CSCI235 Database Systems

Introduction to Transaction Processing (2)

Dr Janusz R. Getta

School of Computing and Information Technology -
University of Wollongong

Introduction to Transaction Processing

Outline

Correctness

What makes concurrent execution of database transaction incorrect ?

How do we define a correct concurrent execution of database transactions ?

Concurrent execution of database transactions is view serializable if there exists a possible serial execution of the same set of transactions such that in both executions each transaction reads the same values and the final states of the database are the same

A concurrent execution of database transactions is correct when it is view serializable

Correctness

A sample view serializable execution of database transactions

Concurrent processing of database transactions

T1	T2
a=read(x)	
write(x, a-10)	
	b=read(x)
c=read(y)	
write(y,c+10)	
	d=read(y)

The execution of database transactions above is view serializable because there exists a serial execution of the same transactions such that in both executions the transactions read the same values and the final states of the database are the same (see next slide)

Correctness

A sample serial execution equivalent to a concurrent execution from the previous slide

Concurrent processing of database transactions

T1	T2
a=read(x)	
write(x, a-10)	
c=read(y)	
write(y,c+10)	
	b=read(x)
	d=read(y)

Correctness

A problem with view serializability

- Verification that concurrent execution of Database transactions is view serializable is NP-complete
- It means that it takes to much time to check whether execution of a database operation violates view serializability correctness criterion

A more practical correctness criterion is conflict serializability

- Concurrent execution of database transactions is conflict serializable if there exists a possible serial execution of the same set of transactions such that in both executions the order of conflicting operations is the same

Which operations are conflicting operations ?

Two perations are conflicting operations if both access the same data item and one or both of them is `write` operation

	read	write	Conflicting operations
read	NO	YES	
write	YES	YES	

Correctness

Conflicting operations in a sample concurrent execution of transactions

```
                                                    Conflicting operations
T1                                  T2
write(x, a-10)
                                    b=read(x)
```

The operations `write(x, a-10)` and `b=read(x)` are conflicting operations because both access the same data item `x` and one of them is `write`

```
                                                    Conflicting operations
read(y)
write(y,a+10)
                                    write(y,b-10)
```

The operations `write(y,a+10)` and `write(y,b-10)` are conflicting operations because both access the same data item `y` and both of them are `write`

Correctness

Concurrent execution of database transactions is conflict serializable if there exists a possible serial execution of the same set of transactions such that in both executions the order of conflicting operations is the same

A sample conflict serializable execution of database transactions

Conflict serializable execution of database transactions

T1	T2
a=read(x)	
write(x, a-10)	
	b=read(x)
c=read(y)	
write(y,c+10)	
	d=read(y)

Order of conflicting operations: `T1 before T2`

Correctness

A sample not conflict serializable execution of database transactions

Not cconflict serializable execution of database transactions

T1	T2	x: $100
a=read(x)		x: $100 a: $100
	b=read(x)	x: $100 a: $100 b: $100
write(x,a-10)		x: $90 a: $100 b: $100
	write(x,b+20)	x: $120 a: $100 b: $100
	commit	x: $120 a: $100 b: $100
commit		x: $120 a: $100 b: $100

Order of conflicting operations: `T1 before T2` and `T2 before T1` means that it is impossible to serialize the concurrent execution of `T1` and `T2`

It means that the concurrent execution of database transactions `T1` and `T2` is incorrect

Introduction to Transaction Processing

Outline

Conflict serializability versus view serializability

Every conflict serializable execution is view serializable

A view serializable execution may not be conflict serializable

An execution below is view serializable because there exists equivalent serial execution where each transaction reads and writes the same data items, see next slide

View serializable execution of database transactions

T1	T2	T3	
write(x, 10)			x: 10
	write(x, 20)		x: 20
		write(x,30)	x: 30
	write(y,10)		x: 30 y: 10
a=read(y)			x: 30 y: 10 a: 10

Conflict serializability versus view serializability

Equivalent serial execution

		Serial execution of database transactions
T1	T2	T3
	write(x, 20)	x: 20
	write(y, 10)	x: 20 y: 10
write(x,10)		x: 10 y: 10
a=read(y)		x: 10 y: 10 a: 10
	write(x,30)	x: 30 y: 10 a: 10

Hence, the original execution is view serializable

Conflict serializability versus view serializability

However, the original execution is not conflict serializable

Not conflict serializable execution of database transactions

T1	T2	T3
write(x, 10)		
	write(x, 20)	
		write(x,30)
	write(y,10)	
a=read(y)		

This is because `T1` processes a conflicting operation `write(x, 10)` before `T2` processes `write(x,20)` and `T2` processes a conflicting operation `write(y, 10)` before `T1` processes `a=read(y)`

Hence, the execution is not conflict serializable

Introduction to Transaction Processing

Outline

Order preserving serializability

Concurrent execution of database transactions is order-preserving conflict serializable if it is

- conflict serializable and
- all non-interleaved transactions have the same order in both original execution and some corresponding serial execution

Every order-preserving conflict serializable execution is conflict serializable

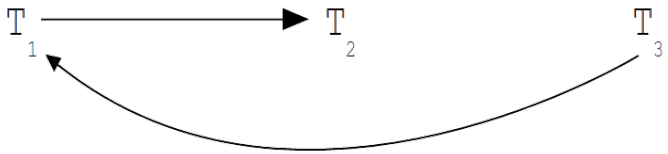
A conflict serializable execution may not be order-preserving conflict serializable

Order preserving serializability

A sample conflict serializable and order-preserving serializable execution

Order-preserving execution of database transactions

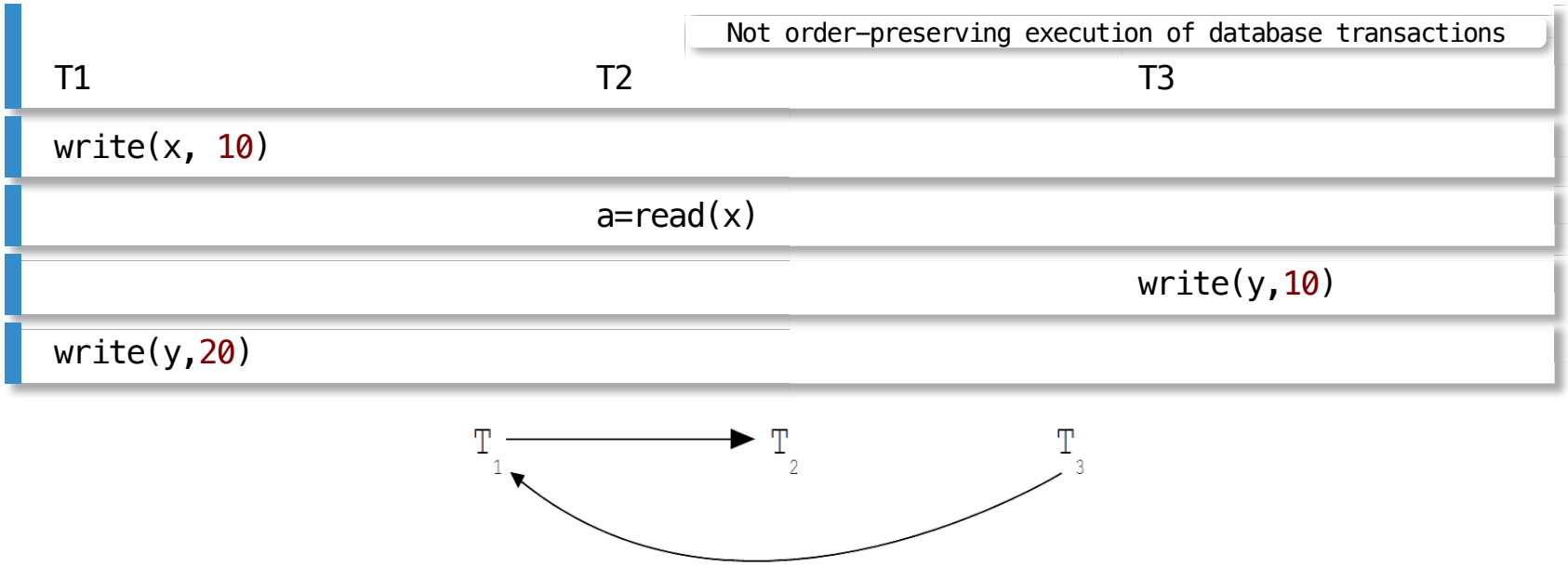
T1	T2	T3
		write(y,10)
write(x, 10)		
	a=read(x)	
write(y,20)		



Order of transactions indicated by their start timestamps is the same as serialization order (`T3` before `T1` and `T1` before `T2`)

Order preserving serializability

A sample conflict serializable and not order-preserving serializable execution



The execution is not order-preserving serializable because order of transactions indicated by their start timestamps (`T1` before `T2` and `T2` before `T3`) is different from serialization order (`T3` before `T1` and `T1` before `T2`)

Introduction to Transaction Processing

Outline

Recoverable executions

Transaction manager must provide the all-or-nothing property of transactions even in the presence of various types of failures

Execution is recoverable if every transaction `T` that reads a data item written by another transaction `T'` commits after `T'` is committed

A sample not recoverable execution

Not recoverable execution of database transactions

T1	T2	x: $100
a=read(x)		x: $100 a: $100
write(x,a-10)		x: $90 a: $100
	b=read(x)	x: $90 a: $100 b: $90
	write(x,b+20)	x: $110 a: $100 b: $90
	commit	x: $110 a: $100 b: $90
abort		x: $100 a: $100 b: $90

Introduction to Transaction Processing

Outline

Cascadeless executions

Execution is cascadeless if none of the transactions reads data item written by any other transaction that is not committed or aborted

A sample cascadeless execution

Cascadeless execution of database transactions

T1	T2	x: $100
a=read(x)		x: $100 a: $100
write(x,a-10)		x: $90 a: $100
	b=read(x) wait	x: $90 a: $100
abort		x: $100 a: $100
	b=read(x)	x: $100 a: $100 b: $100
	write(x,b+20)	x: $120 a: $100 b: $100

Cascadeless executions

A sample not cascadeless execution

Not cascadeless execution of database transactions

T1	T2	x: $100
a=read(x)		x: $100 a: $100
write(x,a-10)		x: $90 a: $100
	b=read(x)	x: $90 a: $100 b: $90
	write(x,b+20)	x: $110 a: $100 b: $90
abort		x: $100 b: $90
	forced abort	x: $100

Introduction to Transaction Processing

Outline

Strict executions

Execution is strict if any data item `d` is written by transaction `T` then any other transaction cannot either read or write data item `d` until `T` is committed or aborted

A sample strict execution

Strict execution of database transactions

T1	T2	x: $100
a=read(x)		x: $100 a: $100
write(x, a+10)		x: $110 a: $100
	write(x,20) wait	x: $110 a: $100
abort		x: $100
	write(x,20)	x: $20
	commit	x: $20

Strict executions

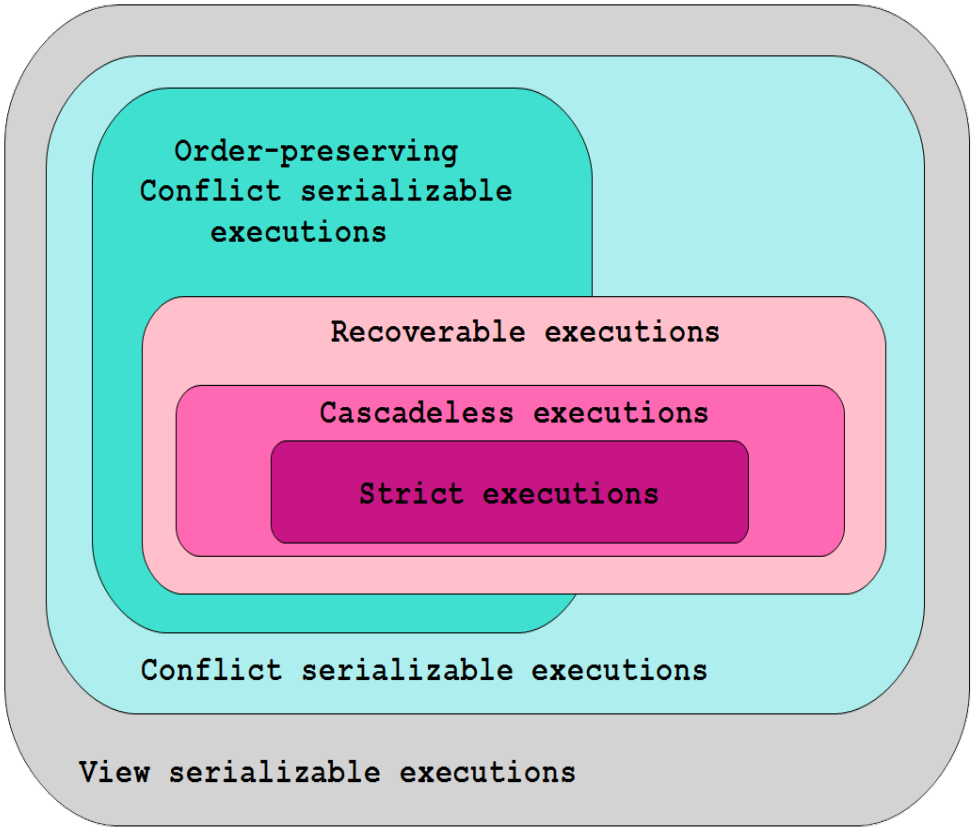
A sample not strict execution

Not strict execution of database transactions

T1	T2	x: $100
a=read(x)		x: $100 a: $100
write(x, a+10)		x: $110 a: $100
	write(x,20)	x: $20 a: $100
	commit	x: $20
abort		x $100

Rollback of `T1` destroys committed `T2 :write(x,20)`

Summary


Order-preserving
Conflict serializable
executions
Recoverable executions
Cascadeless executions
Strict executions
Conflict serializable executions
View serializable executions

References

T. Connoly, C. Begg, Database Systems, A Practical Approach to Design, Implementation, and Management, Chapter 22.1 Transaction Support, Chapter 22.2 Concurrency Control, Pearson Education Ltd, 2015