



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 数据库镜像

为什么需要事务

- 例如，银行转账问题：

假定资金从账户A转到账户B，至少需要两步：

- 账户A的资金减少
- 然后账户B的资金相应增加



- 假定张三的账户直接转账1000元到李四的账户

```
CREATE TABLE bank
```

```
(  customerName CHAR(10), --顾客姓名  
  currentMoney MONEY    --当前余额  
)
```

创建账户表，存放用户的账户信息

```
GO
```

```
ALTER TABLE bank
```

```
  ADD CONSTRAINT CK_currentMoney  
      CHECK(currentMoney>=1)
```

添加约束：根据银行规定，账户余额不能少于1元，否则视为销户

```
GO
```


```
INSERT INTO bank(customerName,currentMoney)
```

```
  VALUES('张三',1000)
```

张三开户，开户金额为1000元；李四开户，开户金额

```
INSERT INTO bank(customerName,currentMoney)
```

```
  VALUES('李四',1)
```



customerName	currentMoney
--------------	--------------

张三	1000.00
李四	1.00

(2 行受影响)

- 目前两个账户的余额总和为： $1000+1=1001$ 元

- 模拟实现转账：
从张三的账户转账1000元到李四的账户

```
/*--
customerName currentMoney
-----
张三          1000.00
李四          1.00
```

请问：
执行转账语句后，张三、李四的账户
余额为多少？

消息 547，级别 16，状态 0，第 4 行
UPDATE 语句与 CHECK 约束"CK_currentMoney"冲突。该冲突发生于数据库"stuDB"，
表"dbo.bank"，column 'currentMoney'。
语句已终止。

(1 行受影响)
customerName currentMoney

张三 1000.00
李四 1001.00

张三的账户没有减少
但李四的账户却多了1000元
 $1000 + 1001 = 2001$ 元
总额多出了1000元！

■ 错误原因分析:

消息 547, 级别 16, 状态 0, 第 4 行
UPDATE 语句与 CHECK 约束"CK_currentMoney"冲突。该冲突发生于数据库 "stuDB",
表 "dbo.bank", column 'currentMoney'。
语句已终止。

(1 行受影响)
customerName currentMoney

张三 1000.00
李四 1001.00

UPDATE语句违反约束:
余额>=1元

--张三的账户减少1000元, 李四的账户增加1000元

UPDATE bank

执行失败, 所以张三还是1000元

SET currentMoney=currentMoney-1000

WHERE customerName='张三'

如何解决呢? 使用事务

UPDATE bank

SET currentMoney=currentMoney+1000

继续往下执行: 执行成功, 所以李四变为1001元

WHERE customerName='李四'

GO



10.1 事务的基本概念

一、事务定义

二、事务的特性



一、事务(Transaction)

■ 定义

- 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
- 恢复和并发控制的基本单元

■ 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,也可以包含一个或多个程序
- 一个程序通常包含多个事务



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，DBMS按缺省规定自动划分事务。



事务结束

■ 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永久生效


■ 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
- 回滚事务的所有更新操作
 - 事务滚回到开始时的状态




二、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. 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：数据库中包含失败事务的结果



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



4. 持续性

■ 持续性也称永久性

- 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
 - 多个事务并行运行时，不同事务的操作交叉执行
 - 事务在运行过程中被强行停止

■ 使用事务解决银行转账问题

.....关键语句讲解.....

BEGIN TRANSACTION

/*--定义变量，用于累计事务执行过程中的错误--*/

DECLARE @errorSum INT

SET @errorSum=0 --初始化为0，即无错误

/*--转账：张三的账户少1000元，李四的账户多1000元*/

UPDATE bank SET currentMoney=currentMoney-1000

WHERE customerName='张三'

SET @errorSum=@errorSum+@@error

UPDATE bank SET currentMoney=currentMoney+1000

WHERE customerName='李四'

SET @errorSum=@errorSum+@@error --累计是否有错误

开始事务（指定事务从此处开始，后续的T-SQL语句都是一个整体）

累计是否有错误



```
IF @errorSum<>0 --如果有错误
BEGIN
    print '交易失败，回滚事务'
    ROLLBACK TRANSACTION
END
ELSE
BEGIN
    print '交易成功，提交事务，写入硬盘，永久的保存'
    COMMIT TRANSACTION
END
GO
print '查看转账事务后的余额'
SELECT * FROM bank
GO
```

根据是否有错误，确定
事务是提交还是撤销

如果有错，则回滚操作，事务结束

如果成功，则提交操作，事务结束

■ 演示：转账1000，转账失败的情况

结果

查看转账事务前的余额

customerName	currentMoney
张三	1000.00
李四	1.00

消息 547，级别 16，状态 0，第 7 行
UPDATE 语句与 CHECK 约束"CK_currentMoney"冲突。
该冲突发生于数据库"stuDB"，表"dbo.bank"，column 'currentMoney'。
语句已终止。

查看转账事务过程中的余额

customerName	currentMoney
张三	1000.00
李四	1001.00

交易失败，回滚事务
查看转账事务后的余额

customerName	currentMoney
张三	1000.00
李四	1.00

转账事务前

转账事务过程中

转账事务结束后

■ 演示：转账800，转账成功的情况

结果		转账事务前
查看转账事务前的余额		
customerName	currentMoney	

张三	1000.00	
李四	1.00	
查看转账事务过程中的余额		转账事务过程中
customerName	currentMoney	

张三	200.00	
李四	801.00	
交易成功，提交事务，写入硬盘，永久的保存		
查看转账事务后的余额		转账事务结束后
customerName	currentMoney	

张三	200.00	
李四	801.00	




10.2 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。

■ 故障的影响

- 运行的事务非正常中断
- 破坏数据库

- 
- DBMS提供恢复子系统
 - 数据库恢复：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）
 - 恢复技术是衡量数据库管理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 数据库镜像



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

- 什么是事务[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

- 分类：

- 可预期的，可以通过事务程序本身发现
 - 非预期的

可预期的例子:

- 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

BALANCE=BALANCE-AMOUNT; ---(BALANCE账户甲余额,
AMOUNT 为转账金额)

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

{ --乙的余额BALANCE1;

BALANCE1=BALANCE1+AMOUNT;

COMMIT; }

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。



非预期的例子

Begin transaction


fac=1;

for i=1 to n do

fac=fac*i;

.....

commit

- 
- 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。
 - 输入数据有误
 - 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
 - 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)
- 在不影响其他事务运行的情况下，强行回滚 (ROLLBACK) 该事务
- 撤销该事务已经做出的任何对数据库的修改，使得这个事务象根本没有启动过一样



二、系统故障

- 称为**软故障**，是指造成系统停止运转的任何事件，使得系统重新启动。
- 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
- 外部存储设备上的数据未受影响 （**不破坏数据库**）



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

- 称为**硬故障**，指外存故障，破坏性最大
- 故障常见原因
 - 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



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

- 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



计算机病毒故障的恢复

- **装入**发生故障前某个时刻的数据**副本**。



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：介质故障，计算机病毒
- 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：事物内部故障，系统故障



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)
- 登录日志文件 (logging)

2. **如何利用这些冗余数据实施数据库恢复**



10.4.1 数据转储

一、什么是数据转储

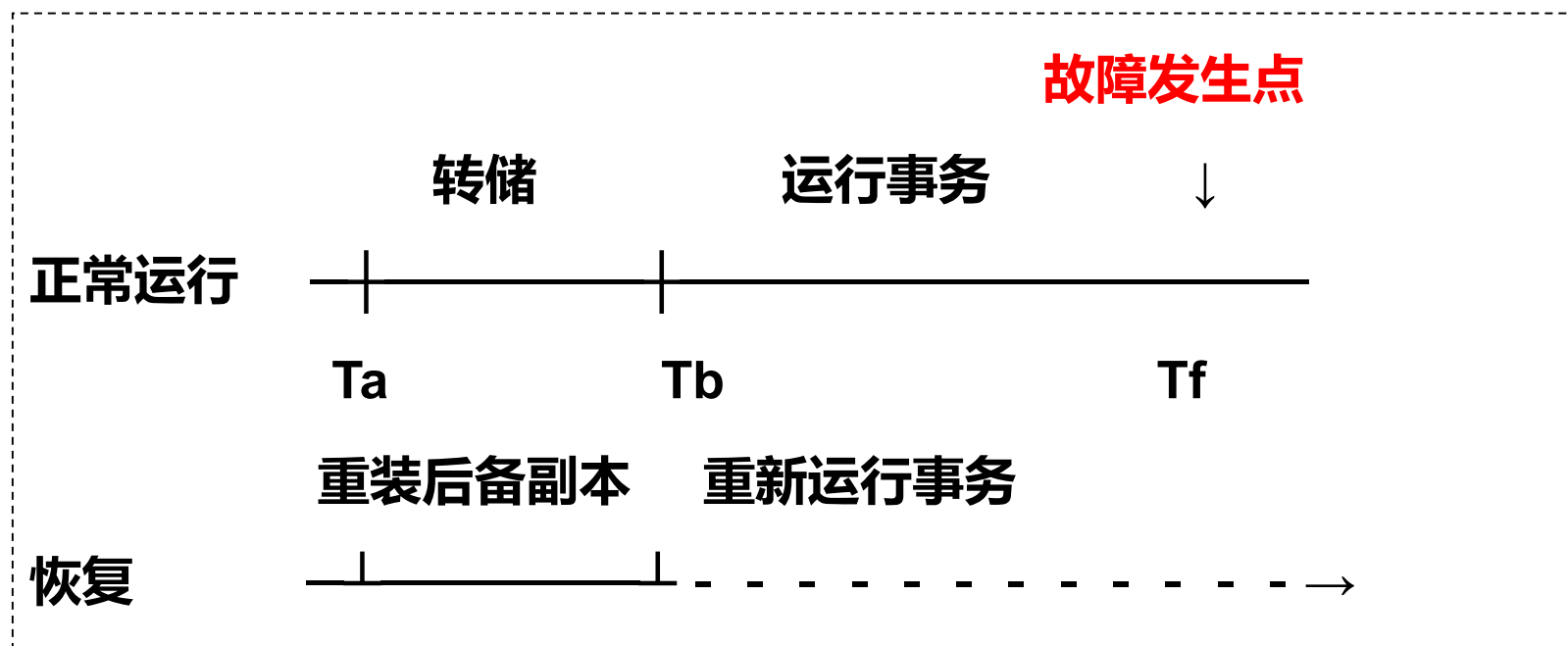
二、转储方法



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**

转储





二、转储方法

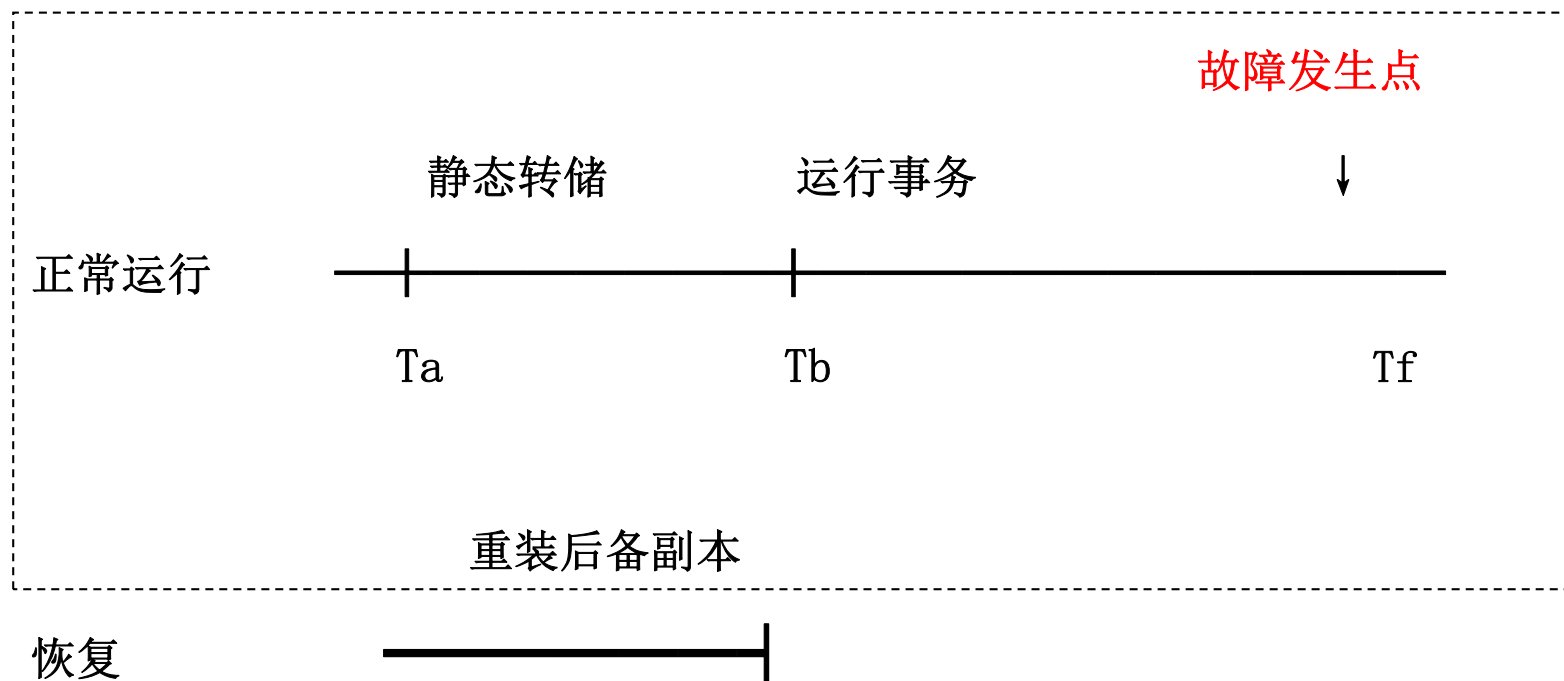
1. 静态转储与动态转储
2. 海量转储与增量转储
3. 转储方法小结



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

利用静态转储副本进行恢复





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

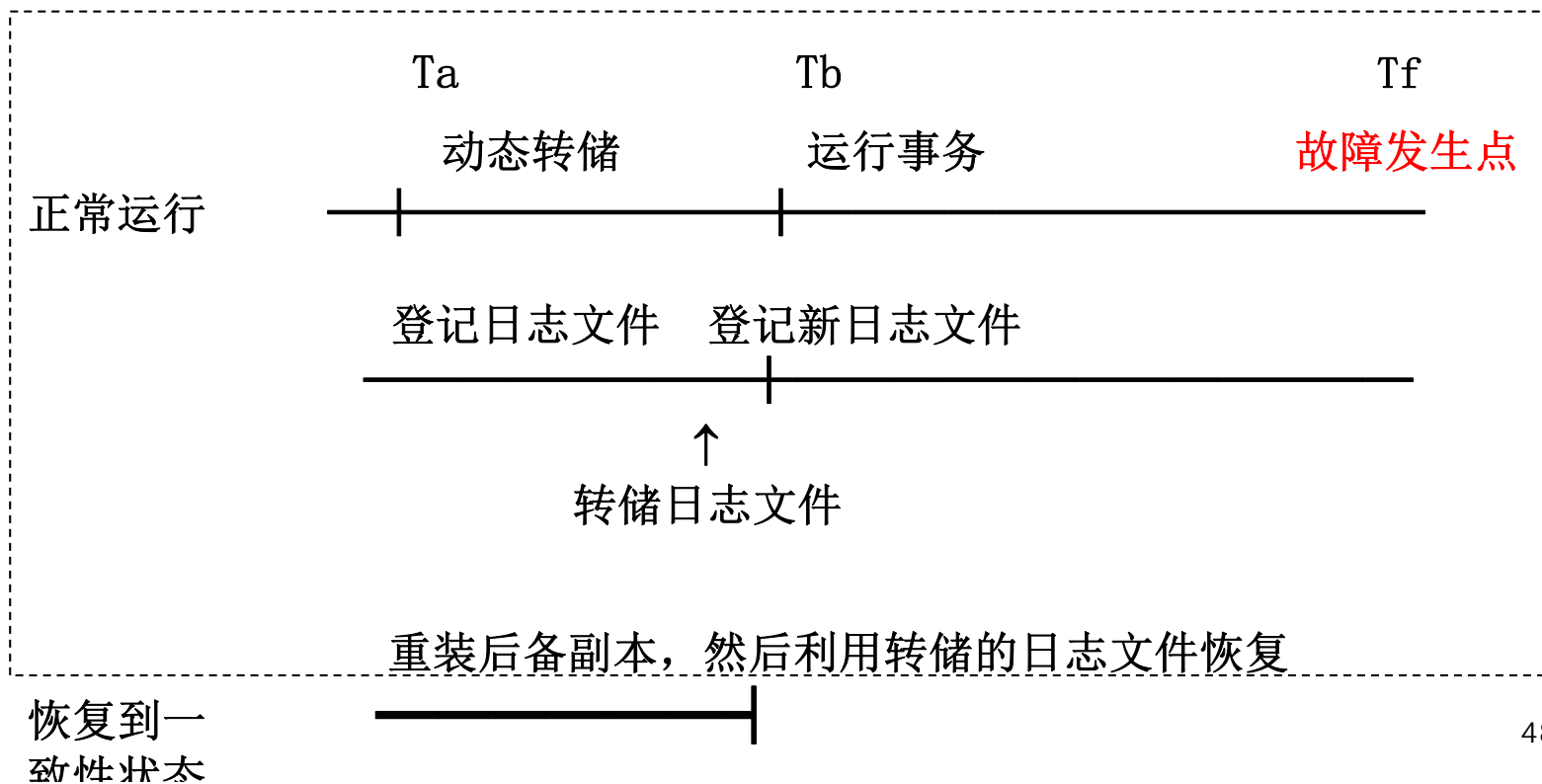
[例]在转储期间的某个时刻 T_c ，系统把数据 $A=100$ 转储到磁带上，而在下一时刻 T_d ，某一事务将 A 改为 200。转储结束后，后备副本上的 A 已是过时的数据了



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
 - 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





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
		动态海量转储	静态海量转储
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



■ 转储策略

- 应经常进行数据转储，制作后备副本。
- 但转储又是十分耗费时间和资源的，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：

- 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



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 (Log File)

- 用来记录事务对数据库的**更新操作**的文件

■ 日志文件中需要记录的内容

- 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

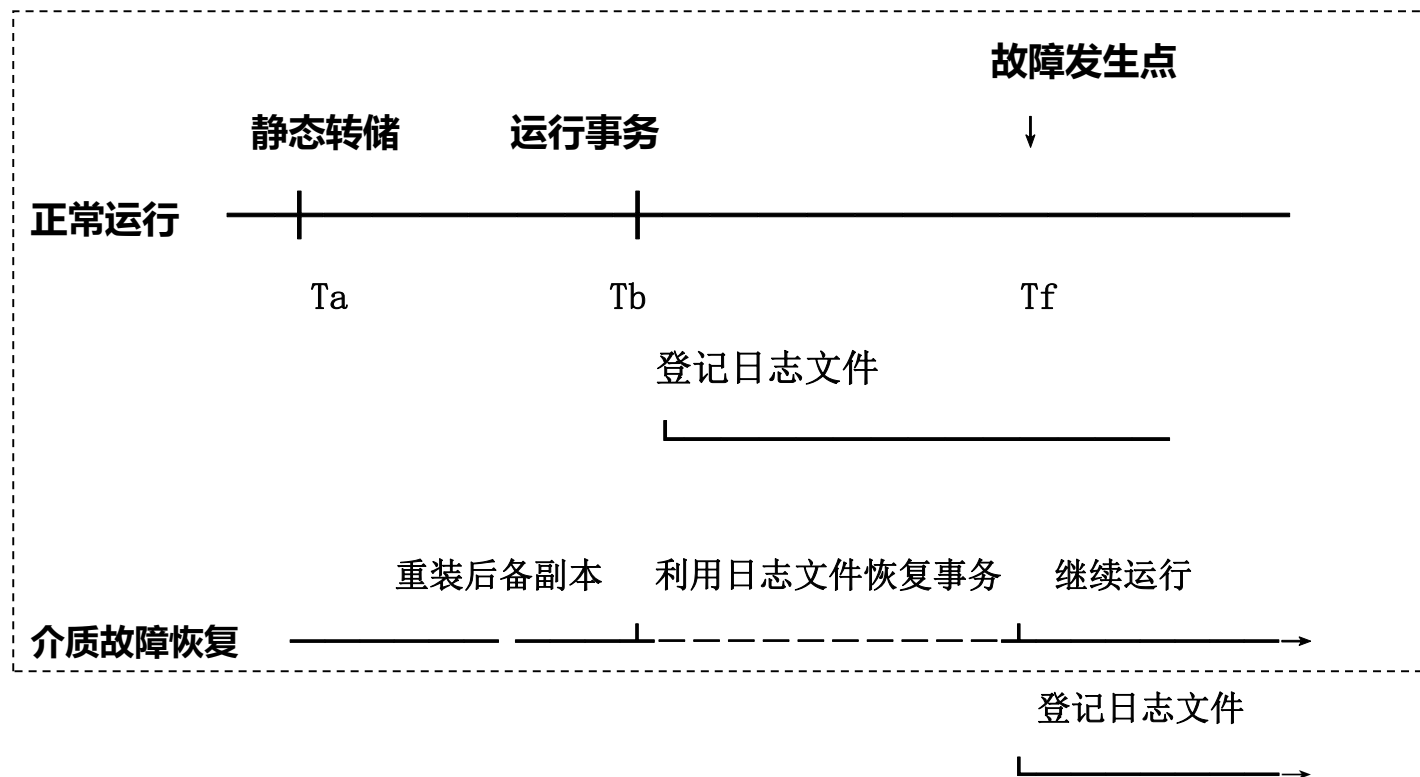
- ◆ 操作类型 (插入、删除或修改)
- ◆ 操作对象 (记录内部标识)
- ◆ 更新前数据的旧值
- ◆ 更新后数据的新值



二、日志文件的作用

- 事务故障恢复和系统故障恢复，**必须用日志文件**
- **动态转储**后的后备副本进行介质故障恢复，**必须用日志文件**
- 静态转储后的后备副本进行介质故障恢复，也可用日志文件

利用静态转储副本和日志文件进行介质故障恢复





上图中：

- 系统在 T_a 时刻停止运行事务，进行数据库转储
- 在 T_b 时刻转储完毕，得到 T_b 时刻的数据库一致性副本
- 系统运行到 T_f 时刻发生故障
- 为恢复数据库，首先由DBA重装数据库后备副本，将数据库恢复至 T_b 时刻的状态
- 重新运行自 $T_b \sim T_f$ 时刻的所有更新事务，把数据库恢复到故障发生前的一致状态



三、登记日志文件

- 为保证数据库是可恢复的，登记日志文件必须遵循的基本原则：
 - 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

- 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



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
 - 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



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，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。



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没来得及写入数据库
- 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
- 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）

- Redo队列: 在故障发生前已经提交的事务T1, T3, T8.....
- Undo队列: 故障发生时尚未完成的事务 T2, T4, T5, T6, T7, T9



系统故障的恢复步骤

2. 对Undo队列事务进行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 T9, T7, T6, T5, T4, T2
3. 对Redo队列事务进行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 T1, T3, T8.....



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

■ 恢复方法

- 利用数据库后备副本和日志文件进行恢复

■ 需要DBA介入，具体的恢复操作仍由DBMS完成。


□ 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



■ 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。



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。



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 数据库镜像



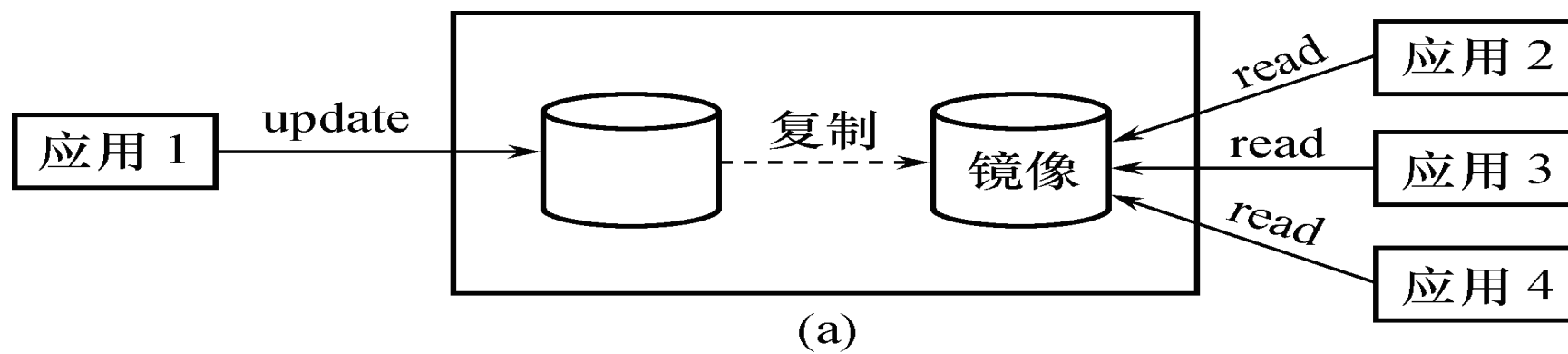
10.6 数据库镜像

- 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）



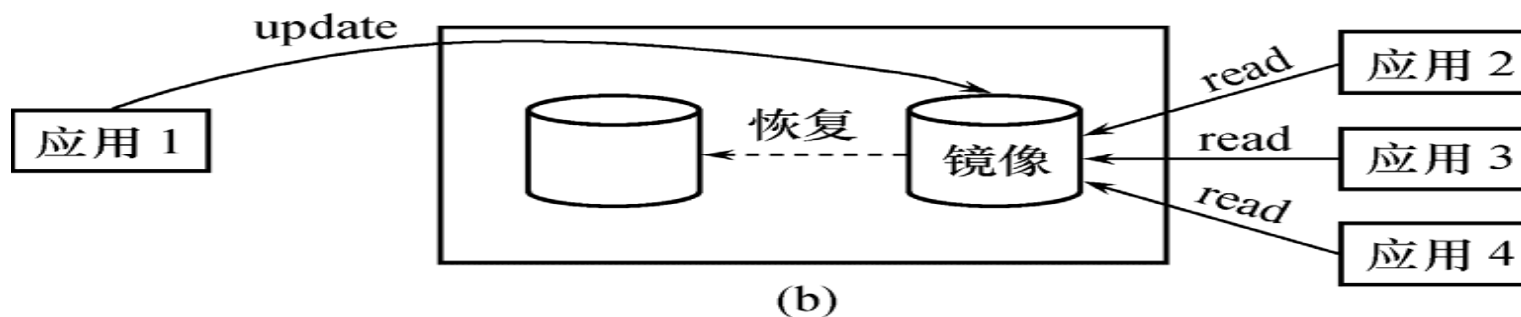
■ 数据库镜像

- 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（如下图所示）



数据库镜像的用途


- 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(如下图所示)





■ 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

- 
- 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



本章小结

- 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

- 
- 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



■ 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