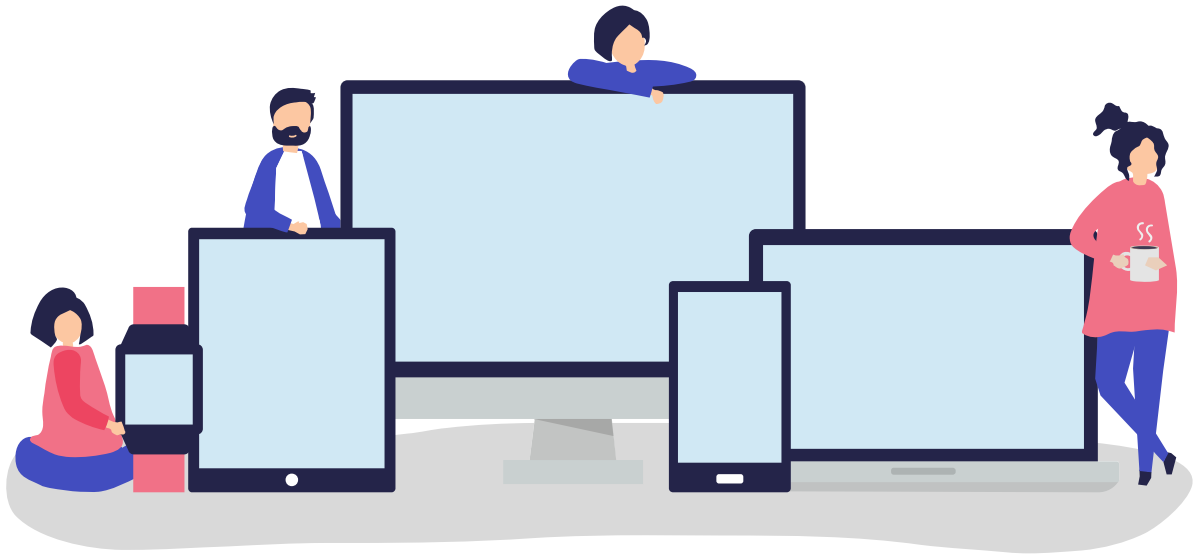
CONTENTS

事务并发执行可能出现的问题



读脏数据

丢失更新

不可重复读

事务并发执行问题举例

■[例 3] 设 A 航班的剩余票数为 10 张，有两个事务 T_1 和 T_2 同时请求出售该航班机票 2 张和 3 张。它们各自的执行序列如图 -1 所示 (这里只考虑对航班剩余票数的更新) 。

T_1	T_2
R(A)	R(A)
A=A-2	A=A-3
W(A)	W(A)

图 -1　更新事务 T_1 和 T_2

事务并发执行问题举例

■如果是串行执行，则不管是 T_1 先执行再执行 T_2(图 -2(a))，还是 T_2 先执行再执行 T_1(图 -2(b))，都可得到正确的执行结果，即剩余票数都为 5 。

T_1	T_2	A
R(A)		10
A=A-2		
W(A)		8
	R(A)	8
	A= A-3	
	W(A)	5

(a)

T_1	T_2	A
	R(A)	10
	A=A-3	
	W(A)	7
R(A)		7
A=A-2		
W(A)		5

(b)

图 -2 T_1 和 T_2 串行执行

事务并发执行问题举例

■读脏数据：在图 -3 中，事务 T_1 在 T_2 读取其更新值 (8) 后回滚，而 T_2 仍然使用读到 T_1 修改后的值进行运算，得到的结果是 5 。但实际上 T_1 未执行成功，系统只出售了 3 张票，余票数应为 7 。

■ **读脏数据。**如果事务T_2读取事务T_1修改但未提交的数据后，事务T_1由于某种原因中止而撤销，这时事务T_2就读取了不一致的数据。数据库中将这种读未提交且被撤销的数据为读“脏数据”。

T_1	T_2	A
R(A)		10
A=A-2		
W(A)		8
	R(A)	8
rollback		
	A=A-3	
	W(A)	5

图 -3 T_2 读脏数据

事务并发执行问题举例

■不可重复读：如图 -4 所示，T_3 第一次读时余票数为 10 张，第二次读时为 8 张，两次读结果不一致。

■ **不可重复读**。是指事务T_i两次从数据库中读取的结果不同，可分为三种情况：

- 事务T_i读取一数据后，事务T_j对该数据进行了更改。当事务T_i再次读该数据时，则会读到与前一次不同的值。
- 事务T_i按某条件读取数据库中某些记录后，事务T_j删除了其中部分记录。当事务T_i再次按相同条件读取时，发现记录数变少了。（幻影现象1）
- 事务T_i按某条件读取数据库中某些记录后，事务T_j插入了新的记录。当事务T_i再次按相同条件读取时，发现记录数变多了。（幻影现象2）

T_1	T_3	A
	R(A)	10
R(A)		10
A=A-2		
W(A)		8
	R(A)	8

图 -4　T_3 不可重复读

事务并发执行问题举例

■丢失更新：在图 -5 中，T_1 和 T_2 都读到余票数为 10，由于 T_1 后于 T_2 提交，导致 T_2 的更新操作没有发生作用，被 T_1 的更新值 (8) 覆盖。

■ **丢失更新**。两个或多个事务都读取了同一数据值并修改，最后提交事务的执行结果覆盖了前面提交事务的执行结果，从而导致前面事务的更新被丢失。

T_1	T_2	A
R(A)		10
	R(A)	10
	$A=A-3$	
	W(A)	7
$A=A-2$		8
W(A)		8

是不是所有并发执行事务都会出现这些问题呢?

图 -5 T_2 更新丢失

事务并发执行得到正确的结果

启示：如果一组并发执行事务的执行结果与它们串行执行得到的结果是相同的，那么就可以认为该并发执行的结果是正确的。

可以验证图 -6 中并发执行的结果与 T_4、T_5 按先后顺序串行执行的结果是一样的，也是正确的。

T_4	T_5	A	B
		10	
R(A)			
A=A-2			
		8	
W(A)			
	R(A)	8	
	A=A-3		
	W(A)	5	
			15
R(B)			
B=B-2			
			13
W(B)			
	R(B)		13
	B=B-3		
	W(B)		10

图 -6 T_4 和 T_5 并发执行

事务调度定义

■事务并发执行顺序是随机的，将由多个事务操作组成的随机执行序列称为一个调度。

■对由一组事务操作组成的调度序列而言，应满足下列条件：

- 该调度应包括该组事务的全部操作；
- 属于同一个事务的操作应保持在原事务中的执行顺序。

T1:a1a2a3a4a5a6

T2: b1b2b3b4b5b6b7

T3: c1c2c3c4c5c6

三个事务的其中一个调度:

c1c2c3b1a1b2a2b3b4a3a4a5b5b6a6b7c4c5c6

串行调度

- 定义 1(串行调度)　在调度 S 中，如果属于同一事务的操作都是相邻的，则称 S 是串行调度。
- 对由 n 个事务组成的一组事务而言，共有 $n!$ 种有效串行调度。
- 事务串行执行可保证数据库的一致性，如果能判断一个并发调度的执行结果等价于一个串行调度的结果，就称该并发调度可保证数据库的一致性。

冲突操作与冲突等价

- 假设调度 S 包含两个事务 T_i 与 T_j，若两个相邻操作 $O_i \in T_i$，$O_j \in T_j$ 访问不同的数据对象，则交换 O_i 与 O_j 不会影响调度中任何操作的结果。若 O_i 与 O_j 访问相同的数据对象，并且有一个为写操作时，则不能改变它们被调度执行的顺序。
- 定义 2　在一调度 S 中，如果 O_i 与 O_j 是不同事务在相同数据对象上的操作，并且其中至少有一个是写操作，则称 O_i 与 O_j 是冲突操作；否则称为非冲突操作。
- 定义 3　如果一调度 S 可以经过交换一系列非冲突操作执行的顺序而得到一个新的调度 S'，则称 S 与 S' 是冲突等价的 (conflict equivalent)。

冲突可串行化

■定义 4　如果一调度 S 与一串行调度是冲突等价的，则称 S 是冲突可串行化的 (conflict serializable).

■在图 -6 的调度中，通过交互非冲突操作可得到图 -7 所示的串行调度 $<T_4, T_5>$，故图 -6 的调度是冲突可串行化的。

T_4	T_5	A	B
R(A)		10	
A=A-2			
W(A)		8	
	R(A)	8	
	A=A-3		
	W(A)	5	
R(B)			15
B=B-2			
W(B)			13
	R(B)		13
	B=B-3		
	W(B)		10

图-6 T_4和T_5并发执行

T_4	T_5
R(A)	
A=A-2	
W(A	
R(B)	
B=B-2	
W(B)	
	R(A)
	A=A-3
	W(A)
	R(B)
	B=B-3
	W(B)

图 -7　交换图 -6 的非冲突操作后得到的串行调度 $<T_4, T_5>$

冲突可串行化

■冲突可串行化仅仅是正确调度的充分条件，并不是必要条件，即冲突可串行化调度执行结果一定是正确的，而正确的调度不一定都是冲突可串行化的。

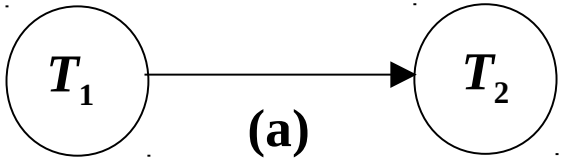
T_4	T_6	A	B
		10	
R(A)			
A=A-2			
		8	
W(A)			
			15
	R(B)		
	B=B-3		12
R(B)	W(B)		12
B=B-2			
W(B)			10
		8	
	R(A)		
	A=A-3	5	
	W(A)		

图 -8　不满足冲突可串行化的正确调度

优先图

■设 S 是一个调度。由 S 构造一个有向图，称为优先图，记为 $G=(V, E)$，其中 V 是顶点集，E 是边集。顶点集由所有参与调度的事务组成，边集由满足下列 3 个条件之一的边 $T_i \rightarrow T_j$ 组成：

- T_i 执行了 $W_i(Q)$ 后 T_j 执行 $R_j(Q)$；
- T_i 执行了 $R_i(Q)$ 后 T_j 执行 $W_j(Q)$；
- T_i 执行了 $W_i(Q)$ 后 T_j 执行 $W_j(Q)$。



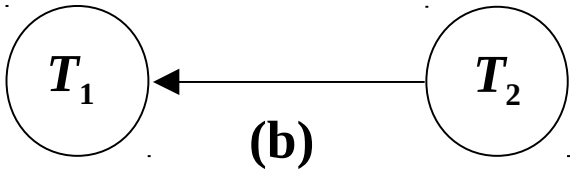


图 -9　图 -2 的优先图

T_1	T_2	A
R(A)		10
A=A-2		
W(A)		8
	R(A)	8
	A=A-3	
	W(A)	5

(a)

T_1	T_2	A
	R(A)	10
	A=A-3	
	W(A)	7
R(A)		7
A=A-2		
W(A)		5

(b)

图-2 T_1和T_2串行执行

优先图

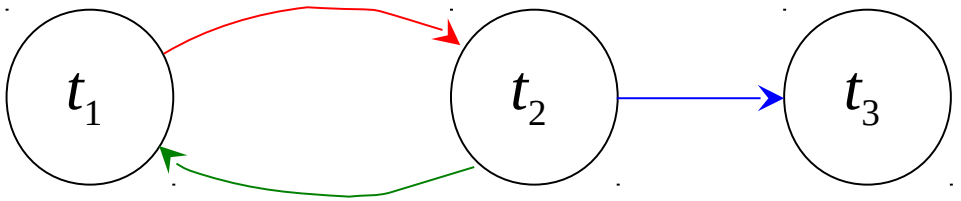
设S是一个调度。由S构造一个有向图，称为优先图，记为$G=(V, E)$，其中V是顶点集，E是边集。顶点集由所有参与调度的事务组成，边集由满足下列3个条件之一的边$T_i \rightarrow T_j$组成：

- T_i执行了$W_i(Q)$后T_j执行$R_j(Q)$；
- T_i执行了$R_i(Q)$后T_j执行$W_j(Q)$；
- T_i执行了$W_i(Q)$后T_j执行$W_j(Q)$。

- T={ t_1: $R_1(X)$ $W_1(X)$ C_1;
  t_2: $R_2(X)$ $R_2(Y)$ $W_2(X)$ C_2;
  t_3: $R_3(Y)$ $W_3(Y)$ C_3}

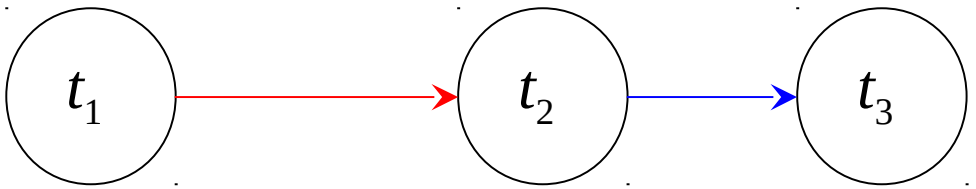
➢ H_T = $R_1(X)$ $R_2(X)$ $W_1(X)$ $R_2(Y)$ C_1 $W_2(X)$ $R_3(Y)$ C_2 $W_3(Y)$ C_3

➢ $SG(H_T)$:



➢ H_T = $R_1(X)$ $W_1(X)$ $R_2(X)$ $R_2(Y)$ C_1 $W_2(X)$ $R_3(Y)$ C_2 $W_3(Y)$ C_3

➢ $SG(H_T)$:



基于优先图的冲突可串行化判别

■基于优先图的冲突可串行化判别准则：如果优先图中无环，则 S 是冲突可串行化的；如果优先图中有环，则 S 是非冲突可串行化的。

■测试冲突可串行化的算法：

- 构建 S 的优先图；
- 采用环路测试算法（如基于深度优先搜索的环检测算法）检测 S 中是否有环；
- 若 S 包含环，则 S 是非冲突可串行化的，否则调度 S 是冲突可串行化的。

基于优先图的冲突可串行化判别

■[例 5] 设并发调度事务集为：

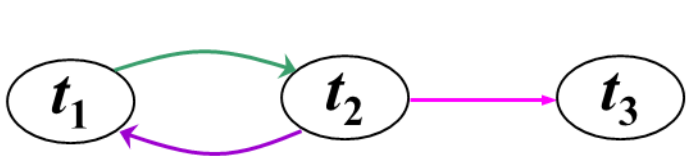
T={t_1: $R_1(X)$ $W_1(X)$, t_2: $R_2(X)$ $R_2(Y)$ $W_2(X)$, t_3: $R_3(Y)$ $W_3(Y)$}

两个并发调度分别为：

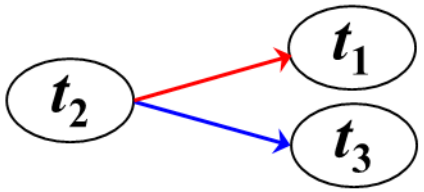
H_T = $R_1(X)$ $R_2(X)$ $W_1(X)$ $R_2(Y)$ $W_2(X)$ $R_3(Y)$ $W_3(Y)$

H_T' = $R_2(X)$ $R_3(Y)$ $R_2(Y)$ $W_2(X)$ $R_1(X)$ $W_3(Y)$ $W_1(X)$

试判断这两个调度是否是冲突可串行化调度?



(a) 调度H_T的优先图

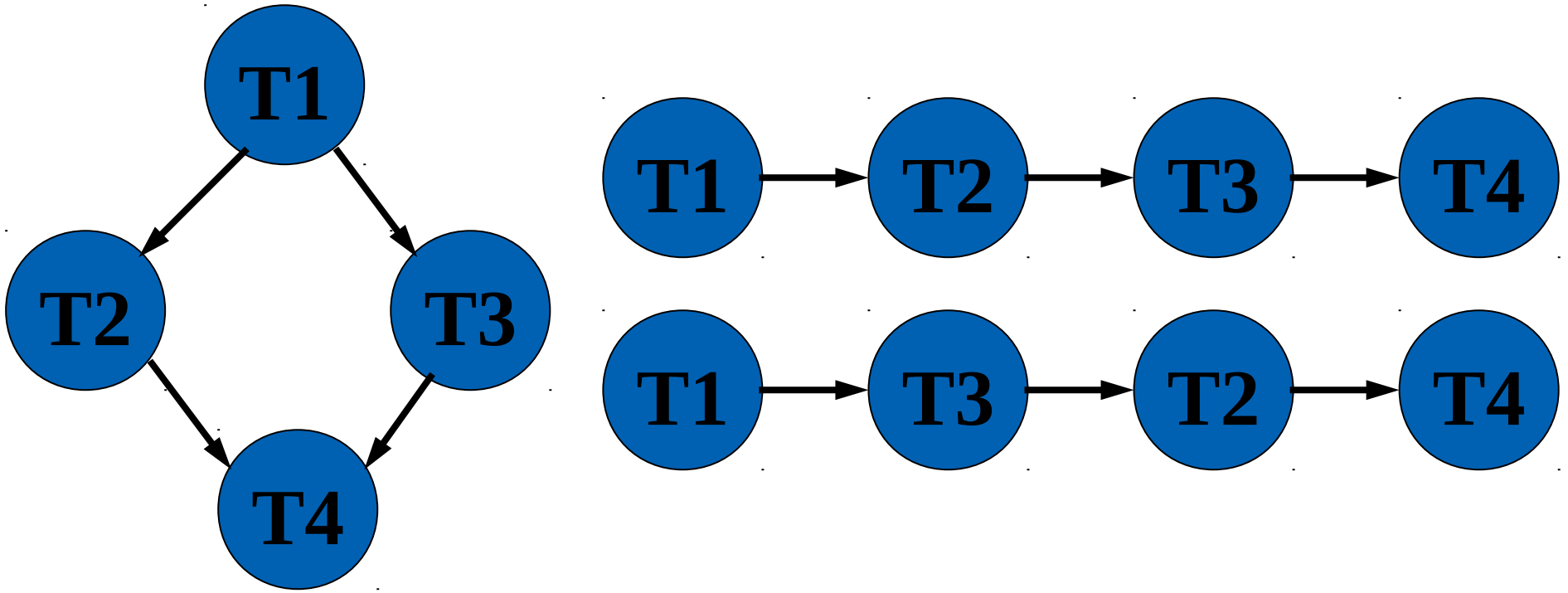


(b) 调度H_T'的优先图

因此，调度 H_T 不是冲突可串行化的；而调度 H_T' 是冲突可串行化的，它冲突等价于串行调度 <t_2, t_1, t_3> 或 <t_2, t_3, t_1> 。

可串行化

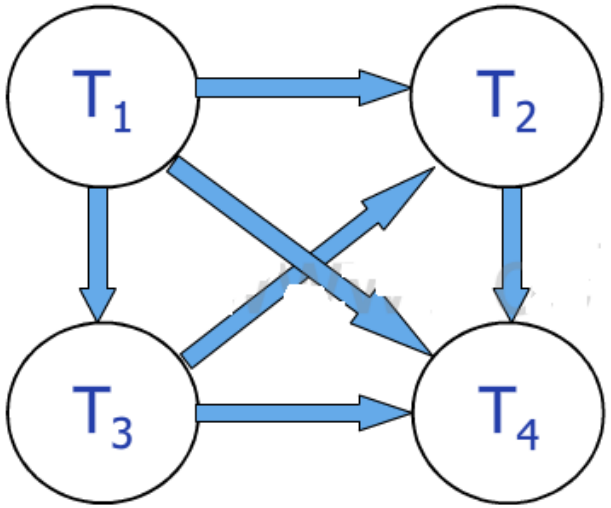
实现的基本方法
拓扑排序方法如下：
(1) 从有向图中选择一个没有前驱 (即入度为 0) 的顶点并且输出它 .
(2) 从网中删去该顶点 , 并且删去从该顶点发出的全部有向边 .
(3) 重复上述两步 , 直到剩余的网中不再存在没有前趋的顶点为止 .



题目

例 1 设有事务集 {T1,T2,T3,T4} 的一个调度

S=W3(y)R1(x)R2(y)W3(x)W2(x)W3(z)R4(z) W4(x) 试检验 S 是否可串行化，试找出其等价的串行调度



实现的基本方法
拓扑排序方法如下 :
(1) 从有向图中选择一个没有前驱 (即入度为 0) 的顶点并且输出它 .
(2) 从网中删去该顶点 , 并且删去从该顶点发出的全部有向边 .
(3) 重复上述两步 , 直到剩余的网中不再存在没有前趋的顶点为止 .

小结

事务并发带来的问题?

如何克服问题? 事务可串行化，判断方法?