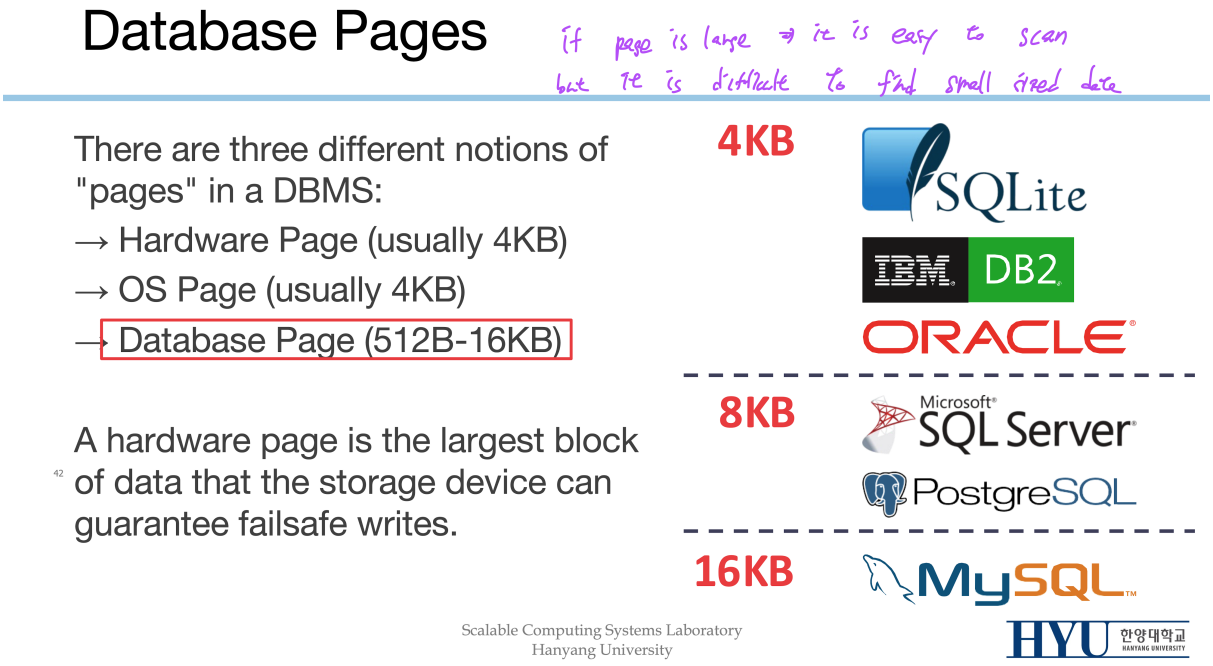
1주차_물리적_데이터베이스_설계

File을 어떻게 조직할 것인지

플래시 메모리에서 데이터를 어떻게 저장을 할건지가 최근 메타



블록(페이지)를 4KiB 씩 잡음 (물론 현재 dbms들은 더 크게 잡고 있음)

솔직히 하드디스크 원리를 정확하게 아는게 중요한가 싶음...

그냥 disk I/O가 왜 발생하는지 정도만 알면 될듯....

disk는 random access가 불가능

버퍼

Magic Number ⇒ 파일이 정확한지 확인하는 번호

버퍼를 사용하면 캐시 효과를 사용할 수 있음 (block 단위로 페이지를 구성해 메모리 영역으로 올림)

LRU 알고리즘이 효과적이지 않은 이유는 sequential flooding 때문

(버퍼 페이지가 약간 많으면 발생하는 현상)

실제로 상황에 따라 사용하는 replacement 전략이 다름

레코드 배치

고정길이, 가변길이 들이 처리하는 방식이 다름

가변길이는 주로 footer 에 저장하고

고정길이는 header에 저장

고정길이의 가장 큰 장점은 random access가 가능 (레코드의 위치를 한번에 파악할 수 있음)

삭제하고 fragmentation을 처리하는 방법이 2가지가 있는데

1. 맨 뒤에있는 레코드를 빈칸에 넣는 방법
2. 모든 record를 shift 하는 방법

블록들을 최대한 연속적으로 저장 ⇒ clustering (locality)

한 파일에서 같이 검색될 가능성이 높은 record를 물리적으로 가까운 위치에 모아두는 것

(디스크가 회전하면서 데이터를 찾으니까)

Heap File (여기서 heap은 운체, 자료구조의 그 heap이 아님)

삽입은 맨 뒤에 하기 때문에 당연히 빠름

근데 검색 속도가 너무 느림 ⇒ 정렬도 안되어 있고 그냥 쭉 나열하니까

검색이 느리기 때문에 삭제 속도도 느림

근데 DB가 주로 검색기능이 핵심인걸 감안하면 좋은 형태의 배치 방식은 아님

Index

각 레코드 별로 entry를 구성

entry는 key값이랑 레코드의 주소값을 묶어서 저장 (slot page)

Primary index

1. 탐색 키가 기본키인 인덱스를 primary index 라고 함
2. primary index는 기본 키에 따라 정렬된 데이터 파일에 의해 정의됨
3. 희소 인덱스 (모든 데이터에 대해서 key를 구성하는 것이 아니라 특정 그룹 마다 key를 구성)
4. 각 릴레이션 마다 최대 한개만 가질 수 있음

Clustering Index

완전 정렬은 아니지만 비슷한 데이터 끼리 하나의 블록에 묶어서 저장하는 방식 (물리적으로)

가장 큰 장점은 range search가 가능

Secondary Index

primary index 이외에 다른 attribute를 기준으로 데이터 정렬을 하는 것

(밀집 인덱스라 모든 레코드마다 index 부여 ⇒ disk I/O 증가)