DB 0324

하드웨어 저장 방법

데이터베이스가 레코드를 기록할 때 어떻게 저장하는데 좋을지?

데이터베이스 저장 장치 구조

Block Structure : block header + data records

데이터 저장 순서 방식

하드웨어 -> 운영체제 -> 파일 시스템 -> 데이터 저장

파일 시스템

주기억장치(CPU와 가까운 저장 장치) : 메모리

보조기억장치 : HDD, SSD, 플래쉬 메모리, CD 등

파일시스템의 역할

1. 파일 저장 및 관리
2. 파일 검색 및 탐색
3. 파일 접근 권한 관리
4. 파일 수정 / 삭제

파일 : 분리할 수 없는 최소 단위

디렉토리(폴더) : 파일들을 관리하기 위한 모음집, 트리 구조를 가진다.

백엔드는 주로 리눅스로 사용, shell프로그래밍, 나노, vi 사용할 줄 알아야 한다.

파티셔닝 : 저장 장치의 논리적인 영역을 구획하는 작업

포매팅 : 파일 시스템 형식, 파일 시스템 종류

빠른 포맷 : 파일 시스템만 기억을 지우는 것 -> 포렌식 가능

일반 포맷 : 디스크 상에 있는 전자기록을 지우는 것 -> 포렌식 불가능

파일 시스템 종류 : ext4, NTFS, exFAT, FAT32, APFS

운영체제가 다르면 파일을 볼 수 없다 (파일 시스템 종류가 다르기 때문이다)

파일 할당 방법

Sector, track, block

Sector : 디스크 컨트롤러 최소 단위

Block : 운영체제가 관리하는 최소 단위

여러 sector => Block

파일 시스템이 파일을 여러 block으로 나누어 저장

각 블록은 위치를 식별하는 주소를 가진다.

파일 할당(Allocation) : 파일을 불러오거나 저장하는 것

연속 할당(Configuous Allocation) : 사용하지 않는다.

* 시작번호와 길이만 설정해주면 된다.
* 장점 : 블록을 연달아 저장, 관리 용이, 읽기/쓰기 빠름,
* 단점 : 외부 단편화에 취약함(파일 수정/삭제)

비연속 할당

연결 할당(Linked Allocation)

* 자료구조(제일 사용 많이 된다.)
* 시작번호, 끝 번호 / 사이즈
* 장점 : 연속 할당의 단점인 단편화 보완
* 단점 : 병렬 처리가 안된다.(순차적), 블록 하나가 고장 나면 찾을 수가 없다.

색인 할당(index block)

* 인덱스 블록(첫번째 블록)에 블록에 관련된 목록들을 불러온다.
* 연결 할당의 단점을 보완

운영체제 공부 필요

리펙터리, 디자인 패턴, 쿠버네티스 컨테이너

유닉스 파일 시스템

데이터는 주로 리눅스 위에서 돌아가기 때문이다.

유닉스는 리눅스의 파일 시스템을 그대로 가져온다.

색인 할당 방식으로 파일들을 저장 및 관리하며, 색인 블록을 inode라고 한다.

Inode

Inode 1개당 15개 블록 주소를 저장가능

12개는 직접 블록 : 12개의 블록이 정해져 있다.

3개 간접 블록(single, double, triple) : 또 다른 inode 블록을 가지고 있다.

15개를 모두 간접 블록으로 구성하게 되면 너무 복잡해짐, 속도 저하

효율성과 시간 저장 공간을 비교하여서 12 + 3개로 분리되었다.

데이터베이스 파일 구성

데이터베이스는 여러 개의 파일들로 구성된 형태로 저장

구조 : Field -> Record -> Files -> Database

레코드 블로킹 방식

Fixed blocking

레코드 크기가 고정되어 있음

Variable-length spanned blocking

레코드 크기가 고정되어 있지 않다.

서로 다른 블록에 레코드가 동일하게 존재, 포인터 존재

효율이 좋음

Variable-length unspanned blocking

레코드 크기가 고정되어 있지 않다.

크기가 부족하면 다음 블록에 레코드를 저장

릴레이션에서는 순서를 무시해도 속성별로 나누어 논리적으로 관리하기 때문이다.

물리적인 환경에서는 레코드의 순서가 중요하다.