- Sc_2等价于一个串行调度T_1，T_2；
- Sc_1是冲突可串行化的调度。

三、冲突可串行化调度

冲突可串行化调度是可串行化调度的充分条件，不是必要条件。

也就是说，不满足冲突可串行化调度也可能是一个可串行化调度。

冲突可串行化调度

练习

下面（　　）是冲突操作。

A. R_1（X）与R_2（X）

B. R_2（X）与W_1（Y）

C. W_1（X）与W_2（Y）

D. R_1（Y）与W_2（Y）

冲突操作：

不同的事务对同一个数据的读写操作

不同的事务对同一个数据的写写操作

冲突可串行化调度

解答

下面（ D ）是冲突操作。

A. R_1 (X) 与R_2 (X)

B. R_2 (X) 与W_1 (Y)

C. W_1 (X) 与W_2 (Y)

D. R_1 (Y) 与W_2 (Y)

冲突可串行化调度

小结

一个调度是冲突可串行化，一定是可串行化的调度。可以用这种方法来判断一个调度是否是可串行化的。


谢谢！

两段锁协议

两段锁协议

总述

DBMS普遍采用两段锁协议实现并发调度的可串行性。本知识点学习两段锁协议的基本原理。

两段锁协议

一、两段锁协议的内容

二、两段锁协议和一次封锁法

两段锁协议

一、两段锁协议的内容

为了保证并发调度的正确性，DBMS的并发控制机制必须提供一定的手段来保证调度是可串行化的。

目前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”锁的含义是，事务分为两个阶段，第一个阶段获得封锁，但不能释放锁；第二阶段是释放封锁，但不能申请锁。

两段锁协议

一、两段锁协议的内容

两段锁协议要求，所有事务应遵守下列规则：

（1）在对任何数据进行读、写操作之前，事务首先要申请和获得该数据的封锁（扩展阶段）；

（2）在释放一个封锁以后，事务不再申请和获得任何其他封锁（收缩阶段）。

两段锁协议

一、两段锁协议的内容

例1：判断事务T_1和事务T_2是否遵守两段锁协议。

T_1的封锁序列:

遵守

Rlock A Rlock B Wlock C Unlock B Unlock A Unlock C

T_2的封锁序列:

不遵守

Rlock A Unlock A Rlock B Wlock C Unlock C Unlock B

两段锁协议

一、两段锁协议的内容

若所有事务均遵守两段锁协议，则这些事务的所有并发调度都是可串行化的。

忽略加锁和解锁操作后并发调度次序为：

R_1 (B) R_2 (C) W_1(A) R_2 (B)

变换为：

R_1 (B) W_1(A) R_2 (C) R_2 (B)

事务T_1	事务T_2
LOCK S(B)	
R (B)=20	
	LOCK S(C)
	R (C)=10
LOCK X(A)	
W(A)=30	
	LOCK S(B)
UNLOCK(B)	WAIT
	R (B)=20
	UNLOCK(B)
UNLOCK(A)	
	UNLOCK(C)

一、两段锁协议的内容

两段锁协议是可串行化调度的充分条件而不是必要条件。

事务不遵守两段锁协议，其并发调度也可能是可串行化的。

二、两段锁协议和一次封锁法

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。

两段锁协议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，这是因为每个事务都不能及时解除被它封锁的数据。

两段锁协议

练习

判断右图所示的调度是否符合两段锁协议。

事务T_1	事务T_2
LOCK S(B)	
Y=R (B)=2	
UNLOCK(B)	
LOCK X(A)	
	LOCK S(A)
A=Y+1=3	WAIT
W(A)	WAIT
UNLOCK(A)	WAIT
	X=R (A)=3
	UNLOCK(A)
	LOCK X(B)
	B=X+1=4
	W(B)
	UNLOCK(B)

两段锁协议

解答

两个事务均不符合两段锁协议，这是一个不遵守两段锁协议的并发调度。

事务T_1	事务T_2
LOCK S(B)	
Y=R (B)=2	
UNLOCK(B)	
LOCK X(A)	
	LOCK S(A)
A=Y+1=3	WAIT
W(A)	WAIT
UNLOCK(A)	WAIT
	X=R (A)=3
	UNLOCK(A)
	LOCK X(B)
	B=X+1=4
	W(B)
	UNLOCK(B)

小结

DBMS普遍采用两段锁协议实现并发调度的可串行性。

若所有事务均遵守两段锁协议，则这些事务的所有并发调度都是可串行化的。


谢谢！