第九章 数据库恢复技术（10）

第九章 数据库恢复技术

9.1 事务的基本概念

9.2 数据库恢复概述

9.3 故障的种类

9.4 恢复的实现技术

9.5 恢复策略

9.6 具有检查点的恢复技术

9.7 数据库镜像

第九章 数据库恢复技术

9.1 事务的基本概念

- **事务(Transaction)**
  - 是用户定义的一个*数据库操作序列*，这些操作要么都做，要么都不做，是一个不可分割的工作单位
  - 恢复和并发控制的基本单位

9.1 事务的基本概念

- **示例**

  银行转帐：事务T从A帐户过户50￥到B帐户

  T： read**(A);**
  A := A – 50;
  write**(A);**
  read**(B);**
  B := B + 50;
  write**(B);**

read**(X)：从数据库传送数据项X到事务的工作区中**
write**(X)：从事务的工作区中将数据项X写回数据库**

9.1 事务的基本概念

- 事务和程序比较
  - 在关系数据库中，一个事务可以是一条或多条SQL语句,或整个程序。
  - 一个程序通常包含多个事务

9.1 事务的基本概念

- **显式定义事务**
  - *BEGIN TRANSACTION*
  - *COMMIT*
  - *ROLLBACK*

  事务以Begin transaction开始，以Commit work或 Rollback work结束

  Commit work表示提交，事务正常结束

  Rollback work表示事务非正常结束，撤消事务已做的操作，回滚到事务开始时的状态

9.1 事务的基本概念

- **显式定义事务**

```
BEGIN TRANSACTION
    SQL 语句1
    SQL 语句2
    ……
COMMIT
```

```
BEGIN TRANSACTION
    SQL 语句1
    SQL 语句2
    ……
ROLLBACK
```

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；}
```

9.1 事务的基本概念

- 隐式方式

  当用户没有显式地定义事务时，DBMS按缺省规定自动划分事务

9.1 事务的基本概念

- AutoCommit事务是SQL Server默认事务方式。它指出每条SQL语句都构成事务，隐含事务的开始和结束控制点。
- 例：

update accounts set bal=bal-100 where accountno='A'

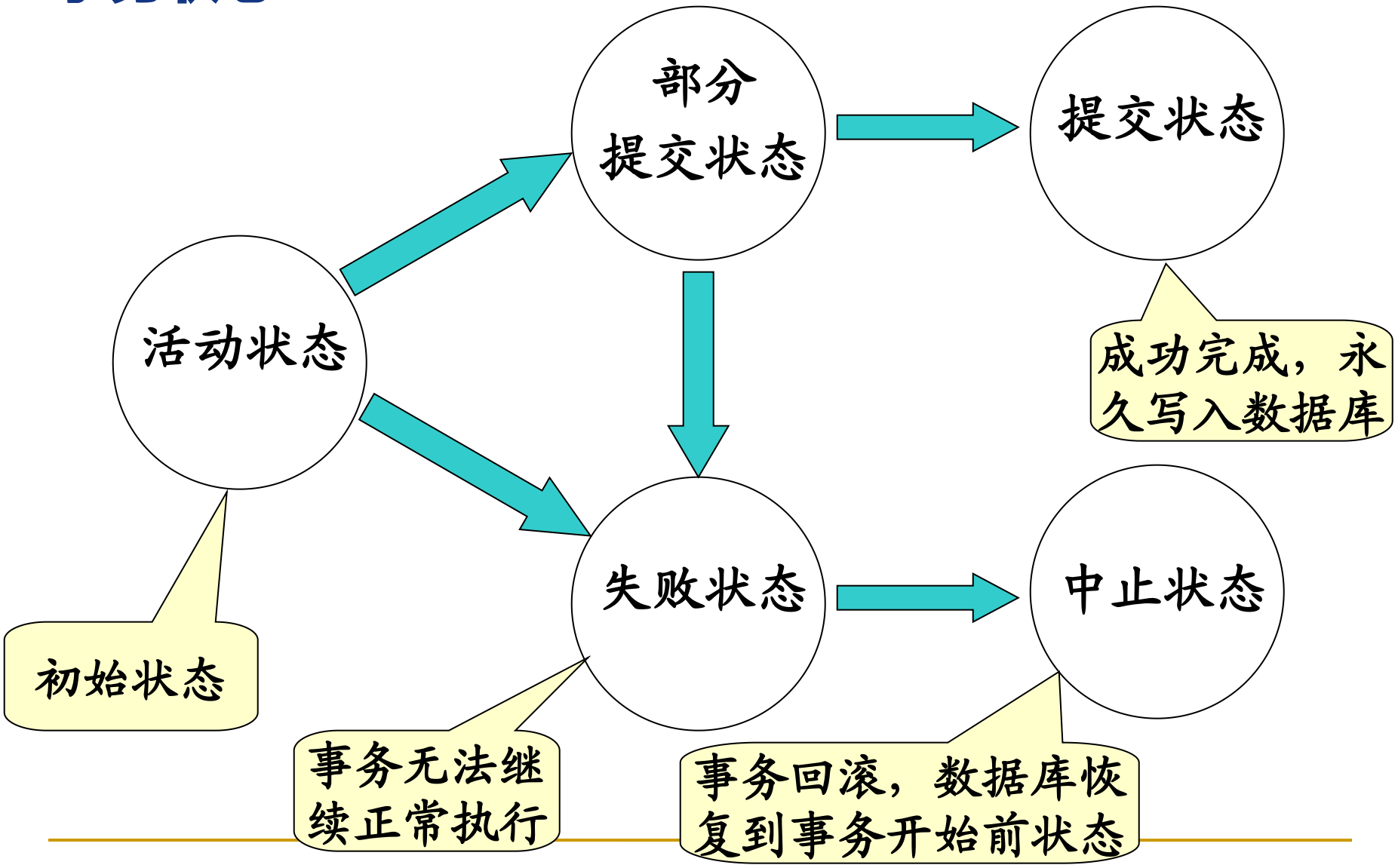
update accounts set bal=bal+100 where accountno='B'

9.1 事务的基本概念

- 例：采用显示事物的处理方式

```
begin tran
update accounts set bal=bal-100 where accountno='A'
 update accounts set bal=bal+100 where accountno='B'
end tran
```

事务状态


部分
提交状态
提交状态
活动状态
成功完成，永
久写入数据库
失败状态
中止状态
初始状态
事务无法继
续正常执行
事务回滚，数据库恢
复到事务开始前状态

9.1 事务的基本概念

- **事务的ACID特性**
  - *原子性（Atomicity）*
  - *一致性（Consistency）*
  - *隔离性（Isolation）*
  - *持续性（Durability）*

9.1 事务的基本概念

- **事务的ACID特性**
  - **原子性（Atomicity）**
    - 事务中包含的所有操作要么全做，要么全不做
    - 原子性由 *恢复机制* 实现

9.1 事务的基本概念

- 事务的ACID特性
  - 一致性（Consistency）
    - 事务开始前，数据库处于一致性的状态；事务结束后，数据库必须仍处于一致性状态

9.1 事务的基本概念

- 事务的ACID特性
  - 隔离性（Isolation）
    - 系统必须保证事务不受其它并发执行事务的影响
    - 对任何一对事务T_1，T_2，在T_1看来，T_2要么在T_1开始之前已经结束，要么在T_1完成之后再开始执行
    - 隔离性通过*并发控制机制*实现

9.1 事务的基本概念

- 事务的ACID特性
  - 持续性（Durability）
    - 一个事务一旦提交之后，它对数据库的影响必须是永久的
    - 系统发生故障不能改变事务的持久性
    - 持久性通过恢复机制实现

9.1 事务的基本概念

- 事务ACID特性可能遭到破坏的因素：
  - 多个事务并行运行时，不同事务交叉执行
  - 事务在运行过程中被强行停止

第九章 数据库恢复技术

9.1 事务的基本概念

9.2 数据库恢复概述

9.3 故障的种类

9.4 恢复的实现技术

9.5 恢复策略

9.6 具有检查点的恢复技术

9.7 数据库镜像

9.2 数据库恢复概述

- 故障是不可避免的
  - 系统故障：计算机软、硬件故障
  - 人为故障：操作员的失误、恶意的破坏等。
- 数据库的恢复

  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)

第九章 数据库恢复技术

9.1 事务的基本概念

9.2 数据库恢复概述

9.3 故障的种类

9.4 恢复的实现技术

9.5 恢复策略

9.6 具有检查点的恢复技术

9.7 数据库镜像

9.3 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

9.3 故障的种类-事务内部的故障

- 事务故障
  - 逻辑错误，事务由于某些内部条件而无法继续正常执行。
  - 系统错误，系统进入一种不良状态（如，死锁），使事务无法继续正常执行。
- 有的是可以通过事务程序本身发现的,有的是非预期的

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；}
```

9.3 故障的种类-事务内部的故障

- 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）*

日志文件

数据库

<T,start>

<T,A,1000,950>

A=950

<T,B,2000,2050>

B=2050

<T,commits>

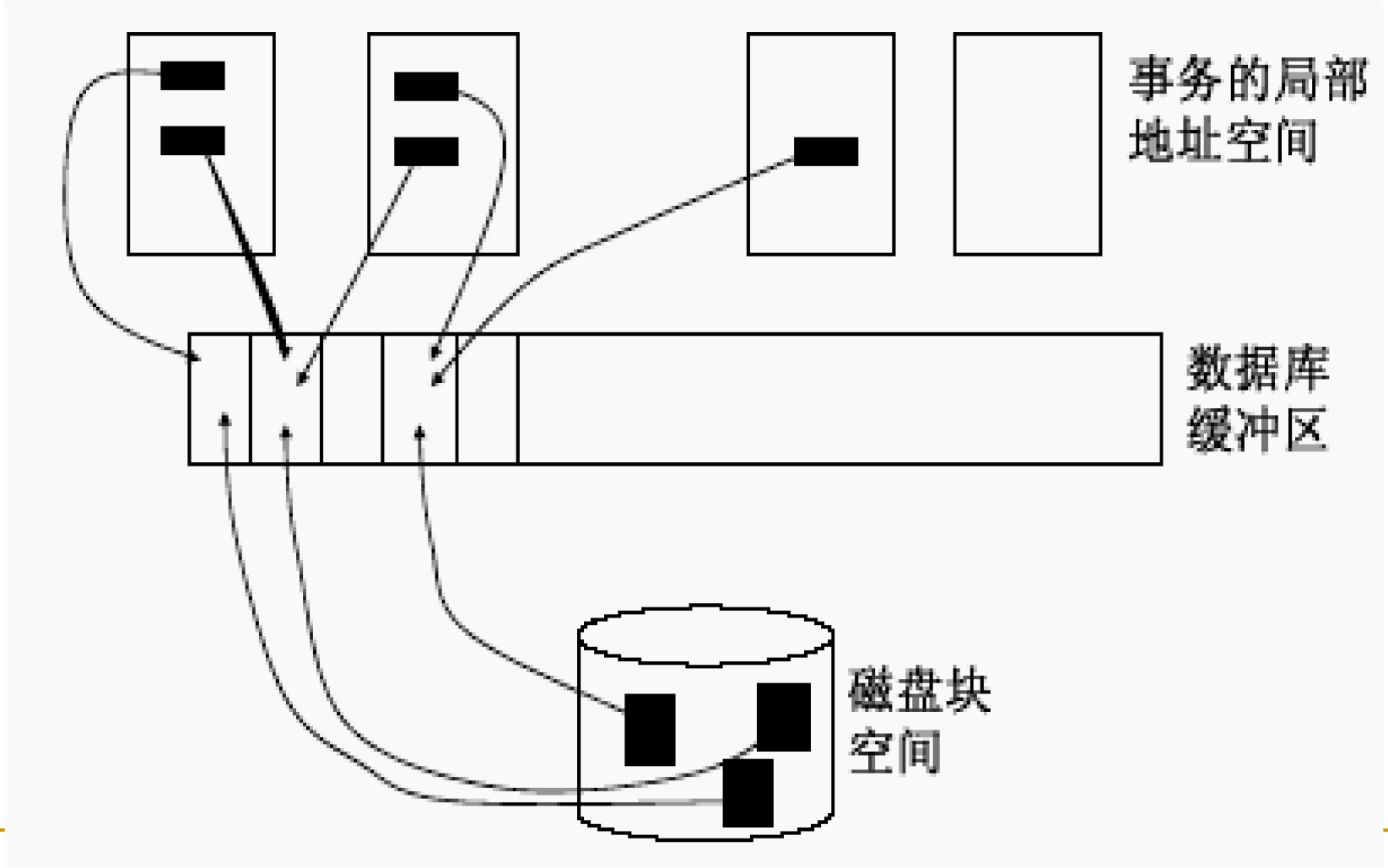
9.3 故障的种类-系统故障

- 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
  - 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

9.3 故障的种类-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
  - 内存中数据库缓冲区的信息全部丢失
  - *不破坏数据库*

事务执行相关的地址空间



日志文件

<T,start>

<T,A,1000,950>

<T,B,2000,250>

<T,commits>

数据库

A=950

B=2050

9.3 故障的种类-系统故障

- 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

9.3 故障的种类-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
  - 操作系统的某种潜在错误
  - 瞬时强磁场干扰
- 破坏数据库或部分数据库，并影响正在存取这部分数据的所有事务。

9.3 故障的种类-介质故障

- 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

9.3 故障的种类-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

9.3 故障的种类

- 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。

第九章 数据库恢复技术

9.1 事务的基本概念

9.2 数据库恢复概述

9.3 故障的种类

9.4 恢复的实现技术

9.5 恢复策略

9.6 具有检查点的恢复技术

9.7 数据库镜像

9.4 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

9.4 恢复的实现技术

- 恢复机制涉及的关键问题
  - 如何建立冗余数据
    - 数据转储（backup）
    - 登录日志文件（logging）
  - 如何利用这些冗余数据实施数据库恢复

9.4 恢复的实现技术-数据转储

- 转储是指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。备用的数据称为后备副本或后援副本
- 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

9.4 恢复的实现技术-数据转储

- 转储种类
  - 静态转储与动态转储
  - 海量转储与增量转储

9.4 恢复的实现技术-数据转储

- 什么是静态转储
  - 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

9.4 恢复的实现技术-数据转储

- 静态转储
  - 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

9.4 恢复的实现技术-数据转储

- 什么是动态转储
  - 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

9.4 恢复的实现技术-数据转储

- 动态转储
  - 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
  - 动态转储的缺点
    - 不能保证副本中的数据正确有效

9.4 恢复的实现技术-数据转储

- 动态转储
  - 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
  - 后备副本加上日志文件才能把数据库恢复到某一时刻的正确状态

9.4 恢复的实现技术-数据转储

- 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

9.4 恢复的实现技术-数据转储

- 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

9.4 恢复的实现技术-登记日志文件

- 什么是日志文件
  - 日志文件(log)是用来记录事务对数据库的*更新操作*的文件，由系统自动记录。
- 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

9.4 恢复的实现技术-登记日志文件

- 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)

9.4 恢复的实现技术-登记日志文件

- 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）

9.4 恢复的实现技术-登记日志文件

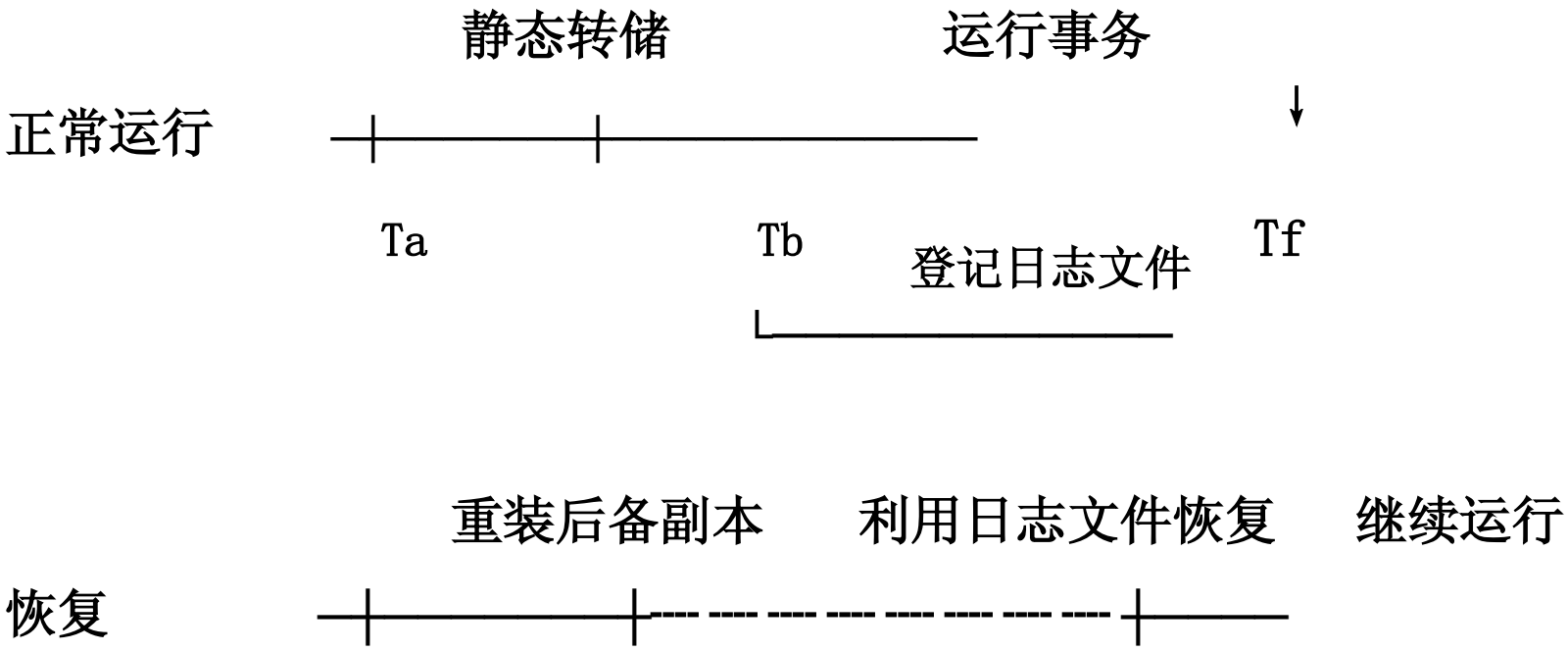
- 以数据块为单位的日志文件，每条日志记录的内容
  - 事务标识（标明是那个事务）
  - 被更新的数据块

9.4 恢复的实现技术-登记日志文件

- 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

9.4 恢复的实现技术-登记日志文件

- **利用静态转储副本和日志文件进行恢复**



9.4 恢复的实现技术-登记日志文件

- 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
    - 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

9.4 恢复的实现技术-登记日志文件

- 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

- **例数据库中有两个元素A和B，这两个元素要满足的约束是在任何一致的状态中它们的值相等。**
  - **事务T逻辑上由下述两步构成：**

    A := A*2; B := B*2;
  - **将T表述为六个相关步骤的序列：**

    READ(A, t); t := t*2; WRITE(A, t);

    READ(B, t); t := t*2; WRITE(B, t);
  - **此外，缓冲区管理器最终将执行OUTPUT步骤，将这些缓冲区写回磁盘。**

step	action	t	M-A	M-B	D-A	D-B	LOG
1							<START T>
2	READ(A,t)	8	8		8	8	
3	t:=t*2	16	8		8	8	
4	WRITE(A,t)	16	16		8	8	<T, A, 8,16>
5	READ(B,t)	8	16	8	8	8	
6	t:=t*2	16	16	8	8	8	
7	WRITE(B,t)	16	16	16	8	8	<T, B, 8,16>
8	FLUSHLOG						
9	OUTPUT(A)	16	16	16	16	8	
10							<COMMIT T>
11	FLUSHLOG						
12	OUTPUT(B)	16	16	16	16	16	

第九章 数据库恢复技术

9.1 事务的基本概念

9.2 数据库恢复概述

9.3 故障的种类

9.4 恢复的实现技术

9.5 恢复策略

9.6 具有检查点的恢复技术

9.7 数据库镜像

9.5 恢复策略

- 事务故障的恢复
- 系统故障的恢复
- 介质故障的恢复

9.5 恢复策略-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
  - 由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

9.5 恢复策略-事务故障的恢复

- 事务故障的恢复步骤
  - 反向扫描文件日志（即从最后向前扫描日志文件），查找该事务的更新操作。
  - 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

9.5 恢复策略-事务故障的恢复

- 事务故障的恢复步骤
  - 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
  - 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

9.5 恢复策略-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没来得及写入数据库

9.5 恢复策略-系统故障的恢复

- 恢复方法
  - Undo 故障发生时未完成的事务
  - Redo 已完成的事务
- 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

9.5 恢复策略-系统故障的恢复

- **系统故障的恢复步骤**
  - **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
      - 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
      - 这些事务只有BEGIN TRANSACTION记录，无相应的COMMIT记录

9.5 恢复策略-系统故障的恢复

- **系统故障的恢复步骤**
  - **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

9.5 恢复策略-系统故障的恢复

- **系统故障的恢复步骤**
  - **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

9.5 恢复策略-介质故障的恢复

- 重装数据库
- 重做已完成的事务

9.5 恢复策略-介质故障的恢复

- **恢复步骤**
  - **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性状态**
    - 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，还须同时装入转储时刻的日志文件副本，利用与恢复系统故障的方法（即REDO+UNDO），才能将数据库恢复到一致性状态。

9.5 恢复策略-介质故障的恢复

- **恢复步骤**
  - **装入有关的日志文件副本(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

9.5 恢复策略-介质故障的恢复

- 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

第九章 数据库恢复技术

9.1 事务的基本概念

9.2 数据库恢复概述

9.3 故障的种类

9.4 恢复的实现技术

9.5 恢复策略

9.6 具有检查点的恢复技术

9.7 数据库镜像

9.6 具有检查点的恢复技术

- 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

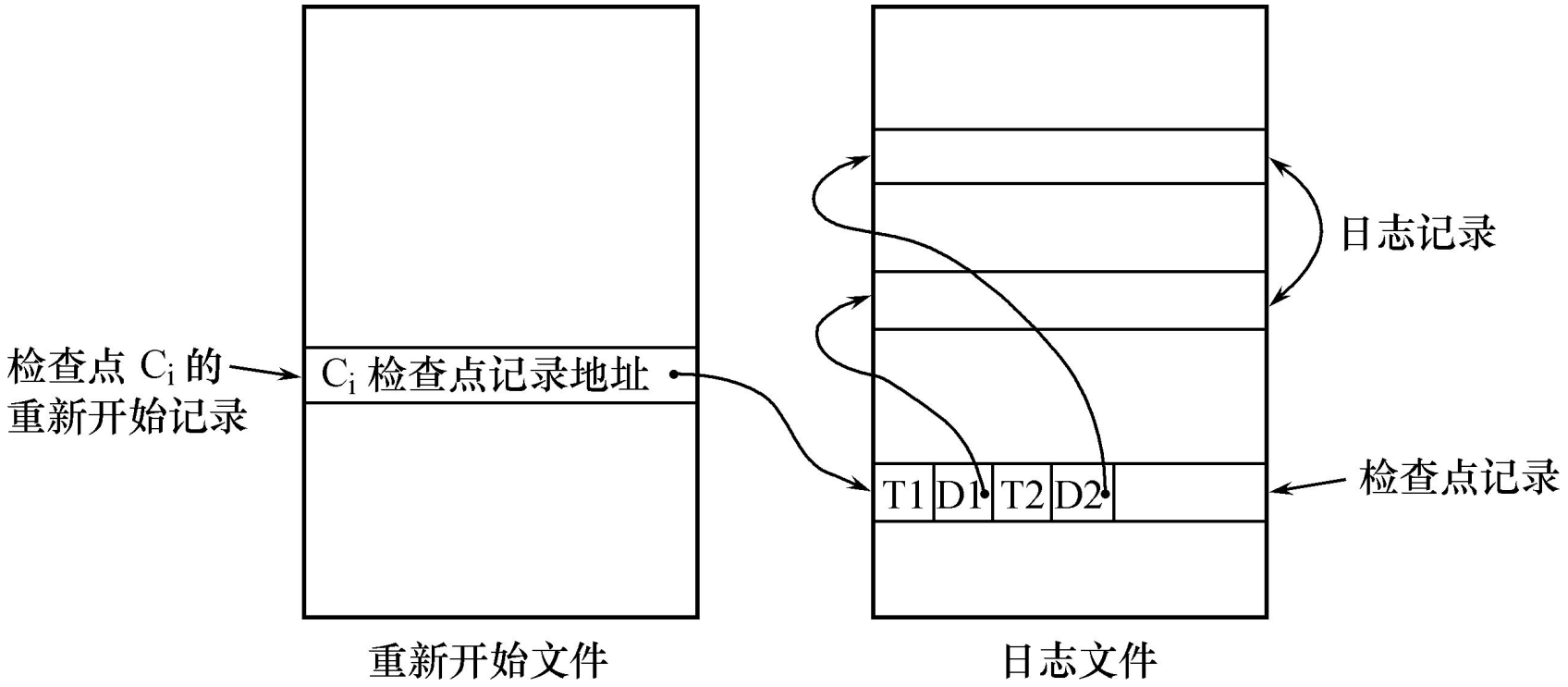
9.6 具有检查点的恢复技术

- 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
  - 恢复子系统在登录日志文件期间动态地维护日志

9.6 具有检查点的恢复技术

- 检查点记录的内容
  - 建立检查点时刻所有正在执行的事务清单
  - 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

9.6 具有检查点的恢复技术



具有检查点的日志文件和重新开始文件

9.6 具有检查点的恢复技术

- 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。

9.6 具有检查点的恢复技术

- 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一个重新开始文件

9.6 具有检查点的恢复技术

- 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

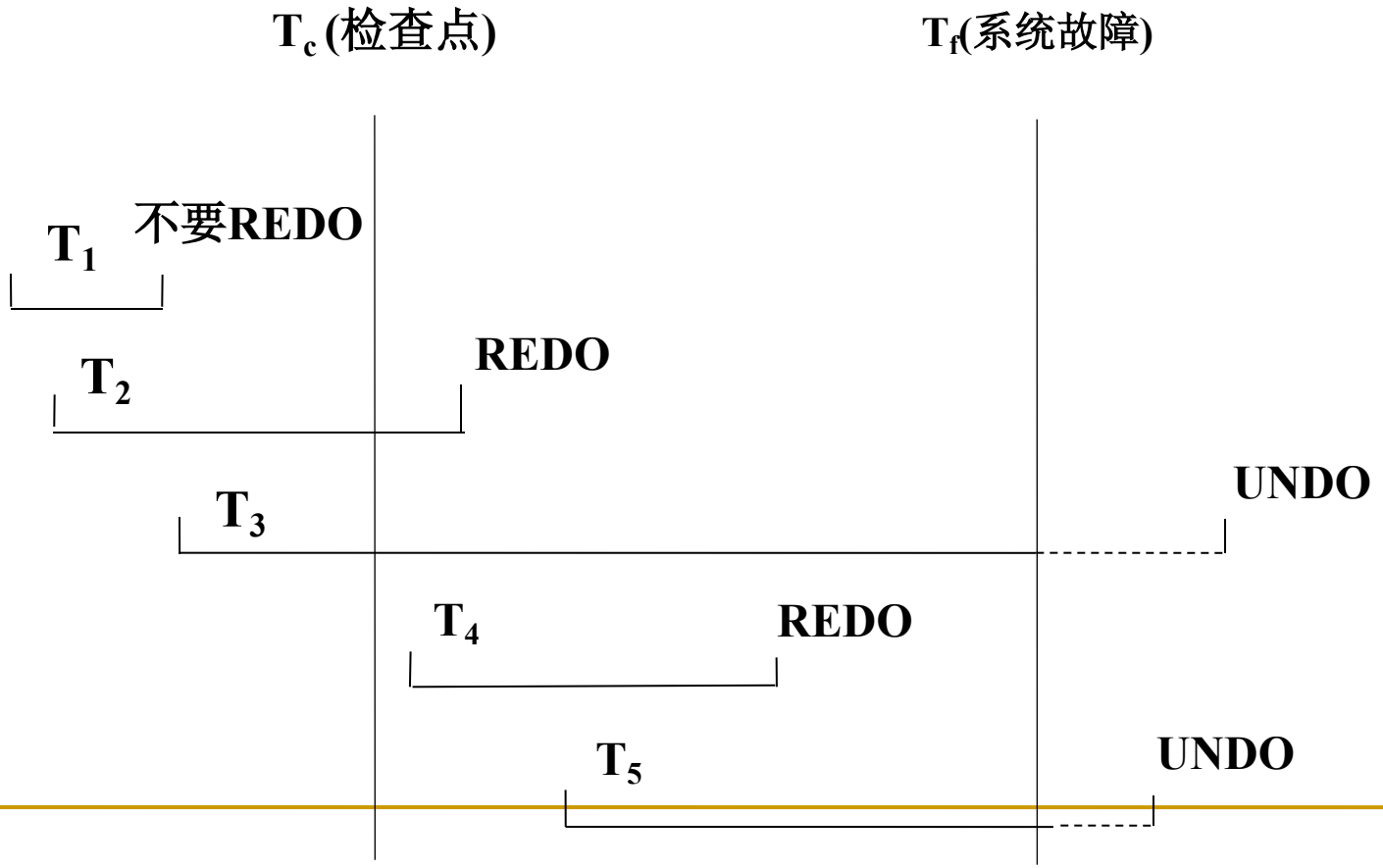
9.6 具有检查点的恢复技术

- 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

9.6 具有检查点的恢复技术

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



9.6 具有检查点的恢复技术

- 利用检查点的恢复步骤
  - 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

9.6 具有检查点的恢复技术

- **利用检查点的恢复步骤**
  - **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**

9.6 具有检查点的恢复技术

- 利用检查点的恢复步骤
  - 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

9.6 具有检查点的恢复技术

- 利用检查点的恢复步骤
  - 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

第九章 数据库恢复技术

9.1 事务的基本概念

9.2 数据库恢复概述

9.3 故障的种类

9.4 恢复的实现技术

9.5 恢复策略

9.6 具有检查点的恢复技术

9.7 数据库镜像

9.7 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

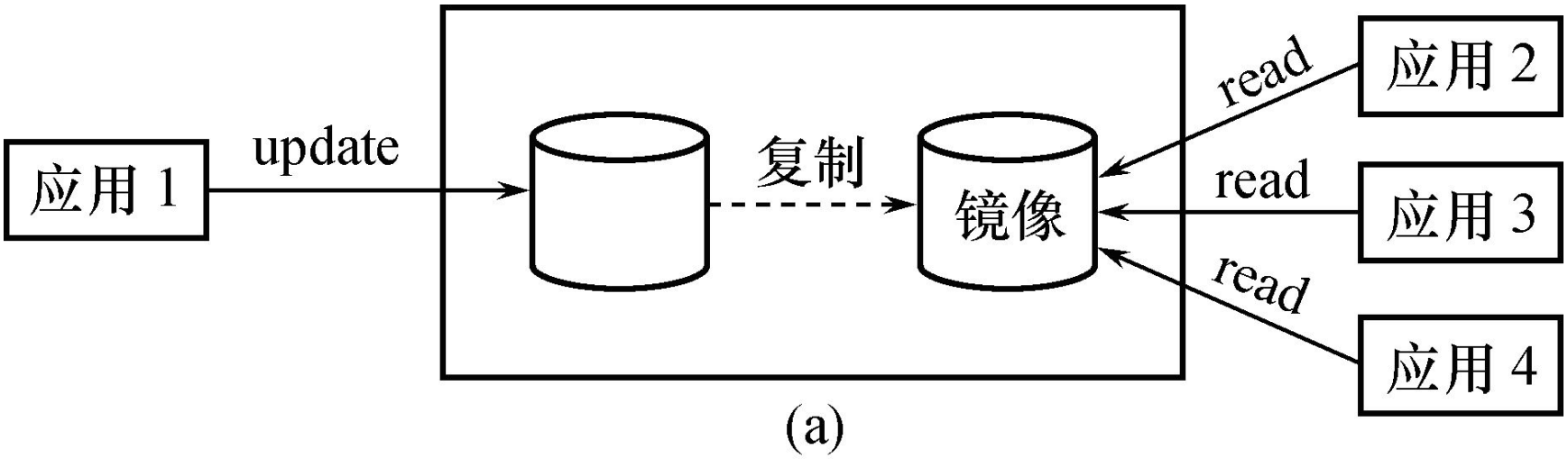
- 提高数据库可用性的解决方案
  - 数据库镜像（Mirror）

9.7 数据库镜像

- 数据库镜像
  - DBMS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
  - DBMS自动保证镜像数据与主数据库的一致性

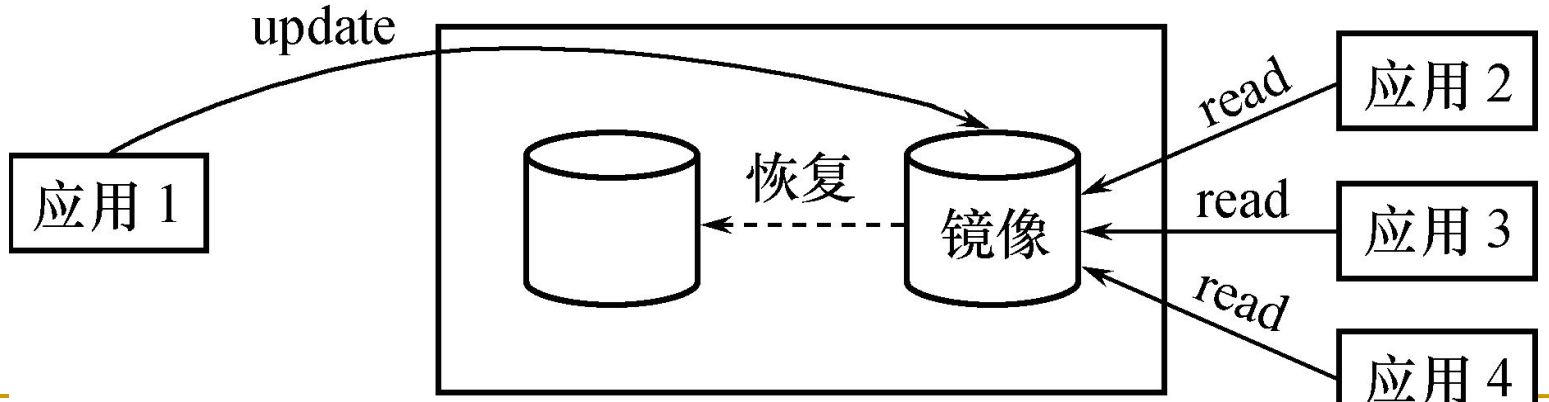
    每当主数据库更新时，DBMS自动把更新后的数据复制过去

9.7 数据库镜像



9.7 数据库镜像

- 出现介质故障时
  - 可由镜像磁盘继续提供使用
  - 同时DBMS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



(b)

9.7 数据库镜像

- 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

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

- 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

- 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

- 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