数据库系统

第10章 数据库恢复技术

胡 敏

合肥工业大学

jsjxhumin@hfut.edu.cn

第10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

- 事务
- 事务的ACID特性

10.1 事务的基本概念

- **事务(Transaction)的定义**
  - ✓一个数据库操作序列；
  - ✓一个不可分割的工作单位；
  - ✓恢复和并发控制的基本单位。

```
  xmldoc:Variant;
  recordCounts:array [0..20] of Longint;
  logType:TLogType;
  logMsg,CString:string;
begin
  Result:=False;
  CString:=dmBM.dbTemp.ConnectionString;
  if Importing then
    Exit;
  Importing:=True;
  try
    xmldoc := CreateOleObject('Micros
    dmBM.dbImport.Connected:=True;
    dmBM.dbImport.BeginTrans;
    try
      xmldoc.async := False;
      xmldoc.load(filename);
      if (xmldoc.parseError.errorCode
        raise Exception.Create('读取数
            '错误原因: ' + xmldoc.par
            '错误文本: ' + xmldoc.par

      ImportDzb(xmldoc,clear,recordCo

      dmBM.dbImport.CommitTrans;
      logType:=ltSuccess;
      logMsg:='更新对照表成功。更新记
      logMsg:=logMsg+#13#10+Format('
      logMsg:=logMsg+#13#10+Format('
      logMsg:=logMsg+#13#10+Format('专业记录%d条',[recordCounts[2]]);
      logMsg:=logMsg+#13#10+Format('区县记录%d条',[recordCounts[3]]);
      logMsg:=logMsg+#13#10+Format('毕业类别记录%d条',[recordCounts[4]]);
      logMsg:=logMsg+#13#10+Format('招生范围记录%d条',[recordCounts[5]]);
      logMsg:=logMsg+#13#10+Format('层次记录%d条',[recordCounts[6]]);
      logMsg:=logMsg+#13#10+Format('民族记录%d条',[recordCounts[7]]);
      logMsg:=logMsg+#13#10+Format('外语语种记录%d条',[recordCounts[8]]);
      logMsg:=logMsg+#13#10+Format('考生特征记录%d条',[recordCounts[9]]);
      logMsg:=logMsg+#13#10+Format('政治面貌记录%d条',[recordCounts[10]]);
      logMsg:=logMsg+#13#10+Format('职业类别记录%d条',[recordCounts[11]]);
      logMsg:=logMsg+#13#10+Format('文化程度记录%d条',[recordCounts[12]]);
      logMsg:=logMsg+#13#10+Format('招生类别记录%d条',[recordCounts[13]]);
      logMsg:=logMsg+#13#10+Format('科类记录%d条',[recordCounts[14]]);
      logmsg:=logmsg+#13#10+format('科类考试科目%d条',[I

      Result:=True;
    except
      on E:Exception do
      begin
        // dmBM.dbImport.RollbackTrans;
        logType:=ltError;
        logMsg:='更新对照表数据发生异常:'+E.Message;
      end;
    end;
    xmldoc:=UnAssigned;
    dmBM.dbImport.Connected:=False;

    LogMessage(deDownload,logType,logMsg,lmMsgBox);
  finally
    Importing:=False;
```

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
- 事务中所有对数
  写回到磁盘上的

上
过程中发生了故障

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

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

2.事务的特性（ACID特性）

- 原子性（Atomicity）
- 一致性（Consistency）
- 隔离性（Isolation）
- 持续性（Durability ）

（1）事务的原子性

- **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）事务的一致性

- 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正确的状态

（3）隔离性

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**

T1	T2
① 读A=16	
②	读A=16
③ A←A-1 写回A=15	
④	A←A-3 写回A=13
	B=B+1

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

(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

第10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
  - 系统软件和应用软件的错误
  - 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
  - 破坏数据库

10.2 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管理系统对故障的对策

- 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10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

10.3 故障的种类

1. 事务内部的故障
2. 系统故障
3. 介质故障
4. 计算机病毒故障的影响

1. 事务内部的故障

- 事务内部的故障
  - 某个事务在运行过程中由于种种原因未运行至正常终止点就夭折了
- 事务内部故障的常见原因
  - 输入数据有误
  - 运算溢出
  - 违反了某些完整性限制
  - 某些应用程序出错
  - 并行事务发生死锁

  …… ……

- 事务内部故障的恢复
  - 发生事务故障时，夭折的事务可能已把对数据库的部分修改写回磁盘
    - ★事务没有达到预期的终点(COMMIT或者显式的ROLLBACK)
    - ★ 数据库可能处于不正确状态。
  - 事务故障的恢复：撤消事务（UNDO）
  - 强行回滚（ROLLBACK）该事务
  - 清除该事务对数据库的所有修改，使得这个事务象根本没有启动过一样

系统故障

- 系统故障，常称为软故障（Soft Crash）
  - 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
- 系统故障的常见原因
  - 操作系统或DBMS代码错误
  - 操作员操作失误
  - 特定类型的硬件错误（如CPU故障）
  - 突然停电

所有活跃事务都只运行了一部分，没有全部完成。

部分已完成事务更新后的数据还在缓冲区中，没有来得及刷到硬盘上，这些更新就丢失了

系统故障的恢复

- 发生系统故障时，一些尚未完成的事务的结果可能已送入物理数据库，造成数据库可能处于不正确状态。
  - 恢复策略：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
- 发生系统故障时，有些已完成的事务可能有一部分甚至全部留在缓冲区，尚未写回到磁盘上的物理数据库中，系统故障使得这些事务对数据库的修改部分或全部丢失
  - 恢复策略：系统重新启动时，恢复程序需要重做

系统故障的恢复需要做两件事情：

1. 撤销所有未完成的事务
2. 重做所有已提交的事务

介质故障

- 介质故障称为硬故障（Hard Crash）
  - 磁盘损坏、
  - 磁头碰撞等
- 硬件故障使存储在外存中的数据部分丢失或全部丢失
- 介质故障比前两类故障的可能性小得多，但破坏性大得多
- 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

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，造成对计算机系统包括数据库的危
- 害计算机病毒是具有破坏性，计算机病毒已成为计算机系统的主要威胁，自然也是数据库系统的主要威胁。
- 数据库一旦被破坏仍要用恢复技术把数据库加以恢复。

总结各类故障

- 各类故障对数据库的影响有两种可能性：
  - 数据库本身被破坏
  - 数据库没有被破坏，但数据可能不正确

故障恢复

- 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全部代码的10%以上

第10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

恢复的实现技术

- 恢复机制涉及的关键问题

1. 如何建立冗余数据

★数据转储

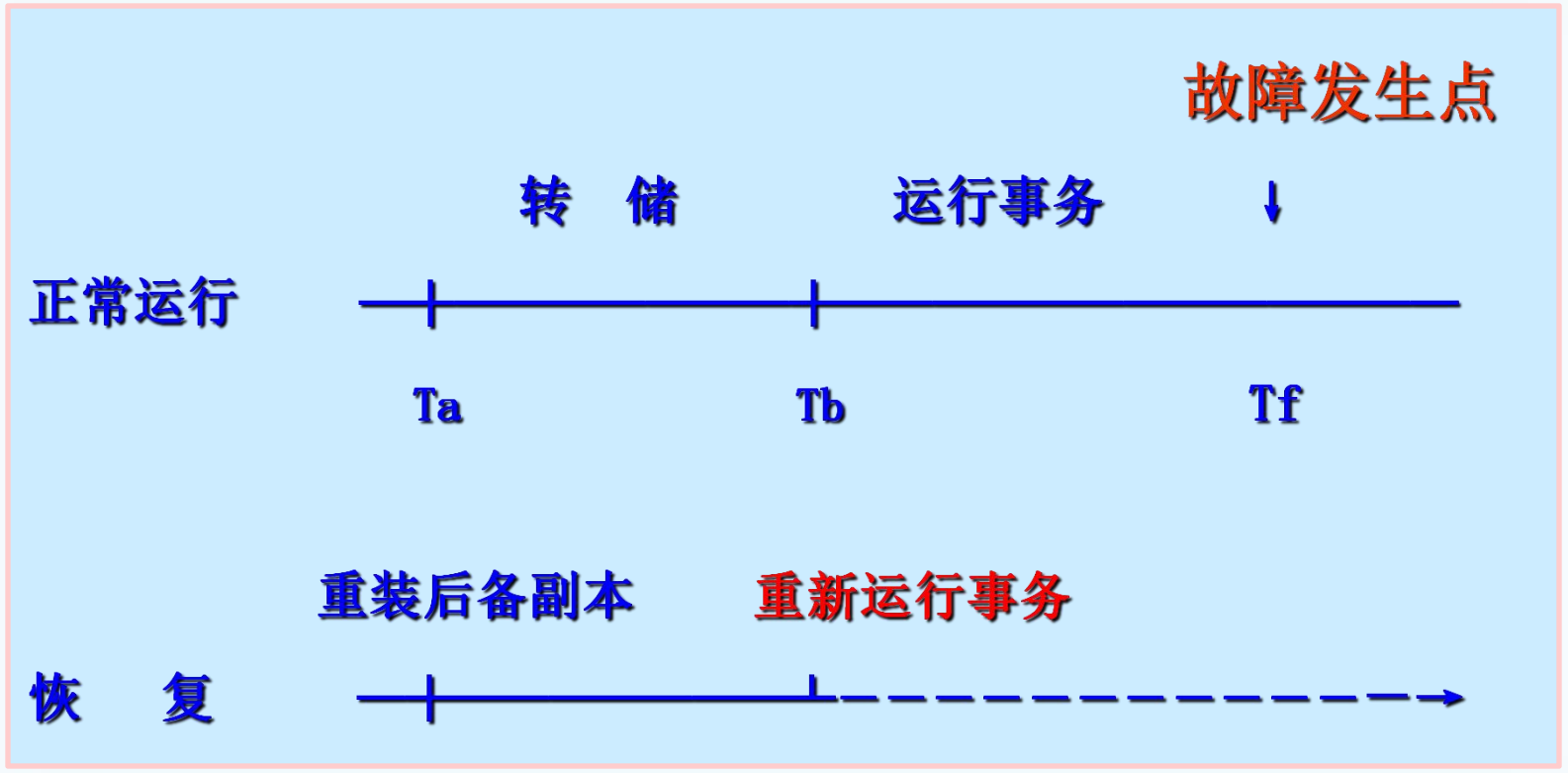
★登录日志文件

2. 如何利用这些冗余数据实施数据库恢复

10.4.1 数据转储

- 数据转储是指DBA定期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。
- 当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；
- 要想恢复到故障发生时的状态，必须重新运行自转储以后的所有更新事务。

转储和恢复



2.转储的方法

- （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

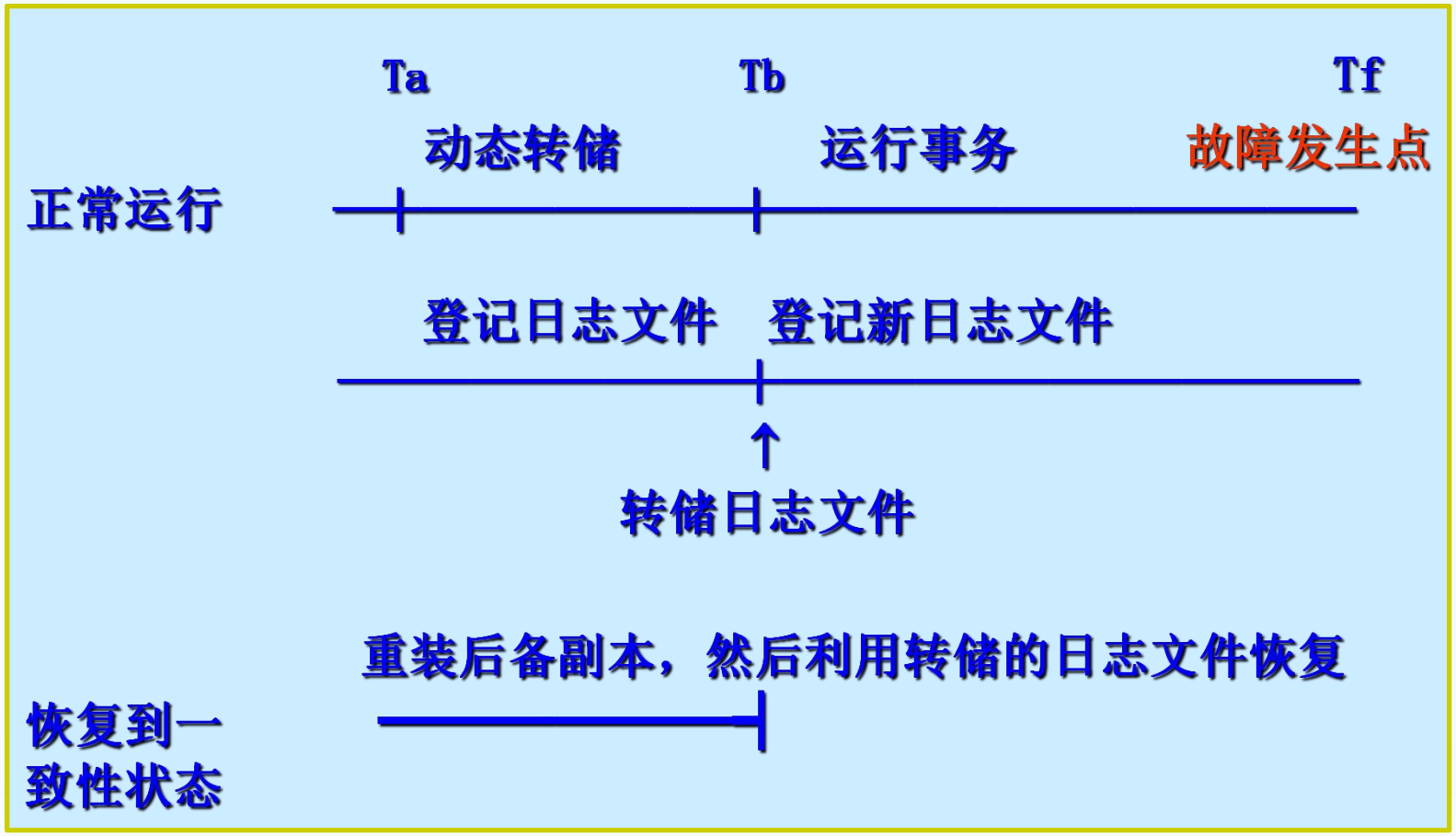
转储的方法

- 静态转储与动态转储
- 静态转储:即在系统中无运行事务时进行转储
  - 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
- 优点：实现简单
- 缺点：降低了数据库的可用性
  - 转储必须等用户事务结束
  - 新的事务必须等转储结束

动态转储

- 动态转储，即转储期间允许对数据库进行存取或修改。转储操作与用户事务并发执行。
- 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
- 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
  - 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

利用动态转储副本进行恢复



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

转储策略

- 转储策略
  - 应定期进行数据转储，制作后备副本。
  - 但转储又十分耗费时间和资源，不能频繁进行。
  - DBA应该根据数据库使用情况确定适当的转储周期和转储方法。
  - 例如：
    - ★ 每天晚上进行动态增量转储
    - ★ 每周进行一次动态海量转储
    - ★ 每月进行一次静态海量转储

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

在数据转储效率、数据库运行效率、故障恢复效率三个方面各有利弊

DBA通常会根据数据库使用情况，确定一个适当的转储周期，并配合使用这类4类方法

登记日志文件

- 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

以记录为单位的日志文件

- 以记录为单位的日志文件内容
  - 各个事务的开始标记(BEGIN TRANSACTION) (事务标志 + BEGIN TRANSACTION)

  如： T1 BEGIN TRANSACTION
  - 各个事务的结束标记(COMMIT或ROLLBACK) 如：T1 COMMIT；T2 ROLLBACK
  - 各个事务的所有更新操作
- 每个日志记录的内容主要包括：
  - 事务标识 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，此项为空值）
  - 更新后数据的新值（对删除操作，此项为空值）

  例： T1 U AA 18 20

  T1 I TU 1

  T1 D TV 20

以数据块为单位的日志文件

- 以数据块为单位的日志文件，日志记录的内容：
  - 事务标识
  - 被更新的数据块
  - 更新前整个数据块的值
  - 更新后整个数据块的值

2.日志文件的用途

- 日志文件在数据库恢复中起着非常重要的作用，可以用来进行事务故障恢复和系统故障恢复，并协助后备副本进行介质故障恢复。具体地讲：
  - 事务故障恢复和系统故障必须用日志文件。
  - 在动态转储方式中必须建立日志文件，后援副本和日志文件综合起来才能有效地恢复数据库。
  - 在静态转储方式中，也可以建立日志文件。

登记日志文件的原则

- 为保证数据库是可恢复的，登记日志文件时必须遵循两条原则：
  - 登记的次序严格按并行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

为什么要先写日志文件

- 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

第10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中止
- 恢复方法
  - 由恢复子系统利用日志文件撤消（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，不需要用户干预

10.5.1事务故障的恢复策略

■ 事务故障的恢复步骤：

1. 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。

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，相当于做删除操作
- 删除操作“更新后的值”为空，相当于做插入操作
- 若是修改操作，则用用修改前值代替修改后值

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。

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
  - 一些未完成事务对数据库的更新已写入数据库
  - 一些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
- 恢复方法
  - Undo 故障发生时未完成的事务
  - Redo已提交的事务
- 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

10.5.3介质故障的恢复

1. 重装数据库
2. 重做已完成的事务

10.5.3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。

(2)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。

故障恢复小结

- 事务故障的恢复

  UNDO

- 系统故障的恢复

  UNDO + REDO

- 介质故障的恢复

  重装后援副本 + REDO

第10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

- 1.问题的提出
- 2.检查点技术
- 3.利用检查点的恢复策略

1. 问题的提出

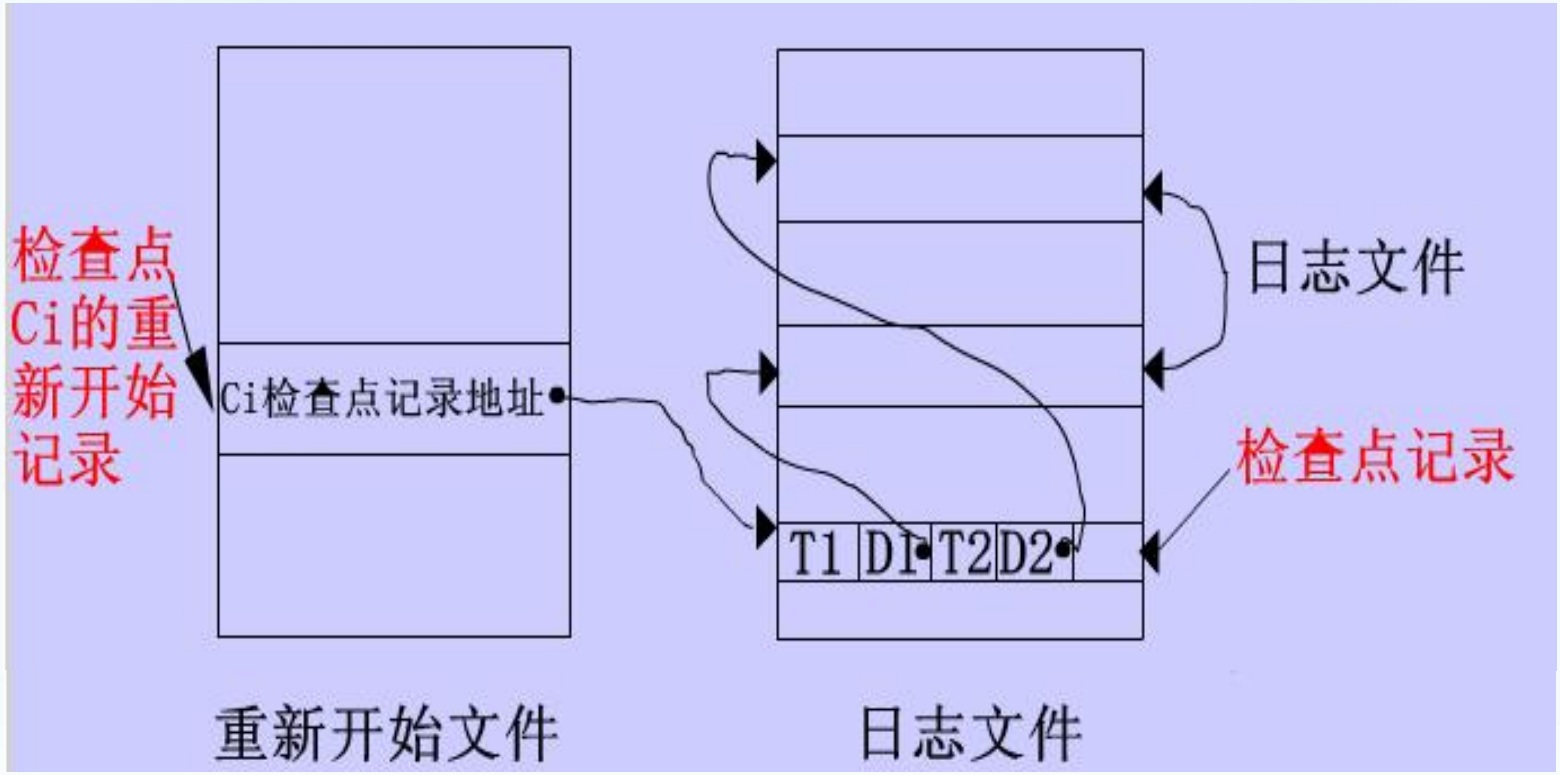
- 两个问题
  - 搜索整个日志将耗费大量的时间
  - REDO处理：重新执行，浪费了大量时间
- 解决方案具有检查点（checkpoint）的恢复技术
  - 在日志文件中增加检查点记录（checkpoint）
  - 增加一个重新开始文件
  - 让恢复子系统在登录日志文件期间动态地维护日志

2.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

具有检查点的恢复技术

- 图中说明了建立检查点Ci时对应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

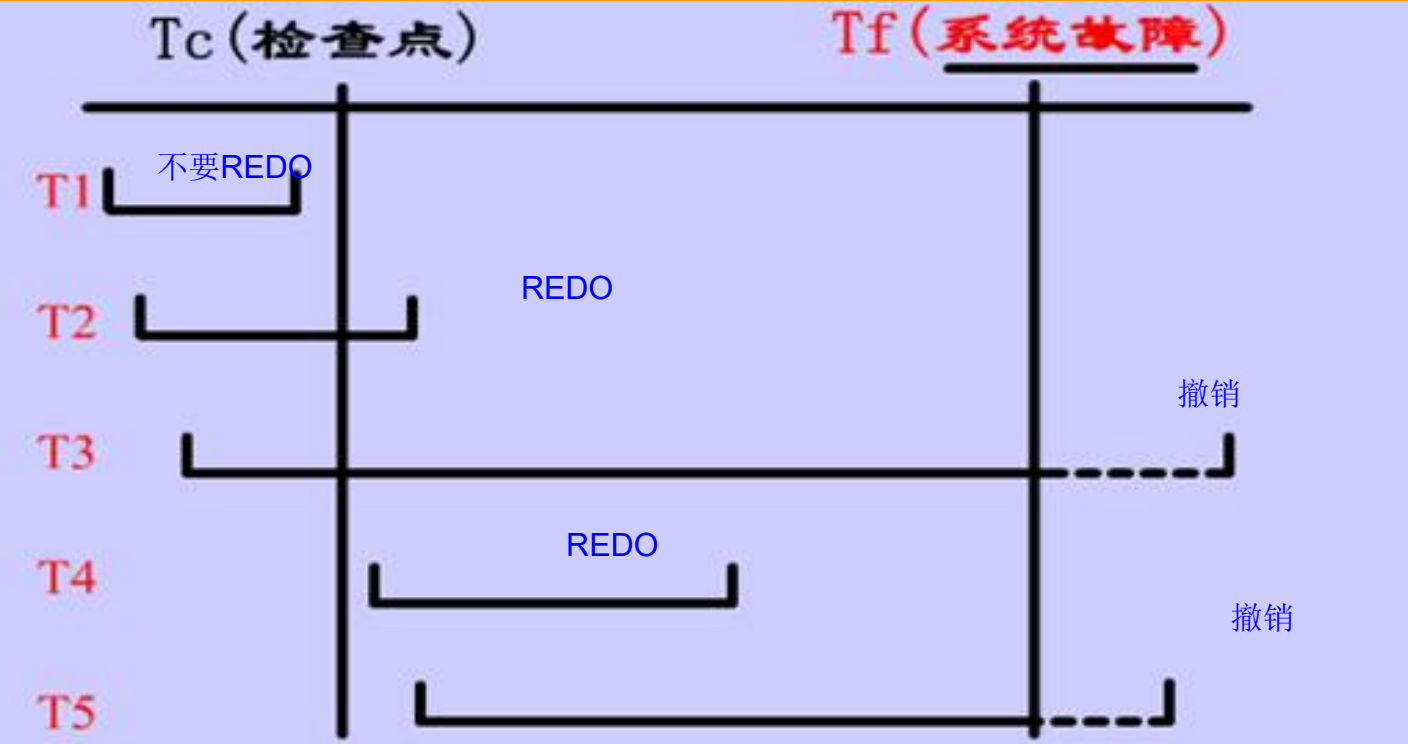
  （4）把检查点记录在日志文件中的地址写入一个重新开始文件

✓ 3.利用检查点的恢复策略

■ 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。
    - 在进行恢复处理时，没有必要对事务T执行REDO操作
  - ✓ 当事务T在检查点时还没有完成
    - T对数据库所做的修改已写入数据库
    - 在进行恢复处理时，如果需要重做T，重做的起始点是检查点。

具有检查点的恢复技术



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

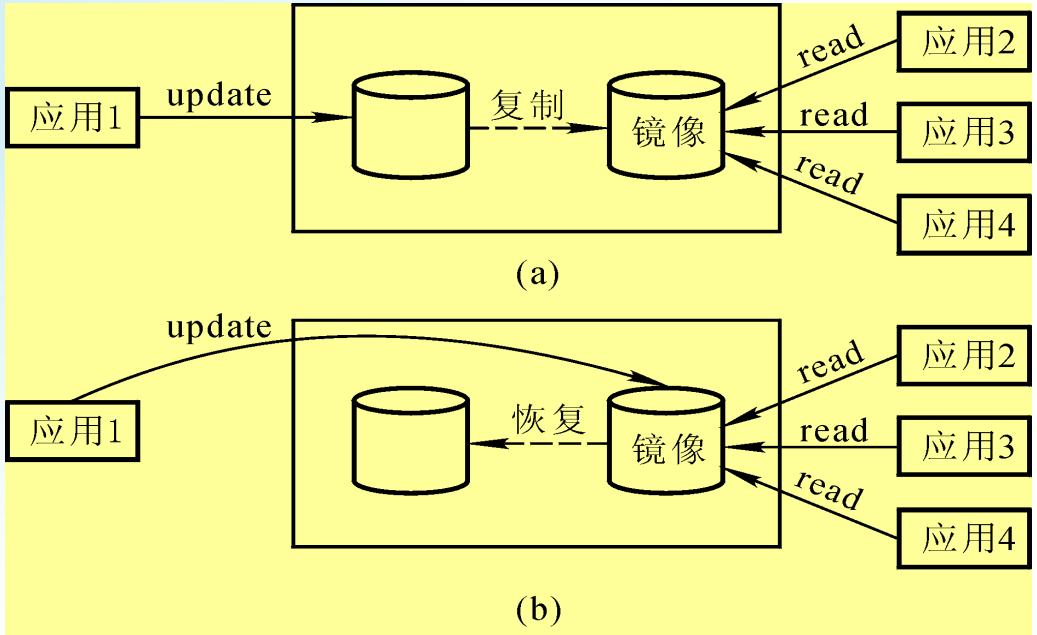
系统出现故障时，恢复子系统将根据事务的不同状态采取不同的恢复策略

第10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

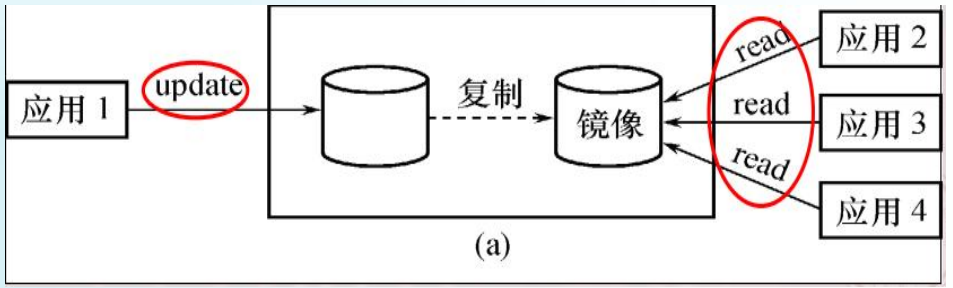
10.7 数据库镜像

- 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



数据库镜像的用途

- 出现介质故障时
  - 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
- 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



由于数据库镜像是通过复制数据实现的，频繁地复制数据自然会降低系统运行效率，因此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

- 事务的概念和性质
  - 事务是数据库的逻辑工作单位
  - 数据库管理系统保证系统中一切事务的原子性、一致性、隔离性和持续性，就保证了事务处于一致状态
- 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
- 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

总 结

- 本章目标
  - 掌握事务的基本概念。
  - 掌握数据库运行中可能产生的故障类型及其恢复技术
- 本章重点
  - 牢固掌握事务的性质。数据库恢复的实现技术。
  - 举一反三：恢复的基本原理，针对不同故障的恢复策略
- 本章难点
  - 日志文件的使用，系统故障恢复策略

作业

- P305：1、3、10

下课了。。

休息。。。