故障恢复

PART 01
数据库
恢复概述

PART 02
日志

PART 03
事务的撤销
与重做

PART 04
检查点

PART 05
数据转储

PART 06
恢复处理

1. 数据库恢复概述

- 数据库系统对付故障的两种措施

  尽可能提高系统的可靠性
  在系统发生故障后，把数据库恢复到一致状态

- 恢复机制涉及两个关键问题

  如何建立冗余数据
  如何利用冗余数据实施数据库恢复

- 恢复技术是衡量数据库管理系统优劣的重要指标

- 事务故障

逻辑错误：事务由于内部条件（如非法输入、溢出等）无法继续正常执行。

系统错误：系统进入一种不良状态（如死锁），事务无法继续正常执行。

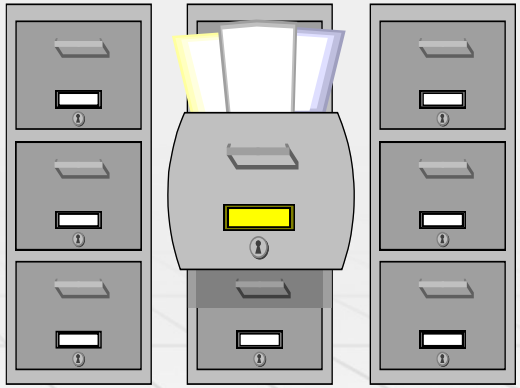
- 系统故障

包括硬件故障、数据库软件或操作系统的漏洞造成的系统停止运转。

- 介质故障

在数据传送操作过程中由于磁头损坏或故障造成磁盘块上的内容丢失。

- 记录日志文件
- 数据转储



- 日志文件

是DBMS用来记录事务对数据库的更新操作的文件，是日志记录的序列。

- 日志记录描述内容

事务标识符:执行写操作事务的唯一标识符
数据项标识符:事务操作对象的唯一标识符
前像（BI）:更新前数据的旧值
后像（AI）:更新后数据的新值

- 日志记录形式

<T START>：表示事务T已开始
<T COMMIT>：表示事务T已提交
<T ABORT>：表示事务T不能成功完成，已中止
<T,X,V1,V2>：表示事务T对数据项X执行写操作。写之前的旧值为V1，写之后的新值为V2

- 日志要求
  - 每次事务执行写操作，必须在数据库修改前建立此次写操作的日志记录；
  - 日志必须存储在稳定存储器上；
  - 稳定存储器中的日志记录顺序必须与写入日志缓冲区中的日志记录顺序完全一样（块写）。

先写日志规则

1. 在日志记录【Ti commit】写入磁盘之后，才允许事务Ti进入提交状态(写入磁盘)；

2. 在日志记录【Ti commit】写入磁盘之前，要保证commit之前的日志记录已经写入磁盘；

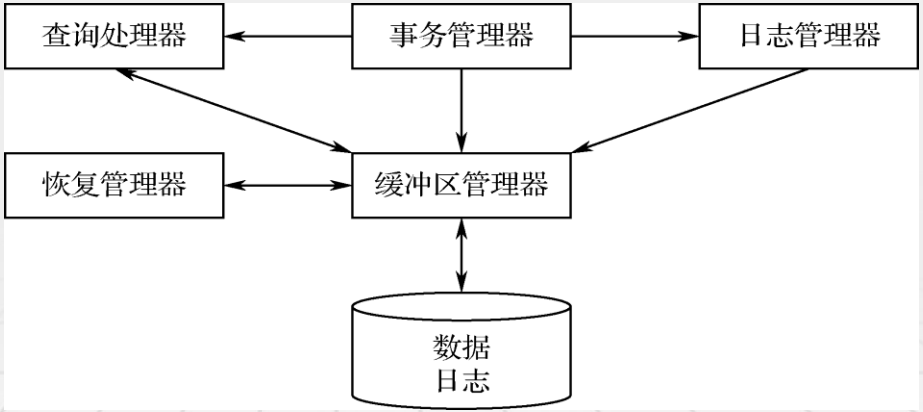
3. 主存中的数据块写入磁盘之前，所有与该数据块相关的日志记录必须已写入磁盘。

日志强制

缓冲的日志写入磁盘。

步骤	动作	日志
1		<T1 START>
2	READ(A)	
3	A:=A+A*0.1	
4	Write（A）	<T1,A,20,22>
5	READ(B)	
6	B:=B-B*0.05	
7	Write(B)	<T1,B,30,28.5>
8	写A到磁盘	
9		<T1,COMMIT>
10	写**B**到磁盘	

- 事务管理器工作包括：将关于事务动作的消息传给日志管理器，使动作信息能以“日志记录”的形式存储；将何时进行I/O操作的消息传给缓冲区管理器；传送消息给查询处理器使之能执行查询及其他数据库操作。
- 日志管理器维护日志。日志最初存放在主存缓冲区中，在一定的时刻缓冲区管理器将存放信息复制到磁盘上。当系统崩溃时恢复管理器被激活，它检查日志并在必要时利用日志恢复数据。



为了维持正确性原则，要求：

- 事务是原子的，即事务必须作为整体执行或根本不执行，如果仅有事务的部分被执行，那么很有可能产生数据库不一致性状态；
- 事务同时执行时进行并发控制，避免可能导致的状态不一致。

- 后像在事务提交后才写入数据库

通过在日志中记录所有的对数据库的修改，将一个事务的所有写操作延迟到事务的操作结束时才执行，以此保证事务的原子性。

- 事务T的日志写入步骤

1. 在T开始执行前，向日志中写入记录<T START>;

2. T的一次write（X）操作导致向日志中写入一条新记录;

3. 最后，当T全部操作结束，向日志中写入记录<T COMMIT> 。

- 事务恢复处理策略

1. 忽略未完成的事务；

2. 重复(Redo(Ti))已提交事务的影响：将事务Ti更新的所有数据项的值设为新值。

- 简化日志内容结构

日志记录<T，X，V1>：事务T对数据项X执行写操作，写入新值V1。

药品价格调整：设T1事务中，药品A上调10%，药品B下浮5%。

T1：
read(A)
A:=A+A*0.1
write(A)
read(B)
B:=B-B*0.05
Write(B)

A=20，B=30

步骤	动作	日志
1		<T1 START>
2	READ(A)	
3	A:=A+A*0.1	
4	Write（A）	<T1,A,22>
5	READ(B)	
6	B:=B-B*0.05	
7	Write(B)	<T1,B,28.5>
8		<T1,COMMIT>
9	写A到磁盘	
11	写**B**到磁盘	

恢复处理步骤

1. 从后向前扫描日志，将提交的事务放入队列redo-list；

2. 从前往后扫描日志。对遇到的每一<T,X,V1>记录:

   如果T不是redo-list中的事务，则什么也不做；

   如果T是redo-list中的事务，则为数据项X写入值V1。

3. 对每个未完成的事务，在日志中写入一个<T,ABORT>记录并刷新日志。

<T1 START> <T1,A,22> <T1,B,28.5> （ a）	<T1 START> <T1,A,22> <T1,B,28.5> <T1,COMMIT> (b)	<T1 START> <T1,A,22> <T1,B,28.5> <T1,COMMIT> 写A 到磁盘 （c ）	<T1 START> <T1,A,22> <T1,B,28.5> <T1,COMMIT> 写A 到磁盘 写B 到磁盘 （d ）

若<T1 COMMIT>

不在磁盘，处理同(a)

若<T1 COMMIT>

在磁盘，处理同(C)

<T1 ABORT>

REDO

再次REDO

- 后像在事务提交前完全写入数据库

在日志中记录所有的数据库修改，一个事务的所有写操作在事务提交前时已写入磁盘。

- 执行规则

日志先写：如果事务T改变了数据项X，则记录变化的日志记录必须在数据项的新值写到磁盘前写入稳定的存储器。

如果事务提交，则<T COMMIT>日志记录必须在事务改变的所有数据项的新值写入到磁盘后再写入稳定的存储器。

- 事务T的日志写入执行步骤

1. 在T开始执行前，向日志中写入记录<T START>;
2. T的一次write（X）操作导致向日志中写入一条新记录;
3. 最后，被改变的所有数据项已写入磁盘后向日志中写入记录<T, COMMIT>

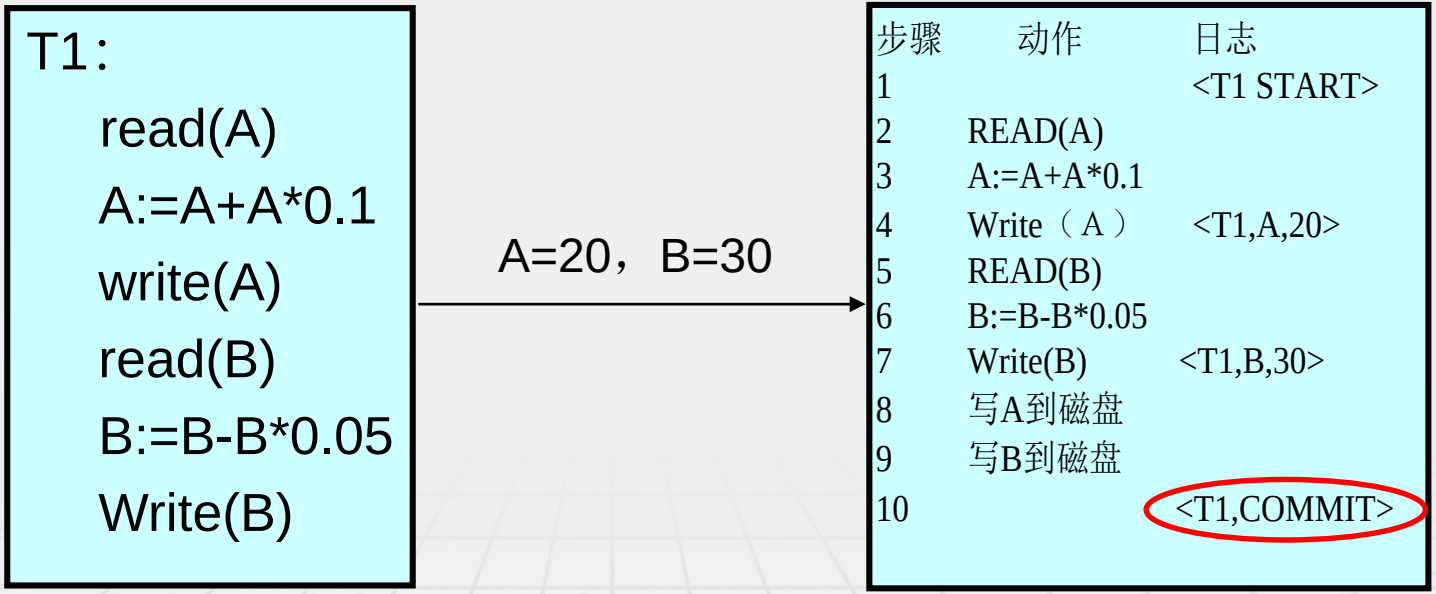
- 事务恢复处理策略

撤销(Undo（Ti）)未完成的事务:将事务Ti更新的所有数据项的值设为旧值。

- 简化日志内容结构

日志记录<T，X，V1>表示：事务T对数据项X执行写操作，写前的旧值为V1。

药品价格调整：设T1事务中，药品A上调10%，药品B下浮5%。



恢复处理步骤

1. 首先对日志文件从后向前进行扫描，将有<T，COMMIT>记录的事务放入redo-list队列；

2. 然后对日志文件从后向前进行扫描；

对遇到的每一个<T，X，V1>记录，若事务T在redo-list队列中，则恢复管理器什么都不做；

对遇到的每一个<T，X，V1>记录，若事务T不在redo-list队列中，则恢复管理器将数据项X在数据库中的值改为旧值V1；

3. 对每个未完成的事务，在日志中写入一个<T,ABORT>记录并刷新日志。

(a)	(b)	(c)	(d)
<T1,START> <T1,A,20> <T1,B,30> 写A到磁盘 写B到磁盘 <T1,COMMIT>	<T1,START> <T1,A,20> <T1,B,30> 写A到磁盘 写B到磁盘	<T1,START> <T1,A,20> <T1,B,30> 写A到磁盘	<T1,START> <T1,A,20> <T1,B,30>

若<T1 COMMIT>在磁盘，
NO ACTION　　UNDO　　UNDO　　UNDO

若<T1 COMMIT>不在磁盘，
UNDO

补<T1 ABORT>

- 后像在事务提交前后写入数据库

被修改数据项写入磁盘的时间可以在日志记录<T，COMMIT>之前进行，也可以放在之后进行。

- 执行规则

日志先写：被更新数据项写入磁盘前，更新记录<T，X，V1，V2>必须已写到稳定存储器上。

- 事务T的日志写入执行步骤

1. 在T开始执行前，向日志中写入记录<T START>；
2. T的一次write（X）操作导致向日志中写入一条新记录；
3. 最后，被改变的所有数据项已写入磁盘后向日志中写入记录<T, COMMIT>。

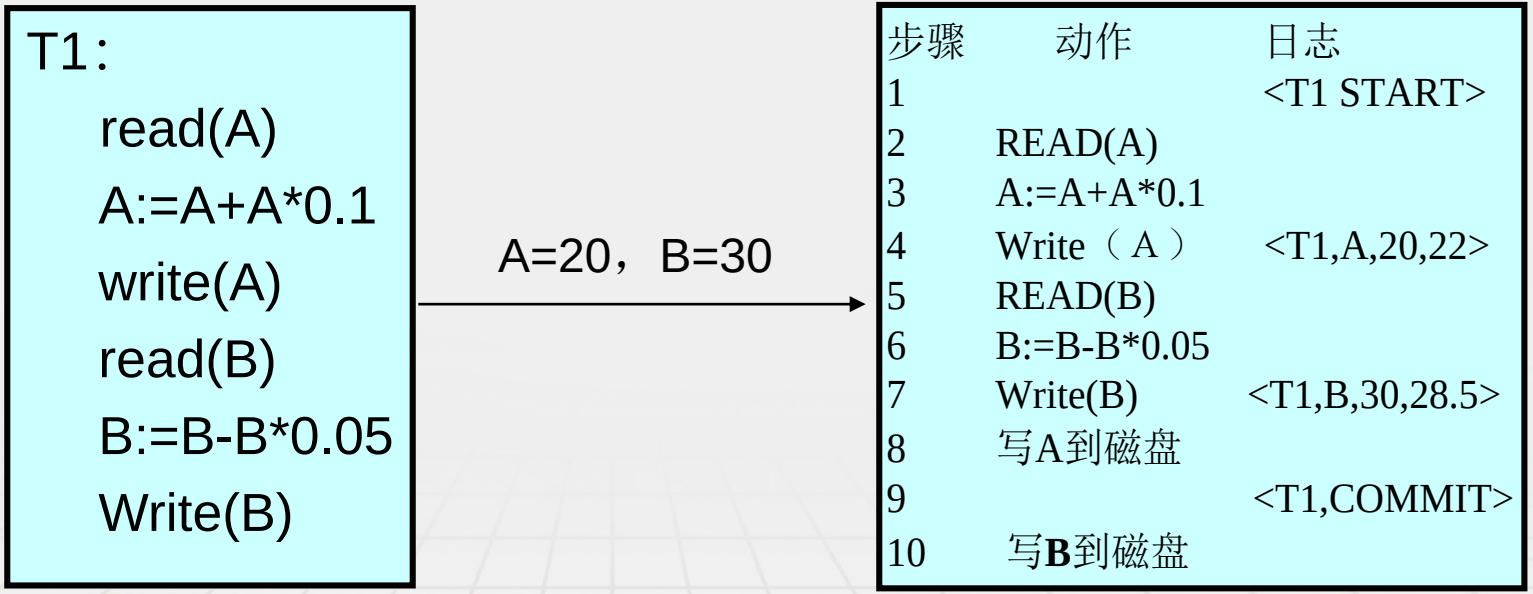
- 事务恢复处理策略

Undo（Ti）：将未提交事务Ti更新的所有数据项的值设为旧值。
Redo（Ti）：将已提交事务Ti更新的所有数据项的值设为新值。

- 日志内容结构

日志记录<T，X，V1，V2>表示：事务 T 对数据项 X 执行写操作，写前的旧值为V1，写后的新值为V2。

药品价格调整：设T1事务中，药品A上调10%，药品B下浮5%。



恢复处理步骤

1. 首先对日志文件从后向前进行扫描，将有<T，COMMIT>记录和没有<T，COMMIT>记录的事务分别放入两个队列：redo-list队列，undo-list队列；

2. 从前向后再次扫描日志记录，重新执行redo-list队列中的事务；

3. 从后向前再次扫描日志记录，撤销undo-list队列中的事务。

(a)	(b)	(c)
<T1,START〉 <T1,A,20,22> <T1,B,30,28.5>	<T1,START> <T1,A,20,22> <T1,B,30,28.5> 写A到磁盘	<T1,START> <T1,A,20,22> <T1,B,30,28.5> 写A到磁盘 <T1,COMMIT>
UNDO	UNDO	UNDO或REDO

附加规则：
<T，COMMIT>记录一旦出现在日志中就必须强制进行日志刷新。

- 问题的提出

搜索整个日志的困难

搜索整个日志将耗费大量的时间。

很多需要REDO处理的事务其更新已经写入数据库，重新执行这些操作会浪费大量的时间，使恢复过程变得更长。

解决办法

系统定期或不定期地建立检查点（CheckPoint），保存数据库的状态。

4. 检查点

- 创建检查点方法

- 提交一致性检查点
- 高速缓存一致性检查点
- 模糊一致性检查点

1. 新的事务不能开始直到检查点完成。

2. 现有的事务继续执行直到提交或中止。

4. 将日志记录<checkpoint>写入稳定存储器。检查点操作完成。

3. 将当前日志缓冲区中的日志记录写回稳定存储器中的日志文件。

< T1， START>
< T1， A， 200>
< T2， START>
< T2， B， 10>
< T1， C， 50>
< T1， COMMIT>
< T2， COMMIT>
< CHECKPOINT>
< T3， START>
< T3， A， 5>

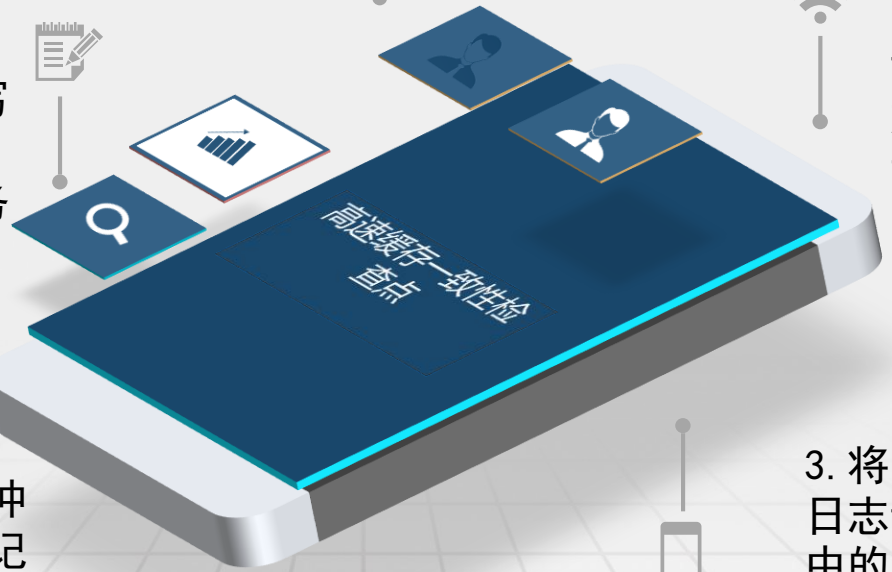
1. 新的事务不能开始直到检查点完成。

2. 已存在的事务不允许执行新的更新操作，如写缓冲块或写日志记录。

3. 将当前日志缓冲区中的日志记录写回稳定存储器中的日志文件。

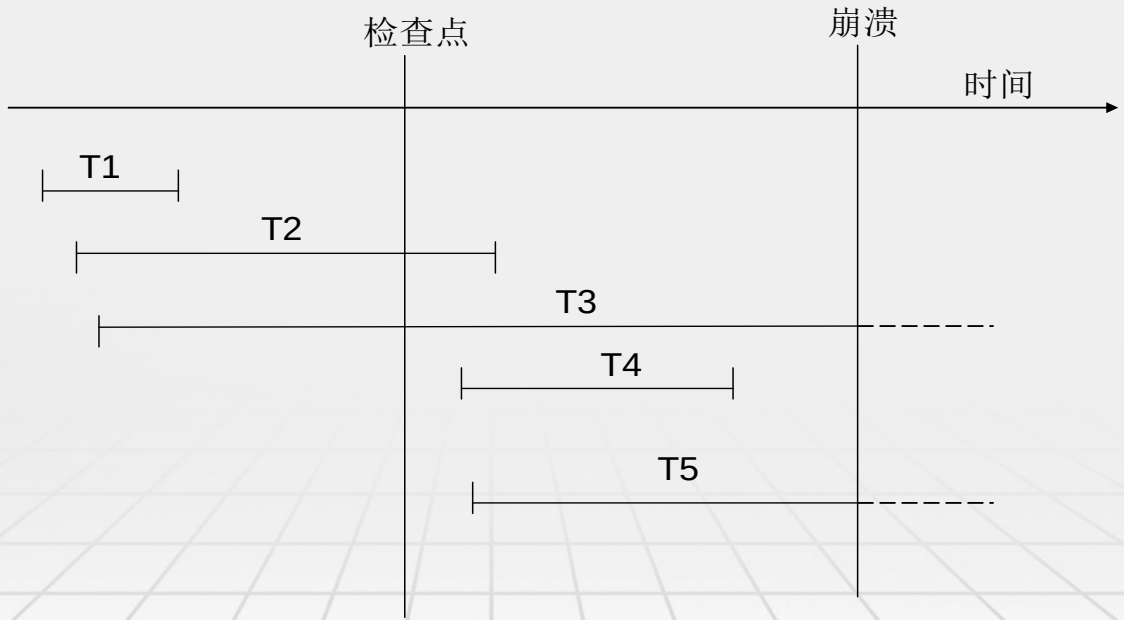
4. 将当前数据缓冲区中的所有数据记录写入磁盘。

5. 将日志记录<checkpoint，L>写入稳定存储器，其中L是所有活动事务的列表。



4. 检查点

1. T1事务在检查点建立时已经提交，该事务在恢复过程中不处理；
2. T2，T3事务在检查点建立时刻处于活动状态，恢复子系统首先将其放入活动事务列表L中。
3. T4，T5在崩溃之前，检查点建立之后开始执行，恢复子系统也将其放入待处理事务集合T中。
4. 对集合T和L中的所有事务Ti，若日志中没有<Ti，COMMIT>记录，则执行UNDO（Ti）；若日志中有<Ti，COMMIT>记录，则执行REDO（Ti）。



数据转储

DBA定期地将整个数据库复制到某种存储介质（如磁带、磁盘、光盘等）上保存起来的过程。

转储方法

静态转储：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、修改活动；
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；
完全转储：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。

- 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、修改活动
- 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

- 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
  - 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

- 完全转储：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
- 完全转储与增量转储比较
  - 从恢复角度看，使用完全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

回顾：数据库应用系统在运行过程中可能发生的故障分为三类：事务故障、系统故障、介质故障。

- 事务故障
  - 逻辑错误

    事务由于内部条件（如非法输入、溢出等）无法继续正常执行。
  - 系统错误

    系统进入一种不良状态（如死锁），事务无法继续正常执行。

- 恢复策略

利用日志文件撤销此事务对数据库已经进行做过的修改。

- 恢复处理

◆ 后像后写

恢复管理器忽略这些未完成的事务。

◆ 后像前写

使用日志文件撤销此事务对数据库的修改

◆ 后像前后写

使用日志文件撤销此事务对数据库的修改

系统故障

包括硬件故障、数据库软件或操作系统的漏洞造成的系统停止运转。

恢复策略

利用日志文件撤销未完成事务，重做已提交事务。

恢复处理

- 当系统崩溃重新启动时，它构造两个队列：undo-list存放需要撤销的事务标识符，redo-list存放需要重做得事务标识符。
- 这两个队列刚开始时都是空的。
- 队列构造步骤如下：
  - ✓ 系统反向扫描日志，直到发现第一个<checkpoint>记录。
  - ✓ 对每一个<Ti，COMMIT>记录，将Ti加入redo-list。
  - ✓ 对每一个<Ti，START>记录，如果Ti不属于redo-list，则将Ti加入undo-list。

- 系统故障的恢复：后像后写
  - 对undo-list中的事务，在日志中写入一个<Ti，ABORT>记录并刷新日志。
  - 对redo-list中的事务执行REDO操作：从前面发现的<checkpoint〉记录开始，正向扫描日志文件，对遇到的每一个<Ti，X，V1>记录，将数据库中的X数据项更新为新值V1。
- 系统故障的恢复：后像前写
  - 对undo-list中的某一事务，执行UNDO操作：
  - 再次反向扫描日志文件，对遇到的每一个<Ti，X，V1>记录，将数据库中的X数据项更新为旧值V1，扫描到<Ti，START>记录时，扫描停止。
  - 在日志中写入一个<Ti，ABORT>记录并刷新日志。
  - 重复上两个步骤，直到处理完撤销队列中的每一个事务。

- 系统故障的恢复：后像前后写
  - 系统重新反向扫描日志文件，对undo-list中的每一事务执行UNDO操作，即对遇到的每一个<Ti，X，V1，V2>记录，将数据库中的X数据项更新为旧值V1。
  - 在日志中写入一个<Ti，ABORT>记录并刷新日志。当undo-list中所有事务Ti所对应的<Ti，START>记录都找到时，扫描结束。
  - 系统找出日志中最后一条<checkpoint>记录。
  - 系统由最后一条<checkpoint>记录开始，正向扫描日志文件，对redo-list中的事务Ti的每一个日志记录执行redo操作。即对遇到的每一个<Ti，X，V1，V2>记录，将数据库中的X数据项更新为新值V2。

介质故障

在数据传送操作过程中由于磁头损坏或故障造成磁盘块上的内容丢失。

恢复处理

1. 如果有后续的增量转储，按照从前往后的顺序，根据增量转储来修改数据库。
2. 装入最近的完全转储后备副本，若数据库副本是动态转储的，还需要同时装入转储开始时刻的日志文件副本，利用恢复系统故障的方法将数据库恢复到某个一致性状态。
3. 装入转储结束后的日志文件副本，重做已完成的事务。首先反向扫描日志文件，找出故障发生时已经提交的事务，将事务标识符写入redo-list。然后正向扫描日志文件，对redo-list中的所有事务进行redo操作。