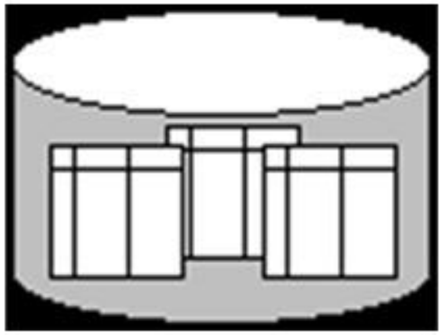
ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

Συστήματα Διαχείρισης Βάσεων Δεδομένων

Φροντιστήριο 10: Transactions - part 2

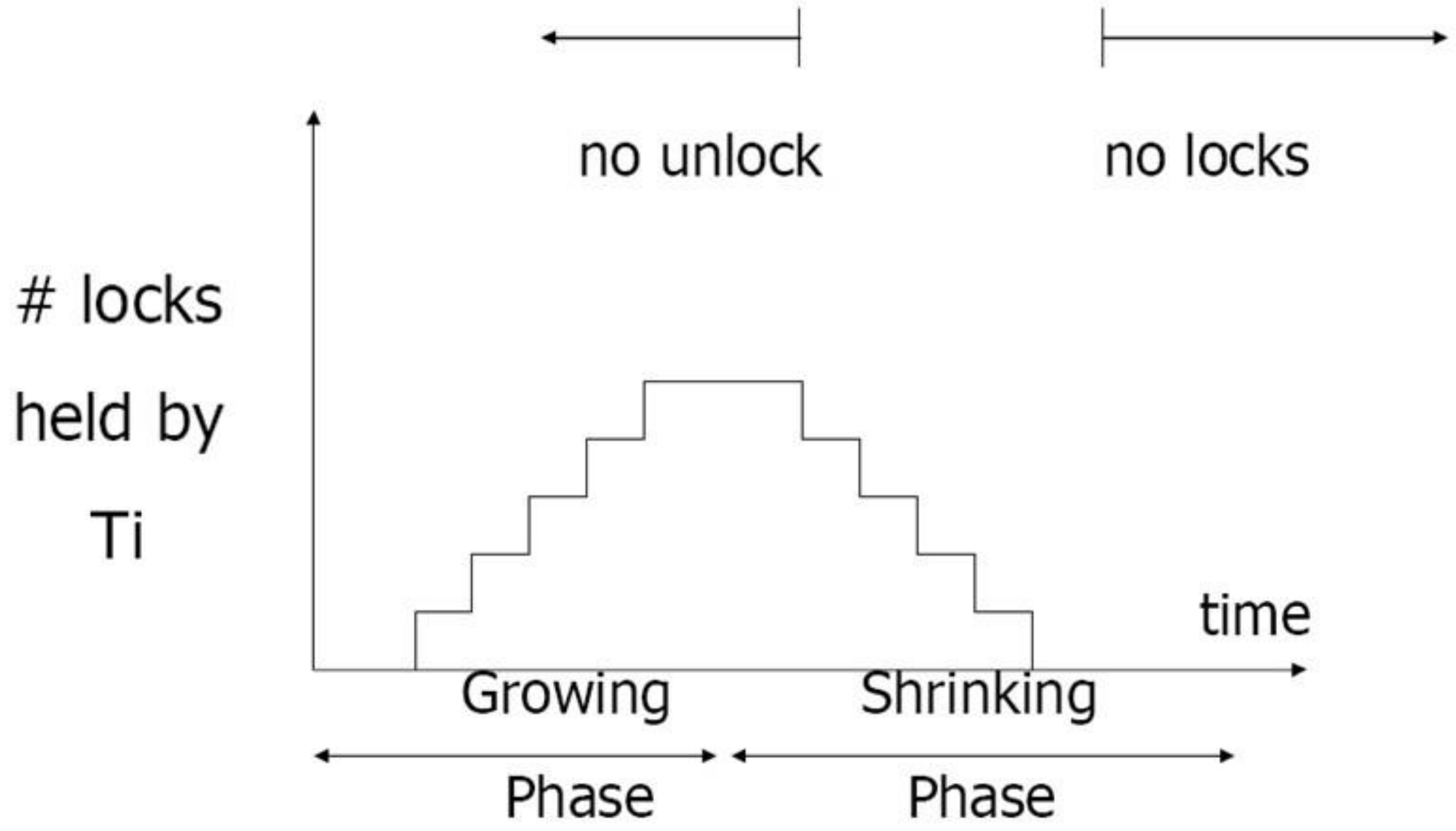
Δημήτρης Πλεξουσάκης
Τμήμα Επιστήμης Υπολογιστών

Tutorial on 2-Phase Locking and Other Lock Modes



2-Phase Locking Protocol

- 2-Phase Locking: All lock requests precede all unlock requests.



Exercise 1: 2PL

- For each of the following schedules, tell what the locking scheduler would do, i.e., what requests would get delayed and when would they be allowed to resume? Assume each lock is taken immediately before the corresponding read or write and that all locks are released immediately after the last element access.

a) R1(A); R2(A); W1(B); W2(B); R1(B); W2(C); W1(D);

b) R1(A); R2(A); R3(B); W1(A); R2(C); R2(B); W2(B); W1(C);

c) R1(A); W2(C); W1(B); R3(C); R2(B); W3(A);

d) W3(A); R1(A); W1(B); R2(B); W2(C); R3(C); R2(A);

e) R1(A); R2(A); R1(B); R2(B); R3(B); W1(A); W2(B);

Exercise 1: 2PL

- a) R1(A); R2(A); W1(B); W2(B); R1(B); W2(C); W1(D);

T1	T2
L1(A); R1(A);	
	L2(A); Denied
L1(B); W1(B);	
R1(B);	
L1(D); W1(D);	
U1(A); U1(B); U1(D);	
	L2(A); R2(A);
	L2(B); W2(B);
	L2(C); W2(C);
	U2(A); U2(B); U2(C);

Exercise 1: 2PL

- b) R1(A); R2(A); R3(B); W1(A); R2(C); R2(B); W2(B); W1(C);

T1	T2	T3
L1(A); R1(A);		
	L2(A); Denied	
		L3(B); R3(B); U3(B);
W1(A);		
L1(C); W1(C);		
U1(A); U1(C);		
	L2(A); R2(A);	
	L2(C); R2(C);	
	L2(B); R2(B); W2(B);	
	U2(A); U2(C); U2(B);	

Exercise 1: 2PL

- c) R1(A); W2(C); W1(B); R3(C); R2(B); W3(A);

T1	T2	T3
L1(A); R1(A);		
	L2(C); W2(C);	
L1(B); W1(B);		
U1(A); U1(B);		
		L3(C); Denied
	L2(B); R2(B);	
	U2(C); U2(B);	
		L3(C); R3(C);
		L3(A); W3(A);
		U3(C); U3(A);

Exercise 1: 2PL

- d) W3(A); R1(A); W1(B); R2(B); W2(C); R3(C); R2(A);

T1	T2	T3
		L3(A); W3(A);
L1(A); Denied		
	L2(B); R2(B);	
	L2(C); W2(C);	
		L3(C); Denied
	L2(A); Denied	

<u>DEADLOCK</u>

Exercise 1: 2PL

- e) R1(A); R2(A); R1(B); R2(B); R3(B); W1(A); W2(B);

T1	T2	T3
L1(A); R1(A);		
	L2(A); Denied	
L1(B); R1(B);		
		L3(B); Denied
W1(A);		
U1(A); U1(B);		
	L2(A); R2(A);	
	L2(B); R2(B);	
		L3(B); Denied
	W2(B);	
	U2(A); U2(B);	
		L3(B); R3(B); U3(B);

Exercise 2: 2PL

- T1: L1(A); R1(A); W1(A); L1(B); R1(B); W1(B); U1(A); U1(B);
  T2: L2(B); R2(B); W2(B); L2(A); R2(A); W2(A); U2(B); U2(A);

How many legal schedules of all the read and write actions of these transactions are there?

- Suppose the schedule starts with T1 locking and reading A. If T2 locks B before T1 reaches its unlocking phase, then there is a deadlock, and the schedule cannot complete. Thus, if T1 performs an action first, it must perform all its actions before T2 performs any. Likewise, if T2 starts first, it must complete before T1 starts, or there is a deadlock. Thus, only the two serial schedules of these transactions are legal.

Compatibility Matrix for Lock Modes

- Compatibility matrix for shared, exclusive, update and increment locks.

Locks requested

Locks held in mode

	S	X	U	I
S	Y	N	Y	N
X	N	N	N	N
U	N	N	N	N
I	N	N	N	Y

Y - Yes
N - No

Exercise 3: Other Lock Modes

- Insert shared, exclusive and update locks, together with unlock actions. Place a shared lock in front of every read action that is not going to be upgraded, place an update lock in front of every read action that will be upgraded and place an exclusive lock in front of every write action. Place unlocks at the ends of transactions.

a) R1(A); R2(B); R3(C); W1(B); W2(C); W3(D);

b) R1(A); R2(B); R3(C); W1(B); W2(C); W3(A);

c) R1(A); R2(B); R3(C); R1(B); R2(C); R3(A); W1(A); W2(B); W3(C);

d) R1(A); R2(B); R3(B); R1(C); R2(C); R3(C); W1(A); W2(C);

e) R1(A); R2(B); INC1(B); INC2(C); R3(B); INC3(C); W2(D);

Exercise 3: Other Lock Modes

- a) R1(A); R2(B); R3(C); W1(B); W2(C); W3(D);

T1	T2	T3
SL1(A); R1(A);		
	SL2(B); R2(B);	
		SL3(C); R3(C);
XL1(B); Denied		
	XL2(C); Denied	
		XL3(D); W3(D);
		U3(C); U3(D);
XL1(B); Denied		
	XL2(C); W2(C);	
	U2(B); U2(C);	
XL1(B); W1(B);		
U1(A); U1(B);		

Exercise 3: Other Lock Modes

- b) R1(A); R2(B); R3(C); W1(B); W2(C); W3(A);

T1	T2	T3
SL1(A); R1(A);		
	SL2(B); R2(B);	
		SL3(C); R3(C);
XL1(B); Denied		
	XL2(C); Denied	
		XL3(A); Denied

DEADLOCK

Exercise 3: Other Lock Modes

- c) R1(A); R2(B); R3(C); R1(B); R2(C); R3(A); W1(A); W2(B); W3(C);

T1	T2	T3
UL1(A); R1(A);		
	UL2(B); R2(B);	
		UL3(C); R3(C);
SL1(B); Denied		
	SL2(C); Denied	
		SL3(A); Denied

<u>DEADLOCK</u>

Exercise 3: Other Lock Modes

- d) R1(A); R2(B); R3(B); R1(C); R2(C); R3(C); W1(A); W2(C);

T1	T2	T3
UL1(A); R1(A);		
	SL2(B); R2(B);	
		SL3(B); R3(B);
SL1(C); R1(C);		
	UL2(C); R2(C);	
		SL3(C); Denied
XL1(A); W1(A);		
U1(A); U1(C);		
		SL3(C); Denied
	XL2(C); W2(C);	
	U2(B); U2(C);	
		SL3(C); U3(B); U3(C);

Exercise 3: Other Lock Modes

- e) R1(A); R2(B); INC1(B); INC2(C); R3(B); INC3(C); W2(D);

T1	T2	T3
SL1(A); R1(A);		
	SL2(B); R2(B);	
IL1(B); Denied		
	IL2(C); INC2(C);	
		SL3(B); R3(B);
		IL3(C); INC3(C);
		U3(B); U3(C);
	XL2(D); W2(D);	
	U2(B); U2(C); U2(D);	
IL1(B); INC1(B);		
U1(A); U1(B);		

Exercise 4: Other Lock Modes

- T1: R1(A); R1(B); INC1(A); INC1(B);
  T2: R2(A); R2(B); INC2(A); INC2(B);

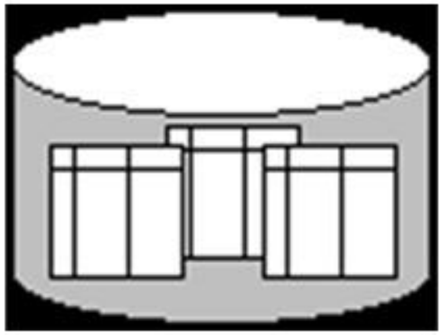
How many interleavings of these transactions are serializable?

- First, let us count the number of serializable orders that are conflict-equivalent to the order (T1, T2). In order for a schedule to be equivalent to (T1, T2), INC1(A) must precede R2(A). Thus, the first three steps of T1 must be first in the schedule.

  Also, INC1(B) must come before R2(B), so the former can either be the fourth action of the schedule, or it can be fifth, coming after R2(A). Thus, there are two serializable schedules equivalent to (T1, T2).

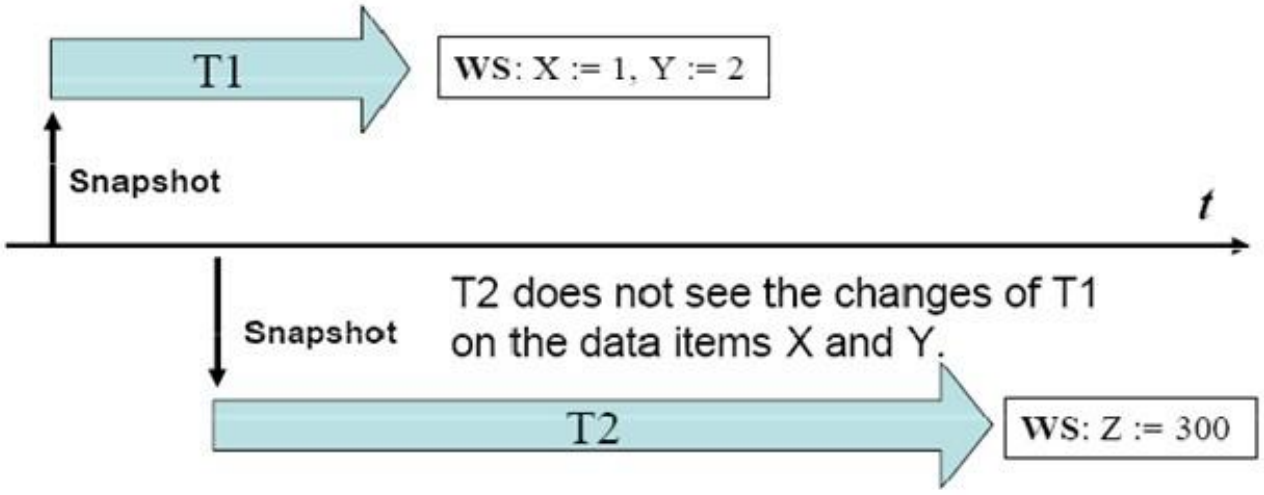
  By symmetry, there will be the same number equivalent to the opposite order, (T2, T1). Thus, there are four serializable schedules altogether.

Tutorial on Snapshot Isolation



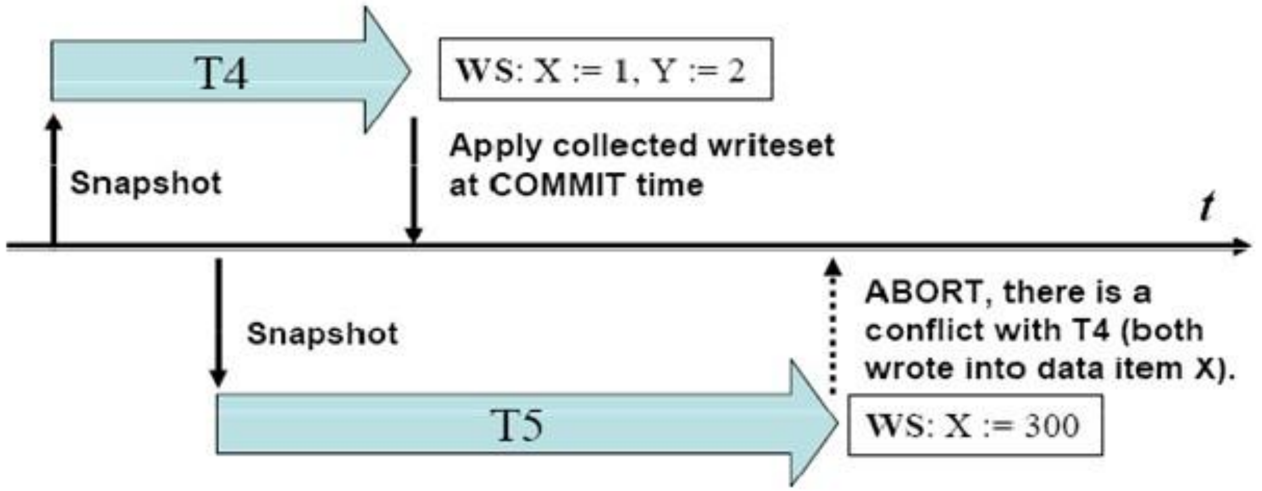
Snapshot Isolation - Basic Idea

- Every transaction reads from its own snapshot (copy) of the database (will be created when the transaction starts).
- Writes are collected into a writeset (WS), not visible to concurrent transactions. Two transactions are considered to be concurrent if one starts (takes a snapshot) while the other is in progress.

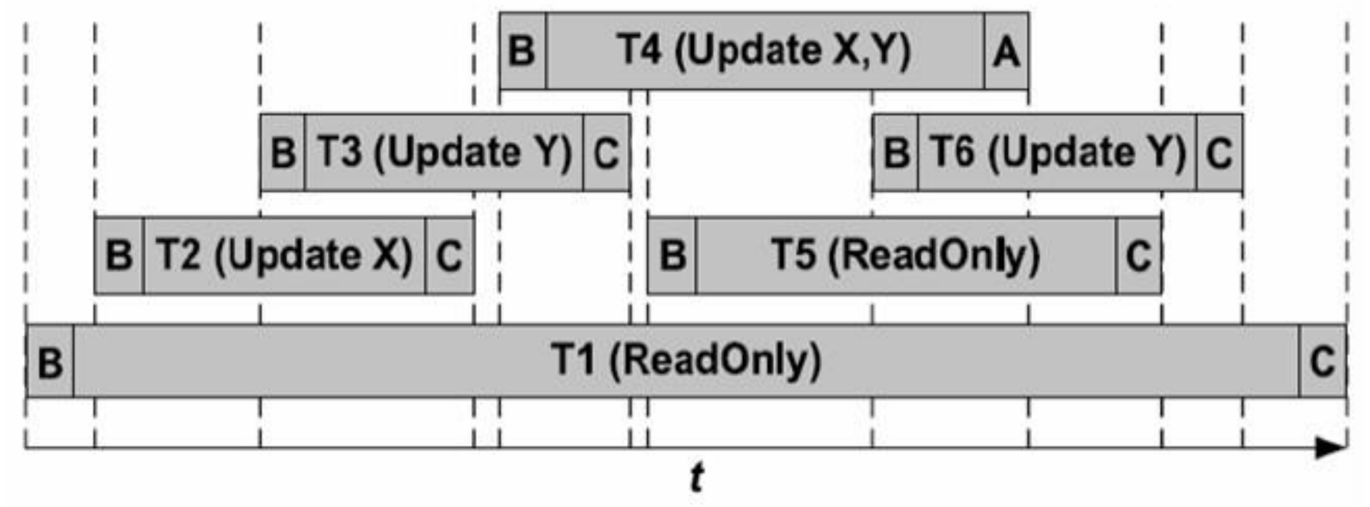

T1
WS: X := 1, Y := 2
Snapshot
t
Snapshot
T2 does not see the changes of T1 on the data items X and Y.
T2
WS: Z := 300

Snapshot Isolation - Conflict Resolution

- At the commit time of a transaction its writeset WS is compared to those of concurrent committed transactions. If there is no conflict (overlapping), then the WS can be applied to stable storage and is visible to transactions that begin afterwards.
- However, if there is a conflict with the WS of a concurrent, already committed transaction, then the transaction must be aborted. → "First Committer Wins Rule"


T4
WS: X := 1, Y := 2
Snapshot
Apply collected writeset at COMMIT time
t
Snapshot
ABORT, there is a conflict with T4 (both wrote into data item X).
T5
WS: X := 300

Example Transactions on SI Database



The symbols B, C and A refer to the begin, commit and abort of a transaction

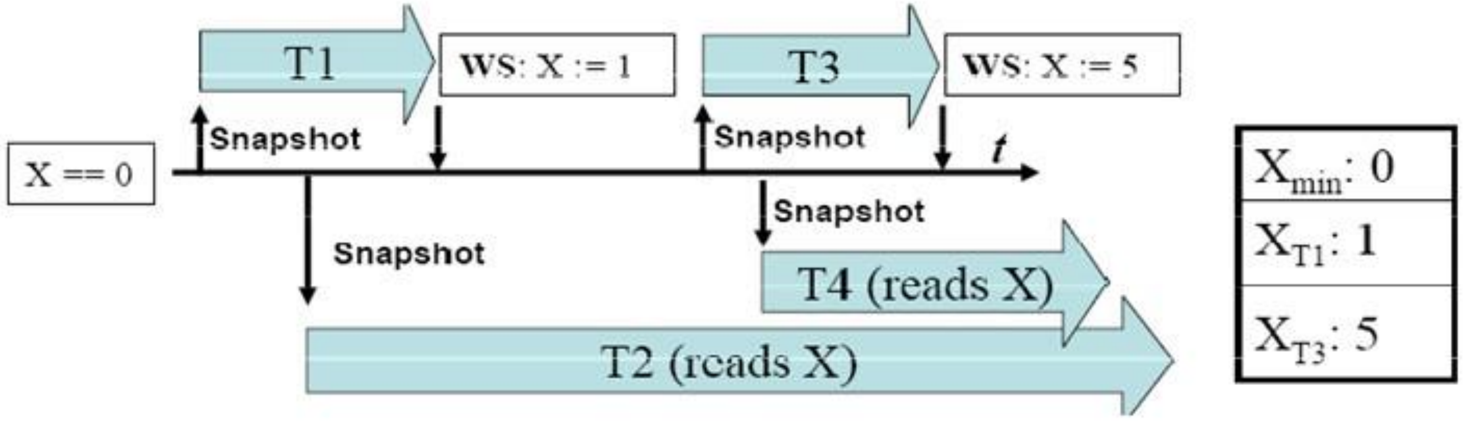
- T1 is read-only and will never conflict with any other transaction. Updates from concurrent transactions (like T2, T3, T4 and T6) are invisible to T1.
- T4 must be aborted, because its writeset overlaps with that from T3 which has already committed.
- Due to this fact, the overlapping sets of T6 and T4 do not impose a conflict.

Implementation of SI in real Systems

- Of course, making a copy of the database and managing and comparing (possibly huge) writesets for every transaction is not that efficient…
- Real SI implementations use an incremential variant of Snapshot Isolation, using
  - Different versions of the same data row (to simulate snapshots)
  - Row level (tuple) locks (to detect write-write conflicts between concurrent transactions).

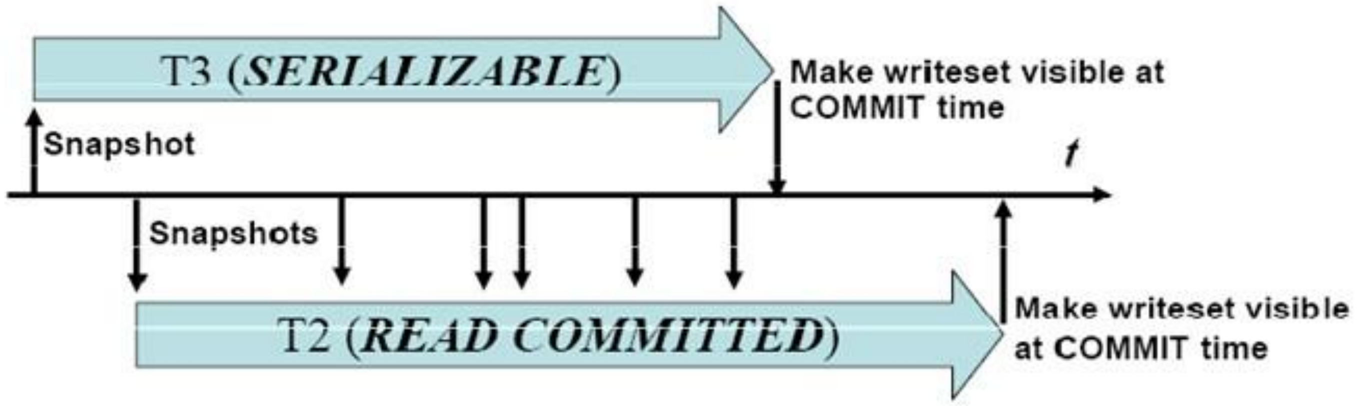
Implementation of SI in real Systems

- Snapshots are implemented by having multiple versions (hence, multi version concurrency control) of the same data item (e.g., data rows). A transaction that modifies a row generates automatically a new version of this row (which is only visible to transactions that begin (i.e., take a snapshot) after this transaction has committed).



Implementation of SI in real Systems

- Oracle and PostgreSQL offer two variants of Snapshot Isolation: SERIALIZABLE (as described so far) and READ COMMITTED (the default isolation level in both products)
- READ COMMITTED: the main difference to SERIALIZABLE is the implementation of the snapshot: a transaction running in this isolation mode gets a new snapshot for every issued SQL statement (every statement sees the latest committed values (generated versions) of the database).


T3 (SERIALIZABLE)
Make writeset visible at COMMIT time
Snapshot
t
Snapshots
T2 (READ COMMITTED)
Make writeset visible at COMMIT time

Τέλος Ενότητας

Χρηματοδότηση

•Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.

•Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.

•Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ
2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Σημειώματα

Σημείωμα αδειοδότησης

Σημείωμα Αναφοράς

Copyright Πανεπιστήμιο Κρήτης, Δημήτρης Πλεξουσάκης. «**Συστήματα Διαχείρισης Βάσεων Δεδομένων. Φροντιστήριο 10: Transactions - part 2**». Έκδοση: 1.0. Ηράκλειο/Ρέθυμνο 2015. Διαθέσιμο από τη δικτυακή διεύθυνση: http://www.csd.uoc.gr/~hy460/