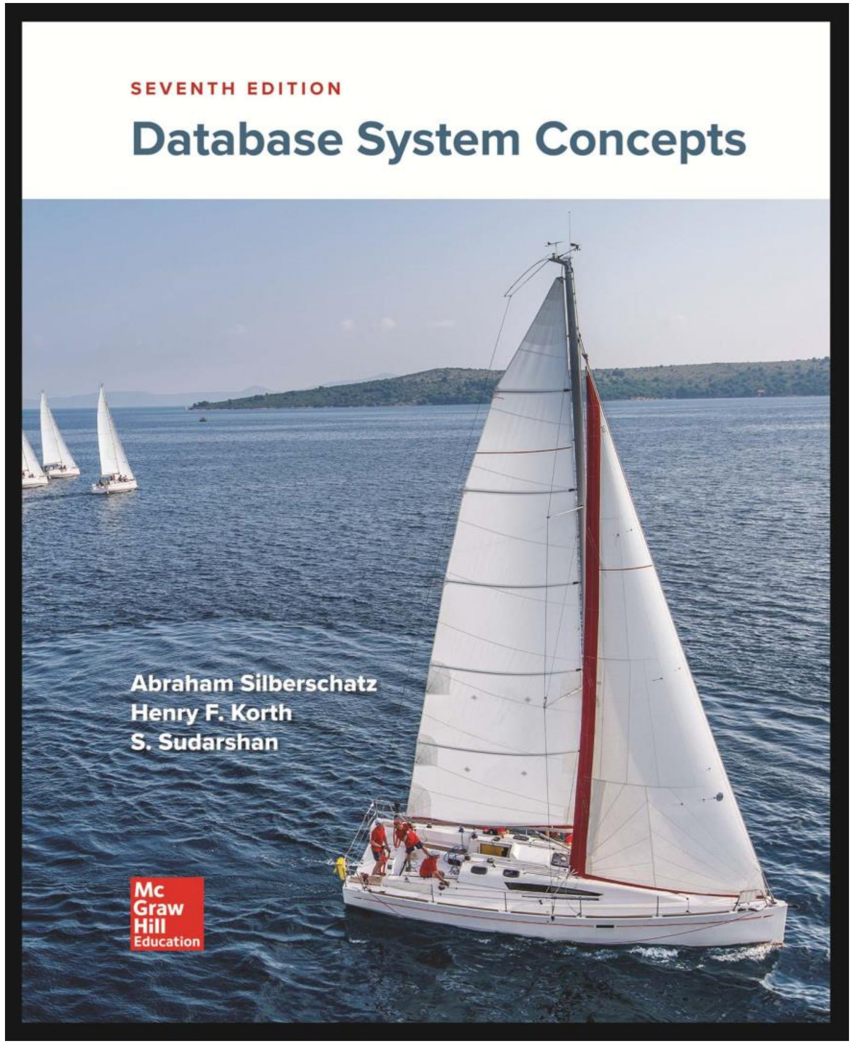
IF3140 – Sistem Basis Data

Concurrency Control:

- Introduction
- Lock-based Protocol

KNOWLEDGE & SOFTWARE ENGINEERING



Sumber

Silberschatz, Korth, Sudarshan: "Database System Concepts", 7th Edition

- Chapter 18: Concurrency Control

Objectives

Students are able to:

- Explain the effect of different isolation levels on the concurrency control mechanisms
- Choose the proper isolation level for implementing a specified transaction protocol

KNOWLEDGE & SOFTWARE ENGINEERING

Outline

- Lock-Based Protocols
  - 2-phase locking
  - Graph-based protocols
  - Deadlock handling
  - Multiple granularity
- Timestamp-Based Protocols
- Validation-Based Protocols
- Insert and delete operations
- Multiversion Schemes
  - MV Timestamp ordering
  - MV 2-phase locking
  - Snapshot isolation
- Weak levels of consistency

Lock-Based Protocols

- A lock is a mechanism to control concurrent access to a data item
- Data items can be locked in two modes :
  1. **exclusive** *(X) mode*. Data item can be both read as well as written. X-lock is requested using **lock-X** instruction.
  2. **shared** *(S) mode*. Data item can only be read. S-lock is requested using **lock-S** instruction.
- Lock requests are made to concurrency-control manager. Transaction can proceed only after request is granted.

Lock-Based Protocols (Cont.)

- **Lock-compatibility matrix**

	S	X
S	true	false
X	false	false

- A transaction may be granted a lock on an item if the requested lock is compatible with locks already held on the item by other transactions
- Any number of transactions can hold shared locks on an item,
- But if any transaction holds an exclusive on the item no other transaction may hold any lock on the item.

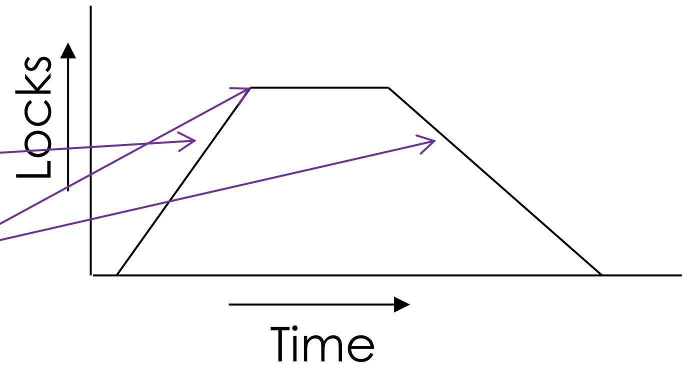
Schedule With Lock Grants

- Grants omitted in rest of chapter
  - Assume grant happens just before the next instruction following lock request
- This schedule is not serializable (why?)
- A **locking protocol** is a set of rules followed by all transactions while requesting and releasing locks.
- Locking protocols enforce serializability by restricting the set of possible schedules.

T_1	T_2	concurrency-control manager
lock-X(B)		
		grant-X(B, T_1)
read(B)		
$B := B - 50$		
write(B)		
unlock(B)		
	lock-S(A)	
		grant-S(A, T_2)
	read(A)	
	unlock(A)	
	lock-S(B)	
		grant-S(B, T_2)
	read(B)	
	unlock(B)	
	display($A + B$)	
lock-X(A)		
		grant-X(A, T_1)
read(A)		
$A := A + 50$		
write(A)		
unlock(A)		

The Two-Phase Locking Protocol

- A protocol which ensures conflict-serializable schedules.
- Phase 1: **Growing Phase**
  - Transaction may obtain locks
  - Transaction may not release locks
- Phase 2: **Shrinking Phase**
  - Transaction may release locks
  - Transaction may not obtain locks
- The protocol assures serializability. It can be proved that the transactions can be serialized in the order of their **lock points** (i.e., the point where a transaction acquired its final lock).



KNOWLEDGE & SOFTWARE ENGINEERING

Deadlock

- Consider the partial schedule

T_3	T_4
lock-X(B)	
read(B)	
$B := B - 50$	
write(B)	
	lock-S(A)
	read(A)
	lock-S(B)
lock-X(A)	

- Neither T_3 nor T_4 can make progress — executing **lock-S**(B) causes T_4 to wait for T_3 to release its lock on B, while executing **lock-X**(A) causes T_3 to wait for T_4 to release its lock on A.
- Such a situation is called a **deadlock**.
    - To handle a deadlock one of T_3 or T_4 must be rolled back and its locks released.

Deadlock (Cont.)

- The potential for deadlock exists in most locking protocols. Deadlocks are a necessary evil.
- **Starvation** is also possible if concurrency control manager is badly designed. For example:
  - A transaction may be waiting for an X-lock on an item, while a sequence of other transactions request and are granted an S-lock on the same item.
  - The same transaction is repeatedly rolled back due to deadlocks.
- Concurrency control manager can be designed to prevent starvation.

The Two-Phase Locking Protocol (Cont.)

- Two-phase locking *does not* ensure freedom from deadlocks
- Extensions to basic two-phase locking needed to ensure recoverability or freedom from cascading roll-back
  - **Strict two-phase locking:** a transaction must hold all its exclusive locks till it commits/aborts.
    - Ensures recoverability and avoids cascading roll-backs
  - **Rigorous two-phase locking**: a transaction must hold *all* locks till commit/abort.
    - Transactions can be serialized in the order in which they commit.
- Most databases implement rigorous two-phase locking, *but refer to it as simply two-phase locking*

KNOWLEDGE & SOFTWARE ENGINEERING

The Two-Phase Locking Protocol (Cont.)

- Two-phase locking is not a necessary condition for serializability
  - There are conflict serializable schedules that cannot be obtained if the two-phase locking protocol is used.
- In the absence of extra information (e.g., ordering of access to data), two-phase locking is necessary for conflict serializability *in the following sense:*
  - *Given a transaction T_i that does not follow two-phase locking, we can find a transaction T_j that uses two-phase locking, and a schedule for T_i and T_j that is not conflict serializable.*

T_1	T_2
lock-X(B)	
read(B)	
$B := B - 50$	
write(B)	
unlock(B)	
	lock-S(A)
	read(A)
	unlock(A)
	lock-S(B)
	read(B)
	unlock(B)
	display($A + B$)
lock-X(A)	
read(A)	
$A := A + 50$	
write(A)	
unlock(A)	

KNOWLEDGE & SOFTWARE ENGINEERING

Locking Protocols

- Given a locking protocol (such as 2PL)
  - A schedule S is **legal** under a locking protocol if it can be generated by a set of transactions that follow the protocol
  - A protocol **ensures** serializability if all legal schedules under that protocol are serializable

KNOWLEDGE & SOFTWARE ENGINEERING

Exercise 9.1

contoh alternatif yg akan deadlock ←

T1	T2
L-S(A)	
	L-S(B)
wait T2 ← L-X(B)	
	L-X(A) → wait T1
R(A) R(B)	
	R(B) R(A

contoh eksekusi yg ga akan deadlock, tp jd serial karena T2 harus nunggu T1 all beres dulu

T1	T2
L-S(A) L-X(B)	
	L-S(B) L-X(A) ← harus wait soalnya di lock di T1.
R(A) R(B)	
	R(B) R(A)
W(B) UL(B) UL(A)	
	W(A) UL(A) UL(B)

Consider the following two transactions:

T1:
read (A)
read (B)
B := B + 0.1*A
write (B)

T2:
read (B)
read (A)
A := A – 0.05*B
write (A)

a) Add lock and unlock instructions to both transactions so that they follow the two-phase locking protocol.

b) Can the execution of these two transactions result in a deadlock?

KNOWLEDGE & SOFTWARE ENGINEERING

Solution

Exercise 9.2

NOTE: automatic bila ubah dr L(S) → read
ke L(X) → read & write

Is the following schedule a two-phase locking (2PL) schedule (legal under 2PL protocol)?

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

T1	T2	T3
L-S(A)		
R(A)		
	L-S(A) ←	wait
	R(A)	
		L-S(B)
		R(B)

Solution

KNOWLEDGE & SOFTWARE ENGINEERING

Lock Conversions

- Two-phase locking protocol with lock conversions:
  - Growing Phase:
    - can acquire a lock-S on item
    - can acquire a lock-X on item
    - can **convert** a lock-S to a lock-X (**upgrade**)
  - Shrinking Phase:
    - can release a lock-S
    - can release a lock-X
    - can convert a lock-X to a lock-S (**downgrade**)
- This protocol ensures serializability

KNOWLEDGE & SOFTWARE ENGINEERING

Automatic Acquisition of Locks

- A transaction T_i issues the standard read/write instruction, without explicit locking calls.
- The operation **read**(*D*) is processed as:

if T_i has a lock on *D*
  then
    read(*D*)
  else begin
    if necessary wait until no other
      transaction has a **lock-X** on *D*
    grant T_i a **lock-S** on *D*;
    read(*D*)
  end

Automatic Acquisition of Locks (Cont.)

- The operation **write**(*D*) is processed as:

```
if Ti has a lock-X on D
  then
    write(D)
  else begin
      if necessary wait until no other trans. has any lock on D,
      if Ti has a lock-S on D
         then
           upgrade lock on D to lock-X
         else
           grant Ti a lock-X on D
         write(D)
    end;
```

- **All locks are released after commit or abort**

KNOWLEDGE & SOFTWARE ENGINEERING

Exercise 9.3

Instructions from T1, T2, and T3 arrive in the following order.

R1(A); R2(A); R3(B); W1(A); R2(C); R2(B); C3; W2(B); C2; W1(C); C1;

What is the final *schedule* if the *2-phase locking with automatic acquisition of locks* is implemented by CC Manager?

Solution