Lecture 12: Transactions

Database

Author: Forliage

Email: masterforliage@gmail.com

Date: June 7, 2025

College: 计算机科学与技术学院

Abstract

本讲笔记围绕关系型数据库的事务（Transaction）机制展开，共分为以下九个核心部分：首先，“Transaction定义”介绍了事务的概念及其在保证并发与故障容忍中的重要作用，并阐述了事务的 ACID 属性；接着，“Transaction状态”说明了事务从活跃、部分提交、失败到中止与已提交的生命周期；第三部分“原子性和持久性的实现”主要讲解了恢复管理组件中的影子数据库方案及其工作流程；“并发执行”一节分析了并行运行多个事务的优势与潜在一致性破坏问题；“可串行化”章节细分为冲突指令与冲突可串行性概念，论述了如何判断并发调度与串行调度的等价性；“可恢复性”部分则讨论了可恢复调度、级联回滚与无级联调度的区别；“隔离性的实现”论述了并发控制方案在保证隔离级别上的权衡；“SQL中的事务定义”简要演示了通过提交和回滚语句在 SQL 中管理事务的方式；最后，“可串行性测试”介绍了构建优先图并进行环检测以验证调度的冲突可串行化。通过本讲笔记的学习，读者将系统掌握事务管理的基本原理、实现技术与测试方法，为后续深入研究并发控制与故障恢复打下坚实基础。
（该Abstract由ChatGPT-o4-mini-high生成）

Contents

1 Transaction定义

1.1 概念

对于数据库管理系统，必须处理两个主要问题：

- 多个用户或多程序的并发执行
- 各种故障，如硬件故障和系统崩溃

数据库并发执行时，如何保持其正确性、一致性和完整性？

事务是程序执行的一个单元，它会访问并可能更新各自数据项：

- 通常一个事务包含多条SQL语句，以提交或回滚语句结束
- 事务必须看到一个一致的数据库。

在事务执行期间，数据库可能处于不一致状态，但当事务提交时，数据库必须保持一致。

ACID属性

事务是程序执行的一个单元，它会访问并可能更新各自数据项。为了保持数据的完整性，数据库系统必须确保：

- Atomicity:原子性。事务的所有操作要么都正确地反映在数据库中，要么都不反映。
- Consistency:一致性。事物的隔离执行能保持数据库的一致性。
- Isolation:隔离性。虽然多个事务可以并发执行，但每个事务必须不知道其它并发执行的事务。事物的中间结果必须对其它并发执行的事务隐藏。也就是说，对于每对事务T_i和T_j，对T_i而言，要么T_j在T_i开始之前就已完成执行，要么T_j在T_i完成之后才开始执行。
- Durability:持久性。事务成功完成后，它对数据库所做的更改将持久保存，即使发生系统故障也是如此。

2 Transaction状态

- 活动态：初始状态；事务在执行期间保持此状态
- 部分提交：执行完最后一条语句后（此时要输出的结果数据可能还在内存buffer中）
- 失败：发现无法继续正常执行之后。
- 中止：事务回滚且数据库恢复到事务开始前的状态之后。中止后有两个选项：
  - 重启事务：仅在无内部逻辑错误时方可执行。
  - 终止事务。
- 已提交：成功提交后。

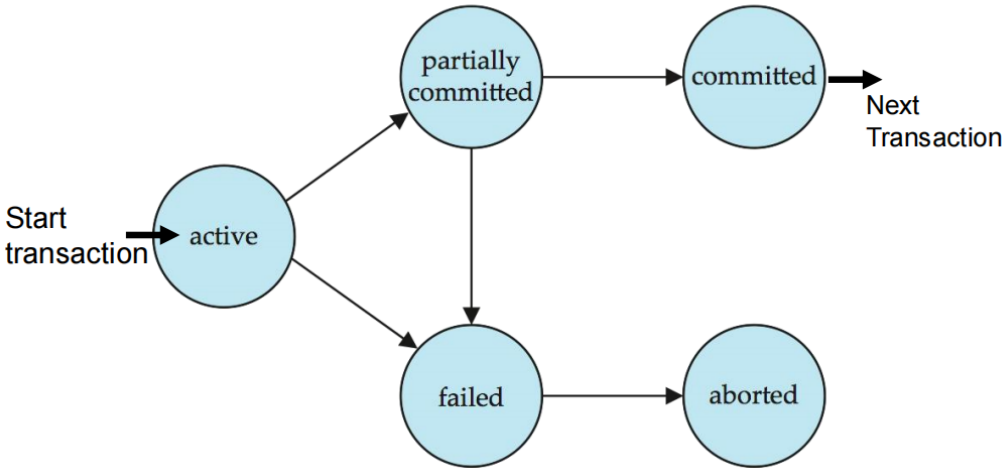

Figure 1. Transactions States

3 原子性和持久性的实现

数据库系统的恢复管理组件实现了对原子性和持久性的支持。

影子数据库方案——一种简单但效率低下的方案：一个名为db_pointer的指针始终指向数据库的当前一致副本。

所有更新都在新创建的数据库副本上进行。原始副本——影子副本（影子拷贝）保持不变——由db_pointer的指针始终指向数据库的当前一致副本。

- 如果中止：只需删除新副本。
- 如果提交：1.将新副本内存中的所有页面写入磁盘（在Unix系统中，使用刷新命令）
  2.修改数据库指针，使其指向新副本——新副本成为当前副本，同时删除旧副本。

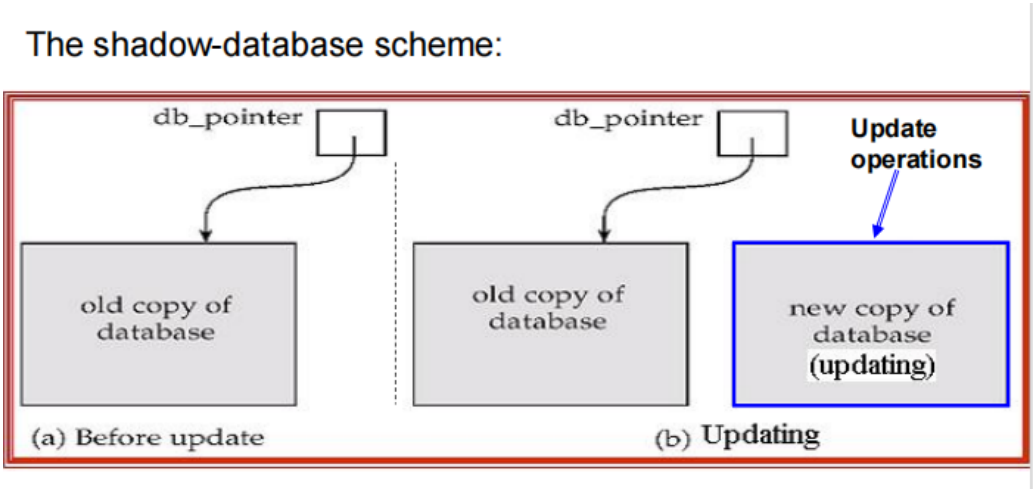

Figure 2

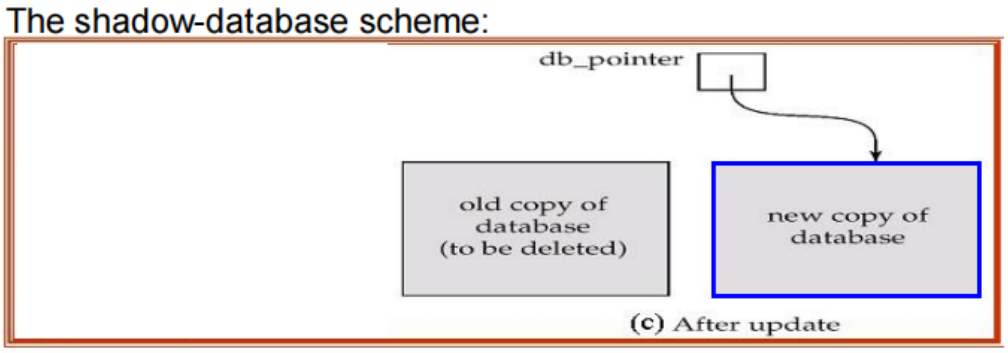

Figure 3

要求：原子性地更新数据库指针（磁盘系统可保证这一点，将其存储在单个扇区中）；无并发事务；假设磁盘不会发生故障。

这个想法对文本编辑器很有用。

但对于大型数据库而言效率极低；执行单个事务需要复值整个数据库。

4 并发执行

数据库系统允许多个事务并发运行。

并发执行的优点如下：

- 提高处理器和磁盘利用率，从而提高事务吞吐量：一个事务使用CPU时，另一个事务可以进行磁盘读写操作。
- 减少事务的平均响应时间：短事务无需等待长事务。

问题：尽管每个单独的事务都是正确的，但并发可能会破坏数据的一致性。（如并发售票问题）

并发控制方案——实现隔离的机制，即控制并发事务之间的交互，以防止它们破坏数据库的一致性。——这是数据库管理系统的重要工作

这里我们研究并发执行的正确性、可串行化、可恢复性等概念。

调度——指示并发事务的指令执行时间顺序的序列

- 一组事务的调度必须包含这些事务的所有指令
- 必须保留指令在每个单独事务中出现的顺序。

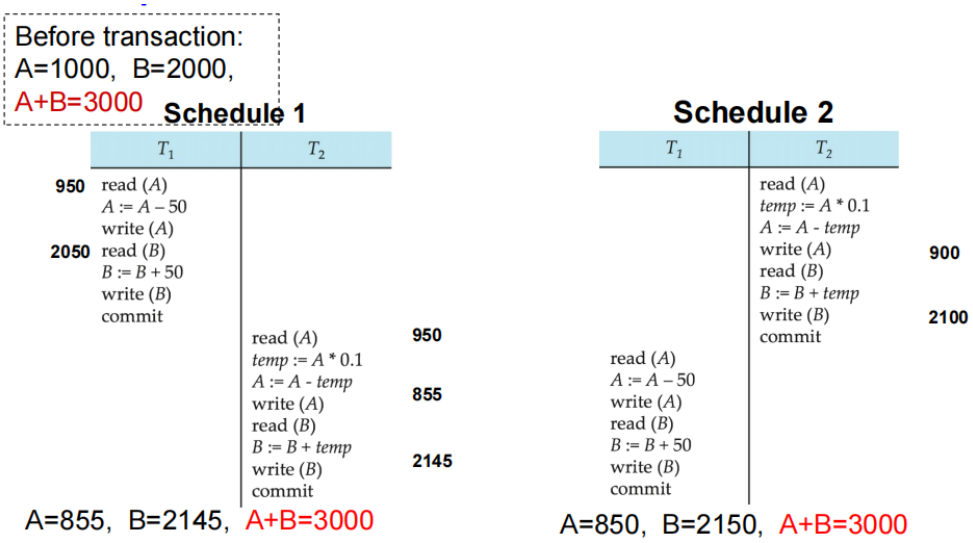

Figure 4

调度1和调度2均为串行调度。

N个并行事务有n!种可选的串行调度。串行调度必定保持一致性，但是效率低下。

5 可串行化

5.1 可串行化

基本假设：每个事务都能保持数据库一致性。

因此，一组事务的串行执行能保持数据库一致性。

如果一个（可能是并发的）调度等同于一个串行调度，那么它就是可串行化的。不同形式的调度等价性引出以下概念：

1. 冲突可串行化
2. 视图可串行化

5.2 事务的简化视图

我们忽略读写指令之外的操作。

我们假设事务可以在读写操作之间对本地缓冲区中的数据进行任意计算。

我们简化调度仅由读写指令组成。

5.3 冲突指令

事务T_i和T_j的指令I_i和I_j分别冲突，当且仅当存在某个项目Q被I_i和I_j同时访问，并且这些指令中至少有一个写入了Q。

1. $I_i = read(Q), I_j = read(Q)$.$I_i$和$I_j$不冲突。
2. $I_i = read(Q), I_j = write(Q)$.它们冲突。
3. $I_i = write(Q), I_j = read(Q)$.它们冲突。
4. $I_i = write(Q), I_j = write(Q)$。它们冲突。

直观的说，I_i和I_j之间的冲突会在它们之间强制形成一种（逻辑上的）时间顺序。

如果I_i和I_j在一个调度中是连续的，并且它们不冲突，那么即使在调度中交换它们的顺序，其结果也不变。若两个操作是冲突的，则两者执行次序不可交换。若两个操作不冲突，则可以交换次序。

5.4 冲突可串行性

如果一个调度S可以通过一系列非冲突指令的交换转换为调度S'，则称S和S'是冲突等价的。

如果一个调度S与一个串行调度冲突等价，则称该调度S是冲突可串行化的。

6 可恢复性

6.1 可恢复调度

如果事务T_j读取了事务T_i先前写入的数据项，那么T_i的提交操作必须出现在T_j的提交操作之前。

6.2 级联回滚

单个事务失败会导致一系列的事务回滚。

6.3 无级联调度

不会发生级联回滚；对于每一对事务T_i和T_j，若T_j读取了先前由T_i写入的数据项，则T_i的提交操作出现在T_j的读取操作之前。

每个无级联调度也是可恢复的。

将调度限制为无级联调度是可取的。

7 隔离性的实现

一种一次只允许一个事务执行的策略会生成串行调度，但并发程度较差。

为了保证数据库是一致性，调度必须是冲突可串行化或视图可串行化的，并且是可恢复的，最好是无级联的。

并发控制方案需要在允许的并发量和产生的开销吗之间进行权衡。

一些方案只允许生成冲突可串行化的调度，而另一些方案则允许生成非冲突可串行化的视图可串行化调度。

8 SQL中的事务定义

数据操作语言必须包含一种结构，用于指定构成事物的一组操作。在SQL中，事务会隐式开始。

SQL中的事务通过以下方式结束：

- 提交工作会提交当前事务并开始一个新事务
- 回滚工作会导致当前事务中止。

在几乎所有数据库系统中，默认情况下，每个SQL语句如果执行成功也会隐式提交。

9 可串行性测试

9.1 可串行性测试

考虑一组事务$T_1, T_2, ..., T_n$的某个调度。

优先图——一种有向图，其顶点为事务（名称）。

如果两个事务发生冲突，且T_i更早访问了产生冲突的数据项，则我们从T_i向T_j绘制一条弧。

我们可以用被访问的项来标记这条弧。

9.2 冲突串行性测试

如果一个调度满足以下条件，则它是冲突串行化的，当且仅当其优先图是无环的。

存在时间复杂度为n^2的环检测算法，其中n是图中顶点的数量。（更好的算法时间复杂度为$n + e$）

如果优先图是无环的，那么可通过对该图进行拓扑排序来获得可串行化顺序。这是一种

与图的偏序一致的线性顺序。

9.3 并发控制与可串行性测试

并发控制协议允许并发调度，但要确保这些调度是冲突/视图可串行化的，并且是可恢复且无级联回滚的。

并发控制协议通常不会在优先级创建时对其进行检查。相反，协议会施加一种规则来避免非可串行化调度。

不同的并发控制协议在允许的并发量和产生的开销之间提供了不同的权衡。

可串行性测试有助于我们理解并发控制协议为何是正确的。