CSCI235 Database Systems

Introduction to Transaction Processing (3)

Dr Janusz R. Getta

School of Computing and Information Technology -
University of Wollongong

Introduction to Transaction Processing

Outline

Serialization graph

Serialization graph is constructed in the following way

- If a transaction T participates in a concurrent execution then we add a node labeled with T to a serialization graph
- If the transactions T_i and T_j process conflicting operations such that T_i processes its operation first then we add an edge directed from T_i to T_j

Sample construction of a serialization graph

```
                                                Concurrent execution of database transactions
T1                                  T2                                  T3
```

```
write(x, 10)
```

Create a node T1

```
                                    write(x, 20)
```

Create a node T2 and add an edge from T1 to T2

Serialization graph

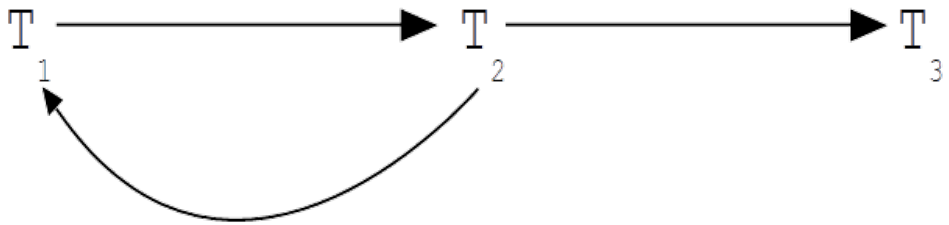
```
                                                write(x,30)
```

Create a node `T3` and add the edges from `T1` to `T3` and from `T2` to `T3`

```
                                write(y,10)
```

```
read(y)
```

Add an edge from `T2` to `T1`



Serialization graph

A serialization graph for nonconflict serializable execution of database transactions

```
                                        Nonconflict serializable execution of database transactions
T1                                                                  T2
```

```
read(x)
```

Create a node `T1`

```
                                                        read(x)
```

Create a node `T2`

```
write(x,x-10)
```

Add an edge from `T2` to `T1`

```
                                                        write(x,x+20)
```

Add an edge from `T1` to `T2`

Serialization graph

A serialization graph for nonconflict serializable execution of database transactions

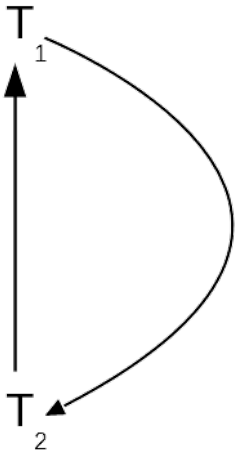
Nonconflict serializable execution of database transactions

```
T1                                  T2
read(x)

                                    read(x)

write(x,x-10)

                                    write(x,x+20)
```



Introduction to Transaction Processing

Outline

Serialization graph testing protocol (SGT)

Principles

- Scheduler maintains and tests serialization graph
- If an operation issued by a transaction violates conflict serializability, i.e. if it creates a cycle in serialization graph then such transaction is aborted

Problems

- Cascading aborts: if a transaction `T` that created a data item `x` is aborted then all transactions that read a new vlaue of `x` must be aborted
- Performance: testing acyclicity of serialization graph has O(n2) complexity

Introduction to Transaction Processing

Outline

Two-phase locking (2PL) protocol

Principles

- Access to data items is controlled by shared (read) and exclusive (write) locks
- A transaction can read a data item only if a data item is read or write locked by the same transaction or a data item is read locked by another transaction
- A transaction can write a data item only if a data item is write locked by the same same transaction
- Each transaction must acquire all locks before releasing any lock

Two-phase locking protocol belongs to a class of pessimistic protocols

Problems

- Deadlocks
- Unnecessary locks and delays when an execution is conflict serializable

Two-phase locking (2PL) protocol

A sample execution controlled by two-phase locking protocol

```
                    Concurrent execution of database transactions controlled by 2PL protocol
T1                                          T2
lock(u) read(u)
                                            lock(v) write(v,10)
write(u,u+2)
lock(v) wait
                                            lock(x) read(x)
                                            unlock(v)
                                            write(x,x+2)
lock(v)
                                            unlock(x)
write(v,v+1)
unlock(v)
unlock(u)
```

Two-phase locking (2PL) protocol

A sample execution that ends in a deadlock

Concurrent execution of database transactions that ends in a deadlock

T1	T2
lock(u) read(u)	
	lock(v) write(v,v+1)
lock(v) wait	
	lock(u) wait

Both transactions are in a wait state

In a database system deadlock is eliminated through either wait for graph or throught timeout

Two-phase locking (2PL) protocol

A sample execution that unnecessarily delays a transaction

```
T1                  Concurrent execution of database transactions that unnecessarily delays a transaction
lock(u) read(u)
                                                lock(u) wait
lock(v) write(v,v+1)
unlock(v,u)
                                                lock(u) write(u,u+1)
```

The transactions `T1` and `T2` never get into non conflict serializable execution

Therefore, there is no need to delay a transaction `T2`

Introduction to Transaction Processing

Outline

Timestamp ordering (TO) protocol

Principles

- Each transaction obtains a timestamp at its start point
- Data items are stamped each time any transaction accesses data items in a read or write mode
- Access to data items is permitted in an increasing order of timestamps

Timestamp ordering protocol belongs to a class of optimistic protocols

Problems

- Cascading aborts
- Unnecessary aborts when execution is conflict serializable

Timestamp ordering (TO) protocol

A sample execution controlled by the timestamp ordering protocol

Concurrent execution of database transactions controlled by TO protocol

T1	T2	x
timestamp(t1)		
read(x)		x:t1
write(x,x-10)		
	timestamp(t2)	
	write(x)	x:t1:t2
	read(y)	y:t2
read(y)		y:t2:t1
write(y,y+1)		y:t2:t1
abort		

Timestamp ordering (TO) protocol

A sample excution controlled by the timestamp ordering protocol with cascading abort

Concurrent execution of database transactions controlled by TO protocol with cascading abort

T1	T2	x
timestamp(t1)		
read(x)		x:t1
write(x,x-10)		
	timestamp(t2)	
	read(x)	x:t1:t2
fail		
	forced abort	

A transaction T2 is forced to abort because it reads a dirty data item written by a failed transaction T1

Timestamp ordering (TO) protocol

A sample excution controlled by the timestamp ordering protocol where a transaction is unnecessarily aborted

Concurrent execution of database transactions controlled by TO protocol with unnecessary abort

T1	T2	x
	timestamp(t2)	
timestamp(t1)		
read(x)		x:t1
write(x,x-10)		
	read(x) forced abort	x:t1:t2

A transaction T2 is aborted even the execution is conflict serializable

References

T. Connoly, C. Begg, Database Systems, A Practical Approach to Design, Implementation, and Management, Chapter 22.2 Concurrency Control, Pearson Education Ltd, 2015