第十一章 并发控制

11.1 并发控制概述

11.2 封锁

11.3 活锁和死锁

11.4 并发调度的可串行性

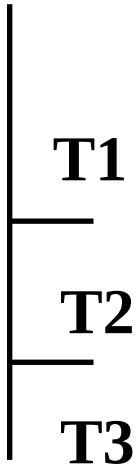
11.5 两段锁协议

问题的产生

- 多用户数据库系统的存在（允许多个用户同时使用）
  - 飞机、火车订票数据库系统
  - 银行数据库系统
- 不同的多事务执行方式

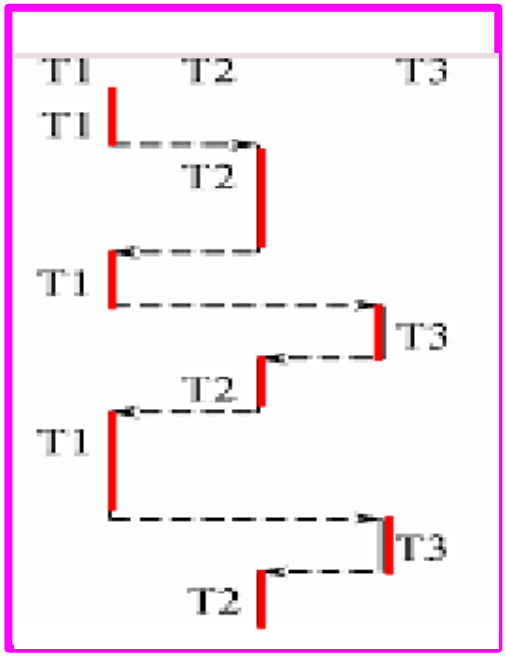
(1)事务串行执行

- 每个时刻只有一个事务运行，其他事务必须等到这个事务结束以后方能运行
- 不能充分利用系统资源，发挥数据库共享资源的特点



(2)交叉并发方式

- 并行事务的并行操作轮流交叉运行
- 单处理机系统中的并发方式能够减少处理机的空闲时间，提高系统的效率



(3)同时并发方式

- 多处理机系统中每个处理机可运行一个事务，多个处理机可同时运行多个事务，实现多个事务真正的并行运行
- 最理想的并发方式，但受制于硬件环境
- 更复杂的并发控制机制

本章仅讨论单处理机系统

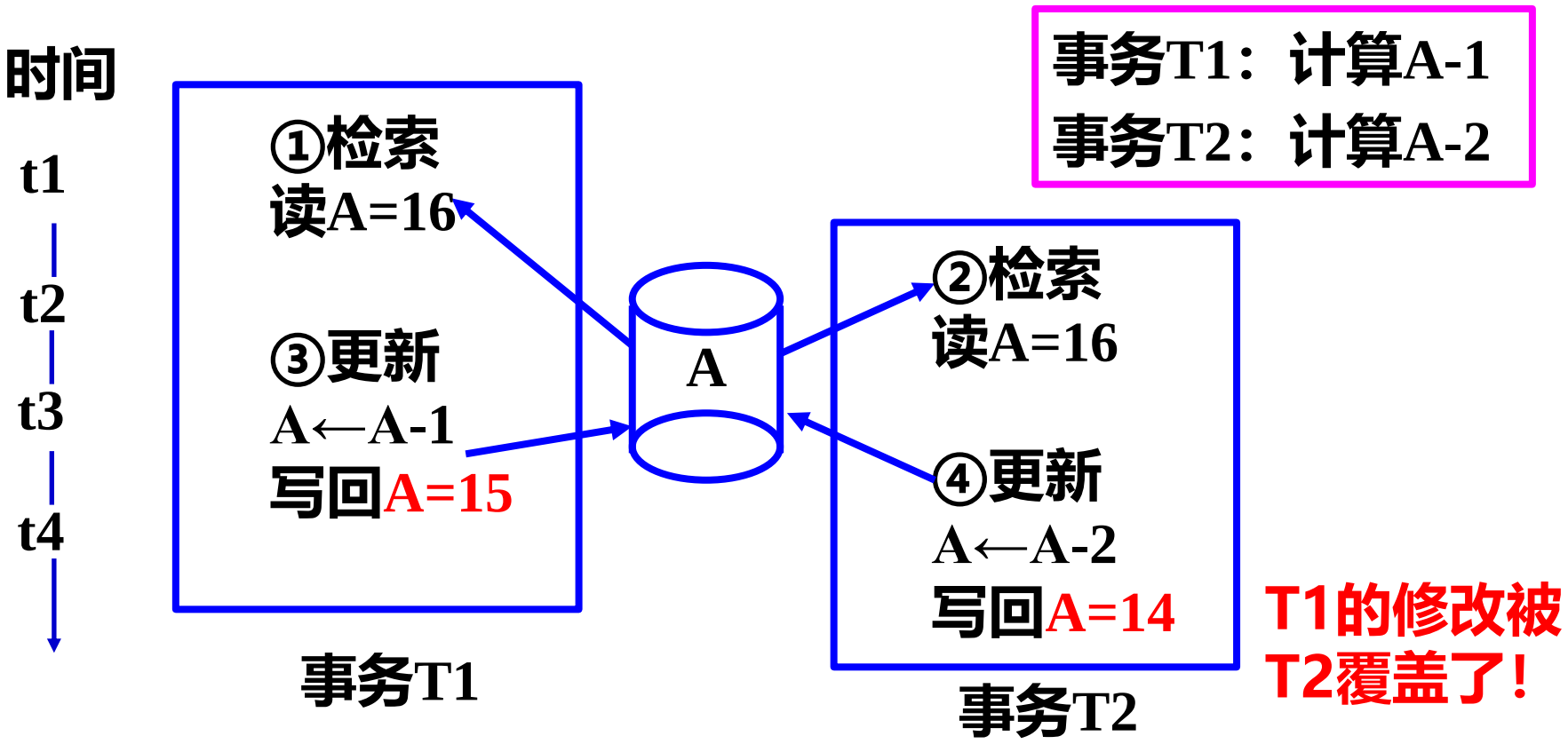
第十一章 并发控制

- 事务并发执行带来的问题
  - 会产生多个事务同时存取同一数据的情况
  - 可能会存取和存储不正确的数据，破坏事务的隔离性和数据库的一致性
- DBMS必须提供并发控制机制
- 事务是并发控制的基本单位
- 并发控制机制的任务
  - 对并发操作进行正确调度
  - 保证事务的隔离性
  - 保证数据库的一致性

并发操作带来的三类数据不一致性

(1) 丢失修改　写-写冲突

- 两个事务读入同一数据并修改，事务2的提交结果破坏了事务1提交的结果，导致事务1的修改被丢失。

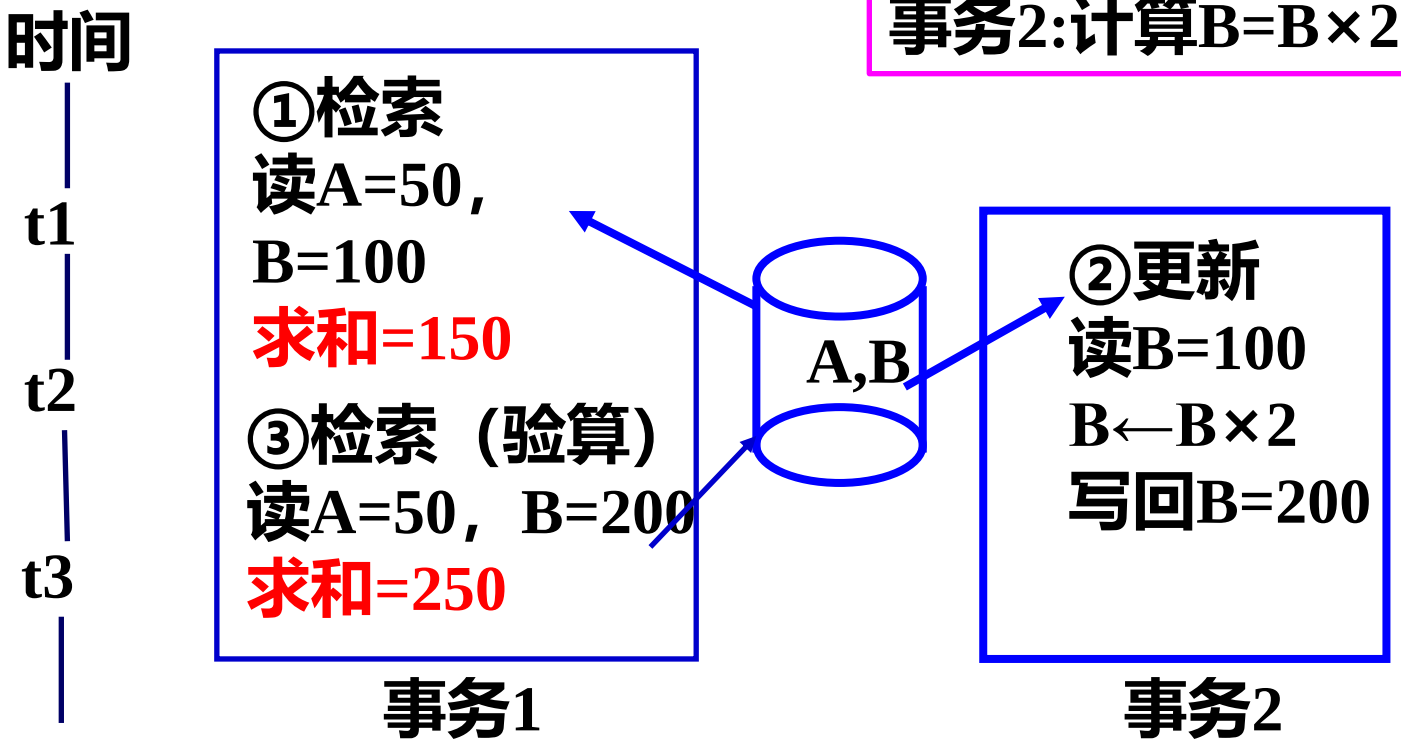


并发操作带来的三类数据不一致性

(2) 不可重复读 读-写冲突

- 事务1读取数据后，事务2执行更新操作，使事务1无法再现前一次读取结果。

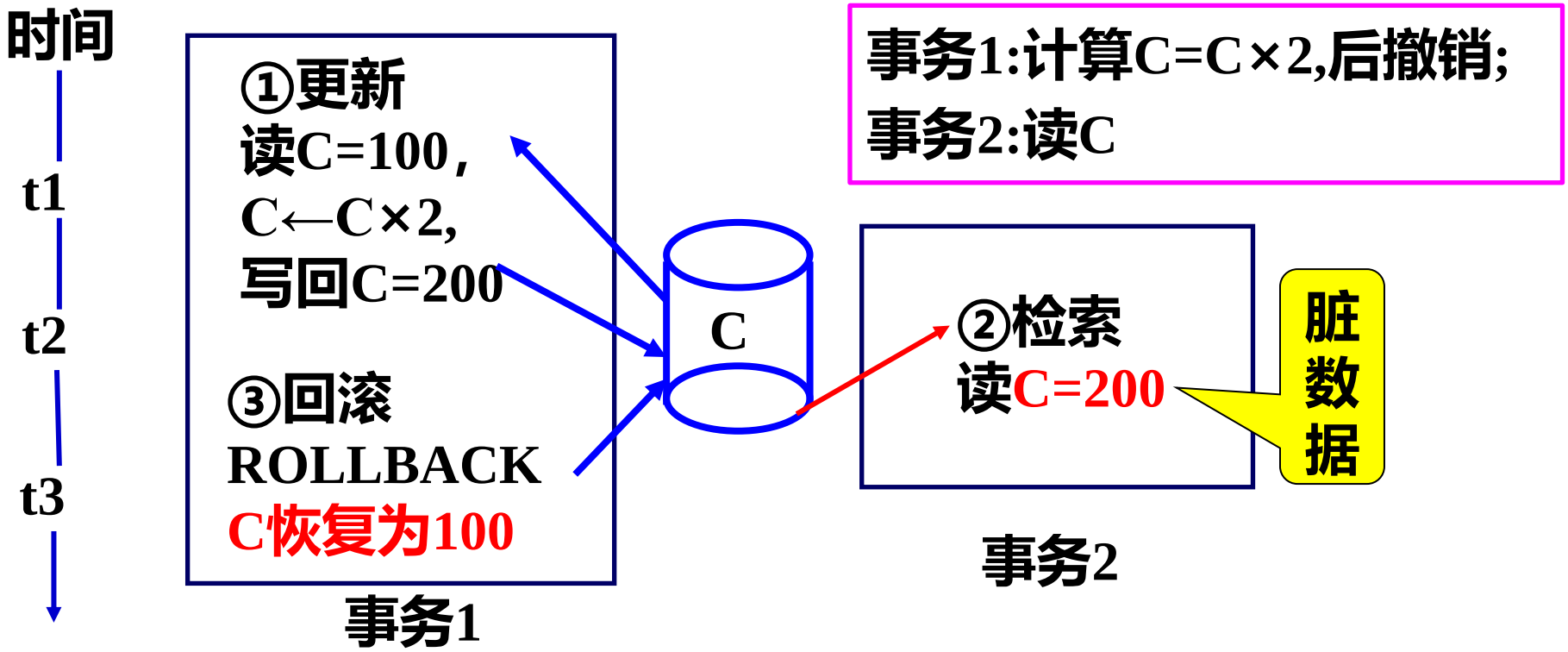
事务1:计算A+B并验算;
事务2:计算B=B×2



并发操作带来的三类数据不一致性

(3) 读脏数据 写-读冲突

- 事务1修改某一数据并写回磁盘;事务2读同一数据后,事务1由于某种原因被撤消，已修改过的数据恢复原值,事务2读到的数据与数据库中的数据不一致，是脏数据。



- 数据不一致性：并发操作破坏了事务的隔离性
- 并发控制就是用正确的方式调度并发操作，使一个事务的执行不受其他事务的干扰，从而避免造成数据的不一致性
- 并发控制的主要技术
  - 封锁(Locking)
  - 时间戳(Timestamp)
  - 乐观控制法
- 商用的DBMS一般都采用封锁方法

11.2 封锁

- 封锁是实现并发控制的一个非常重要的技术
- 封锁就是事务T在对某个数据对象（例如表、记录等）操作之前，先向系统发出请求，对其加锁
- 一个事务对某个数据对象加锁后究竟拥有什么样的控制由封锁的类型决定

1. 基本的封锁类型

排它锁（Exclusive locks，写锁，简记为X锁）

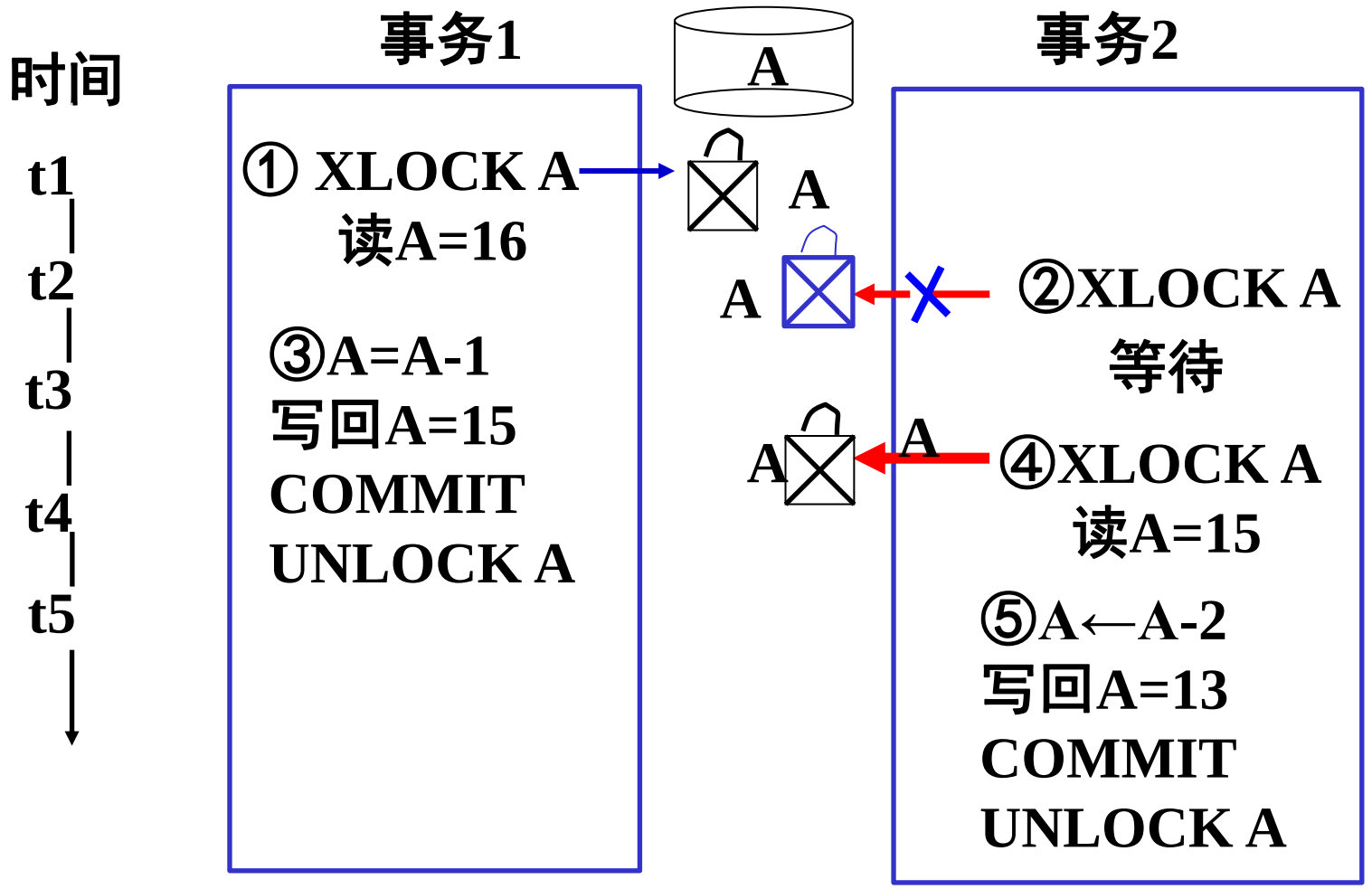
共享锁（Share locks，读锁，简记为S锁）

- **排它锁：**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，其它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T释放A上的锁之前**其它事务**不能再**读取和修改**A。
- **共享锁：**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，其它事务只能再对A加S锁，而不能加X锁，直到T释放A上的S锁。
  - 共享锁保证其它事务**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**。

- **封锁协议：运用X锁和S锁对数据对象加锁时约定的规则。**
  - 何时申请X锁或S锁
  - 持锁时间
  - 何时释放
- **对封锁方式规定不同的规则，就形成了各种不同的封锁协议，在不同的程度上保证并发操作的正确调度。**

- 一级封锁协议：事务T在修改数据R之前必须先对其加X锁，直到事务结束才释放。
  - 正常结束（COMMIT）
  - 非正常结束（ROLLBACK）
- 一级封锁协议可防止丢失修改。
- 一级封锁协议中，如果仅仅是读数据不对其进行修改，是不需要加锁的，所以它不能保证可重复读和不读“脏”数据。

使用封锁机制解决丢失修改问题



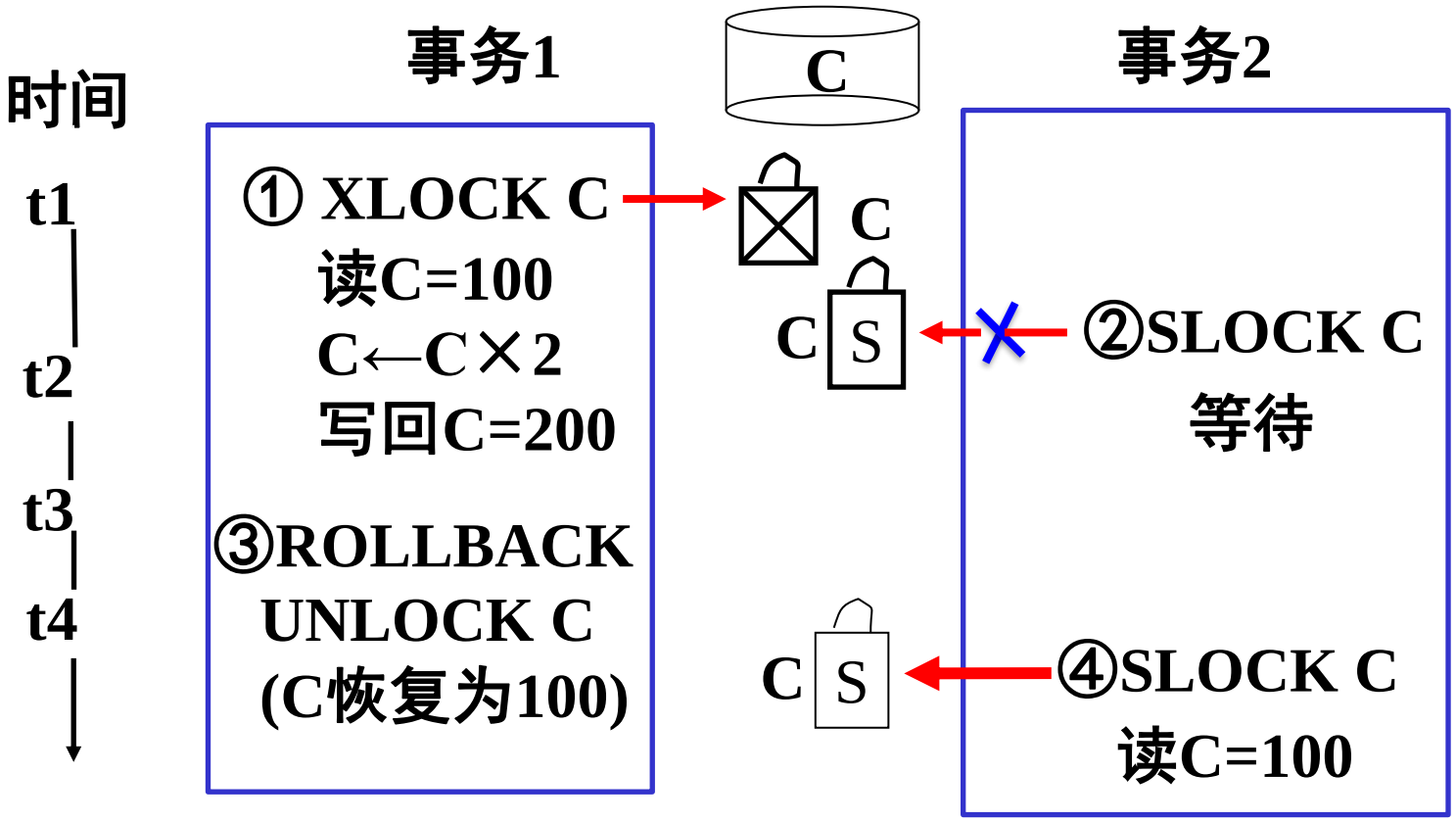
没有丢失修改

- 二级封锁协议：一级封锁协议加上事务T在读取数据R之前必须先对其加S锁，读完后即可释放S锁。
- 二级封锁协议可以防止丢失修改和读“脏”数据
- 二级封锁协议中，由于读完数据后即可释放S锁，所以它不能保证可重复读

使用封锁机制解决读“脏”数据问题

C=C×2
后rollback

不读“脏”数据



- 三级封锁协议：一级封锁协议加上事务T在读取数据R之前必须先对其加S锁，直到事务结束才释放
- 三级封锁协议可防止丢失修改、读脏数据和不可重复读。

使用封锁机制解决读不可重复读问题

时间	事务T1	事务T2	A、B的值
1	Slock (A，B)		50、100
2	读A=50，B=100		
3	求和150		
4		Xlock(B)等待	
5	读A=50，B=100	……	
6	求和150	……	
7	Commit	……	
8	Unlock(A，B)	……	
9		Xlock(B)	
10		读B=100	
11		B=B*2	
12		写回B=200	50、200
13		Commit	
14		Unlock(B)	

可重复读

11.2 封锁

	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		√		√	√	√	√

- 三级协议的主要区别：什么操作需要申请封锁以及何时释放锁（即持锁时间）
- 不同的封锁协议使事务达到的一致性级别不同，封锁协议级别越高，一致性程度越高

封锁技术可有效解决并发操作的一致性问题，但也带来一些新的问题。

1、活锁

某个事务**永远处于等待封锁**的状态。

T_1	T_2	T_3	T_4
lock R	.	.	.
.	lock R	.	.
.	等待	Lock R	.
Unlock	等待	.	Lock R
.	等待	Lock R	等待
.	等待	.	等待
.	等待	Unlock	等待
.	等待	.	Lock R

活锁的避免：采用“先来先服务”策略。

2、死锁

■ **事务间等待关系**：一个事务申请锁而未获准，则需等待其他事务释放锁，从而形成事务间的等待关系。

■ **死锁**：当事务出现循环等待时，如不加干预，则会一直等待下去，形成死锁。

T_1	T_2
Xlock R_1	.
.	.
.	Xlock R_2
.	.
Xlock R_2	.
等待	Xlock R_1
等待	等待

■ **解决死锁的二个方法**：

① 预防死锁；（不适合数据库）

② 死锁的诊断与解除。

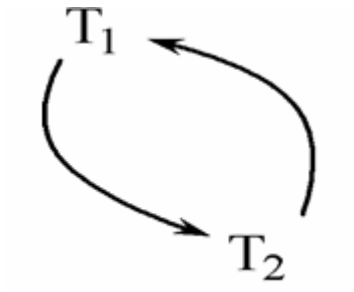
死锁的诊断与解除

1)超时法

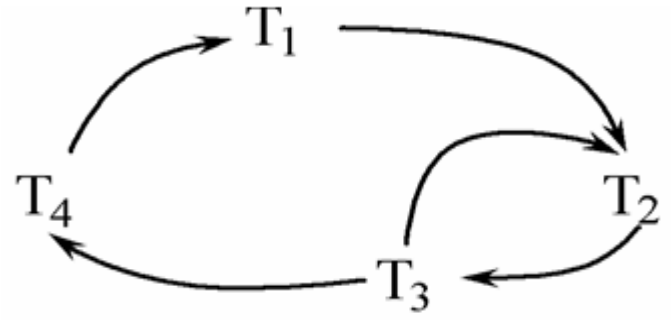
- 如果一个事务的等待时间超过了规定的时限，就认为发生了死锁
- **优点**：实现简单
- **缺点**
  - 有可能误判死锁
  - 时限若设置得太长，死锁发生后不能及时发现

2)等待图法

- 用事务等待图（是有向图）动态反映所有事务的等待情况
  - 结点 T_i 表示正运行的事务
  - 若T1等待T2，则T1、T2之间划一条有向边，从T1指向T2
- 并发控制子系统周期性地生成事务等待图，检测事务。如果发现图中存在**回路**，则表示系统中出现了**死锁**。



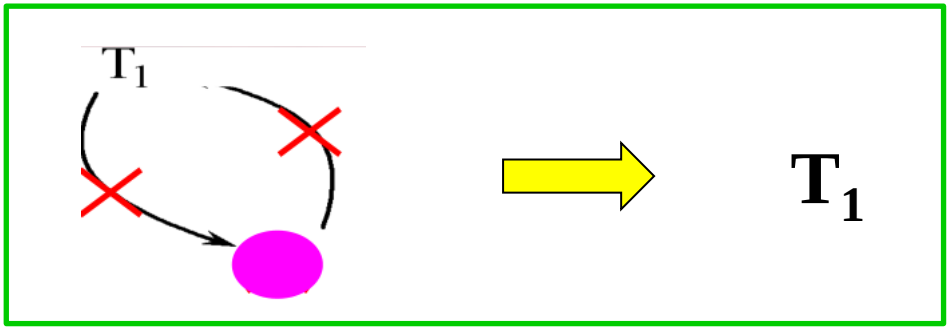
事务T1等待T2，T2等待T1，产生了死锁

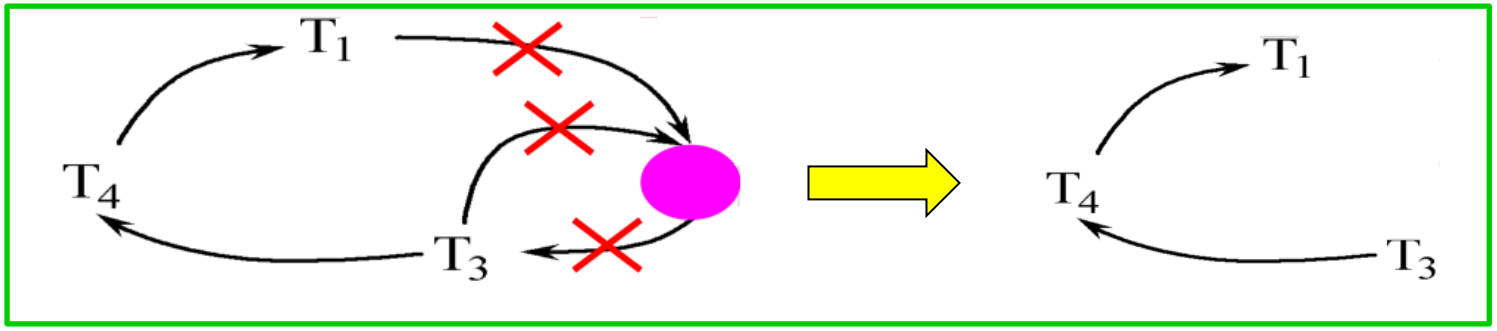


大回路中又有小的回路，2个死锁

解除死锁

- 在循环等待的事务中，选择一个处理死锁代价最小的事务，将其撤消。
- 释放此事务持有的所有锁，使其它事务能继续运行下去。





11.4 并发调度的可串行性

- 事务的调度：**事务的执行顺序称为一个调度**，表示事务的操作在系统中执行的时间顺序。
- 事务调度的要求：
  - 包含所有事务的操作
  - 一个事务中操作的顺序必须保持不变
- **串行调度**：
  - 属于同一事务的操作紧挨在一起
  - 对n个事务，可以有n!个有效调度
- **并行调度**：
  - 来自不同事务的操作可以交叉执行

11.4.1 可串行化调度

- DBMS对并发事务不同的调度可能会产生不同的结果

什么样的调度是正确的

串行调度是正确的

执行结果等价于串行调度的调度也是正确的，称为可串行化调度。

可串行化调度：多个事务的并行执行是正确的，当且仅当其结果与按**某一次序串行执行**这些事务时的**结果相同**

可串行性是并发事务正确调度的准则

11.4.1 可串行化调度

例：现在有两个事务，分别包含下列操作：

事务1：读B；A=B+1；写回A；

事务2：读A；B=A+1；写回B；

假设A和B的初值为2

串行调度

按T1→T2次序执行

时间	事务T1	事务T2
1	读B=2	
2	A=B+1=3	
3	写回A=3	
4		读A=3
5		B=A+1=4
6		写回B=4
7		

结果A=3，B=4

按T2→T1次序执行

时间	事务T1	事务T2
1		读A=2
2		B=A+1=3
3		写回B=3
4	读B=3	
5	A=B+1=4	
6	写回A=4	
7		

结果A=4，B=3

11.4.1 可串行化调度

可串行化调度与不可串行化调度

时间	事务T1	事务T2
1	读B=2	
2		读A=2
3	A=B+1=3	
4	写回A=3	
5		B=A+1=3
6		写回B=3
7		

结果A=3、B=3，是不可串行化调度，错误的调度

时间	事务T1	事务T2
1	读B=2	
2		等待
3		……
4	A=B+1=3	……
5	写回A=3	……
6		读A=3
		B=A+1=4
		写回B=4

结果A=3、B=4，是可串行化调度，是正确的调度

如何保证并发调度的正确性

两段锁协议

11.5 两段锁协议

□ 两段封锁协议

- 所有事务必须分**两个阶段**对数据项加锁和解锁：

① 对任何数据进行读写操作之前，事务首先要申请并获得对该数据的封锁

② 释放一个封锁之后，事务不再申请和获得任何其他封锁

□“两段”锁的含义: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**

Slock A Slock B Xlock C Unlock B Unlock A Unlock C;

扩展阶段 | 收缩阶段

遵守两段锁协议

Slock A Unlock A Slock B Xlock C Unlock C Unlock B

不遵守两段锁协议

若并发事务都遵守两段锁协议，则对这些事务的任何并发调度策略都是可串行化。

时间	事务T1	事务T2
1	Slock(B)	
2	读B=2	
3		Slock(A)
4		读A=2
5	Xlock(A)等待	
6	……	Xlock(B)等待
7		……

遵守两段锁协议的事务会不会发生死锁

会

课堂练习

1、在数据库系统中死锁属于__事务__故障。

2、封锁技术中主要有两种：排他型封锁和__共享__型封锁。

3、某个事务永远处于等待封锁的状态称为__活锁__，它的避免采用__先来先服务__策略。

4、数据库中把未提交的随后被撤销的数据称为__脏数据__。

5、并发操作带来的数据不一致性包括____、____和____。

6、一级封锁协议使用__X__锁解决丢失更新问题。

四、简述SQL基本表的完整性约束如何分类，并给出实现上述完整性约束的语法。

1、实体完整性约束

语法：Primary Key， 主码约束

2、参照完整性约束

语法：Foreign Key Reference ，外码约束

3、用户自定义完整性约束

语法：Not Null，非空约束

Unique，唯一约束

Check，检查约束

五、工厂要建立一个管理数据库系统,该系统包括以下实体（含属性）如下:

部门:部门号、部门名、电话、地址;

职工:职工号、职工名、性别、职务;

设备:设备号、名称、价格;

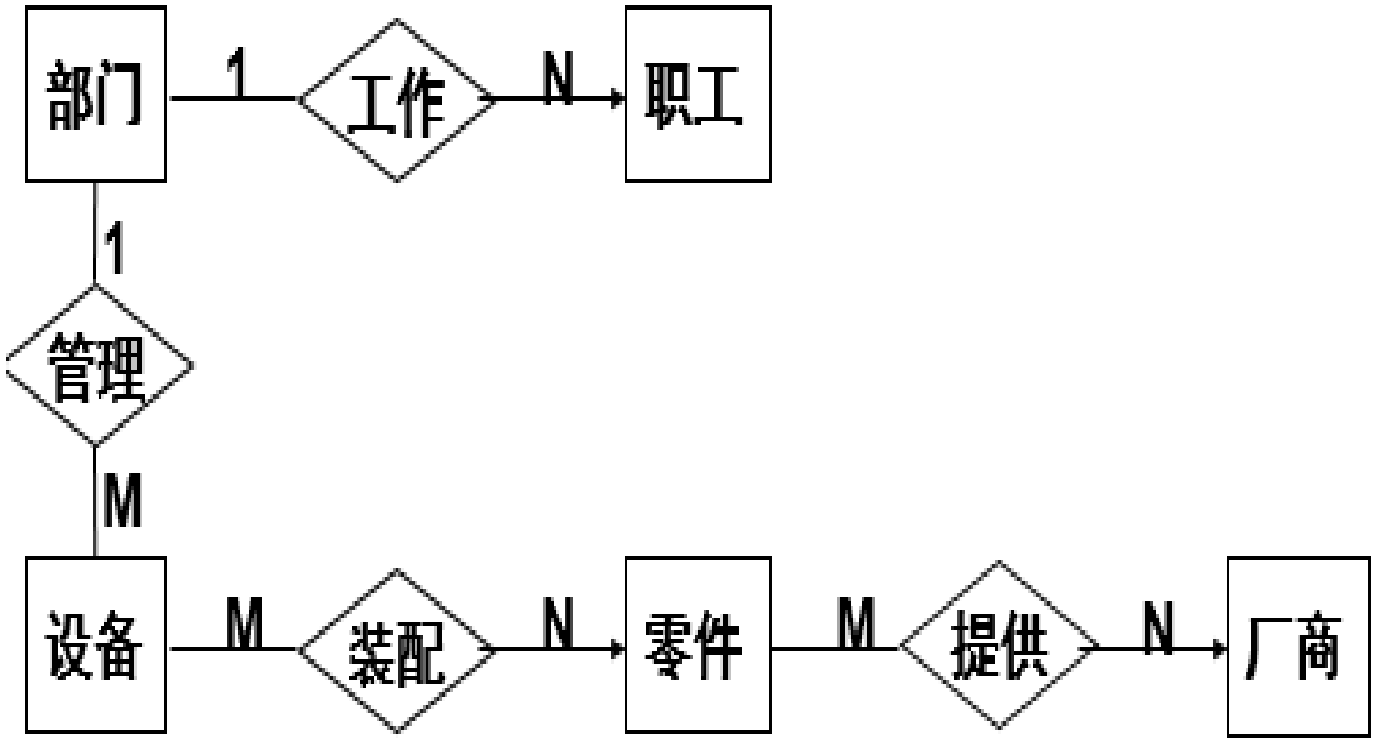
零部件:零部件号、名称、规格、价格;

厂商:单位号、名称、电话、地址。

其中,一个部门有多个职工,每个职工只能在一个部门工作;一个部门管理多台设备, 每台设备只属于一个部门;一台设备装配多种零部件,每种零部件可以装配在多台设备上;一个厂商可以提供多种零部件,每种零部件可以由多个厂商提供。

(1)根据上述语义画出ER图,要求在图中注明联系的类型,实体属性省略

(2)将ER模型转换成关系模式,并指出每个关系模式的主码(用下划线表示)和外码(用波浪线表示)。

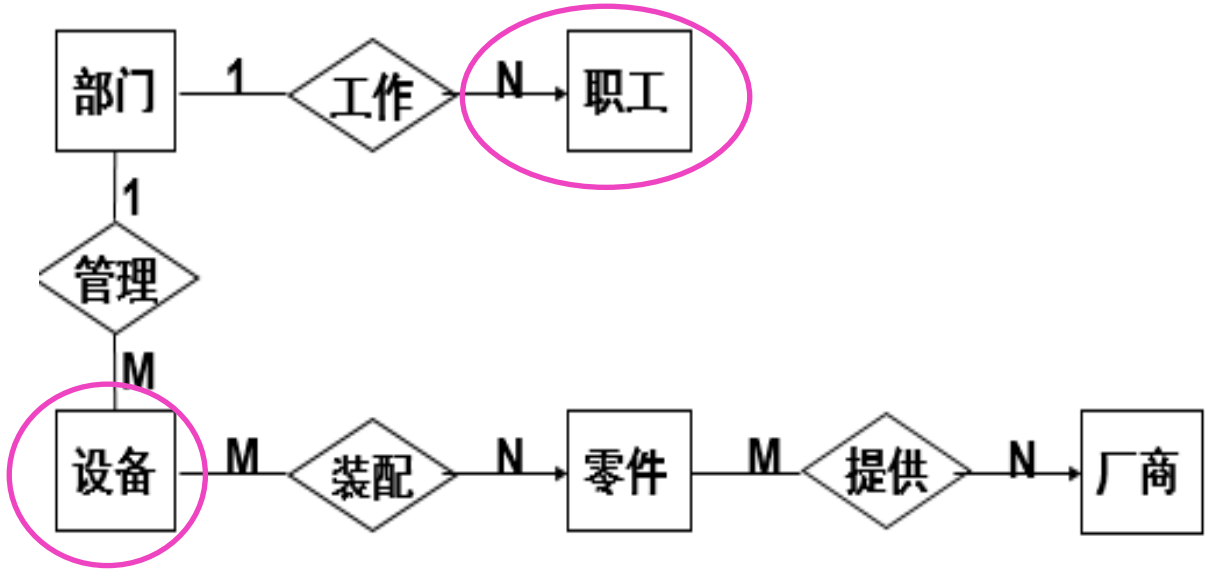


一个部门有多个职工，每个职工只能在一个部门工作；

一个部门管理多台设备，每台设备只属于一个部门；

一台设备装配多种零部件，每种零部件可以装配在多台设备上；

一个厂商可以提供多种零部件，每种零部件可以由多个厂商提供。



部门(<u>部门号</u>，部门名，电话，地址)

职工(<u>职工号</u>，职工名，性别，职务，部门号)

设备(<u>设备号</u>，名称，价格，部门号)

零部件(<u>零部件号</u>，名称，规格，价格)

厂商(<u>单位号</u>，名称，电话，地址)

装配(<u>设备号，零部件号</u>)

提供（<u>零部件号，单位号</u>）

红色的属性是外码

六、设一个图书借阅管理数据库中包括三个关系模式:

图书(图书编号,书名,作者,出版社,单价)

book(bno,bname,author,press,price)

读者(借书证号,姓名,地址)

reader(rno,rname,address)

借阅(借书证号,图书编号,借阅日期,归还日期)

borrow(rno,bno,bdate,rdate)

1、用关系代数检索借阅“数据库系统概论”的读者姓名。

$$\prod_{rname}(\sigma_{bname='数据库系统概论'}(book \bowtie reader \bowtie borrow))$$

2、用SQL语句将图书编号为1301的图书单价减少10%

```
Update book Set price=price*0.9  Where bno='1301'
```

book(bno,bname,author,press,price)

reader(rno,rname,address)

borrow(rno,bno,bdate,rdate)

3、用SQL语句查询各出版社图书最高价格、最低价格和平均价格。

```
select press,max(price),min(price),avg(price)
from book  group by press
```

4、用SQL语句查询李兰借阅的所有图书的书名及借阅日期。

```
Select  bname,bdate from book,reader,borrow
where rname='李兰' and reader.rno=borrow.rno
and borrow.bno=book.bno
```