一、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

1、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。

> 当事务是可串行化的事务调度员即是正确的事务调度

例：有两个事务 T_0和T_1，事务T_0从账号A转2000元到账户B；事务T_1从账户A转20。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)

2、并发调度：利用分时的方法同时处理多个事务。

> 并发调度执行的结果与一次串行调度执行的结果相同，则称这个并发调度是正确的。

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)

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);

3、可恢复调度与不可恢复调度（了解）

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。

例：以下不可恢复调度举例

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

2、并发操作代理的问题

- 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。主要原因是事务的并发操作破坏了事务的隔离性。**
  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]

2. **不看重复读：**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] (验算不对)	

3. **读脏数据：**某事务读取的数据是其他事务修改后的值，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）	

4. **幻读：**也称幻影现象。事务T_1读取数据后，事务T_2执行插入或删除操作，使T_1无法再现前一次的读取结果。

3、并发调度的客串行性

以下两条背诵。

- **多个事务的并发执行是正确的，当且仅当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**

4、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，从而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，但任何事务都只能再对其加S锁读取而不能加X锁修改。

  排它锁（X锁/写锁）：A对B加了X锁就只有A能对B读数据、写数据。其他人不能对B加任何锁也不能读和写数据
  共享锁（S锁/读锁）：能读数据不能改数据，对加了S锁的数据只能加S锁才可以读取，且不能加X锁

```
XLOCK（A）     //此命令为对A加X锁
SLOCK（A）      //此命令为对A加S锁
UNLOCK（A）     //此命令为释放对A的加锁
```

5、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，知道事务结束才释放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	X锁		S锁		要求	可解决
	操作结束释放	事务结束释放	操作结束释放	事务结束释放		
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、读脏数据
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、读脏数据、不可重复读

6、两段锁协议

必考两段锁协议(背)

- **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改数据加锁；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。

> 两段的含义：
>
> 事务分为两个阶段；第一阶段是获得封锁，也称为扩展阶段。第二阶段是释放封锁，也称为收缩阶段。

- 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锁被封锁的数据，可能会导致多个事务都要求对方以及封锁的数据而不能继续运行。

7、事务的隔离级别

> 背

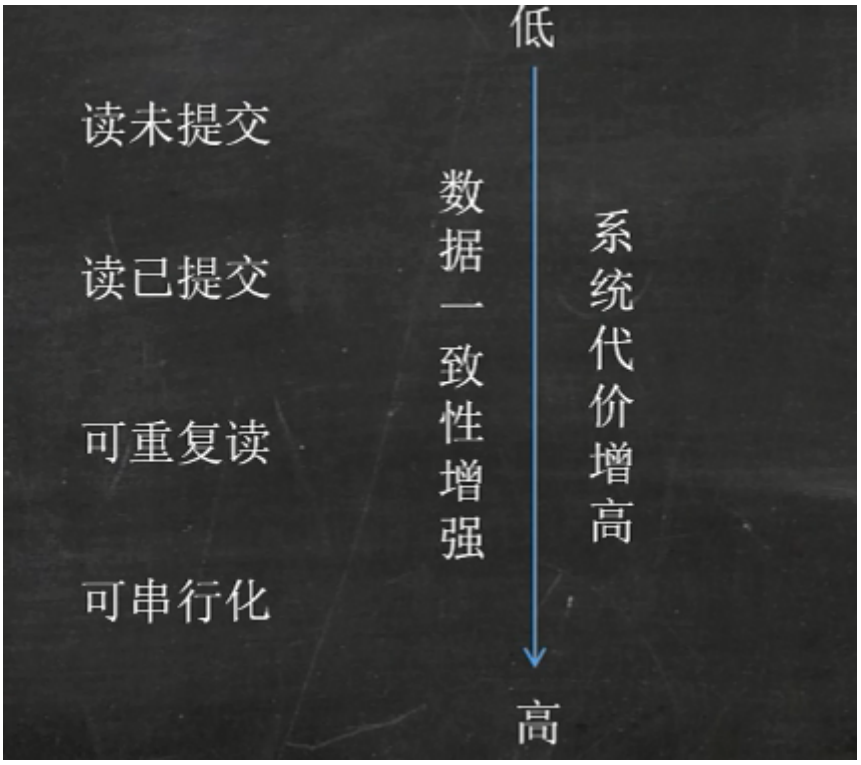
在SQL标准中给出了事务的4类隔离级别，由低到高依次如下：

1. **READ UNCOMMITTED（读未提交）**：允许一个事务可以读取一个未提交事务正在修改的数据。
2. **READ COMMITTED（读已提交）**：只允许一个事务读其他事务已提交的数据。
3. **REOEATABLE READ（可重复读）**：一个事务开始读取数据后，其他事务就不能再对该数据执行更新（UPDATE）操作了。
4. **SERIALZABLE（可串行化）**：最高级别，在该级别下，事务的执行顺序是可串行化的。

事务的隔离级别和数据不一致性的关系				
事务隔离级别	丢失修改	读脏数据	不可重复读	幻读
读未提交	×	√	√	√
读已提交	×	×	√	√
可重复读	×	×	×	√
可串行化	×	×	×	×
注意：√代表可能出现此不一致性，X代表不会出现此不一致性				

事务的隔离级别并不是越高越好，它与数据一致性以及系统代价的关系如下图：

> 事务的隔离级别越高，数据一致性越强，系统代价（开销）越高


低
读未提交
读已提交
可重复读
可串行化
数据一致性增强
系统代价增高
高