第10章 数据库恢复技术

备份与恢复技术?

文件管理系统

Ghost、恢复出厂设置、副本文件、操作系统补丁更新时的备份、备份软件（例如：云备份）

以文件为单位，涉及版本、快照、副本。。。

数据库管理系统

备份+日志、镜像、备份软件

细化到数据结构（项）、备份涉及数据变化的**过程**，数据的取值对应任务的状态

应用背景：

DBMS——提供数据共享和管理服务的系统软件，数据动态更新、面向并发访问的用户

故障的系统、科学分析，数据的问题（内存/外存、干净/脏、。。。）

DBMS的备份与恢复

➢正确性的标准？（不*“丢失”*数据？）

- 面临的问题
  1）正确;
  2）智能（自我保护的程序机制）;
  3）故障的种类多（应用程序出错、网络断了、操作系统蓝屏、宕机、硬盘坏了。。。）， 响应机制不同;
  4）性能（备份的开销、恢复的开销、服务的性能） 。

- 学习内容
  基本原理、内含的协议、优化机制
- 问题复杂的原因
  1）DBMS内部程序结构及程序执行的复杂性（多任务）;
  2）应用逻辑的语义需求。

事务

10.1 什么是数据库恢复

将因破坏或故障而导致的数据库数据的错误状态恢复到最近一个正确状态的技术。

10. 2 目标

1、保持事务原子性（Atomicity）；

2、保持事务持久性（Durability）。

事务——transaction——交易、买卖

现代计算机的运行原理及硬件特点使得事务处理任务艰巨，DBMS提供数据共享服务的需求使得事务处理更加复杂，事务处理成为数据库的核心问题之一。标志性人物：Jim Gray

事务（transaction）

1、定义

构成一个独立逻辑工作单位的数据库操作集。

- 一条SQL语句；

例：将2号课程的成绩记录增加10分

- 一组SQL语句序列

例：（1）将2号课程的成绩记录增加10分；

（2）在应用日志表中增加描述上述操作的一条记录；

- 一个包含对数据库操作的应用程序

例：可能多条SQL语句，存在于不同的程序分支。

事务（transaction）

2、构成方式

① 显式

BEGIN TRANSACTION

…

COMMIT　或者ROLLBACK

其中：

COMMIT：提交，事务对DB修改写回到磁盘上的DB中去。

ROLLBACK：回滚，撤消对DB之修改，回滚到事务开始状态。

ABORT???——底层实现技术

② 隐式（可能是系统默认方式）

某种环境或者程序中的一条SQL语句、

应用程序或操作窗口退出

3、事务的ACID性质

1）原子性（Atomicity）

① 定义

事务是一个不可分割的工作单元，其对DB的操作要么都做，要么都不做。

② 目标

保证DB数据的正确性（例如：所有员工涨工资、转帐、售票的事务不能只做一部分动作）。

③ 技术

日志+ROLLBACK（UNDO）（意外终止）、影子数据;

并发控制（隔离保护）。

原子性需要依靠DBMS内部的自动保障机制。

2）一致性（Consistency）

① 定义

事务的执行必须是将数据库从一个正确（一致）状态转换到另一个正确（一致）状态。

例1：一个人的工龄不能大于年龄，则工龄的增长和年龄的增长必须一致的、配套的修改。

例2：所有员工工资的公积金应该依据政策同步调整。

例3：转帐问题，A有100万人民币是一个正确状态，减去50万，B帐上相应增加50万，数据库从一个正确状态转变另一个正确状态。

这两个操作，若只做其中一个，则不能实现数据库从一个正确状态转到另一个正确状态，破坏了事务一致性。

例4：将数据集合划分为三个子集，分三次读取三个子集的数据并进行小计和总计（一个事务内部的多次相关的读写操作，内容之间在全局逻辑上应该是相容的、一致的）。

2）一致性（consistency）

② 目标

保证DB数据正确性（防止丢失更新、读脏、读不可重复）。

③ 技术

并发控制、恢复机制

④ 实现

用户定义事务（保证相关操作在一个事务中）；

DBMS负责维护事务执行导致数据库状态变化过程中的一致性。

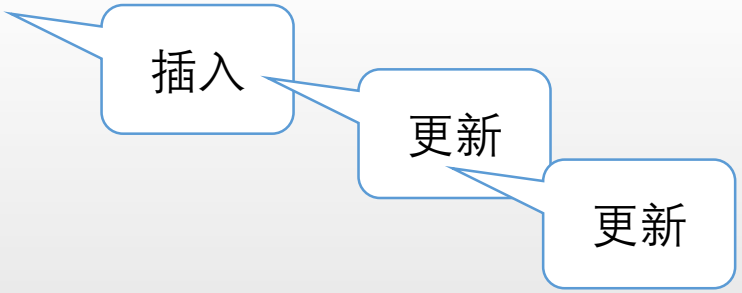
3）隔离性（isolation）

① 定义

一个事务中对数据库的操作及使用的数据与其它并发事务无关，并发执行的事务间不能互相干扰。

② 目标

避免链式干扰。



③ 技术

并发控制。

④ 实现

DBMS依据应用程序设定的事务隔离级别自动实现。

4）持久性（Durability）

① 定义

一个已提交事务对数据库的更新是永久性的，不受后来故障的影响。

② 目标

保证数据库可靠性

③ 技术

提交持久（内存是挥发装置，外存是抗挥发装置）。

（事务终止前应完成commit）

备份 + 日志。

外存!

④ 实现

持久性需要依靠DBMS的恢复子系统。

ACID特性带来的DBMS技术需求

恢复：

复杂系统如何保存数据（记录过程）、恢复策略（算法）、高可用性、其他技术手段

并发：

锁、协议、标准

10.3 数据库系统故障

1、事务故障

1）表现形式

①应用处理异常

可能产生自程序预留的异常情况的应对方案。

更多的故障来自于非预期的，是不能由应用程序处理的。

断网、应用程序进程僵死、应用程序进程被意外杀死、应用程序端电脑死机、断电

②系统异常

事务超时、死锁、活锁等

2）事务故障的特征

① 特定的事务没有到达预期的终点（COMMIT），事务夭折;

② 夭折事务对数据库的部分修改可能已写入数据文件。

（数据库可能因此处于不正确、不一致状态）

客户的理财账户余额已减少，但在写出理财产品的购买记录到数据文件前，事务发生了异常。

事务的ACID特性，原子性A，一致性C。

2、系统故障

1）表现形式

① 特定类型的硬件故障（CPU、内存、主板等**非外存储**设备）；

② 系统软件故障

DBMS：ORACLE、SQL SERVER、MYSQL、DB2、。。。

OS：UNIX、WINDOWS、LINUX、。。。

死机　蓝屏　意外重启　某系统功能意外退出　。。。

③系统操作失误：非正常关机/重启、强行终止系统进程、意外卸载相关系统运行环境。。。

④ 系统异常断电（重启之后系统未发现数据库的存储文件错误或者磁盘错误）

2）特征

① 内存数据丢失或不再可靠；

② 外存数据未受到破坏；

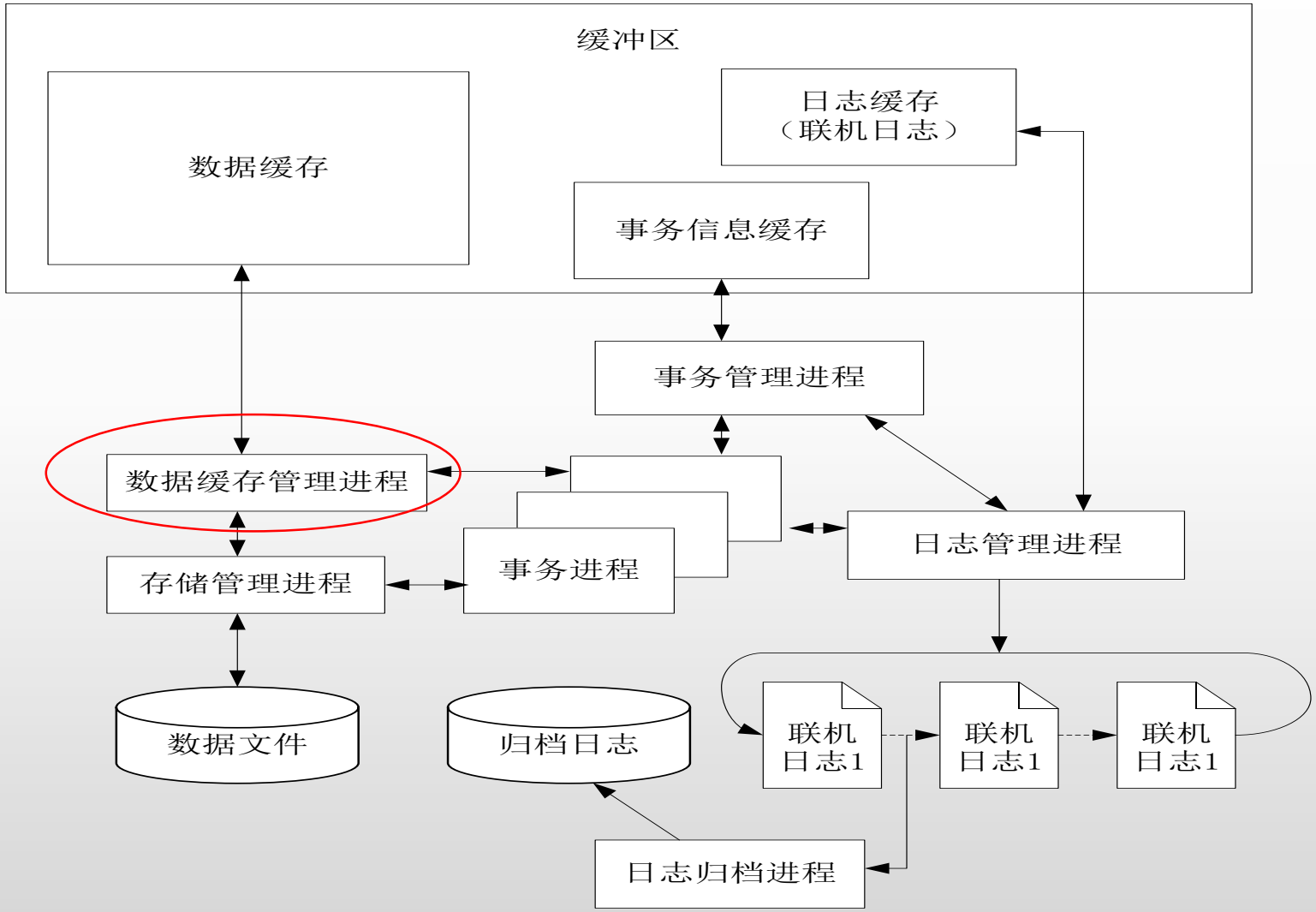
③ 一些尚未完成事务的更新结果可能已写入数据库存储介质；

④ 已完成事务的更新结果可能部分还未写入数据库（数据文件，也可能正处于提交过程之中）；

⑤ 已完成事务的结果可能全部未写入数据库（例如正在等待检查点）。

数据库的数据存储介质依然可靠，但是数据处于不正确或不一致状态

DBMS的缓存与核心进程



存储、缓存和读写

数据块：数据库系统中，数据库的存储单位一般是数据块（物理块），数据块可能包含多个数据项。

缓冲块：当数据库规模较大时，不可能所有的数据库内容驻留内存，一般选取事务操作需要访问的数据块驻留内存，主存中的数据块称为缓冲块（缓存页面）。

磁盘缓冲区：DBMS中会有用于集中临时存放缓冲块的内存区域，在后续讨论中，称为磁盘缓冲器，或简称缓存。

因为数据访问、缓存有限等原因，磁盘和缓存间有块移动，对应的两个操作：

input（B）：读取物理块B到缓存。

output（B）：将缓存中的缓冲块B覆盖写到磁盘上的物理块。

事务和数据库的交互：交互是通过read或write操作完成的。在执行这两个操作时，若X所在块B_X不在缓存中，会触发执行input（B_X）（不同DBMS可能会有差别，“写缓层”）。

缓冲块写到磁盘（output操作）的原因：

- 缓冲区管理器基于缓存使用策略调整资源（内存资源有限，“定期写”、“写缓层满了”）
- 数据库系统强制输出（force-output（），例如检查点、转储、关机、。。。。）

Write（X）和output（B_X）操作之间的关系：

二者不必紧密相邻，可能write操作后过一段时间才真正执行output操作。

问题出现：在write（X）和output（B_X）操作之间系统可能崩溃，此时的故障恢复需要分析针对X该采取何种动作。

各类故障对数据库的影响

- 故障发生时数据可能不正确（事务的运行被恶意干扰或非正常终止）。

  当涉及多个output操作的事务出现故障时，如果只有数据本身的当前值状态，是无法判断哪些值是完成了相应的output操作的。需要借助系统的容错机制，找出不正确的数据，恢复正确的数据。

- 不同的故障影响的范围不同，采取的恢复策略也不尽相同。

  自动、人工启动、借助外部资源

- 数据文件本身可能被破坏

  需要去寻找可用的数据，重建系统状态。

- 恢复的基本原理：冗余（包括对于数据变化过程的记录）

3、介质故障

1）分类

① 磁盘故障

磁盘损坏（磁道、扇区、分区、文件分配信息。。。），

磁盘读写装置损坏（磁头、电机。。。）。

② 外界干扰

强磁场干扰（磁性数据被清洗），灾害。

2）特征

外存数据库中数据的部分或全部丢失，数据存储文件本身被破坏。

目前比较成熟的DBMS软件一般能够在服务启动时校验存储介质上的数据文件是否异常。

10.4 恢复技术

备份 + 日志

10.4.1 备份技术

1、备份方式

1）静态备份（转储：dump）

——数据库系统中无事务运行时进行转储。

① 特征：

- 转储期间不对数据库进行任何操作；
- 得到一个一致性付本。

② 优点——简单

③ 缺点：停止一切事务运行；降低数据库可用性。

2）动态备份

转储与事务并发执行。

① 特征

转储期间可对数据库进行存取与修改操作。

② 优点

不影响事务运行。

③ 缺点

获得一致性副本较麻烦。

时刻	事务1	事务2
转储A时	A=100,B=200	
转储B时		A=200,B=100

如转储A时，其值为100，B值为200，但在随后转储B时，但另一事务修改数据库为A=200，B=100，这样备份副本是与DB中实际值不一致的过时数据。

2、备份策略

1）海量备份

① 方法：定期或不定期将数据库全部数据转储。

② 优点：简单。

③ 缺点：重复转储；

转储量大；

停止运行（多为静态转储）。

2）增量备份（incremental clumping）

① 方法：每次转储上次转储后更新过的数据。

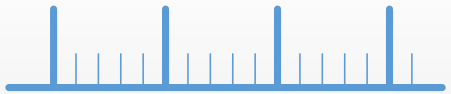
② 优点：备份量小。

③ 缺点：恢复过程较复杂。

（完全、

累积——自上次完全备份或者累积备份之后的修改、

增量——自上次完全或者累积或者增量备份之后的修改）



3）写副本

① 方法：每次写时，同时写另一个副本。

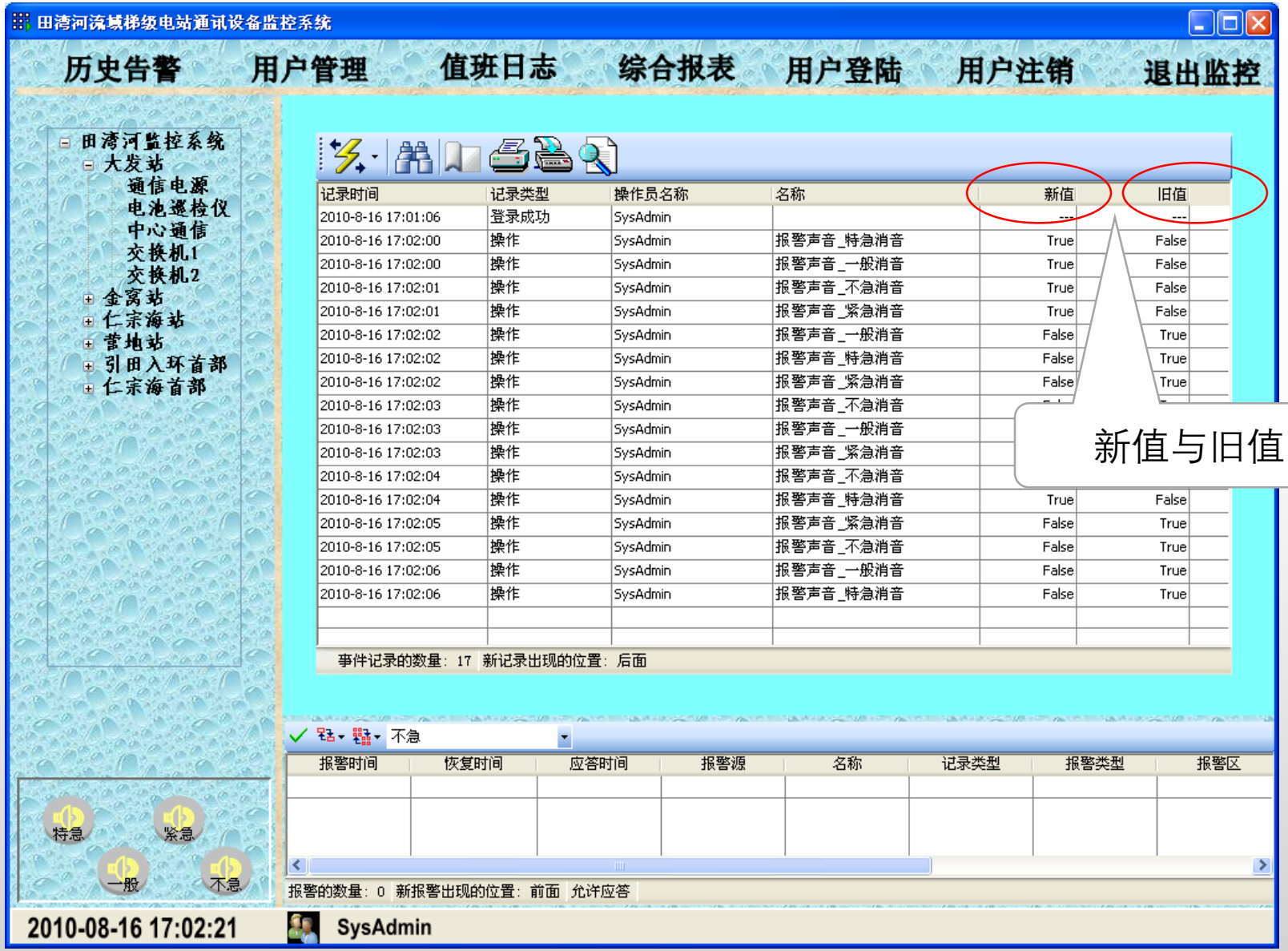
② 优点：简单。

③ 缺点：重复写，操作效率下降。

10.4.2 日志（logging）

某工厂机组值班日志

某电站通讯设备监控日志



某市场经理值班日志

事件处理状态：
正在处理、
处理完毕

10. 4.2 日志（logging）

实际生活中的日志——对某个设备、某种资源的动态变化过程的历史记录，以便对曾经发生的问题进行分析

——故障分析

——安全性分析

。。。

例如时间、地点

包括先后顺序

需求：客观记录事件或者操作的当时的环境、动作的过程、造成的影响，并且便于查找。

例如旧状态和新状态、是否正常完成

在数据库系统中，数据库文件也是一种设备和资源，也有类似的需求，对应的有日志文件及其维护机制。

有了日志，数据库的恢复子系统就有了“自己的数据”。

10. 4.2 日志（logging）

1、日志概念

——记录事务对数据库更新操作的文件称之为日志文件。

2、日志文件类型

1）记录为单位日志文件;

2）数据块为单位日志文件。

两种日志都对应数据库操作

3、以记录为单位的日志文件内容

1）事务开始标记（一个日志记录）；
2）事务结束标记（一个日志记录，提交/撤销）；
3）每个事务的所有更新操作（每个操作一个日志记录）。

如何描述“操作”？事务夭折有无“操作？”

每个**日志记录**内容（记录头+记录体）：

1）事务标识（TRID）；

2）操作类型（插入/删除/修改）；

3）操作对象标识;

4）更新前数据旧值;

5）更新后数据新值。

4、以数据块为单位的日志文件内容

事务标识+数据块

1）数据块（整块）更新前内容;

2）数据块更新后内容。

5、影子拷贝与影子页面

影子拷贝是一种数据库的模式，要更新数据库的事务首先创建数据库的一个完整拷贝，所有更新在该拷贝上进行。

数据库指针：影子拷贝中使用一个指针来标识数据库的当前拷贝，称为数据库指针，该指针存放于磁盘上。

影子拷贝事务的执行

- 若事务半途中止，仅需删除新生成的拷贝，旧版数据库拷贝不受影响。
- 若事务提交，先将新拷贝的所有页面写到磁盘，完成这一批写操作后，更新数据库指针使其指向新版的拷贝，更新后的数据库指针写到磁盘上时意味着事务提交完成。更新完成后，可删除旧拷贝。

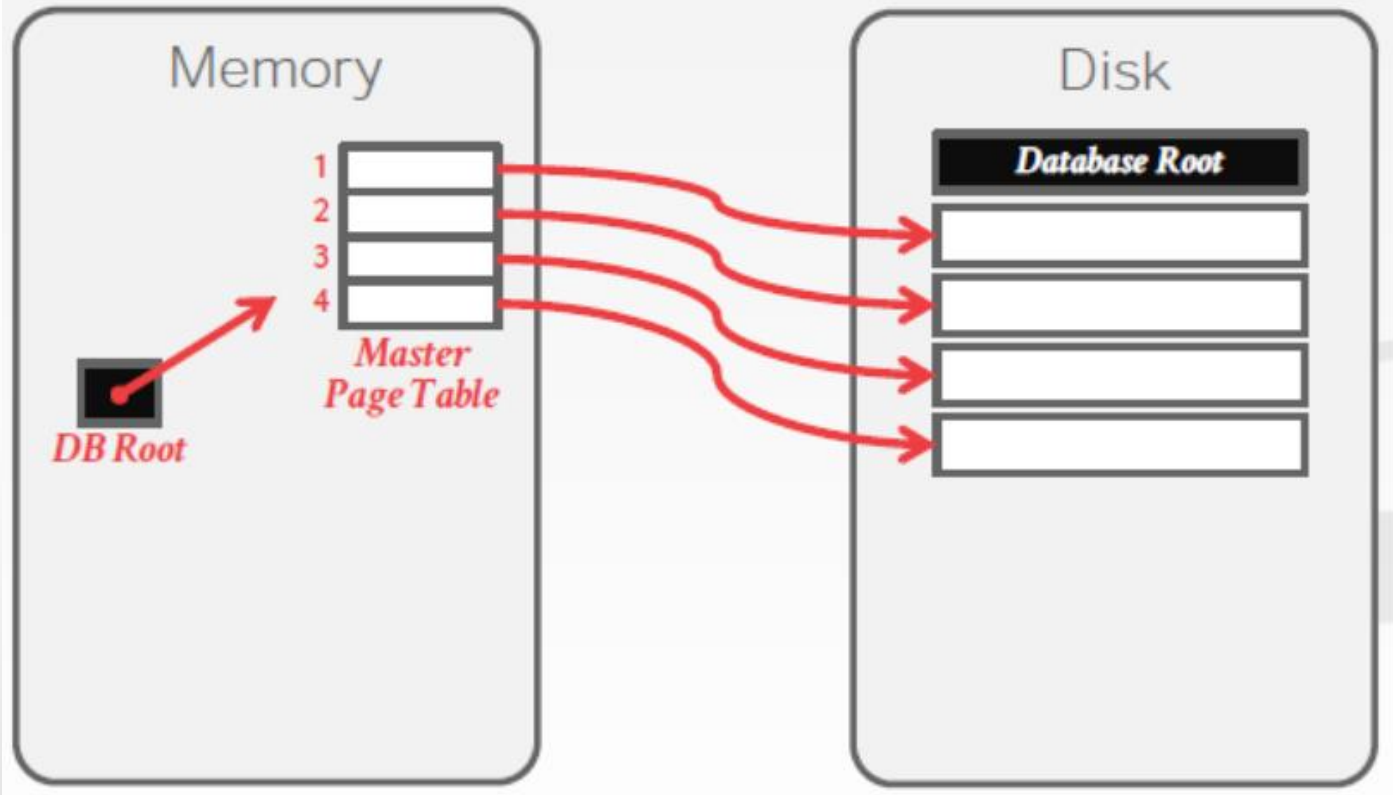
影子拷贝事务的原子性：依赖于对数据库指针的写操作的原子性，该原子性来源于磁盘系统对单个块（单个磁盘扇区）的原子性更新机制。只要能保证数据库指针处于单个磁盘块（或者单个扇区），即可实现影子拷贝事务的原子性。

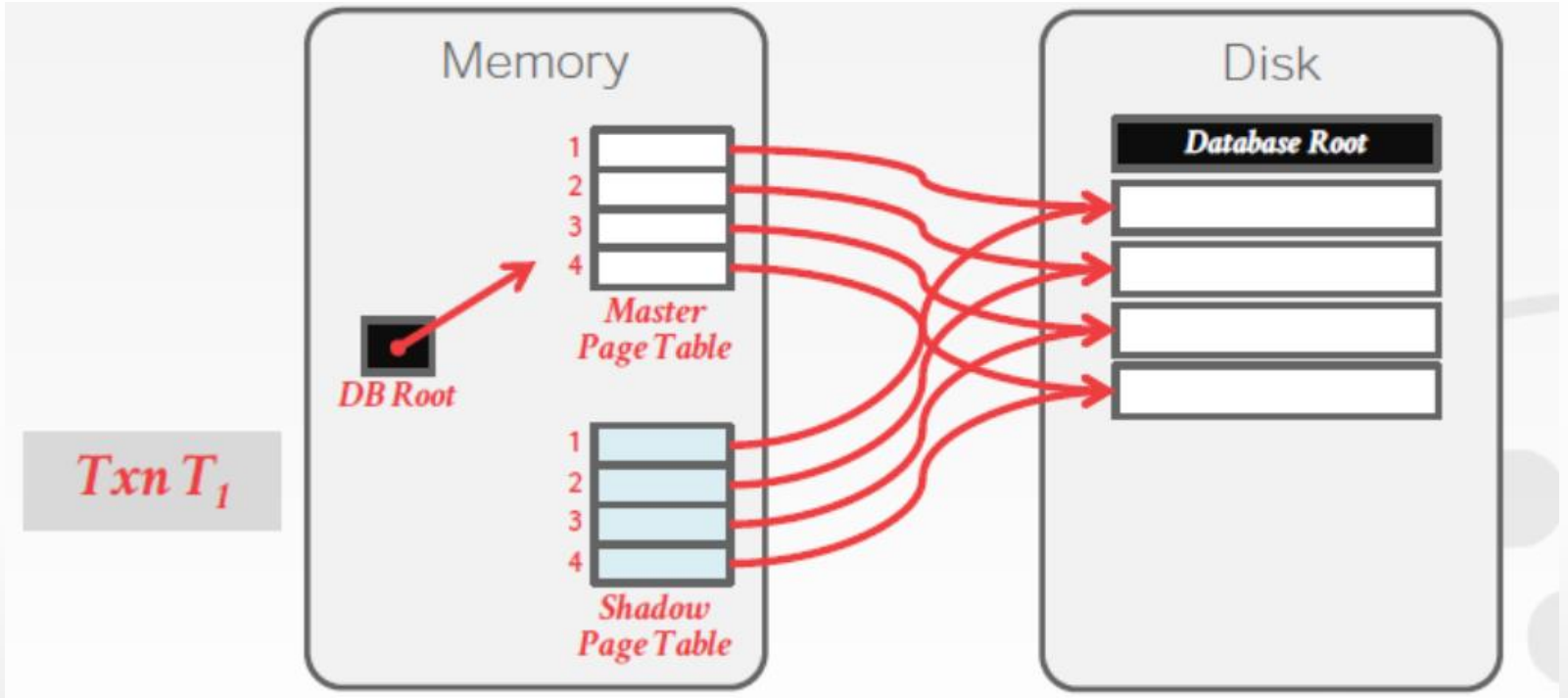
影子拷贝模式可用于小的数据库，也普遍用于正文编辑器，但对于大型数据库则开销过大，此时可采用影子拷贝方法的一个变种——影子页面，以减少拷贝的开销。

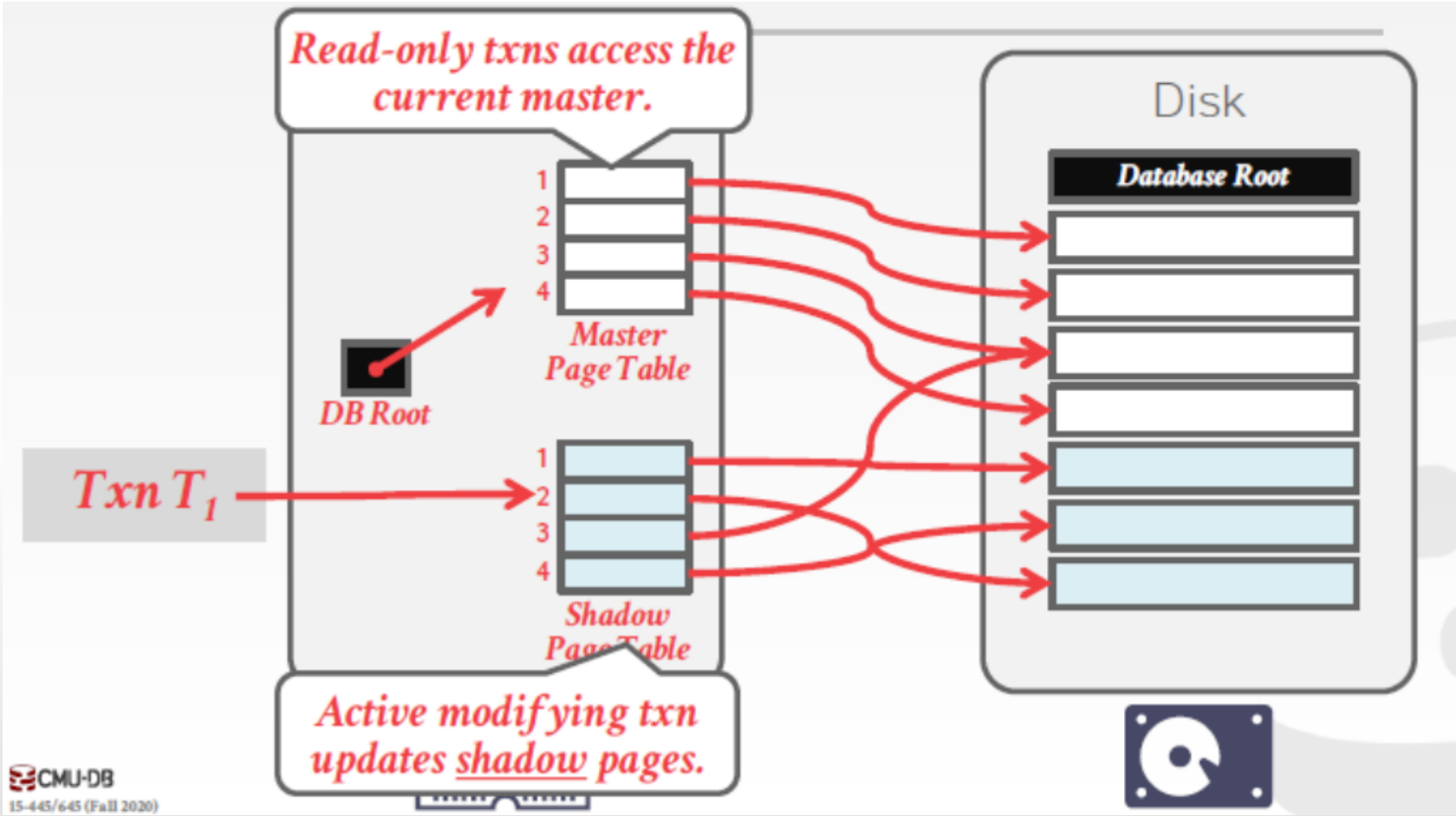
影子页面：使用一个包含指向所有页面的指针的页表，页表的作用和数据库指针相同。影子拷贝只将页表和所有更新的页面拷贝到一个新位置，当提交事务时，原子性的更新指向页表的指针以指向新拷贝。

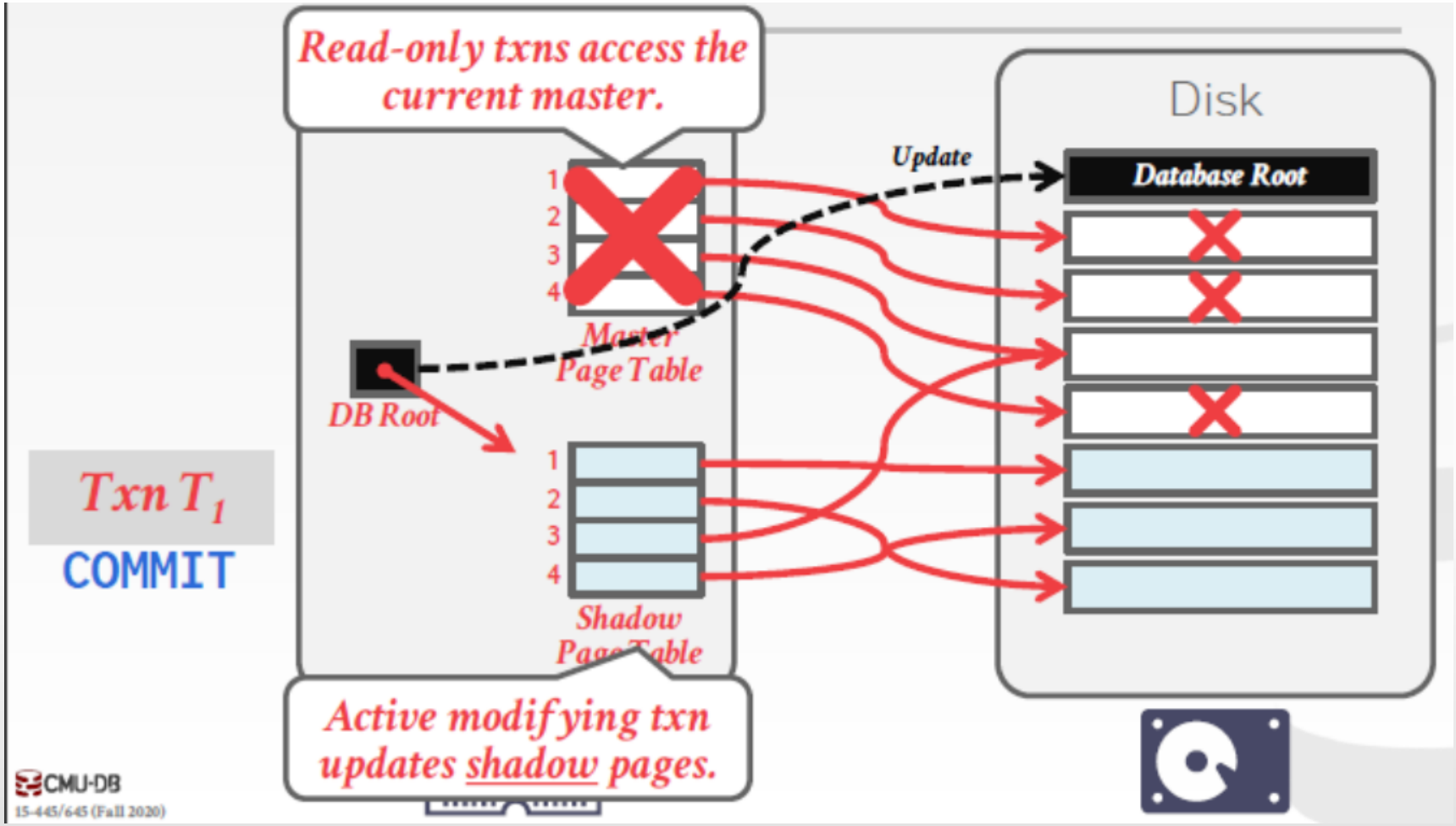
影子页面的局限性：对并发事务支持较弱，在数据库中未广泛使用。

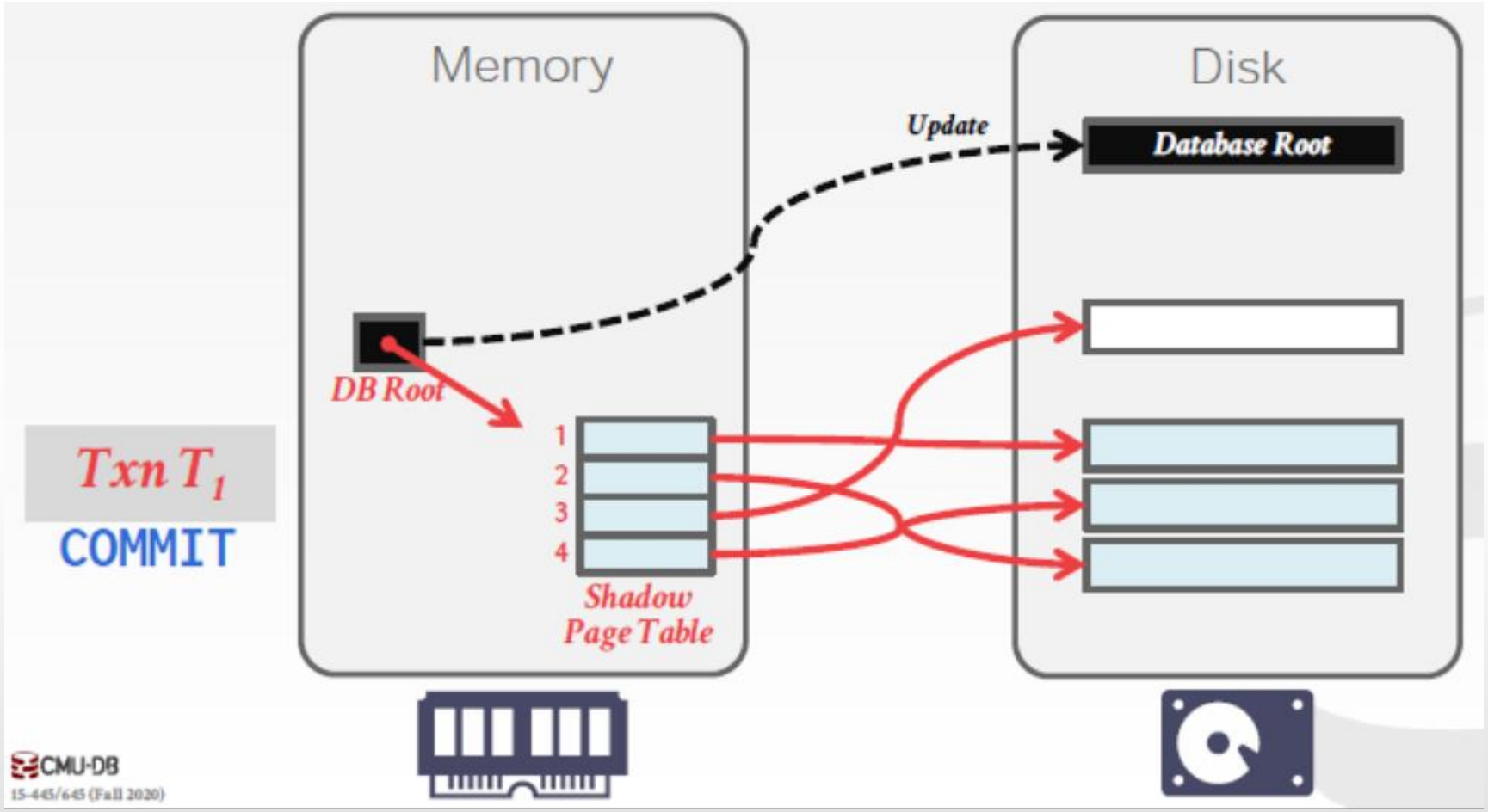
例：影子页面




Memory
Disk
Database Root
1
2
3
4
Master
Page Table
DB Root
Txn T_1
1
2
3
4
Shadow
Page Table


Read-only txns access the current master.
Disk
Database Root
1
2
3
4
Master Page Table
DB Root
Txn T_1
1
2
3
4
Shadow Page Table
Active modifying txn updates shadow pages.
CMU-DB
15-445/645 (Fall 2020)


Read-only txns access the current master.
Disk
Update
Database Root
1
2
3
4
Master Page Table
DB Root
Txn T_1
COMMIT
1
2
3
4
Shadow Page Table
Active modifying txn updates shadow pages.
CMU-DB
15-445/645 (Fall 2020)


Memory
Disk
Update
Database Root
DB Root
1
2
3
4
Shadow
Page Table
Txn T_1
COMMIT
CMU-DB
15-445/645 (Fall 2020)

6、日志管理

1）按事务操作执行时间顺序记日志（多个事务操作并发）；

2）须先写日志后写DB文件！！！！！

先写日志协议——WAL

7、用途

1）事务恢复

2）DB故障恢复

3）系统分析

联机日志文件和归档日志文件

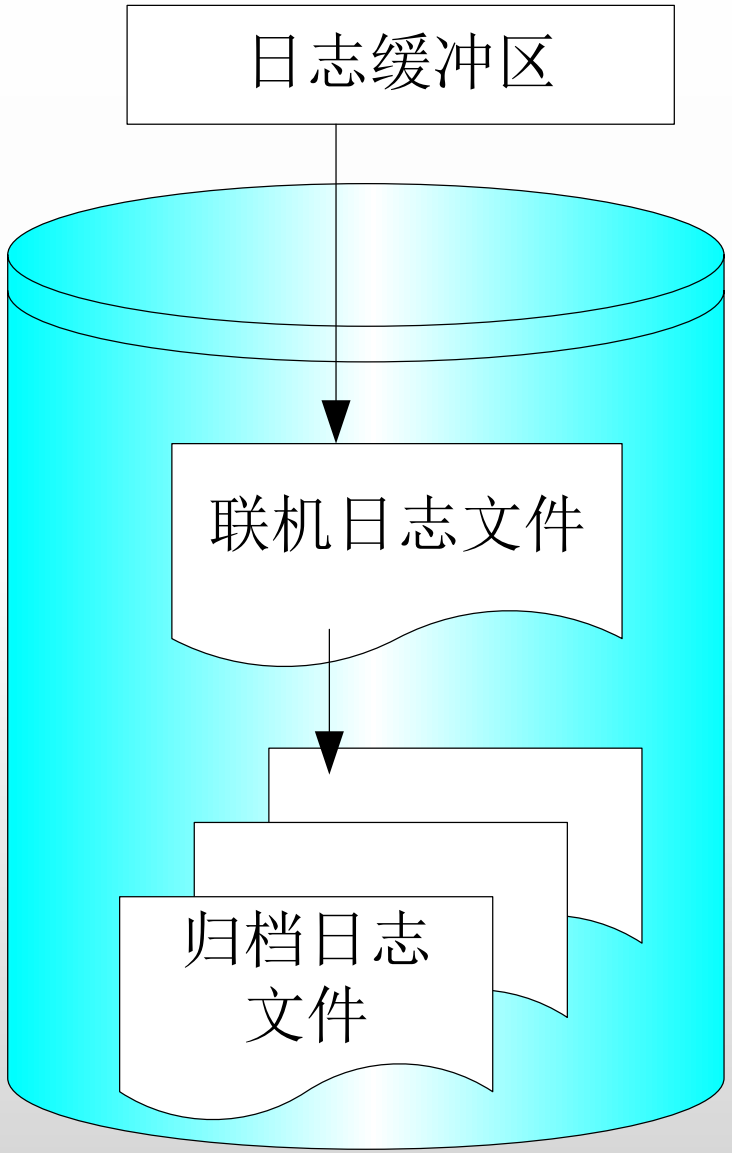
日志文件可分为两类:

1）联机日志文件

大小有限，直接和DBMS日志缓冲区关联，保存数据库当前一段时间内的事务执行的变化过程，主要用于事务故障和系统故障。

2）归档日志文件。

保存历史上所有的不再用于联机处理的日志记录，由联机日志文件归档而成，主要用于介质故障的恢复。



日志记录的使用（操作）

REDO——可理解为将日志记录对应的操作执行一遍，运用数据的后像。

UNDO——可理解为将日志记录对应的操作的逆操作执行一遍，运用数据的前像。

注：1）执行的方向；
2）DB的状态。

- 幂等性

Why?

每个日志记录的UNDO操作和REDO操作都具有幂等性，即无论重复执行多少次，效果等同于执行一次。

日志文件是一个单调递增的文件

每个日志记录在日志中都有一个唯一的码，叫做日志序号（Log Sequence Number，简称LSN）。

日志文件是按照LSN单调递增的顺序文件，如果操作A的日志记录在操作B的日志记录之后生成，则LSN(A)>LSN(B)。

LSN的实现：

一般由日志文件序号和记录在文件中的相对地址两部分组成。

问题：事务的撤销如何遵守日志文件的单调递增特性？

redo-only日志（补偿日志，compensation log）：Undo操作除了将数据项设置成旧值，还额外生成一条“redo-only”日志记录来体现该数据的更新动作，该日志记录不需要包含数据项的旧值。

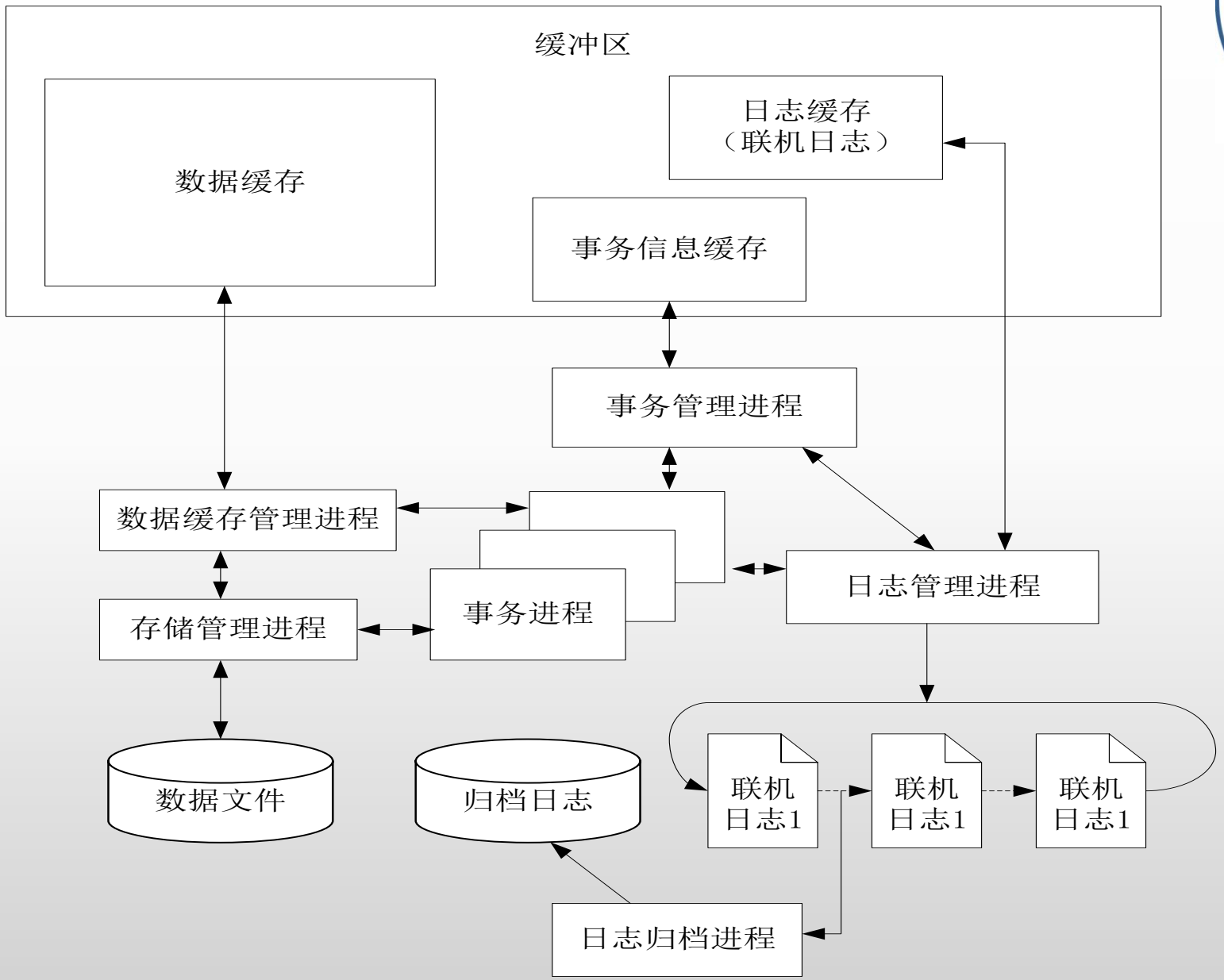
日志记录的REDO和UNDO执行方式

对日志文件中的多条日志记录进行REDO，要按照日志序号(LSN)递增的顺序进行，即正向扫描日志文件进行REDO。

对日志文件中的多条日志记录进行UNDO，要按照日志序号(LSN)递减的顺序进行，即反向扫描日志文件进行UNDO。

当要对日志文件中的多条交错而无排列规律的日志记录进行REDO和UNDO操作时，只需各自按照上述规则进行，从基本原理上分析，REDO和UNDO之间原则上没有先后的要求，执行多次也没有影响。

DBMS的缓存与核心进程



■ 缓存中的日志内容单调递增，缓存资源有限，因此，DBMS需要建立缓存中的日志写出到磁盘日志文件的机制，其中，触发日志缓存页面写出到外存联机日志文件的原因包括：

事务提交

缓冲区容量使用达到一定限度

DBMS的shutdown、检查点

日志相关的协议（保证事务的原子性和持久性）：

先写日志协议WAL——在覆盖一个外存页面之前，必须先强制写出该页面新版本对应的日志记录。

先写日志协议实现技术：每个数据页面有一个字段记录最近版本对应的日志记录的LSN，写出该页面前，调用Log_flush内部函数将该LSN之前的所有日志记录写出。

提交时强制写日志协议force-log-at-commit——作为提交工作的一部分，必须强制写出该事务的所有日志记录。

实现技术：调用Log_flush将该事务的commit日志记录及其之前的所有日志记录写出。

成组提交技术：当日志记录产生较为频繁时，凑齐一组（一个日志缓存页面）的日志记录才执行日志缓存写出操作，在此之前改组日志对应的事务提交均处于等待状态。

7、缓存管理

■数据缓存中的页面可分为两类：干净页面和脏页面。

干净页面——由已提交的事务更新完的页面

脏页面 ——由未完成的事务更新的页面

■系统性能相关的问题

▪ 干净页面的写出策略

强制写（force）——事务提交时先写出所有该事务相关的干净页面，然后才能完成提交。

非强制写（no-force）——即使事务修改的磁盘块没有全部写回到磁盘，也允许它提交完成。

▪ 强制写与非强制写对性能的影响

强制写策略增加了事务提交的响应时间，还造成频繁的I/O操作，降低了系统的事务吞吐率。

▪脏页面写出的策略

隐　　形（steal）——脏页面可随时由于页面替换被写出

非隐形（no-steal）——脏页面被固定在缓冲区中，不允许被替换出去，直到事务结束（提交或者夭折）

▪隐形与非隐形对性能的影响

非隐形策略增加了对缓冲区容量的要求

实现机制： 缓存页面固定

原理： 对数据页面封锁

锁的类型： 闩锁（latch）

缓冲页面上的闩锁与事务并发控制的锁无关，是一种短期持有的锁。

先写日志协议要求缓冲页B_X的相关日志必须在B_X输出之前输出到磁盘，并且，为了保证先写日志协议，在输出缓冲页B_X到磁盘时，不能允许同时有向B_X缓冲页的写操作。

相应措施：缓冲页面当事务要对一个数据项执行写操作时，需要先获得该数据项所在页面的排它锁（闩锁），并且更新完后立即释放该锁。

使用闩锁时的缓冲页B_X输出到磁盘的过程：

（1）获取B_X上的排他锁闩锁，以确保没有其他事务正在对B_X执行写操作；

（2）将B_X相关的全部日志记录输出到磁盘；

（3）将B_X输出到磁盘；

（4）释放B_X上的排他锁闩锁。

申请和释放缓冲页面的闩锁即使不遵循两阶段锁协议，也不影响事务的可串行性。

- 强制写与非强制写对日志与恢复的影响

  强制写策略减少了对日志记录中REDO信息的需求，恢复时一般不需要REDO操作。

- 隐形与非隐形对日志与恢复的影响

  非隐形策略减少了对日志记录中UNDO信息的需求，恢复时一般不需要UNDO操作。

No REDO?

No UNDO?

■强制写 + 非隐形策略？→日志

日志不能省略

1）介质故障→REDO信息的必要性

2）提交的过程中，页面是脏的，隐含着采用了隐形策略，除非采用更加复杂的影子页算法，保证提交操作的原子性。

■ 一般采用非强制写 + 隐形策略，以降低系统开销，提高响应能力。→日志的开销。

10.5 恢复策略

1、事务故障恢复
——因各种故障导致事务未执行完而abort时的恢复。
1）目标：维护原子性

2）恢复步骤
① 反向扫描日志文件：
——查事务执行过的更新操作；
② 执行该事务的最晚日志记录的UNDO操作；
③ 循环执行上述操作并同样处理，直至事务开始标记。

3）特点
DBMS自动完成

2、DB故障恢复

1）系统故障

——撤消故障发生时未完成事务和重做已完成事务的恢复。

① 目标：原子性、持久性

执行方向

② 步骤

·正向扫描日志文件；
·找出故障发生前已提交事务，该事务标识记入REDO队列；
·找出故障发生时未完成事务，该事务标识记入UNDO队列；
·依照日志记录反向顺序对UNDO队列中事务进行UNDO操作：
（反向扫描日志文件，执行该事务的UNDO操作）；
·依照日志记录正向顺序对REDO队列中事务进行REDO操作；
（正向扫描日志文件，执行该事务的REDO操作）。

③ 特点：DBMS自动完成。

扫描的范围

3、 介质故障的恢复

1. 重装数据库，

   使数据库恢复到一致性状态

2. 重做已完成的事务

9.5.3 介质故障的恢复

- 恢复步骤

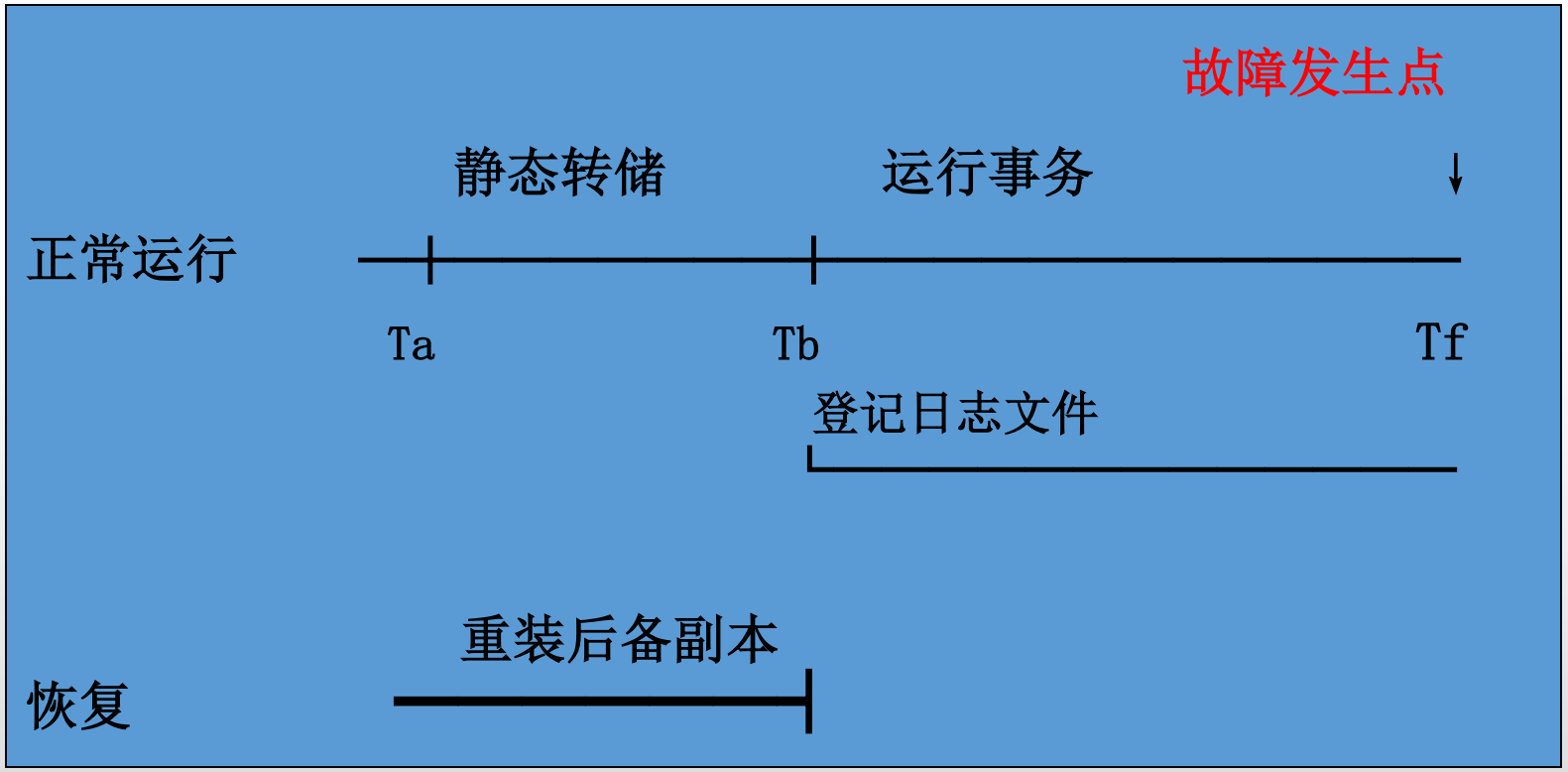
1. 装入最新的后备数据库副本，使数据库恢复到最近一次转储时的一致性状态。
   - 对于静态转储的数据库副本，装入后数据库即处于一致性状态
   - 对于动态转储的数据库副本，还须同时装入转储时刻的日志文件副本，利用与恢复系统故障相同的方法（即REDO+UNDO），才能将数据库恢复到一致性状态。

介质故障的恢复（续）

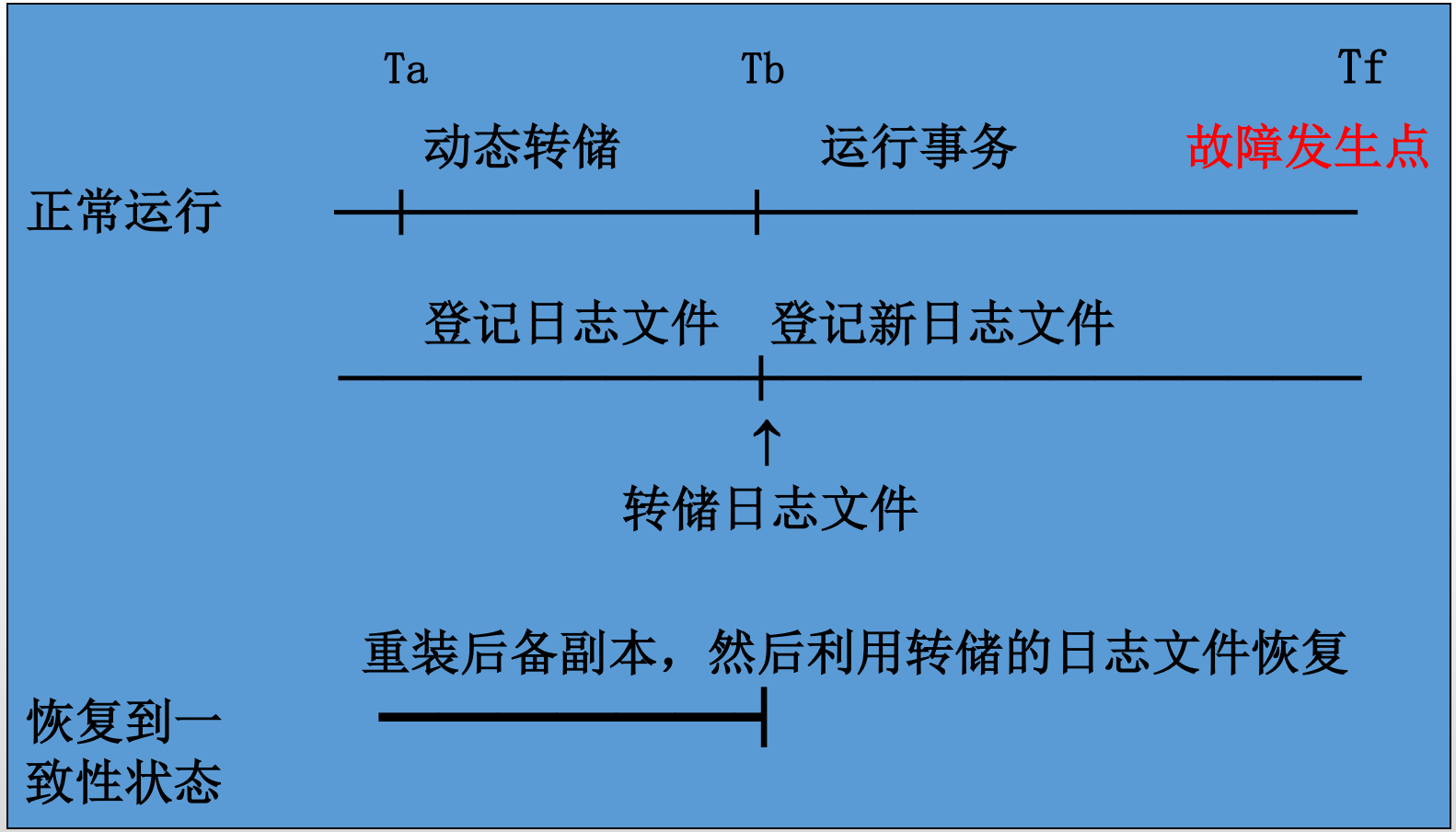
2. 装入有关的日志文件副本（转储结束时刻的日志副本），重做已完成的事务。

- 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。

利用静态转储副本将数据库恢复到一致性状态



利用动态转储副本将数据库恢复到一致性状态



介质故障的恢复（续）

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

10. 6 具有检查点的恢复技术

10.6.1 产生的原因

利用日志恢复的过程需要扫描全部的日志记录，进行相关的恢复操作，而日志文件过大将带来大量的恢复操作。

恢复涉及两类操作，redo和undo，二者都是“必须”的么？

很多需要redo的事务的更新操作结果已经被写入磁盘文件中，对其redo没有必要。

↓ 问题：存在不确定性，解决思路：增加确定性

优化机制——周期性的建立检查点（checkpoint）。

10.6.2 检查点机制

在日志中增加新的一类记录（检查点记录），并增设重新开始文件。

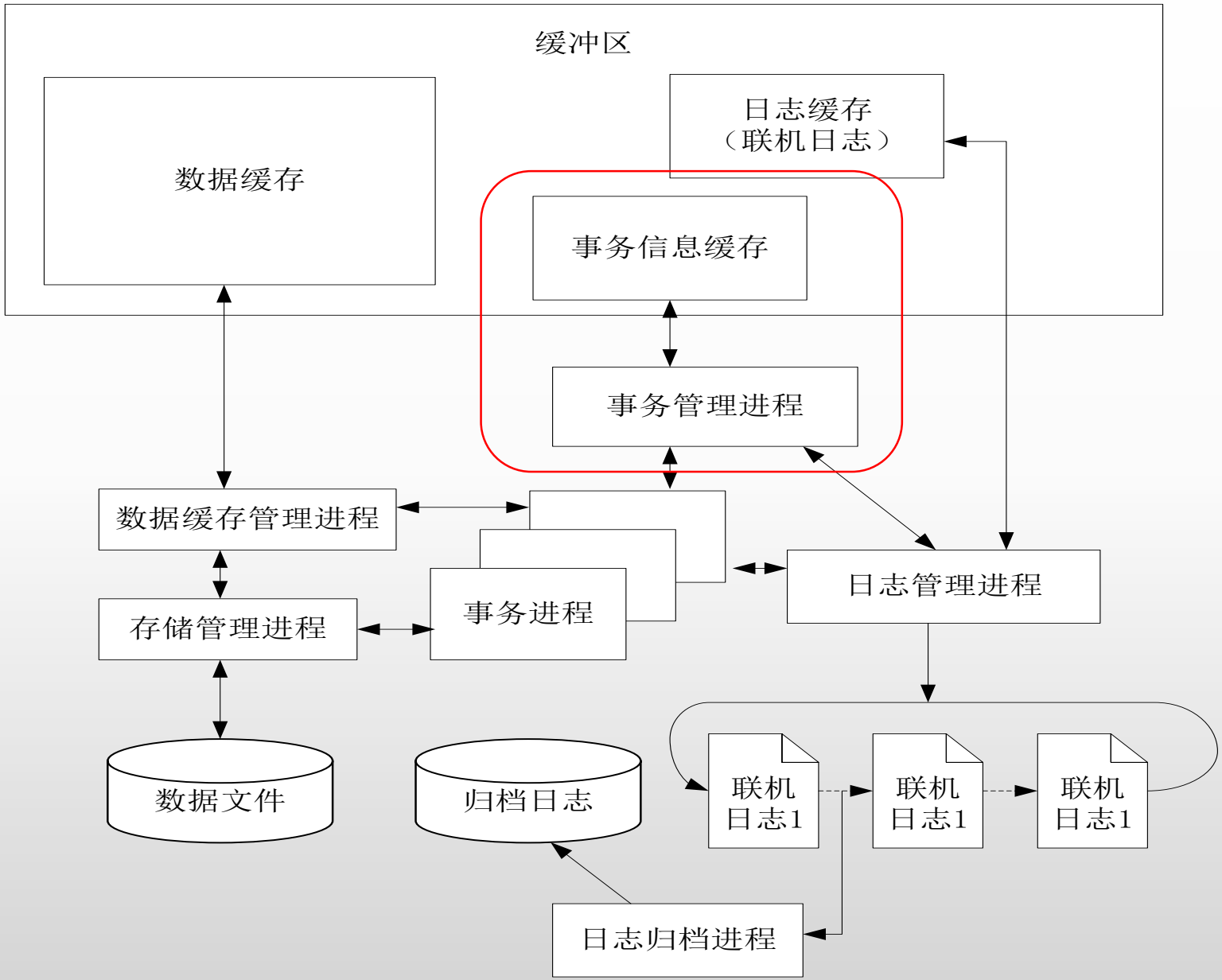
•检查点记录内容：

1）建立检查点时刻所有正在执行的事务清单（Active Transaction Table）；

2）上述事务最近一个日志记录的地址。

目的?

【注】：检查点→日志归档的可行性


缓冲区
数据缓存
日志缓存
（联机日志）
事务信息缓存
事务管理进程
数据缓存管理进程
存储管理进程
事务进程
日志管理进程
数据文件
归档日志
联机
日志1
联机
日志1
联机
日志1
日志归档进程

生成检查点的时机——周期性

时间周期

日志记录周期

•检查点的动作（基本检查点策略，清晰检查点）

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;

2）在日志文件中写入一个检查点记录;

3）把当前数据缓冲区的所有数据记录写入磁盘数据文件;

4）在重新开始文件中记录检查点记录的地址。

缓冲页面的闩锁可以在检查点过程中保护缓冲页面不更新，从而没有产生新的日志记录，检查点执行完成后释放闩锁。

•使用检查点的恢复技术

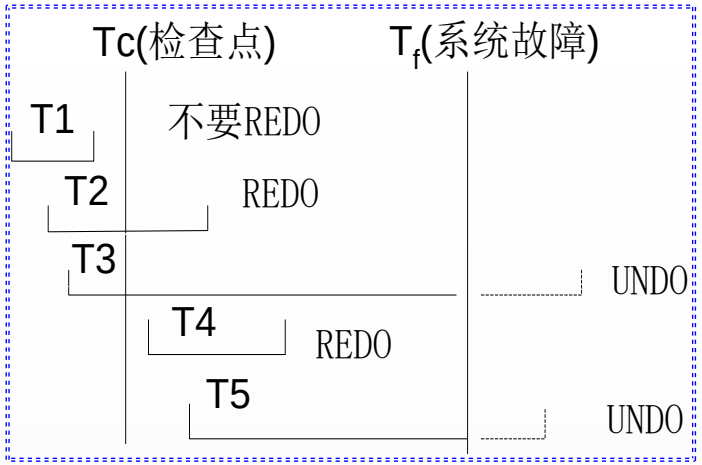
1）从重新开始文件中找到最后一个检查点的信息，并从日志文件中找到该检查点记录；

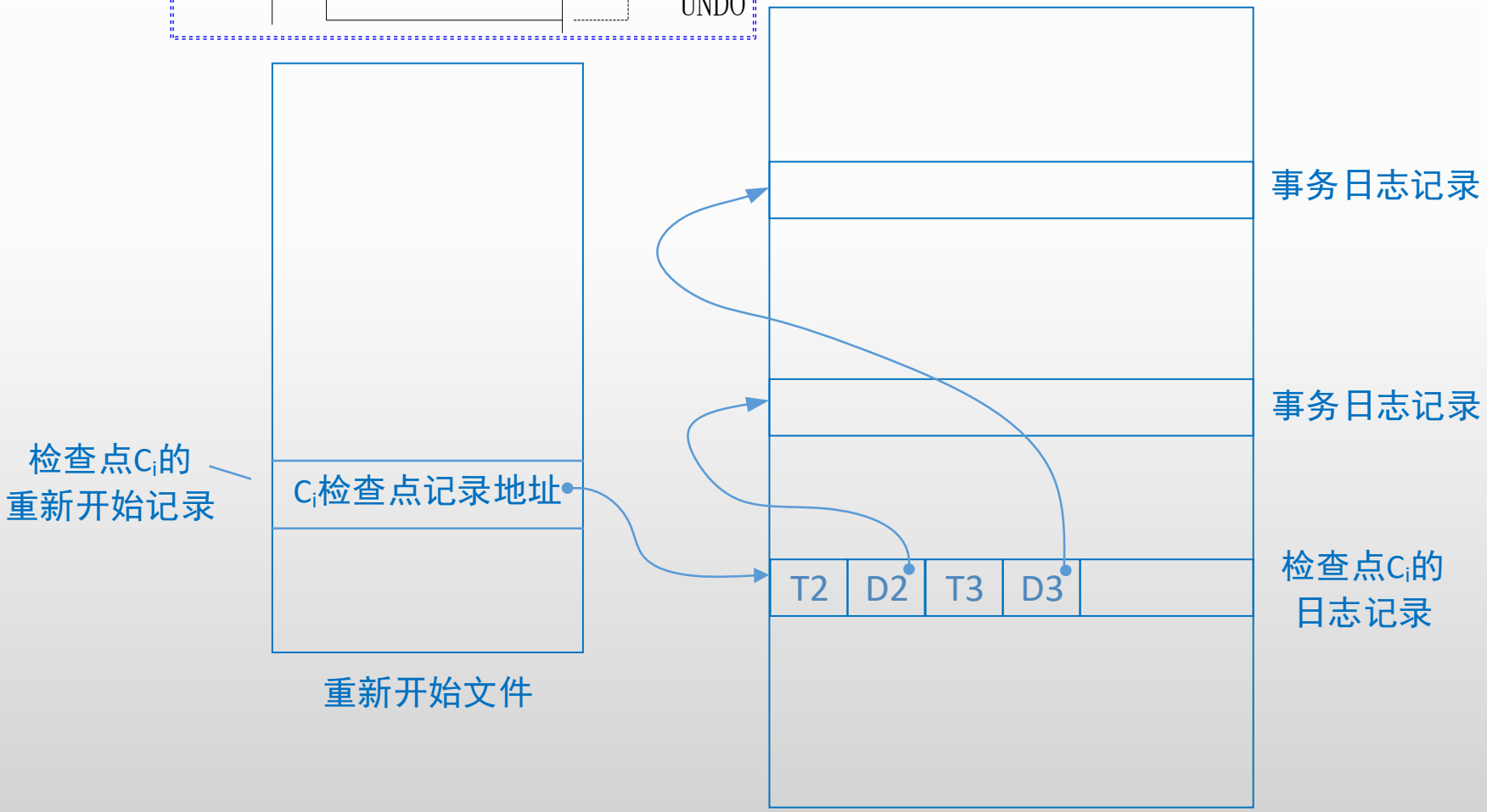
2）从检查点记录中得到ACTIVE-TRANSACTION-LIST，并暂时将其全部列入UNDO-LIST队列，而REDO-LIST队列初始化为空；

3）从检查点开始正向扫描日志文件，新开始的事务并入UNDO-LIST，遇到事务提交的日志记录，则该事务从UNDO-LIST移入REDO-LIST，直到日志文件尾；

4）以检查点记载的最早日志记录和日志文件末尾为界，分别对UNDO-LIST和REDO-LIST执行UNDO和REDO操作。


Tc(检查点)
T_f(系统故障)
T1
不要REDO
T2
REDO
T3
UNDO
T4
REDO
T5
UNDO


Tc(检查点)
T_f(系统故障)
T1
不要REDO
T2
REDO
T3
UNDO
T4
REDO
T5
UNDO


检查点C_i的
重新开始记录
C_i检查点记录地址
重新开始文件
事务日志记录
事务日志记录
T2
D2
T3
D3
检查点C_i的
日志记录

10. 7 DB镜像（DB mirror）

1、原因：介质故障：中断运行，周期备份，恢复麻烦

2、方法：利用自动复制技术（例如日志文件镜像）

3、策略

1）整个DB/关键数据复制到另一个介质（镜像磁盘）；

2）DB更新时，DBMS自动将更新结果复制到该副本；

3）故障发生时，利用该镜像磁盘进行恢复。

4、优点

1）无需关闭系统（自动进行镜像复制）；

2）无需重装副本，自动保证一致性；

3）提高可用性；

4）提高并发性。

5、缺点

频繁复制更新，效率下降。

实用案例：双机热备（双机互备援Dual Active、双机热备份Hot Standby）、远程备份。

6、关键技术

如何监测（心跳线、PARTNER TIMEOUT、证人机制）、是否自动恢复。

例：假设日志记录有如下几种类型：

<START T>：表示事务T开始；　　<COMMIT T>：表示事务T已经提交；

<T,X,u,v>：表示事务T修改了数据X，其原来的值是u,更新后的值是v。

若某系统故障发生时，磁盘上日志文件的内容为：

<START T>;<T,C,29,30> <T,A,10,11>; <START U>; <U,B,20,21>;<T,C,30,31>;<U,D,40,41>;

<U,B,21,22> <COMMIT U>，请简述恢复的过程。

例：若某系统故障发生时，磁盘上日志文件的内容为：
<START T>;<T,C,29,30> <T,A,10,11>; <START U>;
<U,B,20,21>;<T,C,30,31>;<U,D,40,41>;
<U,B,21,22> <COMMIT U>，如何进行恢复？

恢复过程：

(1)正向扫描日志，由于事务U有START和COMMIT，放到REDO队列中，而事务T只有START，故而放到UNDO队列。

(2)对UNDO队列中进行UNDO操作，即对T进行UNDO操作，即反向扫描T的日志，即对C写30，对A写10，对C写29。

(3)对REDO队列中进行REDO操作，即对U进行REDO操作，即正向扫描U的日志，即对B写21，对D写41，对B写22。

注意：写成类似“C由29变做30”都是错误的。

思考：如果增加检查点记录，本题有何变化？

课后作业第2题，
日志表述方式不同

慕课讨论题

- 数据库系统哪些情况下会将缓存中的日志文件写出到磁盘?

日志在数据库系统的恢复中发挥着重要的作用，日志在什么情况下需要写出到磁盘文件?

- 检查点机制对性能可能产生哪些影响?

数据库系统的检查点是其恢复子系统的一种周期性执行的机制，该机制对于数据库系统的性能有哪些影响?