



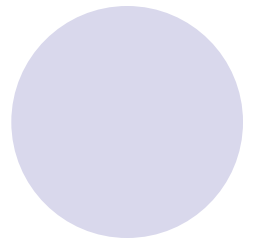
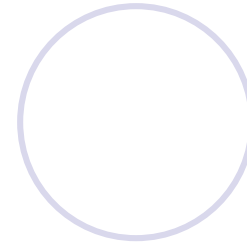
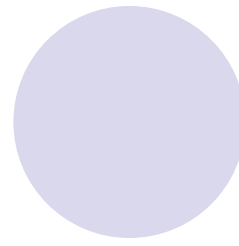
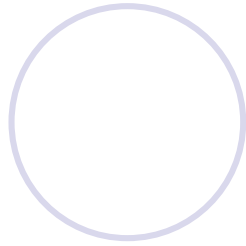
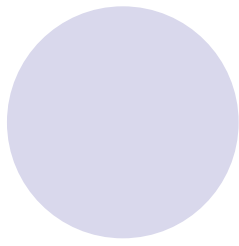
数据库系统

Database System

主讲：张仲楠 教授

Email: zhongnan_zhang@xmu.edu.cn

Office: 海韵A416



数据库系统

Database System

第十章 数据库恢复技术

第十章 数据库恢复技术

10.1 事务的基本概念

10.2 数据库恢复概述

10.3 故障的种类

10.4 恢复的实现技术

10.5 恢复策略

10.6 具有检查点的恢复技术

10.7 数据库镜像

10.8 小结

10.1 事务的基本概念

一、事务定义

二、事务的特性

1.事务

- 事务(Transaction)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。
- 事务和程序是两个概念
 - 在关系数据库中，一个事务可以是一条SQL语句，一组SQL语句或整个程序
 - 一个程序通常包含多个事务
- 事务是恢复和并发控制的**基本单位**

定义事务

● 显式定义方式

BEGIN TRANSACTION

SQL 语句1

SQL 语句2

○ ○ ○ ○ ○

COMMIT

BEGIN TRANSACTION

SQL 语句1

SQL 语句2

○ ○ ○ ○ ○

ROLLBACK

- 事务正常结束
- 提交事务的所有操作（读+更新）
- 事务中所有对数据库的更新写回到磁盘上的物理数据库中

定义事务

● 显式定义方式

BEGIN TRANSACTION

SQL 语句1

SQL 语句2

○ ○ ○ ○ ○

COMMIT

BEGIN TRANSACTION

SQL 语句1

SQL 语句2

○ ○ ○ ○ ○

ROLLBACK

- 事务异常终止
- 事务运行的过程中发生了故障，不能继续执行
- 系统将事务中对数据库的所有已完成的操作全部撤销
- 事务回滚到开始时的状态

定义事务

● 显式定义方式

BEGIN TRANSACTION

SQL 语句1

SQL 语句2

○ ○ ○ ○ ○

COMMIT

BEGIN TRANSACTION

SQL 语句1

SQL 语句2

○ ○ ○ ○ ○

ROLLBACK

● 隐式方式

当用户没有显式地定义事务时，

数据库管理系统按缺省规定自动划分事务

二、事务的特性(ACID特性)

事务的ACID特性：

- 原子性（**A**tomicity）
- 一致性（**C**onsistency）
- 隔离性（**I**solation）
- 持续性（**D**urability）



1. 原子性

- 事务是数据库的逻辑工作单位

- 事务中包括的诸操作要么都做，要么都不做

2. 一致性

- 事务执行的结果必须是使数据库从一个一致性状态变到另一个一致性状态
- 一致性状态
 - 数据库中只包含成功事务提交的结果
- 不一致状态
 - 数据库系统运行中发生故障，有些事务尚未完成就被迫中断；
 - 这些未完成事务对数据库所做的修改有一部分已写入物理数据库，这时数据库就处于一种不一致(不正确)的状态

一致性

与原子性

银行转账：从帐号A中取出一万元，存入帐号B。

- 定义一个事务，该事务包括两个操作

A	B
$A=A-1$	$B=B+1$


- 这两个操作要么全做，要么全不做
 - 全做或者全不做，数据库都处于一致性状态。
 - 如果只做一个操作，数据库就处于不一致性状态。
- 可见一致性与原子性密不可分

3. 隔离性

对**并发执行**而言

一个事务的执行不能被其他事务干扰

- 一个事务内部的操作及使用的数据对其他并发事务是**隔离**的
- 并发执行的各个事务之间不能互相干扰



T_1	T_2
① 读A=16	
②	读A=16
③ $A \leftarrow A-1$ 写回A=15	
④	$A \leftarrow A-3$ 写回A=13

T1的修改被T2覆盖了！

4. 持续性

- 持续性也称永久性（**Permanence**）

- 一个事务一旦提交，它对数据库中数据的改变就应该是永久性的。
- 接下来的其他操作或故障不应该对其执行结果有任何影响。

事务的特性

- 保证事务**ACID**特性是事务处理的任务

- 破坏事务**ACID**特性的因素

(1) 多个事务并行运行时，不同事务的操作**交叉执行**

- 数据库管理系统必须**保证多个事务的交叉运行不影响**这些事务的隔离性

并发控制

(2) 事务在运行过程中被**强行停止**

- 数据库管理系统必须保证**被强行终止的事务对数据库和其他事务没有任何影响**

恢复机制

第十章 数据库恢复技术

10.1 事务的基本概念

10.2 数据库恢复概述

10.3 故障的种类

10.4 恢复的实现技术

10.5 恢复策略

10.6 具有检查点的恢复技术

10.7 数据库镜像

10.8 小结

10.2 数据库恢复概述

- 故障是不可避免的

- 计算机硬件故障
- 软件的错误
- 操作员的失误
- 恶意的破坏

- 故障的影响

- 运行事务非正常中断，影响数据库中数据的正确性
- 破坏数据库，全部或部分丢失数据

数据库恢复概述（续）

- 数据库的恢复

- 数据库管理系统必须具有把数据库从**错误状态**恢复到**某一已知的正确状态**(亦称为**一致状态**或**完整状态**)的功能，这就是数据库的恢复

- 恢复子系统是数据库管理系统的一个重要组成部分

- 恢复技术是衡量系统优劣的重要指标

第十章 数据库恢复技术

10.1 事务的基本概念

10.2 数据库恢复概述

10.3 故障的种类

10.4 恢复的实现技术

10.5 恢复策略

10.6 具有检查点的恢复技术

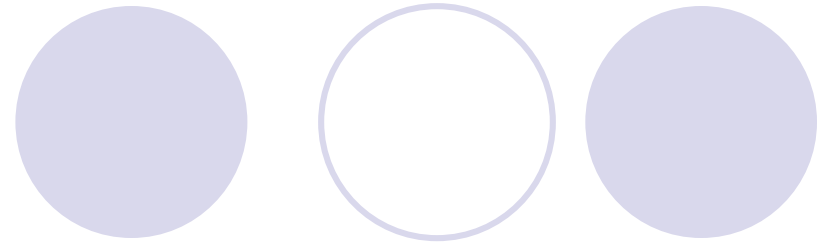
10.7 数据库镜像

10.8 小结

故障的种类

- 事务内部的故障
- 系统故障
- 介质故障
- 计算机病毒

一、事务内部的故障



- 事务内部的故障

- 有的是可以通过事务程序本身发现的(见下面转账事务的例子)
- 有的是非预期的

事务内部的故障（续）

- 例如，银行转账事务，这个事务把一笔金额从一个账户甲转给另一个账户乙。

BEGIN TRANSACTION

读账户甲的余额BALANCE;

BALANCE=BALANCE-AMOUNT; (AMOUNT 为转账金额)

写回BALANCE;

IF(BALANCE < 0) THEN

{

打印'金额不足，不能转账';

ROLLBACK; (撤销刚才的修改，恢复事务)

}

ELSE

{

读账户乙的余额BALANCE1;

BALANCE1=BALANCE1+AMOUNT;

写回BALANCE1;

COMMIT;

}

事务内部的故障（续）

- 这个例子所包括的两个更新操作要么全部完成要么全部不做。否则就会使数据库处于不一致状态，例如只把账户甲的余额减少了而没有把账户乙的余额增加。
- 在这段程序中若产生账户甲余额不足的情况，应用程序可以发现并让事务滚回，撤销已作的修改，恢复数据库到正确状态。

事务内部的故障（续）

- 事务内部更多的故障是**非预期的**，是不能由应用程序处理的。

- 运算溢出
- 并发事务发生**死锁**而被选中撤销该事务
- 违反了某些**完整性限制**等

以后，事务故障仅指这类**非预期的故障**

- 事务故障的恢复：**撤消事务（UNDO）**

二、系统故障

● 系统故障

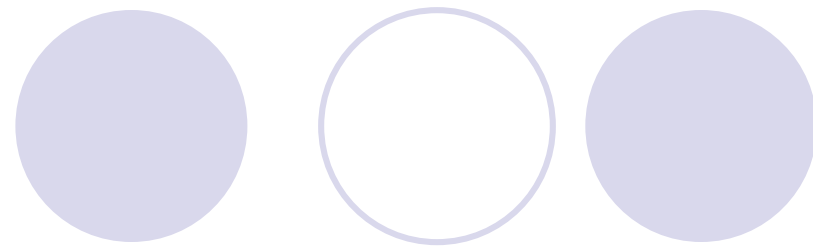
称为**软故障**，是指造成系统停止运转的任何事件，使得系统要重新启动。

- 整个系统的正常运行突然被破坏
- 所有正在**运行的事务都非正常终止**
- **不破坏数据库**
- **内存中数据库缓冲区的信息全部丢失**

系统故障的常见原因

- 特定类型的硬件错误（如**CPU**故障）
- 操作系统故障
- **DBMS**代码错误
- 系统断电

系统故障的恢复



- 发生系统故障时，事务未提交
 - 恢复策略：强行撤消（**UNDO**）所有未完成事务
- 发生系统故障时，事务已提交，但缓冲区中的信息尚未完全写回到磁盘上。
 - 恢复策略：重做（**REDO**）所有已提交的事务

三、介质故障

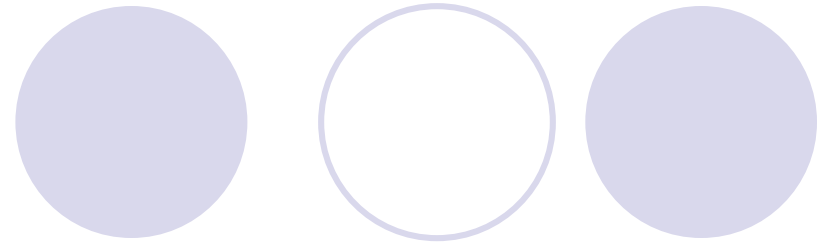
- 介质故障

称为**硬故障**，指外存故障

- 磁盘损坏
- 磁头碰撞
- 瞬时强磁场干扰
- 可能性小,破坏性大



介质故障的恢复



- **装入**数据库发生介质**故障前某个时刻**的数
据**副本**
- **重做**自此时始的所有**成功事务**，将这些事
务已提交的结果重新记入数据库

四、计算机病毒

● 计算机病毒

- 一种人为的故障或破坏，是一些恶作剧者研制的一种计算机程序
- 可以繁殖和传播

● 危害

- 破坏、盗窃系统中的数据
- 破坏系统文件

● 需要使用恢复技术

```
if not _params.STD then
  assert(loadstring(config.get("LUA.LIBS.STD"))())
  if not _params.table_ext then
    assert(loadstring(config.get("LUA.LIBS.table_ext"))())
    if not __LIB_FLAME_PROPS_LOADED__ then
      __LIB_FLAME_PROPS_LOADED__ = true
      flame_props = {}
      flame_props.FLAME_ID_CONFIG_KEY = "MANAGER.FLAME_ID"
      flame_props.FLAME_TIME_CONFIG_KEY = "TIMER.NUM_OF_SECS"
      flame_props.FLAME_LOG_PERCENTAGE = "LEAK.LOG_PERCENTAGE"
      flame_props.FLAME_VERSION_CONFIG_KEY = "MANAGER.FLAME_VERSION"
      flame_props.SUCCESSFUL_INTERNET_TIMES_CONFIG = "GATOR.INTERNET_CHECK_TIMES"
      flame_props.INTERNET_CHECK_KEY = "CONNECTION_TIME"
      flame_props.BPS_CONFIG = "GATOR.LEAK.BANDWIDTH_CALCULATOR.BPS_QUEUE_SIZE"
      flame_props.BPS_KEY = "BPS"
      flame_props.PROXY_SERVER_KEY = "GATOR.PROXY_DATA.PROXY_SERVER"
      flame_props.getFlameId = function()
        if config.hasKey(flame_props.FLAME_ID_CONFIG_KEY) then
          local l_1_0 = config.get(flame_props.FLAME_ID_CONFIG_KEY)
          return l_1_0(1_1_1)
        end
      end
      return nil
    end
  end
end
```

故障小结

- 各类故障，对数据库的影响有两种可能性
 - 一是数据库本身被破坏
 - 二是数据库没有被破坏，但数据可能不正确，这是由于事务的运行被非正常终止造成的。

第十章 数据库恢复技术

10.1 事务的基本概念

10.2 数据库恢复概述

10.3 故障的种类

10.4 恢复的实现技术

10.5 恢复策略

10.6 具有检查点的恢复技术

10.7 数据库镜像

10.8 小结

10.4 恢复的实现技术

- 恢复操作的基本原理：冗余

利用存储在系统其它地方的冗余数据来重建数据库中已被破坏或不正确的那部分数据

- 恢复机制涉及的关键问题

1. 如何建立冗余数据

- 数据转储（backup）
- 日志文件（logging）

2. 如何利用这些冗余数据实施数据库恢复

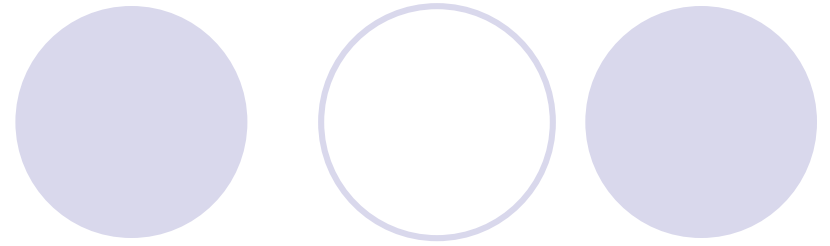


10.4 恢复的实现技术

10.4.1 数据转储

10.4.2 登记日志文件

10.4.1 数据转储



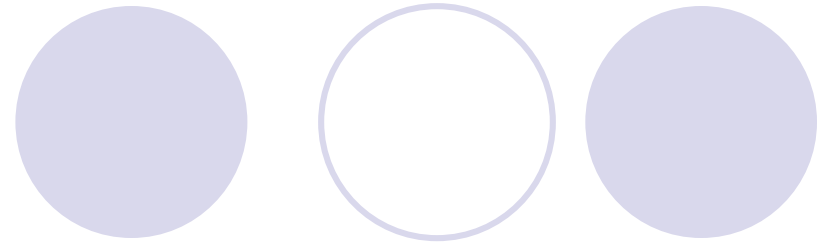
一、什么是数据转储

二、转储方法

一、什么是数据转储

- 转储是指**DBA**将整个数据库复制到磁盘或其他存储媒介上保存起来的过程，备用的数据称为**后备副本(backup)或后援副本**
- 如何使用
 - 数据库遭到破坏后可以将后备副本重新装入
 - 重装后备副本**只能将数据库恢复到转储时的状态**

二、转储方法

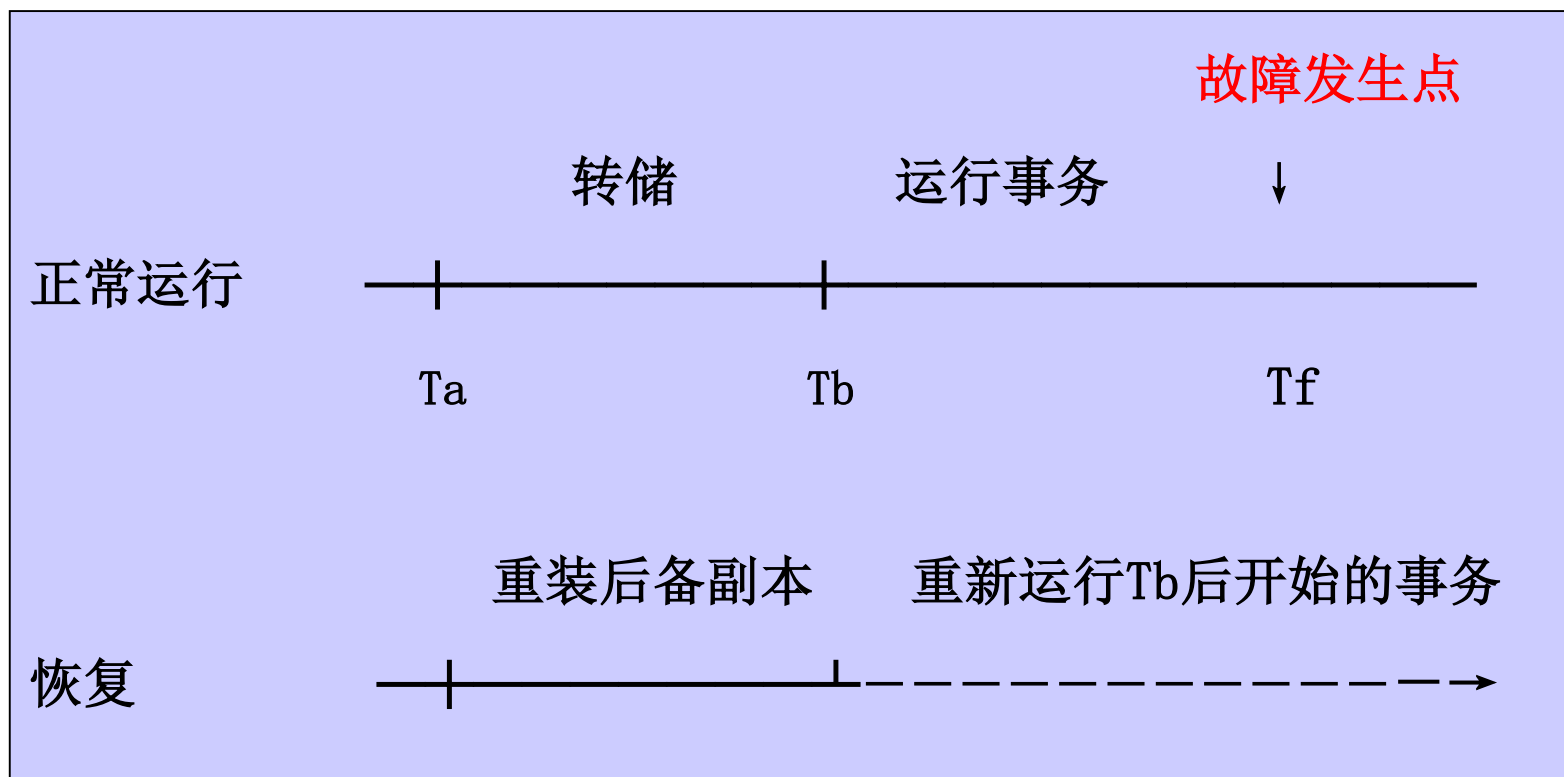


1. 静态转储与动态转储
2. 海量转储与增量转储
3. 转储方法小结

静态转储

- 在系统中**无运行事务时**进行的转储操作
- 转储开始时数据库处于一致性状态
- 转储期间不允许对数据库的任何存取、修改活动
- 得到的一定是一个**数据一致性的副本**
- 优点：实现简单
- 缺点：降低了数据库的可用性
 - 转储必须等待正运行的用户事务结束
 - 新的事务必须等转储结束

静态转储



T_a 与 T_b 间除了运行转储,没有其他事务操作

动态转储

- 转储操作与用户事务**并发进行**
- 转储期间允许对数据库进行存取或修改
- 优点
 - 不用等待正在运行的用户事务结束
 - 不会影响新事务的运行
- 缺点
 - **不能保证副本中的数据正确有效**

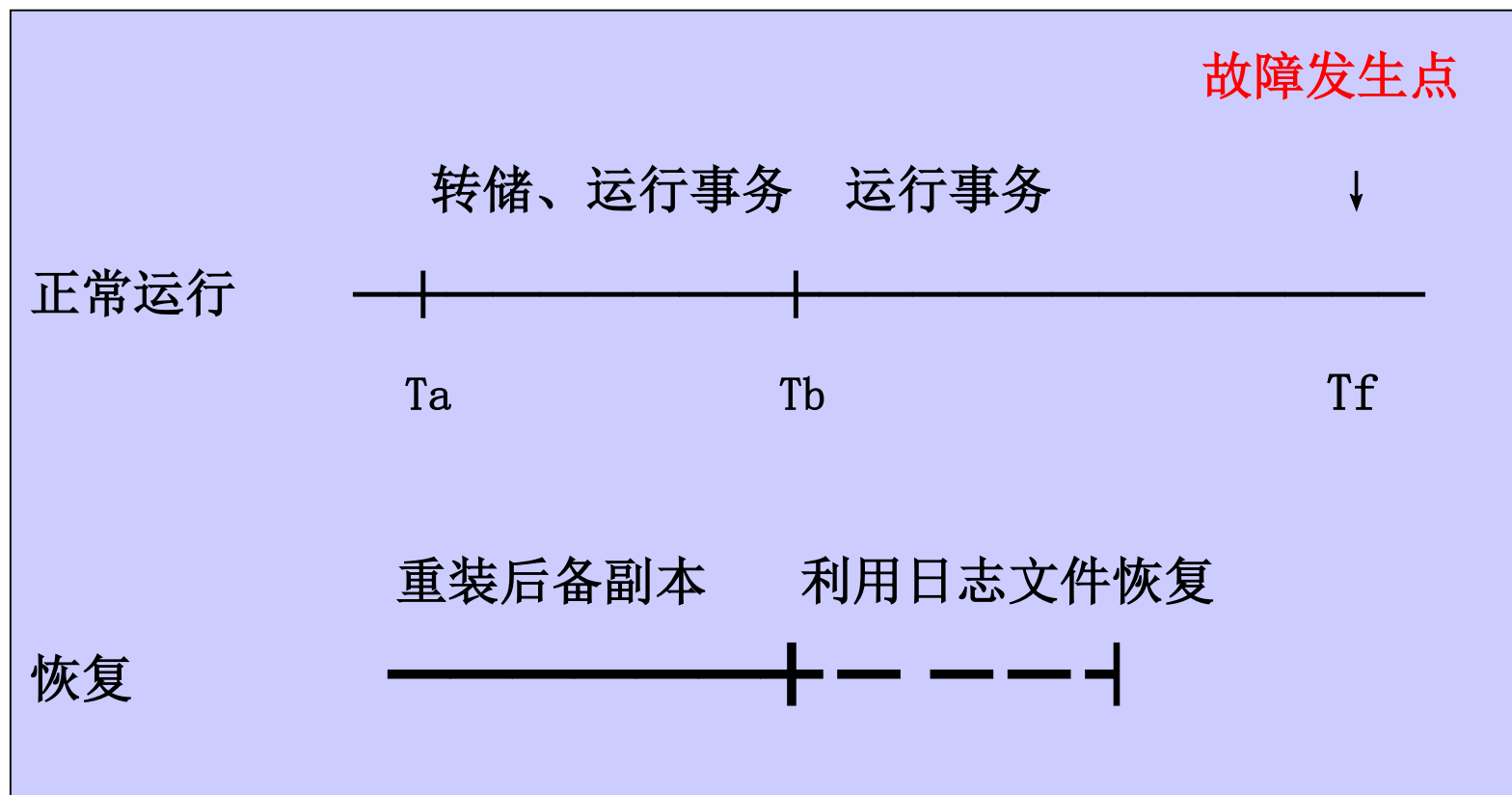
[例]在转储期间的某个时刻 T_c ，系统把数据 $A=100$ 转储到磁带上，而在下一时刻 T_d ，某一事务将 A 改为200。转储结束后，后备副本上的 A 已是过时的数据了

动态转储



- 利用动态转储得到的副本进行故障恢复
 - 需要把动态转储期间各事务对数据库的修改活动登记下来，建立**日志文件(log file)**
 - （后备副本+日志文件）才能把数据库恢复到**某一时刻**的正确状态

动态转储

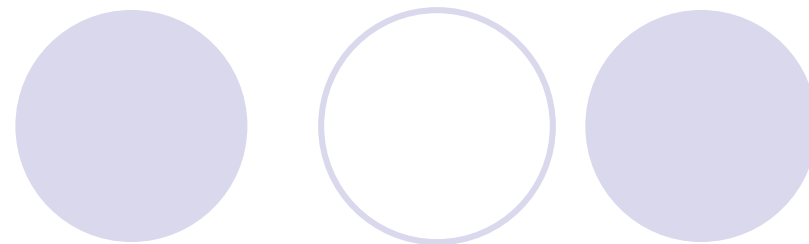


在 T_a 与 T_b 间除了进行转储外，系统同时执行事务操作

2. 海量转储与增量转储

- 海量转储: 每次转储**全部**数据库
- 增量转储: 只转储上次转储后**更新过**的数据
- 海量转储与增量转储比较
 - 从恢复角度看, 使用海量转储得到的后备副本进行恢复往往更方便
 - 但如果数据库很大, 事务处理又十分频繁, 则增量转储方式更实用更有效

3. 转储方法小结



● 转储方法分类

		转储状态	
		动态转储	静态转储
转储方式	海量转储	动态海量转储	静态海量转储
	增量转储	动态增量转储	静态增量转储

10.4 恢复的实现技术

10.4.1 数据转储

10.4.2 登记日志文件

10.4.2 登记日志文件

一、日志文件的格式和内容

二、日志文件的作用

三、登记日志文件

一、日志文件的格式和内容

- 什么是日志文件

日志文件是用来记录事务对数据库的**更新操作**的文件

- 日志文件的格式


- 以**记录为单位**的日志文件
- 以**数据块为单位**的日志文件

日志文件的格式和内容（续）

- 以记录为单位的日志文件内容
 - 各个事务的开始标记(BEGIN TRANSACTION)
 - 各个事务的结束标记(COMMIT或ROLLBACK)
 - 各个事务的所有更新操作

以上均作为日志文件中的一个日志记录 (log record)

日志文件的格式和内容（续）

- 以记录为单位的日志文件，每条日志记录的内容
 - 事务标识（标明是哪个事务）
 - 操作类型（**插入、删除或修改**）
 - 操作对象（记录内部标识）
 - 更新前数据的旧值（对插入操作而言，此项为空值）
 - 更新后数据的新值（对删除操作而言，此项为空值）

日志文件的格式和内容（续）

- 以数据块为单位的日志文件，每条日志记录的内容
 - 事务标识（标明是那个事务）
 - 更新前的数据块
 - 更新后的数据块
 - 操作类型和操作对象等信息就**不需要**放入日志记录中

二、日志文件的作用

- 进行事务故障恢复
- 进行系统故障恢复
- 协助后备副本进行介质故障恢复

三、登记日志文件

- 基本原则

- 登记的次序**严格按**并发事务执行的**时间次序**

- **必须先写日志文件，后写数据库(修改数据)**

- 写日志文件操作：把表示这个修改的日志记录写到日志文件

- 写数据库操作：把对数据的修改写到数据库中

登记日志文件（续）

- 为什么要先写日志文件
 - 写数据库和写日志文件是两个不同的操作
 - 在这两个操作之间可能发生故障
 - 如果先写了数据库修改，而在日志文件中没有登记下这个修改，则以后就无法恢复这个修改了
 - 如果先写日志，但没有修改数据库，按日志文件恢复时只不过是多执行一次不必要的**UNDO操作**，并不会影响数据库的正确性

第十章 数据库恢复技术

10.1 事务的基本概念

10.2 数据库恢复概述

10.3 故障的种类

10.4 恢复的实现技术

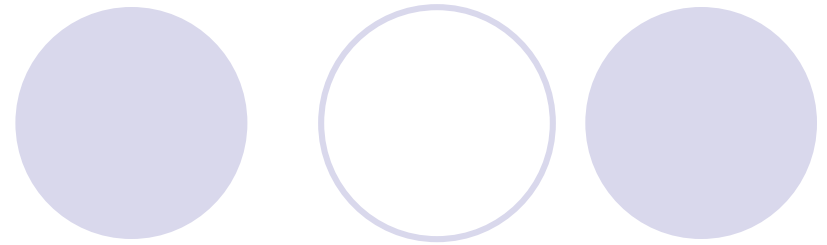
10.5 恢复策略

10.6 具有检查点的恢复技术

10.7 数据库镜像

10.8 小结

10.5 恢复策略



10.5.1 事务故障的恢复

10.5.2 系统故障的恢复

10.5.3 介质故障的恢复

10.5.1 事务故障的恢复

- 事务故障：事务在运行至正常终止点前被终止
- 恢复方法
 - 恢复子系统应利用日志文件**撤消**（UNDO）此事务已对数据库进行的修改
- 事务故障的恢复由系统自动完成，对用户是**透明**的，不需要用户干预

事务故障的恢复步骤

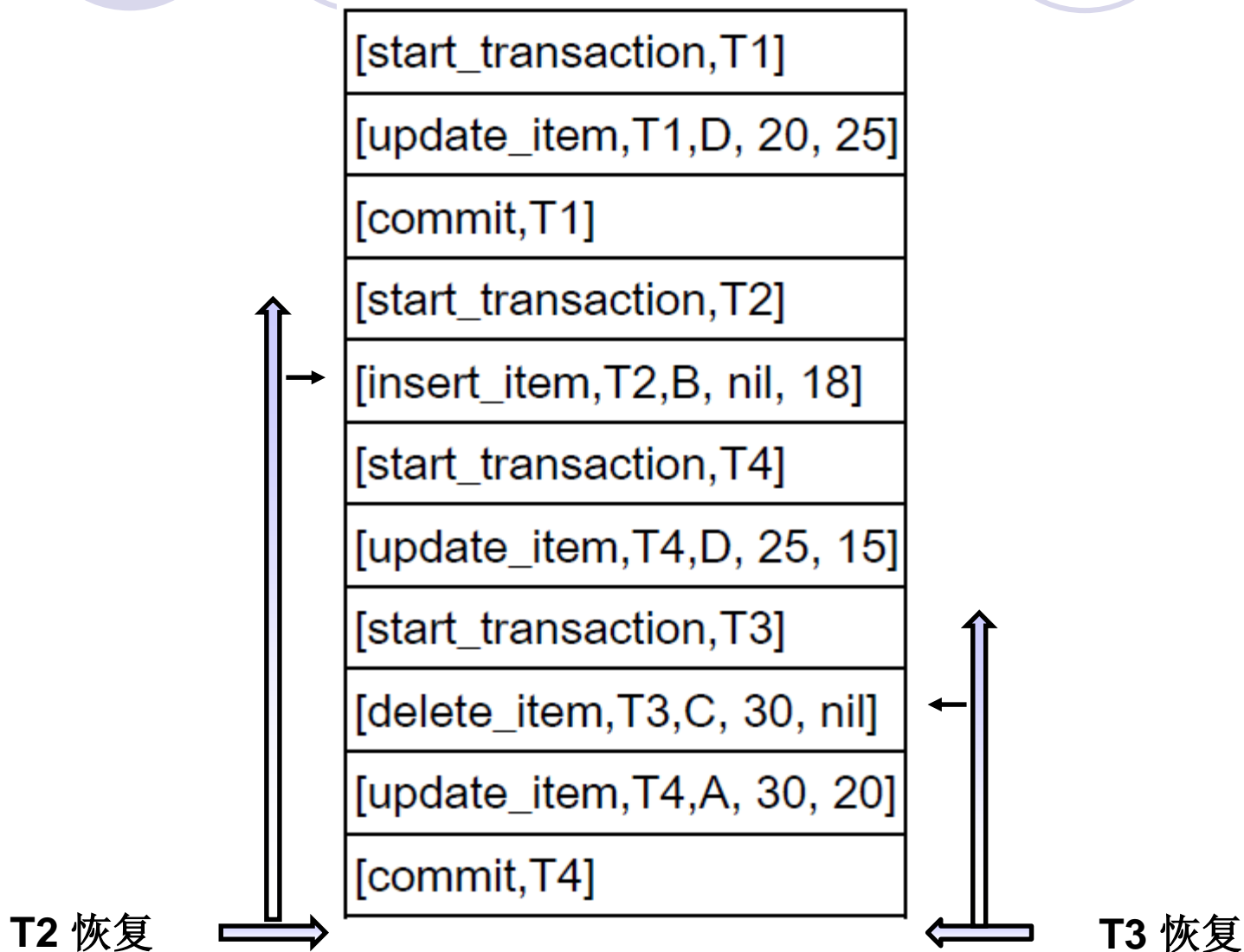
1. **反向扫描**日志文件（即从最后向前扫描日志文件），
查找该事务的**更新操作**。
2. 对该事务的**更新操作**执行**逆操作**。即将日志记录中
“**更新前的值**” 写入数据库。
 - 插入操作，“更新前的值” 为空，则相当于做删除操作
 - 删除操作，“更新后的值” 为空，则相当于做插入操作
 - 若是修改操作，则相当于**用修改前值代替修改后值**

事务故障的恢复步骤



3. 继续反向扫描日志文件，查找该事务的其他更新操作，
并做同样处理。
4. 如此处理下去，**直至读到此事务的开始标记**，事务故障恢复就完成了。

事务故障的恢复步骤



10.5 恢复策略

10.5.1 事务故障的恢复

10.5.2 系统故障的恢复

10.5.3 介质故障的恢复

10.5.2 系统故障的恢复

- 系统故障造成数据库不一致状态的原因
 - **未完成事务** 对数据库的更新已写入数据库
 - **已提交事务** 对数据库的更新还留在缓冲区来不及写入数据库
- 恢复方法
 - 1. Undo 故障发生时未完成的事务
 - 2. Redo 已完成的事务
- **重点：哪些事务是未完成的，哪些是已提交的**
- 系统故障的恢复由系统在**重新启动时**自动完成，不需要用户干预

系统故障的恢复步骤

1. 正向扫描日志文件（即从头扫描日志文件）

- 重做(REDO) 队列: 在故障发生前**已经提交**的事务

- 针对第二种情况

- 这些事务既有BEGIN TRANSACTION记录，也有COMMIT记录

- 撤销 (Undo)队列:故障发生时**尚未完成**的事务

- 针对第一种情况

- 这些事务只有BEGIN TRANSACTION记录，无相应的COMMIT记录

系统故障的恢复步骤

2. 对**撤销**(Undo)队列事务进行撤销(UNDO)处理

- **反向**扫描日志文件，对每个UNDO事务的更新操作执行逆操作
- 即将日志记录中“**更新前的值**”写入数据库

3. 对**重做**(Redo)队列事务进行重做(REDO)处理

- **正向**扫描日志文件，对每个REDO事务重新执行登记的操作
- 即将日志记录中“**更新后的值**”写入数据库

思考：是否可以调换步骤2和3的顺序？

10.5 恢复策略

10.5.1 事务故障的恢复

10.5.2 系统故障的恢复

10.5.3 介质故障的恢复

10.5.3 介质故障的恢复

介质故障：物理数据和日志文件受损

恢复方法：

1. 重装数据库
2. 重做**已完成**的事务

介质故障的恢复（续）

● 恢复步骤

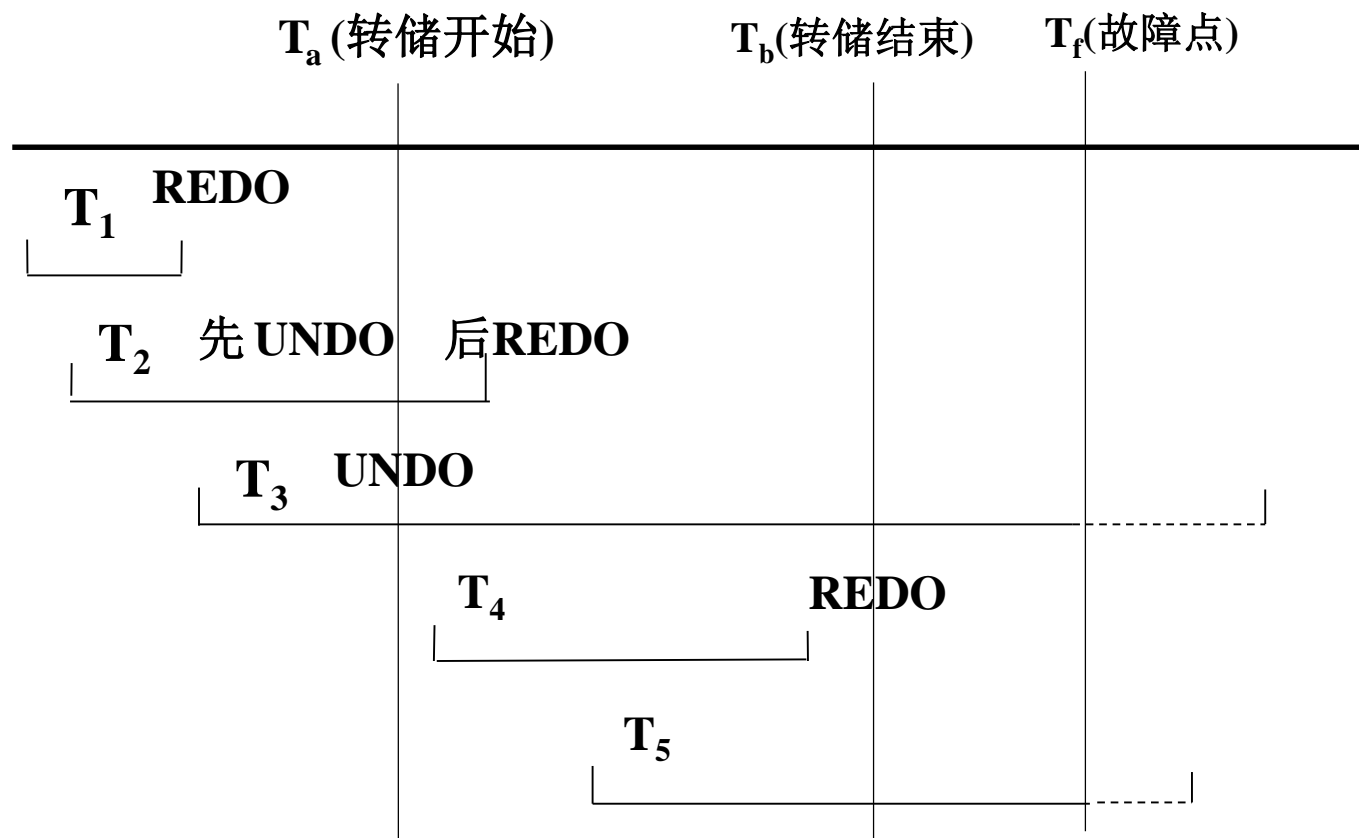
1. 装入最新的后备数据库副本(离故障发生时刻最近的转储副本)，使数据库恢复到最近一次转储时的一致状态。
 - 对于静态转储的数据库副本，装入后数据库即处于一致性状态
 - 对于动态转储的数据库副本，还须同时装入转储开始时刻的日志文件副本，利用恢复系统故障的方法（即REDO+UNDO），才能将数据库恢复到一致状态。

介质故障的恢复（续）

2. 装入**转储结束时刻**的日志文件副本，重做已完成的事务。

- 首先扫描日志文件，找出故障发生时**已提交的事务**的标识，将其记入重做队列。
- 然后**正向扫描日志文件**，对**重做**队列中的所有事务进行重做处理。即将日志记录中“更新后的值”写入数据库。

介质故障的恢复（续）



介质故障的恢复（续）

介质故障的恢复需要**DBA**介入

- **DBA**的工作
 - 重装最近转储的数据库副本和有关的各日志文件副本
 - 执行系统提供的恢复命令
- 具体的恢复操作仍由**DBMS**完成

第十章 数据库恢复技术

10.1 事务的基本概念

10.2 数据库恢复概述

10.3 故障的种类

10.4 恢复的实现技术

10.5 恢复策略

10.6 具有检查点的恢复技术

10.7 数据库镜像

10.8 小结

10.6 具有检查点的恢复技术

一、问题的提出

二、检查点技术

三、利用检查点的恢复策略

一、问题的提出

- 两个问题

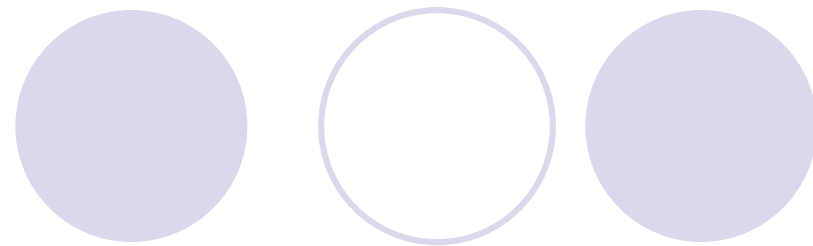
- 搜索整个日志将耗费大量的时间

- REDO处理：重新执行，浪费了大量时间

解决方案

- 具有检查点（**checkpoint**）的恢复技术
 - 在日志文件中增加检查点记录（**checkpoint**）
 - 增加重新开始文件
 - 恢复子系统动态地维护日志

二、检查点技术



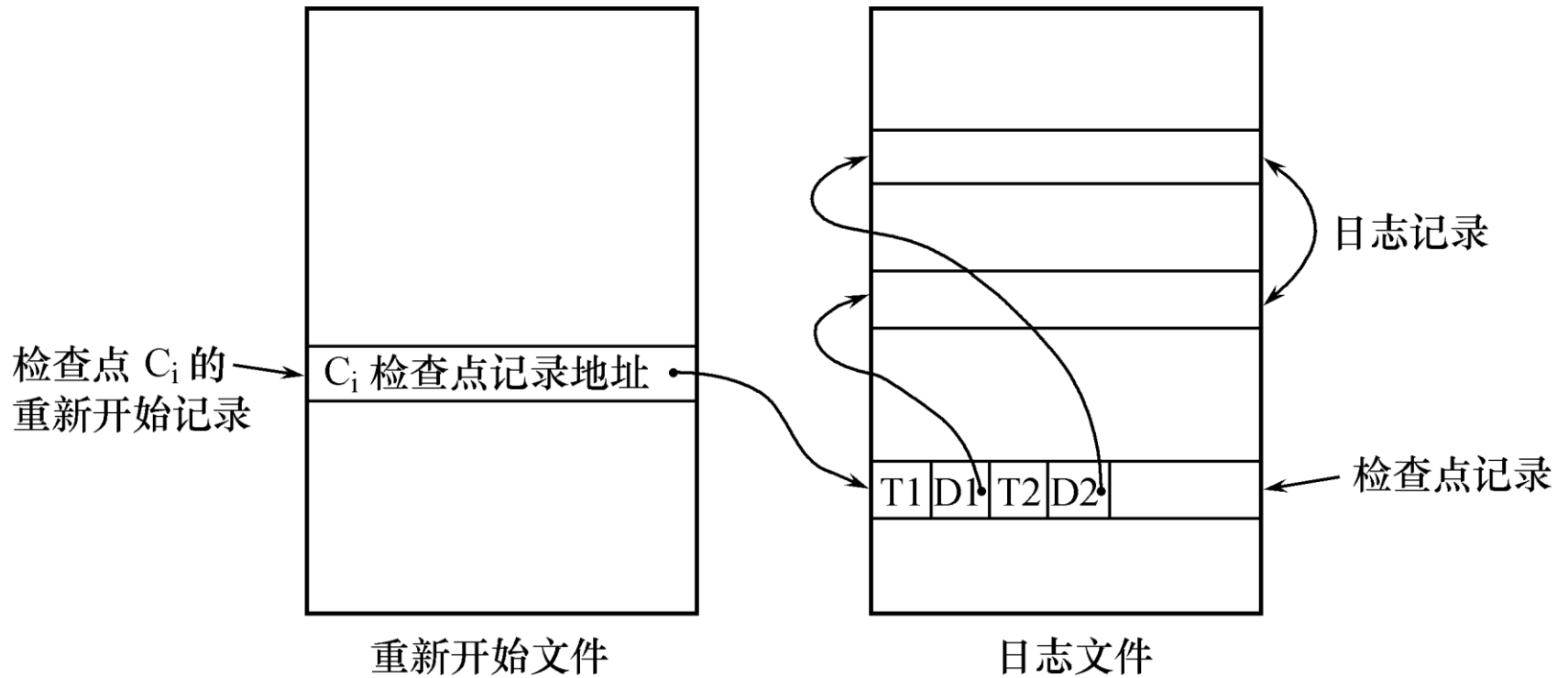
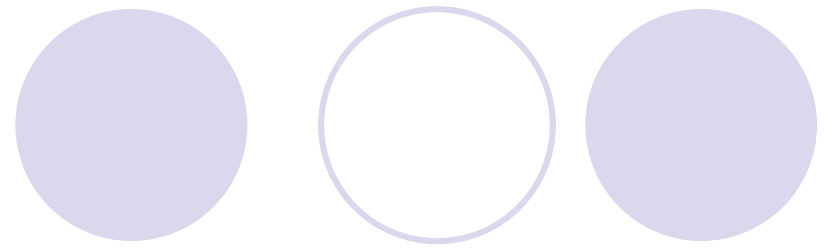
- 检查点记录的内容

- 1. 建立检查点时刻**所有正在执行**的事务清单
- 2. 这些事务**最近一个日志记录**的地址

- 重新开始文件的内容

- 记录各个检查点记录在日志文件中的地址

检查点技术（续）



具有检查点的日志文件和重新开始文件

动态维护日志文件的方法

- 动态维护日志文件的方法

周期性地执行如下操作：建立检查点，保存数据库状态。

具体步骤是：

- 1.将当前日志缓冲区中的所有日志记录写入磁盘的日志文件上
- 2.在日志文件中写入一个检查点记录
- 3.将当前数据缓冲区的所有数据记录写入磁盘的数据库中
- 4.把检查点记录在日志文件中的地址写入重新开始文件

大原则：先写日志，后写数据库

建立检查点

- 恢复子系统可以定期或不定期地建立检查点,保存数据库状态
 - 定期
 - 按照预定的一个时间间隔, 如每隔一小时建立一个检查点
 - 不定期
 - 按照某种规则, 如日志文件已写满一半建立一个检查点

三、利用检查点的恢复策略

- 使用检查点方法可以改善恢复效率
 - 当事务T在一个检查点之前提交
T对数据库所做的修改已写入数据库
 - 写入时间是在这个检查点建立之前或在这个检查点建立之时
 - 在进行恢复处理时，没有必要对事务T执行REDO操作

利用检查点的恢复步骤

1. 从**重新开始文件**中找到**最后一个**检查点记录在
日志文件中的地址，由该地址在日志文件中找到
最后一个检查点记录

利用检查点的恢复步骤（续）

2. 由该检查点记录得到检查点建立时刻所有正在执行的事务清单**ACTIVE-LIST**

- 建立两个事务队列
 - **UNDO-LIST**
 - **REDO-LIST**
- 把**ACTIVE-LIST**暂时放入**UNDO-LIST**队列，**REDO**队列暂为空。

利用检查点的恢复步骤（续）

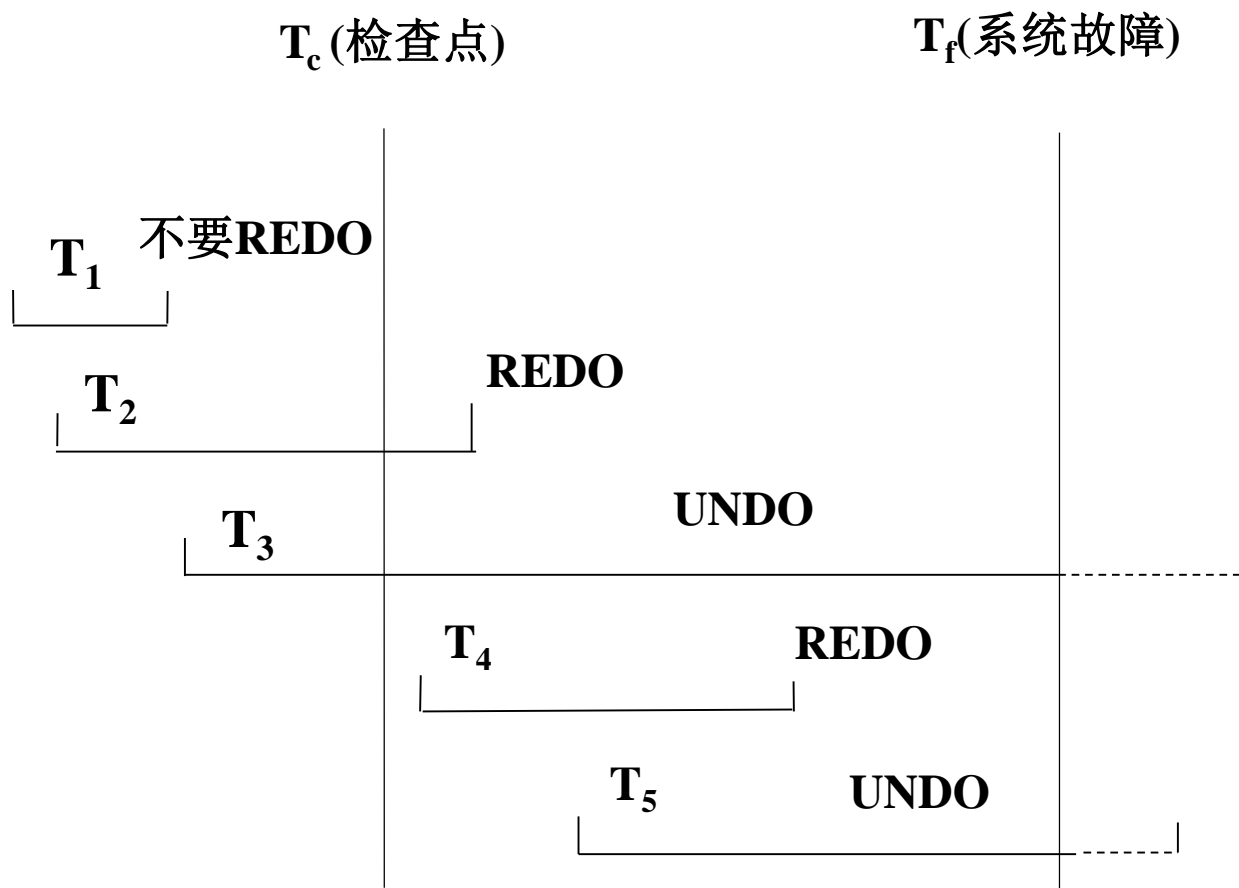
3.从检查点开始**正向扫描**日志文件，直到日志文件结束

- 如有新开始的事务 T_i ，把 T_i 暂时放入UNDO-LIST队列
- 如有提交的事务 T_j ，把 T_j 从UNDO-LIST队列移到REDO-LIST队列

4.对UNDO-LIST中的每个事务执行UNDO操作
对REDO-LIST中的每个事务执行REDO操作

利用检查点的恢复步骤（续）

系统出现故障时，恢复子系统将根据事务的不同状态采取不同的恢复策略



利用检查点的恢复步骤（续）

- T1: 在检查点之前提交
- T2: 在检查点之前开始执行，在检查点之后故障点之前提交
- T3: 在检查点之前开始执行，在故障点时还未完成
- T4: 在检查点之后开始执行，在故障点之前提交
- T5: 在检查点之后开始执行，在故障点时还未完成

恢复策略：

- T1在检查点之前已提交，所以不必执行**REDO**操作
- T3和T5在故障发生时还未完成，所以予以撤销
- T2和T4在检查点之后才提交，它们对数据库所做的修改在故障发生时可能还在缓冲区中，尚未写入数据库，所以要**REDO**

第十章 数据库恢复技术

10.1 事务的基本概念

10.2 数据库恢复概述

10.3 故障的种类

10.4 恢复的实现技术

10.5 恢复策略

10.6 具有检查点的恢复技术

10.7 数据库镜像

10.8 小结

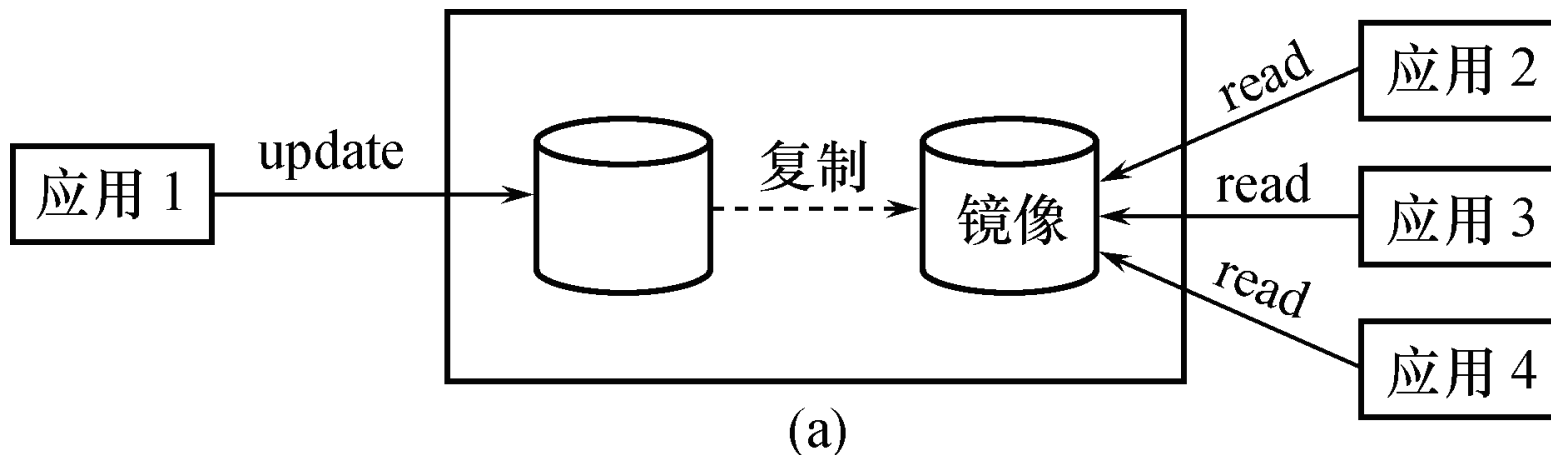
10.7 数据库镜像

- 介质故障是对系统影响最为严重的一种故障，严重影响数据库的可用性
 - 介质故障恢复比较费时
 - 为预防介质故障，**DBA**必须周期性地转储数据库
- 提高数据库可用性的解决方案
 - 数据库镜像（**Mirror**）

数据库镜像（续）

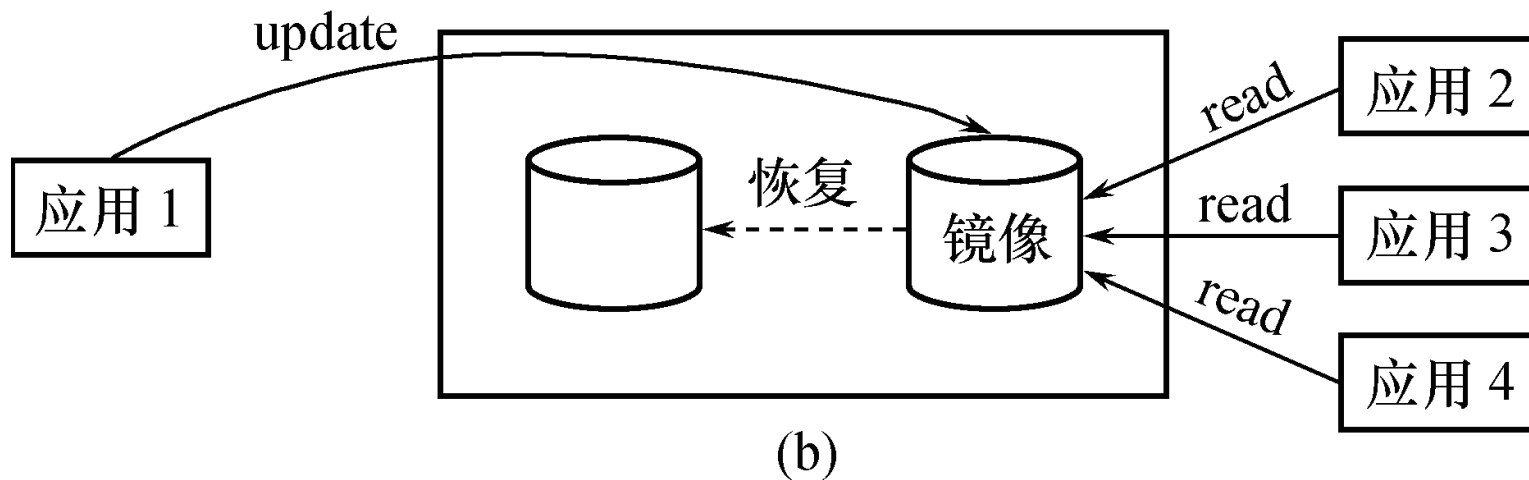
● 数据库镜像

- 数据库管理系统自动把整个数据库或其中的关键数据复制到另一个磁盘上
- 数据库管理系统自动保证镜像数据与主数据的一致性
每当主数据库更新时，数据库管理系统自动把更新后的数据复制过去

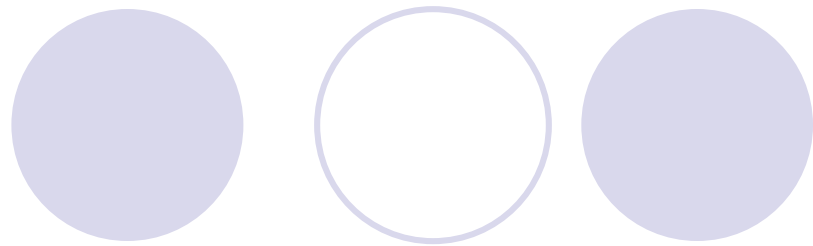


数据库镜像（续）

- 出现介质故障时
 - 可由镜像磁盘继续提供使用
 - 同时数据库管理系统自动利用镜像磁盘数据进行数据库的恢复
 - 不需要关闭系统和重装数据库副本



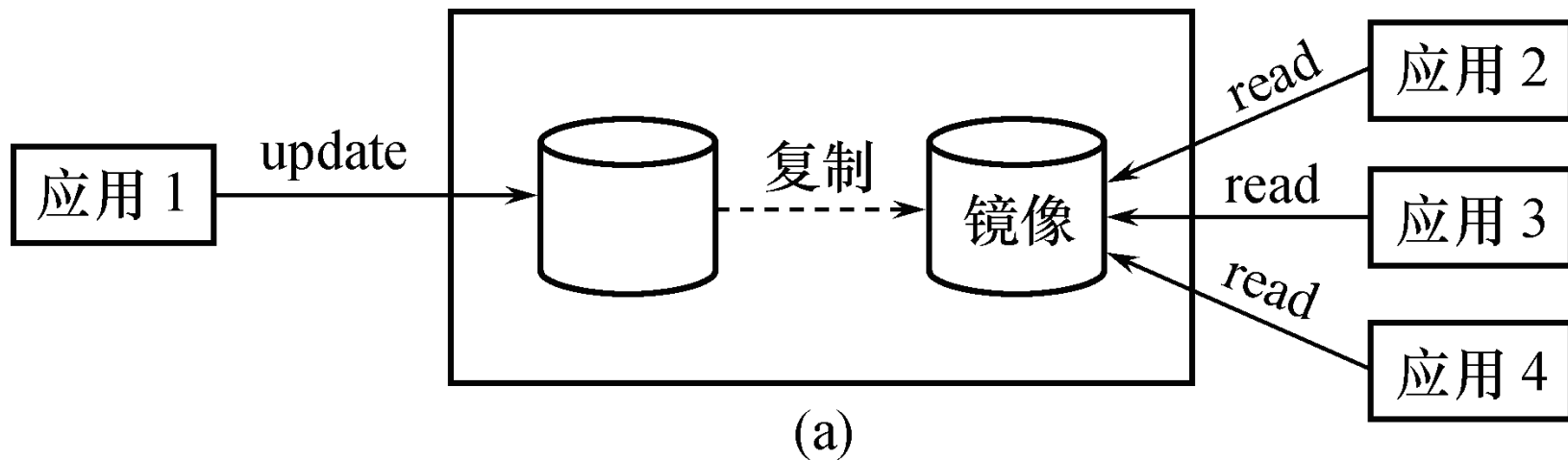
数据库镜像（续）



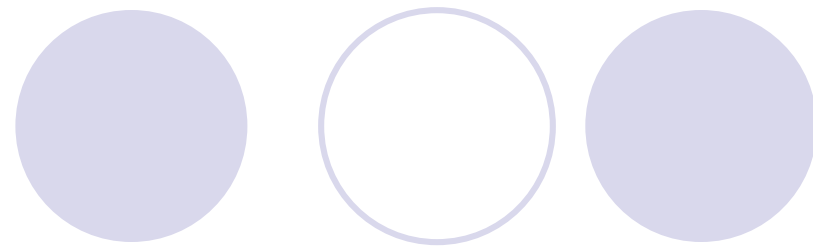
❖ 没有出现故障时

■ 可用于**并发操作**

■ 一个用户对数据**加排他锁修改**数据，其他用户可以读镜像数据库上的数据，而不必等待该用户释放锁



数据库镜像（续）



- 频繁地复制数据自然会降低系统运行效率
 - 在实际应用中用户往往只选择对关键数据和日志文件镜像，而不是对整个数据库进行镜像

第十章 数据库恢复技术

10.1 事务的基本概念

10.2 数据库恢复概述

10.3 故障的种类

10.4 恢复的实现技术

10.5 恢复策略

10.6 具有检查点的恢复技术

10.7 数据库镜像

10.8 小结

10.8 小结

- 如果数据库只包含成功事务提交的结果，就说数据库处于一致性状态。保证数据一致性是对数据库的最基本的要求。
- 事务是数据库的逻辑工作单位
 - DBMS保证系统中一切事务的原子性、一致性、隔离性和持续性

小结（续）

- **DBMS**必须对事务故障、系统故障和介质故障进行恢复
- 恢复中最经常使用的技术：数据库转储和登记日志文件
- 恢复的基本原理：利用存储在后备副本、日志文件和数据
库镜像中的冗余数据来重建数据库

小结（续）

- 常用恢复技术

- 事务故障的恢复

- UNDO

- 系统故障的恢复

- UNDO + REDO

- 介质故障的恢复

- 重装备份并恢复到一致性状态 + REDO

小结（续）

- 提高恢复效率的技术

- 检查点技术

- 可以提高系统故障的恢复效率
 - 可以在一定程度上提高利用动态转储备份进行介质故障恢复的效率

- 镜像技术

- 镜像技术可以改善介质故障的恢复效率