数据库系统概论
An Introduction to Database System

第十一章 并发控制

中国人民大学信息学院
陈红

第十一章 并发控制

11.1 并发控制概述

11.2 封锁

11.3 活锁和死锁

11.4 并发调度的可串行性

11.5 两段锁协议

11.6 封锁的粒度

11.7 小结

11.3 活锁和死锁

- 封锁技术可以有效地解决并行操作的一致性问题，但也带来一些新的问题
  - 死锁
  - 活锁

11.3.1 活锁

- 事务 T1 封锁了数据 R
- 事务 T2 又请求封锁 R，于是 T2 等待。
- T3 也请求封锁 R，当 T1 释放了 R 上的封锁之后系统首先批准了 T3 的请求，T2 仍然等待。
- T4 又请求封锁 R，当 T3 释放了 R 上的封锁之后系统又批准了 T4 的请求……
- T2 有可能永远等待，这就是活锁的情形

活锁（续）

T_1	T_2	T_3	T_4
lock R	.	.	.
.	lock R	.	.
.	等待	Lock R	.
Unlock	等待	.	Lock R
.	等待	Lock R	等待
.	等待	.	等待
.	等待	Unlock	等待
.	等待	.	Lock R
.	等待	.	.

活 锁

活锁（续）

- 避免活锁：采用先来先服务的策略
  - 当多个事务请求封锁同一数据对象时
  - 按请求封锁的先后次序对这些事务排队
  - 该数据对象上的锁一旦释放，首先批准申请队列中第一个事务获得锁

11.3.2 死锁

- 事务 T1 封锁了数据 R1
- T2 封锁了数据 R2
- T1 又请求封锁 R2，因 T2 已封锁了 R2，于是 T1 等待 T2 释放 R2 上的锁
- 接着 T2 又申请封锁 R1，因 T1 已封锁了 R1， T2 也只能等待 T1 释放 R1 上的锁
- 这样 T1 在等待 T2，而 T2 又在等待 T1， T1 和 T2 两个事务永远不能结束，形成死锁

死锁（续）

T_1	T_2
lock R_1	•
•	Lock R_2
•	•
Lock R_2.	•
□□	•
□□	Lock R_1
□□	□□
□□	□□
	•

死　锁

解决死锁的方法

两类方法

1. 预防死锁
2. 死锁的诊断与解除

1. 死锁的预防

- 产生死锁的原因是两个或多个事务都已封锁了一些数据对象，然后又都请求对已为其他事务封锁的数据对象加锁，从而出现死等待。
- 预防死锁的发生就是要破坏产生死锁的条件

死锁的预防（续）

预防死锁的方法

- 一次封锁法
- 顺序封锁法

(1) 一次封锁法

- 要求每个事务必须一次将所有要使用的数据全部加锁，否则就不能继续执行
- 存在的问题
  - 降低系统并发度

一次封锁法（续）

- 难于事先精确确定封锁对象
  - 数据库中数据是不断变化的，原来不要求封锁的数据，在执行过程中可能会变成封锁对象，所以很难事先精确地确定每个事务所要封锁的数据对象
  - 解决方法：将事务在执行过程中可能要封锁的数据对象全部加锁，这就进一步降低了并发度。

(2) 顺序封锁法

- 顺序封锁法是预先对数据对象规定一个封锁顺序，所有事务都按这个顺序实行封锁。
- 顺序封锁法存在的问题
  - 维护成本

    数据库系统中封锁的数据对象极多，并且随数据的插入、删除等操作而不断地变化，要维护这样的资源的封锁顺序非常困难，成本很高。
  - 难以实现

    事务的封锁请求可以随着事务的执行而动态地决定，很难事先确定每一个事务要封锁哪些对象，因此也就很难按规定的顺序去施加封锁

死锁的预防（续）

- 结论
  - 在操作系统中广为采用的预防死锁的策略并不很适合数据库的特点
  - DBMS 在解决死锁的问题上更普遍采用的是诊断并解除死锁的方法

2. 死锁的诊断与解除

- 死锁的诊断
  - 超时法
  - 事务等待图法

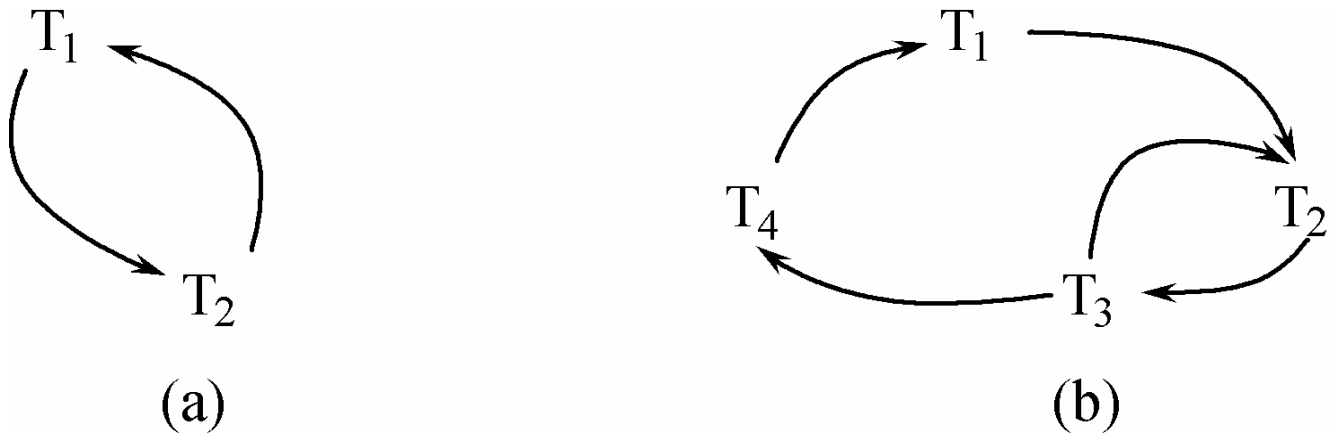
(1) 超时法

- 如果一个事务的等待时间超过了规定的时限，就认为发生了死锁
- 优点：实现简单
- 缺点
  - 有可能误判死锁
  - 时限若设置得太长，死锁发生后不能及时发现

(2) 等待图法

- 用事务等待图动态反映所有事务的等待情况
  - 事务等待图是一个有向图 $G=(T, U)$
  - T 为结点的集合，每个结点表示正运行的事务
  - U 为边的集合，每条边表示事务等待的情况
  - 若 T_1 等待 T_2，则 T_1，T_2 之间划一条有向边，从 T_1 指向 T_2

等待图法（续）



事务等待图

- 图 (a) 中，事务 T1 等待 T2， T2 等待 T1，产生了死锁
- 图 (b) 中，事务 T1 等待 T2， T2 等待 T3， T3 等待 T4， T4 又等待 T1，产生了死锁
- 图 (b) 中，事务 T3 可能还等待 T2，在大回路中又有小的回路

等待图法（续）

- 并发控制子系统周期性地（比如每隔数秒）生成事务等待图，检测事务。如果发现图中存在回路，则表示系统中出现了死锁。

死锁的诊断与解除（续）

- 解除死锁
  - 选择一个处理死锁代价最小的事务，将其撤消
  - 释放此事务持有的所有的锁，使其它事务能继续运行下去