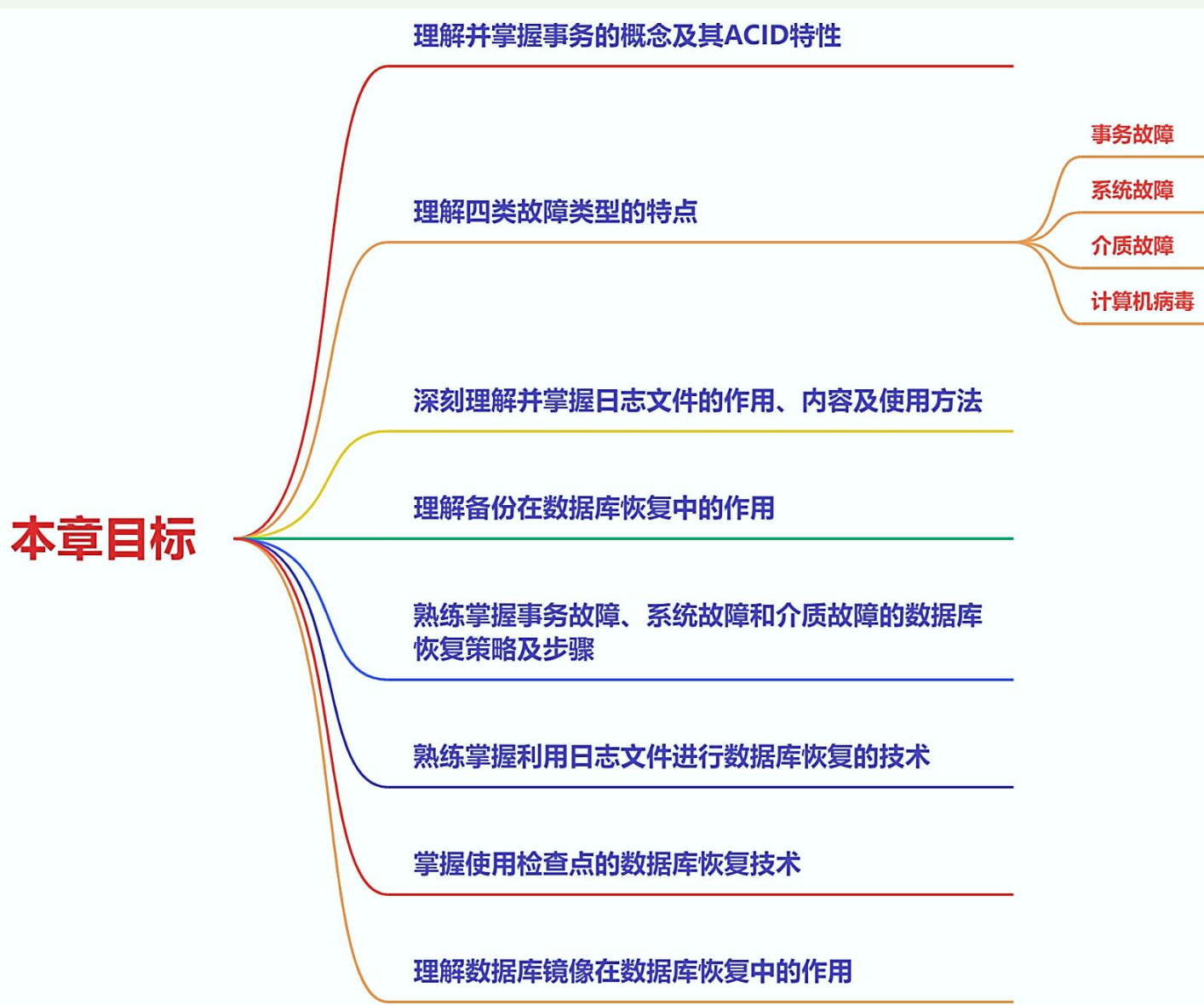


第11章 数据库恢复技术



- 事务的基本概念
- 数据库恢复概述
- 故障的种类
- 恢复的实现技术
- 恢复策略
- 具有检查点的恢复技术
- 数据库镜像
- 本章小结



- **事务处理**也称**联机事务处理**(Online transaction processing, **OLTP**)
 - 事务处理技术主要包括**数据库恢复技术**和**并发控制技术**
 - **事务是恢复与并发控制的基本单位**
- **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，是一个**不可分割的工作单位**
- **事务与程序**
 - **事务是数据库应用程序的基本逻辑单元**
 - 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



▪ 定义事务的两种方式:

– 1.显示定义

- 事务的开始和结束由用户定义
- 事务定义的主要语句: **BEGIN TRANSACTION; COMMIT; 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

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
- 事务运行过程中发生了故障, 不能继续下去
- 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

– 2.隐式定义

- 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



- 启动事务
 - START TRANSACTION
 - BEGIN
- 设置事务
 - SET TRANSACTION
- 提交事务
 - COMMIT
 - END
- 回滚事务
 - ROLLBACK

openGauss事务的具体用法参见:

<https://www.opengauss.org/zh/docs/3.1.0/docs/Developerguide/%E4%BA%8B%E5%8A%A1%E6%8E%A7%E5%88%B6.html>

墨天轮对openGauss高可靠事务的介绍:

<https://www.modb.pro/db/30010>



■ 事务的ACID特性

原子性Atomicity	一致性Consistency	隔离性Isolation	持久性Durability
<ul style="list-style-type: none">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	<ul style="list-style-type: none">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：只包含成功事务提交的结果不一致性状态：不正确的状态（因故障造成）确保单个事务的一致性_{是编写该事务的应用程序员的责任}完整性约束的自动检查是实现一致性的一种方法	<ul style="list-style-type: none">一个事务的执行不能被其他事务干扰即一个事务的内部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	<ul style="list-style-type: none">指一个事务一旦提交，它对数据库中数据的改变就是永久性的，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
<ul style="list-style-type: none">保证事务ACID特性是事务管理的重要任务事务ACID特性可能遭到破坏的因素有：<ul style="list-style-type: none">➢ 多个事务并行运行时，不同事务的操作交叉执行；事务中运行过程中被强行终止			



■ 事务原子性示例：

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

银行转账场景，业务逻辑要求：

- 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，用户逻辑上就会发生错误，少了1万元，数据库就处于不一致性状态

■ 事务隔离性示例：

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

- T₁和T₂是两个事务
- 这两个事务同时读取到A的值为16
- T₁更新A的值为15
- T₂更新A的值为13
- 因为没有隔离机制，这导致T₁的修改被T₂的结果所覆盖！
即T₁的更新操作没有被体现



- 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
- 数据库的恢复
 - **DBMS**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**DBMS**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

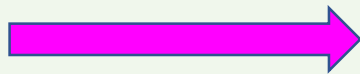


- 常见故障分类：
 - 事务内部的故障
 - 系统故障
 - 介质故障



- 事务故障意味着事务没有达到预期的终点(**COMMIT**或显式的**ROLLBACK**), 因此数据库可能处于不正确的状态。

- 事务故障的发现

- 有些可通过事务程序本身发现 
- 更多的是非预期的, 不能由应用程序处理
 - 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而被终止

```
BEGIN TRANSACTION
读账户甲的余额BALANCE;
BALANCE=BALANCE-AMOUNT;
IF(BALANCE < 0 ) THEN
{打印 '金额不足, 不能转账' ;
ROLLBACK;
}
ELSE
{读账户乙的余额BALANCE1;
BALANCE1=BALANCE1+AMOUNT;
写回BALANCE1;
COMMIT;
}
```

- 事务故障的恢复操作

- 事务撤销(**UNDO**)



- **系统故障**是指造成系统停止运转的任何事件，使得系统要**重新启动**。也称为**软故障**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
- **系统故障特点：**
 - 所有运行的事务都非正常终止，但不破坏数据库
 - 内存中数据库缓冲区的信息全部丢失
 - 部分尚未完成的事务的结果可能已送入物理数据库，从而造成数据库可能处于不正确的状态
- **系统故障的恢复操作**
 - 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**
 - 系统重新启动时，恢复程序**重做(RED0)**所有**已完成的事务**



- 介质故障也称为硬故障(Hard crash)，指外存故障。
 - 磁盘损坏
 - 磁头碰撞
 - 瞬时强磁场干扰
- 介质故障特点：
 - 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
 - 与事务故障和系统故障相比，发生的可能性小，但破坏性最大
- 需要借助存储在其他地方的数据备份来恢复数据库



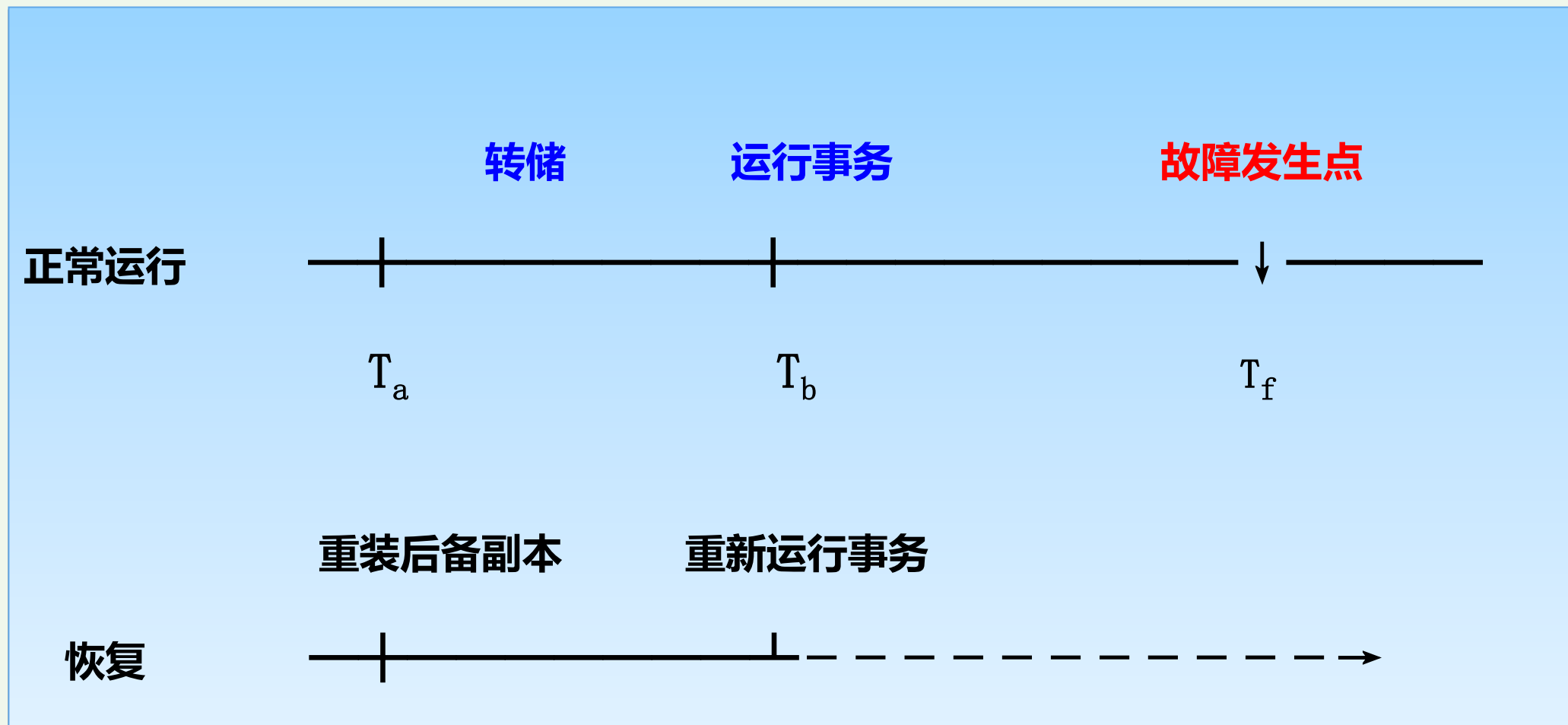
- 各类故障对数据库的影响有两种可能性:
 1. 数据库本身被破坏
 2. 数据库没有被破坏, 但数据可能不正确, 这是由于事务的运行被非正常终止造成的
- 恢复的基本原理: **冗余**
 - 即, 可以利用存储在系统别处的**冗余数据**来**重建**数据库中已被破坏或不正确的那部分数据
- 恢复的基本原理简单, 但**实现技术的细节**却相当**复杂**
 - 一个大型数据库产品, 恢复子系统的代码要占全部代码的**10%**以上



- 恢复机制涉及的关键问题
 - 如何建立冗余数据
 - 如何利用这些冗余数据实施数据库恢复
- 建立冗余数据最常用的技术
 - 数据转储(**dump**)
 - 登记日志文件(**logging**)
- 通常在一个数据库系统中，这两种方法一起使用



- **数据转储**是指**DBA**定期地将整个数据库复制到磁带、磁盘或其他存储介质上保存起来的过程
 - 备用的数据称为**后备副本**或**后援副本(backup)**
 - 数据库恢复采用的基本技术
- **利用后备副本进行数据库的恢复**
 - 当数据库遭到破坏后可以将后备副本重新装入,
 - 但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



转储和恢复示意图

- 转储是十分耗费时间和资源的，不能频繁进行。DBA应根据数据库使用情况确定一个适当的转储周期
- 转储可分为静态转储和动态转储

静态转储	动态转储
<ul style="list-style-type: none">• 是在系统中无运行事务时进行的转储操作，即转储操作开始的时刻数据库处于一致性状态，在转储期间不允许(或不存在)对数据库的任何存取、修改。• 静态转储得到的一定是一个数据一致性的副本	<ul style="list-style-type: none">• 是指转储期间允许对数据库进行存取或修改，即转储和用户事务可以并发执行
<p>优点：</p> <ul style="list-style-type: none">• 实现简单	<p>优点：</p> <ul style="list-style-type: none">• 克服了静态转储的缺点，不用等待正在运行的用户事务结束，也不会影响新事务的运行
<p>缺点：</p> <ul style="list-style-type: none">• 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	<p>缺点：</p> <ul style="list-style-type: none">• 不能保证转储结束后后援副本的数据正确有效

■ 动态转储缺点示例

- 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

■ 动态转储缺点的解决方案

- 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)
- 后援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。

■ 转储也可分为海量转储和增量转储

- 海量转储是指每次转储全部数据库
- 增量转储是指每次只转储上一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
- 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



■ 转储方法分类

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

- openGauss的备份与恢复可参见官网：

<https://www.opengauss.org/zh/docs/3.1.0/docs/Administratorguide/%E5%A4%87%E4%BB%BD%E4%B8%8E%E6%81%A2%E5%A4%8D.html>

- 墨天轮：

<https://www.modb.pro/doc/46420>



- **Oracle**的逻辑备份是用使用**Oracle**提供的操作系统工具**Export**、**Import**将数据库中的数据导出、导入
 - **Export**、**Import**都是在操作系统而不是**SQL*PLUS**环境下使用
 - 在每一个**Oracle**数据库中，可以使用**Export**命令将数据库中的数据备份成一个二进制的操作系统文件，文件格式为**DMP(Export Dump File)**，称为**输出转储文件**
 - 导出的文件可以使用另一个操作系统命令**Import**重新导入到另一个数据库中
- 物理备份是操作系统文件的备份，即，即使某个数据文件、没有数据也必须备份。逻辑备份是数据的备份，**不拷贝物理文件**，可以只将数据文件中的某一个表导出，节省空间。逻辑备份中导出数据时没有操作系统信息，所以可以在不同的平台之间传输。

```
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\apple>exp hr/hr@XE file=d:\hr_2019_04_05.dmp tables=(locations, jobs)

Export: Release 11.2.0.2.0 - Production on 星期五 4月 5 11:37:25 2019

Copyright (c) 1982, 2009, Oracle and/or its affiliates. All rights reserved.

连接到: Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
已导出 ZHS16GBK 字符集和 AL16UTF16 NCHAR 字符集
服务器使用 AL32UTF8 字符集 (可能的字符集转换)

即将导出指定的表通过常规路径...
. . 正在导出表                LOCATIONS导出了                23 行
EXP-00091: 正在导出有问题的统计信息。
EXP-00091: 正在导出有问题的统计信息。
EXP-00091: 正在导出有问题的统计信息。
EXP-00091: 正在导出有问题的统计信息。
EXP-00091: 正在导出有问题的统计信息。
. . 正在导出表                JOBS导出了                19 行
EXP-00091: 正在导出有问题的统计信息。
EXP-00091: 正在导出有问题的统计信息。
导出成功终止, 但出现警告。
```

参考资料: <https://blog.csdn.net/jiushancunmonkeyking/article/details/78851461>



- 什么是日志文件?
 - 日志文件(log file)是用来记录事务对数据库的更新操作的文件
- 本小节主要内容:
 - 日志文件的格式和内容
 - 日志文件的作用
 - 登记日志文件遵循的两条原则



■ 以记录为单位的日志文件

以记录为单位的每一条日志记录	
格式	内容
<ul style="list-style-type: none">• 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	<ul style="list-style-type: none">• 各个事务的所有更新操作• 事务标识 (标明是哪个事务)• 操作的类型 (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)

■ 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
- 被更新的数据块

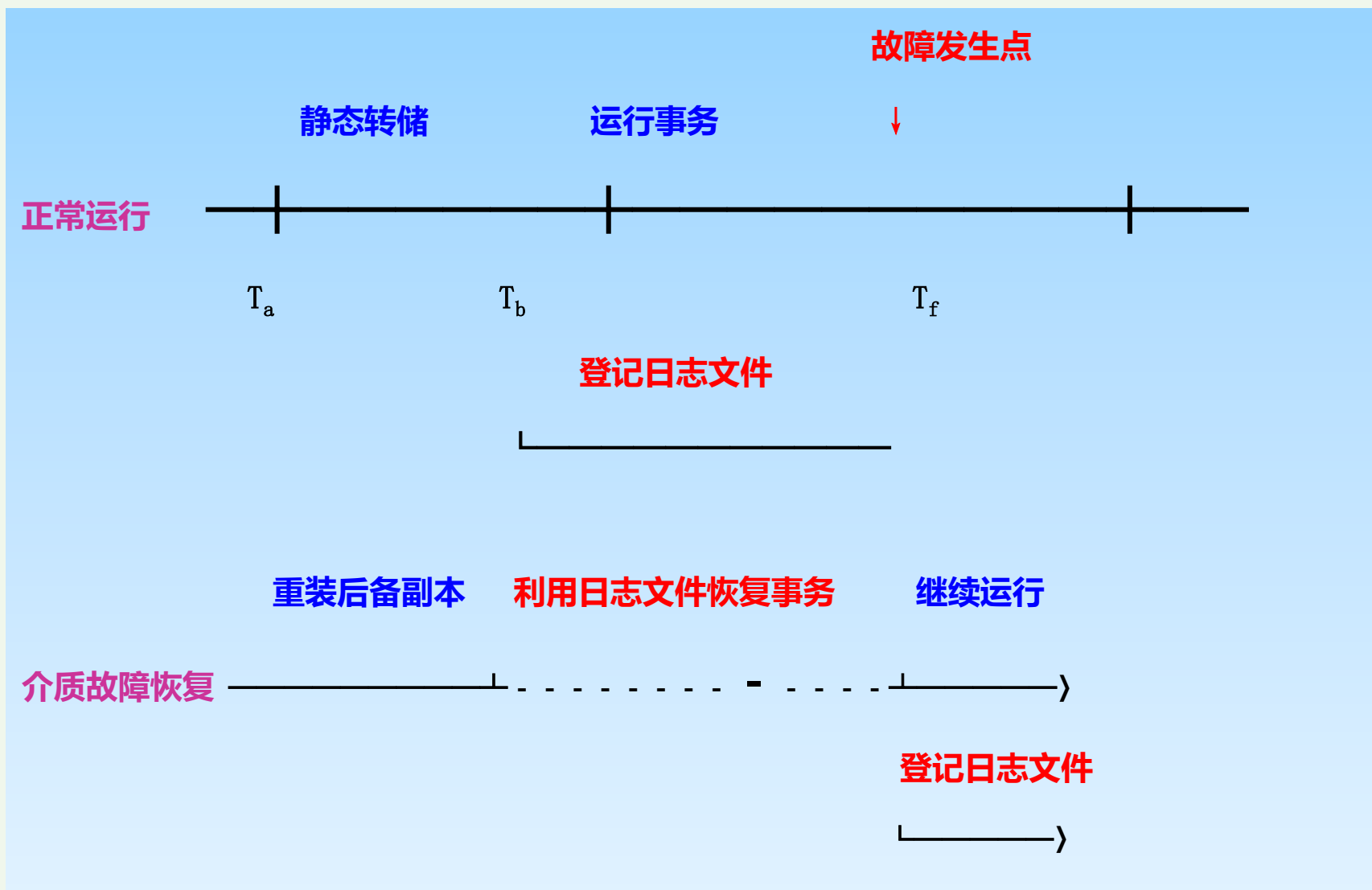
■ 日志文件的作用

- 用来进行事务故障恢复和系统故障恢复
- 协助后备副本进行介质故障恢复

■ 日志文件的具体作用

- 事务故障恢复和系统故障恢复必须用日志文件
- 在动态转储方式中必须建立日志文件，后备副本和日志文件结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
 - 当数据库毁坏后可重新装入后援副本把数据库恢复到转储结束时刻的正确状态
 - 利用日志文件，把已完成的事务进行重做处理
 - 对故障发生时尚未完成的事务进行撤销处理
 - 不必重新运行那些已完成的事务程序就可把数据库恢复到故障前某一时刻的正确状态





利用日志文件恢复示意图

- 为保证数据库是可恢复的，**登记日志文件必须遵循两条原则**：
 - **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再写数据库？**
 - 把对数据的修改写到数据库中和把表示这个修改的日志记录写到日志文件中是两个不同的操作。有可能在这两个操作之间发生故障，即这两个写操作只完成了一个。如果先写了数据库修改，而在运行记录中没有登记这个修改，则以后就无法恢复这个修改。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多执行一次不必要的**UNDO**操作，并不会影响数据库的正确性



- 若系统运行过程中发生故障，利用**数据库后备副本+日志文件**就可以将数据库恢复到故障前的某个一致性状态
- 三类不同故障的恢复策略：
 - **事务故障的恢复**
 - **系统故障的恢复**
 - **介质故障的恢复**

- **事务故障**是指事务在运行至正常终止点前被终止
- **恢复方法：UNDO**
 - 由恢复子系统利用日志文件**撤销(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**，对用户是隐蔽
- **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
 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



- 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可能已写入数据库
 - 已提交事务对数据库的更新可能还留在缓冲区没来得及写入数据库
- 恢复方法: **UNDO+REDO**
 - 撤消(**UNDO**)故障发生时未完成的事务;
 - 重做(**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, 对用户透明
- 系统故障的恢复步骤:
 1. 正向扫描日志文件(即从头向后扫描日志文件), 找出故障发生前已提交的事务(特征: 既有**BEGIN TRANSACTION**记录, 也有**COMMIT**记录), 将其事务标识记入重做队列(**REDO-LIST**)。同时找出故障发生时尚未完成的事务(特征: 只有**BEGIN TRANSACTION**记录, 无相应的**COMMIT**记录), 将其事务标识记入撤销队列(**UNDO-LIST**)
 2. 对撤销队列中的各个事务进行撤销(**UNDO**)处理
 - 反向扫描日志文件, 对每个撤销事务的更新操作执行逆操作, 即将日志文件中“更新后的值”写入数据库
 3. 对重做队列中的各个事务进行重做处理
 - 正向扫描日志文件, 对每个重做事务重新执行日志文件登记的操作, 即将日志文件中“更新后的值”写入数据库



- 发生介质故障后，磁盘上的物理数据和日志文件被破坏，这是最严重的一种故障
- 恢复方法：
 - 重装数据库，然后重做已完成的事务
- 介质故障的恢复需要DBA介入
 - DBA只需要重装最近转储的数据库副本和有关的各日志文件副本，然后执行系统提供恢复命令即可，具体的恢复操作仍由DBMS完成。
- 介质故障的恢复步骤：
 1. 装入最新的数据库后备副本(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. 装入相应的日志文件副本(转储结束时刻的日志文件副本)，重做已完成的事务
 - 即首先扫描日志文件，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



- 利用日志技术进行数据库恢复时，恢复子系统必须搜索日志，确定哪些事务需要重做，哪些事务需要撤销
- 两个问题：
 - 搜索整个日志将耗费大量的时间
 - 重做处理：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
 - 恢复子系统在登录日志文件期间动态地维护日志



■ 1.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

■ 2.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

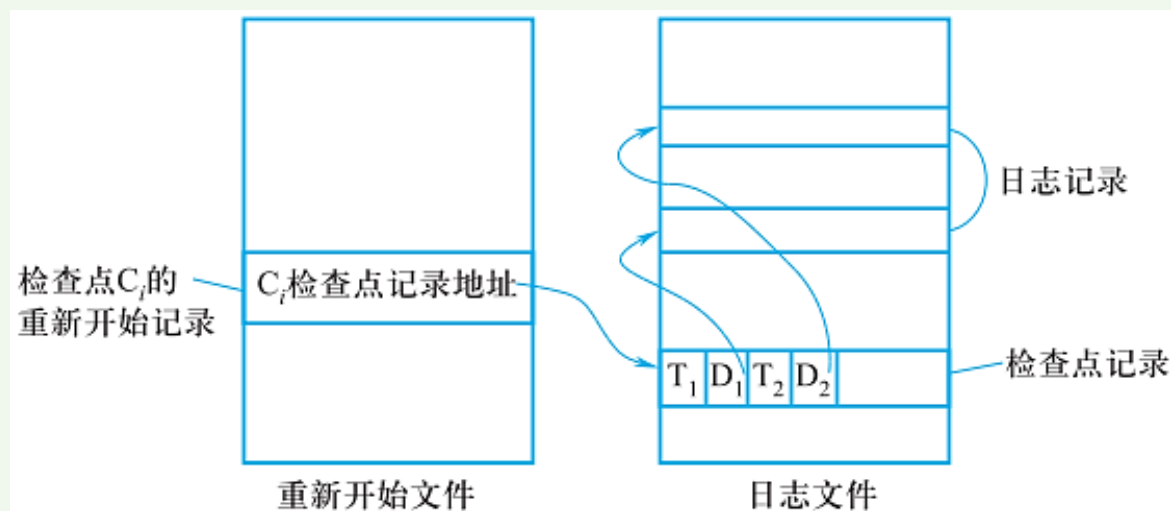


图11.3 具有检查点的日志文件和重新开始文件

■ 3.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
- 具体步骤如下：
 - 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
- 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



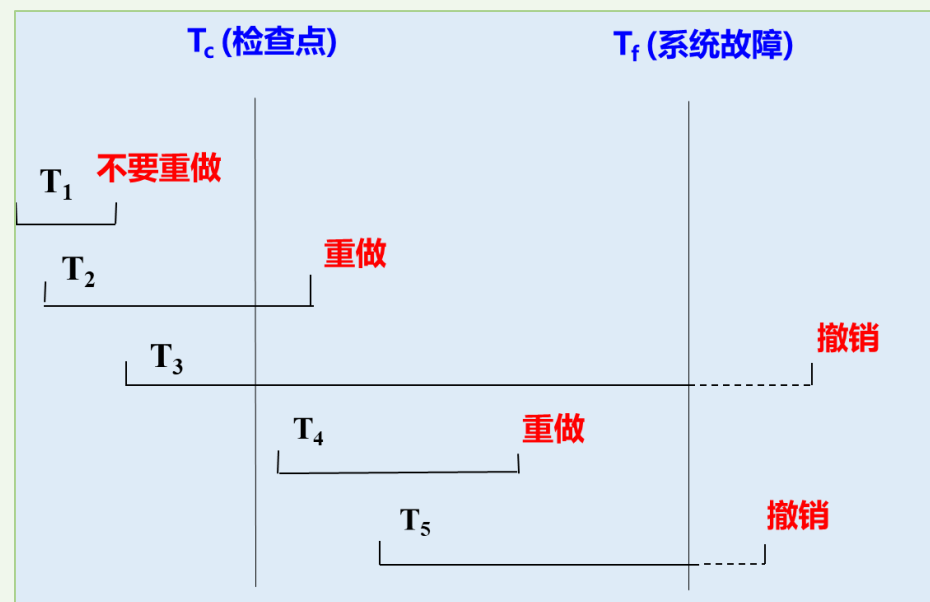
■ 4.恢复子系统的恢复策略

– 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

■ 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执行重做操作



恢复子系统将根据事务的不同状态采取不同的恢复策略

■ 恢复子系统使用检查点方法进行恢复的步骤：

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
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**
 - **UNDO-LIST**：需要执行**UNDO**操作的事务集合
 - **REDO-LIST**：需要执行**REDO**操作的事务集合
3. 从检查点开始正向扫描日志文件：
 - 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
 - 继续以上过程，直到日志文件结束
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



- 官网:

<https://www.opengauss.org/zh/docs/3.1.0/docs/Developerguide/%E6%A3%80%E6%9F%A5%E7%82%B9.html>

- 墨天轮:

<https://www.modb.pro/db/30794>

<https://www.modb.pro/db/214502>

- 考虑如下的日志记录，假设开始时A,B,C的值都是9：
如果系统故障发生在13之后，系统该如何恢复？请
写出系统恢复后A, B, C的值。

序号	日志
1	T1:开始
2	T1:写C, C=8
3	T1:提交
4	检查点
5	T4:开始
6	T4:写B, B=5
7	T4:写A, A=2
8	T4:提交
9	T2:开始
10	T2:写B, B=-3
11	T3:开始
12	T3:写A, A=1
13	T2:写C, C=7

T1:不操作; T2和T3都是撤销; T4: 重做

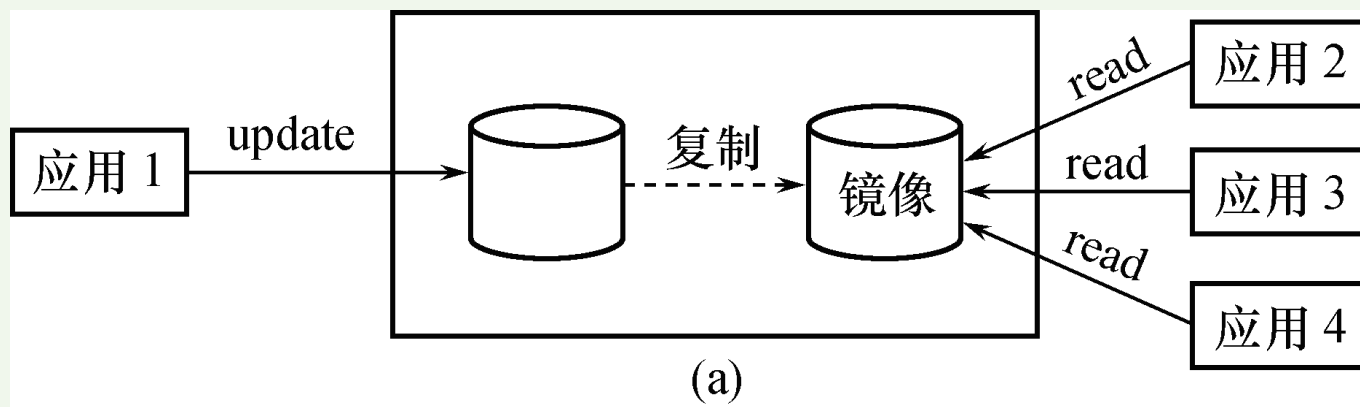


- 考虑如图所示的日志记录，假设开始时B,C,D的值都为0，如果系统故障发生在9之后，系统如何进行恢复？写出系统恢复后B, C, D的值。

序号	日志
1	T1:开始
2	T1:写D, D=2
3	T1:提交
4	检查点
5	T2:开始
6	T2:写B,B=1
7	T4:开始
8	T4:写D,D=5
9	T3:开始
10	T3:写C,C=-6
11	T4:提交
12	T2:写D,D=7

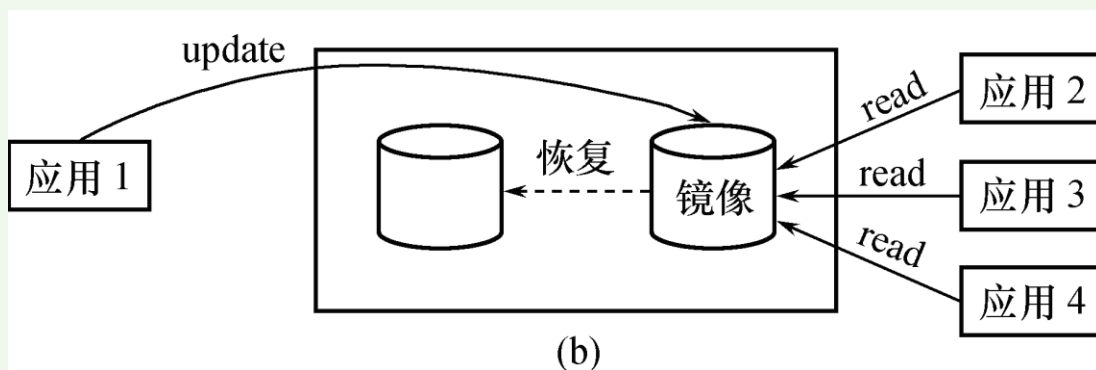


- 介质故障是对系统影响最为严重的一种故障，严重影响数据库的可用性
 - 介质故障恢复比较费时
 - 为预防介质故障，数据库管理员必须周期性地转储数据库
- 数据库镜像(**Mirror**)是解决介质故障、提高数据库可用性的一种常用方法
 - 即根据**DBA**的要求，自动把整个数据库或其中的关键数据复制到另一个磁盘上。每当主数据库更新时，**DBMS**自动把更新后的数据复制过去，由**DBMS**自动保证镜像数据与主数据库的一致性



■ 镜像数据库的使用:

- 当出现介质故障时, 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



- 当没有故障时, 数据库镜像还可以用于并发操作, 即当一个用户对数据加排他锁修改数据时, 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
- 由于数据库镜像是通过**复制数据**实现, 频繁地复制数据自然会降低系统运行效率, 因此在实际应用中只对**关键数据**和**日志文件**进行镜像, 而不是整个数据库

- 官网

<https://www.opengauss.org/zh/docs/3.1.0/docs/CharacteristicDescription/%E4%B8%BB%E5%A4%87%E6%9C%BA.html>

- 墨天轮

<https://www.modb.pro/db/30014>

<https://www.modb.pro/db/31066>

- 事务的概念和性质
 - 事务是数据库的逻辑工作单位
 - **DBMS**保证系统中一切事务的**ACID**特性，就保证了事务处于一致状态
 - 事务既是数据库恢复的基本单位，也是并发控制的基本单位
- 故障的种类
 - 事务故障
 - 系统故障
 - 介质故障
- 系统恢复最经常使用的技术
 - 数据库转储+日志文件
- 系统恢复的基本原理
 - 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



- 教材第十一章全部习题.
- 要求：作业布置后一周内完成并提交到课程网站

