

第10章 并发控制



课程知识结构

Chp.1 数据库系统概述

Chp.2 数据库系统体系结构

Chp.3 关系数据模型

Chp.9 事务与恢复

Chp.4 SQL

Chp.6 关系数据库模式设计

Chp.10 并发控制

Chp.5 过程化SQL

Chp.7 数据库设计

Chp.11 完整性

Chp.8 数据库应用开发

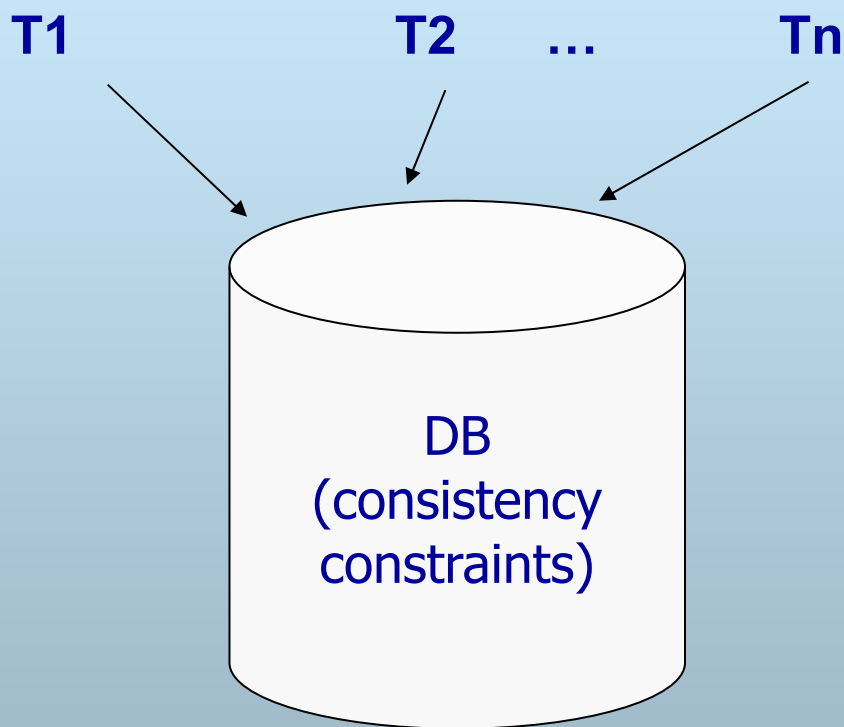
Chp.12 安全性

Chp.13 高级主题

Databases Protection

- 数据库保护：排除和防止各种对数据库的干扰破坏，确保数据安全可靠，以及在数据库遭到破坏后尽快地恢复
- 数据库保护通过四个方面来实现
 - 完整性控制技术
 - ◆ Enable constraints
 - 安全性控制技术
 - ◆ Authorization and authentication
 - 数据库的恢复技术
 - ◆ Deal with failure
 - 并发控制技术
 - ◆ Deal with data sharing

Concurrency Control



多个事务同时存取共享的数据库时，如何保证数据库的一致性？

主要内容

- 并发操作与并发问题
- 并发事务调度与可串性
(Scheduling and Serializability)
- 锁与可串性实现 (Locks)
- 事务的隔离级别
- 死锁

一、并发操作和并发问题

■ 并发操作

- 在多用户**DBS**中，如果多个用户同时对同一数据进行操作称为**并发操作**
- 并发操作使多个事务之间可能产生相互干扰，破坏事务的**隔离性**（**Isolation**）
- **DBMS**的并发控制子系统负责协调并发事务的执行，保证数据库的一致性，避免产生不正确的数据

■ 并发操作通常会引起三类问题（**三大并发问题**）

- 丢失更新（**Lost update**）
- 脏读（**Dirty read / Uncommitted update**）
- 不一致分析（**Inconsistent analysis**）

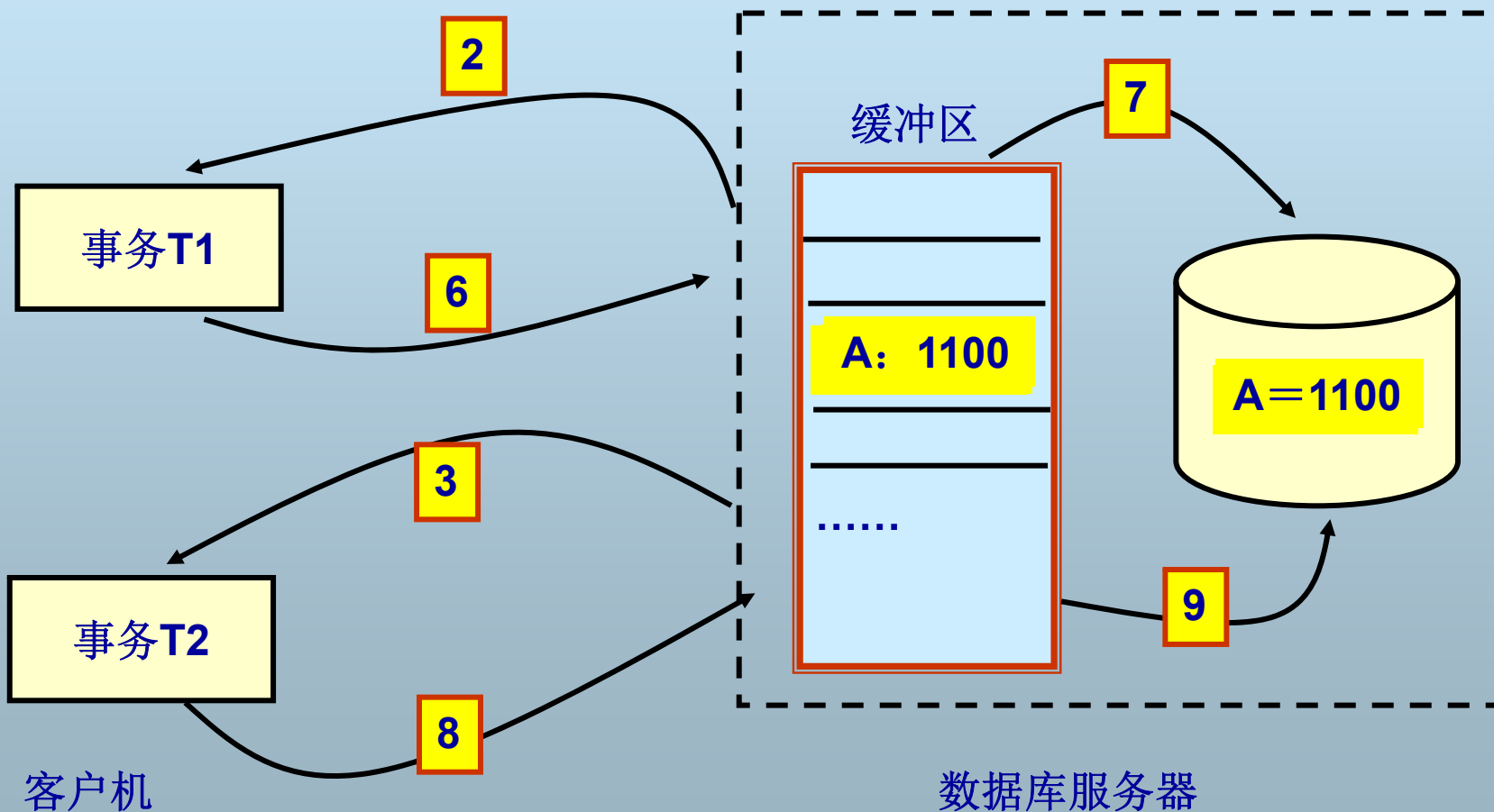
1、丢失更新问题

时间	事务T1	事务T2	数据库中A的值
1			1000
2	Read(A,t)		
3		Read(A,t)	
4	t=t-100		
5		t=t+100	
6	Write(A,t)		
7	Commit		900
8		Write(A,t)	
9		Commit	1100

基于延迟更新的执行示例

延迟更新：事务Commit后才将所有更新写入数据库

时间	事务T1	事务T2	数据库中A的值
1			1000
2	Read(A,t)		
3		Read(A,t)	
4	t=t-100		
5		t=t+100	
6	Write(A,t)		
7	Commit		900
8		Write(A,t)	
9		Commit	1100



2、脏读问题

时间	事务T1	事务T2	数据库中A的值
1			1000
2	Read(A,t)		
3	t=t-100		
4	Write(A,t)		
5		Read(A,t)	
6	Rollback	t=t+100	900
7		Write(A,t)	
8		Commit	1000

脏数据：事务在内存中更新了但还未最终提交的数据

3、不一致分析问题

时间	事务T1	事务T2
1		
2	Read(A,t)	Read(B,t)
3	t=t-100	
4		Read(A,v)
5	Write(A,t)	
6	Commit	
7		Sum=t+v
8		Commit

不一致分析问题：事务读了过时的数据

4、问题如何解决？

■ 一种方法

- 让所有事务一个一个地串行执行
 - ◆ 一个事务在执行时其它事务只能等待
 - ◆ 不能充分利用系统资源，效率低下

■ 另一种方法

- 为了充分发挥**DBMS**共享数据的特点，应允许事务内部的读写操作并发执行
- 挑战
 - ◆ 必须保证事务并发执行的正确性；必须用正确的方法调度执行事务的并发操作

二、调度(Schedule)

Example

T1: Read(A, t)
 $t \leftarrow t+100$
 Write(A, t)
 Read(B, t)
 $t \leftarrow t+100$
 Write(B, t)

T2: Read(A, s)
 $s \leftarrow s \times 2$
 Write(A, s)
 Read(B, s)
 $s \leftarrow s \times 2$
 Write(B, s)

Constraint: $A=B$

二、调度(Schedule)

Schedule A

T1	T2	A	B
Read(A, t); $t \leftarrow t+100$		25	25
Write(A, t);		125	25
Read(B, t); $t \leftarrow t+100$;			
Write(B, t);		125	125
	Read(A, s); $s \leftarrow s \times 2$;		
	Write(A, s);	250	125
	Read(B, s); $s \leftarrow s \times 2$;		
	Write(B, s);	250	250

二、调度(Schedule)

Schedule B

T1	T2	A	B
	Read(A, s); $s \leftarrow s \times 2$;	25	25
	Write(A, s);	50	25
	Read(B, s); $s \leftarrow s \times 2$;		
	Write(B, s);	50	50
Read(A, t); $t \leftarrow t + 100$			
Write(A, t);		150	50
Read(B, t); $t \leftarrow t + 100$;			
Write(B, t);		150	150

二、调度(Schedule)

Schedule C

T1	T2	A	B
Read(A, t); $t \leftarrow t+100$		25	25
Write(A, t);		125	25
	Read(A, s); $s \leftarrow s \times 2$;		
	Write(A, s);	250	25
Read(B, t);			
$t \leftarrow t+100$;			
Write(B, t);		250	125
	Read(B, s); $s \leftarrow s \times 2$;		
	Write(B, s);	250	250

二、调度(Schedule)

Schedule D

T1	T2	A	B
Read(A, t); $t \leftarrow t+100$		25	25
Write(A, t);		125	25
	Read(A, s); $s \leftarrow s \times 2$;		
	Write(A, s);	250	25
	Read(B, s); $s \leftarrow s \times 2$;		
	Write(B, s);	250	50
Read(B, t);			
$t \leftarrow t+100$;			
Write(B, t);		250	150

二、调度(Schedule)

Schedule D

T1	T2'	A	B
Read(A, t); $t \leftarrow t+100$		25	25
Write(A, t);		125	25
	Read(A, s); $s \leftarrow s \times 1$;		
	Write(A, s);	125	25
	Read(B, s); $s \leftarrow s \times 1$;		
	Write(B, s);	125	25
Read(B, t);			
$t \leftarrow t+100$;			
Write(B, t);		125	125

1、调度的定义

■ 调度

- 多个事务的读写操作按时间排序的执行序列

T1: r1(A) w1(A) r1(B) w1(B)

T2: r2(A) w2(A) r2(B) w2(B)

Sc = r1(A) w1(A) r2(A) w2(A) r1(B) w1(B) r2(B) w2(B)

Note

- 调度中每个事务的读写操作保持原来顺序
- 事务调度时不考虑
 - ◆ 数据库的初始状态 (**Initial state**)
 - ◆ 事务的语义 (**Transaction semantics**)

1、调度的定义

■ 多个事务的并发执行存在多种调度方式

Example:

$S_c = r_1(A) \ w_1(A) \ r_2(A) \ w_2(A) \ r_1(B) \ w_1(B) \ r_2(B) \ w_2(B)$

$S_a = r_1(A) \ w_1(A) \ r_1(B) \ w_1(B) \ r_2(A) \ w_2(A) \ r_2(B) \ w_2(B)$

T1

T2

What is a correct schedule?

And how to get a correct schedule?

2、可串化调度 (Serializable Schedule)

■ What is a correct schedule?

- Answer: a serializable schedule!

■ 串行调度 (Serial schedule)

- 各个事务之间没有任何操作交错执行，事务一个一个执行
- $S = T1 T2 T3 \dots Tn$

■ Serializable Schedule

- 如果一个调度的结果与某一串行调度执行的结果等价，则称该调度是可串化调度，否则是不可串调度

2、可串化调度

■ 可串化调度的正确性

- **Consistence of transaction:** 单个事务的执行保证**DB**从一个一致状态变化到另一个一致状态
- **N个事务串行调度执行仍保证 Consistence of DB**

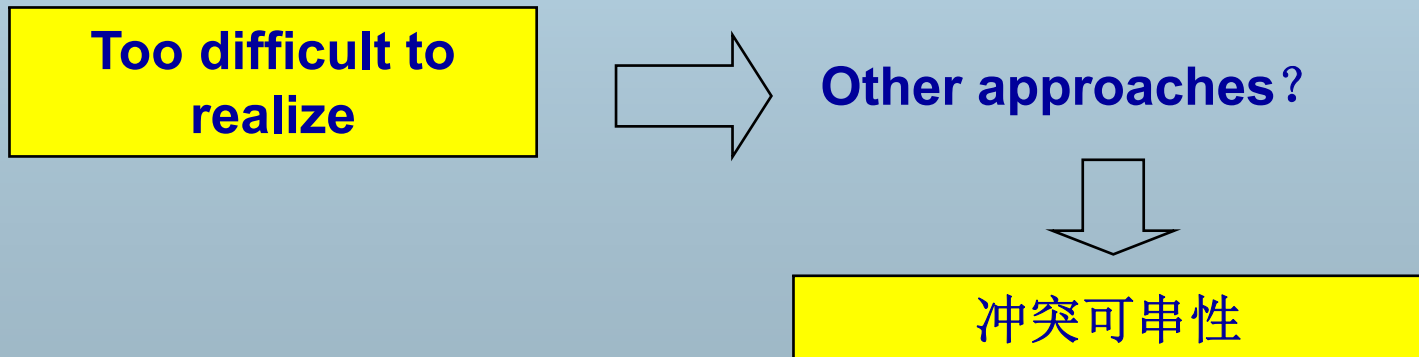


2、可串化调度

■ Is a schedule a serializable one?

● We MUST

- ◆ Get all results of serial schedules, $n!$
- ◆ See if the schedule is equivalent to some serial schedule



3、冲突可串性 (conflict serializable)

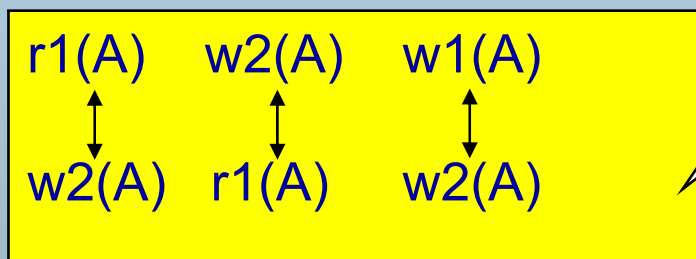
■ Conflicting actions

● Say

◆ $r_i(X)$: 事务 T_i 的读 X 操作 ($Read(X, t)$)

◆ $w_i(X)$: 事务 T_i 的写 X 操作 ($Write(X, t)$)

● 冲突操作



涉及同一个数据库元素，并且至少有一个是写操作

3、冲突可串性 (conflict serializable)

■ Conflicting actions

- 如果调度中一对连续操作是冲突的，则意味着如果它们的执行顺序交换，则至少会改变其中一个事务的最终执行结果
- 如果两个连续操作不冲突，则可以在调度中交换顺序

3、冲突可串行性

Schedule C

T1	T2	A	B
Read(A, t); $t \leftarrow t+100$		25	25
Write(A, t);		125	25
	Read(A, s); $s \leftarrow s \times 2$;		
	Write(A, s);	250	25
Read(B, t);			
$t \leftarrow t+100$;			
Write(B, t);		250	125
	Read(B, s); $s \leftarrow s \times 2$;		
	Write(B, s);	250	250

$$Sc = r1(A) w1(A) r2(A) w2(A) r1(B) w1(B) r2(B) w2(B)$$

3、冲突可串行性

$Sc = r1(A) w1(A) r2(A) w2(A) r1(B) w1(B) r2(B) w2(B)$

$Sc' = r1(A) w1(A) r2(A) r1(B) w2(A) w1(B) r2(B) w2(B)$

$Sc' = r1(A) w1(A) r1(B) r2(A) w2(A) w1(B) r2(B) w2(B)$

$Sc' = r1(A) w1(A) r1(B) r2(A) w1(B) w2(A) r2(B) w2(B)$

$Sc' = r1(A) w1(A) r1(B) w1(B) r2(A) w2(A) r2(B) w2(B)$

T1

T2

Schedule A

3、冲突可串行性

$Sc = r1(A) w1(A) r2(A) w2(A) r1(B) w1(B) r2(B) w2(B)$

同一个事务的操作必须符合原来的顺序

冲突操作

3、冲突可串行性

Schedule C

此步读入的B为25

T1	T2	A	B
Read(A, t); $t \leftarrow t+100$		25	25
Write(A, t);		125	25
	Read(A, s); $s \leftarrow s \times 2$;		
	Write(A, s);	250	25
Read(B, t);			
$t \leftarrow t+100$;			
	Read(B, s); $s \leftarrow s \times 2$;		
Write(B, t);		250	125
	Write(B, s);	250	50

3、冲突可串行性

- **冲突等价 (conflict equivalent)**
 - **S1, S2 are conflict equivalent schedules if S1 can be transformed into S2 by a series of swaps on non-conflicting actions.**
- **冲突可串行性 (conflict serializable)**
 - **A schedule is conflict serializable if it is conflict equivalent to some serial schedule.**

3、冲突可串行性

■ 定理

- 如果一个调度满足冲突可串行性，则该调度是可串行化调度

■ Note

- 仅为充分条件

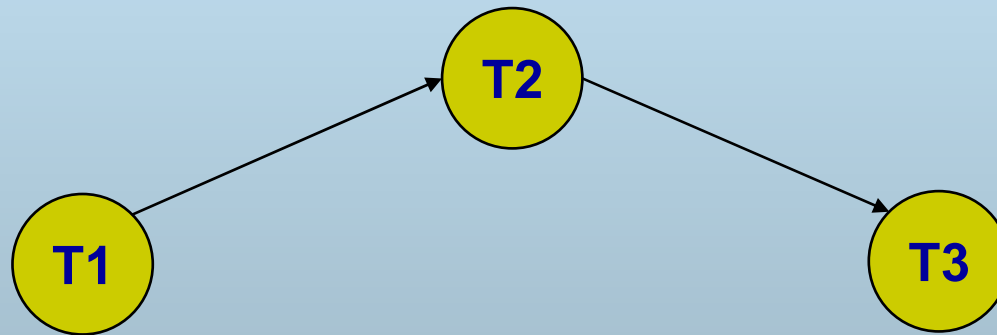
4、优先图 (Precedence Graph)

- 优先图用于冲突可串性的判断
- 优先图结构
 - 结点 (Node): 事务
 - 有向边 (Arc): $T_i \rightarrow T_j$, 满足 $T_i <_s T_j$
 - ◆ 存在 T_i 中的操作A1和 T_j 中的操作A2, 满足
 - A1在A2前, 并且
 - A1和A2是冲突操作

4、优先图

Example

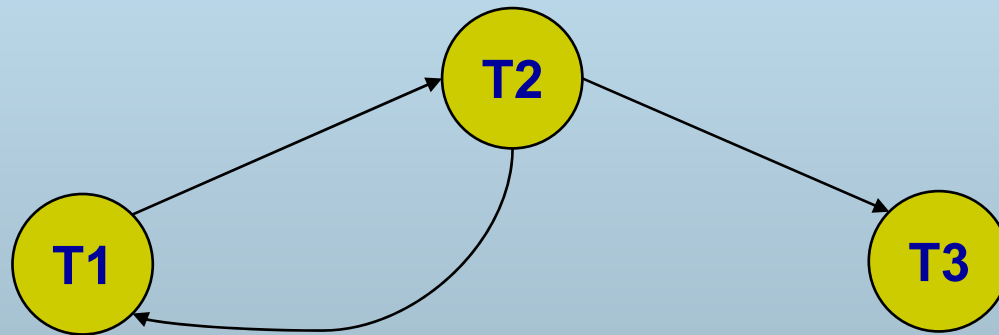
$S = r_2(A) \ r_1(B) \ w_2(A) \ r_3(A) \ w_1(B) \ w_3(A) \ r_2(B) \ w_2(B)$



4、优先图

Example

$S = r_2(A) \ r_1(B) \ w_2(A) \ r_2(B) \ r_3(A) \ w_1(B) \ w_3(A) \ w_2(B)$




4、优先图

■ 优先图与冲突可串行性

- 给定一个调度 S ，构造 S 的优先图 $P(S)$ ，若 $P(S)$ 中无环，则 S 满足冲突可串行性
- 证明：归纳法
 - ◆ see "H. Molina et al. *Database System Implementation*"

Next

- 并发操作与并发问题
- 并发事务调度与可串行性
(Scheduling and Serializability)
- 锁与可串行性实现 (Locks) 
- 事务的隔离级别