数据库系统原理

第10章　数据库恢复技术

分级通关平台，QQ群

基础篇

第1章 绪论
第2章 关系数据库*3
第3章 标准语言SQL*3
第4章 数据库安全性
第5章 数据库完整性
实验1

设计与应用开发篇

第6章 关系数据理论*2
第7章 数据库设计
实验2
第8章 数据库编程

系统篇

第9章 *关系查询处理和优化
实验3
第10章 数据库恢复技术
第11章 并发控制
实验4

内容提要

- ✓ 事务的基本概念
- ✓ 数据库恢复概述
- ✓ 恢复策略
- ✓ 数据库镜像

事务 (Transaction)

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

事务(Transaction)

- 显式定义方式

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

●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

```
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

- 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
- 事务滚回到开始时的状态

事务(Transaction)

- 原子性(Atomicity)
  - ✓ 事务中包括的诸操作要么都做，要么都不做
- 一致性(Consistency)
  - ✓ 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.数据库中只包含成功事务提交的结果
- 隔离性(Isolation)
  - ✓ 并发执行的各个事务之间不能互相干扰
- 持续性(Durability)
  - ✓ 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

银行转帐：从A帐户中取出一万元存入B帐户。

- 这两个操作要么全做，要么全不做

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

恢复

- 故障的影响
  - 故障是不可避免的
  - 造成事务非正常中断，数据库全部或部分丢失数据
- 数据库的恢复是数据库管理系统的重要功能
  - 把数据库从错误状态恢复到某一已知的正确状态
  - 恢复技术是衡量系统优劣的重要指标

1. 事务内部的故障 ➢ 事务操作无法完成
2. 系统故障 ➢ 系统无法正常运行，包括软硬件
3. 介质故障 ➢ 外存/磁盘损坏,数据丢失
4. 计算机病毒 ➢ 复杂的不确定范围的数据篡改

恢复

1. 事务内部的故障
   - 软故障
   - 事务没有达到预期的终点(COMMIT或者显式的ROLLBACK)
   - 事务撤消（UNDO）：强行回滚（ROLLBACK）该事务
2. 系统故障
   - 软故障
   - 系统重新启动时，恢复程序让所有非正常终止的事务回滚，强行撤消（UNDO）所有未完成事务
   - 恢复策略：系统重新启动时，恢复程序需要重做（REDO）所有已提交的事务
3. 介质故障　4. 计算机病毒
   - 可能是硬故障——数据库恢复

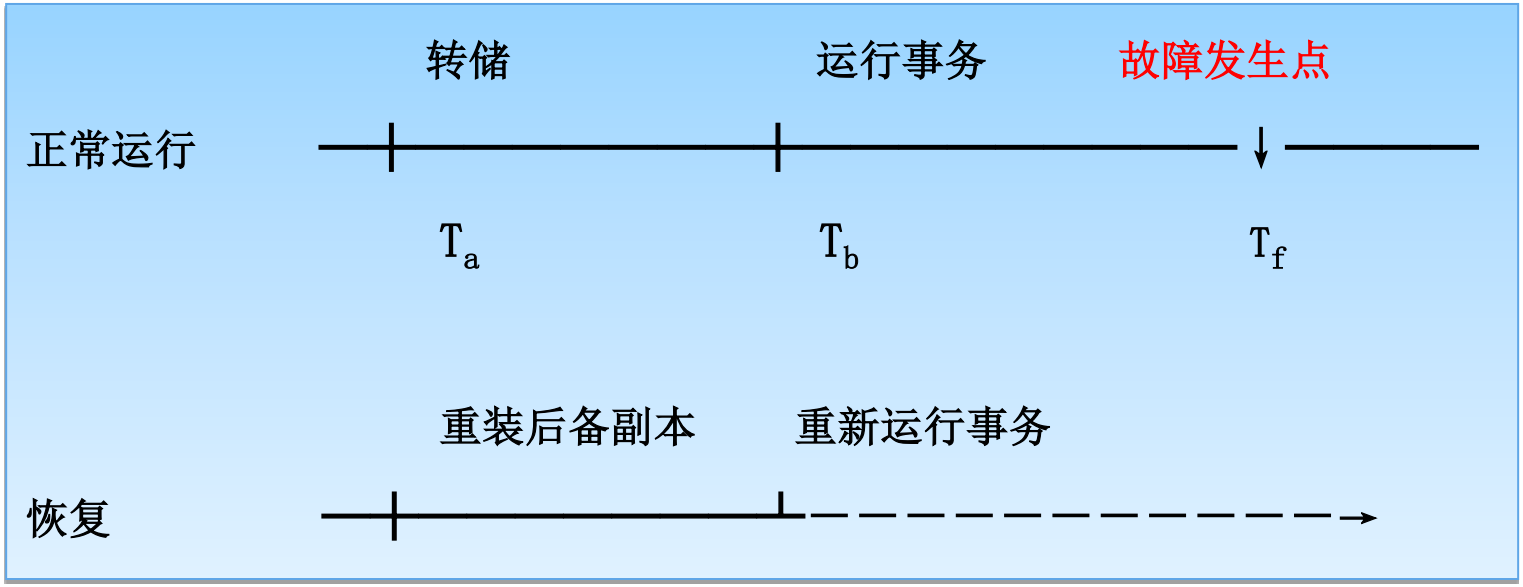
恢复

- 恢复操作的基本原理：冗余
  - 利用存储在系统别处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
- 恢复的实现技术：复杂
  - 一个大型数据库产品，恢复子系统的代码要占10%以上

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

恢复

——转储



- 转储是指数据库管理员定期地将整个数据库复制到存储介质上保存起来的过程，备用的数据称为后备副本(backup)
- 将后备副本重新装入，将数据库恢复到转储时的状态；然后重新运行自转储以后的所有成功的事务

恢复--转储

- 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

恢复--转储

- 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
  - 动态转储的缺点
    - 不能保证副本中的数据正确有效
    - 例：在转储期间的某时刻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。后备副本上的A过时了
  - 把动态转储期间各事务对数据库的修改活动记录到日志文件
  - 后备副本加上日志文件就能把数据库恢复到某一时刻的正确状态

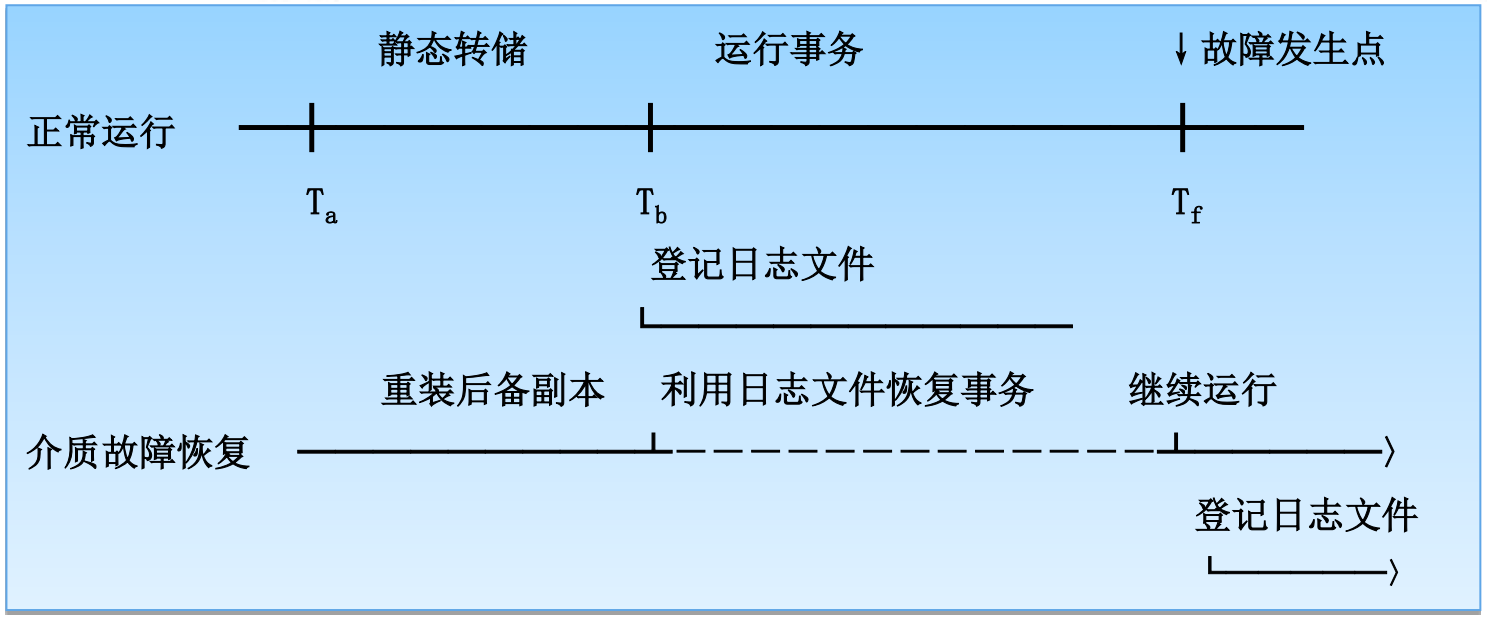
恢复--转储

- 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更方便
  - 如果数据库很大，事务处理又十分频繁，则增量转储更实用

转储方式	转储状态	
	动态转储	静态转储
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

恢复

--日志



- 日志文件(log file)：用来记录事务对数据库的更新操作的文件
  - 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
    - ✓ 不必重新运行那些已完成的事务

恢复--日志

- 以记录为单位的日志文件
  - 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ID、Block NO.）
  - 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）
- 以数据块为单位的日志文件
  - 事务标识
  - 被更新的数据块

恢复--日志

- 为保证数据库是可恢复的，记录日志文件时必须遵循
  - 记录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：把本次修改的记录到日志文件中
    - 写数据库操作：把对数据的修改写到数据库中

- 为什么要先写日志文件
  - 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，则恢复时多执行一次UNDO

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
- 系统故障的恢复
  - 系统故障：数据库不一致状态，数据更新的一部分写入数据库，其余部分还在缓冲区
- 介质故障的恢复
  - 介质故障：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

恢复策略

- 事务故障的恢复
  - 事务故障的恢复由系统自动完成，对用户是透明的
  - **利用日志文件撤消（UNDO）此事务已对数据库进行的修改**

1） 从日志最反向向前扫描日志文件，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。

- 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

3） 继续反向扫描日志文件，并做同样处理，直至读到此事务的开始标记

恢复策略

- 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
  - **Undo 故障发生时未完成的事务；Redo 已完成的事务**

1）正向扫描日志文件

- 既有BEGIN TRANSACTION也有COMMIT->重做(REDO) 队列
- 只有BEGIN TRANSACTION无COMMIT记录->撤销 (UNDO)队列

2）反向扫描日志文件，对UNDO队列事务进行撤销处理

- 将“更新前的值”写入数据库

3）正向扫描日志文件，对REDO队列事务进行重做处理

- 将“更新后的值”写入数据库

恢复策略

- 介质故障的恢复
  - 重装数据库，重做已完成的事务
  - 介质故障的恢复需要数据库管理员介入

1）装入最新的后备数据库副本

- 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，还须利用恢复系统故障的方法恢复

2）装入转储结束时刻的日志文件副本，重做已完成的事务

- 扫描日志文件，找出已提交的事务的标识，记入重做队列
- 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

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（各个检查点记录在日志文件中的地址）
  - 恢复子系统在登录日志文件期间动态地维护日志

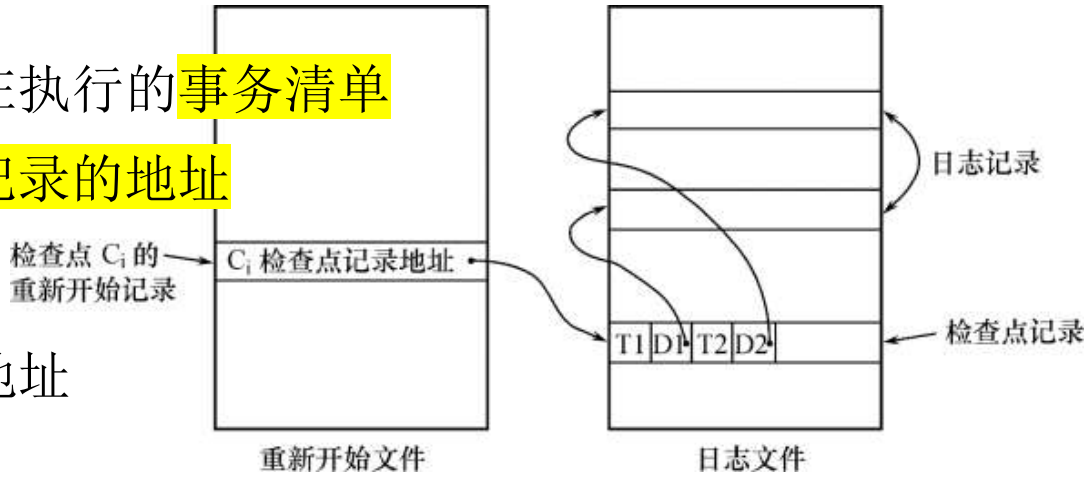
- 利用日志的问题

1）搜索整个日志将耗费大量的时间

2）重做处理：重新执行，浪费了大量时间

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的地址



- 动态维护：周期性建立检查点，保存数据库状态。

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

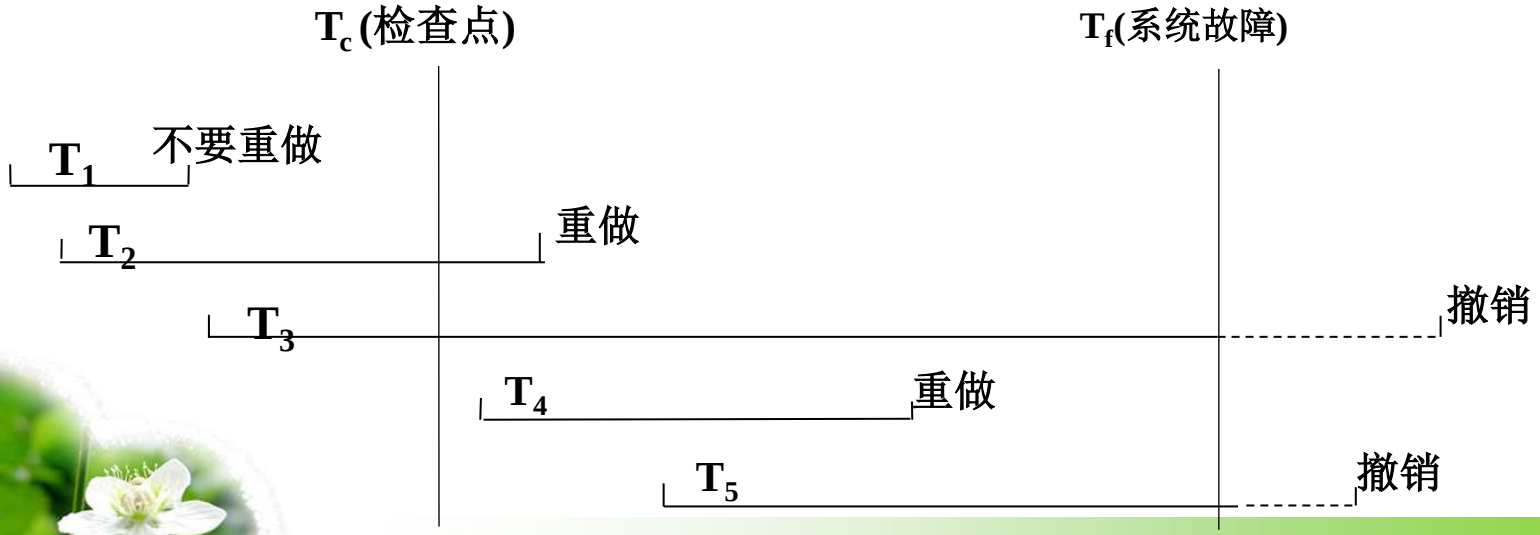
2）在日志文件中写入一个检查点记录

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

具有检查点的恢复技术

- 恢复策略
  - T1在检查点之前已提交，所以不必执行重做操作
  - 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，所以要重做
  - T3和T5在故障发生时还未完成，所以予以撤销



具有检查点的恢复技术

- 恢复操作

T_c(检查点)　　T_f(系统故障)

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

2）由该检查点记录得到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

3）从检查点开始正向扫描日志文件，直到日志文件结束

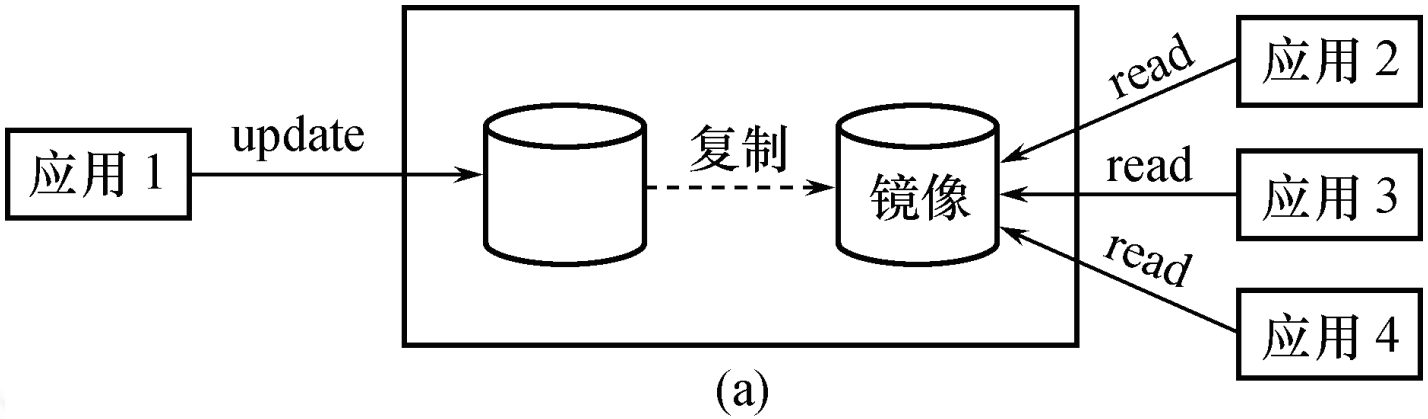
- 如有新开始的事务Ti，把Ti暂时放入UNDO-LIST队列
- 如有提交的事务Tj，把Tj从UNDO-LIST队列移到REDO-LIST队列

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

数据库镜像

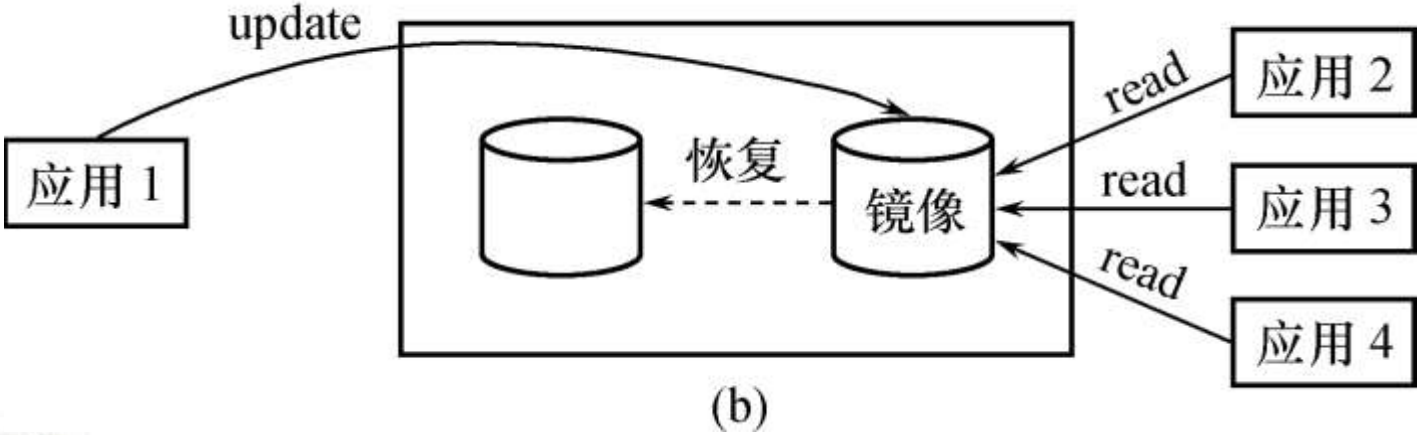
- 介质故障是对系统影响最为严重的一种故障
  - 介质故障恢复比较**费时**
  - 为预防介质故障，数据库管理员必须周期性地**转储**数据库
- 提高数据库可用性的解决方案
  - 数据库镜像（Mirror）



(a)

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

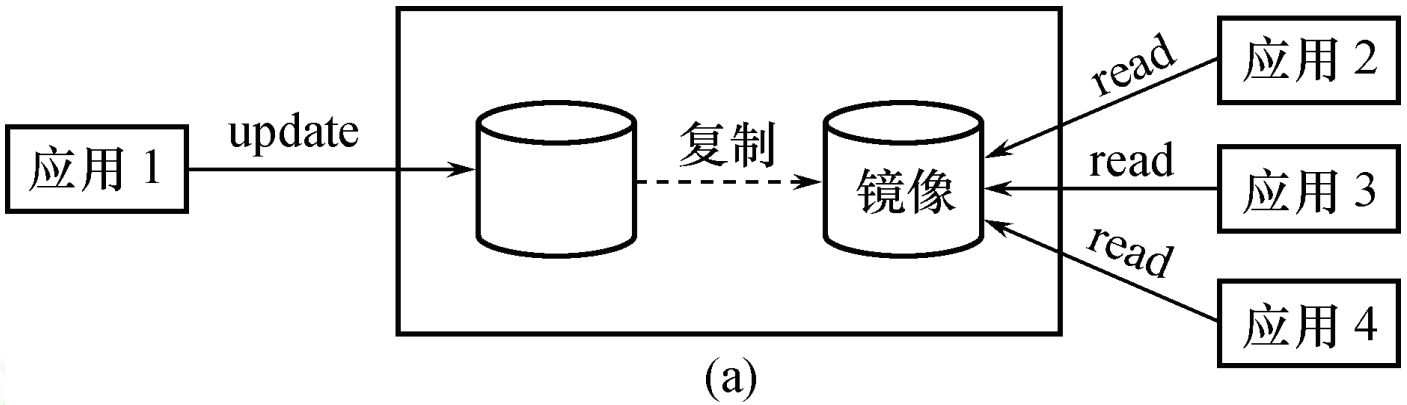


(b)

数据库镜像

- 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

- 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



(a)

小结

- 事务的概念和性质
  - ✓ 事务是数据库的逻辑工作单位
- 故障的种类
  - ✓ 事务故障，系统故障，介质故障
- 恢复策略
  - ✓ 数据库转储，日志文件
  - ✓ 针对各种故障的恢复操作
  - ✓ **具有检查点的恢复技术**
- 数据库镜像