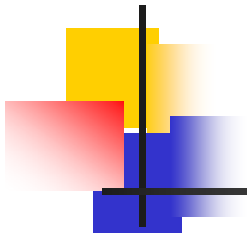
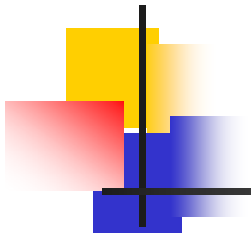


2. Concurrency Control의 종류

- Concurrency Control의 법칙
  - 동시에 실행되는 트랜잭션의 정확성 유지
  - 트랜잭션의 동시 실행은 순차 실행보다 좋은 성능 보장

- Concurrency Control의 종류
  - Locking
  - Timestamp Ordering
  - Optimistic Concurrency Control

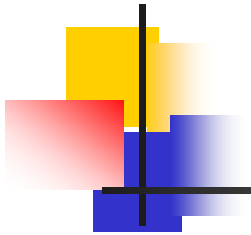


3. Locking

- Lock이란?
  - 데이터에 대한 액세스 권한
  - Lock을 가지고 있는 트랜잭션만이 데이터 액세스

- Lock Modes
  - Shared (S) Lock: 데이터를 read할 때 사용
  - Exclusive (X) Lock: 데이터를 write할 때 사용

	S	X
S	OK	Not OK
X	Not OK	Not OK



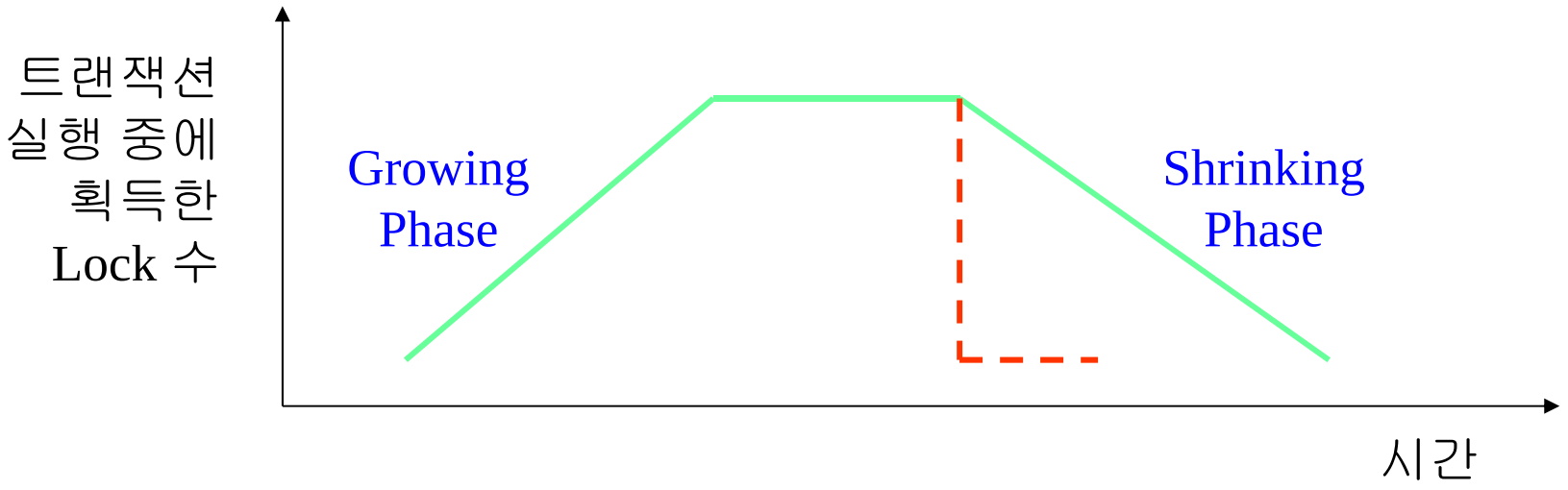
Locking Rule

- Lock을 획득하는 시점
  - 데이터를 액세스하기 전에 Lock 요청 (well-formed)
  - Conflict mode의 Lock이 이미 걸려 있을 경우
    - Lock이 해제될 때까지 대기 ← Deadlock 발생 가능
    - 여러 트랜잭션이 대기 중일 때 ← Starvation 발생 가능

- Lock을 해제(Unlock)하는 시점
  - 데이터를 액세스한 후, 즉시 Unlock할 경우
    - Lost Update, Dirty Read, Unrepeatable Read
  - 트랜잭션이 완료할 때까지 보유 (two-phase rule)

Two Phase Locking (2PL)

- Two Phase Locking Protocol
  - 정의: Unlock을 한 후 Lock 요청 불가
    - Growing Phase: Lock을 요청하는 단계
    - Shrinking Phase: Unlock 단계
  - Conservative 2PL
    - End of Transaction에서 모든 Lock을 해제



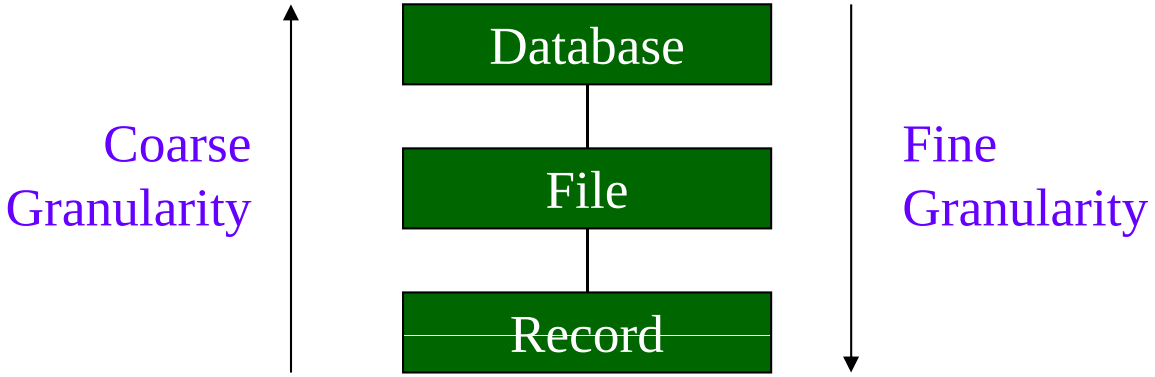
Concurrent Schedule – Locking

Schedule 3	
T1	T2
read(A) ← S(A) A = A – 50 write(A) ← X(A)	
	read(A) ← S(A) temp = A * 0.1 A = A – temp write(A)
read(B) ← S(B) B = B + 50 write(B) ← X(B)	
	read(B) B = B + temp write(B)

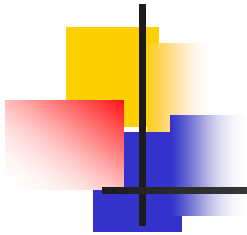
Schedule 4	
T1	T2
read(A) ← S(A) A = A – 50	
	read(A) ← S(A) temp = A * 0.1 A = A – temp write(A) ← X(A) read(B)
write(A) ← X(A) read(B) B = B + 50 write(B)	
	B = B + temp write(B)

4. Multiple Granularity Locking

- Lock (Lock Granularity)
  - Fine granularity: High concurrency, High overhead
  - Coarse granularity: Low concurrency, Low overhead

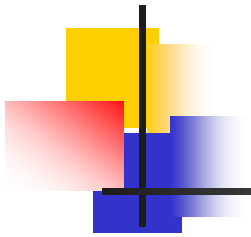


- Multiple Granularity Locking (MGL)
  - 목적: High Concurrency with Low Overhead
  - Explicit Locking, Implicit Locking, Intention Locking



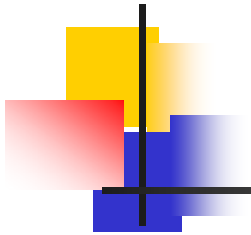
Intention Mode

- Intention Shared (IS) Mode
  - 하위 레벨에서 Explicit S mode의 lock 보유
- Intention Exclusive (IX) Mode
  - 하위 레벨에서 Explicit X mode의 lock 보유
- Shared and Intention Exclusive (SIX) Mode
  - Explicit S mode의 lock 보유
  - 하위 레벨에서 Explicit X mode의 lock 보유



Compatibility Matrix of MGL

	IS	IX	S	SIX	X
IS	OK	OK	OK	OK	NOK
IX	OK	OK	NOK	NOK	NOK
S	OK	NOK	OK	NOK	NOK
SIX	OK	NOK	NOK	NOK	NOK
X	NOK	NOK	NOK	NOK	NOK

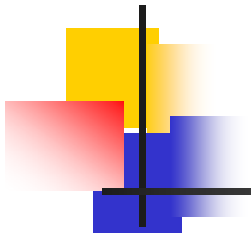


Locking Protocol of MGL

- Lock의 획득과 해제
  - Lock의 획득: Root → Leaf
  - Lock의 해제: Leaf → Root

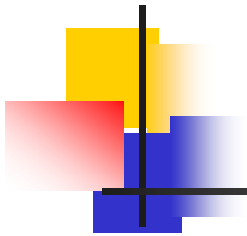
- Lock Mode
  - Explicit S lock를 요청할 노드의 모든 상위 노드: IS
  - Explicit X lock를 요청할 노드의 모든 상위 노드: IX
  - SIX lock를 요청할 노드의 모든 상위 노드: IX

- Lock Compatibility Matrix 준수



Examples

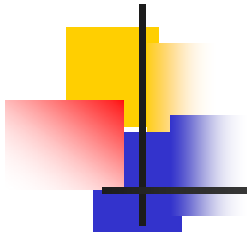
- Lock a record R for read
  - IS(DB) → IS(R을 포함한 File) → S(R)
- Lock a record R for update
  - IX(DB) → IX(R을 포함한 File) → X(R)
- 파일 F의 모든 레코드를 검색
  - IS(DB) → S(F)
- 파일 F를 읽으면서, 특정 레코드를 갱신
  - IX(DB) → SIX(F) → X(갱신할 레코드)
- 전체 DB를 single user용으로 사용
  - X(DB)



5. Deadlock Handling

- Deadlock Scenario - Example

T1	T2
Lock-X(B) read(B) B = B – 50; write(B) Lock-X(A)	 Lock-S(A) read(A) Lock-S(B)



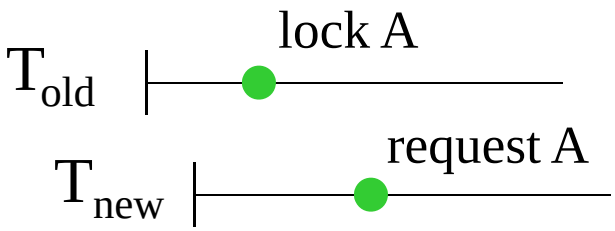
Deadlock 해결 방법

- Deadlock 해결 방법
  - Deadlock Prevention
  - Deadlock Detection and Resolution

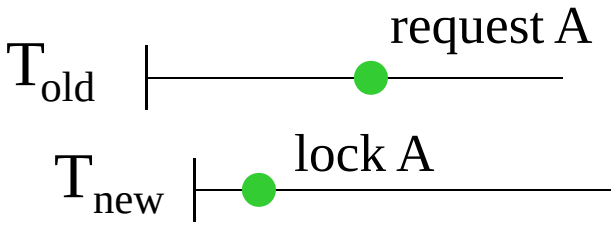
- Deadlock Prevention
  - 정의: Deadlock이 발생할 가능성을 봉쇄
  - 종류
    - 모든 lock을 한번에 요청. 거부될 경우 모두 해제.
    - 데이터에 순서 부여. 순서대로 lock 요청
    - Preemption: wait-die, wound-wait

Wait-Die, Wound-Wait

Wait-Die

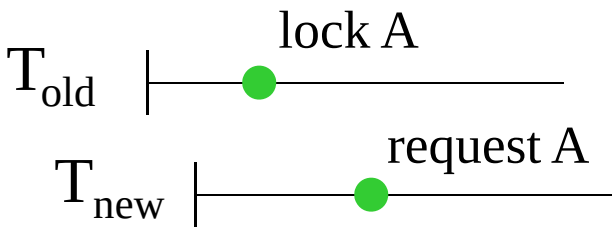


T_{new}: Rollback

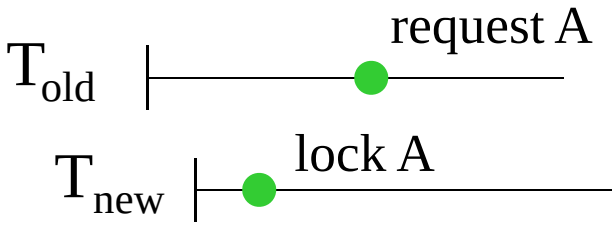


T_{old}: Wait

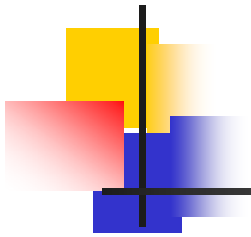
Wound-Wait



T_{new}: Wait



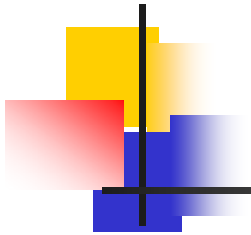
T_{new}: Rollback



Deadlock Detection

- Wait-For Graph: G = (V, E)
  - V: Set of transactions
  - E: Set of $T_w \rightarrow T_h$, where T_w is waiting for T_h

- Deadlock Detection and Resolution
  - Deadlock 조건: WFG에 cycle 발생
  - Resolution
    - Selection of a victim transaction
    - Victim transaction의 rollback
    - Starvation의 발생 여부



6. Other Concurrency Controls

- Timestamp Ordering
  - 트랜잭션과 데이터에 타임스탬프 할당(TS, R, W)
  - 트랜잭션의 타임스탬프 순서로 데이터 액세스
  - 순서를 만족하지 못할 경우, 트랜잭션 철회

- Optimistic Concurrency Control
  - 트랜잭션들간에 충돌이 없다고 가정
    - Local data에 대해 트랜잭션 실행 후, 검증
    - 검증의 결과 충돌이 발생할 경우, rollback
    - 충돌이 발생하지 않을 경우, 실행 결과를 저장
  - 대부분의 트랜잭션이 read-only일 경우, 최선의 선택