12.2 数据库的并发控制

主要考点

1、事务调度
2、并发操作带来的问题
3、并发调度的可串行性
4、并发控制技术
5、两段锁协议
6、事务的隔离级别

事务调度

1、串行调度：是指多个事务依次串行执行，且只有当一个事务的所有操作都执行完成才执行另一个事务的所有操作。

- 例：有两个事务T_0和T_1，事务T_0从账号A转2000元到账号B；事务T_1从账号A转20%的款到账号B。T_0和T_1的定义如下所示。

T_0	T_1
read(A);	read(A);
A:=A-2000;	temp:=A*0.2;
write(A);	A:=A-temp;
read(B);	write(A);
B:=B+2000;	read(B);
write(B).	B:=B+temp;
	write(B).

图：银行转账举例

事务调度

时间	T_0	T_1
t1	read(A);	
t2	A:=A-2000;	
t3	write(A);	
t4	read(B);	
t5	B:=B+2000;	
t6	write(B).	
t7		read(A);
t8		temp:=A*0.2;
t9		A:=A-temp;
t10		write(A);
t11		read(B);
t12		B:=B+temp;
t13		write(B).

（a）调度S1：先T_0后T_1

T_0	T_1
	read(A);
	temp:=A*0.2;
	A:=A-temp;
	write(A);
	read(B);
	B:=B+temp;
	write(B).
read(A);	
A:=A-2000;	
write(A);	
read(B);	
B:=B+2000;	
write(B).	

（b）调度S2：先T_1后T_0

图：事务的串行调度

事务调度

2、并发调度：利用分时的方法同时处理多个事务。

时间	T_0	T_1
t1	read(A);	
t2	A:=A-2000;	
t3	write(A);	
t4		read(A);
t5		temp:=A*0.2;
t6		A:=A-temp;
t7		write(A);
t8	read(B);	
t9	B:=B+2000;	
t10	write(B).	
t11		read(B);
t12		B:=B+temp;
t13		write(B).

（a）调度S3：正确的调度

T_0	T_1
read(A);	
A:=A-2000;	
	read(A);
	temp:=A*0.2;
	A:=A-temp;
	write(A);
	read(B);
write(A);	
read(B);	
B:=B+2000;	
write(B).	
	B:=B+temp;
	write(B).

（b）调度S4：错误的调度

事务调度

3、可恢复调度：

指满足这样的条件的调度：当事务T_j要读事务T_i写的数据时，T_i事务必须要先于事务T_j提交。

时间	T_0	T_1
t1	read(A);	
t2	write(A);	
t3		read(A);
t4		/*COMMIT*/
t5	read(B);	
t6	/*ROLLBACK*/	

（b）不可恢复的调度举例

例：某银行信息系统有两项业务对应的事务T1、T2与存款关系有关。其中，转账业务：T1(A,B,50)，从账户A向账户B转50元；计息业务：T2，对当前所有账户的余额计算利息，余额为X*1.01。针对上述业务流程，回答下列问题：

（1）若当前账户A余额为100元，账户B余额为200元。有两个事务分别为T1(A, B, 50），T2。可能的串行执行为：T1→T2 或 T2→T1，请计算串行执行结果。

（2）若上述两个事务的一个并发调度顺序如下图所示，请问调度是否正确，为什么？

T1（A, B, 50）	T2
Read(A)	
Write(A)	
	Read(A)
	Write(A)
	Read(B)
	Write(B)
Read(B)	
Write(B)	

并发操作带来的问题

- 并发操作带来的数据不一致性有三类：丢失修改、不可重复读和读脏数据。

1、丢失修改：两个事务对同一个数据进行修改，导致事务A对数据库的修改被事务B的修改所覆盖。

时间	T_1	T_2
t1	read(A) [16]	
t2		read(A) [16]
t3	A=A-1 [15]	
t4		A=A-1 [15]
t5	write(A) [15]	
t6		write(A) [15]

并发操作带来的问题

2、不可重复读：事务对同一数据进行两次读取的结果不同。原因是两次读取的间隙数据被另一事务修改了。

时间	T_1	T_2
t1	read(A) [50]	
t2	read(B) [100]	
t3	C=A+B [150]	
t4		read(B) [100]
t5		B=B*2 [200]
t6		write(B) [200]
t7	read(A) [50]	
t8	read(B) [200]	
t9	C=A+B [250]	

并发操作带来的问题

3、读脏数据：某事务读取的数据是其它事务修改后的值，但该修改后来又被撤销了。

时间	T_1	T_2
t1	read(C) [100]	
t2	C=C*2 [200]	
t3	write(C) [200]	
t4		read(C) [200]
t5		…
t6		…
t7	ROLLBACK（C=100）	

并发调度的可串行性

- 多个事务的并发执行是正确的，当且仅当其结果与某一次序串行地执行它们的结果相同，称这种调度策略是可串行化的调度。
- 可串行性是并发事务正确性的准则。即：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。

这两句话熟读甚至背下来

并发控制技术

- 并发事务如果对数据读写时不加以控制，会破坏事务的隔离性和一致性。为了保持事务的隔离性，系统必须对事务之间的相互作用加以控制，最典型的方式就是加锁。

1、排它锁(Exclusive Locks，简称X锁)：也称为写锁，用于对数据进行写操作时进行锁定。如果事务T对数据A加上X锁后，就只允许事务T对数据A进行读取和修改，其他事务对数据A不能再加任何锁，也不能读取和修改数据A，直到事务T释放A上的锁。

2、共享锁(Share Locks，简称S锁)：也称为读锁，用于对数据进行读操作时进行锁定。如果事务T对数据A加上了S锁后，事务T就只能读数据A但不可以修改，其他事务可以再对数据A加S锁来读取，只要数据A上有了S锁，任何事务都只能再对其加S锁读取而不能加X锁修改。

- 例：若事务T1对数据D1已加排它锁，事务T2对数据D2已加共享锁，那么事务T2对数据D1 （1） ；事务T1对数据D2 （2） 。

（1）A. 加共享锁成功，加排它锁失败　　B. 加排它锁成功，加共享锁失败
　　C. 加共享锁、排它锁都成功　　D. 加共享锁、排它锁都失败
（2）A. 加共享锁成功，加排它锁失败　　B. 加排它锁成功，加共享锁失败
　　C. 加共享锁、排它锁都成功　　D. 加共享锁、排它锁都失败

封锁协议

（1）一级封锁协议：是指事务T在修改数据A之前必须先对其加X锁，直到事务结束才释放X锁。解决了丢失修改的问题。

（2）二级封锁协议：是一级封锁协议加上事务T在读取数据A之前必须对其加上S锁，读完后即可释放S锁。解决了读脏数据的问题。

（3）三级封锁协议：是一级封锁协议加上事务T在读取数据A之前必须对其加上S锁，直到事务结束才释放S锁。解决了不可重复读的问题。

封锁协议	要求	可解决
一级封锁协议	修改前加X锁，事务结束后释放	丢失修改
二级封锁协议	在一级之上再规定：读取前加S锁，读完后释放	丢失修改、读脏
三级封锁协议	在一级之上再规定：读取前加S锁，事务结束后释放	丢失修改、读脏，不可重复读

两段锁协议

- 两段锁协议（2PL）：是指同一事务对任何数据进行读写之前必须对该数据加锁；在释放一个封锁之后，该事务不再申请和获得任何其他封锁。
- 所谓“两段”的含义是：事务分为两个阶段。第一阶段是获得封锁，也称为扩展阶段。第二阶段是释放封锁，也称为收缩阶段。

例：

T1：Slock A...Slock B...Xlock C...Unlock B...Unlock A...Unlock C

T2：Slock A...Unlock A...Slock B...Xlock C...Unlock C...Unlock B

- 如果事务遵循两段锁协议，那么它们的并发调度是可串行化的。两段锁是可串行化的充分条件，但不是必要条件。即：遵循两段锁协议，一定是可串行化的；不遵循两段锁协议，可能是可串行化的，也可能不是。
- 注意：采用两段锁协议也有可能产生死锁，这是因为每个事务都不能及时解除被封锁的数据，可能会导致多个事务都要求对方已经封锁的数据而不能继续运行。

PS：熟读甚至背下来

事务的隔离级别

1、READ UNCOMMITTED（读未提交）：可避免丢失修改。

2、READ COMMITTED（读已提交）：可避免丢失修改、读脏数据。

3、REPEATABLE READ（可重复读）：可避免丢失修改、读脏数据，不可重复读。

4、SERIALIZABLE（串行化）：最高级别，可避免丢失修改、读脏数据、不可重复读、幻读。

幻读：事务A查询得到N条数据，然后事务B又插入了M条数据，或者改变了这N条数据之外的M条符合事务A搜索条件的数据，导致事务A再次搜索发现有N+M条数据了，就产生了幻读。

案例： 某航空售票系统负责所有本地起飞航班的机票销售业务，并设有多个机票销售网点。各售票网点使用相同的售票程序。假设系统有如下业务及规则。

（1）售票程序中用到的伪指令如表11-2所示。

伪指令	说明
R(A, x)	返回航班A当前的剩余机票数给变量x
W(A, x)	当前数据库中航班A的剩余机票数置为x

（2）若某售票网点一次售出a张航班A的机票，则售票程序的伪指令序列为：R(A, X)；W(A, x-a)。

根据上述业务及规则，完成下列问题。

【问题1】假设有两个售票网点同时销售航班A的机票，那么在数据库服务器端可能出现如下的调度：

A：$R_1(A, x)$，$R_2(A, x)$，$W_1(A, x-1)$，$W_2(A, x-2)$；

B：$R_1(A, x)$，$R_2(A, x)$，$W_2(A, x-2)$，$W_1(A, x-1)$；

C：$R_1(A, x)$，$W_1(A, x-1)$，$R_2(A, x)$，$W_2(A, x-2)$；

其中$R_i(A, x)$，$W_i(A, x)$分别表示第 i 个销售网点的读写操作，其余类同。

假设当前航班A剩余10张机票，请分析上述三个调度各自执行完后的剩余票数，并指出错误的调度及产生错误的原因。

【问题2】（1）判定事务并发执行正确性的准则是什么？如何保证并发事务正确地执行？

（2）采用相应的加锁、解锁指令，重写售票程序的伪指令序列，以保证正确的并发调度。