2. 인덱스 방법(indexed method)

먼저, 인덱스는 데이터베이스 검색 성능을 향상시키기 위한 것이다.**(응답시간 단축)**

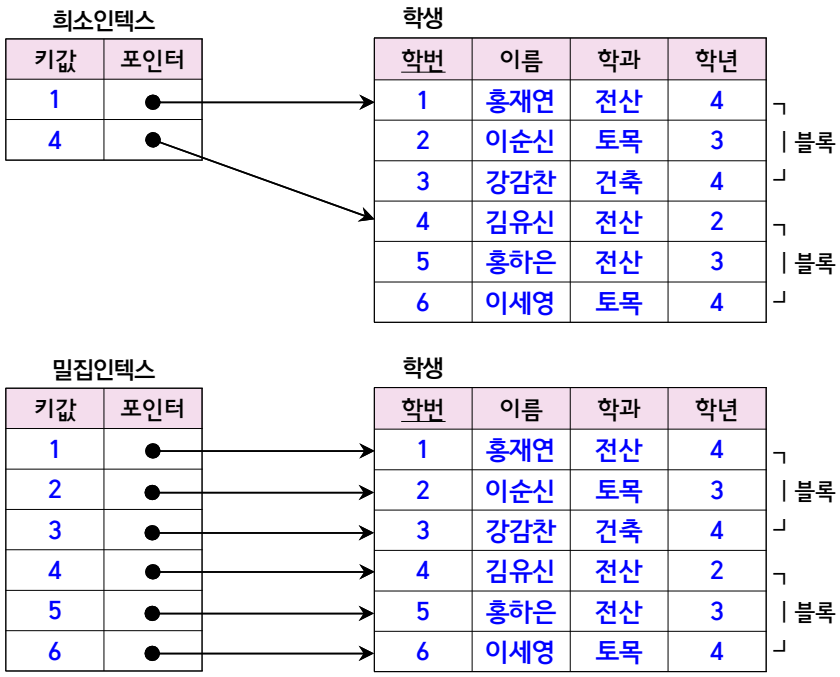
〈인덱스 종류〉

희소인덱스, 밀집인덱스, 기본인덱스, 보조인덱스, 클러스트링인덱스, 비클러스트링인덱스
단일단계인덱스, 다단계인덱스, B-트리, 비트맵인덱스 등

• 이들 **인덱스** 종류는 독립적으로 사용되는 용어가 아니고 **중복**된 개념을 가진다.

// 희소인덱스(sparse index) / 밀집인덱스(dense index)

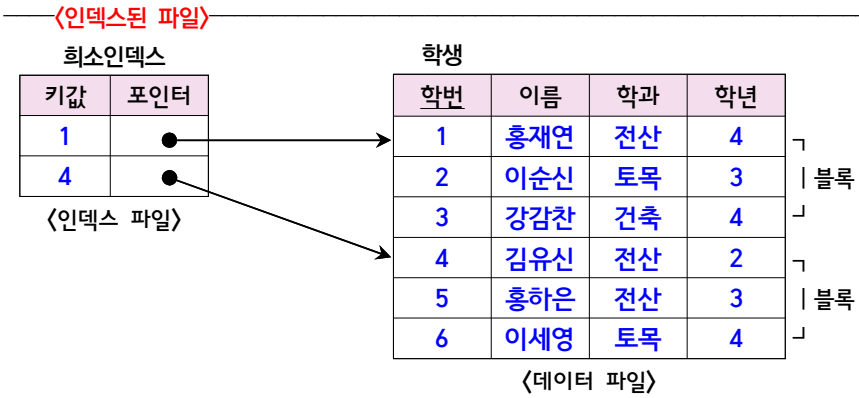
희소인덱스	파일의 **일부** 레코드에 대해서 인덱스 생성**(블록 포인터)**
밀집인덱스	파일의 **모든** 레코드에 대해서 인덱스 생성**(레코드 포인터)**



• 블록은 여러 개의 레코드로 구성된다.

2 http://cafe.daum.net/pass365(홍재연)

1. 인덱스된 파일(indexed file)



- 인덱스된 파일(indexed file)은 **인덱스 파일**(index file)과 **데이터 파일**(data file)로 구성된다.
- 인덱스 파일은 **(키값, 포인터)** 쌍으로 구성된다. 포인터는 **주소**이다.
- 인덱스 파일에서 키값은 데이터 파일을 탐색하는데 사용되는 **탐색키**가 된다.
- 인덱스(index)는 개별 레코드에 대한 임의접근 요구를 지원하는데 사용된다.

2. 기본인덱스와 보조인덱스

기본인덱스(primary index)	기본키(primary key)를 포함하는 인덱스
보조인덱스(secondary index)	기본키(primary key)를 포함하지 않는 인덱스

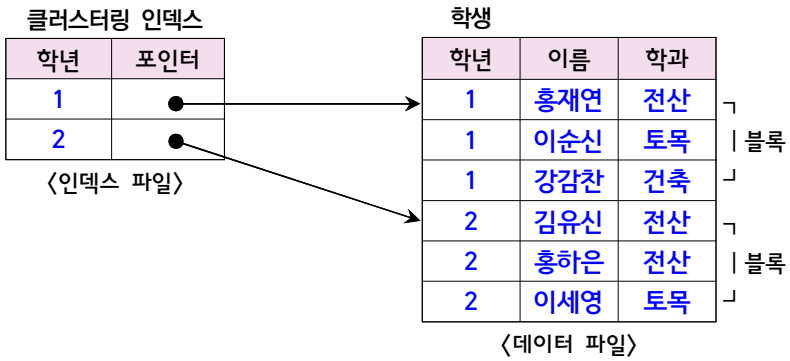
- 학번은 기본키이고(기본인덱스)
- 학과는 기본키가 아니다.(보조인덱스)
- 기본인덱스는 각 테이블마다 최대 한 개만 가질 수 있다.

// 희소인덱스(sparse index) / 밀집인덱스(dense index)

희소인덱스	• 희소인덱스는 파일의 **일부** 레코드에 대해서만 인덱스 생성**(블록 포인터)** • 희소인덱스는 각 블록마다 한 개의 탐색키에 대한 인덱스를 가진다. • 일반적으로, 기본키에 대한 **기본인덱스**는 희소인덱스로 구현된다. • 예 : 학번 int not null primary key **//기본키 학번에 대한 기본인덱스 생성** • 클러스터링(clustering) 인덱스는 희소인덱스에 해당한다.
밀집인덱스	• 파일의 **모든** 레코드에 대해서 인덱스 생성**(레코드 포인터)** • 일반적으로, 보조인덱스는 밀집인덱스로 구현된다. • 비클러스터링(non-clustering) 인덱스는 밀집인덱스에 해당한다.

// 클러스터링(clustering) 인덱스

〈학년에 대한 클러스터링 인덱스〉



- 클러스터링은 논리적으로 연관된 데이터들을 디스크에 인접시켜 저장하는 것을 의미한다.
- 클러스터링 인덱스는 인덱스의 엔트리 순서가 레코드의 **물리적 순서**와 동일하게 유지된다.
- 클러스터링 인덱스는 테이블 당 1개만 생성 가능하다.(국어/영어사전과 같은 구조)
- 클러스터링 인덱스로 지정한 필드에 대해 데이터 파일은 **자동**으로 **정렬**된다.
- 클러스터링 인덱스는 동시에 비슷한 값들을 조회하는 것에 중점을 두는 것이다.
- 클러스터링 인덱스는 비밀집 인덱스(희소 인덱스)의 일종이다.
- 클러스터링 인덱스 설정이 부적절한 경우 : Insert, Delete 연산이 많은 경우
  → Insert, Delete 연산시마다 물리적으로 재정렬 해야 하므로

[예] 클러스터 인덱스 / 비클러스터 인덱스 생성

- 사번 char(4) not null primary key //기본키에 대해 자동으로 클러스터 인덱스 생성
- 사번 char(4) not null primary key **nonclustered** //기본키에 대한 비클러스터 인덱스 생성
- 이름 char(10) unique **clustered** //기본키가 아닌 속성에 대한 클러스터 인덱스 생성

- 기본키(primary key)를 설정하면서 클러스터링 또는 비클러스터링 인덱스 생성이 가능하다.

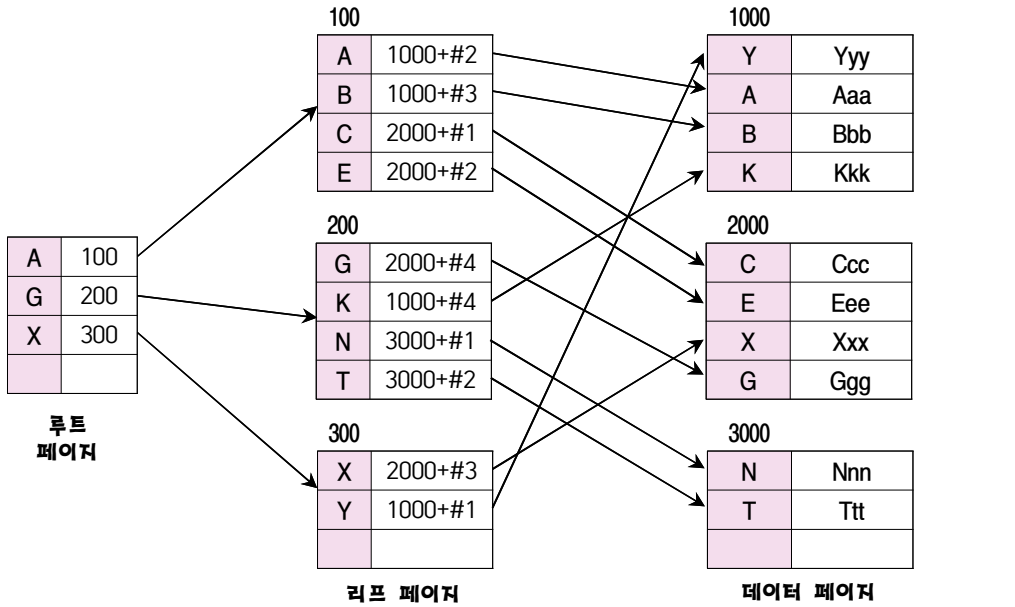
3. 다단계 구조의 보조인덱스

다음과 같은 순서로 데이터가 삽입되었을 때, **다단계 구조의 보조인덱스**를 살펴본다.

```
Insert Into 테이블 Values('Y', 'Yyy');
Insert Into 테이블 Values('A', 'Aaa');
Insert Into 테이블 Values('B', 'Bbb');
Insert Into 테이블 Values('K', 'Kkk');
Insert Into 테이블 Values('C', 'Ccc');
Insert Into 테이블 Values('E', 'Eee');
Insert Into 테이블 Values('X', 'Xxx');
Insert Into 테이블 Values('G', 'Ggg');
Insert Into 테이블 Values('N', 'Nnn');
Insert Into 테이블 Values('T', 'Ttt');
```

↓

〈다단계 구조의 보조인덱스〉



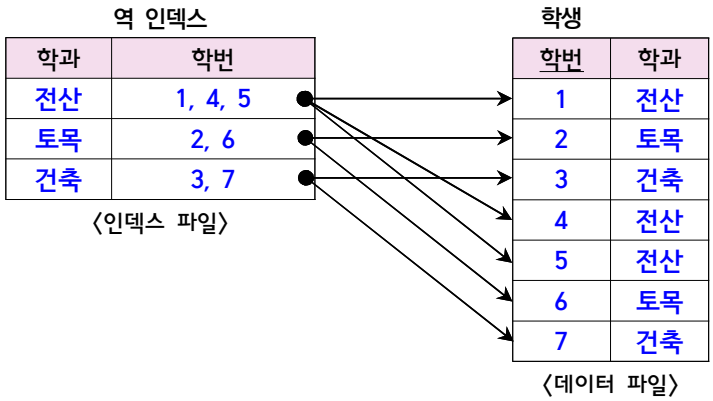
- 보조인덱스는 키 값에 따라 정렬되지 않은 데이터 파일에 대해 정의되는 인덱스이다.
  → 루트 페이지와 리프 페이지는 인덱스이고, 데이터 페이지는 데이터가 저장된 파일이다.
- 보조인덱스라고 하는 이유는 기본적으로 설정된 기본인덱스를 보조하기 때문이다.
- 보조인덱스에 사용된 필드를 보조키(secondary key)라 한다.
- 하나의 데이터 파일에 대해 여러 개의 보조인덱스가 존재할 수 있다.

4. 다중키 파일(multikey file)

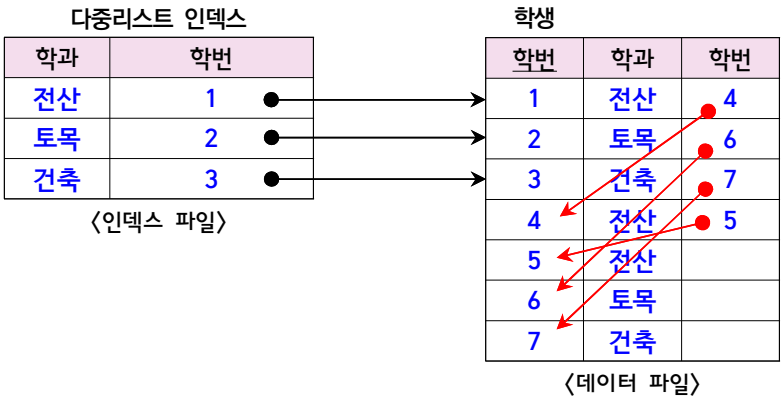
하나의 데이터 파일에 대해 여러 개의 상이한 다중 접근경로를 지원하는 구조이다.

역파일 (inverted file)	· 역 인덱스를 이용하는 구조이다. · 인덱스 엔트리는 〈키, 동일키를 가지는 모든 레코드 포인터〉로 구성 · 가변길이 엔트리 · 데이터 파일 구조를 변경할 필요가 없다.
다중리스트 파일 (multilist file)	· 각 키에 대한 인덱스는 그 키를 가지는 하나의 레코드 포인터만 가진다. · 인덱스 엔트리는 〈키, 하나의 레코드에 대한 포인터〉로 구성 · 내부적으로, 하나의 인덱스에 대한 하나의 데이터 레코드 리스트를 구축 · 고정길이 엔트리 · 데이터 파일 구조 변경이 필요하다.

〈역파일의 인덱스 구조〉



〈다중리스트 파일의 인덱스 구조〉



기출문제 분석

1. 다음은 파일 내의 레코드들에 대한 인덱스 생성 방법을 설명하고 있다. 괄호 안에 들어갈 말로 옳은 것은? [2015년 국가 7급]

○ (㉠)는 인덱스의 엔트리 순서가 레코드의 물리적 순서와 동일하게 유지되는 인덱스이다.
○ (㉡)는 탐색키 값에 따라 정렬되지 않은 데이터 파일에 대하여 정의되는 인덱스이다.
○ (㉢)는 각 레코드마다 하나의 인덱스 엔트리를 갖도록 만드는 인덱스이다.

	㉠	㉡	㉢
①	기본 인덱스	희소 인덱스	클러스터링 인덱스
②	보조 인덱스	밀집 인덱스	희소 인덱스
③	희소 인덱스	클러스터링	인덱스 기본 인덱스
④	클러스터링 인덱스	보조 인덱스	밀집 인덱스

♤ **인덱스 종류**

// 클러스터링 인덱스

- 클러스터링 인덱스는 2개의 필드로 구성된 순서파일이다.
  → 첫 번째 필드는 데이터 파일의 클러스터링 필드 타입의 값을 가지고
  → 두 번째 필드는 데이터 파일의 블록 포인터를 가진다.
- 클러스터링 인덱스는 테이블 당 1개만 생성 가능하다.(국어/영어사전과 같은 구조)
- 테이블 생성에서 기본키로 설정된 필드에 대해 자동으로 클러스터링 인덱스가 생성된다.

// 비클러스터링 인덱스

- 비클러스터링 인덱스는 하나의 테이블에 대해 여러 개 생성할 수 있다.
- 비클러스터링 인덱스는 지정된 속성을 기준으로 자동으로 정렬되지 않는다.

// 보조 인덱스

- 보조 인덱스는 키 값에 따라 정렬되지 않은 데이터 파일에 대해 정의되는 인덱스이다.
- 보조 인덱스라고 하는 이유는 기본적으로 설정된 기본 인덱스를 보조하기 때문이다.

// 밀집 인덱스

- 밀집 인덱스는 각 레코드마다 하나의 인덱스 엔트리를 가지는 인덱스이다.(레코드 포인터)
- 예 : 보조 인덱스에서 리프 페이지 형태의 인덱스 구조가 밀집 인덱스이다.

정답 : ④

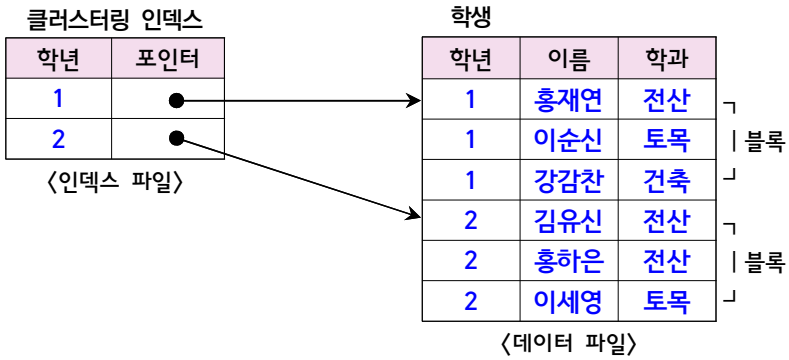
2. 인덱스에 대한 설명으로 옳지 않은 것은? [2022년 국가 7급]

① B+-트리는 다단계 인덱스 구조이다.
② 기본키가 아닌 속성(필드)에도 인덱스를 생성할 수 있다.
③ 삽입과 삭제 연산이 빈번하게 발생하면 인덱스 유지를 위한 부하가 커진다.
④ 탐색하고자 하는 속성이 물리적으로 정렬되어 있지 않은 경우, 클러스터링 인덱스가 효율적이다.

♤ **인덱스**

- 탐색하고자 하는 속성이 **물리적**으로 **정렬되어 있지 않은 경우**, **클러스터링 인덱스**가 **효율적**이다.(×)
  → **클러스터링 인덱스**는 인덱스의 엔트리 순서가 레코드의 **물리적 순서**와 **동일**하게 유지된다.
  → 속성이 **물리적**으로 **정렬되어 있지 않은 경우**, **비클러스터링 인덱스**가 **효율적**이다.(밀집인덱스)
- 클러스터링(clustering) 인덱스는 희소인덱스에 해당한다.
- 희소인덱스는 각 블록마다 한 개의 탐색키에 대한 인덱스를 가진다.
- 일반적으로, 기본키에 대한 기본인덱스는 희소인덱스로 구현된다.
- 예 : 학번 int not null primary key **//기본키 학번에 대한 기본인덱스 생성**

〈학년에 대한 클러스터링 인덱스〉



- 클러스터링은 논리적으로 연관된 데이터들을 디스크에 인접시켜 저장하는 것을 의미한다.
- 클러스터링 인덱스는 인덱스의 엔트리 순서가 레코드의 **물리적 순서**와 **동일**하게 유지된다.
- 클러스터링 인덱스는 테이블 당 1개만 생성 가능하다.(국어/영어사전과 같은 구조)
- 클러스터링 인덱스로 지정한 필드에 대해 데이터 파일은 **자동**으로 **정렬**된다.
- 클러스터링 인덱스는 동시에 비슷한 값들을 조회하는 것에 중점을 두는 것이다.
- 클러스터링 인덱스는 비밀집 인덱스(희소 인덱스)의 일종이다.
- 클러스터링 인덱스 설정이 부적절한 경우 : Insert, Delete 연산이 많은 경우
  → Insert, Delete 연산시마다 물리적으로 재정렬 해야 하므로

정답 : ④

3. 다음 직원 테이블과 부서 테이블로 구성된 데이터베이스에서 '기본인덱스를 사용하여 여러 개의 레코드를 검색하는 방식'으로 구현되는 SELECT 문으로 옳은 것은? (단, 직원 테이블의 부서번호 속성은 부서 테이블 기본키의 외래키이다) [2022년 국가 7급]

○ 직원(직원번호, 이름, 봉급, 부서번호)

○ 부서(부서번호, 부서명, 관리자)

---

① SELECT * FROM 직원 WHERE 부서번호 = 5;

② SELECT * FROM 부서 WHERE 부서번호 > 5;

③ SELECT * FROM 직원 WHERE 직원번호 = 100;

④ SELECT * FROM 직원 WHERE 부서번호 = 5 AND 봉급 > 30000;

♤ **기본인덱스를 사용하여 여러 개의 레코드를 검색하는 방식**

// 기본인덱스와 보조인덱스

기본인덱스(primary index)	기본키(primary key)를 포함하는 인덱스
보조인덱스(secondary index)	기본키(primary key)를 포함하지 않는 인덱스

- 기본인덱스는 각 테이블마다 최대 **한 개**만 가질 수 있다.
- **외래키**는 기본키가 아니다.(보조인덱스)

직원

직원번호	이름	봉급	부서번호
100	John	30000	5
200	Marry	40000	5
300	Jeremy	50000	1
400	Iaan	20000	4
500	John	40000	5

부서

부서번호	부서명	관리자
1	John	데이터베이스
2	Marry	네트워크
3	Jeremy	데이터베이스
4	Iaan	인공지능
5	John	데이터베이스

① SELECT * FROM 직원 WHERE 부서번호 = 5;**(×)**

→ 직원 테이블의 부서번호는 **외래키**로 **기본인덱스가 아니다.**

② SELECT * FROM 부서 WHERE 부서번호 > 5;**(○)**

→ 부서 테이블의 부서번호는 기본키로 **기본인덱스**이다.

③ SELECT * FROM 직원 WHERE 직원번호 = 100;**(×)**

→ 직원 테이블의 직원번호는 **기본키**이므로 **하나의 레코드만 검색**된다.

④ SELECT * FROM 직원 WHERE 부서번호 = 5 AND 봉급 > 30000;**(×)**

→ 직원 테이블의 부서번호는 **외래키**로 **기본인덱스가 아니다.**

정답 : ②