

页面中靠下的在时间上靠后。此外，所给  $A$  和  $B$  值指的是它们在主存缓冲区中的值，而不一定是它们在磁盘上的值。

$T_1$	$T_2$
READ( $A, t$ )	READ( $A, s$ )
$t := t+100$	$s := s*2$
WRITE( $A, t$ )	WRITE( $A, s$ )
READ( $B, t$ )	READ( $B, s$ )
$t := t+100$	$s := s*2$
WRITE( $B, t$ )	WRITE( $B, s$ )

图 7-2 两个事务

$T_1$	$T_2$	$A$	$B$
READ( $A, t$ )		25	25
$t := t+100$			
WRITE( $A, t$ )		125	
READ( $B, t$ )			
$t := t+100$			
WRITE( $B, t$ )			125
	READ( $A, s$ )		
	$s := s*2$		
	WRITE( $A, s$ )	250	
	READ( $B, s$ )		
	$s := s*2$		
	WRITE( $B, s$ )		250

图 7-3  $T_1$  在  $T_2$  前的串行调度

图 7-4 给出了  $T_2$  在  $T_1$  前的另一个串行调度，初态仍假设为  $A = B = 25$ 。请注意，两个调度  $A$  和  $B$  最终的值是不同的； $A$  和  $B$  在先做  $T_1$  时都是 250，而在先做  $T_2$  时都是 150。通常，我们不能期望数据库终态与事务顺序无关。□

我们可以像图 7-3 和图 7-4 中那样，通过按发生顺序列出所有动作来表示串行调度。但是，由于串行调度中动作的顺序只依赖于事务本身的顺序，我们有时通过事务列表来表示串行调度。因此，图 7-3 中的调度表示为  $(T_1, T_2)$ ，而图 7-4 中的调度表示为  $(T_2, T_1)$ 。

### 7.1.3 可串行化调度

事务的正确性原则告诉我们，每个串行调度都将保持数据库状态的一致性。但是还有其他能保证可保持一致性的调度吗？有，下面的例子可以说明。通常，如果存在串行调度  $S'$ ，使得对于每个数据库初态，调度  $S$  和调度  $S'$  的效果相同，我们就说这个调度  $S$  是可串行化的。

**例 7.3** 图 7-5 给出了例 7.1 中事务的一个调度，此调度是可串行化的，但不是串行的。在这个调度中， $T_2$  在  $T_1$  作用于  $A$  后而在  $T_1$  作用于  $B$  前作用于  $A$ 。但是，我们看到两个事务按这种方式调度，结果和在图 7-3 中看到的串行调度  $(T_1, T_2)$  一样。为了说服自己这一陈述是正确的，我们必须不仅考虑像图 7-5 中那样从数据库状态  $A = B = 25$  开始产生的结果，还要考虑从任何一致的状态开始的情况。由于所有一致的数据库状态满足  $A = B = c$ ，不难推断在图 7-5 的调度中， $A$  和  $B$  得到的值都是  $2(c + 100)$ ，因此从任意一致的状态开始一致性都能得到保持。

另一方面，考虑图 7-6 的非可串行化的调度。我们之所以能确定它不是可串行化的，原因在于它从一致的状态  $A = B = 25$  开始，最后使数据库处于不一致的状态  $A = 250$  而  $B = 150$ 。请注意，按照这个动作的顺序，即  $T_1$  先作用于  $A$ ，而  $T_2$  先作用于  $B$ ，我们实际上在  $A$  和  $B$  上实施了不同的运算，也就是说  $A := 2(A + 100)$ ，而  $B := 2B + 100$ 。图 7-6 的调度是并发控制机制必须避免的行为类型。□

$T_1$	$T_2$	$A$	$B$
		25	25
	READ( $A, s$ )		
	$s := s*2$		
	WRITE( $A, s$ )	50	
	READ( $B, s$ )		
	$s := s*2$		
	WRITE( $B, s$ )		50
READ( $A, t$ )			
$t := t+100$			
WRITE( $A, t$ )		150	
READ( $B, t$ )			
$t := t+100$			
WRITE( $B, t$ )			150

图 7-4  $T_2$  在  $T_1$  前的串行调度