SFU SIMON FRASER UNIVERSITY
ENGAGING THE WORLD

Database Systems I

Transaction Processing (2)

Instructor: Ouldooz Baghban Karimi

CMPT 354 - Summer 2019

Transactions (ACID)

Atomic
The all-or-nothing execution of transactions

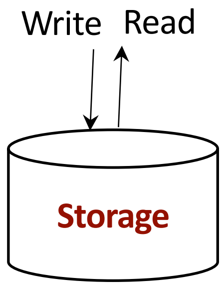
Consistent
Transactions are expected to preserve the consistency and integrity of the database

Isolated
Transactions to be executed as if no other transaction is executing at the same time

Durable
Once a transactions has committed, its effects remain in the database

Transaction Management

Definition



Transaction = A list of writes and reads

For example: {Read, Read, Write}

Two Big Problems

1. Support multiple transaction at the same time

2. Make sure the data stored is reliable

Techniques

1. Concurrency Control

2. Database Recovery

Properties

Atomicity

Consistency

Isolation

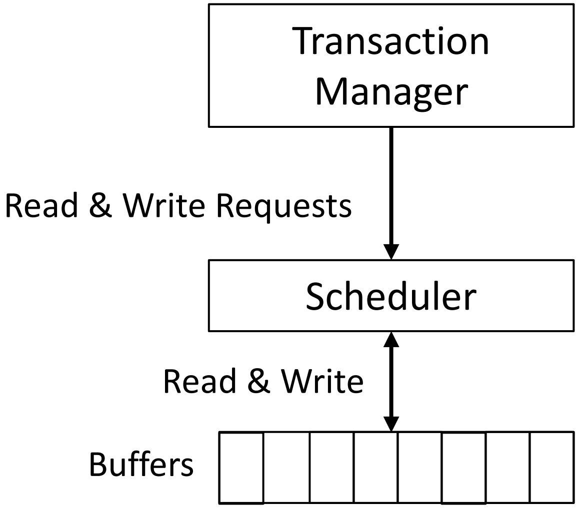
Durability

Transaction Management

- Concurrency Control
  - Consistency
  - Isolation
- Database Recovery
  - Atomicity
  - Durability

Concurrency Control

- **Schedule**: Sequence of important actions taken by one or more transactions



Scheduler

T1	*T2*
`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)`

Example Transactions

From Chapter 18, The complete book

Scheduling

- A **serial schedule** is one that does not interleave the actions of different transactions

- A **serializable schedule** is a schedule that is equivalent to **some** serial schedule

- Two schedules are **equivalent** if, **for any database state**, the effect of executing them on database **is identical**

Example

T1	*T2*	*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

time

From Chapter 18, The complete book

Example

T1	*T2*	*A*	*B*
	`READ(A,s)`	25	25
	`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

time

From Chapter 18, The complete book

Example

T1	*T2*	*A*	*B*
`READ(A,t)`		25	25
`t:=t+100`			
`WRITE(A,t)`		125	
	`READ(A,s)`		
	`s:=s*2`		
	`WRITE(A,s)`	250	
`READ(B,t)`			
`t:=t+100`			
`WRITE(B,t)`			125
	`READ(B,s)`		
	`s:=s*2`		
	`WRITE(B,s)`		250

time

From Chapter 18, The complete book

Concurrency Control

- The DBMS has freedom to interleave transactions

- Must pick an **interleaving** or **schedule** such that **isolation** and **consistency** are maintained
  → **Serializable schedule**

- If schedule is different than **any serial order: Not serializable**

Interleaving: Anomalies

- Various anomalies that break isolation and serializability occur because of (or with) certain **conflicts** between interleaved transactions

- **Interleaving anomalies occur with or because of conflicts between transactions** (conflicts can also occur without causing anomalies)

Anomalies

- Dirty read
  (*Occurring with / because of a **WR conflict)***
- $w_1(A)$; $r_2(A)$; $w_2(A)$; $r_1(A)$;

Changed A
Data written by a transaction that has not yet committed
Dirty read is a read of dirty data written by another transaction
Risk: the transaction that wrote it might **abort**

- Unrepeatable read
  (*Occurring with / because of a **RW conflict)***
- $r_1(A)$; $r_2(A)$; $w_2(A)$; $r_1(A)$;

Changed A

- Lost update
  (*Occurring because of a **WW conflict)***
- $w_1(A)$; $w_2(A)$; $w_2(B)$; $w_1(B)$;

Changed B

Changed B

Conflicts

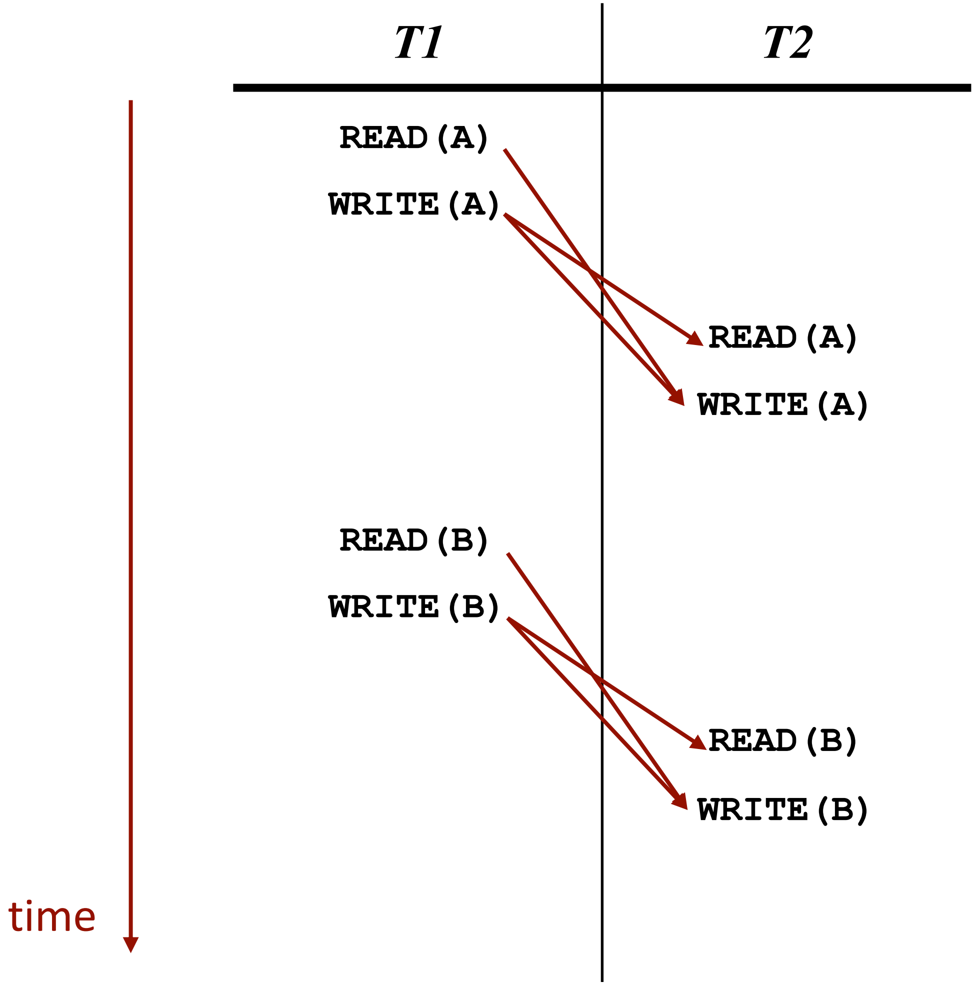
- Two actions **conflict** if
  - They are part of **different transactions**
  - Involve the **same** variable
  - At least one of them is a **write**

- Three types of conflicts
  - Read-Write conflicts (**RW**)
  - Write-Read conflicts (**WR**)
  - Write-Write conflicts (**WW**)

Example

- Two actions **conflict** if
  - They are part of **different transactions**
  - Involve the **same** variable
  - At least one of them is a **write**

- What are all the conflicts here?
- Why?



Interleaving: Anomalies

- Conflict serializability is not necessary for serializability

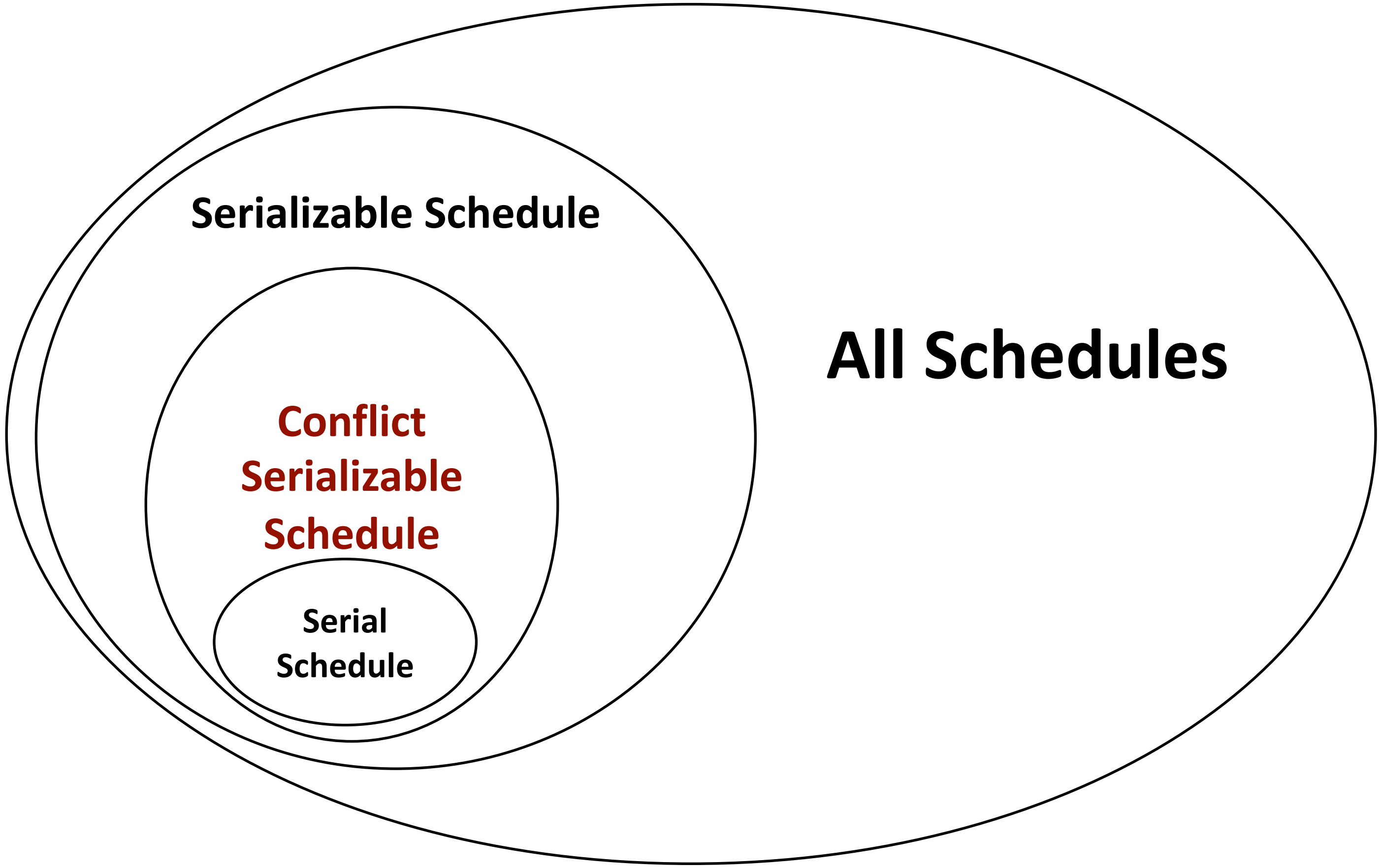


Figure From Slides, CMPT 354: Database Systems I (Jiannan Wang - SFU), Fall 2018

Conflict Serializing

- Schedule **S** is **conflict serializable** if S is conflict equivalent to some serial schedule

- Two schedules are conflict equivalent if
  - They involve the **same actions** of the **same transactions**
  - **Every pair** of conflicting actions of two transactions are **ordered in the same way**

Example

- Conflict equivalent?

T1	*T2*
READ(A)	
WRITE(A)	
READ(B)	
WRITE(B)	
	READ(A)
	WRITE(A)
	READ(B)

T1	*T2*
READ(A)	
WRITE(A)	
READ(B)	
WRITE(B)	READ(A)
	WRITE(A)
	READ(B)
	WRITE(B)

time

Example

- Conflict equivalent?

T1	*T2*
`READ(A)`	
`WRITE(A)`	
`READ(B)`	
`WRITE(B)`	
	`READ(A)`
	`WRITE(A)`
	`READ(B)`
	`WRITE(B)`

T1	*T2*
`READ(A)`	
`WRITE(A)`	
	`READ(A)`
	`WRITE(A)`
	`READ(B)`
	`WRITE(B)`
`READ(B)`	
`WRITE(B)`	

time

Example

- Conflict equivalent?

T1	*T2*
`READ(A)`	
`WRITE(A)`	
`READ(B)`	
`WRITE(B)`	
	`READ(A)`
	`WRITE(A)`
	`READ(B)`
	`WRITE(B)`

T1	*T2*
`READ(A)`	
`WRITE(A)`	
	`READ(A)`
	`WRITE(A)`
`READ(B)`	
`WRITE(B)`	
	`READ(B)`
	`WRITE(B)`

time

Conflict Graph

- **Conflict** Graph or **Precedence** Graph

- A graph with the **nodes as transactions**
- There is an edge from $T_i \rightarrow T_j$
  if any actions in T_i precede and conflict with any actions in T_j

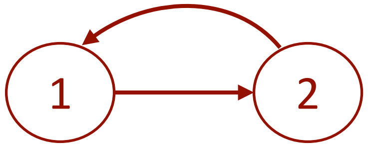
- **Theorem**: Schedule is **conflict serializable**
  if and only if
  its conflict graph is **acyclic**

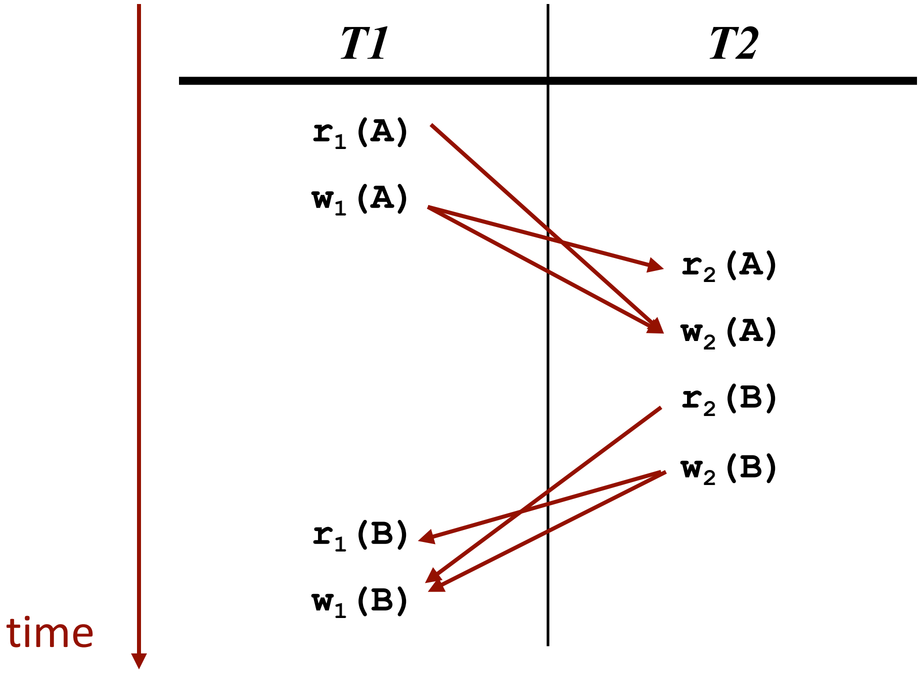
Conflict Serializable?

- Find all conflicts
- Model the schedule as a conflict graph
- Check whether the graph has a cycle
  - Yes → not conflict serializable
  - No → conflict serializable

Example

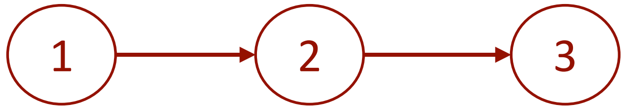
- $r_1(A)$; $w_1(A)$; $r_2(A)$; $w_2(A)$; $r_2(B)$; $w_2(B)$; $r_1(B)$; $w_1(B)$



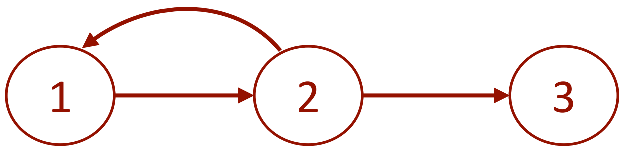


Example

- $r_2(A); r_1(B); w_2(A); r_3(A); w_1(B); w_3(A); r_2(B); w_2(B)$



- $r_2(A); r_1(B); w_2(A); r_2(B); r_3(A); w_1(B); w_3(A); w_2(B)$



From Chapter 18, The complete book

Concurrency Control Algorithms

- Locking
- Timestamp Ordering

Database Recovery

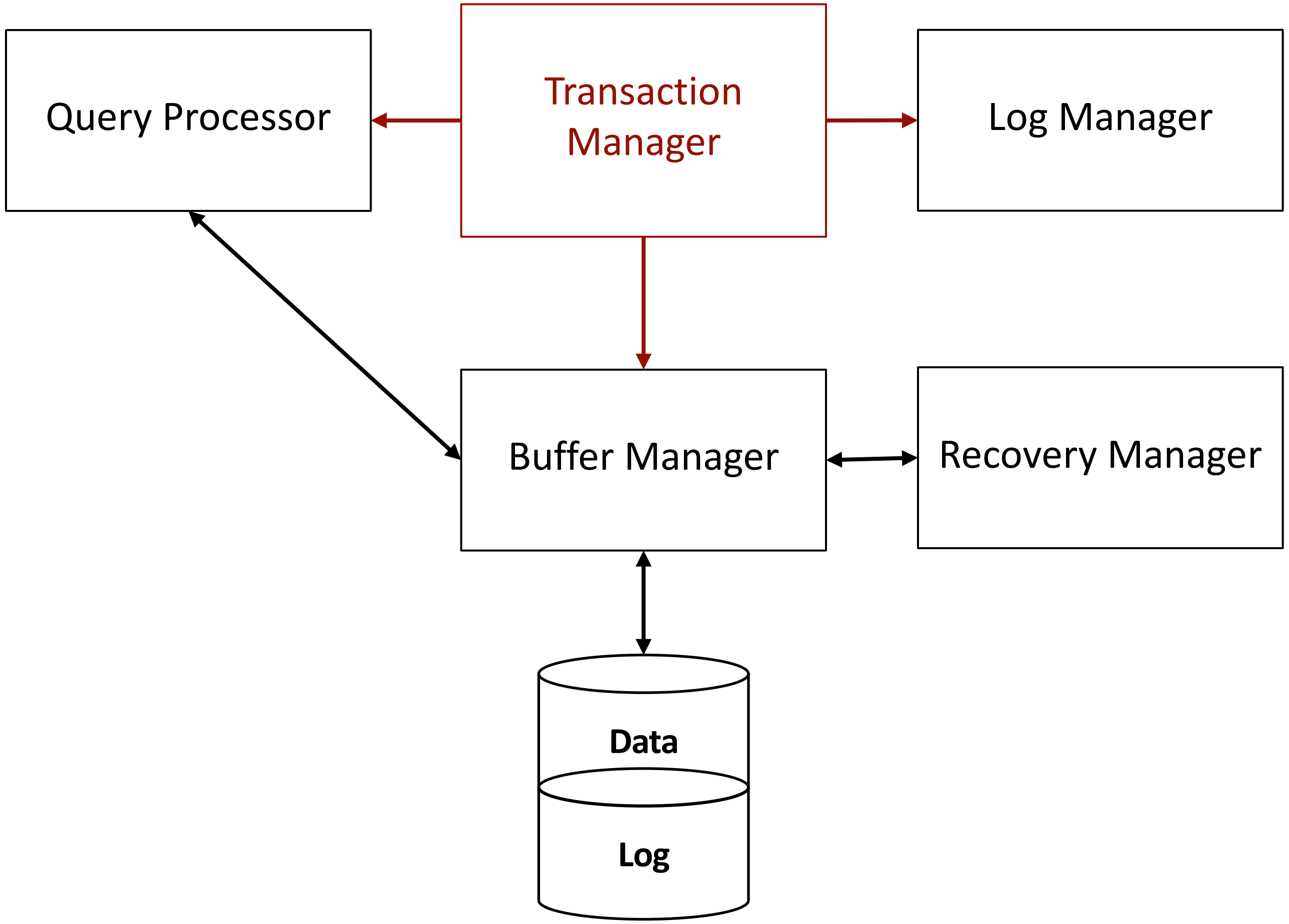
- Failures
  - Erroneous Data Entry
    - Solution: Constraints and triggers
  - Media Failures
    - RAID
    - Archive
  - Catastrophic Failure
    - Archive
    - Redundant distributed copies
  - System Failures
    - Logging
    - Checkpointing

Acknowledgements

I have used materials from the following resources in preparation of this course:

- **Database Systems: The Complete Book**
- Database Systems (Kifer, Bernstein, Lewis)
- Database System Concepts: https://www.db-book.com
- Course offerings
  - **CMPT 354 (Jiannan Wang - SFU): https://sfu-db.github.io/cmpt354/**
  - W 4111 (Eugene Wu - Columbia): https://w4111.github.io/
  - CS 245 (Matei Zaharia - Stanford): http://web.stanford.edu/class/cs245/
  - CS 186 (Joe Hellerstein - Berkeley): https://sites.google.com/site/cs186fall17/
  - CSE 344 (Dan Suciu - Washington): https://courses.cs.washington.edu/courses/cse344/17au/

Transaction Management



From Chapter 17, The complete book