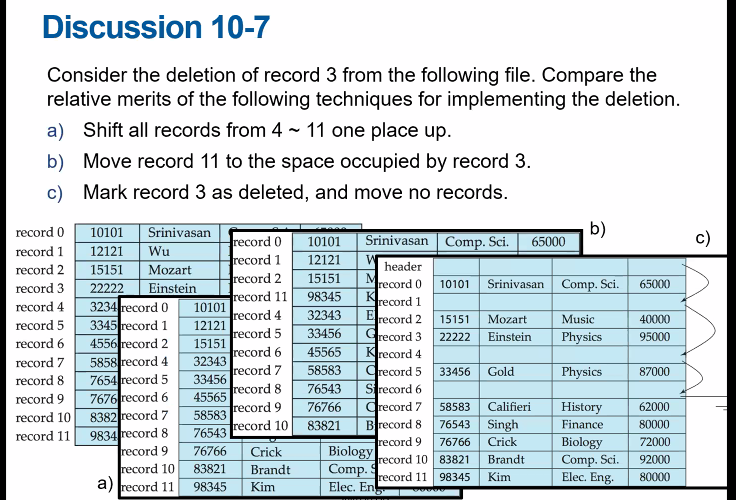
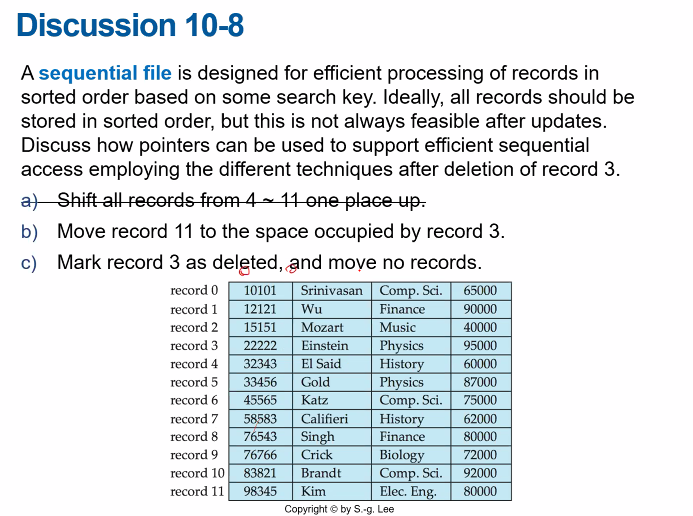
**Discussion 05/18**



a)의 장점은 sequential read 속도가 빠르다. Record 순서가 보장된다. Maximum available space 보장. 단점은 record 하나가 지워지면 그 밑의 모든 record를 shift 해야 한다.

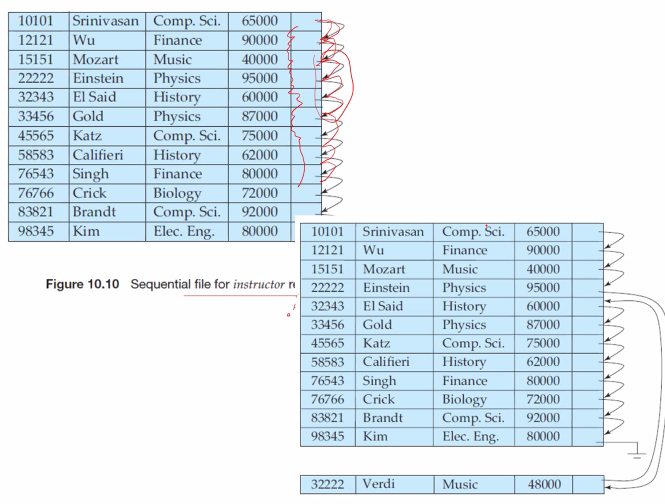
b)의 장점은 delete의 overhead가 줄어듦. Maximum available space 보장. 단점은 record 순서가 지켜지지 않음.

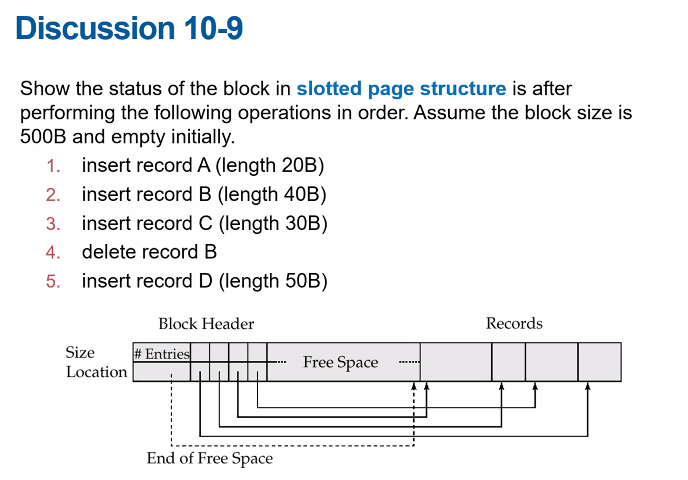
c)의 장점은 delete나 insert가 상수 시간. Record 순서 보장. 단점은 record 사이에 빈 공간이 생기는 것.



Header에 시작과 끝을 저장하는 pointer를 둔다. 시작 pointer는 id가 가장 작은 record, 끝 pointer는 가장 큰 record를 가리키게 하고, 각 record마다도 포인터를 둬서 id의 크기를 따라 다음 record를 가리키게 한다. Insert를 할 경우 id 값을 보고 pointer끼리의 관계에서 맞는 곳에 pointer를 insert 하고, Delete가 되면 그냥 해당 record의 pointer도 delete하고 앞 뒤 관계를 갱신해주면 된다.

이러한 방식으로 포인터를 사용하면 id의 정렬된 순서를 보장할 수 있다.

=>  이런 식으로 pointer 사용!

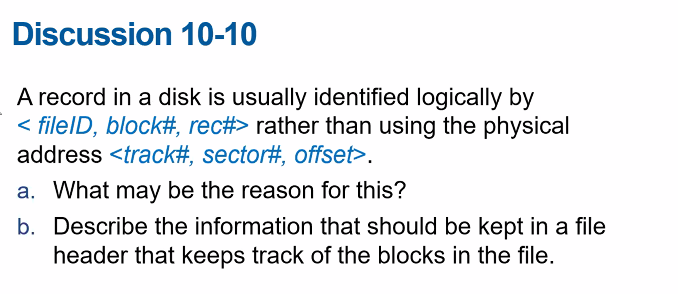


Entry 수가 3개, block header는 A, 빈 공간, C, D의 위치와 크기 등 정보를 가지고 있음.

오른쪽 record 공간은 오른쪽에서부터 순서대로 A 20B, 빈 공간 40B (B가 삭제됐기 때문에), C 30B, D 50B가 저장되어 있음.

가운데에는 free space가 있음.

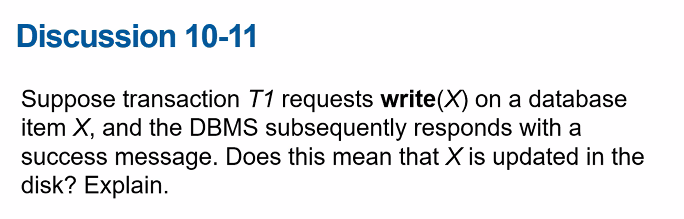
=> delete 연산이 되면 블락 내부에서는 shift를 함! Free space를 한 덩어리로 크게 보장하는 것이 핵심. Record를 shift해서 빈 공간을 없애야 함. Header는 shift하지 않고 그냥 -1 등 삭제됐다는 표시만 해줌.



a. 일관성을 유지하기 위해서. Logical address를 사용하면 OS가 하드웨어 구조에 따라 적절하게 physical로 맵핑을 해주지만 physical address를 사용하면 하드웨어에 따라 구조가 달라져야 함.

=> 교수님 설명: 대형 DBMS들은 physical address도 관리할 수 있음. 근데도 쓰는 이유는 record들이 옮겨질 수도 있고, record의 size가 변경될 수도 있음. 이렇게 데이터가 변경되는 순간 해당 데이터를 참조하고 있는 정보들의 위치가 모두 바뀌어야 함. Logical로 관리하면 physical을 바꿀 필요가 없음.

b. logical address와 physical address의 맵핑 정보가 있어야 함.



그럴 수도 있고 아닐 수도 있다.

DBMS에서 write를 할 때 메인 메모리의 buffer에 block을 불러와서 한다. 그 후 다른 transaction이 X를 참조할 경우가 생기면 바로 disk에 write를 하지 않고 그 작업까지 buffer의 block에서 모두 끝낸 후 disk에 update를 한다.

=> 디스크에 반영이 되지 않은 상태로 메모리에만 저장되어 있다가 문제가 생길 경우 recovery system이 해결을 해줘야 한다고 한다. 16장에서 배울 예정.