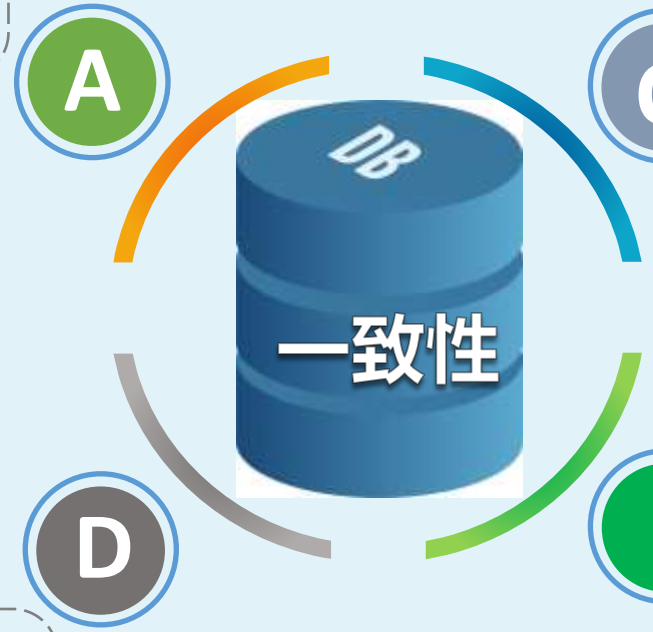
并发控制

引 言

原子性
Atomicity

一致性
Consistency



并发事务 ← 并发控制

持久性
Durability

隔离性
Isolation

讲授内容

1. 事务的调度
2. 数据不一致问题
3. 非串行调度的可串行化

事务的调度

- 多个事务中的并发操作按照它们的执行时间排序形成的一个操作序列称为调度。
  - 一个事务中的操作执行顺序是固定不变的
  - 并发事务中的操作可以交错执行

事务的调度

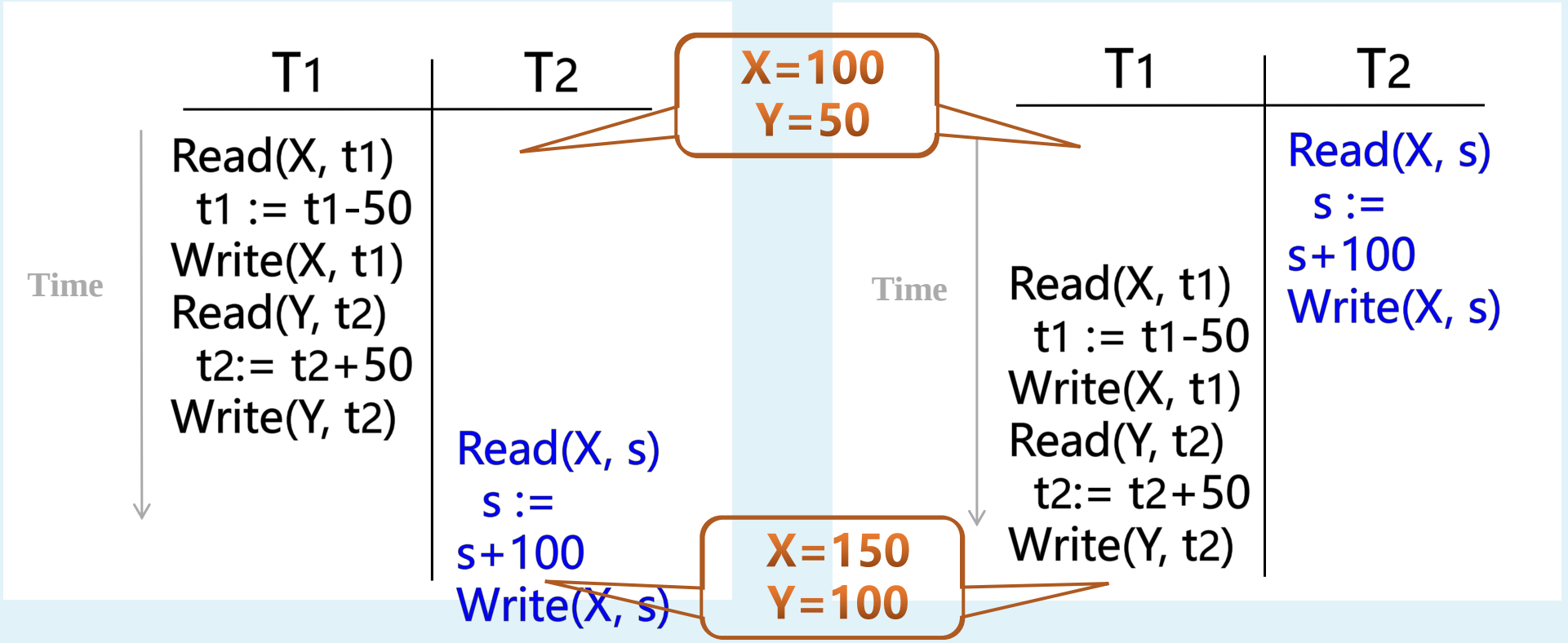
- 串行调度：每个事务中的操作都是连续执行的，不存在不同事务中的操作的交错执行。

```
BEGIN TRANSACTION T1
    Read(X, t1)
      t1 := t1-50
    Write(X, t1)
    Read(Y, t2)
      t2:= t2+50
    Write(Y, t2)
COMMIT
```

```
BEGIN TRANSACTION T2
    Read(X, s)
      s := s+100
    Write(X, s)
COMMIT
```

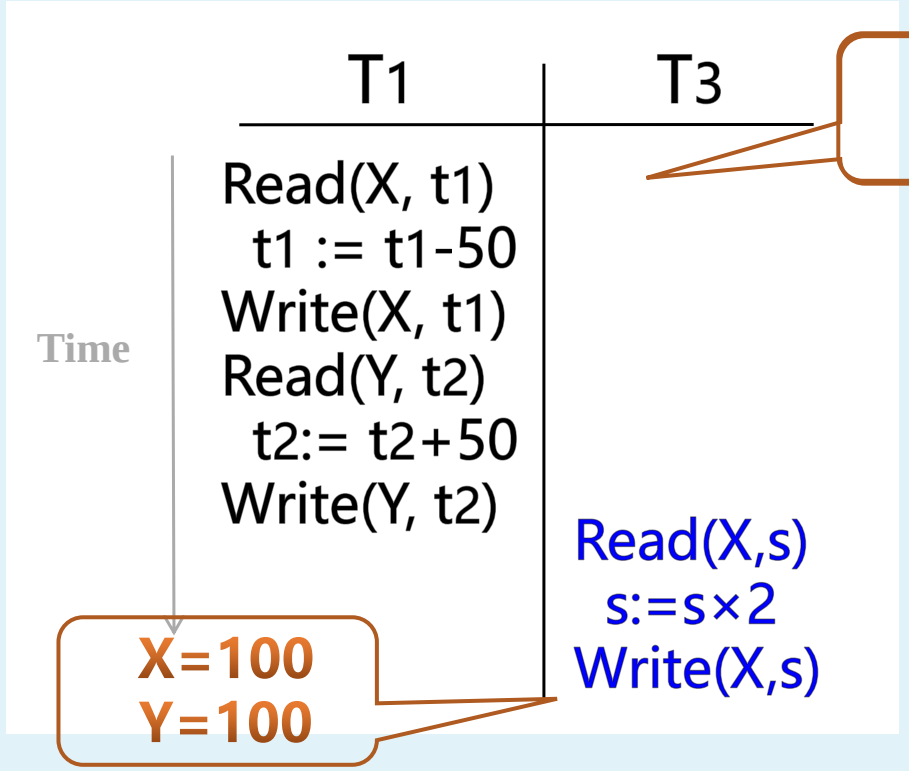
事务的调度

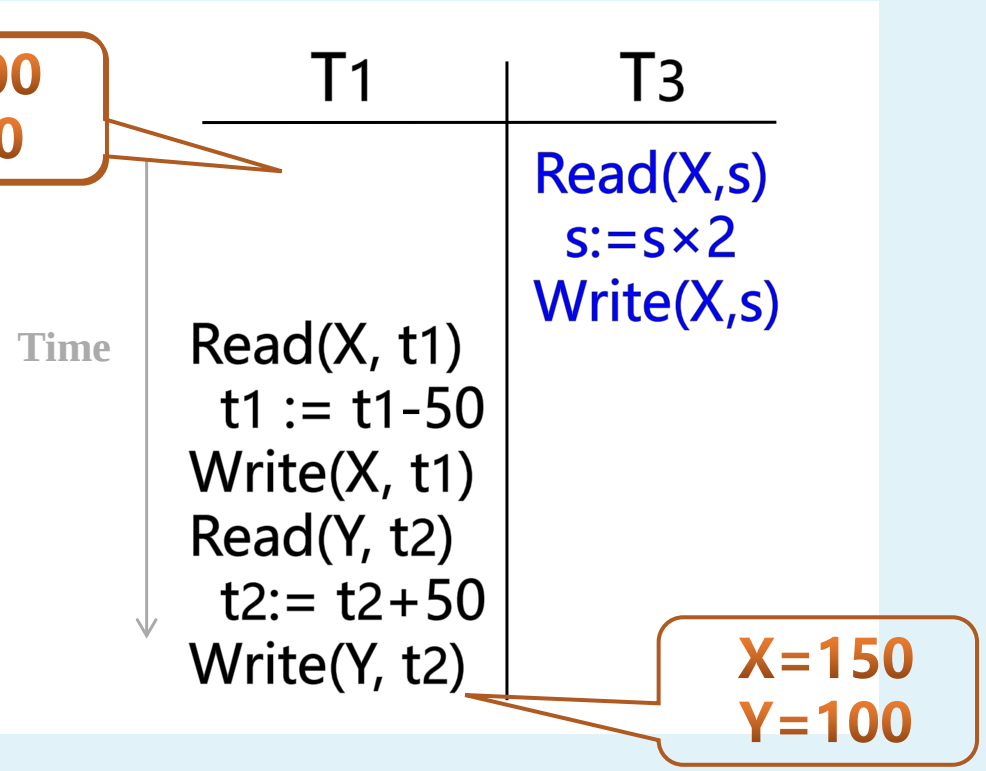
- 串行调度：每个事务中的操作都是连续执行的，不存在不同事务中的操作的交错执行。



事务的调度

- 串行调度：每个事务中的操作都是连续执行的，不存在不同事务中的操作的交错执行。





事务的调度

- 非串行调度：不同事务中的并发操作的交错执行。

	T1	T2
Time ↓	Read(X, t1)	
	t1 := t1-50	
		Read(X, s)
		s := s+100
	Write(X, t1)	
	Read(Y, t2)	
	t2:= t2+50	
		Write(X, s)
	Write(Y, t2)	

	T1	T2
Time ↓	Read(X, t1)	
	t1 := t1-50	
	Write(X, t1)	
		Read(X, s)
		s := s+100
	Read(Y, t2)	
	t2:= t2+50	
	Write(Y, t2)	
		Write(X, s)

事务的调度

- 非串行调度：不同事务中的并发操作的交错执行。

	T1	T2
Time ↓	Read(X, t1)	
	t1 := t1-50	
		Read(X, s)
		s := s+100
	Write(X, t1)	
	Read(Y, t2)	
	t2:= t2+50	
		Write(X, s)
	Write(Y, t2)	

	T1	T2
Time ↓	Read(X, t1)	
	t1 := t1-50	
	Write(X, t1)	
		Read(X, s)
		s := s+100
	Read(Y, t2)	
	t2:= t2+50	
	Write(Y, t2)	
		Write(X, s)

数据不一致问题

- 更新丢失（Lost Update）
- 脏读（Dirty Read）
- 不可重复读（Non_Repeatable Read）

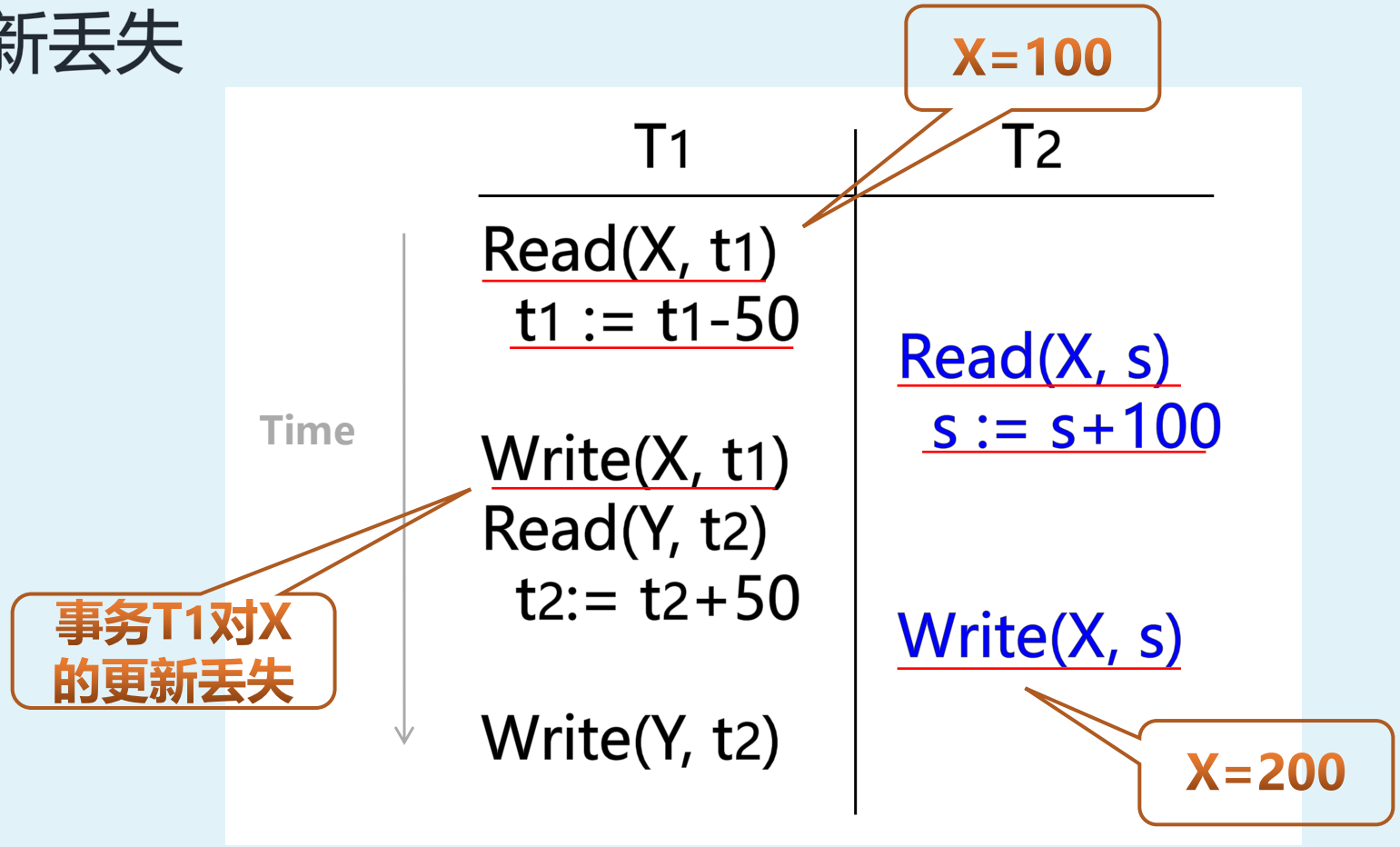
数据不一致问题

更新丢失

- 在并发事务的非串行调度中，并发操作先后读取同一数据对象并进行更新，一个事务的数据更新结果覆盖了另一个事务的数据更新结果，导致先写的数据更新结果丢失。

数据不一致问题

更新丢失



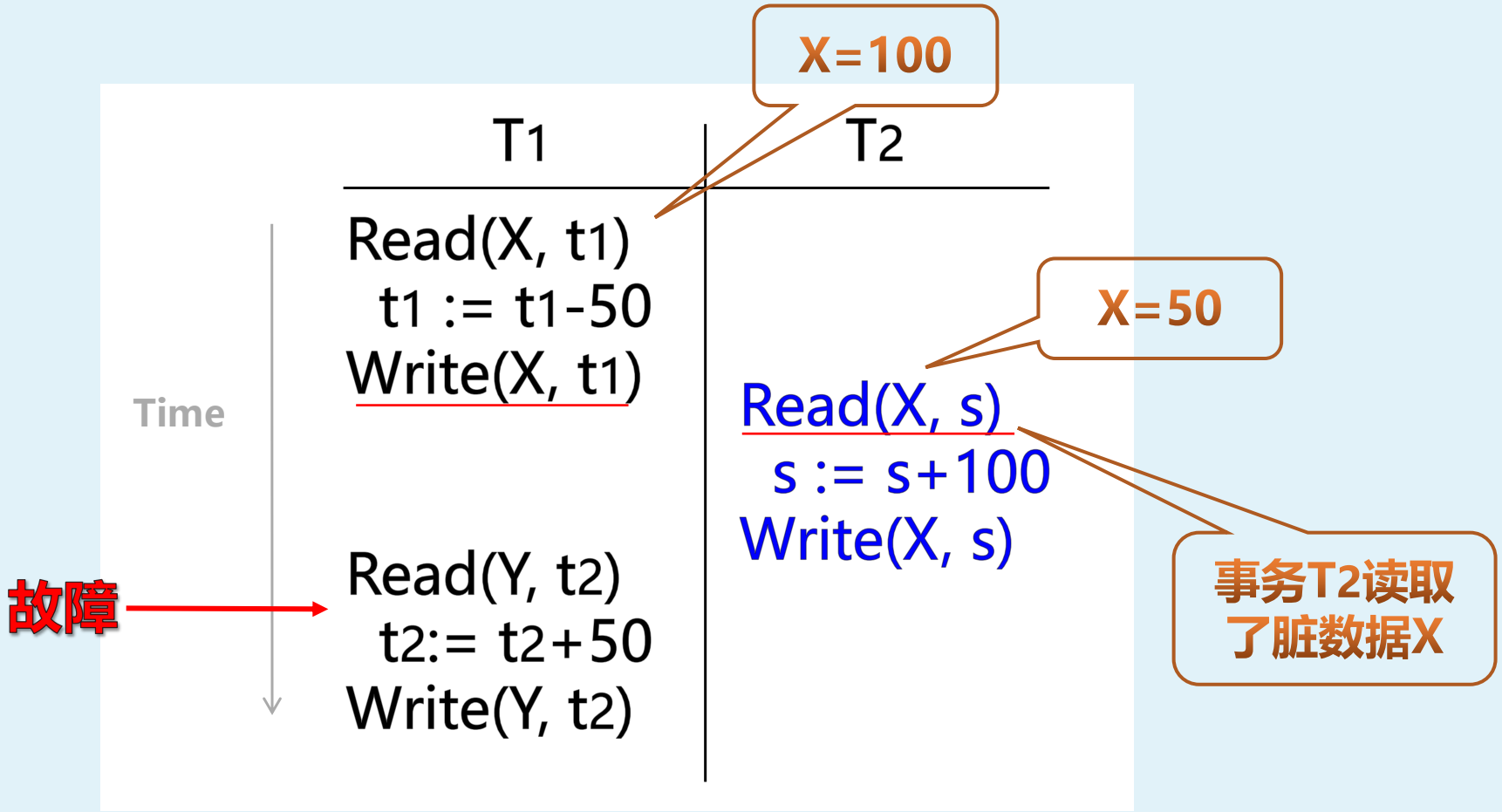
数据不一致问题

脏读

- 在并发事务的非串行调度中，一个事务读取了另一个还没有提交事务所写的中间结果数据（脏数据）。

数据不一致问题

脏读


X=100
T1
T2
Read(X, t1)
t1 := t1-50
Write(X, t1)
Time
X=50
Read(X, s)
s := s+100
Write(X, s)
故障
Read(Y, t2)
t2:= t2+50
Write(Y, t2)
事务T2读取了脏数据X

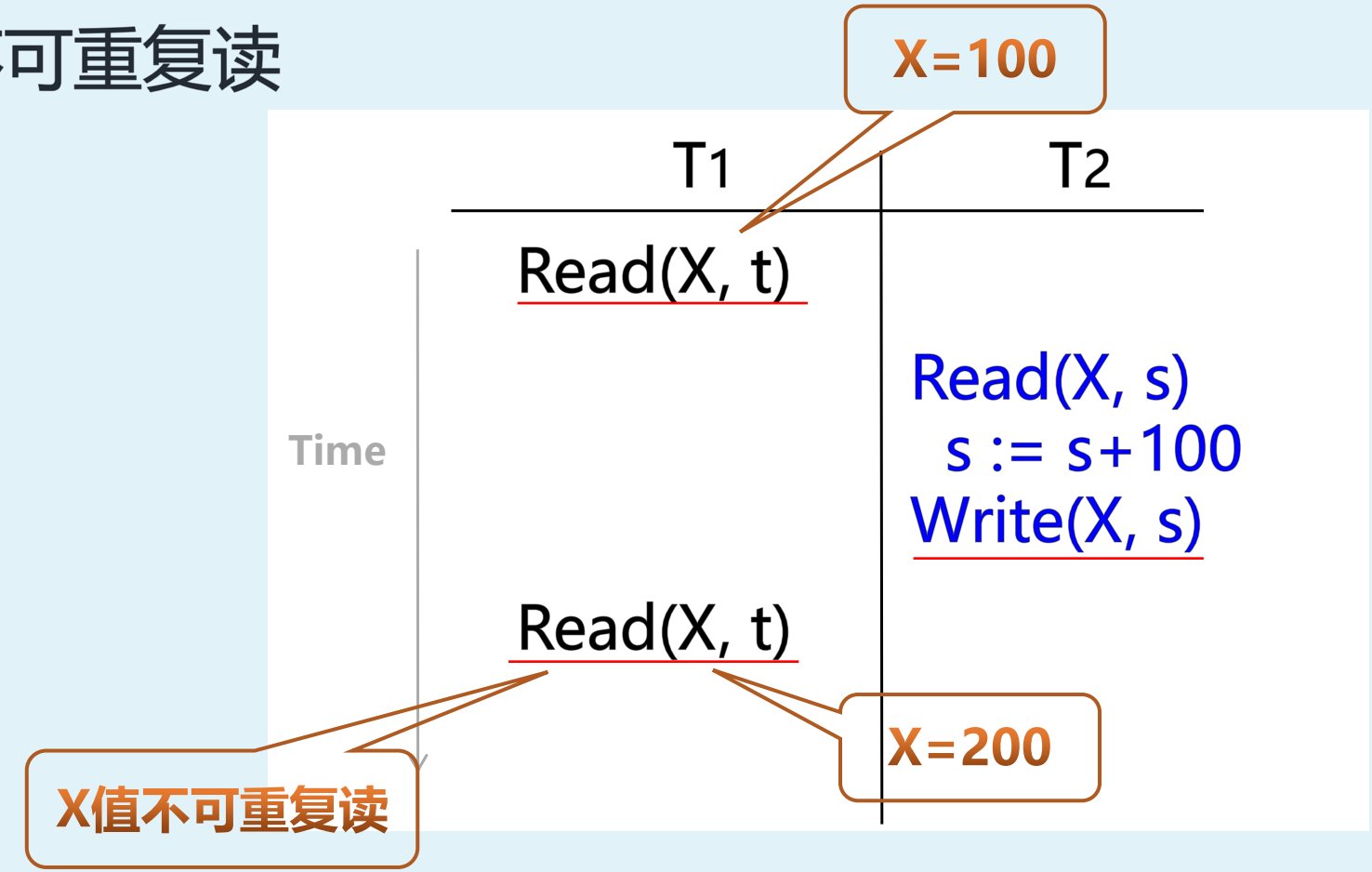
数据不一致问题

不可重复读

- 在并发事务的非串行调度中，同一事务对同一数据对象的多次读取操作得到不同的结果。

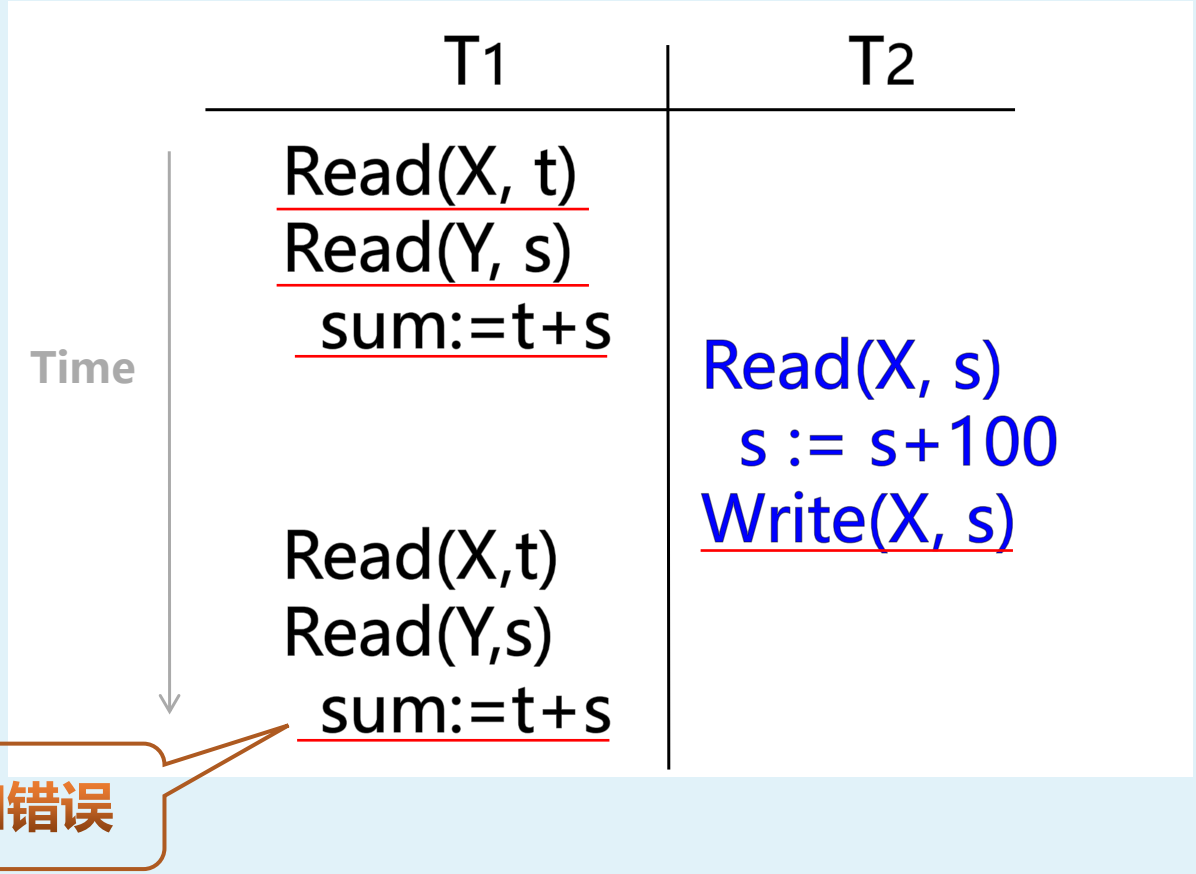
数据不一致问题

不可重复读



数据不一致问题

不可重复读



数据不一致问题

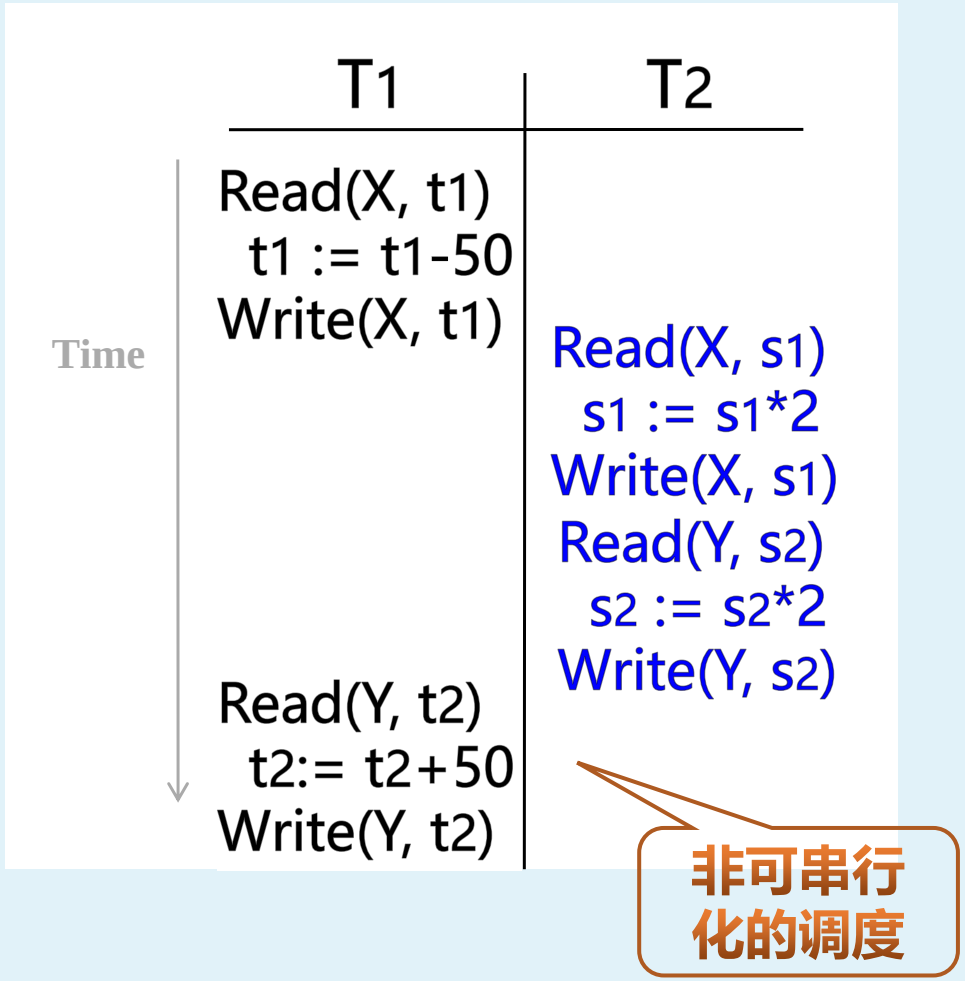
- 产生数据不一致问题的主要原因是并发事务的非串行调度的执行，使并发的事务之间互相干扰，破坏了并发事务之间的隔离性。
- 通过并发控制将并发事务的非串行调度的执行效果与这些并发事务的串行调度的执行效果相同，则可保持数据库的一致性。

非串行调度的可串行化

- 可串行化调度：若n个并发事务的一个非串行调度S等价于这n个并发事务的某个串行调度S’，那么S就是可串行化的。
  - 等价是指对于任意的数据库初始状态，调度S和调度S’的执行效果都相同。

非串行调度的可串行化


T1
T2
Time
Read(X, t1)
t1 := t1-50
Write(X, t1)
Read(X, s1)
s1 := s1*2
Write(X, s1)
Read(Y, t2)
t2:= t2+50
Write(Y, t2)
Read(Y, s2)
s2 := s2*2
Write(Y, s2)
可串行化的调度


T1
T2
Time
Read(X, t1)
t1 := t1-50
Write(X, t1)
Read(X, s1)
s1 := s1*2
Write(X, s1)
Read(Y, s2)
s2 := s2*2
Write(Y, s2)
Read(Y, t2)
t2:= t2+50
Write(Y, t2)
非可串行化的调度

非串行调度的可串行化

- 并发事务的非串行调度当且仅当是可串行化的，才能保持并发事务的隔离性。
- “可串行化”作为对并发事务进行并发控制的目标。
- 大多数DBMS的并发控制机制实现并发事务的非串行调度的冲突可串行化。

非串行调度的可串行化

- 冲突操作：并发事务非串行调度中来自不同事务的一对连续的操作（读操作或写操作），如果它们的执行顺序交换后，至少有一个事务的后续操作结果会改变，则这对操作就是冲突的。

非串行调度的可串行化

- 不同事务对不同数据对象的读写操作是不冲突的
- 不同事务对同一数据对象的读操作是不冲突的
- 不同事务对同一数据对象的读写操作是冲突的

非串行调度的可串行化

- 不同事务对同一数据对象的读写操作是冲突的
  - $w_i(X)$ 和 $w_j(X)$
  - $r_i(X)$ 和 $w_j(X)$

$r_i(X)$：事务Ti读数据X

$w_i(X)$：事务Ti写数据X

非串行调度的可串行化

- 不同事务对同一数据对象的读写操作是冲突的
  - $w_i(X)$ 和 $w_j(X)$
  - $r_i(X)$ 和 $w_j(X)$

交换后丢失的更新结果不同

Time

T1	T2
Read(X, t1)	
t1 := t1-50	
	Read(X, s)
	s := s+100
Write(X, t1)	
	Write(X, s)
Read(Y, t2)	
t2:= t2+50	
Write(Y, t2)	

非串行调度的可串行化

- 不同事务对同一数据对象的读写操作是冲突的
  - $w_i(X)$ 和 $w_j(X)$
  - $r_i(X)$ 和 $w_j(X)$

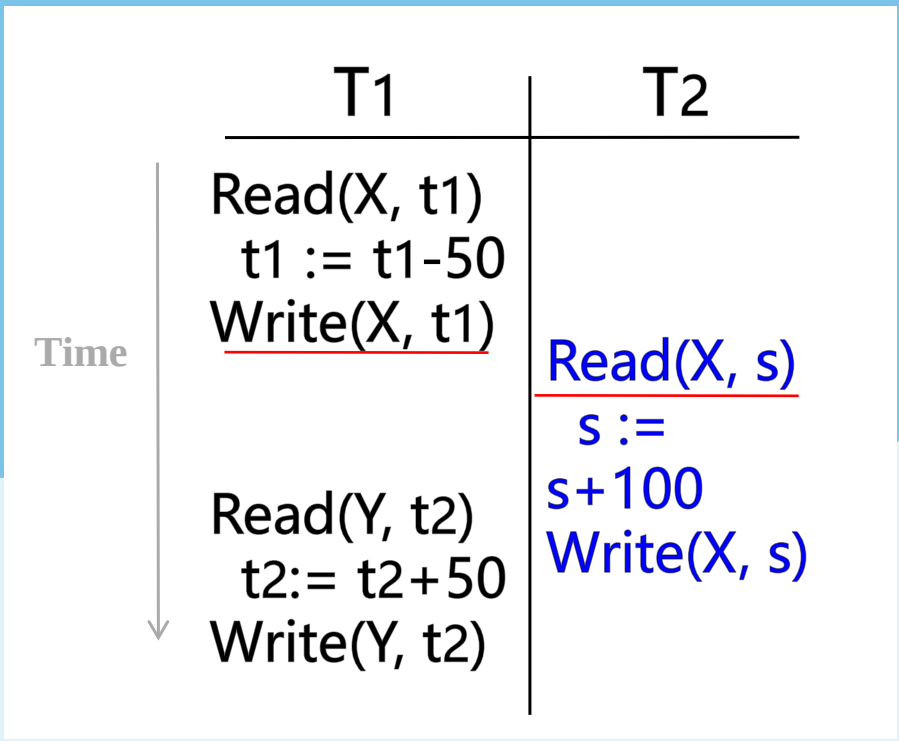
Time

T1	T2
Read(X, t1)	
t1 := t1-50	
	Read(X, s)
Write(X, t1)	
	s := s+100
Read(Y, t2)	
t2:= t2+50	
Write(Y, t2)	
	Write(X, s)

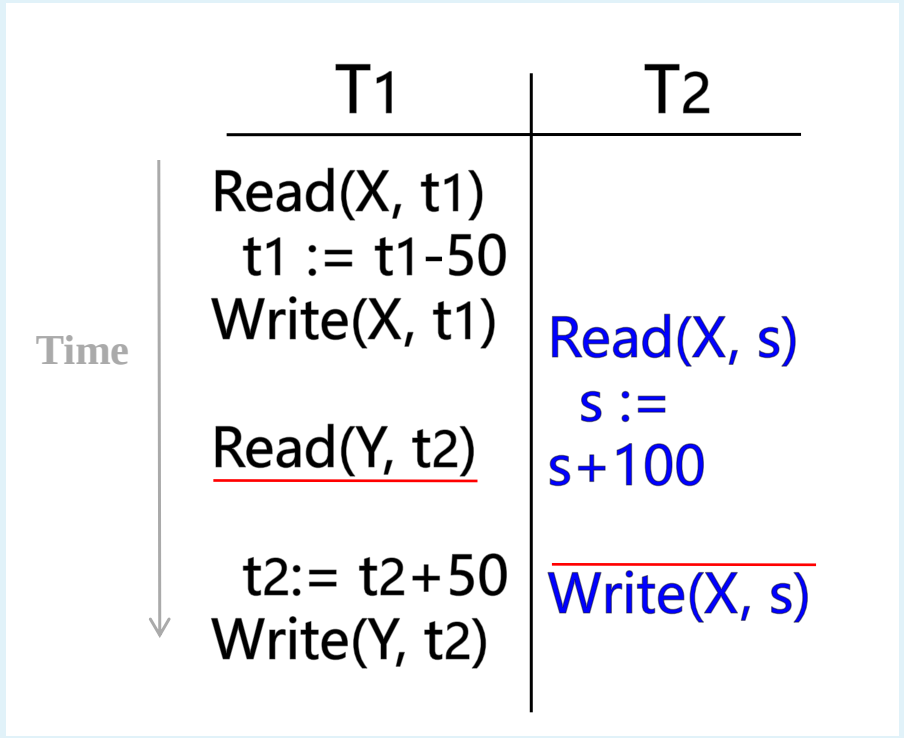
非串行调度的可串行化

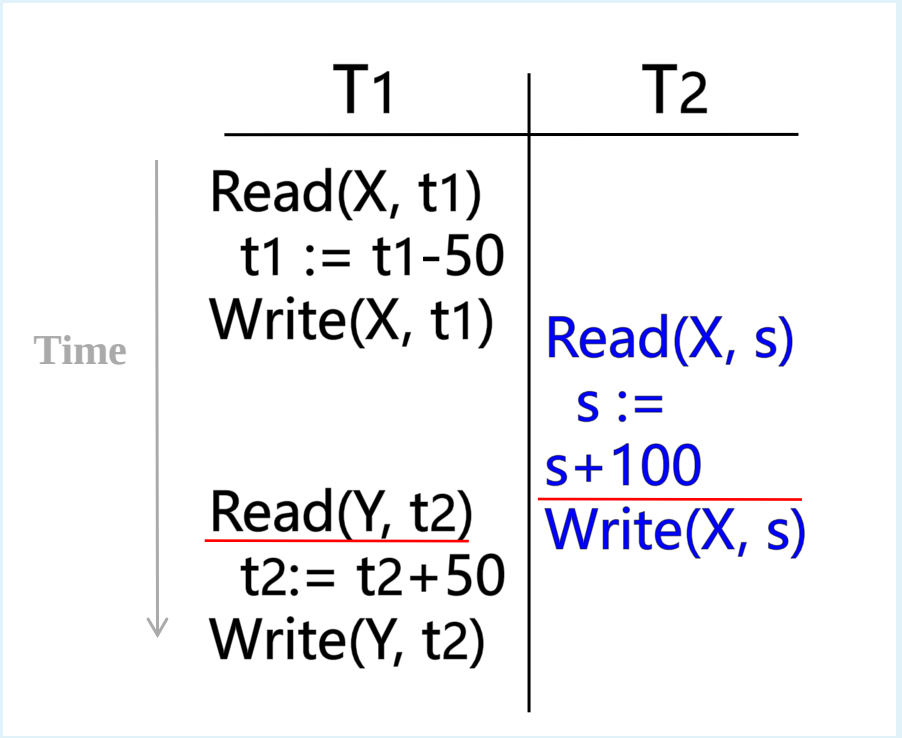
- 不同事务对同一数据对象的读写操作是冲突的
  - $w_i(X)$ 和 $w_j(X)$
  - $r_i(X)$ 和 $w_j(X)$

交换后可能出现
不可重复读或脏读现象



非串行调度的可串行化


T1
T2
Time
Read(X, t1)
t1 := t1-50
Write(X, t1)
Read(Y, t2)
t2:= t2+50
Write(Y, t2)
Read(X, s)
s :=
s+100
Write(X, s)


T1
T2
Time
Read(X, t1)
t1 := t1-50
Write(X, t1)
Read(Y, t2)
t2:= t2+50
Write(Y, t2)
Read(X, s)
s :=
s+100
Write(X, s)

非串行调度的可串行化

- 如果将一非串行调度通过一系列相邻的非冲突操作的交换转换为另一个调度，则这两个调度是冲突等价的。
- 如果一个非串行调度冲突等价于一个串行调度，则称该非串行调度是冲突可串行化的。

非串行调度的可串行化

- 下面这个调度是否是冲突可串行化的?

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

Time →

$r_i(X)$：事务Ti读数据X
$w_i(X)$：事务Ti写数据X

非串行调度的可串行化

- 下面这个调度是否是冲突可串行化的?

调度是冲突可串行化的

$r_1(A);\ w_1(A);\ r_2(A);\ w_2(A);\ r_1(B);\ w_1(B);\ r_2(B);\ w_2(B)$

$r_1(A);\ w_1(A);\ r_2(A);\ r_1(B);\ w_2(A);\ w_1(B);\ r_2(B);\ w_2(B)$

$r_1(A);\ w_1(A);\ r_1(B);\ r_2(A);\ w_1(B);\ w_2(A);\ r_2(B);\ w_2(B)$

$r_1(A);\ w_1(A);\ r_1(B);\ w_1(B);\ r_2(A);\ w_2(A);\ r_2(B);\ w_2(B)$

T1　　T2

非串行调度的可串行化

- 冲突可串行化是可串行化的一个充分条件，冲突可串行化调度是可串行化调度。

T1：$w_1(A)$; $w_1(B)$;　　T2：$w_2(A)$; $w_2(B)$;　　T3：$w_3(B)$;

非串行调度S1：$w_1(A)$; $w_2(A)$; $w_2(B)$; $w_1(B)$; $w_3(B)$;

等价

串行调度S2：$w_1(A)$; $w_1(B)$; $w_2(A)$; $w_2(B)$; $w_3(B)$;

是可串行化调度

不是冲突可串行化调度

小 结

- DBMS的并发控制机制采用一定的技术来保证并发事务非串行调度是可串行化的。
- 常用的并发控制技术
  - 封锁
  - 时间戳
  - 有效性确认

实现冲突可串行化