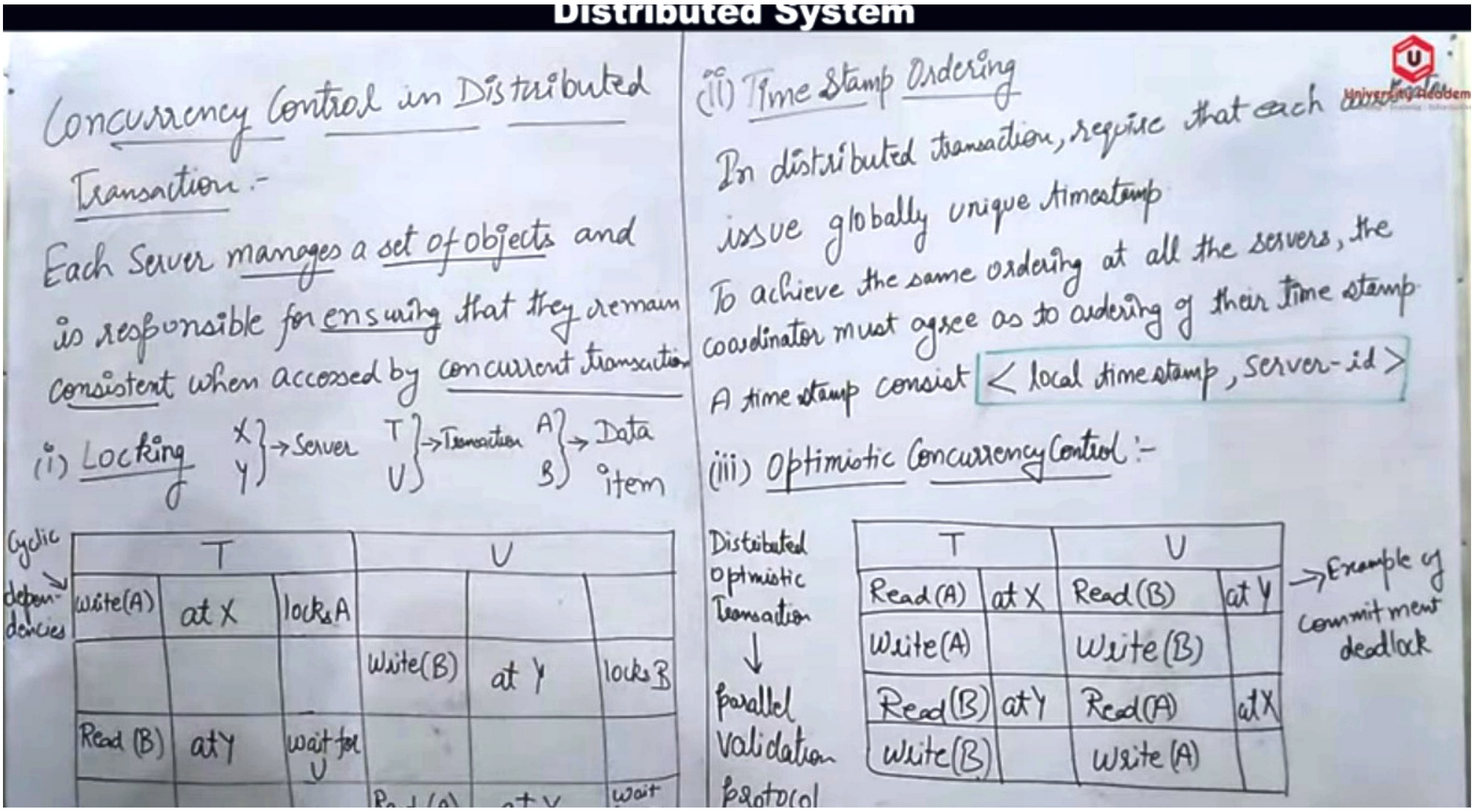


Distributed System

Concurrency Control in Distributed Transaction :-

Each Server manages a set of objects and is responsible for ensuring that they remain consistent when accessed by concurrent transaction

(i) Locking X, Y → Server; T, U → Transaction; A, B → Data item

Cyclic dependencies

T			U		
write(A)	at X	locks A			
			Write(B)	at Y	locks B
Read (B)	at Y	wait for U			
			Read (A)	at X	wait

(ii) Time Stamp Ordering

In distributed transaction, require that each [illegible] issue globally unique timestamp

To achieve the same ordering at all the servers, the coordinator must agree as to ordering of their time stamp

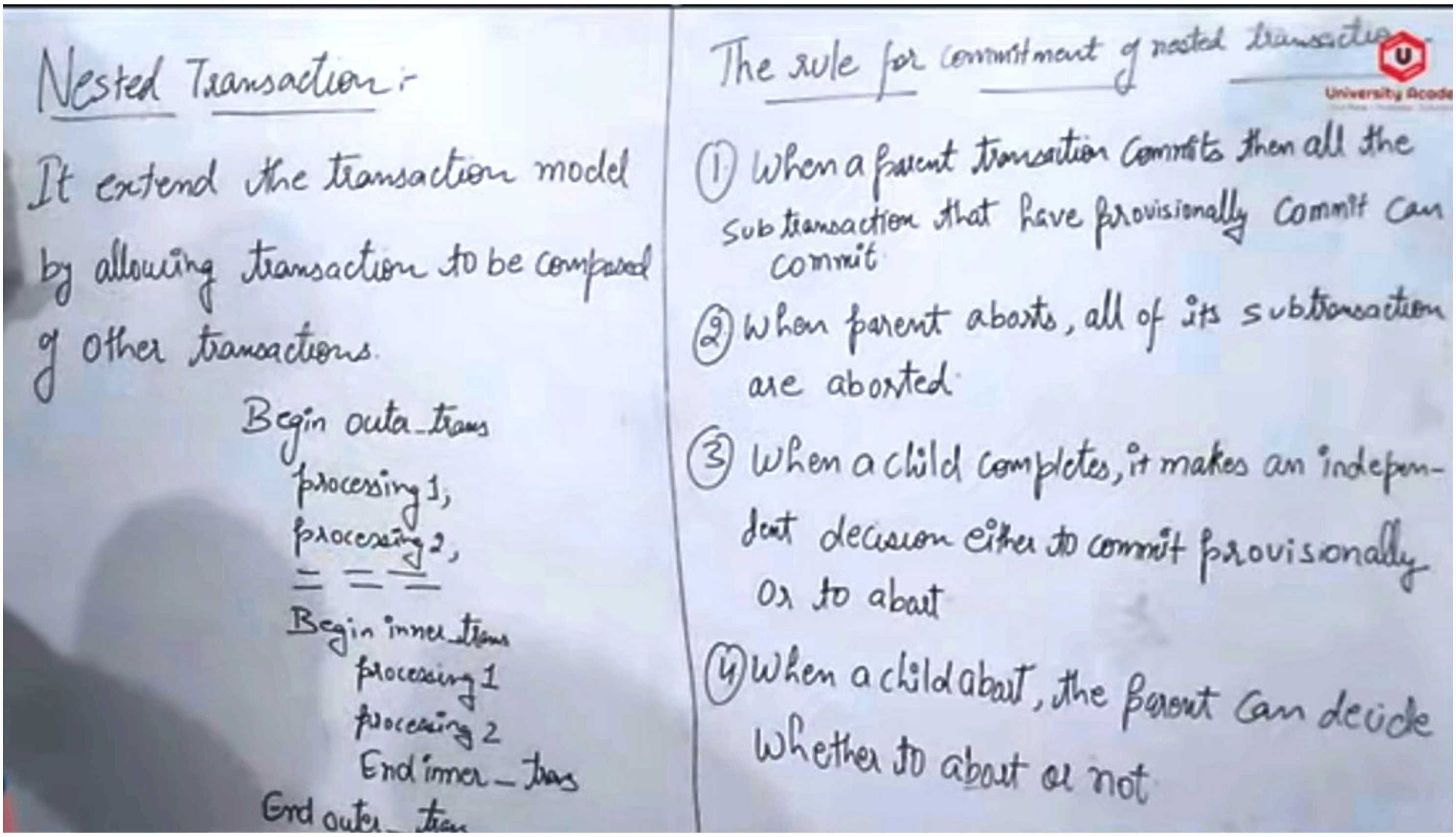
A time stamp consist < local time stamp, server-id >

(iii) Optimistic Concurrency Control :-

Distributed optimistic Transaction ↓ parallel Validation protocol

T		U	
Read (A)	at X	Read (B)	at Y
Write (A)		Write (B)	
Read (B)	at Y	Read (A)	at X
Write (B)		Write (A)	

→ Example of commitment deadlock



Nested Transaction :-

It extend the transaction model by allowing transaction to be composed of other transactions.

```
Begin outer_trans
    processing 1,
    processing 2,
    — — —
    Begin inner_trans
        processing 1
        processing 2
        End inner_trans
End outer_trans
```

The rule for commitment of nested transaction

1. When a parent transaction commits then all the subtransaction that have provisionally commit can commit
2. When parent aborts, all of its subtransaction are aborted
3. When a child completes, it makes an independent decision either to commit provisionally or to abort
4. When a child abort, the parent can decide whether to abort or not