

# 데이터 저장과 파일

컴퓨터과학과 정재화

# 학습목차

— ① 물리적 저장장치와 파일

— ② 저장장치 관리



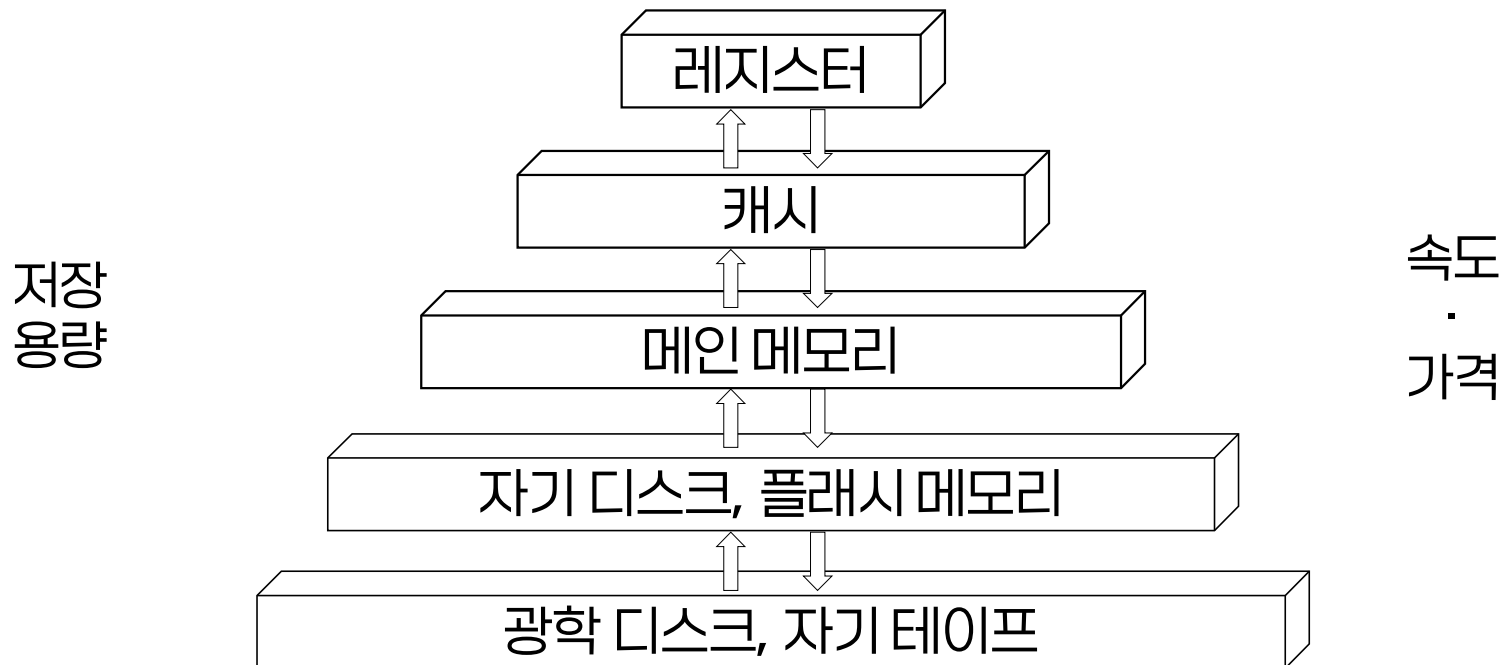
01

# 물리적 저장장치와 파일

- 물리적 저장장치의 구성
- 파일의 구성
- 파일 구조화

# 물리적 저장장치의 구성

- 물리적 저장장치는 데이터 접근 속도, 용량을 기준으로 다양한 장치로 계층적(hierarchical) 구성





## 물리적 저장장치별 특징

◇ 기억 지속성 관점으로 휘발성과 비휘발성으로 구분

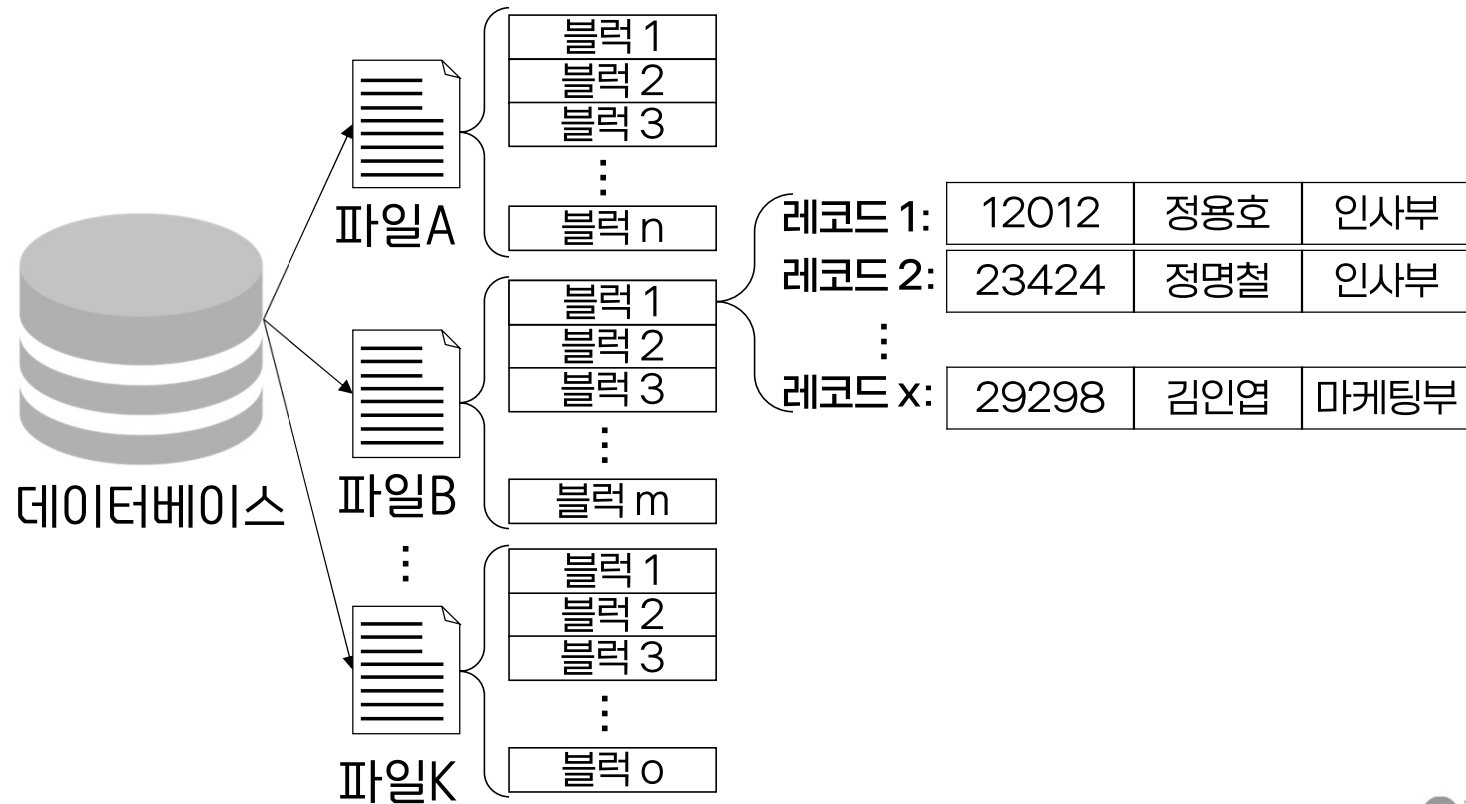
◇ 휘발성 저장장치

- + 캐시: CPU 내부에 위치하여 자주 사용될 것으로 예상되는 데이터를 저장
- + 메인 메모리: 임의 접근이 가능한 고속의 저장공간

◇ 비휘발성 저장장치

- + 플래시 메모리: 메인 메모리와 유사하나 비휘발성
- + 자기 디스크: 자성체를 통해 영구적으로 데이터를 저장
- + 광학 디스크 드라이브: CD, DVD, Blu-ray 등
- + 테이프: 용량이 크고 저렴하나 순차 접근 방식으로 접근 속도가 매우 느림

# 데이터베이스 저장 구조





# 데이터베이스 구성 요소

---

## ▷ 파일

- ⊕ 데이터를 영구적으로 저장하기 위해 사용되는 가장 기초적인 논리적 구조

## ▷ 블록

- ⊕ 파일을 고정적인 길이로 분할하여 생기는 균등한 크기의 데이터 묶음
- ⊕ 일반적으로 메모리와 디스크간 데이터 전송 단위로 결정

## ▷ 레코드

- ⊕ 블록에 저장되는 요소
- ⊕ 관계형 모델에서 분리될 수 없는 최소 데이터 저장 단위

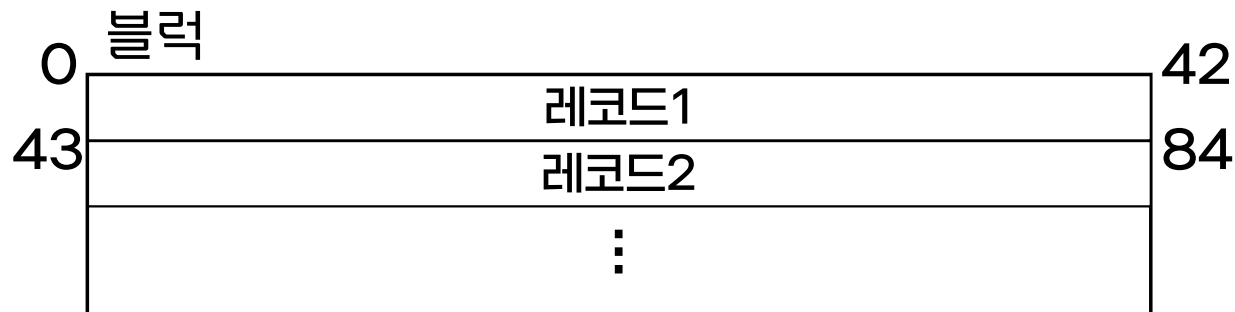
# ☆ 고정 길이 레코드

## ▶ 고정적인 바이트 수를 갖는 레코드 저장 시 고려되는 기법

	CHAR(8)	CHAR(20)	CHAR(10)	INT
사원 릴레이션	사번	이름	부서명	연봉
	12012	정용호	인사부	90,000,000
	12034	임객정	재무부	80,000,000
	13019	이순신	법무지원부	90,000,000
	13030	장보고	인사부	75,000,000
	13044	나철수	시설관리부	80,000,000
	14001	김영희	마케팅부	90,000,000
	14004	유관순	총무부	92,000,000
	14017	안창호	생산부	98,000,000



## ☆ 고정 길이 레코드



### ▷ 데이터 접근

- + 모든 레코드는 42 바이트 크기로 구성
- + i번째 레코드 접근
  - $(i - 1) * 42 + 1$ 번째 바이트부터 42개의 바이트를 읽어 접근



## 고정 길이 레코드 할당

- ▶ 블록의 길이가 레코드 길이로 정확히 나뉘지지 않아 잔여 공간을 비워두는 방법
  - + 블록 내의 남은 공간 낭비

블록1	레코드1
	⋮
	레코드k
블록2	레코드k+1
	⋮



## 고정 길이 레코드 할당

- ▶ 블록의 길이가 레코드 길이로 정확히 나뉘지지 않아 한 레코드를 두 블록에 나누어 저장하는 방법
  - ⊕ 레코드 접근 시 두 개의 블록에 접근 필요

블록1	레코드1
	⋮
	레코드k
	레코드k+1 (1/2)
블록2	레코드k+1 (2/2)
	⋮

## 고정 길이 레코드 할당 시 고려사항

---

### ▷ 레코드 삭제 시

- + 해당 레코드가 저장된 위치에 빈 공간이 생성
- + 장시간 레코드의 삽입 및 삭제 발생 시, 저장 공간에 많은 낭비가 발생

### ▷ 레코드 삭제 시 대처 방안

- + 마지막 레코드로 공백 대체
- + 삭제 레코드 이후의 레코드를 이동
- + 가용 리스트 관리

# ☆ 레코드 삭제 대처

## 1. 마지막 레코드로 공백 대체

사번	이름	부서명	연봉
12012	정용호	인사부	90,000,000
12034	임객정	재무부	80,000,000
13019	이순신	법무지원부	90,000,000
13030	장보고	인사부	75,000,000
13044	나철수	시설관리부	80,000,000
14001	김영희	마케팅부	90,000,000
14004	유관순	총무부	92,000,000
14017	안창호	생산부	98,000,000

# 레코드 삭제 대처

## 2. 삭제 레코드 이후의 레코드를 이동

	사번	이름	부서명	연봉
레코드1	12012	정용호	인사부	90,000,000
레코드2	12034	임객정	재무부	80,000,000
레코드3	13019	이순신	법무지원부	90,000,000
레코드4	<del>13030</del>	<del>장보고</del>	<del>인사부</del>	<del>75,000,000</del>
레코드5	13044	나철수	시설관리부	80,000,000
레코드6	14001	김영희	마케팅부	90,000,000
레코드7	14004	유관순	총무부	92,000,000
레코드8	14017	안창호	생산부	98,000,000

# 고정 길이 레코드


## 3. 가용 리스트 관리

파일헤더	공백 레코드 포인터			
레코드1	12012	정용호	인사부	90,000,000
레코드2	12034	임객정	재무부	80,000,000
레코드3	13019	이순신	법무지원부	90,000,000
레코드4	13030	장보고	인사부	75,000,000
레코드5	13044	나철수	시설관리부	80,000,000
레코드6	14001	김영희	마케팅부	90,000,000
레코드7	14004	유관순	총무부	92,000,000
레코드8	14017	안창호	생산부	98,000,000

## 가변 길이 레코드

---

- ▷ 블록에 저장되는 레코드의 길이가 서로 다른(가변적) 레코드를 할당하는 방법
- ▷ 가변 길이 레코드가 고려되어야 하는 상황
  - + 한 블록 내에 저장되는 레코드 유형이 둘 이상
  - + 길이가 고정되지 않은 컬럼의 개수가 하나 이상
  - + 레코드가 멀티셋을 허용하는 컬럼을 가질 때

 멀티셋

레코드의 컬럼값이 여러 개인 컬럼



# 가변 길이 레코드

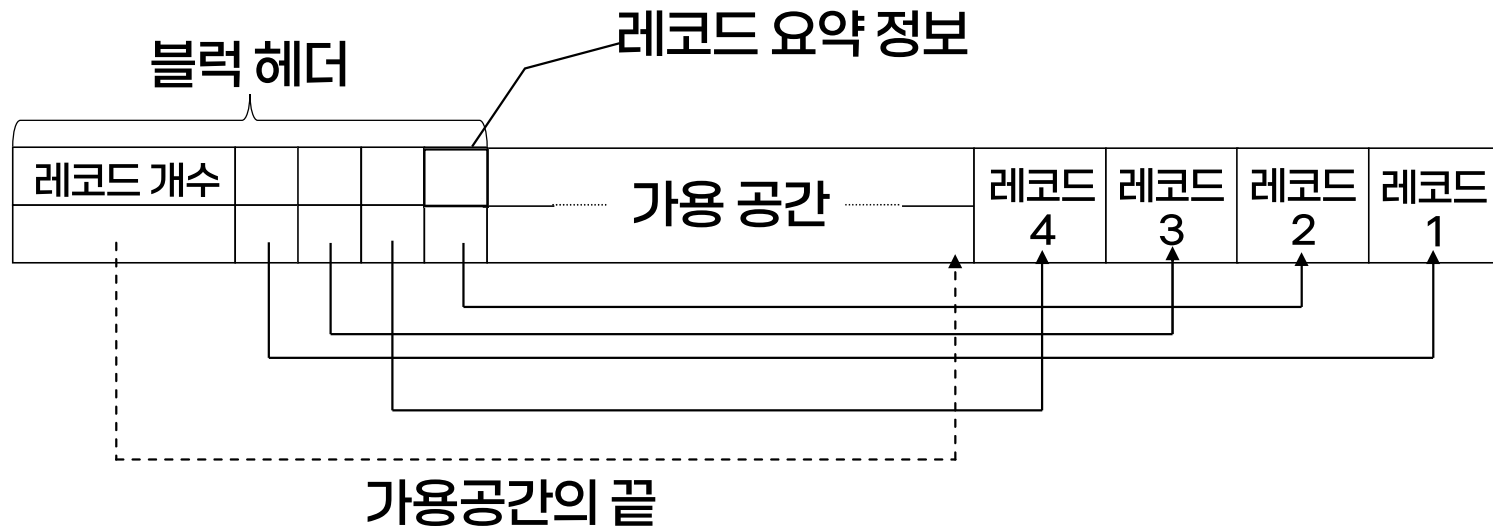
CHAR(8)	VARCHAR(20)	CHAR(10)	INT
사번	이름	부서명	연봉
12012	홍길동	인사부	90,000,000
12034	임꺽정	재무부	80,000,000
13019	이순신	법무지원부	90,000,000
13030	장보고	인사부	75,000,000
13044	나철수	시설관리부	80,000,000
14001	김영희	마케팅부	90,000,000
14004	유관순	총무부	92,000,000
14017	안창호	생산부	98,000,000

바이트 번호 → 0                      4                      12                      22                      26    27                      32

27, 6	12012	인사부	90,000,000	NULL	홍길동
-------	-------	-----	------------	------	-----



# 슬롯페이지 구조

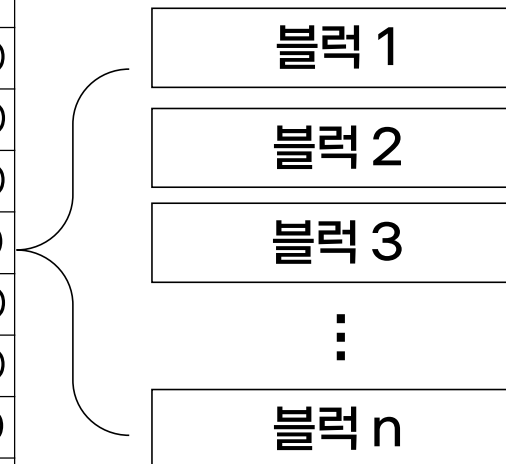


# 파일 구조화 방법

## 파일 구조화

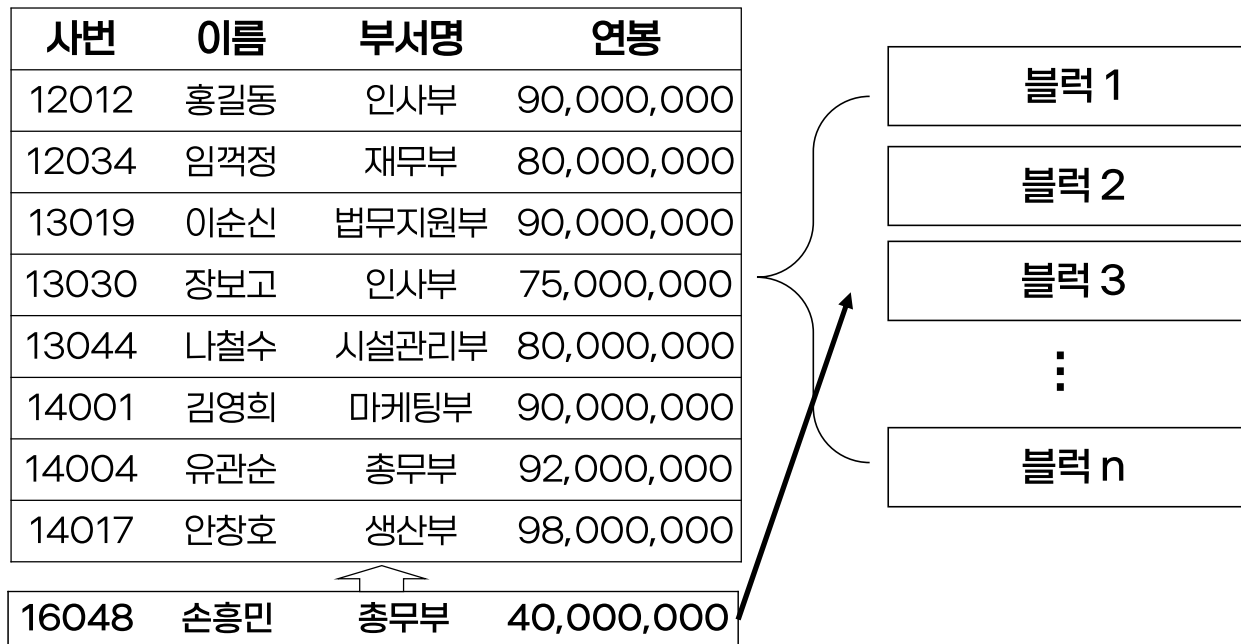
+ 파일 수준에서 레코드를 관리(순서 등)하는 기법

사번	이름	부서명	연봉
12012	홍길동	인사부	90,000,000
12034	임객정	재무부	80,000,000
13019	이순신	법무지원부	90,000,000
13030	장보고	인사부	75,000,000
13044	나철수	시설관리부	80,000,000
14001	김영희	마케팅부	90,000,000
14004	유관순	총무부	92,000,000
14017	안창호	생산부	98,000,000



## **파일 구조화 방법의 종류**

- ◇ **힙 파일 구조:** 저장순서 고려없이 파일 내 임의 블록에 배치
- ◇ **순차 파일 구조:** 레코드가 탐색키 기준으로 정렬되어 저장
- ◇ **해시 파일 구조:** 해시 함수를 사용하여 블록 주소를 계산



# ☆ 순차 파일 구조의 예

탐색키

사번	이름	부서명	연봉
14001	김영희	마케팅부	90,000,000
13044	나철수	시설관리부	80,000,000
14017	안창호	생산부	98,000,000
14004	유관순	총무부	92,000,000
13019	이순신	법무지원부	90,000,000
12034	임객정	재무부	80,000,000
13030	장보고	인사부	75,000,000
12012	홍길동	인사부	90,000,000
16048	손흥민	총무부	40,000,000



## 순차 파일 구조

---

- ▷ 레코드가 탐색키 순서대로 정렬
- ▷ 레코드가 파일에 삽입되는 시점에서 키 값이 부여
- ▷ 장점
  - ⊕ 검색키에 대한 정렬 연산이 불필요, 키 값들의 순서로 레코드를 판독하는 연산에 효율적
  - ⊕ 현재 레코드에서 정렬된 키 순서로 다음 레코드를 찾을 때 추가적인 블록 접근이 불필요
  - ⊕ 이진 탐색을 사용하면 더 빠르게 레코드를 검색
- ▷ 단점
  - ⊕ 레코드 삽입, 삭제에 많은 비용 소요

## **오버플로우 블럭**

-  순차 파일 구조에서 레코드의 정렬된 상태 유지를 위해 삽입된 신규 블럭

사번	이름	부서명	연봉	
14001	김영희	마케팅부	90,000,000	
13044	나철수	시설관리부	80,000,000	
14017	안창호	생산부	98,000,000	
14004	유관순	총무부	92,000,000	
13019	이순신	법무지원부	90,000,000	
12034	임객정	재무부	80,000,000	
13030	장보고	인사부	75,000,000	
12012	홍길동	인사부	90,000,000	
16048	손흥민	총무부	40,000,000	

02

# 저장장치 관리

- 저장장치 접근
- 버퍼 관리자
- 버퍼 교체 전략

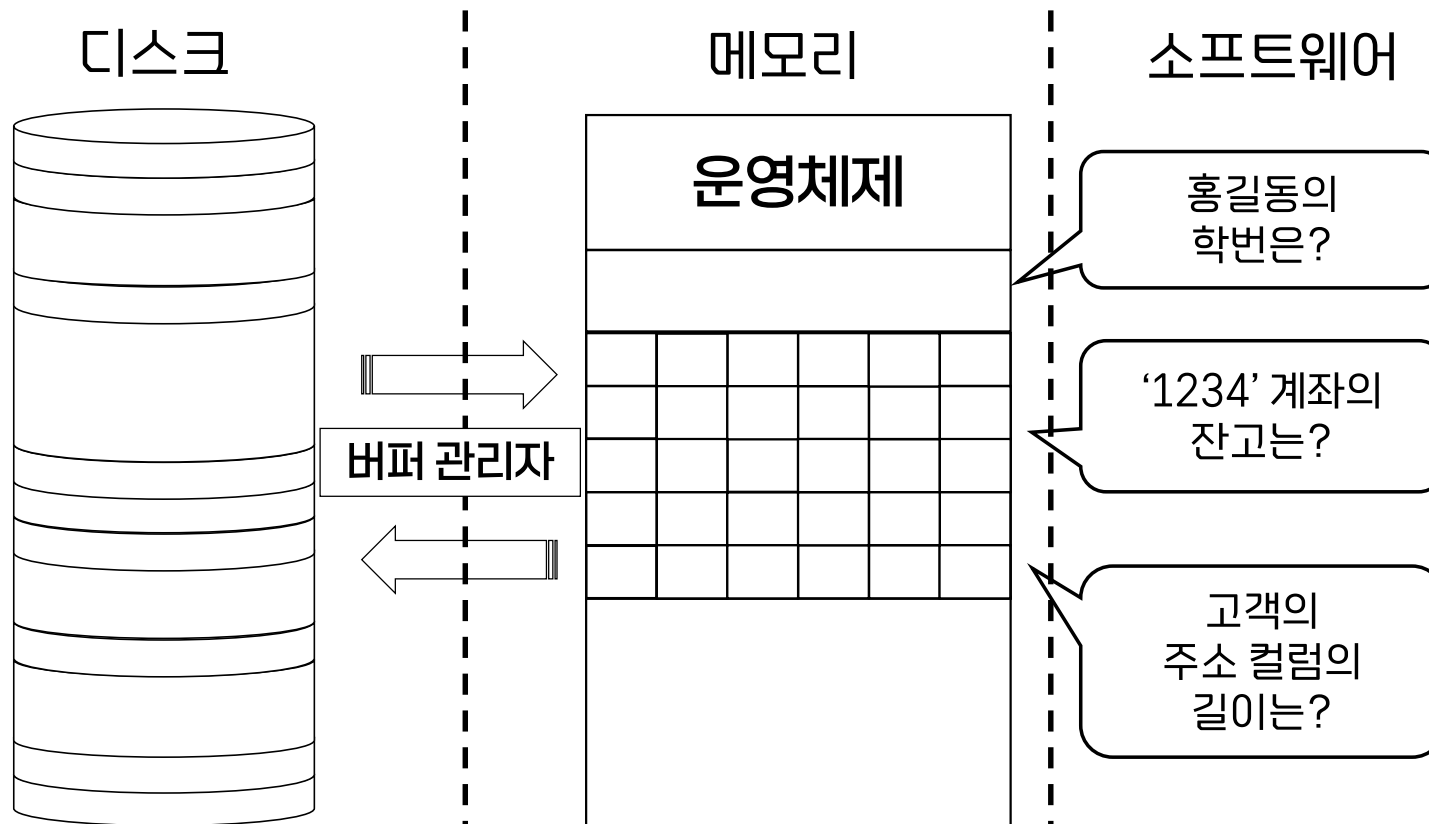




## 저장장치 접근

- ▷ 파일은 논리적 관점에서의 저장 객체
- ▷ 실제 저장될 때에는 여러 개의 물리적 단위인 블록으로 저장
  - + 블록은 메모리와 디스크 간 데이터의 전송 단위
  - + 일반적으로 2KB ~ 32KB 사용
  - + 블록 전송을 최소화할 수록 입출력 소요 시간이 단축
    - 사용 중인 블록을 지속적으로 메모리에 적재
    - 한정적 공간으로 인하여 필요에 따라 특정 블록 할당을 해지
    - 메모리 내부에 버퍼라는 공간에 블록을 저장하고, 이를 관리하기 위한 버퍼 관리자를 사용

# ☆ 저장장치 접근

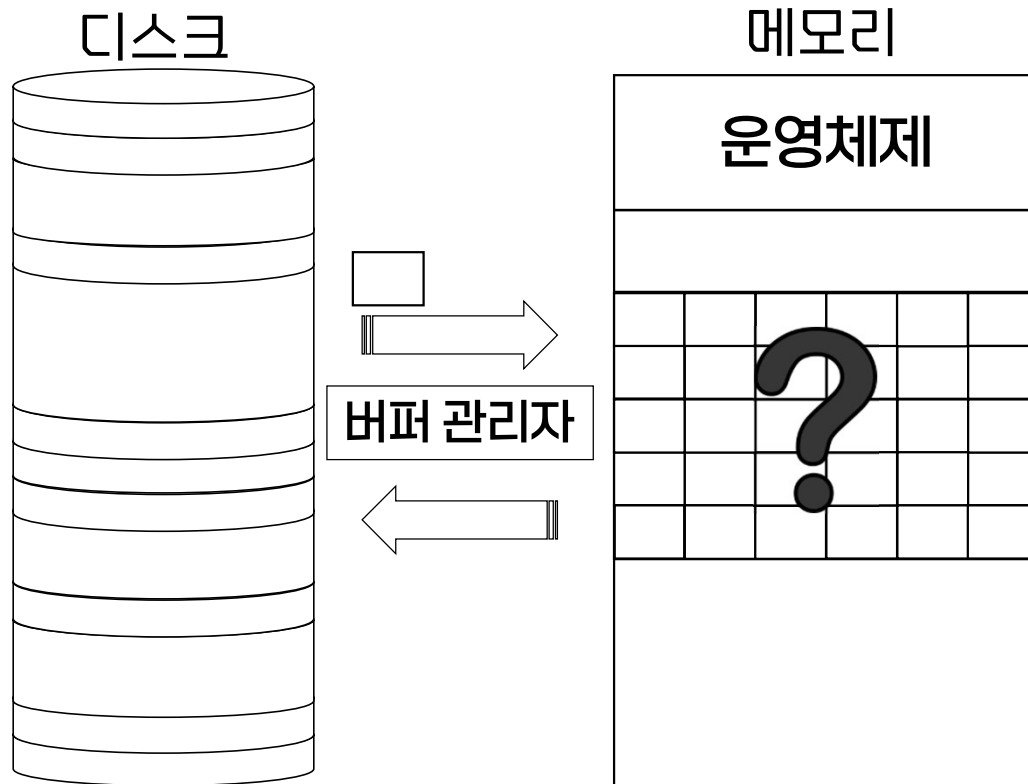


## 버퍼 관리자

- ▷ DBMS가 메모리 내부의 공간인 버퍼(buffer)를 효율적으로 관리하기 위한 하위 시스템
- ▷ DBMS상의 소프트웨어는 필요한 블록이 있을 때 버퍼 관리자에게 해당 블록을 요청
  - ⊕ 요청된 블록이 버퍼에 있다면, 버퍼 관리자는 블록이 위치한 메모리 주소를 프로그램에게 전달
  - ⊕ 요청된 블록이 없는 경우, 버퍼 관리자는 버퍼내의 새로운 공간을 할당하고 해당 블록을 적재
  - ⊕ 더 이상 적재할 공간이 없다면, 버퍼에 있는 기존 블록을 선택하여 할당을 해지하고 해당 블록을 적재

# ☆ 버퍼 교체 전략

---





# 버퍼 관리자의 기능

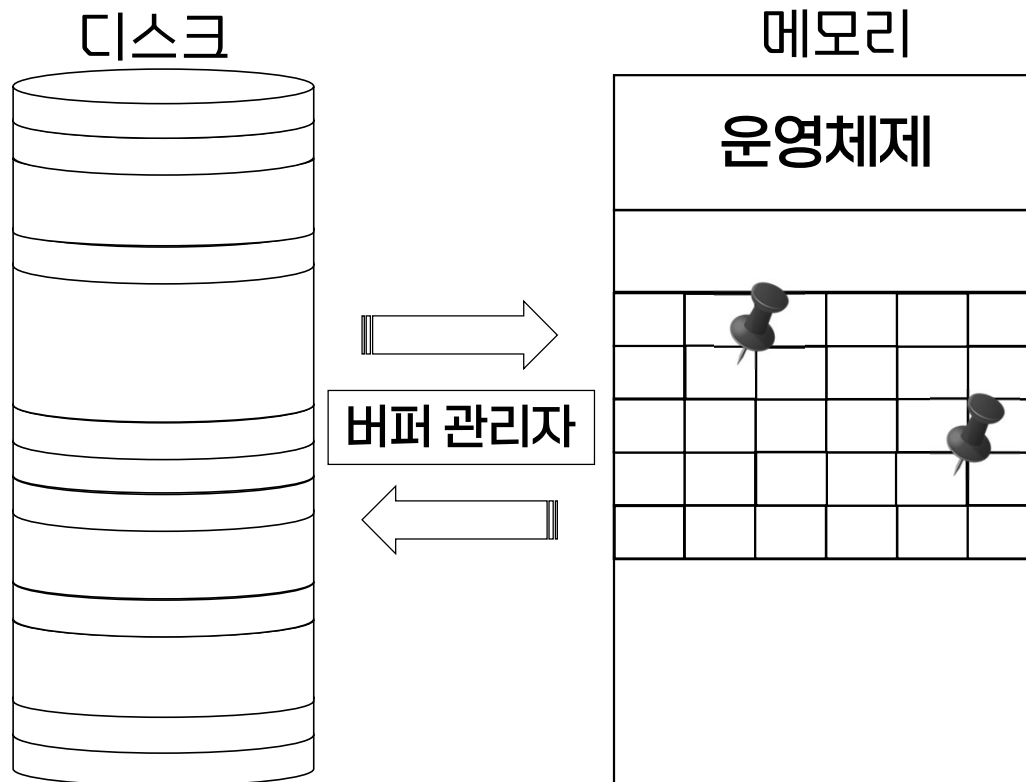
---

## ▷ 버퍼 교체

- + 가용 공간을 확보 하기 위해 기존에 적재된 블록을 특정 기준에 의하여 해지
- + 미래에 가장 적게 사용될 블록을 선택하여 디스크로 내보내는 것이 이상적인 버퍼 교체 전략
- + 버퍼 교체 전략

- LRU(Least Recently Used): 최근에 가장 적게 참조된 블록을 교체
- MFU(Most Frequently Used): 특정 기간동안 가장 여러 번 사용된 블록을 선택하여 블록을 교체

# ☆ 고정 블록과 블록 강제 출력





# 버퍼 관리자의 기능

---

## ▷ 블럭 고정

- + 장애로 메모리의 데이터가 손실되어 작업이 중단될 경우, 중단된 작업의 결과물이 디스크에 기록되는 것을 방지
- + 디스크 블럭이 교체되는 것을 제한

## ▷ 블럭 강제 출력

- + 시스템 로그와 같이 중요한 데이터는 디스크에 영구적으로 기록되어야 할 필요
- + 버퍼 공간이 필요 없어도 강제로 디스크에 기록



다음 시간

**인덱싱**