数据库系统原理

Database System Principles

四川大学计算机学院

段 磊

leiduan@scu.edu.cn

2014.9

第十章 数据库恢复技术

- 事务处理技术主要包括数据库恢复技术和并发控制技术
- 数据库恢复机制和并发控制机制是数据库管理系统的重要组成部分

本章目录

10.1 事务的基本概念

- 数据库恢复机制是数据库管理系统的重要组成部分。数据库恢复是基于事务的，所以必须先弄清楚事务的概念。
- 事务(最早为了解决多个用户同时使用数据库的并发问题而引入)

  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。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。

事务的基本概念

事务的开始与结束可以由用户显式控制，或由DBMS按缺省自动划分。在SQL语言中，定义事务的语句有三条：

- BEGIN TRANSACTION
- COMMIT
- ROLLBACK

事务的基本概念

- 事务的特性(**ACID**)
  - 原子性(Atomicity) 最小工作单位
  - 一致性(Consistency) 一个正确状态到另一个正确状态
  - 隔离性(Isolation) 事务相互不干扰
  - 持续性(Durability) 事务提交对数据库影响是永久性的
- 事务是恢复和并发控制的基本单位。
- 保证事务ACID特性是事务处理的重要任务。

本章目录

10.2 数据库恢复概述

- 尽管数据库系统中采取了各种保护措施来防止数据库的安全性和完整性被破坏，保证并发事务的正确执行，但是计算机系统中硬件的故障、软件的错误、操作员的失误以及恶意的破坏仍是不可避免的，这些故障轻则造成运行事务非正常中断，影响数据库中数据的正确性，重则破坏数据库，使数据库中全部或部分数据丢失。

数据库恢复概述

- 因此数据库管理系统必须具有把数据库**从错误状态恢复到某一已知的正确状态**的功能，这就是数据库的恢复。
- 恢复子系统是数据库管理系统的一个重要组成部分，恢复技术是否行之有效，是衡量系统性能优劣的重要指标。

本章目录

10.3 故障的种类

- 根据不同故障的不同特点，采用不同的恢复策略。

一. 事务内部的故障 (UNDO)

- 事务故障意味着事务没有达到预期的终点（COMMIT或ROLLBACK），从而数据库可能处于不正确状态。
- 此时必须强行回滚（ROLLBACK）该事务。这类恢复操作称为事务撤消（UNDO）。

故障的种类

二. 系统故障 (UNDO + REDO)

- 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。
  - 如硬件故障、操作系统故障、DBMS代码错误、突然停电等。
- 系统重新启动时，恢复子系统需要撤消（UNDO）所有未完成（非正常终止）的事务，并重做（REDO）所有已提交的事务。

故障的种类

三. 介质故障

- 系统故障称为软故障，介质故障称为硬故障。
- 硬故障指外存故障，如磁盘损坏等。
- 这类故障将破坏数据库或部分数据库。
- 这类故障发生的可能性很小，但**破坏性**最大。

故障的种类

四. 计算机病毒或恶意攻击

- 这是一种人为的故障或破坏。

本章目录

10.4 恢复的实现技术

- 恢复的原理 ——冗余
- 恢复机制涉及两个关键问题：
  （1）如何建立冗余数据
  （2）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是数据转储（DBA定期进行）和登录日志文件（记录事务对数据库的更新操作）。通常在一个数据库系统中，这两种方法是一起使用的。

10.4.1 数据转储

- 静态转储　转储时系统无事务运行。
- 动态转储　转储时系统有事务运行。

- 海量转储　转储全部数据。
- 增量转储　仅转储更新数据。

10.4.2 登记日志文件

一. 日志文件的格式和内容

- 以记录为单位的
  - 示例

$< T_0$ **commit**$>$

$< T_1$ **start**$>$

$< T_1$, C, 700, 600$>$

- 以数据块为单位的

登记日志文件

二. 日志文件的作用

1. 事务故障和系统故障的恢复
2. 动态转储时必须建立日志文件
3. 静态转储时，利用日志文件可以REDO

登记日志文件

三. 登记日志文件

1. 登记严格按并发事务执行的时间顺序
2. 先写日志文件，再写数据库

想想为什么要先写日志文件？

本章目录

10.5 恢复策略

- 总原则
  - 当系统运行过程中发生故障，利用数据库后备副本和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。

(1) 利用数据库（转储）后备副本重装数据库。

(2) 通过搜索（正向或反向扫描）日志文件，确定并对事务进行REDO或UNDO处理。

  - 不同故障其恢复策略和方法也不一样。

10.5.1 事务故障的恢复

UNDO　系统自动完成

1. 反向扫描日志文件，查找未完成事务的更新操作。
2. 对该事务的更新操作做逆操作。
3. 继续反向扫描，查找该事务的其他更新操作，并做同样处理。
4. 直至该事务的开始标记为止。

10.5.2 系统故障的恢复

UNDO＋REDO　系统自动进行

1. 正向扫描日志文件，将已提交事务(故障发生前已COMMIT)标志记入REDO队列，未完成事务，将其事务标志记入UNDO队列。
2. 对UNDO队列中的事务进行UNDO处理。
3. 对REDO队列中的事务进行REDO处理。

10.5.3 介质故障的恢复

后备副本＋REDO

需要DBA的介入。

1. 装入最新的数据库后备副本。使数据库恢复到故障发生前最近转储时的一致性状态。
2. 装入日志文件副本，利用日志文件进行REDO。

本章目录

10.6 具有检查点的恢复技术

- 不带检查点技术的恢复技术的一大缺陷：系统重新启动时需要大量REDO实际上没有必要做
- 检查点技术的思想
  - 在没有故障时增加检查点记录，表明此时没有故障
    - <checkpoint L> (L是检查点时刻正在执行的事务列表)
  - 故障发生后，在最后一次检查点前已经提交的事务不需要再REDO了。

具有检查点的恢复技术

- 建立检查点的步骤
  - 将当前日志缓冲区中的所有日志记录写入磁盘上的日志文件
  - 将当前数据缓冲区的所有被修改过的数据记录写入磁盘的数据库
  - 在日志文件中写入一个检查点记录

具有检查点的恢复技术

- 有检查点的系统恢复策略
  - 反向扫描日志文件
  - 如果发现<T_i **commit**>，将T_i记入REDO队列
  - 如果发现<T_i **begin**>，而T_i不在REDO队列中，将T_i记入UNDO队列
  - 一直找到最近的一个<*checkpoint L*>记录，如果L中的事务不在REDO队列中则将其记入UNDO队列
  - UNDO队列中的事务执行UNDO处理；对REDO队列中的事务执行REDO处理

示例：

<T_0 **begin**>

<T_0, A, 0, 10>

<T_0 **commit**>

<T_1 **begin**>

<T_1, B, 0, 10>

<T_2 **begin**>

<T_2, C, 0, 10>

<T_2, C, 10, 20>

<**checkpoint** {T_1, T_2}>

<T_3 **start**>

<T_3, A, 10, 20>

<T_3, D, 0, 10>

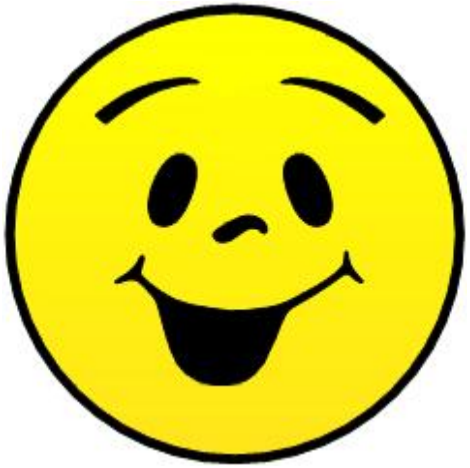
<T_3 **commit**>

本章目录

10.7 数据库镜像

- 自动地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的一致性。
- 一旦出现介质故障，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。
- 在没有出现故障时，数据库镜像还可以用于并发操作。

Any Question?



Thank you !