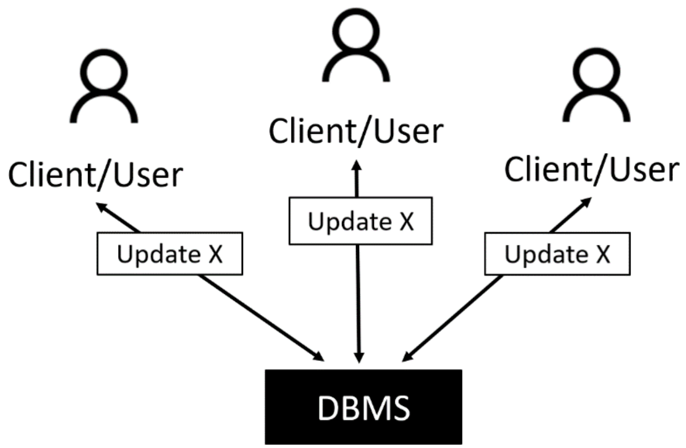
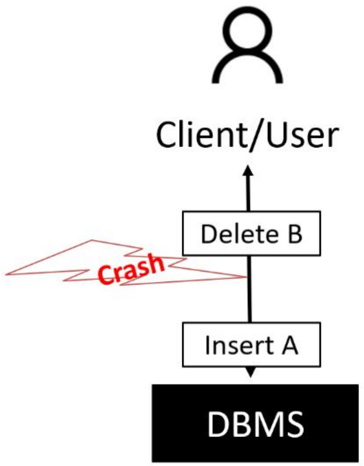
Introduction to Operating Systems and SQL for Data Science

Practice 12 – Transactions

Challenges in a real database

Challenges in a DB with **many users** that may **fail**





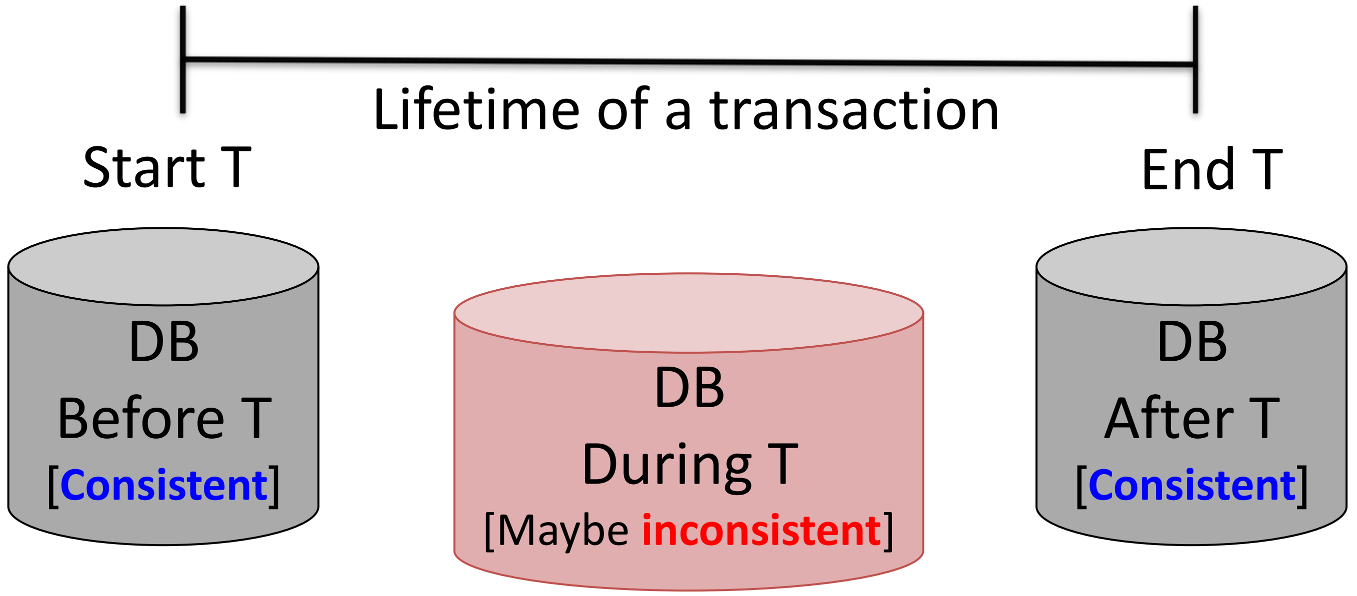
Transactions to save the day!

Transactions

Transactions to the Rescue!

Transaction: a **sequence of SQL operations** that are considered as a single unit

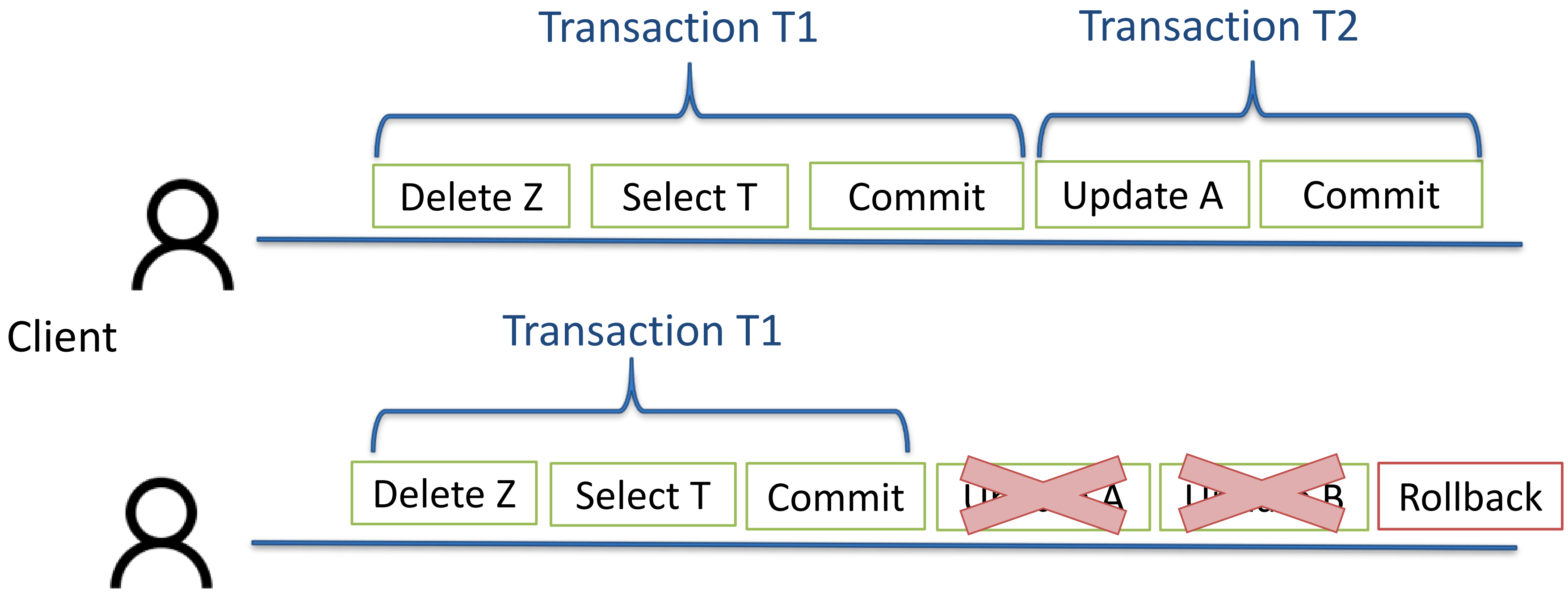
- In a transaction either **all actions are done or none**
- Transitions the DB from one consistent state to another



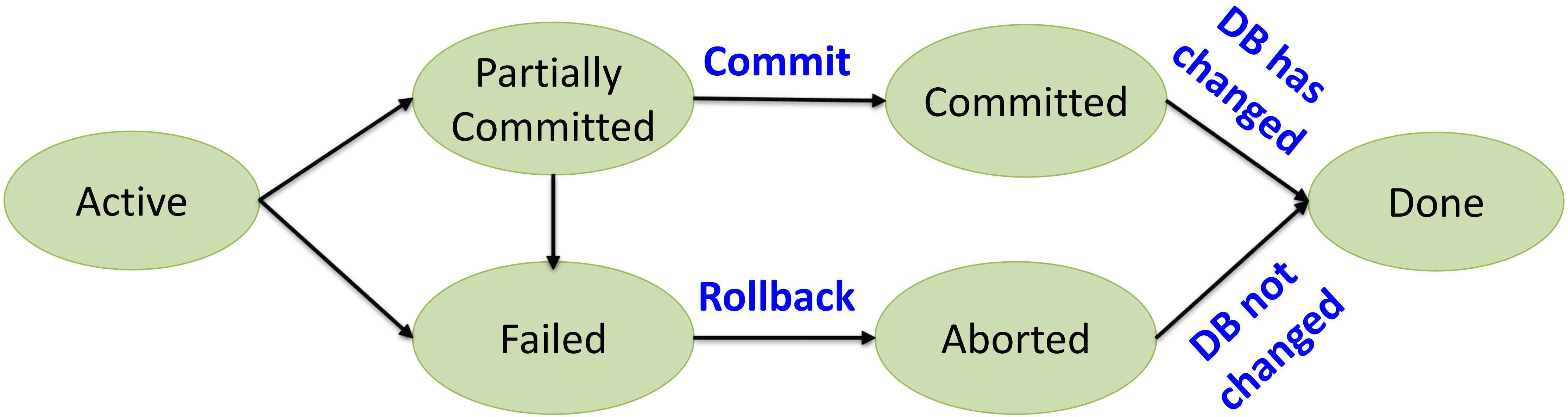
Transaction mechanism

- How we creates transactions?
  1. We apply SQL actions.
  2. When we want to “close” a Transaction we call this command **Commit**.
  3. If we want to “cancel” a started Transaction we do a **Rollback.**

- In many languages we can define auto commit



Transaction lifecycle



ACID: Desired Properties of Transactions

- Atomicity
- Consistency
- Isolation
- Durability

Serializable schedule

- A schedule is a serializable schedule if it is equivalent to some serial schedule.
- Need a way to determine that a schedule is serializable without knowing what calculations are being performed.
- That is, for every possible calculation, the timing should be equivalent to a serial schedule.

Reichman University

Equivalent schedules

Two schedules are equivalent if:

- They are composed of the same actions.
- Both schedules have the same effect on the database, meaning that the database has the same values at the end of each of the schedules.
- Both schedules produce the same output to the user, i.e., present the user with the same values for each item they read.

How to check if schedule is Conflict serializable?

- Definition: Precedence graph
  - Every transaction is a vertex.
  - There is an edge from T_i to T_j if and only if one of the following conditions is met:
    - T_i performs Read(x) before T_j performs Write (x)
    - T_i performs Write(x) before T_j performs Read(x)
    - T_i performs Write (x) before T_j performs Write(x)

    - Note: read(X) read(X) do not add an edge

Reichman University

How to check if schedule is conflict serializable?

- A Schedule is conflict serializable iff there are no circles in the precedence graph.

Example

Given 4 transaction over X Y Z P

$T_1 = R_1(X)\ W_1(X)\ R_1(P)\ W_1(P)$

$T_2 = R_2(Y)\ W_2(Y)\ R_2(P)$

$T_3 = R_3(Z)\ R_3(Y)\ W_3(Z)\ R_3(P)$

$T_4 = R_4(X)\ W_4(X)\ R_4(Z)$

And the following schedle S:

$R_4(X)W_4(X)R_1(X)W_1(X)R_2(Y)W_2(Y)R_1(P)$ ...

... $W_1(P)R_2(P)R_3(Z)R_3(Y)W_3(Z)R_4(Z)R_3(P)$

Build dependency graph of S

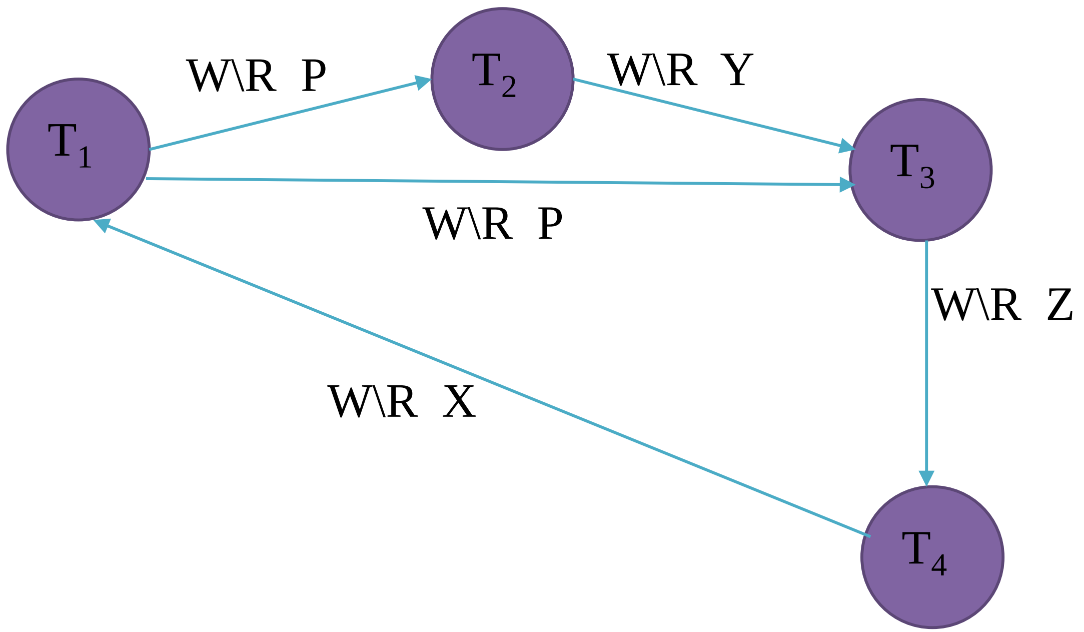
Reichman University

$$R_4(X)W_4(X)R_1(X)W_1(X)R_2(Y)W_2(Y)R_1(P)W_1(P)R_2(P)R_3(Z)R_3(Y)W_3(Z)R_4(Z)R_3(P)$$

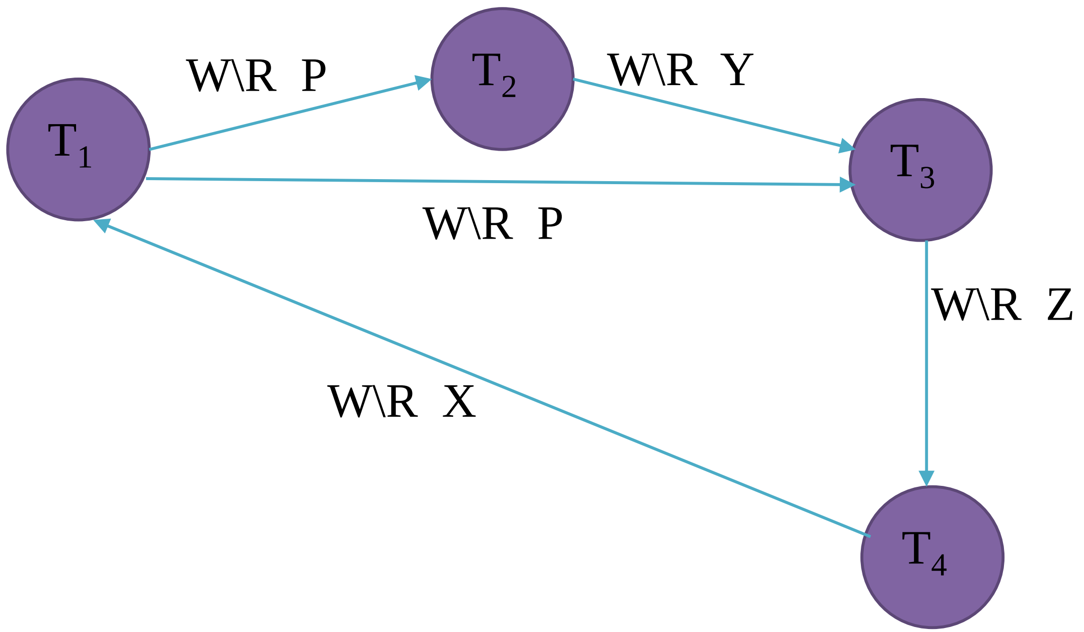
$$R_4(X)W_4(X)R_1(X)W_1(X)R_2(Y)W_2(Y)R_1(P)W_1(P)R_2(P)R_3(Z)R_3(Y)W_3(Z)R_4(Z)R_3(P)$$

T_4	T_3	T_2	T_1
READ X			
WRITE X			
			READ X
			WRITE X
		READ Y	
		WRITE Y	
			READ P
			WRITE P
		READ P	
	READ Z		
	READ Y		
	WRITE Z		
READ Z			
	READ P		

Reichman University



Is S conflict serializable?



Is S conflict serializable? Yes since there are no circles in the dependency graph

concurrency control: locks

Locks

Locks are data structures and protocols designed to prevent pre-creation of conflicts betwe transaction en transaction and prevent circuits in the conflict graph .

X-LOCK (Exclusive Lock): Write lock.
Only one transaction can hold a writing lock on the same item.

S-LOCK (Shared Lock): Read lock.
Several transactions can hold S-LOCK on the same item in parallel, as long as the item does not have a write lock (X-LOCK)

Reichman University

Access and lock conditions

- A transaction can write item A only after locking x-lock on A
- A transaction can read item A only after locking s-lock on A
- A transaction cannot x-lock A it there is already **any** lock on it
- A transaction cannot s-lock A if there is already x-lock on it
- Each transaction must release its locks before it finishes.

Two-Phase Locking (2PL)

Two-Phase Locking (2PL)

- A transaction holds 2PL requirements if:
  - It satisfied access and lock conditions
  - All locking actions happens **before** releases actions (once the transaction released a lock it cannot lock any longer)

- That is why the protocol is called "two-phase locking“
  - First a phase of making locks.
  - Followed by a phase of release of locks.

Reichman University

Two-Phase Locking (2PL)

- When all transactions in a schedule S are 2PL we say that the scheduling is 2PL.

Two-Phase Locking (2PL)

- When all transactions in a schedule S are 2PL we say that the scheduling is 2PL.

An example:

	T2	T1	
		READ X	XLOCK X
		WRITE X	SLOCK Z ULOCK X
XLOCK X	READ X		
XLOCK Y	READ Y		
	WRITE X		
		READ Z	ULOCK Z
ULOCK Y ULOCK X	WRITE Y		

The Downside of Using Locks Is …

Deadlocks

How handle deadlocks?

Detect & handle

Avoid

Reichman University

Deadlock

- A state of deadlock occurs when there is a circle of transactions, so that every transaction in the circle waits for a lock that is currently held by the next transaction in the circle.

- Two ways to deal with deadlock:
  - Prevention of deadlock.
  - Detection of deadlock and cancellation.

Reichman University

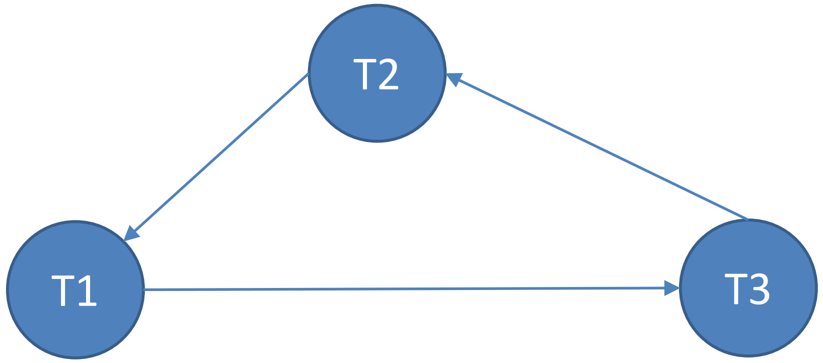
Deadlock detection

- We will draw a graph:
  - Transaction Ti as vertexes
  - Edge Ti->Tj if Ti waits for a lock held by Tj

There is a deadlock iff there is a circle in the graph

Deadlock detection

T1	T2	T3
X-LOCK A		
	X-LOCK B	
		X-LOCK C
S-LOCK C		
	X-LOCK A	
		S-LOCK B



Deadlock detection

Given the following 2 transactions:

T1: R(A), R(B), W(B)

T2: R(B), R(A), W(A)

a. Add lock and release actions so each transaction satisfied 2PL.

b. There is a schedule S with T1 and T2 that ends up with a deadlock?

Even if both transactions are 2PL. If so – give an example, if not- explain

Given the following 2 transactions:

T1: R(A), R(B), W(B)

T2: R(B), R(A), W(A)

a. Add lock and release actions so each transaction satisfied 2PL.

Given the following 2 transactions:

T1: R(A), R(B), W(B)

T2: R(B), R(A), W(A)

a. Add lock and release actions so each transaction satisfied 2PL.

T1	T2
S-LOCK (A)	S-LOCK (B)
R (A)	R (B)
X-LOCK (B)	X-LOCK (A)
R (B)	R (A)
W (B)	W (A)
UNLOCK (A)	UNLOCK (B)
UNLOCK (B)	UNLOCK (A)

b. There is a schedule S with T1 and T2 that ends up with a deadlock?

Even if both transactions are 2PL. If so – give an example, if not- explain

Yes. Look at the following schedule:

T1	T2
S-LOCK (A)	
	S-LOCK (B)
	R (B)
R (A)	
X-LOCK (B)	
	X-LOCK (A)
	R (A)
	W (A)
R (B)	
W (B)	
UNLOCK (A)	
UNLOCK (B)	
	UNLOCK (B)
	UNLOCK (A)

There is a circle in the graph => there is a deadlock