

# 数据库系统原理

教程：数据库系统概论（第5版）

结合：CMU 15-445/645 INTRO TO DATABASE SYSTEMS

华中科技大学 计算机学院

左琼



# 第十章 数据库恢复技术

*Principles of Database Systems*

# 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 具有检查点的恢复技术

## 10.7 数据库镜像

## 10.8 小结

- 故障种类：  
事务故障、系统故障、介质故障、病毒
- 恢复的实现技术：“冗余”
  - 数据转储
  - 登记日志文件

## 10.5 恢复策略

### 1. 事务故障的恢复

**事务故障：**事务在运行至正常终点前被终止。

**目标：**维护事务的原子性

例：

```
begin tran
```

```
insert into sales values('A0002',0,10);
```

```
update sales set b=b-3 where id = 'A0001'
```

```
update sales set b=b-3 where id = 'A0001'
```

```
commit tran
```

□ **日志文件：**

```
<T0 start>
```

```
<T0, I, sales, null, ('A0002',0,10)>
```

```
<T0, U, sales, b(id='A0001'),30,27 >
```

发生故障，  
事务终止

## 10.5 恢复策略


### 1. 事务故障的恢复方法（续）

恢复方法：利用log **UNDO**此事务对DB的修改。

步骤：UNDO操作

- 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。
- ③ 继续①→②。
- 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。

特点：由DBMS自动完成



日志1	日志2	日志3	日志4	日志5	日志6	日志7	日志8	日志9
事务1 开始	事务2 开始	事务3 开始	事务1 插入A	事务2 插入B	事务3 插入C	事务3 提交	事务2 删除C	事务1 插入D

# 10.5 恢复策略

## 2. 系统故障的恢复

**目标：**撤销故障发生时未完成的事务，重做已完成的事务。

**步骤：** UNDO未完成事务， REDO已完成事务

- ① **正向扫描**日志文件，找出在故障发生前已经提交的事务，将其事务标识记入**重做(REDO)队列**；同时找出故障发生时尚未完成的事务，将其事务标识记入**撤销(UNDO)队列**。
- 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UNDO事务执行UNDO操作；
- ③ 对重做队列中的各个事务进行重做(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。即将日志记录中“更新后的值”写入数据库。

**特点：**由DBMS在**重启**时自动完成

# 10.5 恢复策略

## 2. 系统故障的恢复

日志文件的内容:

序号	日志
1	T1: 开始 <span>A=0, B=0, C=0</span>
2	T1: 写A, A=10
3	T2: 开始
4	T2: 写B, B=9
5	T1: 写C, C=11
6	T1: 提交 <span>A=10, C=11</span>
7	T2: 写C, C=13
8	T3: 开始
9	T3: 写A, A=8
10	T2: 回滚 <span>C=11, B=0</span>
11	T3: 写B, B=7
12	T4: 开始
13	T3: 提交 <span>A=8, B=7, C=11</span>

若系统故障发生在11后, 则:

UNDO\_LIST为{T2,T3}

REDO\_LIST为{T1}

恢复结果:

A=10,

B=0,

C=11

# 10.5 恢复策略

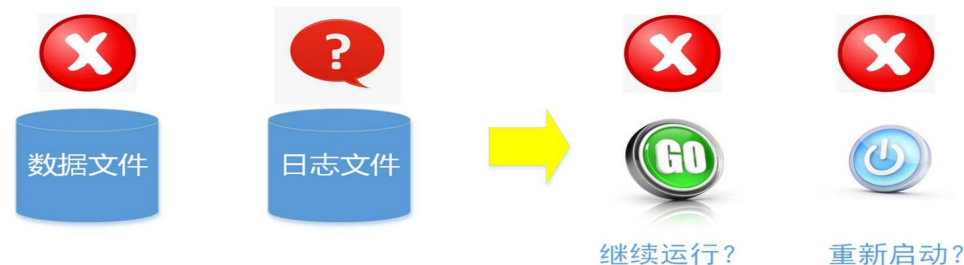
## 3. 介质故障的恢复

**目标：**数据和日志文件被破坏时的恢复，维护事务的持久性。

**步骤：**

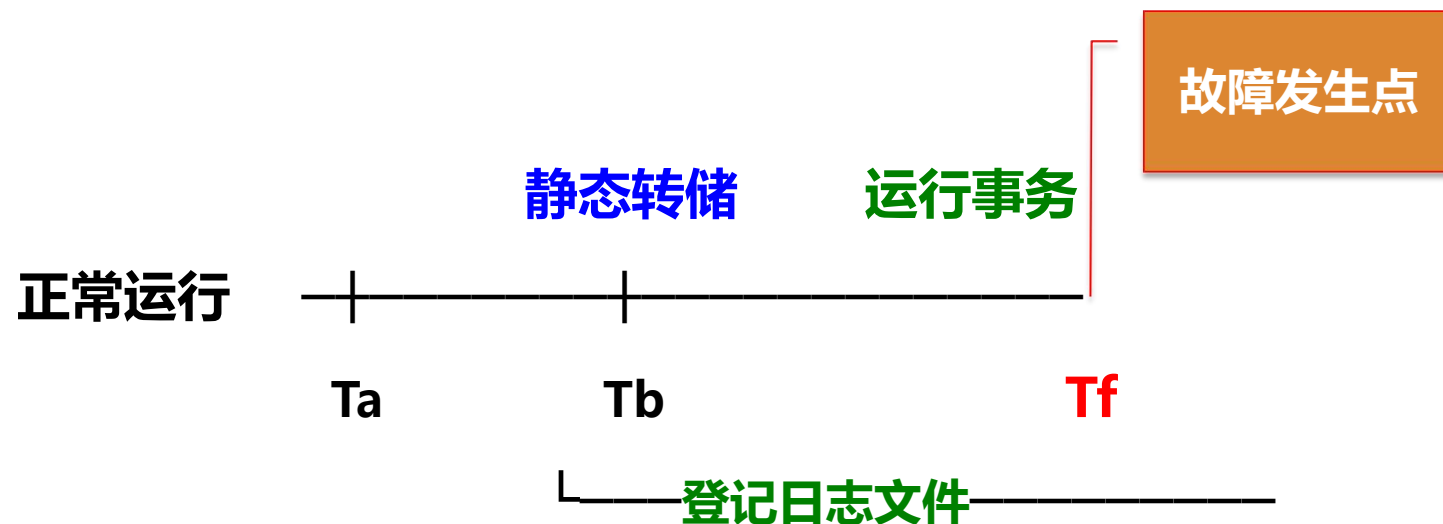
- ①装入最新的**后备数据库副本**，使数据库恢复到最近一次转储时的一致性状态。
  - 对于静态转储的数据库副本，装入后数据库即处于一致性状态；
  - 对于动态转储的数据库副本，须同时装入转储时刻的日志文件副本，利用与恢复系统故障相同的方法(即REDO+UNDO)，才能将数据库恢复到一致性状态。
- ②装入有关的**日志文件副本**，**重做已完成的事务**。

**特点：**需要DBA干预。DBA负责：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





# 介质故障的恢复



## 10.6 CheckPoint技术

### □ 问题的提出:

- 日志文件的缺点: 耗费大量时间; 重复执行; 耗费大量空间;
- REDO操作, 重新执行, 浪费。

### □ 基本策略:

周期性地对日志做检查点, 保存DB状态, 避免故障恢复时检查整个日志。

### □ 方法: 在日志文件中增加一类新的记录——检查点(Checkpoint)记录, 并增加一个 “重新开始文件”。周期性执行如下步骤:

- ① 将日志缓冲区中的日志记录写入磁盘;
- ② 在日志文件中写一个检查点记录;
- ③ 将数据缓冲区中的数据写入磁盘DB中;
- ④ 将检查点记录在日志文件中的地址写入重新开始文件。

# 10.6 CheckPoint技术

## □ 检查点记录的内容:

- checkpoint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在日志文件中的地址。

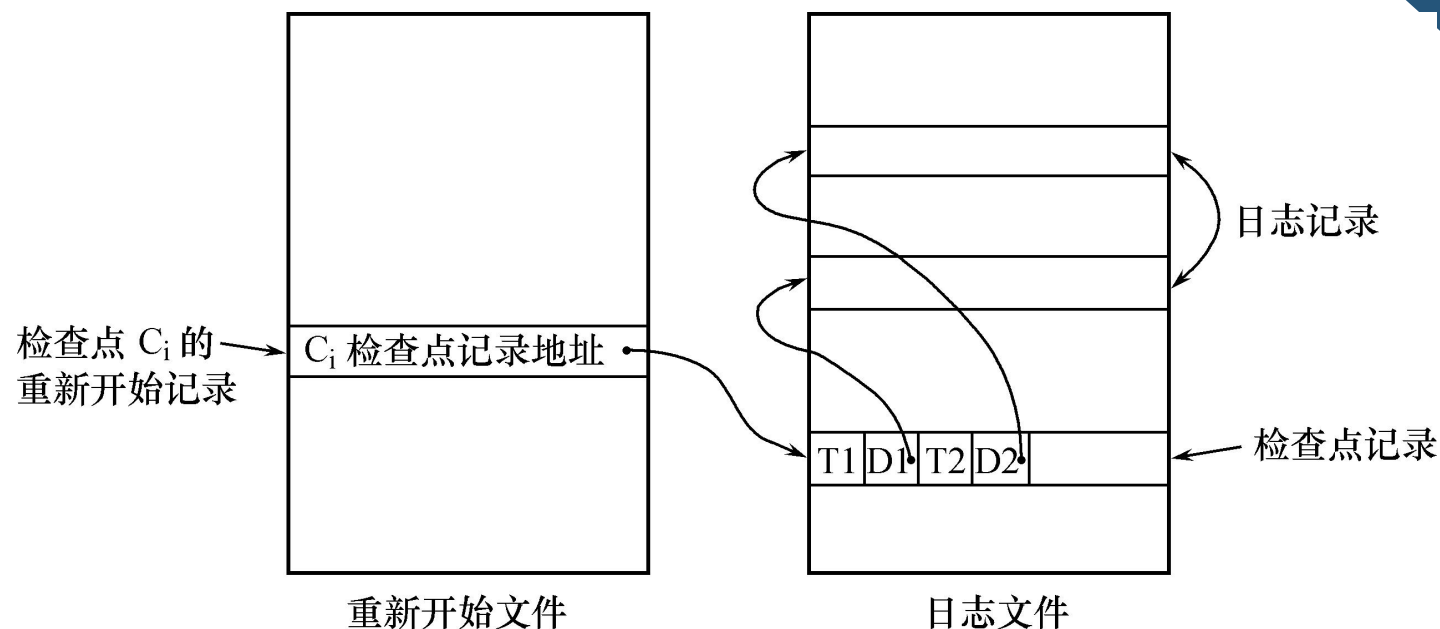


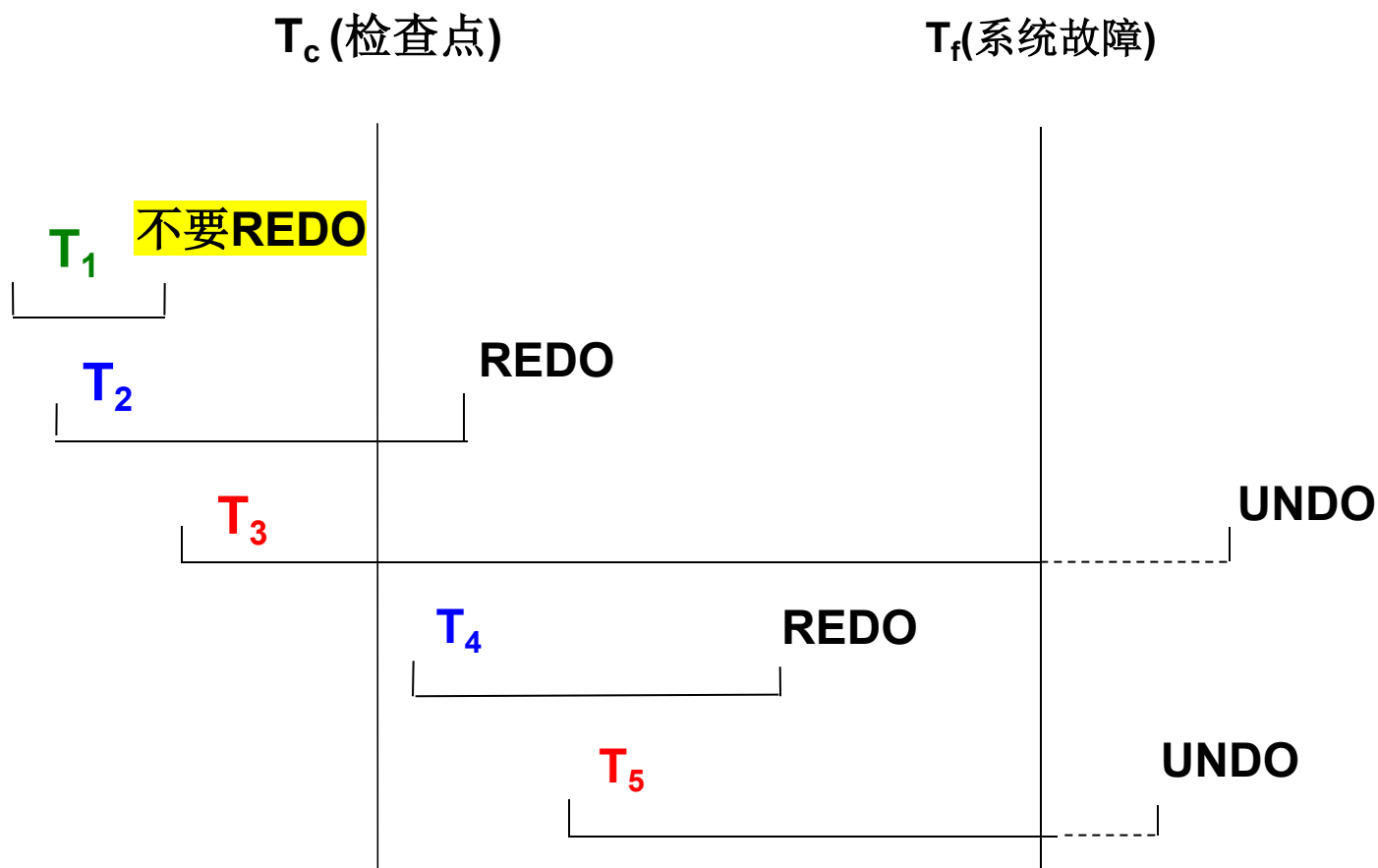
图 具有检查点的日志文件和重新开始文件

# CheckPoint技术

- 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REDO操作。

# 利用检查点的恢复策略（续）

- 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：

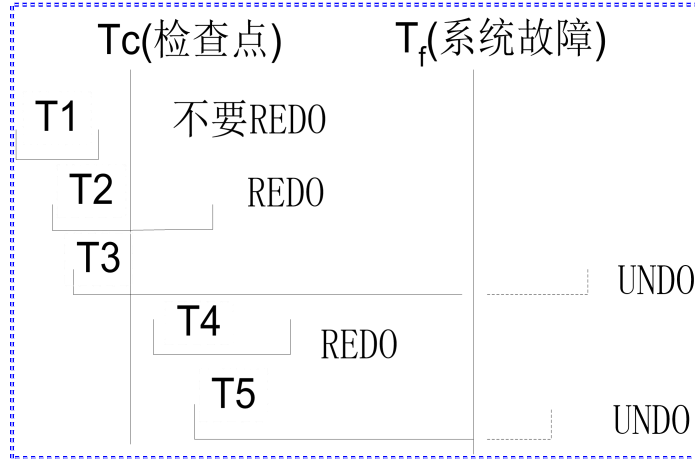


# Checkpoint技术

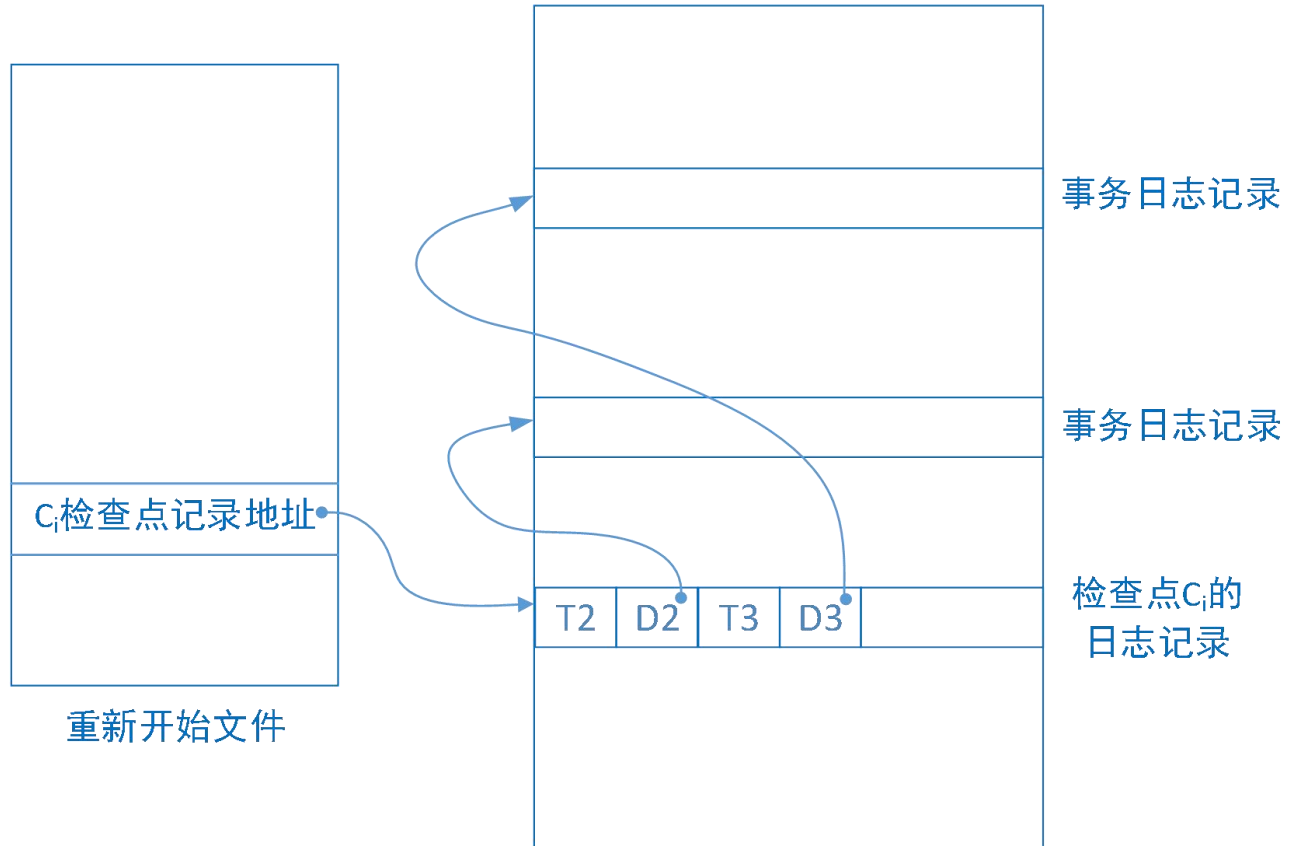
## □ 利用Checkpoint进行恢复的步骤:

1. 根据重新开始文件中最后一个检查点记录的地址，在日志文件中找到最后一个检查点记录；
2. 由检查点记录得到该检查点记录时刻的事务清单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 $T_i$ ，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；
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。

# CheckPoint技术



检查点C<sub>i</sub>的  
重新开始记录



# 例题

□ 现有系统故障发生时的日志序列：

$\langle START\ T1 \rangle - \langle START\ T2 \rangle - \langle T2,\ A,\ 3,\ 5 \rangle - \langle T1,\ B,\ 10,\ 30 \rangle$   
 $- \langle START\ T3 \rangle - \langle T3,\ C,\ 2,\ 4 \rangle - \langle COMMIT\ T1 \rangle - \langle CHECKPOINT \rangle$   
 $- \langle START\ T4 \rangle - \langle T4,\ B,\ 30,\ 50 \rangle - \langle COMMIT\ T2 \rangle - \langle T3,\ A,\ 5,\ 7 \rangle$   
 $- \langle ABORT\ T3 \rangle - \langle T4,\ A,\ 5,\ 15 \rangle。$

□ 请写出该检查点的活动事务集？

活动事务集为{T2, T3}

□ 叙述DBMS使用此日志序列进行系统故障恢复的过程，写出数据A, B, C在恢复后的值。

过程如前页。

Undo\_List={T3,T4}, Redo\_List={T2}

恢复后：A=5, B=30, C=2。

T1	T2	T3	T4
START T1			
	START T2 <T2, A, 3, 5>		
<T1, B, 10, 30>			
		START T3 <T3, C, 2, 4>	
COMMIT T1			
			START T4 <T4, B, 30, 50>
	COMMIT T2		
		<T3, A, 5, 7> ABORT T3	
			<T4, A, 5, 15>



# CheckPoint技术



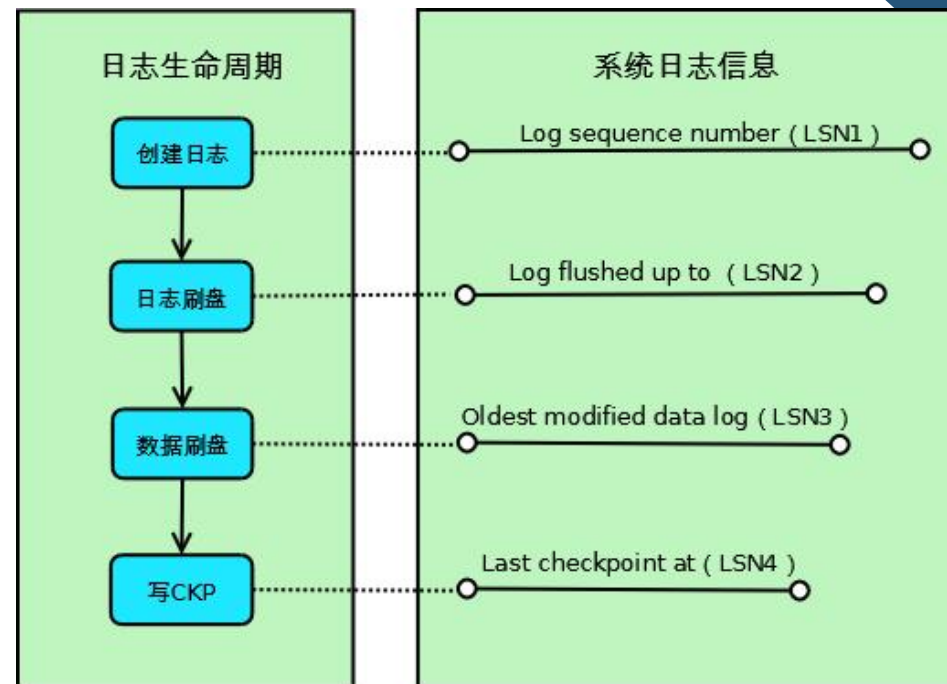
□ CheckPoint类型：完全检查点、模糊检查点

■ **完全检查点**：将所有脏页都刷新回磁盘。

□ 检查点进行同步过程中需要暂停对缓冲区页面的更新（可通过门锁实现），会导致系统的停顿。如果缓冲区页面数量很大，则完全检查点的时间会很长，导致对事务处理的中断影响较大。

□ 引入概念：

**脏页面列表**（Dirty Page Table, DPT）：缓冲区中所有未提交事务修改过的页面列表，每个脏页面中有信息箱记录了最开始导致该页面为脏的事务日志记录的LSN（recLSN）。



$LSN1 \geq LSN2 \geq LSN3 \geq LSN4$

# CheckPoint技术

## □ 完全检查点改进思路：

在检查点记录写入日志后、在修改过的缓存页写到磁盘之前，允许当前事务更新数据。

## □ 带来的问题：检查点日志后，缓存页内容可能由于写之前系统崩溃而缺失。 检查点本身内容的完整性问题。

## □ 基本改进策略：

(1) 将最后一个检查点记录在日志中的位置存在磁盘上专用的位置  
last\_checkpoint;

(2) 在写检查点记录之前，创建所有修改过的缓冲页列表，仅当该列表中的所有缓冲页都输出到磁盘后，才更新磁盘上的last\_checkpoint信息。

# CheckPoint技术

- **模糊检查点**：即只刷新一部分脏页，而不是刷新所有的脏页回磁盘。  
在日志中增加新的日志记录来描述检查点的边界：
- **<CHECKPOINT-BEGIN>**：记录检查点的开始；
- **<CHECKPOINT-END>**：记录检查点的完整过程结束，其中包含活跃事务列表（ATT）和脏页表（DPT）。
  
- 例如：MySQL fuzzy CheckPoint:
  1. Master Thread Checkpoint;
  2. FLUSH\_LRU\_LIST Checkpoint;
  3. Async/Sync Flush Checkpoint;
  4. Dirty Page too much Checkpoint1

# 转储 (dump) 实现技术

□ **归档转储 (archival dump)**，可用于查看数据库的旧状态。

典型实现方法：在转储过程中不允许有活跃事务，执行过程类似检查点。

- (1) 日志缓存内容写出到磁盘；
- (2) 数据缓存页写出到磁盘；
- (3) 将数据库的内容拷贝到转储介质；
- (4) 将日志记录拷贝到转储介质。

其中第1、2、4步类似检查点动作。

类似于模糊检查点策略，可以有模糊转储机制，允许转储过程中事务仍然是活跃的。

□ **SQL转储 (SQL dump)**，将SQL DDL和SQL insert语句写到文件中，可基于这样的文件重建数据库。在移植数据库（例如版本更新）时，数据库的物理位置、布局可能变化，此时SQL转储比较实用。

# DBMS的启动意味着什么？

- 1) 先检查各个外存文件是否完好、正常，若发现问题，则需要人工进行介质故障恢复；
- 2) 查看联机日志，是否存在未提交事务，如存在则进行类似于系统故障恢复的处理。

□ 好处：增强可靠性（不能保证DBMS每次都是正常SHUTDOWN）。

## 10.7 数据库镜像

### 1. 原因

**介质故障**：中断运行，周期备份，恢复麻烦。

### 2. 方法

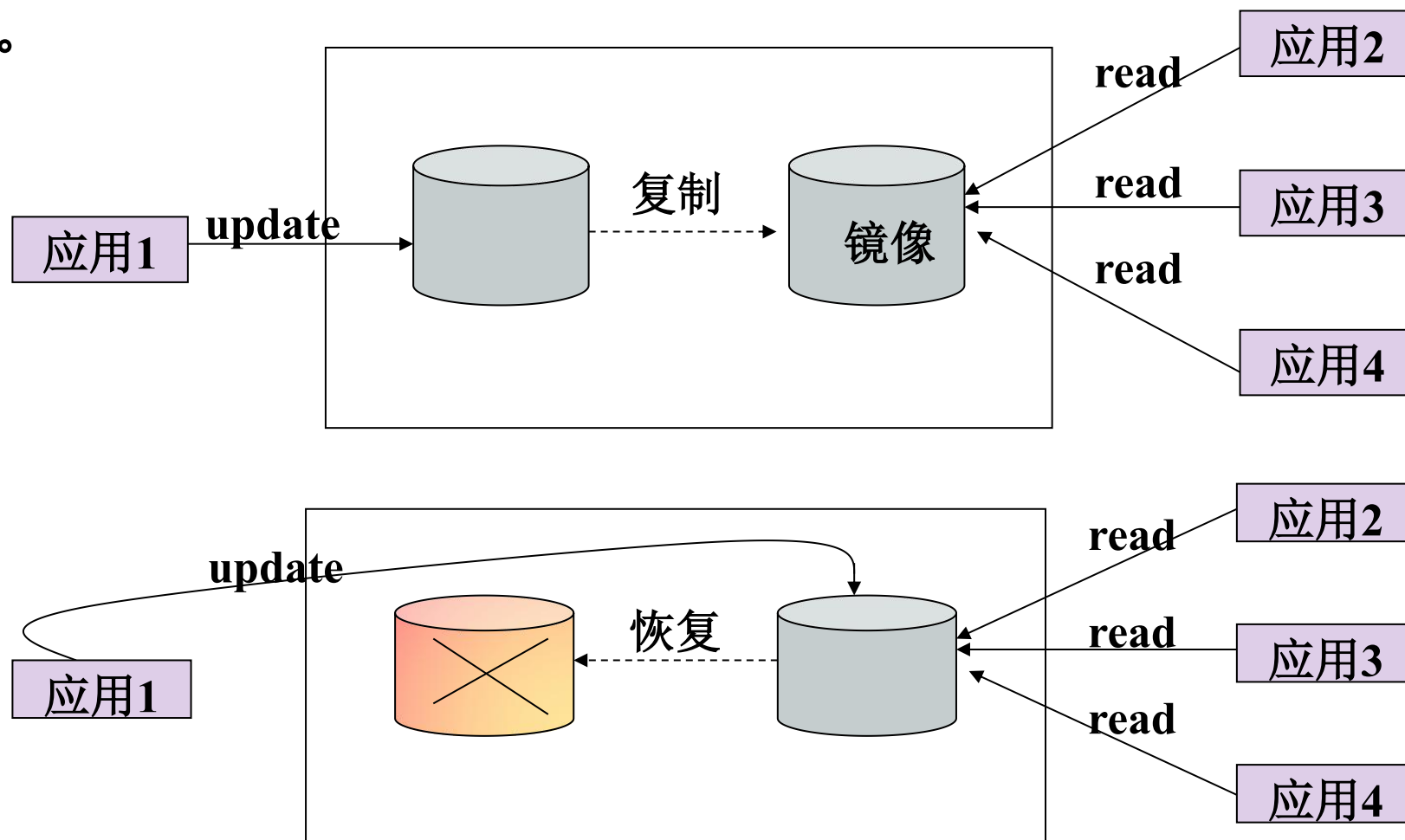
利用**自动复制技术**，**数据库镜像 (Mirror)**

### 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自动保证镜像数据与主数据库的一致性，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。

# 镜像

- 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。



# 数据库镜像

## 4. 优点

- 1) 无需关闭系统
- 2) 无需重装付本
- 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。



# 小结

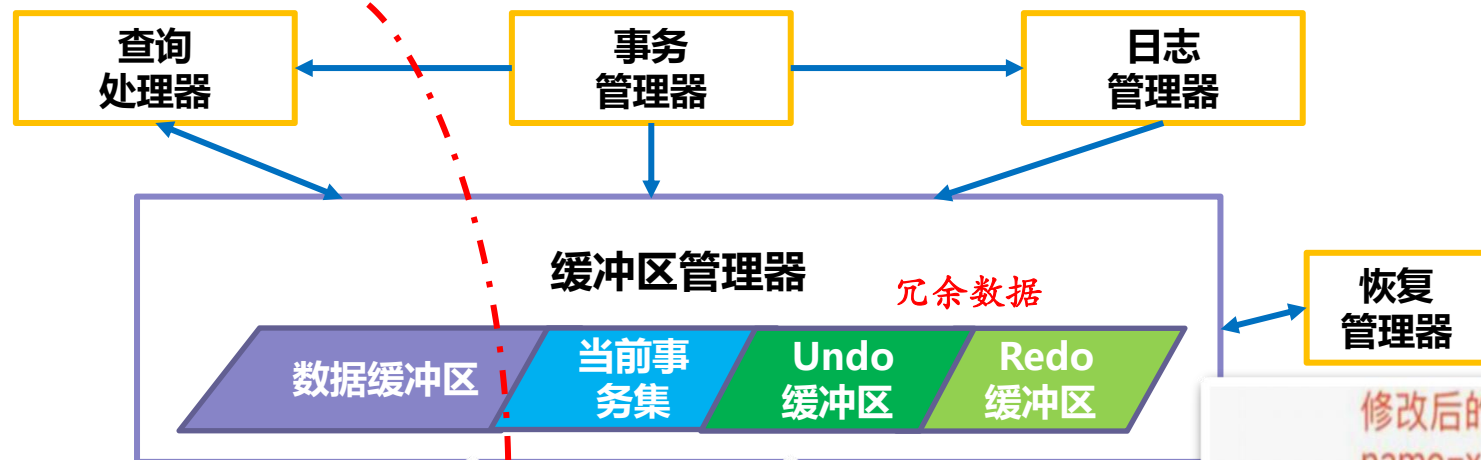
- 如果数据库只包含成功事务提交的结果，就说数据库处于一致性状态。 **保证数据一致性**是对数据库的最基本的要求。
- **事务是数据库的逻辑工作单位**：  
DBMS保证系统中一切事务的**原子性、一致性、隔离性和持续性**。
- DBMS必须对**事务故障、系统故障和介质故障**进行恢复。
- 恢复中最经常使用的技术：**数据库转储和登记日志文件**。
- 恢复的基本原理：利用存储在后备副本、日志文件和数据库镜像中的**冗余**数据来重建数据库。

**本章作业 P330 4, 5, 9**

# 第十章 数据库恢复技术 总结

## 数据库的恢复:

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)。 ACID

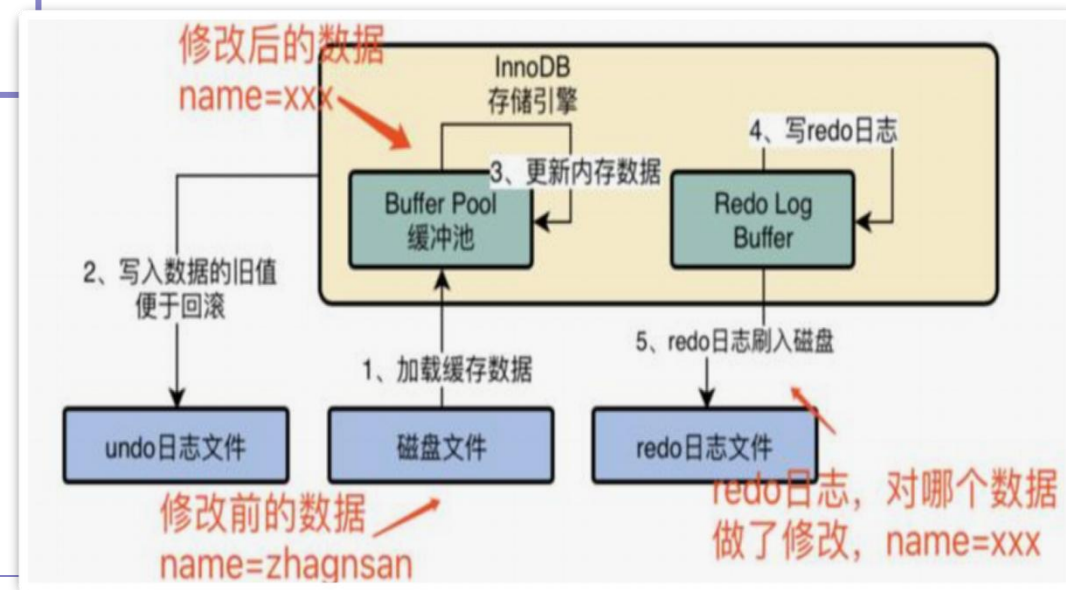


数据读写

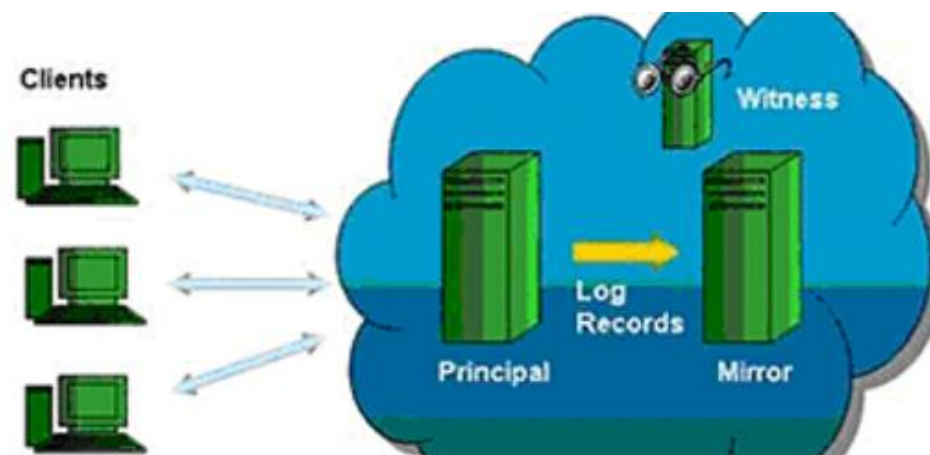
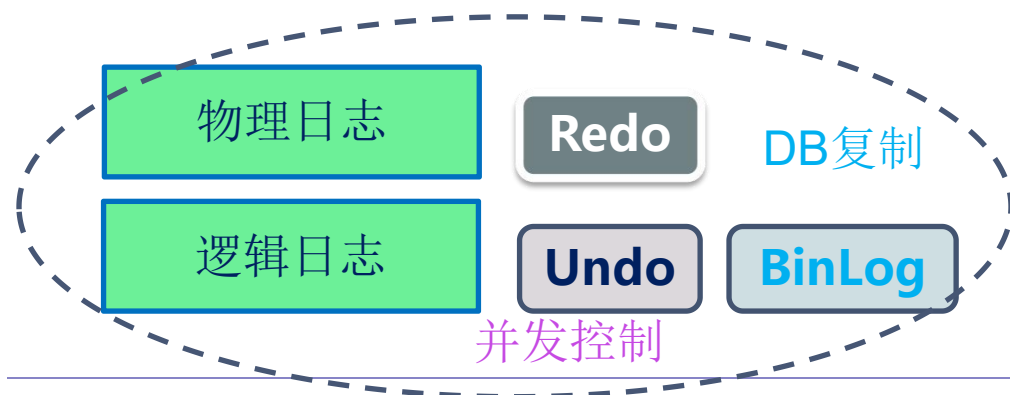
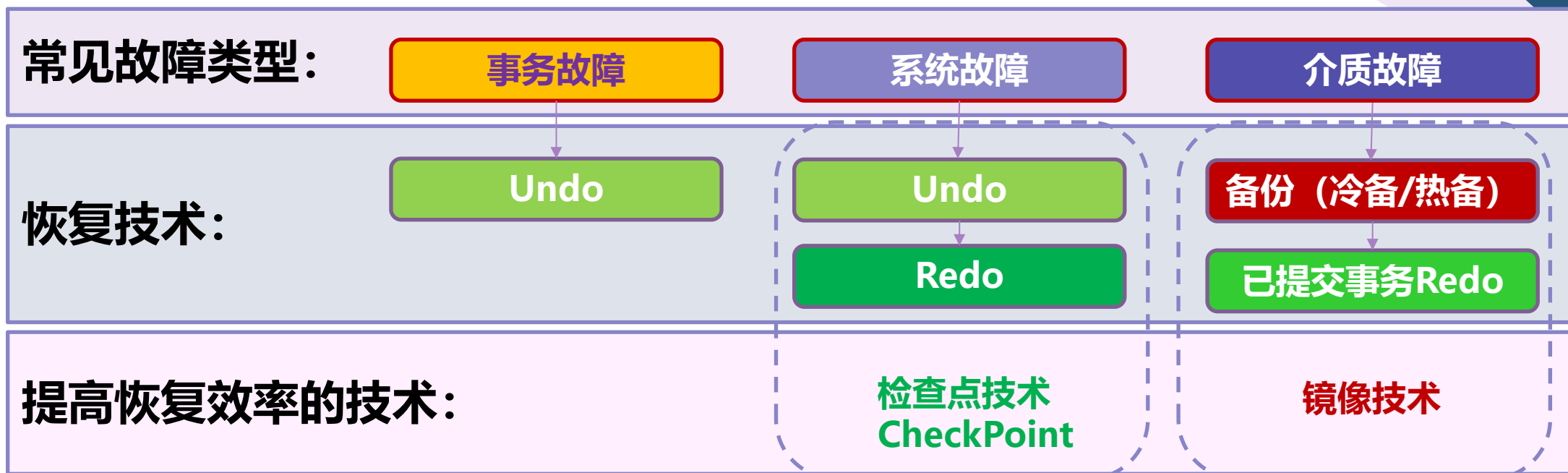
日志读写



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。



# 总结 (续)



# 课堂练习

□ **Q1:** 请简述记录日志的方式的2大原则。

严格时序；日志先写盘，数据后写盘

□ **Q2:** 为什么事务**非**正常结束时会影响数据库数据的正确性？请举例说明。

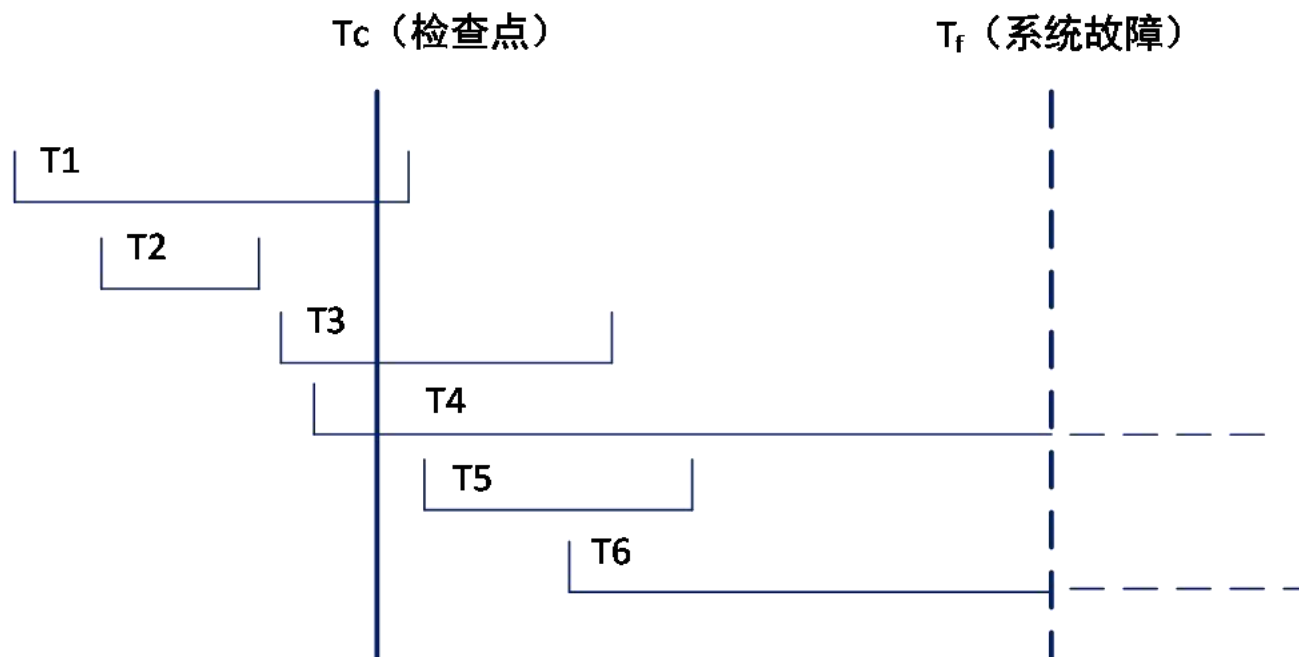
**事务一致性问题**。若事务执行过程中，系统运行发生故障，导致某事务部分执行被迫中断，未完成事务对DB的修改一部分已经写入磁盘，则数据库处于不正确状态，出现DB非一致性状态。  
例子：转账事务。

事务写内存完成但未写回时故障，内存与存储的数据库不一致。

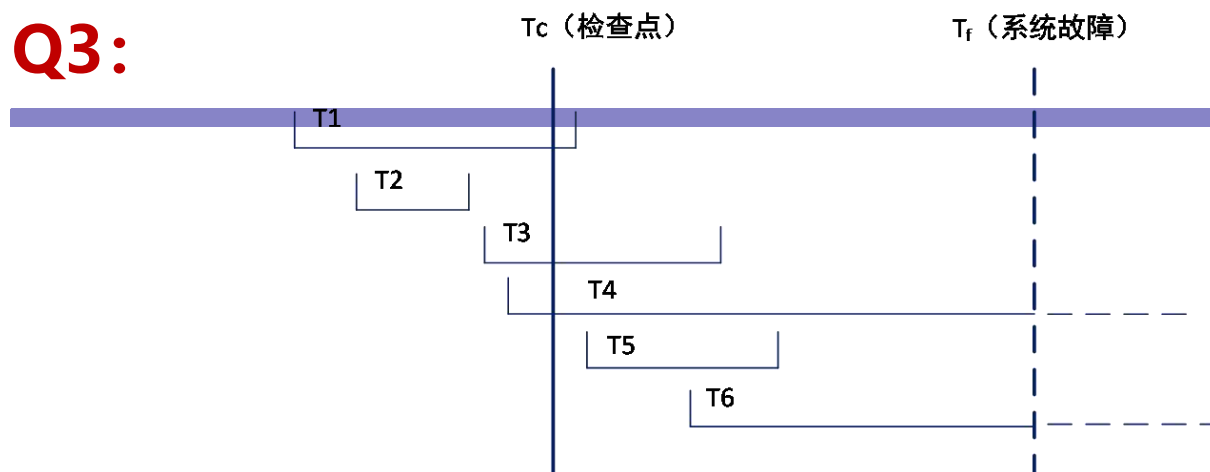
?

# 课堂作业

- Q3: 设有系统故障时，恢复子系统基于日志得出如下图所示数据库运行过程，请简述相应的恢复动作，要求说出检查点记录中的事务集合、两种恢复操作队列的**变化过程**。



Q3:



- 首先读取重新开始文件，获得最后一条检查点记录信息，并从日志文件中获取该检查点记录；（2分）
- 得到检查点时刻活动事务集{T1, T3, T4}，建立UNDO和REDO两个队列，将{T1, T3, T4}暂时放入UNDO队列，REDO队列为空；（2分）
- 自检查点记录后继续正向扫描日志文件，遇事务T1提交，UNDO事务队列变为{T3, T4}，REDO队列变为{T1}；（2分）
- 再遇事务T5, T6开始，UNDO事务队列变为{T3, T4, T5, T6}，REDO事务队列不变{T1}；（1分）
- 再遇事务T3, T5提交，UNDO事务队列变为{T4, T6}，REDO事务队列变{T1, T3, T5}；（1分）
- 反向扫描日志文件对UNDO队列中的事务执行UNDO操作；正向扫描日志文件对REDO队列中的事务执行REDO操作。（2分）