

# KEY

데이터베이스

문승은



# 목차

1. 키(Key) 특성
2. 키(Key) 종류
3. 인덱스(INDEX)와 기본키(PK)
4. 질문에 답변해보기

키(Key)의 특성 및 종류

# 키 특성

## 유일성

키는 각 레코드를 고유하게 식별할 수 있어야 한다  
동일한 키 값을 가진 두 개의 레코드는 존재하지 않는다

## 불변성

한 번 설정된 키 값은 변경되지 않는다  
키 값이 바뀌면 데이터의 식별이 어려워짐: 키 값은 항상 일정하게 유지해야한다

## 최소성

키는 속성의 최소 집합이어야 한다.  
키는 중복되지 않도록 최소한의 속성들로 구성되어야 한다.

# example

유일성 O		최소성 X	
유일성 O		유일성 X	
학번(student_id)		과목코드(course_id)	성적(grade)
100		CS101	A
200		CS102	B
300		CS101	A
400		CS103	C

# 키 종류

## 후보키

: 유일성과 최소성을 만족하는 키

## 슈퍼키

: 유일성은 만족하지만 최소성을 만족하지 않는 키

: 각 튜플을 구분할 수 있는 모든 키

## 대체키(보조키)

: 여러 개의 후보키 중에서 기본키로 선정되고 남은 키

# 기본키

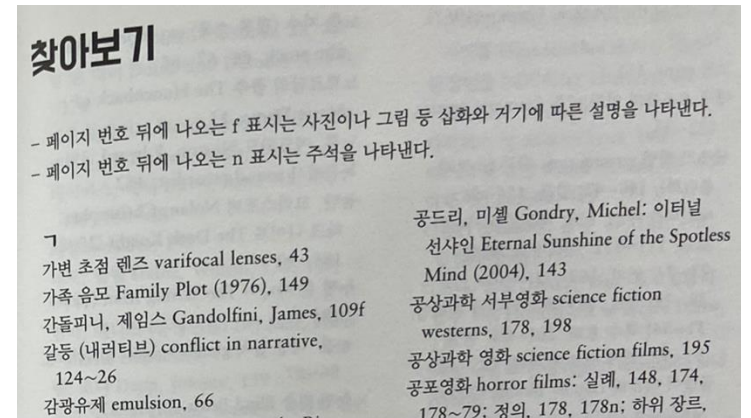
- 후보키 가운데 튜플을 식별하는데 기준으로 사용하는 키
- NULL 불가
- 중복 허용 안됨
- 일반적으로 변경이 불가

인덱스와 기본키



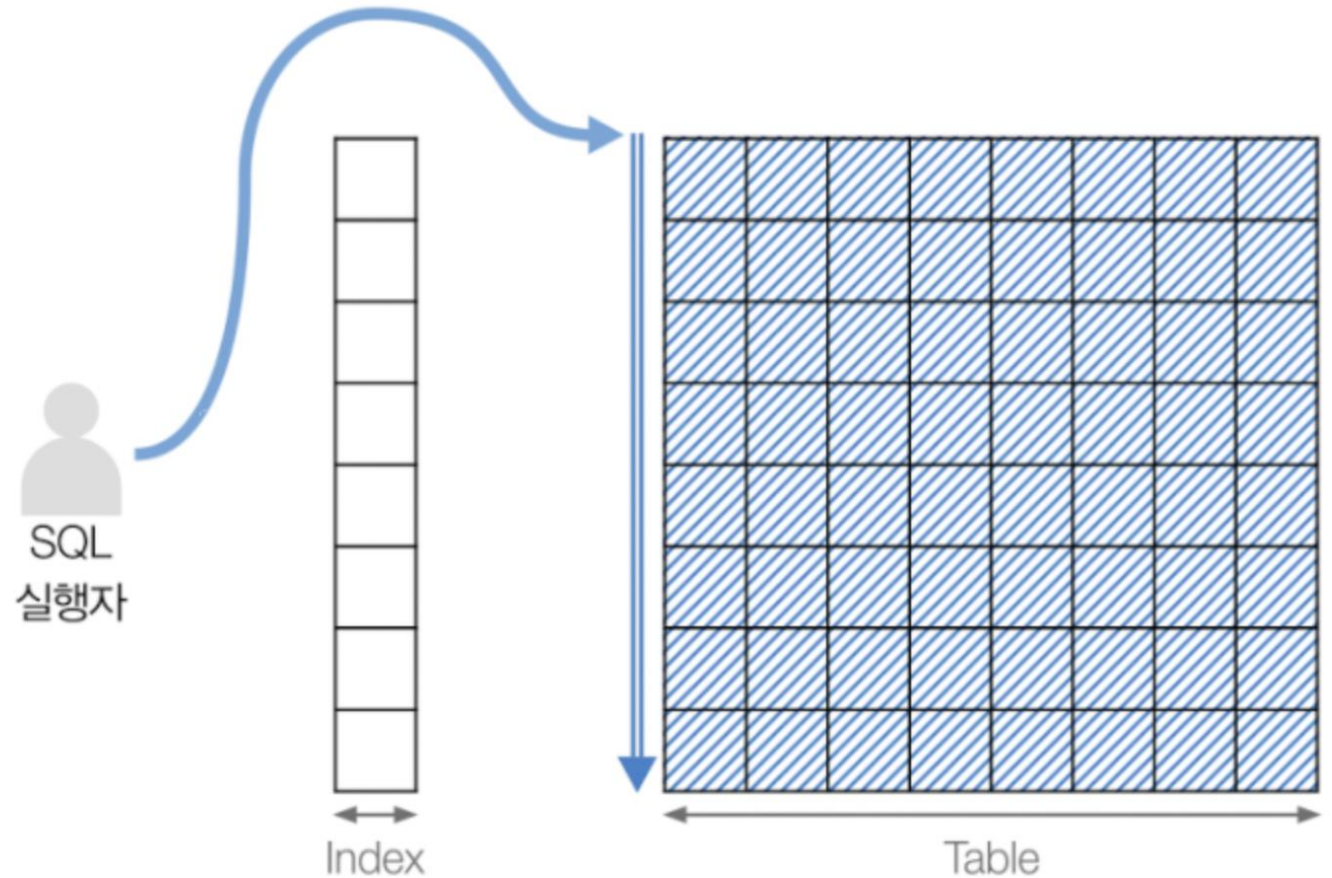
# 인덱스

- 테이블에서 원하는 데이터를 빠르게 찾는 데 사용
- 데이터베이스의 검색 기능을 향상
- WHERE 절, JOIN 등을 통해 활용됨
- READ 성능 향상
- INSERT, DELETE, UPDATE 시 성능 저하 (페이지 분할, 인덱스의 조각화 등)



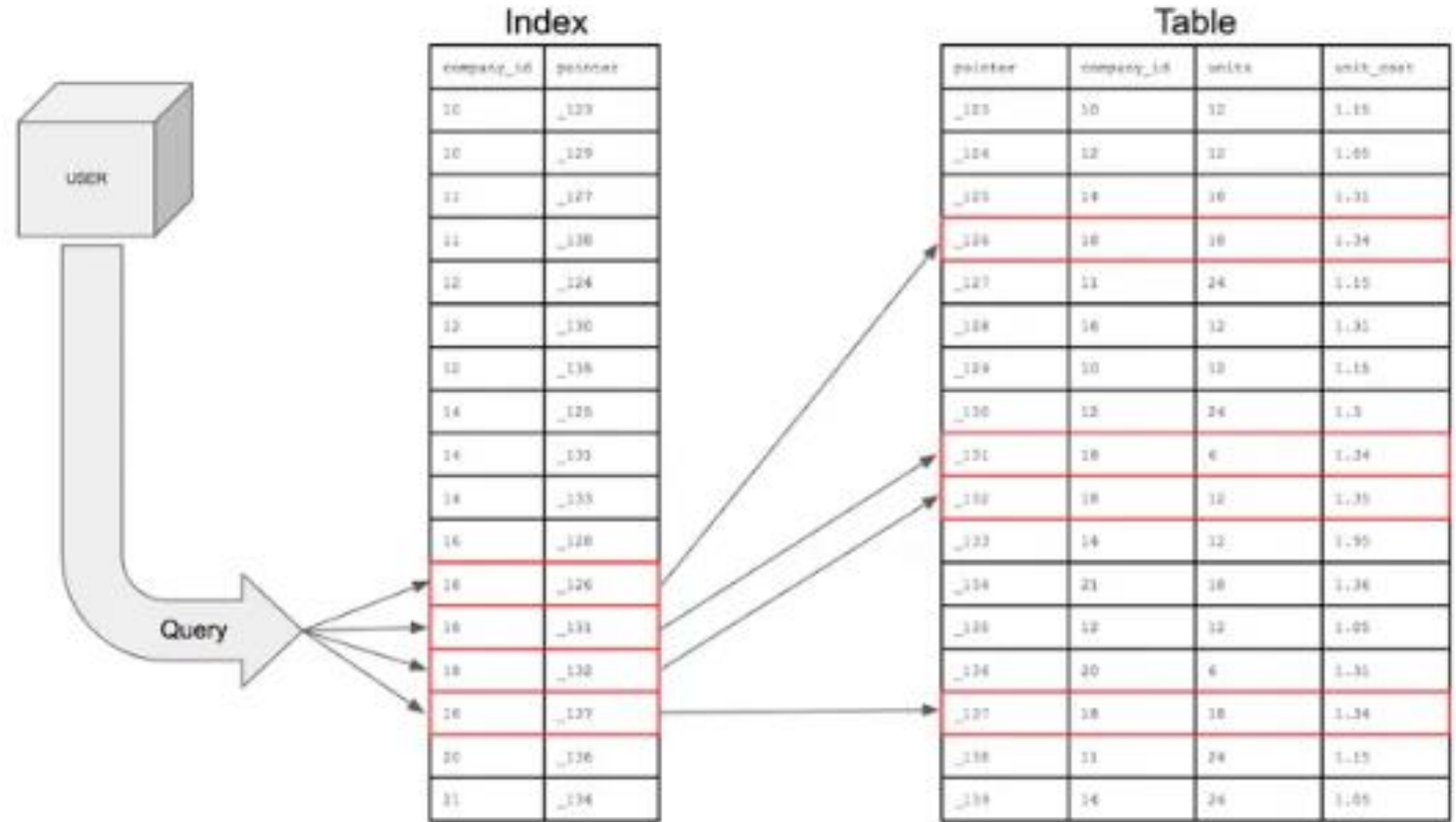
# 조회 성능 향상: 인덱스 없이

```
SELECT *  
FROM User_Profile  
WHERE username = "Jenny"
```



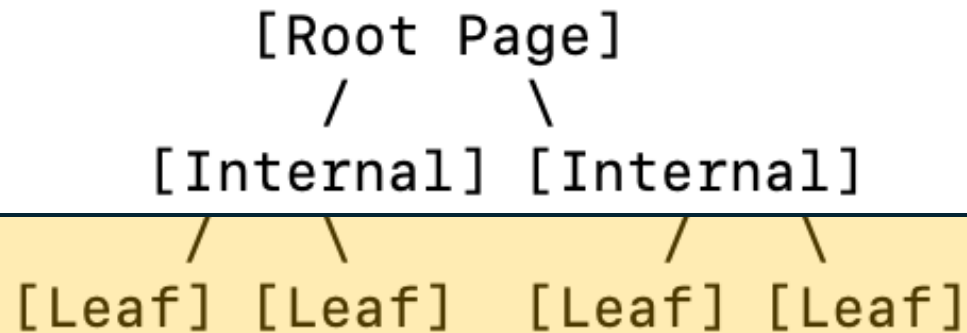
# 조회 성능 향상: PK

```
SELECT *  
FROM User_Profile  
WHERE user_id = 100
```



# 삽입, 삭제, 수정 성능 저

PK 값 수정은 '수정'이 아니라 (1)기존 데이터 삭제 후 (2)새 위치에 데이터 삽입으로 동작합니다.  
이때 새 위치의 페이지가 꽉 찼다면(여기선 5개), (3)페이지 분할이 발생합니다.



수정할 PK:

새 PK:

PK 수정 실행

초기화

## Page 1

[PK: 10] 데이터...

[PK: 11] 데이터...

[PK: 12] 데이터...

[PK: 13] 데이터...

[PK: 15] 데이터...

## Page 2

[PK: 20] 데이터...

[PK: 21] 데이터...

[PK: 22] 데이터...

[PK: 23] 데이터...

[PK: 24] 데이터...

## Page 3

[PK: 30] 데이터...

[PK: 31] 데이터...

[PK: 32] 데이터...

## Page 4

[PK: 40] 데이터...

[PK: 41] 데이터...

[PK: 42] 데이터...

[PK: 43] 데이터...

[PK: 44] 데이터...

### Page 1

[PK: 10] 데이터...

[PK: 11] 데이터...

[PK: 12] 데이터...

[PK: 13] 데이터...

### Page 2

[PK: 20] 데이터...

[PK: 21] 데이터...

[PK: 22] 데이터...

[PK: 23] 데이터...

[PK: 24] 데이터...

### Page 3

[PK: 30] 데이터...

[PK: 31] 데이터...

[PK: 32] 데이터...

### Page 4

[PK: 40] 데이터...

[PK: 41] 데이터...

[PK: 42] 데이터...

### Page 5

[PK: 43] 데이터...

[PK: 44] 데이터...

[PK: 45] 데이터...

# 클러스터링 인덱스

클러스터링 인덱스 자동생성

1. PK

2. NOT NULL + UNIQUE

- 실제 데이터 자체가 정렬
- 테이블 당 1개만 존재 가능
- 리프 페이지가 데이터 페이지

**Root Node**

(PK < 3 ? Page 1 : Page 2)

**Leaf Node (Page 1)**

[**1**, 'alice',  
'alice@...', 25] (모든 데이터)

[**2**, 'bob', 'bob@...',  
30] (모든 데이터)

**Leaf Node (Page 2)**

[**3**, 'charlie',  
'charlie@...', 40] (모든 데이터)

[**4**, 'david',  
'david@...', 35] (모든 데이터)

# 논클러스터링 인덱스

1. UNIQUE 제약
  2. UNIQUE Index → 중복을 허용하지 않는 인덱스
  3. Index → 중복을 허용하는 인덱스
- 별도의 인덱스 페이지가 추가됨(리프 페이지)
  - 테이블 당 여러 개 존재
  - 리프 페이지에 실제 데이터 페이지 주소를 담고 있음

## Root Node

(Name < 'c' ? Page A : Page B)

## Leaf Node (Page A)

['alice', 1] (PK 포인터만)

['bob', 2] (PK 포인터만)

## Leaf Node (Page B)

['charlie', 3] (PK 포인터만)

['david', 4] (PK 포인터만)

# 클러스터링 vs 논클러스터링

## 사전 vs 색인

### Root Node

(PK < 3 ? Page 1 : Page 2)

### Leaf Node (Page 1)

[**1**, 'alice',  
'alice@...', 25] (모든 데이터)

[**2**, 'bob', 'bob@...',  
30] (모든 데이터)

### Leaf Node (Page 2)

[**3**, 'charlie',  
'charlie@...', 40] (모든 데이터)

[**4**, 'david',  
'david@...', 35] (모든 데이터)

### Root Node

(Name < 'c' ? Page A : Page B)

### Leaf Node (Page A)

['alice', **1**] (PK 포인터만)

['bob', **2**] (PK 포인터만)

### Leaf Node (Page B)

['charlie', **3**] (PK 포인터만)

['david', **4**] (PK 포인터만)



# 인덱스와 기본키

기본 키 생성 시 클러스터링 인덱스가 생성된다.

=> 이 클러스터링 인덱스는 READ 성능에 좋다

=> 다만 INSERT, DELETE, UPDATE 성능 저하

# 외래키

다른 테이블의 기본키를 참조하는 속성

참조 무결성: 외래키 열의 값이 참조되는 테이블의 기본키 열에 존재해야함

Team (부모 테이블)	
team_id (PK)	team_name
10	'영업팀'
20	'개발팀'

Member (자식 테이블)		
member_id (PK)	member_name	team_id (FK)
101	'Alice'	10
102	'Bob'	20
103	'Charlie'	NULL
104	'David'	30

질문에 답하기

기본키는  
수정이 가능한가요?

가능합니다.

그러나 기본키는 클러스터링  
인덱스이기 때문에 수정 시 페  
이지 분할이 일어날 수 있습니  
다.

또한, 외래키의 수정에서 매우  
비효율적인 상황이 발생할 수  
있습니다. 따라서 성능저하로  
기본키는 수정하지 않는 것이  
 좋습니다.

사실 MySQL의 경우, 기본 키를 설정하지 않아도 테이블이 만들어집니다. 어떻게 이게 가능한 걸까요?

기본키를 설정하지 않고 테이블을 만들면

MySQL은 첫번째로

NOT NULL + UNIQUE 조합을 찾아 기본키로 사용합니다.

이 조합조차 없다면

RowID를 자체적으로 생성하는데 1부터 순차적으로 사용하게 되며 인덱스의 기능을 사용할 수 없어 집니다.

따라서 조회에서의 이점이 떨어 집니다.

외래키 값은 NULL이 들어  
올 수 있나요?

NULL 값이 들어갈 수 있습니다.

외래키 값은 참조하는 테이블의 기본키 값만을 참조해야하는 참조 무결성의 특징을 가집니다.

어떤 칼럼의 정의에 UNIQUE 키워드가 붙는다고 가정해 봅시다. 이 칼럼을 활용한 쿼리의 성능은 그렇지 않은 것과 비교해서 어떻게 다를까요?

UNIQUE 키워드 적용 시  
논 클러스팅 인덱스가 만들어집니다.

PK보다는 추가작업이 필요하지만 인덱스가 만들어지면 조회 성능이 매우 빨라집니다.

Thank You