**18.2.1**

ELt the database be in any arbitrary state

Consider, :=c1,B:=c2

The consistency constraint is A=c1+5

B=c2\*6

1. **Serial Order 1:**

|  |  |  |  |
| --- | --- | --- | --- |
| **T1** | **T2** | **A** | **B** |
| **Read(A,t)** |  | **10** | **20** |
| **t:=t+2** |  |  |  |
| **Write(A,t)** |  | **12** |  |
| **Read(B,t)** |  |  |  |
| **t:=t\*3** |  |  |  |
| **Write(B,t)** |  |  | **60** |
|  | **Read(B,s)** |  |  |
|  | **s:=s\*2** |  |  |
|  | **Write(B,s)** |  | **120** |
|  | **Read(A,s)** |  |  |
|  | **s:=s+3** |  |  |
|  | **Write(A,s)** | **15** |  |

**Serial order 2:**

|  |  |  |  |
| --- | --- | --- | --- |
| **T1** | **T2** | **A** | **B** |
|  | **Read(B,s)** | **10** | **20** |
|  | **s:=s\*2** |  |  |
|  | **Write(B,s)** |  | **40** |
|  | **Read(A,s)** |  |  |
|  | **s:=s+3** |  |  |
|  | **Write(A,s)** | **13** |  |
| **Read(A,t)** |  |  |  |
| **t:=t+2** |  |  |  |
| **Write(A,t)** |  | **15** |  |
| **Read(B,t)** |  |  |  |
| **t:=t\*3** |  |  |  |
| **Write(B,t)** |  |  | **120** |

Hence the consistency of the database is maintained in both the serial orders from any arbitrary initial database stat

1. **Serializable schedules :**

|  |  |  |  |
| --- | --- | --- | --- |
| **T1** | **T2** | **A** | **B** |
| **Read(A,t)** |  | **10** | **20** |
| **t:=t+2** |  |  |  |
| **Write(A,t)** |  | **12** |  |
|  | **Read(B,s)** |  |  |
|  | **s:=s\*2** |  |  |
|  | **Write(B,s)** |  | **40** |
| **Read(B,t)** |  |  |  |
| **t:=t\*3** |  |  |  |
| **Write(B,t)** |  | **120** |  |
|  | **Read(A,s)** |  |  |
|  | **s:=s+3** |  |  |
|  | **Write(A,s)** |  | **15** |

|  |  |  |  |
| --- | --- | --- | --- |
| **T1** | **T2** | **A** | **B** |
|  | **Read(B,s)** | **10** | **20** |
|  | **s:=s\*2** |  |  |
|  | **Write(B,s)** |  | **40** |
| **Read(A,t)** |  |  |  |
| **t:=t+2** |  | **12** |  |
| **Write(A,t)** |  |  | **40** |
|  | **Read(A,s)** |  |  |
|  | **s:=s+3** |  |  |
|  | **Write(A,s)** | **15** |  |
| **Read(B,t)** |  |  |  |
| **t:=t\*3** |  |  |  |
| **Write(B,t)** |  |  | **120** |

|  |  |  |  |
| --- | --- | --- | --- |
| **T1** | **T2** | **A** | **B** |
|  | **Read(B,s)** | **10** | **20** |
|  | **s:=s\*2** |  |  |
|  | **Write(B,s)** |  | **40** |
| **Read(A,t)** |  |  |  |
| **t:=t+2** |  |  |  |
| **Write(A,t)** |  | **12** | **40** |
| **Read(B,t)** |  |  |  |
| **t:=t\*3** |  |  |  |
| **Write(B,t)** |  |  | **120** |
|  | **Read(A,s)** |  |  |
|  | **s:=s+3** |  |  |
|  | **Write(A,s)** |  |  |

**Non serializable schedules :**

|  |  |  |  |
| --- | --- | --- | --- |
| **T1** | **T2** | **A** | **B** |
| **Read(A,t)** |  | **10** | **20** |
| **t:=t+2** |  |  |  |
| **Write(A,t)** |  | **12** |  |
|  | **Read(B,s)** |  |  |
|  | **s:=s\*2** |  |  |
| **Read(B,t)** |  |  |  |
| **t:=t\*3** |  |  |  |
| **Write(B,t)** |  |  | **60** |
|  | **Write(B,s)** |  | **40** |
|  | **Read(A,s)** |  |  |
|  | **s:=s+3** |  |  |
|  | **Write(A,s)** | **15** |  |

**2 mre examples:**

**18.2.4:**

**a) r1(A);r2(A);r3(B);w1(A);r2(C);r2(B);w2(B);w1(C)**

2

1

3

* There nodes correspond to the three transaction 1,2 and 3
* There is a conflict between r3(B) and w2(B) .
* Transaction 3 must read database element B before transaction 2 writes the database elemnt B
* Hence there is a directed edge from transaction 3 to 2 to indicate that read in transaction must precede the write in transaction 2 to ensure that the database consistency is preserved.
* Similarly, the read of database element C in transaction 2 must precede the write of DB element C in transaction 1 to ensure that the database consistency is preserved.

**There are no cycles in the precedence graph hence it is conflict serializable**

**The possible serial schedule is as follows:**

**After multiple swaps we get**

r1(A);r2(A);w1(A);r2(C);r2(B);r3(B);w2(B);w1(C);

**It is conflict serializable but not conflict equivalent**

**b)r1(A);w1(B);r2(B);w2(C);r3(C);w3(A)**

2

1

3

* There nodes correspond to the three transaction 1,2 and 3
* There is a conflict between r1(a) and w3(A)
* Transaction 1 must read database element A before transaction 3 writes the database element A
* Hence there is a directed edge from transaction 1 to 3 to indicate that read in transaction 1 must precede the write in transaction 3 to ensure that the database consistency is preserved.
* Similarly, the write of database element B in transaction 1 must precede the read of DB element B in transaction 2 to ensure that the database consistency is preserved, hence a directed edge from node 1 to 2
* Similarly, the write of database element C in transaction 2 must precede the read of DB element C in transaction 3 to ensure that the database consistency is preserved, hence a directed edge from node 2 to 3

**There are no cycles in the precedence graph hence it is conflict serializable**

**Conflict serializable schedule:**

**r1(A);w1(B);r2(B);w2(C);r3(C);w3(A)**

**No swaps required, as it is already serial**

1. **w3(A);r1(A);w1(B);r2(B);w2(C);r3(C);**

2

1

3

* There nodes correspond to the three transaction 1,2,and 3
* There is a conflict between w3(A) and r1(A)
* Transaction 3 must write database element A before transaction 1 reads the database element A
* Hence there is a directed edge from transaction 3 to 1 to indicate that write in transaction 3 must precede the read in transaction 1 to ensure that the database consistency is preserved.
* Similarly, the write of database element B in transaction 1 must precede the read of DB element B in transaction 2 to ensure that the database consistency is preserved, hence a directed edge from node 1 to 2
* Similarly, the write of database element C in transaction 2 must precede the read of DB element C in transaction 3 to ensure that the database consistency is preserved, hence a directed edge from node 2 to 3

**There is a cycle in the precedence graph hence the schedule is not conflict serializable**

**d)r1(A);r2(A);w1(B);w2(B);r1(B);r2(B);w2(C);w1(D);**

1

2

* There nodes correspond to the three transaction 1,2
* The w1(B) must precede r2(B) hence the directed edge from 1 to 2
* W2(B) must precede r1(B) hence the directed edge from 2 to 1
* **There is a cycle in the precedence graph hence the schedule is not conflict serializable**

**e)r1(A);r2(A);r1(B);r2(B);r3(A);r4(B);w1(A);w2(B);**

2

1

4

3

* There nodes correspond to the three transaction 1,2,3 and 4
* There is a conflict between w3(A) and r1(A)
* Transaction 3 must write database element A before transaction 1 reads the database element A
* Hence there is a directed edge from transaction 3 to 1 to indicate that write in transaction 3 must precede the read in transaction 1 to ensure that the database consistency is preserved.
* Similarly, the write of database element B in transaction 1 must precede the read of DB element B in transaction 2 to ensure that the database consistency is preserved, hence a directed edge from node 1 to 2
* Similarly, the read of database element B in transaction 4 must precede the write of DB element B in transaction 2 to ensure that the database consistency is preserved, hence a directed edge from node 4 to 2
* Similarly, the read of database element A in transaction 2 must precede the write of DB element A in transaction 1 to ensure that the database consistency is preserved, hence a directed edge from node 2 to 1
* **There is a cycle in the precedence graph hence the schedule is not conflict serializable**

**18.3.4:**

**a) r1(A);w1(B)**

**2 phase locking has ensures that all the requests for locks are performed prior to thw unlocks**

li(A)- represents request from transaction ‘i ‘ to lock database element A

ui(A)- represents request from transaction ‘i ‘ to unlock database element A

I nthis case threis only one transaction 1

**Consistent and 2 phase locking:**

**Possible Schedules:**

Consider this,

l1(A); r1(A);u1(A); and l1(B);w1(B);u1(B);

The locking of database elements A and B is consistent if the unlocking of the elements is surrounded by reads and writes . Hence there are many explicitly two ways this can be done in 2 ways

**i)**

|  |  |
| --- | --- |
| Possibility 1 | l1(A); r1(A);u1(A); l1(B);w1(B);u1(B); |
| Possibility 2 | l1(B);w1(B);u1(B); l1(A); r1(A);u1(A); |
| 2 ways to get consistent and non 2 phase locking |  |

**ii**) The consistent sequences can be formed by interleaving consistent actions**.** There are six actions in total in the transaction. Taking 3 transactions at a time and rearranging them gives 20 possibilities

Since two of them are non 2 phase locking ( from case i) , we get 18 possibilities

**b)** r2(A);w2(A);w2(B);

**i)**

|  |  |
| --- | --- |
| Possibility 1 | l2(A); r2(A);w2(A); u2(A); l2(B);w2(B);u2(B); |
| Possibility 2 | l2(B);w2(B);u2(B); l2(A); r2(A);w2(A); u2(A); |
|  |  |
| 2 ways to get consistent and non 2 phase locking |  |

**ii)** The consistent sequences can be formed by interleaving consistent actions**.** There are six actions in total in the transaction. Taking 3 transactions at a time and rearranging them gives 20 possibilities

Since two of them are non 2 phase locking ( from case i) , we get 18 possibilities

**18.4.1:**

**a)** *r1(A);r2(B);r3(C);w1(B);w2(C);w3(D);*

*i)*

*xl- Exclusive lock*

*sl- Shared lock*

*xl1(A); r1(A); xl2(B); r2(B); xl3(C); r3(C); sl1(B); r1(B); sl2(C); r2(C); sl3(D); r3(D); w1(A); u1(A); u1(B); w2(B); u2(B); u2(C); w3(C); u3(C); u3(D)*

* *Here r1(A) is followed by w1(A) that is write action from the same transaction hence an exclusive lock is placed*
* *Similarly, r2(B) is followed by w2(B), r3(C) is followed by w3(C).*
* *Placing shared locks:*
* *R1(B) is not followed by a write action from the same transaction hence we place a shared lock*
* *Similarly for r2(C),r3(D)*
* *Once the w1(A) action is completed we can unlock A*
* *Unlocking is done based on the completion of the transactions*

*d)r1(A);r2(B);r3(C);r1(B);r2(C);r3(D);w1(A);w2(B);w3(C);*

**i)** xl1(A);

r1(A);

xl2(B);

r2(B);

xl3(C);

r3(C);

sl1(B);

r1(B);

sl2(C);

r2(C);

sl3(D);

r3(D);

w1(A);

u1(A);

u1(B);

w2(B);

u2(B);

u2(C);

w3(C);

u3(C);

u3(D);

**18.4.1:r1(A);r2(B); ??? w1(C) ;w2(A)**

The schedule becomes non serializable if two actions from different transactions worign on the sme dataset element and involving atleast one write are swapped in appropriately

**a)Read**

Possibility 1:r1(A); r2(B); **r2(C)** ;w1(C); w2(A);

Possibility 1:r1(A); r2(B); **r1(A)** ;w1(C); w2(A);

**b) Write**

Possibility 1: r1(A);r2(B) ; **w1(A)**; w1(C) ; w2(A);

Possibility 1:r1(A); r2(B); **w2(C)** ;w2(A);

Since A is read by the transaction 1 a new value written to A by transaction 1 can affect the value written by transaction 2 to A

**c)Update**

Possibility 1: r1(A);r2(B) ;**ul2(A)** ; w1(C) ; w2(A);

**d)Increment**

r1(A);r2(B) ; **Inc2(C,c)**; w1(C) ; w2(A); Incrementing C by a constant value ‘c’

r1(A);r2(B) ; **Inc1(A,c)**; w1(C) ; w2(A); 🡪 Incrementing A by a constant value ‘c’

Incrementing the value of a a database element before it is written by another transaction it gives a different final value of the database element

**18.8.1 :**

1. **St1;st2;r1(A);r2(B);w2(A);w1(B);**

* St1 and st2 indicate the start time stamps of each of the transactions
* It is when the the time stamps start getting assigned to the transactions

|  |  |  |  |
| --- | --- | --- | --- |
| **T1** | **T2** | **A** | **B** |
| **100** | **100** | **RT=0;WT=0** | **RT=0;WT=0** |
| **R1(A)** | **R2(B)** |  |  |
| **W1(B)** | **W2(A)** |  |  |

The two transactions execute in parallel but no conflict arises as they manipulate different element in two different transactions

1. **st1;st3;st2;r1(A);r2(B);w1(C);r3(B);r3(C);w2(B);w3(A);**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| T1 | T2 | T3 | A | B | C |
| 100 | 100 | 100 | RT=0;WT=0 | RT=0;WT=0 | RT=0;WT=0 |
| R1(A) | R2(B) |  |  |  |  |
| W1(C) |  | R3(B) |  |  |  |
|  | W2(B) | R3(C) |  |  |  |
|  |  | W3(A) |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

There is a possible conflict at W1(c) and R3(C) in case the order is changed since the same databse element is operated upon by two different transactions

**d)**