UNIT 10 事务管理

本讲主要目标

学完本讲后，你应该能够了解：

1、事务是数据库操作的原子单位，具有原子性、一致性、隔离性和持久性等特性;

2、事务的并发导致的四种不一致性:丢失修改、不可重复读、读“脏”数据和幻像.

3、并发控制的目标是保证事务的隔离性，从而保证数据库的一致性

4、并发控制的正确性准则是事务并发执行的可串行化

5、三级封锁协议和两段锁协议的工作原理;

6、数据库恢复的目标是在故障发生时,确保事务的原子性和持久性;

7、DBMS提供备份机制、日志机制、检查点机制来协助数据库故障恢复;

本讲主要内容

一. 事务的定义与特性

二. 事务并发导致的四种不一致性

三. 封锁及封锁协议

四. 可串行性与两段锁协议

五. 封锁导致的问题 —活锁与死锁

六. 封锁粒度

七. 隔离级别

八. 数据库恢复的技术

一、事务的定义与特性（P160）

1、为什么要引入事务?

例：银行转帐：从帐号A中取出一万元，存入帐号B。

- 第一个操作是从帐号A中减去一万元，第二个操作是向帐号B中加入一万元
- 这两个操作要么全做，要么全不做如果只做一个操作则用户逻辑上就会发生错误，少了一万元，

一、事务的定义与特性

1、为什么要引入事务？

例：假定数据库系统自动检测参照完整性约束，有下面操作：

操作一：删除学号为“95002”的学生的信息；

操作二：插入选课信息（“95005”，“C8”，85）；

操作三：修改学号“95001”的学生的学号为“95008”；

回想一下Unit 8中这个例子的讲述。

一、事务的定义与特性

为何出现该问题？

- 参照完整性的检测时机是在关系数据库的原子操作之前或之后
- 一个SQL语句是关系数据库的原子操作；

如何解决该问题？

修改原子操作的定义 —— 可以将多个SQL语句定义为一个原子操作

事务

一、事务的定义与特性

2、事务的概念（P160）

- **事务**

-----是数据库的逻辑工作单位。是用户定义的一个数据库操作序列。这些操作要么全做要么全不做，是一个不可分割的工作单位。

- 事务是数据库并发控制和恢复的基本单位。

为什么数据库需要事务这个概念？

保证数据库状态的一致性

数据库系统与操作系统的基本单位不同

一、事务的定义与特性

3、定义事务的两种方式

- **显式方式**
  - 事务的开始由用户显式控制或DBMS自动隐含
  - 事务结束由用户显式控制

```
BEGIN TRANSACTION
COMMIT
ROLLBACK
```

- **隐式方式**
  - 当用户没有显式地定义事务时，由DBMS按缺省规定自动划分事务

一、事务的定义与特性

- 事务开始
  - BEGIN TRANSACTION
- 事务结束
  - ROLLBACK
    - 事务异常终止，回滚事务的所有操作.
    - 在事务运行的过程中发生了某种故障，事务不能继续执行.系统将事务中对数据库的所有已完成的更新操作全部撤消，滚回到事务开始时的状态
  - COMMIT
    - 事务正常结束，提交事务的所有操作
    - 使事务中所有对数据库的更新永久生效（保存在物理数据库中）。

一、事务的定义与特性

4、事务的特性—— ACID特性（P160）

- **原子性**（Atomicity）

  事务包含的一组更新操作是原子不可分的，即更新操作对于数据库而言，要么全做，要么全不做，不能部分地完成

- **一致性**（Consistency）

  事务执行的结果必须使数据库从一个一致性状态变到另一个一致性状态。

一、事务的定义与特性

- **隔离性**（Isolation）

  一个事务的执行不能被其它事务干扰。即一个事务的内部操作及使用的数据对其它并发事务是隔离的，不可见的。

- **持久性**（Durability）

  事务一旦提交，它对数据库中数据的改变就是永久性的，即使随后系统发生故障也能保持或恢复。

一、事务的定义与特性

破坏事务ACID特性的因素有：

（1）多个事务的并发运行

并发控制

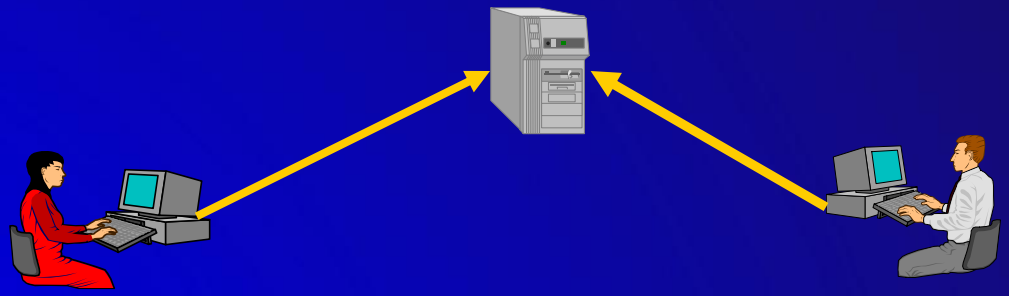
（2）事务的运行过程中被强行停止

故障恢复

一、事务的定义与特性

并发带来的不一致性的例子

例：假定两个顾客使用主副信用卡分别几乎同时刻在两台ATM上取款；



有可能两人各取出500元，而帐户上只减少500元。

导致此种情形出现的原因是什么？

本讲主要内容

一. 事务的定义与特性

二. 事务并发导致的四种不一致性

三. 封锁及封锁协议

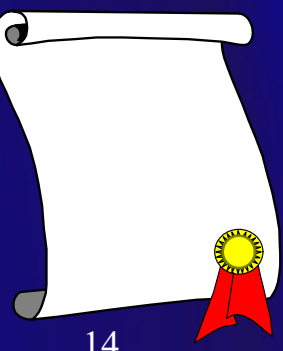
四. 可串行性与两段锁协议

五. 封锁导致的问题 —活锁与死锁

六. 封锁粒度

七. 隔离级别

八. 数据库恢复的技术



二、事务并发导致的四种不一致性

（参见教材P160-162）

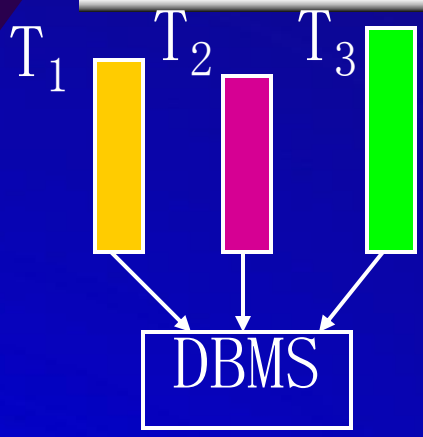
1、并发控制

为了有效地利用系统资源，发挥数据库共享资源的特点，即使是在集中式数据库系统中，多个事务也进行并发（concurrency）。

事物的隔离性要求事务内部的操作及数据的使用对并发的其它事务是隔离的。

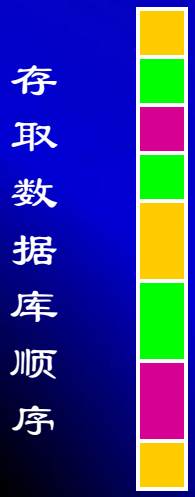
由于操作系统的基本操作单位——进程，与数据库系统的基本操作单位——事务不同，DBMS需要对事务进行并发控制，以保证数据库中数据的一致性。

二、事务并发导致的四种不一致性



并发控制的目标是什么？

保证事务的隔离性，
从而保证数据库的一致性



事务的并发操作何时会破坏事务的隔离性？

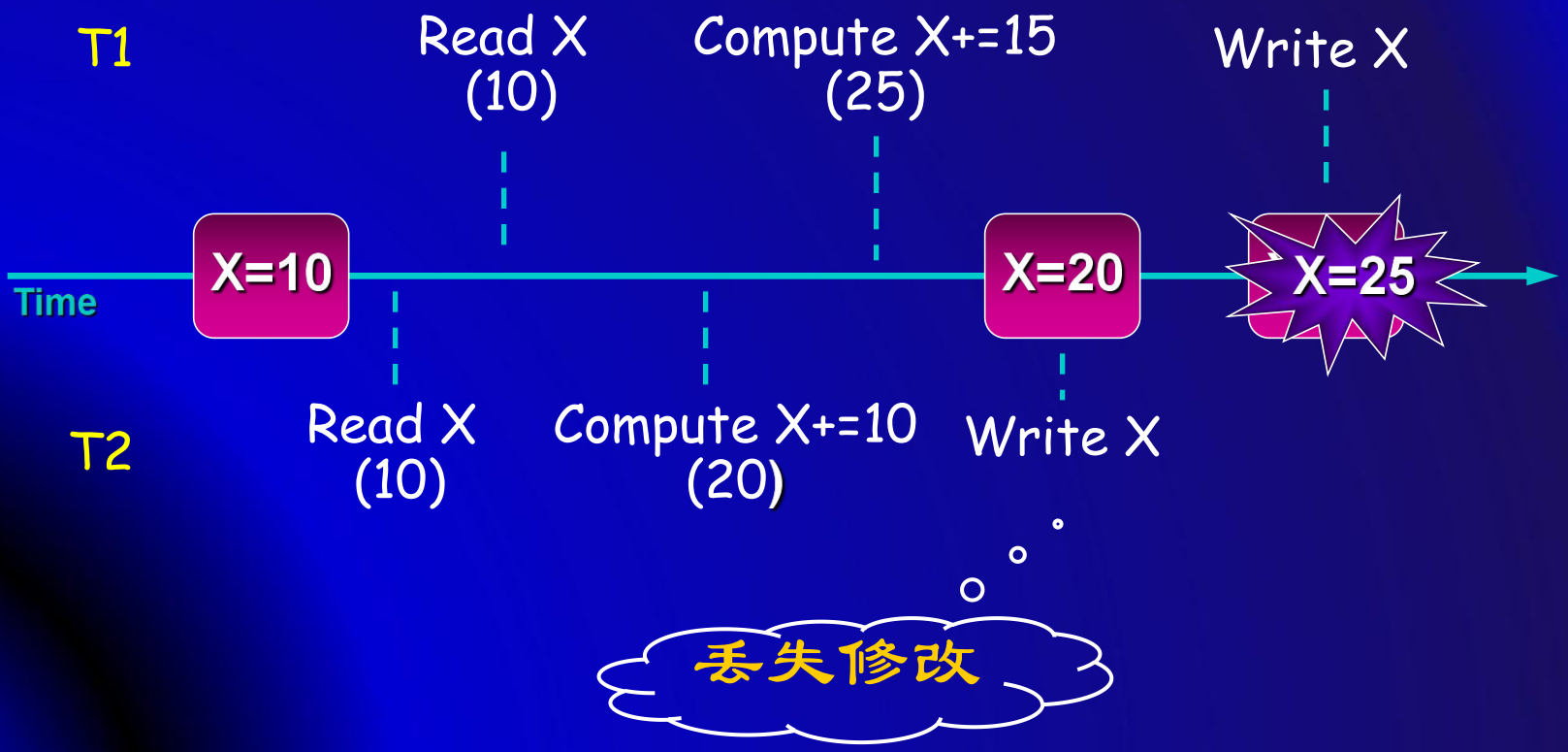
事务操作相同的数据并且
这些操作是冲突的

会出现哪些不一致性？

二、事务并发导致的四种不一致性

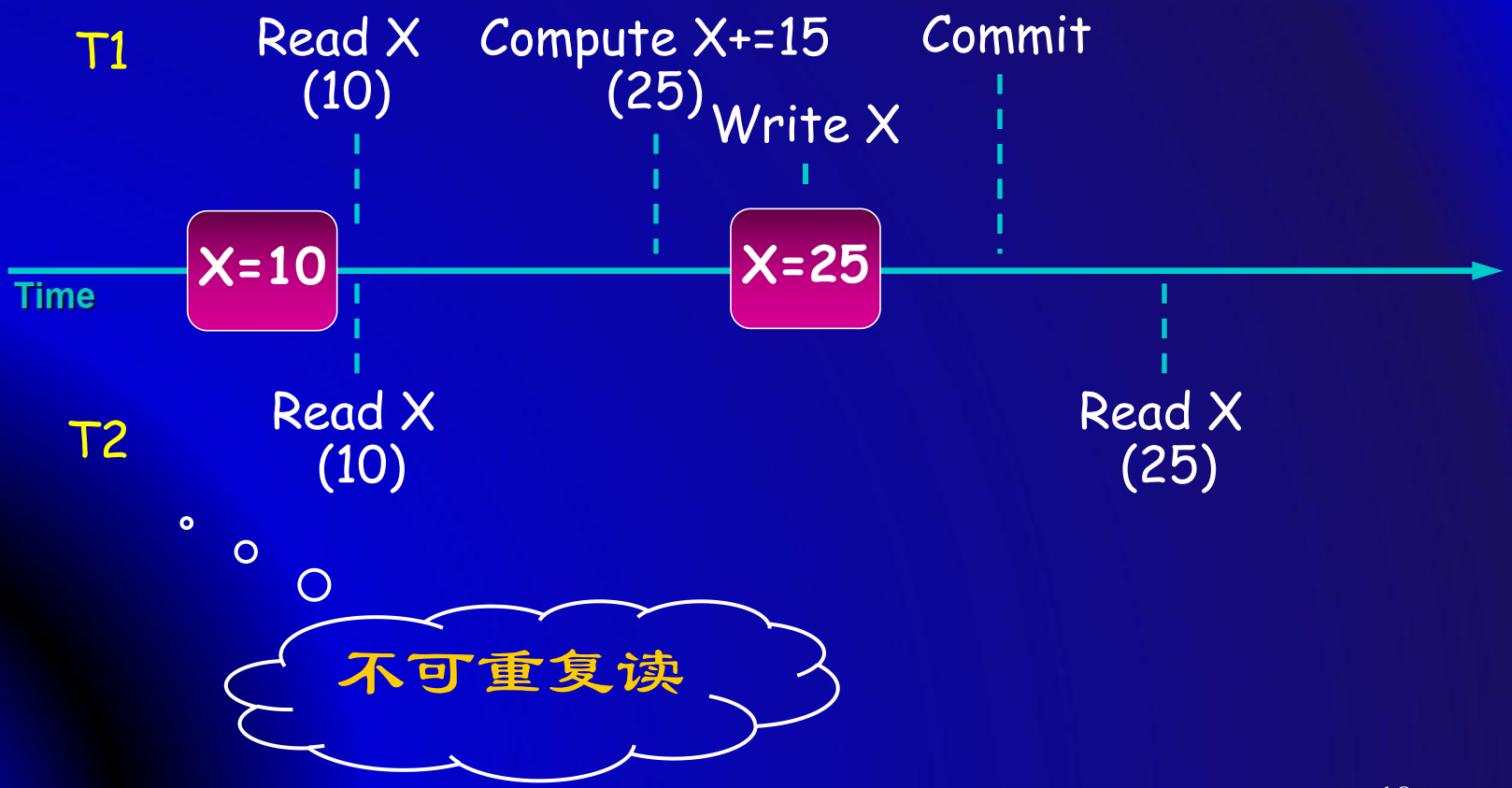
2、事务并发导致的不一致性

例：两个事务T1和T2读入同一数据并修改



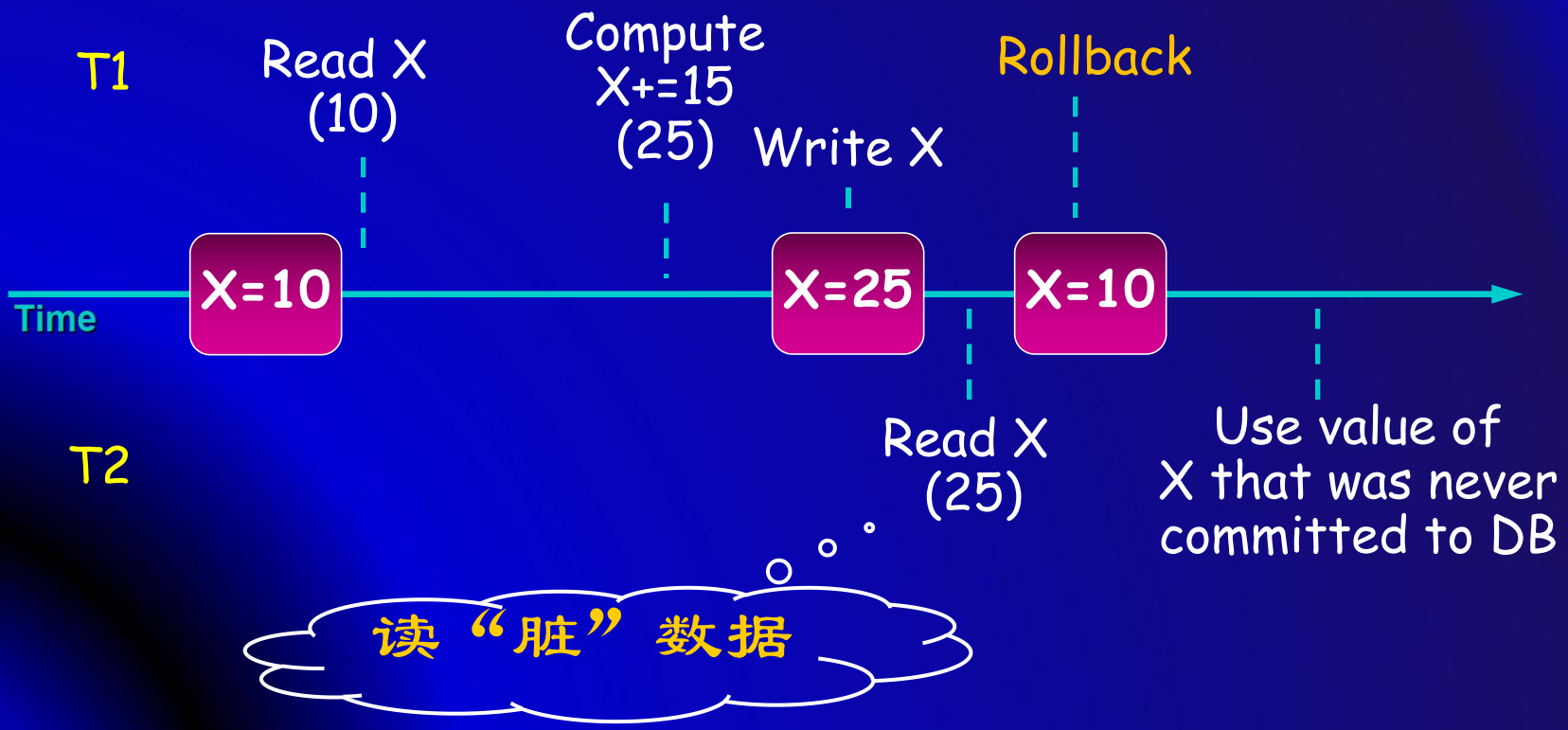
二、事务并发导致的四种不一致性

例：事务T_2读取数据X后，事务T_1更新该数据X，T_2再读X时，得到的结果不同



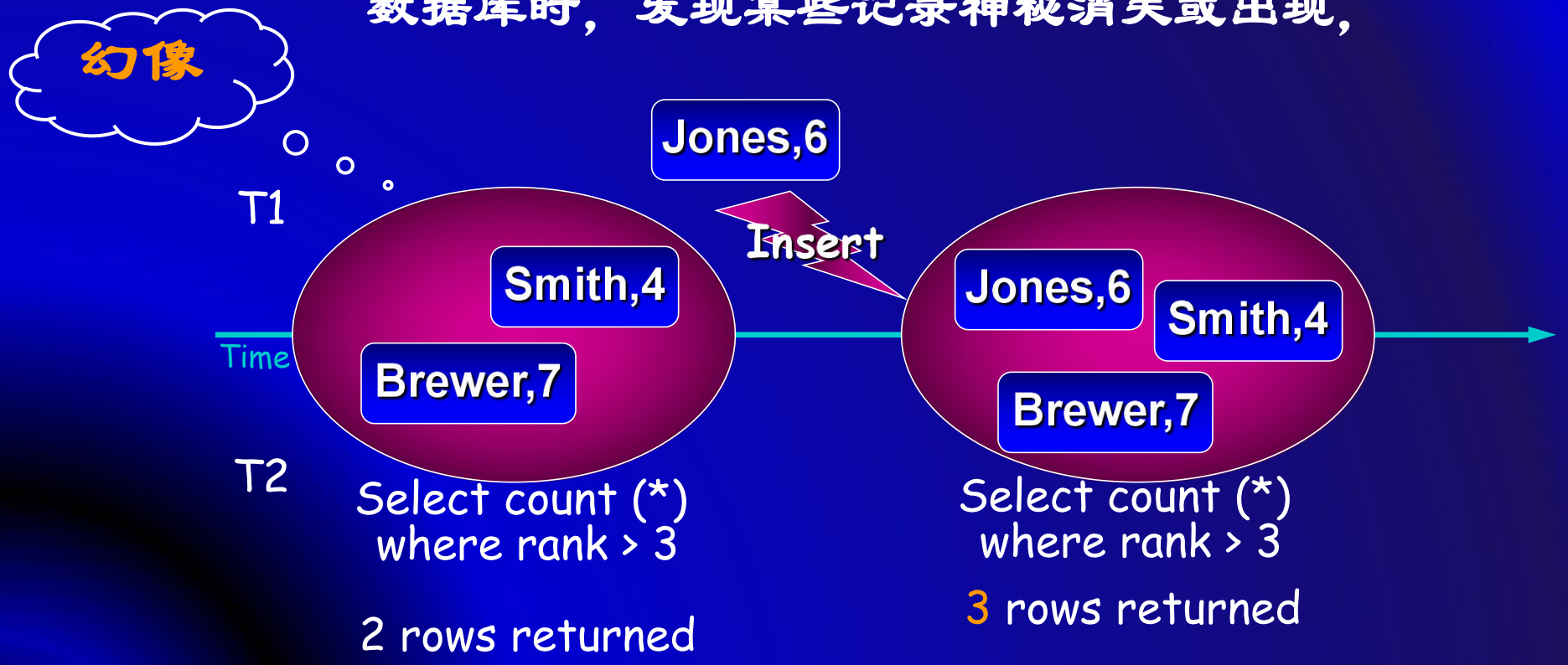
二、事务并发导致的四种不一致性

例：事务T1修改数据X，事务T2读取X后，T1被撤消，并将X恢复原值，T2读到的值与数据库不一致



二、事务并发导致的四种不一致性

例：事务T1按一定条件读数据库后，事务T2删除或插入了某些记录，当事务T1再次按相同条件读数据库时，发现某些记录神秘消失或出现，



二、事务并发导致的四种不一致性

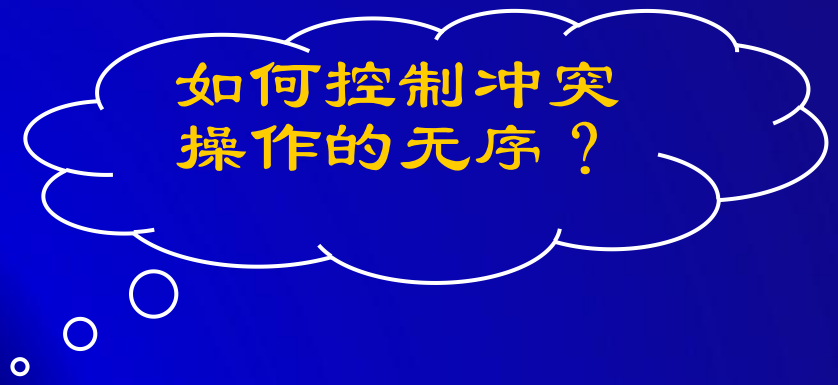
总结：事务并发导致的四个不一致性

- 丢失修改 (Lost Update)
- 不可重复读 (Non-Repeatable Read)
- 读“脏”数据 (Dirty Read)
- 幻像 (Phantom)

二、事务并发导致的四种不一致性

3、出现不一致性的原因

不同事务间冲突操作的无序执行。



二、事务并发导致的四种不一致性

4、并发控制的方法

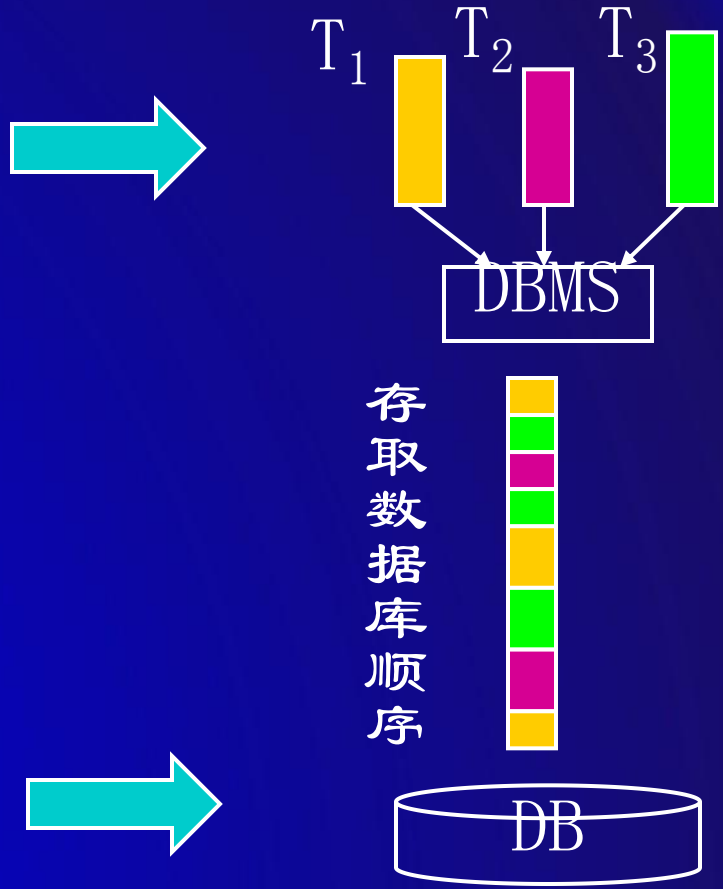
两种途径：

- 对事务规定一定的顺序

  —— **时间戳**

- 对共享的资源DB进行控制

  —— **加锁**



本讲主要内容

一. 事务的定义与特性

二. 事务并发导致的四种不一致性

三. 封锁及封锁协议

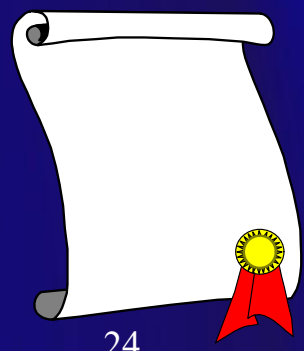
四. 可串行性与两段锁协议

五. 封锁导致的问题 —活锁与死锁

六. 封锁粒度

七. 隔离级别

八. 数据库恢复的技术



三、封锁及封锁协议

1、封锁的定义 （P163）

所谓封锁就是事务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。加锁后事务T就对该数据对象有了一定的控制，在事务T释放它的锁之前，其他事务对该数据的操作受到一定限制。

三、封锁及封锁协议

2、基本的封锁类型

- 排它锁 (Exclusive Locks, 简称X锁, 也称写锁)
- 共享锁 (Share Locks, 简称S锁, 也称读锁)

封锁类型的相容矩阵：

T_1 \ T_2	X	S	-
X	N	N	Y
S	N	Y	Y
-	Y	Y	Y

N=No, 不相容

Y=Yes, 相容

三、封锁及封锁协议

3. 封锁协议包含的内容

- **什么操作需要申请何种锁？**
  ---- 排它锁、共享锁

- **何时加锁？**
  ---- 事务开始前、操作开始前

- **何时解锁？**
  ---- 操作结束后、事务结束前

三、封锁及封锁协议

以下用的三种数据不一致问题的例子：

	T_1	T_2
1）	读A=16	
2）		读A=16
3）	A←A–1 写回A=15	
4）		A←A–1 写回A=15

丢失修改

	T_1	T_2
1）	读A=16 A←A*2 写回A=32	
2）		读A=32
3）	ROLLBACK A恢复为16	

读“脏”数据

	T_1	T_2
1）	读A=16	
2）		读A=16 A←A*2 写回A=32
3）	读A=32	

不可重复读

三、封锁及封锁协议

4. 一级封锁协议

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**。事务结束包括正常结束（COMMIT）和非正常结束（ROLLBACK）。

三、封锁及封锁协议

一级封锁协议功能

可防止丢失修改，并保证事务T是可恢复的

	T_1	T_2
1）	读A=16	
2）	A←A–1 写回A=15	申请A的X锁，等待
3）		读A=15
4）		A←A–1 写回A=14

丢失修改

√

	T_1	T_2
1）	读A=16 A←A*2 写回A=32	
2）		读A=32
3）	ROLLBACK A恢复为16	

读“脏”数据

×

	T_1	T_2
1）	读A=16	
2）		读A=16 A←A*2 写回A=32
3）	读A=32	

不可重复读

×

P165 图5-8 没有丢失修改

T_1	T_2
① Xlock A 获得	
② 读A=16	
	Xlock A
③A←A-1	等待
写回A=15	等待
Commit	等待
Unlock A	等待
④	获得Xlock A
	读A=15
	A←A-1
⑤	写回A=14
	Commit
	Unlock A

三、封锁及封锁协议

5. 二级封锁协议

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。

三、封锁及封锁协议

二级封锁协议功能

除了防止丢失修改，还可进一步防止读“脏”数据

	T_1	T_2
1）	读A=16	
2）	A←A–1 写回A=15	申请A的X锁，等待
3）		读A=15
4）		A←A–1 写回A=14

丢失修改

√

	T_1	T_2
1）	读A=16 A←A*2 写回A=32	
2）		申请A的S锁，等待
3）	ROLLBACK A恢复为16	

读“脏”数据

√

	T_1	T_2
1）	读A=16	
2）		读A=16 A←A*2 写回A=32
3）	读A=32	

不可重复读

×

三、封锁及封锁协议

6. 三级封锁协议

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、封锁及封锁协议

三级封锁协议功能

除了防止丢失修改和防止读“脏”数据外，还防止了不可重复读

	T_1	T_2
1）	读A=16	
2）	A←A–1 写回A=15	申请A的X锁，等待
3）		读A=15
4）		A←A–1 写回A=14

丢失修改 √

	T_1	T_2
1）	读A=16 A←A*2 写回A=32	
2）		申请A的X锁，等待
3）	ROLLBACK A恢复为16	

读“脏”数据 √

	T_1	T_2
1）	读A=16	
2）		申请A的X锁，等待
3）	读A=16	

不可重复读 √

三、封锁及封锁协议

7. 一、二、三级封锁协议的一致性保证

（见教材P166）

	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		√		√	√	√	√

本讲主要内容

一. 事务的定义与特性

二. 事务并发导致的四种不一致性

三. 封锁及封锁协议

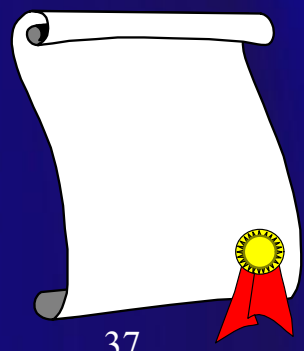
四. 可串行性与两段锁协议

五. 封锁导致的问题 —活锁与死锁

六. 封锁粒度

七. 隔离级别

八. 数据库恢复的技术



四、可串行性与两段锁协议

1. 并发调度的可串行性

系统对并发事务中并发操作的调度是随机的，哪个（些）调度是正确的？

能将数据库置于一致状态的调度

将所有事务串行起来的调度策略一定是正确的调度

与串行调度等价的调度是正确的调度

四、可串行性与两段锁协议

并发事务正确性准则 ----- 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**它们时的结果相同，我们称这种调度策略为**可串行化的调度**。

例如， 现在有两个事务T_1和T_2：

事务T_1 ：读B；A=B+1；写回A；

事务T_2 ：读A；B=A+1；写回B；

假定A，B的初值均为2，则

按$T_1 \rightarrow T_2$次序执行结果为A=3，B=4；

按$T_2 \rightarrow T_1$次序执行结果为A=4，B=3；

四、可串行性与两段锁协议

T_1	T_2
Slock B	
Y=B=2	
Unlock B	
Xlock A	
A=Y+1	
写回A(=3)	
Unlock A	
	Slock A
	X=A=3
	Unlock A
	Xlock B
	B=X+1
	写回B(=4)
	Unlock B

串行调度

T_1	T_2
	Slock A
	X=A=2
	Unlock A
	Xlock B
	B=X+1
	写回B(=3)
	Unlock B
Slock B	
Y=B=3	
Unlock B	
Xlock A	
A=Y+1	
写回A(=4)	
Unlock A	

串行调度

T_1	T_2
Slock B	
Y=B=2	
	Slock A
	X=A=2
Unlock B	
	Unlock A
Xlock A	
A=Y+1	
写回A(=3)	
	Xlock B
	B=X+1
	写回B(=3)
Unlock A	
	Unlock B

不可串行化调度

T_1	T_2
Slock B	
Y=B=2	
Unlock B	
Xlock A	
	Slock A
A=Y+1	等待
写回A(=3)	等待
Unlock A	等待
	X=A=3
	Unlock A
	Xlock B
	B=X+1
	写回B(=4)
	Unlock B

可串行化调度

四、可串行性与两段锁协议

什么样的方法能保证调度的可串行化？

两段锁协议

四、可串行性与两段锁协议

2. 两段锁协议

所谓**两段锁协议**是指所有事务必须分两个阶段对数据项加锁和解锁。

- 在对任何数据进行读、写操作之前，首先要申请并获得对该数据的封锁；
- 在释放一个封锁之后，事务不再申请和获得任何其他封锁。

“两段” ----- 获得（扩展）阶段和释放（收缩）阶段

四、可串行性与两段锁协议

两段锁协议示例

设T1，T2是如下的三个事务

T1：A：=A*B+2

T2：B：=A*2

基于两段锁协议，试给出一个可串行化调度

设A的初值为2，B的初值为4

T1	T2
	Slock(A)
	Read(A)
	Xlock(B)
Slock(B)	
等待	Write(B)
等待	Ulock(B)
等待	Ulock(A)
Slock(B)	
Read(B)	
Xlock(A)	
Write(A)	A=10
Ulock(A)	B=4
Ulock(B)	

四、可串行性与两段锁协议

- 两段锁协议**是可串行化的充分条件**，而不是必要条件

  例：P167 图5011

- 两段锁协议**可能导致死锁**

例：

T_1	T_2
Slock B	
读B=2	
	Slock A
	读A=2
Xlock A	
等待	Xlock B
等待	等待

本讲主要内容

一. 事务的定义与特性

二. 事务并发导致的四种不一致性

三. 封锁及封锁协议

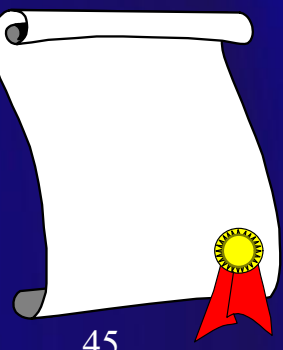
四. 可串行性与两段锁协议

五. 封锁导致的问题 —活锁与死锁

六. 封锁粒度

七. 隔离级别

八. 数据库恢复的技术



五、封锁导致的问题—活锁与死锁

封锁的方法可能引起活锁和死锁等问题

1、活锁（见教材P167-169）

如果事务T_1封锁了数据R，事务T_2又申请封锁R，于是T_2等待。T_3也请求封锁R，当T_1释放了R上的封锁之后系统首先批准了T_3的请求，T_2仍然等待。然后T_4又申请封锁R，当T_3释放了R上的封锁之后系统又批准了T_4的请求……T_2有可能永远等待。

五、封锁导致的问题—活锁与死锁

2、死锁

- **死锁的定义**

如果事务T_1封锁了数据R_1，事务T_2封锁了R_2，然后，T_1请求封锁R_2，等待。接着T_2请求封锁R_1，等待。这样就出现了T_1等待T_2，而T_2又在等待T_1的局面。这两个事务永远不能结束，形成死锁。

五、封锁导致的问题—活锁与死锁

- **死锁的诊断与解除**

诊断：

- **超时法**

—— 超过规定的时间就认为发生死锁

不足：可能误判死锁，也可能不能及时发现死锁

- **等待图法**

—— 事务为结点，边表示事务等待的情况，若T_1等待T_2，则$T_1 \rightarrow T_2$。系统周期性地检测事务等待图，若图中存在回路，则出现死锁。

解除：

选择一个处理死锁代价最小的事务，撤消。

本讲主要内容

一. 事务的定义与特性

二. 事务并发导致的四种不一致性

三. 封锁及封锁协议

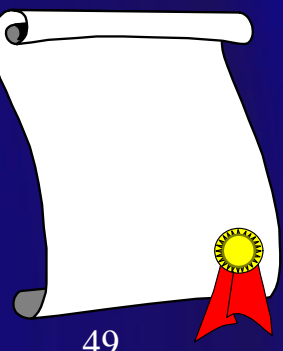
四. 可串行性与两段锁协议

五. 封锁导致的问题 —活锁与死锁

六. 封锁粒度

七. 隔离级别

八. 数据库恢复的技术



六、封锁的粒度

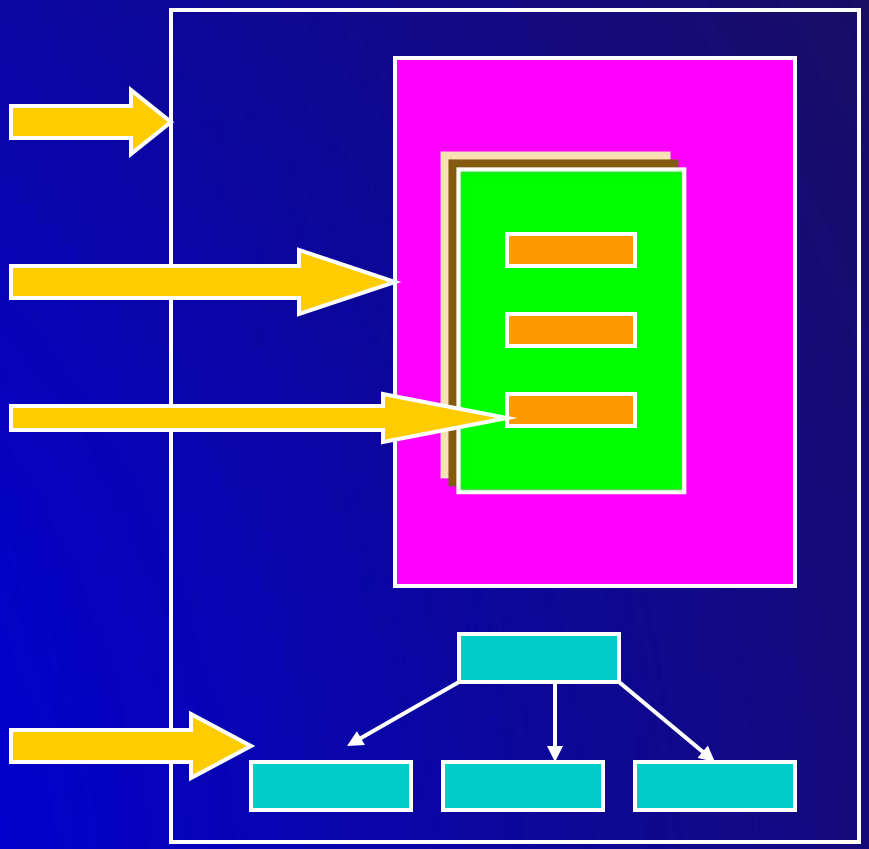
1. 封锁的粒度 （见教材P164）

整个数据库

关系

元组

索引



六、封锁的粒度

2. 封锁粒度的选择

封锁粒度与系统的并发度和并发控制的开销密切相关。

封锁粒度越大

并发度越小

系统开销越小

不同的事务选择不同的封锁粒度——多粒度封锁

选择封锁粒度时应该同时考虑并发度和开销两个因素

六、封锁的粒度

3. 多粒度封锁

----- 在一个系统中同时支持多种封锁粒度供不同的事务选择。

多粒度封锁协议允许多粒度树中的每个结点被独立地加锁。对每个结点的加锁意味着这个结点的所有后裔结点也被加以同样类型的锁。

- 显式封锁

  -- 应事务的要求直接加到数据对象上的封锁

- 隐式封锁

  -- 该数据对象没有独立加锁，是由于其上级结点加锁而使该数据对象加上了锁

六、封锁的粒度

判断能否对一个数据对象加锁：

（1）检查该数据对象有无显式封锁与之冲突；

（2）检查其所有上级结点，看本事务的显式封锁是否与该数据对象上的隐式封锁冲突；

（3）检查其所有下级结点，看上面的显式封锁是否与本事务的隐式封锁冲突。

六、封锁的粒度

4. 意向锁（为了减少对子孙节点加锁情况的搜索，引入意向锁）

---- 如果对一个结点加意向锁，则说明该结点的下层结点正在被加锁；对任一结点加锁时，必须先对它的上层结点加意向锁。

- 意向共享锁（简称IS锁）

  -- 如果对数据对象加IS锁，表示它的后裔结点拟加S锁

- 意向排它锁（简称IX锁）

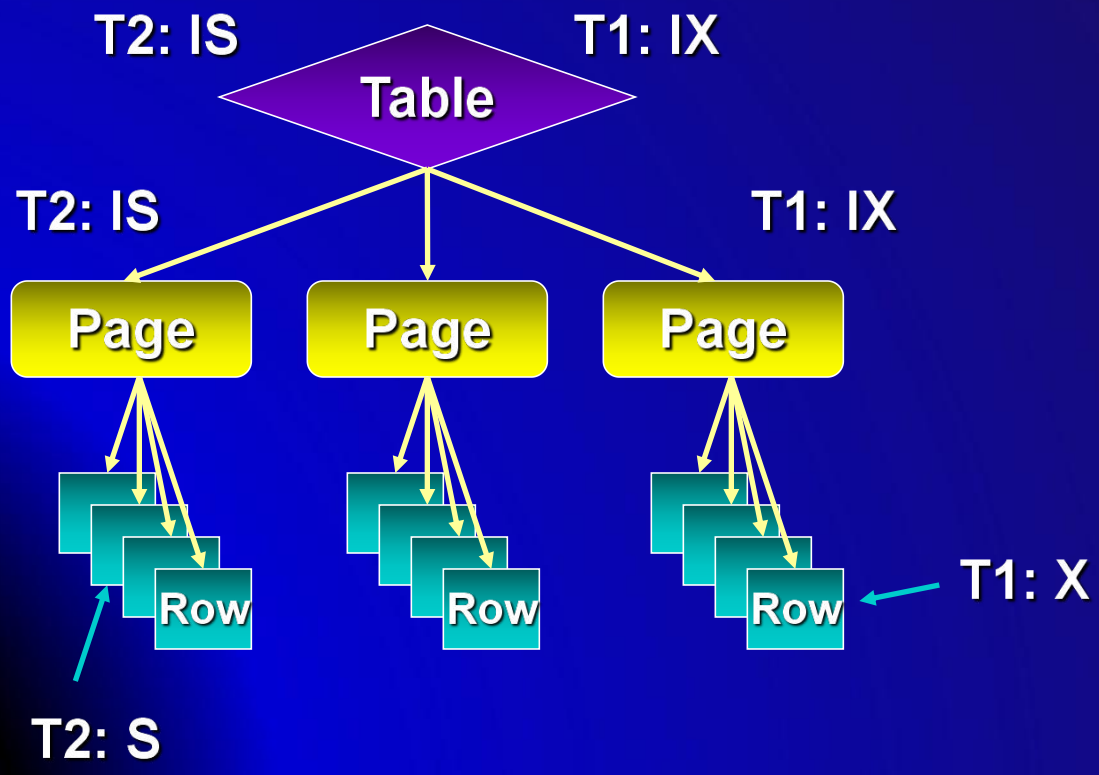
  -- 如果对数据对象加IX锁，表示它的后裔结点拟加X锁

- 共享意向排它锁（简称SIX锁）

  -- 如果对数据对象加SIX锁，表示先加S锁，再加IX

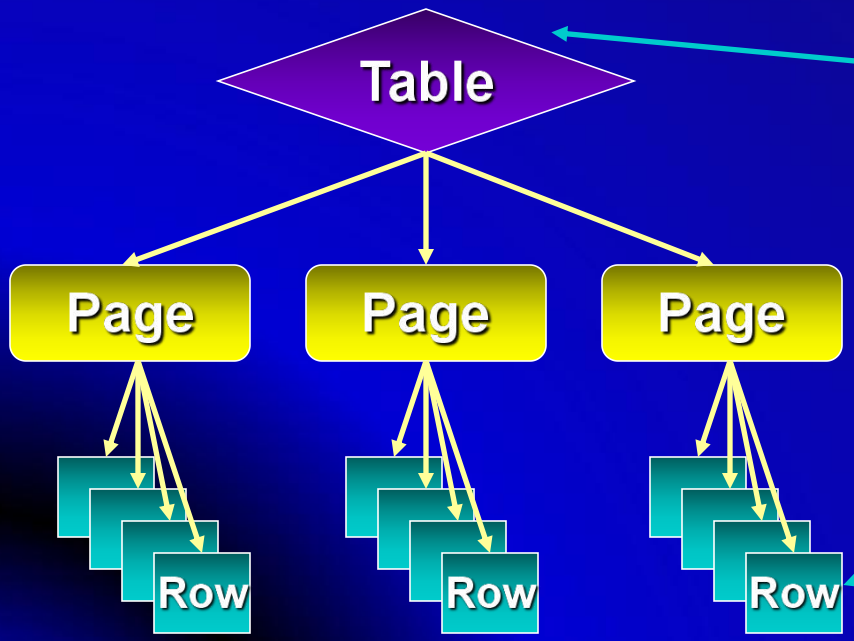
六、封锁的粒度

意向锁图示：



六、封锁的粒度

意向锁图示：



T4: Locks table in "X" mode

T3: Locks row in "X" mode

本讲主要内容

一. 事务的定义与特性

二. 事务并发导致的四种不一致性

三. 封锁及封锁协议

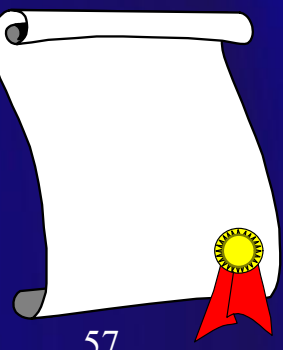
四. 可串行性与两段锁协议

五. 封锁导致的问题 —活锁与死锁

六. 封锁粒度

七. 隔离级别

八. 数据库恢复的技术



七、隔离级别

1、隔离级别

- 事务的隔离性：事务在并发执行时应该相互独立互不干扰。
  - 隔离性与并发性能是一对矛盾，在实践中有时候会放松隔离性的要求，通过“隔离性级别”来权衡事务的隔离性和并发能力。

- 事务的隔离性级别：又称事务的一致性级别，是一个事务必须与其它事务实现隔离的程度，是事务可接受的数据不一致程度。

七、隔离级别

SQL-92 定义了下列四种隔离级别（SQL Server支持所有这些隔离级别）：

- 未提交读（事务隔离的最低级别，仅可保证不读取物理损坏的数据）
- 提交读（SQL Server 默认级别）
- 可重复读
- 可串行读（事务隔离的最高级别，事务之间完全隔离）。

七、隔离级别

SQL92定义了四种隔离性级别

- 读未提交（Read uncommitted）
  - 最低的隔离级别，一个事务可以读到另外一个事务未提交的数据，不允许丢失修改，接受读脏数据和不可重复读现象。
- 读已提交（Read committed）
  - 若事务还没提交，其他事务不能读取该事务正在修改的数据。不允许丢失修改和读脏数据，接受不可重复读现象。
- 可重复读（Repeatable read）
  - 事务多次读取同一数据对象的结果一致。不允许丢失修改、读脏数据和读不一致，接受幻影读现象。
- 可串行化（Serializable）
  - 最高级别的隔离性，保证可串行化，不允许丢失修改、读脏数据、读不一致以及幻影读现象的发生。

七、隔离级别

2、隔离级别允许不同类型的行为

隔离级别	丢失修改	脏读	不可重复读取	幻像
未提交读	否	是	是	是
提交读	否	否	是	是
可重复读	否	否	否	是
可串行读	否	否	否	否

七、隔离级别

3、SQL Server设置隔离级别

SET TRANSACTION ISOLATION LEVEL

语法：

```
SET TRANSACTION ISOLATION LEVEL
    { READ COMMITTED
        | READ UNCOMMITTED
        | REPEATABLE READ
        | SERIALIZABLE
    }
```

本讲主要内容

一. 事务的定义与特性

二. 事务并发导致的四种不一致性

三. 封锁及封锁协议

四. 可串行性与两段锁协议

五. 封锁导致的问题 —活锁与死锁

六. 封锁粒度

七. 隔离级别

八. 数据库恢复的技术

八、数据库恢复的技术

（见教材P169-178）

1、数据库恢复的目标

---- 把数据库从**错误状态**恢复到某一个已知的**正确状态**（亦称为一致状态或完整状态）的功能。

数据库恢复的目标是在故障发生时，确保事务的原子性和持久性。

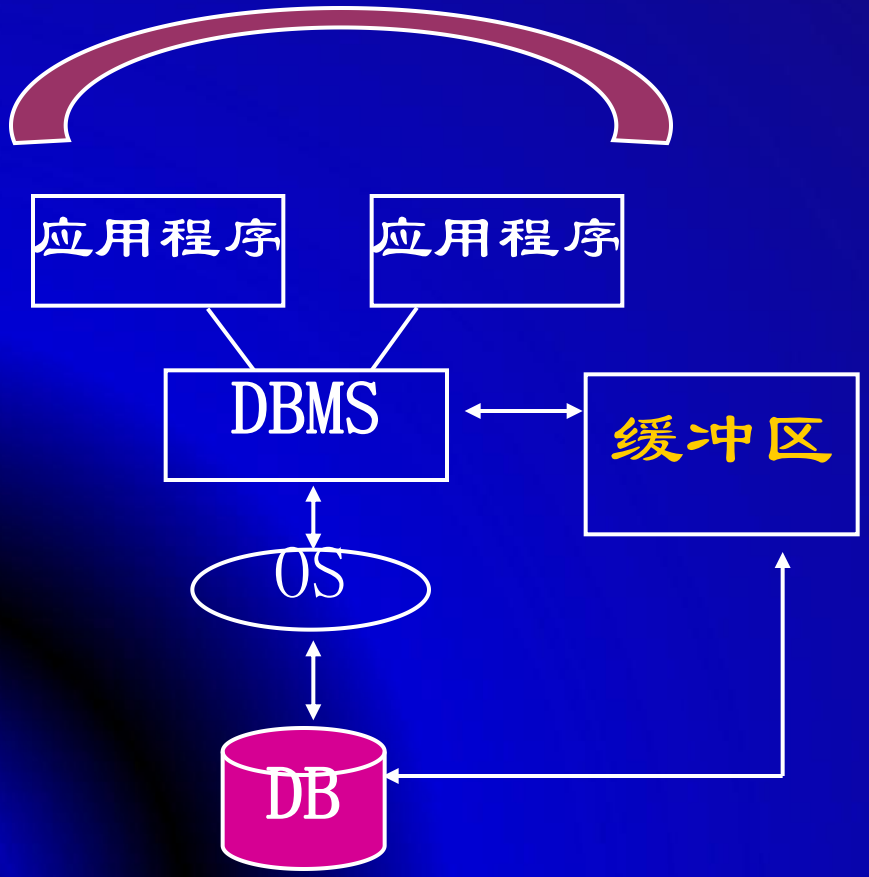
何时需要对数据库进行恢复？

数据库恢复有何特点？

故障

八、数据库恢复的技术

2、数据库恢复的特点



因为DB与内存用户工作区之间的数据交换是通过缓冲区进行的，而这个交换一般是以缓冲区是否满来触发的。因此，有可能提交事务的数据仍在缓冲区而没有写到DB中，而未提交事务的数据却写到了DB中。所以，故障恢复时，可能既要REDO已经提交了的事务，又要UNDO未提交的事务，以保证事务的原子性

八、数据库恢复的技术

3、数据库恢复的基本原理

数据库恢复就是利用存储在系统其他地方的**冗余数据**来修复数据库中被破坏的或不正确的数据

恢复的实现技术　　恢复的实现策略

八、数据库恢复的技术

4、建立冗余数据最常用的技术

- **数据转储**

  ---- 即DBA定期地将整个数据库复制到磁带或另一个磁盘上保存起来的过程。这些备用的数据文本称为**后备副本**或**后援副本**。

- **登记日志文件**

  ---- 是用来记录事务对数据库的更新操作的文件。

八、数据库恢复的技术

5、登记日志文件

- 内容
  - 各个事务的开始标志（BEGIN TRANSACTION）
  - 各个事务的结束标志（COMMIT 或 ROLLBACK）
  - 各个事务的所有更新操作
- 格式

事务标识	操作类型	操作对象	旧值	新值
T_1	BEGIN			
T_2	INSERT	X	NULL	2
T_1	UPDATE	Y	5	7
T_1	DELETE	Z	abc	NULL
T_1	COMMIT			

八、数据库恢复的技术

6、系统故障的恢复实现策略

◆**系统故障导致的数据库不一致**

（1）未完成的事务对DB产生影响

（2）已完成的事务在缓冲区的内容未写入DB

◆**恢复功能**

（1）UNDO未完成的事务

（2）REDO已完成的事务

◆**恢复步骤**

（1）正向扫描日志文件，建立UNDO和REDO队列；

（2）反向扫描日志文件，对每个UNDO事务的更新执行逆操作；

（3）正向扫描日志文件，对每个REDO事务的更新重新执行。

需要REDO日志中所有已完成的事务吗？是否一部分故障发生点前很久的事务不需要REDO？如何确定这些事务？

由此引入了检查点恢复技术

八、数据库恢复的技术

7、检查点技术

检查点（checkpoint）是使物理数据文件与数据库高速缓存当前状态同步的一种操作，方法是将所有高速缓存中的被修改的数据页写回到磁盘，这确保了对数据的一份永久拷贝（在磁盘上）。

检查点将最小化必须REDO的事务数量。

Questions?

本讲主要目标

学完本讲后，你应该能够了解：

1、事务是数据库操作的原子单位，具有原子性、一致性、隔离性和持久性等特性;

2、事务的并发导致的四种不一致性:丢失修改、不可重复读、读“脏”数据和幻像.

3、并发控制的目标是保证事务的隔离性，从而保证数据库的一致性

4、并发控制的正确性准则是事务并发执行的可串行化

5、三级封锁协议和两段锁协议的工作原理;

6、数据库恢复的目标是在故障发生时,确保事务的原子性和持久性;

7、DBMS提供备份机制、日志机制、检查点机制来协助数据库故障恢复;

问题讨论

1、单个SQL语句与操作系统的原子操作相比，哪个操作单位大？

2、如果数据库系统不能保证单个SQL语句的原子性，数据库的更新操作会导致数据库的不一致性吗？

3、如果数据库不能保证事务操作的原子性，数据库的更新操作会导致数据库的不一致性吗？

练习

教材：《数据库系统原理教程》（第2版）

P178

8、9、10、11、12、13、14、16、17