눈복신 초록에 노래 최정

1. ACID는 Atomicity, Consistency, Isolation, Durability를 말한다.

Atomicity는 연산이 @ all or nothing 임을 의미하고

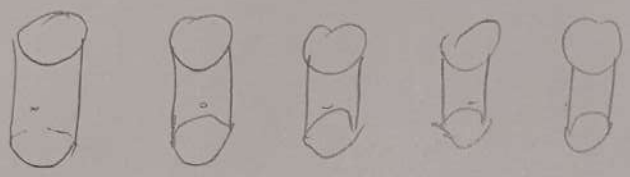
Consistency는 트랜잭션이 valid한 상태를 만든 것임을 의미하고

Isolation은 트랜잭션이 다른 트랜잭션에 의해 간섭당하지 않아야 함을 의미하고

Durability는 트랜잭션이 public, durable 하다 함을 의미 한다.

public이란 트랜잭션의 결과가 반영되지 않았더라도 다른 연산들이 해당 결과를 사용해야 함을 뜻하고, durable이란 트랜잭션의 결과가 시스템 failure가 생겨도 항상 반영되어야 함을 뜻한다.

2. A,



block 단위로 Parity bit를 각각 다른 디스크에 저장하고, 4개의 디스크에 병렬적으로 write하고 나머지 하나의 디스크에 Parity를 쓴다. 이때 Parity bit는 4개 디스크에 분산되어 쓰이게 된다.

B. block 단위로 디스크에 쓰기 읽기 연산을 병렬적으로 할 수 있으므로 Performance가 향상되고, Parity bit를 확인해서 하나의 디스크에 문제가 생겨도 복구할 수 있으므로 Reliability가 향상된다.

3. A, bucket overflow는 서로 다른 레코드에 대해 hash function이 같은 결과를 너무 많이 만들어 낼 경우 발생한다.

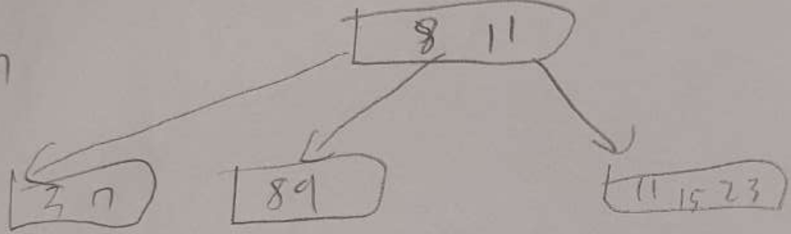
B. hash function이 uniform, random 하게 잘 구현하도록 한다.

4.

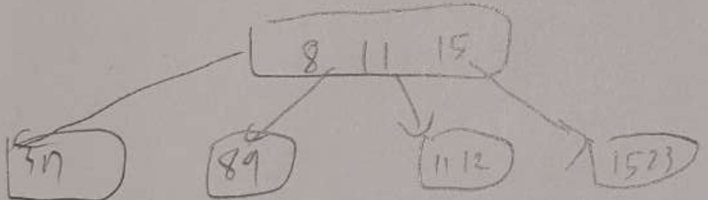
B+ 트리에서 ID가 9500인 교수를 찾기 위해 $\log_{50} 30000$ = 3번의 block seek + transfer가 필요하고, 그 뒤 5%가 9500보다 크므로 약 1500개의 레코드가 9500보다 크다. 이때 각 block에 대략 30개의 레코드가 들어 있으므로 50개의 블록을 더 transfer 해야 한다. 이름이 Kin%인 것은 순서대로 돌면서 확인하면 되므로 추가 transfer가 필요 없다.

따라서 Seek는 3+1 = 4번, transfer는 3+50 = 53번 이다.

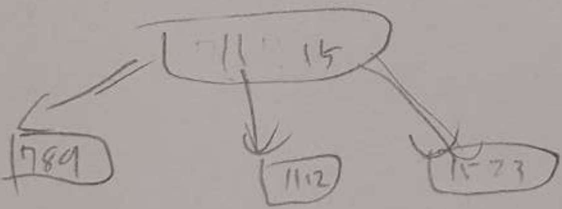
5. A. (1)



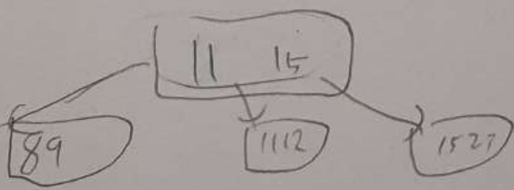
(2)



B. (1)



(2)



눈부신 초록의 노래처럼

6. (A) Cascading rollback이란 트랜잭션을 여러 개 수행할 때, 하나의 트랜잭션을 rollback 하야 할 경우 다른 트랜잭션들이 연쇄적으로 rollback되는 경우를 말한다.

Cascadeless schedule은 이러한 경우가 일어나지 않음이 보장되는 스케줄이다.

(B) Cascadeless schedule은 dirty read, 즉 write된 것이나 commit 되기 전에 읽히는 경우가 없어야 한다.

따라서 strict 2PL을 사용하면 write를 하면 lock-X를 획득하므로 다른 트랜잭션들은 해당 트랜잭션이 커밋되기 전엔 해당 요소를 읽을 수 없다.

따라서 cascadeless 하다.

7. 우선 run을 만들면 30개의 run이 생긴다.

external sort-merge의 block transfer 수는

8. A. $A \to D$ (by transitivity)

$B \to E$ (〃)

$AB \to CD$ (by augmentation)

B. $R_1 = (A\ B\ C)$

$R_2 = (B\ D)$

$R_3 = (D\ E)$

$F_1 = \{A \to B, A \to C\}$

$F_2 = \{B \to D\}$

$F_3 = \{D \to E\}$

C. Yes. R_1, R_2, R_3의 FD를 각각 F_1, F_2, F_3라 하면

$(F_1 \cup F_2 \cup F_3)^+ = F^+$ 가 되기 때문이다.

9. (A) No. precedency 그래프를 그려보면 사이클이 존재한다.

← A를 T_1에서 read → T_2에서 read, write → T_1에서 write,

(B)

T_1	T_2	
read(A)		← 그림을 사용해서 기술하기 좋아서
A := A - 50		T_2에서 write(A)를 하는 부분에서
write(A)		T_1이 A에 대해 lock-S를 가지고 있으므로
read(B)		대기하게 된다.
B := B+50		그 뒤 T_1의 모든 연산이 수행되고,
write(B)		commit 된 뒤 T_2의 모든 연산이 수행된다.
commit		
	read(A)	
	⋮	
	write(A)	
	commit	

10. 감사합니다!

교수님, 조교님들 모두 한 학기 동안 정말 ✓ 고생 많으셨습니다. ^^