**제 6장 물리적 데이터베이스 설계**

물리적인 저장 -> 파일 처리 (보조 기억 장치에 저장)

질의의 종류, 빈도를 분석하여 효율적인 접근을 위한 저장 구조와 접근 방법 설계

특정 DBMS의 특성 고려, 인덱스 구조 사용

**6.1 보조 기억장치**

보조기억장치: 전원이 꺼져도 데이터 보존 ex) 디스크

블록 단위 (512byte)사용

주기억장치, 보조기억장치, 테이프 순으로 access time이 짧음

자기 디스크: 보조 기억 장치로 가장 많이 사용

Platter -> track -> sector

\* cylinder: 헤드가 물리적으로 움직이지 않고 읽을 수 있는 데이터 (track의 모임)

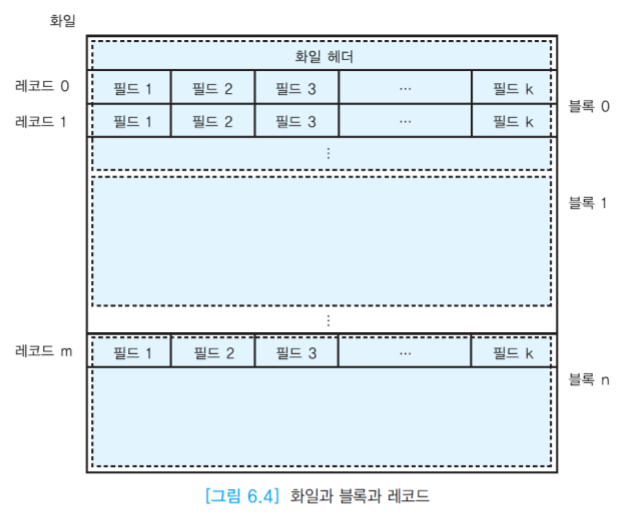
**6.2 버퍼관리와 운영체제**

디스크 입출력 횟수를 줄이면 DBMS의 성능을 올릴 수 있다.

버퍼는 디스크 블록을 저장하는 주기억장치 공간

버퍼관리자는 운영체제의 구성 요소, 버퍼 공간을 할당하고 관리

**6.3 디스크 상에서 파일의 레코드 배치**



필드: 애트리뷰트

레코드: 필드 모임

블록: 레코드 모임

파일: 블록들의 모임

한 파일 상에 속하는 블록은 반드시 인접할 필요가 없다. 인접시키면 fragment가 늘어나게 됨!

But 인접하면 읽는 속도를 증가시킬 수 있다.

BLOB: Binary Large Object 이미지, 동영상 저장

채우기 인수: 블록에 레코드를 채우는 공간의 비율

레코드 삽입 시 기존 레코드들을 이동하는 가능성을 줄일 수 있다.

고정 길이 레코드

레코드의 길이가 정해져 있기 때문에 i번째 레코드가 저장되는 위치를 알 수 있다. = n\*(i-1)+1

Free list: 별도로 존재하는 리스트로 비어있는 공간을 명시. 삭제 시 이동 필요 없음.

클러스터링: arm이 움직이는 데 (seek) 시간이 오래 걸리므로 함께 access될 가능성이 높은 데이터는 같은 cylinder에 위치시킨다. (파일 간: 릴레이션 간 Join으로 실행될 만한 것, 파일 내: 같은 필드를 가진 레코드)

**6.4 파일 조직**

**히프 파일 (비순서 파일)**

단순하게 순서대로 저장, 삽입만 효율적

삽입: 그냥 끝에 저장

검색: 모든 레코드를 순차적으로 검색 -> 특정 파일 검색 시 비효율

삭제: 레코드 삭제 후 공간 재사용X -> 삭제 시 빈 공간이 생기므로 재조직 필요

\*블록킹 인수: 한 블록에 포함되는 레코드 수 (블록크기/레코드크기)

**순차 파일**

한 필드 값 (탐색 키)에 따라 정렬, 탐색 키에 대한 검색만 효율적

삽입: 순서 고려해야 하므로 시간 오래 걸림

검색: 탐색 키에 대한 검색 시 이진 탐색은 효율적. 나머지는 그냥 순차적으로 검색

삭제: 삭제 후 공간 재사용X -> 삭제 시 빈 공간이 생기므로 재조직 필요

직접 파일

**6.5 단일 단계 인덱스**

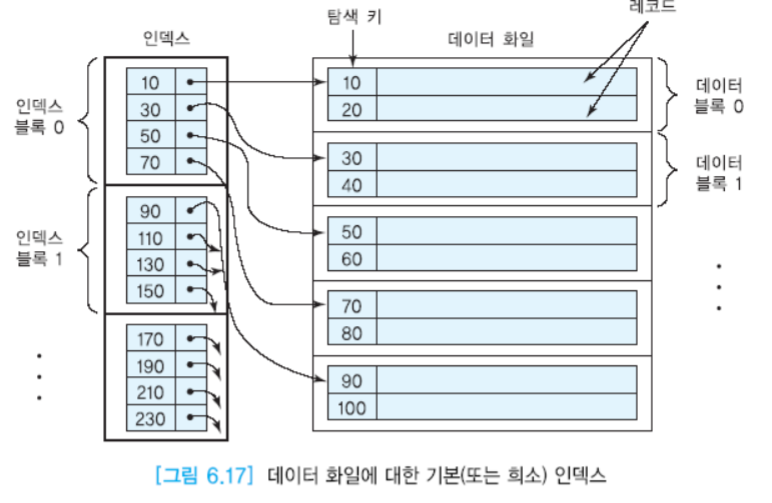
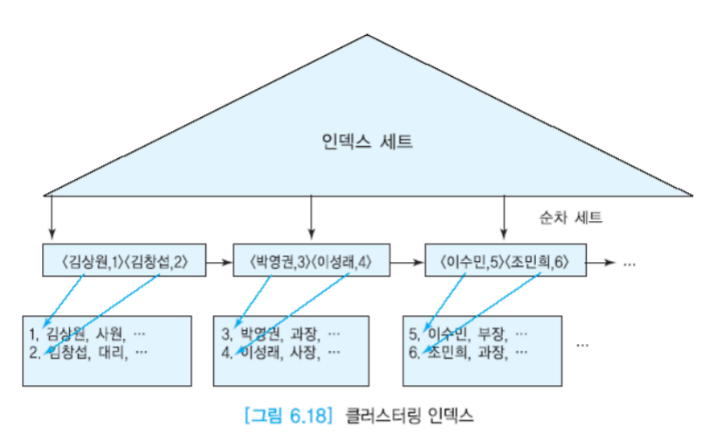
**인덱스된 순차 파일**

인덱스의 각 엔트리는 <탐색 키, 레코드 포인터> 구조, 엔트리는 탐색 키 오름차순 정렬

인덱스를 통해서 레코드에 접근, 인덱스는 별도의 파일에 저장

인덱스의 크기는 데이터 파일에 비해 작기 때문에 한 파일에 데이터보다 많은 인덱스를 저장.

\* access path: 데이터를 얻기 위한 경로, 인덱스 파일 추가 시 선택지가 늘어난다.



기본 인덱스: 탐색 키가 기본 키인 인덱스

희소 인덱스: 블록의 첫 탐색 키만을 인덱스에 포함 (테이블 정렬 필수)

클러스터링 인덱스: 같은 값을 갖는 레코드가 여러 개 있을 경우, 삽입이 빠름 (테이블 정렬 필수)

보조 인덱스: 기본 키가 아닌 필드에 대해 정의. 중복된 것들의 포인터를 가진 리스트를 갖는다.

밀집 인덱스:

**희소 인덱스 vs 밀집 인덱스**

블록마다 엔트리, 레코드마다 엔트리

희소 인덱스의 엔트리 수 < 밀집 인덱스의 엔트리 수

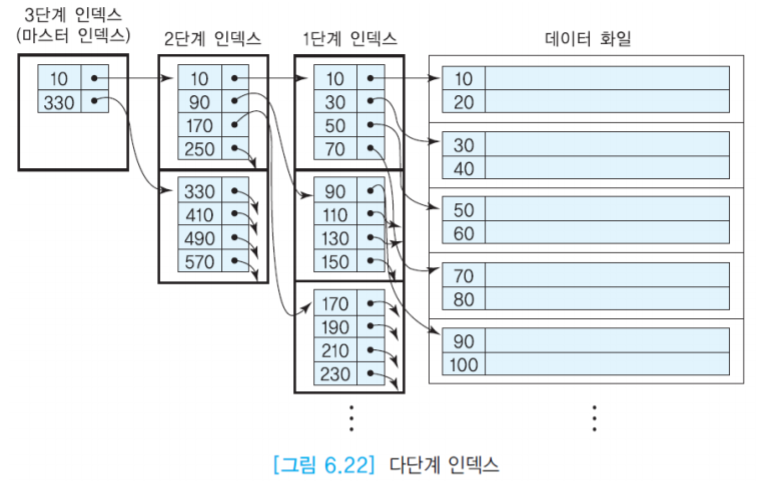
\* 레코드 길이가 블록 크기보다 훨씬 작은 일반 적인 경우

**클러스터링 인덱스 vs 보조 인덱스**

희소 인덱스, 밀집 인덱스

범위 질의에 좋음, 일부 질의는 파일 접근 필요X

**6.6 다단계 인덱스**



단일 단계 인덱스에 대해서 다시 인덱스를 정의

모든 인덱스 엔트리들이 한 블록에 들어갈 수 있을 때까지 반복

가장 상위 단계 인덱스를 마스터 인덱스라고 부름

대부분 B+ 트리 사용

**SQL에서 인덱스 정의문**

Create table문에서 primary key로 명시한 애트리뷰트에 대해서는 자동으로 기본 인덱스 생성

Unique 명시한 애트리뷰트는 자동으로 보조 인덱스 생성

Create index 인덱스이름 On 릴레이션이름 ( 애트리뷰트 )

**인덱스의 장점: 검색 속도 향상, 소수의 레코드 수정 및 삭제 연산 속도 향상**

**인덱스 단점: 삽입, 삭제, 수정 속도 저하**