

# Chapter 09

## 뷰와 인덱스



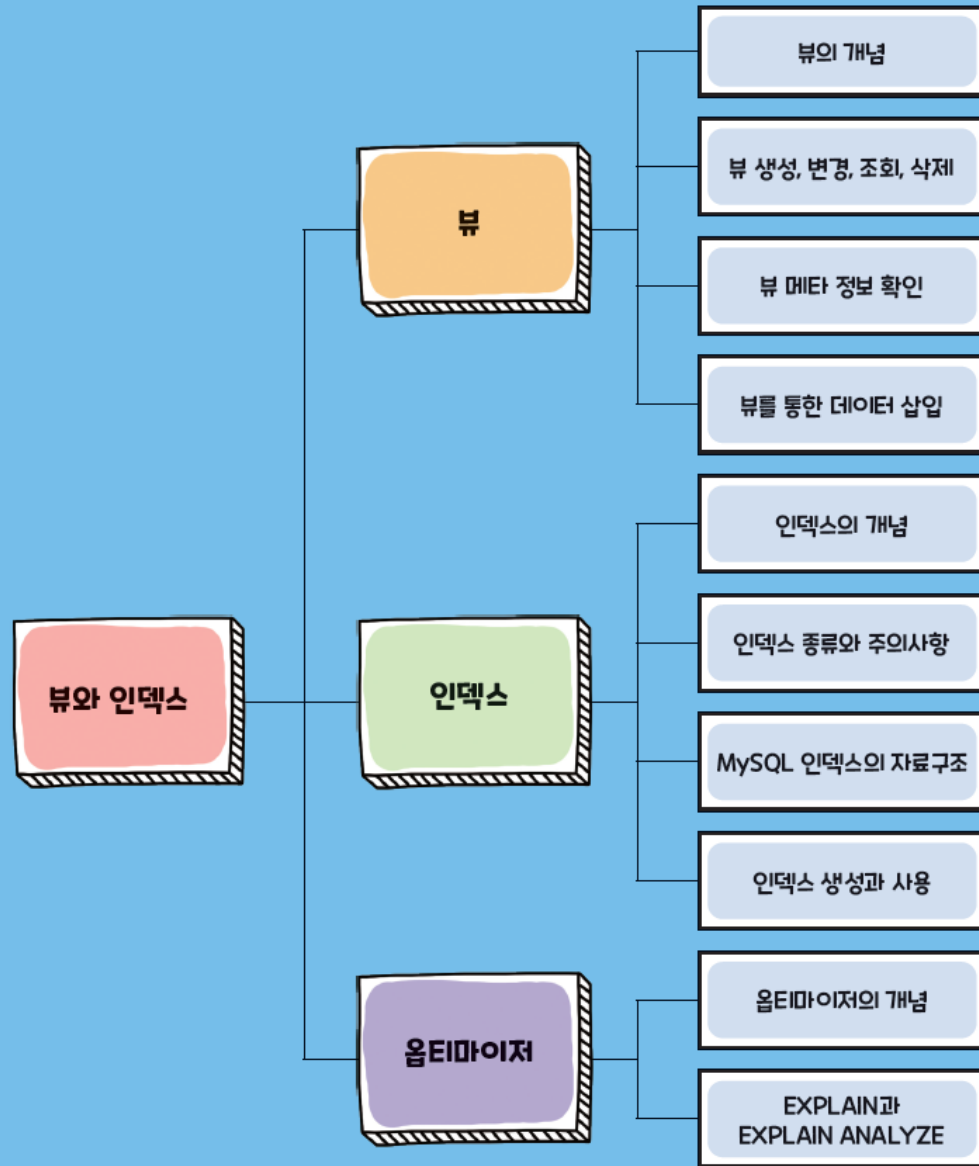
# 목차

1. 뷰
2. 뷰를 통한 데이터 삽입
3. 인덱스
4. 옵티마이저

# 학습목표

- 뷰의 개념과 사용법을 이해하고 뷰를 생성할 수 있습니다.
- 인덱스의 개념과 B-트리 인덱스를 이해할 수 있습니다.
- 옵티마이저의 개념을 이해하고 EXPLAIN, EXPLAIN ANALYZE를 이해할 수 있습니다.

## Preview



# Section 01

부

# 1. 뷰의 개념

## ■ 뷰(view)

- 한 개 이상의 테이블을 기반으로 생성된 가상의 테이블
- 뷰는 실제 데이터를 저장하지 않으며, 쿼리 실행 시점에 생성된 쿼리 결과를 가상의 테이블로 만들어서 제공함
- 다시 말해 뷰는 물리적으로 데이터를 저장하는 것이 아니라 SELECT문을 저장한 형태로서 스토어드 쿼리(Stored Queries)라고도 함
- 뷰 실행 시에는 테이블을 조회한 것과 유사한 형태로 데이터를 확인할 수 있음

# 1. 뷰의 개념

## ■ 뷰의 장점

- 데이터 중복성 최소화
- 데이터 보안성
- 간편한 데이터 접근
- 데이터 가상화

고객 테이블

고객번호	고객회사명	도시	전화번호
ACDDR	국도남서울	서울특별시	(02)970-1984
ADHTR	열개 이 상사	서울특별시	(02)345-1945
AJHTR	씨엔그룹	광명시	(02)31-0345
ANKFR	오리인무역	서울특별시	(02)123-0345
ANKFR	남해종합식품	구리시	(034)756-2091
ANSFR	서보업연매로	서울특별시	(02)211-2221
ATRAH	대정 인터내셔널	서울특별시	(02)681-6889
AUSBL	제철인터내셔널	서울특별시	(02)811-1234

주문 테이블

주문일	고객번호	주문일	요청일	발송일
2020-03-13	MSPTO	2020-03-13	2020-04-24	2020-03-18
2020-03-16	NARHA	2020-03-16	2020-04-13	2020-03-20
2020-03-16	CTEVI	2020-03-16	2020-04-13	2020-03-23
2020-03-17	PRDSU	2020-03-17	2020-04-14	2020-03-19
2020-03-18	NARHA	2020-03-18	2020-04-01	2020-03-24
2020-03-19	OPSCB	2020-03-19	2020-04-16	2020-03-31
2020-03-20	CSLRI	2020-03-20	2020-04-17	2020-03-23
2020-03-23	LLWYE	2020-03-23	2020-04-20	2020-03-25

뷰 생성

view\_고객\_주문

고객번호	고객회사명	도시	전화번호	주문번호	주문일	요청일	발송일
MSPTO	보라리	부천시	(032)110-1238	H0249	2020-03-13	2020-04-24	2020-03-18
NARHA	청운유통	계천시	(0443)110-2181	H0250	2020-03-16	2020-04-13	2020-03-20
CTEVI	한국약품로 무역	인천광역시	(032)58-1984	H0251	2020-03-16	2020-04-13	2020-03-23
PRDSU	황인테크라이시스	서울특별시	(02)934-1984	H0252	2020-03-17	2020-04-14	2020-03-19
NARHA	청운유통	계천시	(0443)110-2181	H0253	2020-03-18	2020-04-01	2020-03-24

뷰를  
사용하니 작업이  
간단해지는구나!

SELECT \*  
FROM View\_고객\_주문;

그림 9-1 뷰의 개념

## 2. 뷰 생성 및 변경하기

### ■ 뷰 생성

- 형식1: OR REPLACE 사용

```
CREATE [OR REPLACE] VIEW 뷰명  
AS  
SELECT문;
```

- 형식2: ALTER 사용

```
ALTER VIEW 뷰명  
AS  
SELECT문;
```



## 2. 뷰 생성 및 변경하기

- [예제 9-1] 한빛무역 데이터베이스에서 사원 테이블을 사용하여 사원의 이름, 집전화, 입사일, 주소를 보이는 뷰를 작성하시오.

- 뷰명: view\_사원

- ✓ 뷰를 생성할 때 컬럼명에 별명을 붙일 수 있으며, 뷰를 사용할 때에는 컬럼명에 붙인 별명을 사용해야 함

```
CREATE OR REPLACE VIEW view_사원
AS
SELECT 이름
      ,집전화 AS 전화번호
      ,입사일
      ,주소
FROM 사원;
```

- ✓ 컬럼 또는 컬럼의 별명을 뷰 이름 옆에 나열할 수도 있음

```
CREATE OR REPLACE VIEW view_사원(이름, 전화번호, 입사일, 주소)
AS
SELECT 이름
      ,집전화
      ,입사일
      ,주소
FROM 사원;
```

## 2. 뷰 생성 및 변경하기

- [예제 9-1] 한빛무역 데이터베이스에서 사원 테이블을 사용하여 사원의 이름, 집전화, 입사일, 주소를 보이는 뷰를 작성하시오.
  - 뷰명: view\_사원
    - ✓ 뷰를 생성한 이후에는 뷰를 테이블인 것처럼 조회할 수 있음

```
SELECT *  
FROM view_사원;
```

### ▶ 실행결과

이름	전화번호	입사일	주소
이소미	(02)578-8988	2019-04-13	강남구 역삼동 36-8
배재용	(032)69-0136	2019-01-01	원미동 16-11
유대현	(062)73-0256	2019-03-14	광산구 숭정동 100-11
최소민	(051)587-4783	2019-04-15	중구 중앙동 57-14

## 2. 뷰 생성 및 변경하기

- [예제 9-2] 제품 테이블, 주문세부 테이블을 조인하여 제품명과 주문수량합을 보이는 뷰를 작성하시오
  - 뷰명: view\_제품별주문수량합
    - ✓ 조인이나 서브쿼리 등 여러 테이블을 사용한 쿼리문으로 뷰를 생성 가능

```
CREATE OR REPLACE VIEW view_제품별주문수량합
AS
SELECT 제품명
      ,SUM(주문수량) AS 주문수량합
FROM 제품
INNER JOIN 주문세부
ON 제품.제품번호 = 주문세부.제품번호
GROUP BY 제품명;
```

```
SELECT *
FROM view_제품별주문수량합;
```

### ▶ 실행결과

제품명	주문수량합
썬 100% 오렌지 주스	828
썬 100% 레몬 주스	1033
썬 제리 시럽	318
썬 100% 복숭아 시럽	

## 2. 뷰 생성 및 변경하기

- [예제 9-3] '여' 사원에 대하여 사원의 이름, 집전화, 입사일, 주소, 성별을 보이는 뷰를 작성하시오
  - 뷰명: view\_사원\_여

```
CREATE OR REPLACE VIEW view_사원_여
AS
SELECT 이름
      ,집전화 AS 전화번호
      ,입사일
      ,주소
      ,성별
FROM 사원
WHERE 성별 = '여';
```

```
SELECT *
FROM view_사원_여;
```

### ▶ 실행결과

이름	전화번호	입사일	주소	성별
이소미	(02)578-8988	2019-04-13	강남구 역삼동 36-8	여
최소민	(051)587-4783	2019-04-15	중구 중앙동 57-14	여
선하라	(0652)983-1985	2019-02-16	목진구 고랑동 116	여
유가를	(053)465-1248	2019-10-29	남구 대명동 19-7	여

### 3. 뷰 조회하기

#### ■ 뷰 조회하기

- 형식

```
SELECT *  
FROM 뷰명;
```

- [예제 9-4] [예제 9-3]에서 생성한 'view\_사원\_여' 뷰로 전화번호에 '88'이 들어간 사원의 정보를 검색하시오.

```
SELECT *  
FROM view_사원_여  
WHERE 전화번호 LIKE '%88%';
```

#### ▶ 실행결과

이름	전화번호	입사일	주소	성별
이소미	(02)578-8988	2019-04-13	강남구 역삼동 36-8	여

- [예제 9-5] [예제 9-2]에서 생성한 'view\_제품별주문수량합' 뷰로 주문수량합이 1,200개 이상인 레코드를 검색하시오.

```
SELECT *  
FROM view_제품별주문수량합  
WHERE 주문수량합 >= 1200;
```

#### ▶ 실행결과

제품명	주문수량합
통루 바닐라 아이스크림	1397
대림 옥수수 가루	1263
그린 포장 치즈	1496
그린 파메상 치즈	1575

## 4. 뷰 메타 정보 확인하기

### ■ 뷰 메타 정보 확인하기

- 형식1: INFORMATION\_SCHEMA.VIEWS 사용

```
SELECT *  
FROM INFORMATION_SCHEMA.VIEWS  
WHERE TABLE_NAME = '뷰명';
```

- 형식2: SHOW CREATE VIEW 사용

```
SHOW CREATE VIEW 뷰명;
```

## 4. 뷰 메타 정보 확인하기

- [예제 9-6] 'view\_사원' 뷰의 메타 정보를 확인하시오.

```
SELECT *  
FROM INFORMATION_SCHEMA.VIEWS  
WHERE TABLE_NAME = 'view_사원';
```

### ▶ 실행결과

TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	VIEW_DEFINITION	CHECK_OPTION	IS_UPDATABLE	DEFINER	SECURITY_TYPE	CHARACTER_SET_CLIENT	COLLATION_CONNECTION
def	항빛무역	view_사원	select `항빛무역`.`사원`.`이름` AS `이름`,...	NONE	YES	root@localhost	DEFINER	utf8mb4	utf8mb4_0900_ai_ci

```
SHOW CREATE VIEW view_사원;
```

### ▶ 실행결과

View	Create View	character_set_client	collation_connection
view_사원	CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY DEFINER VIEW `view_사원` (`이름`, `전화번호`, ...	utf8mb4	utf8mb4_0900_ai_ci

## 5. 뷰 삭제하기

### ■ 뷰 삭제하기

- 형식

```
DROP VIEW 뷰명;
```

### ■ [예제 9-7] 'view\_사원' 뷰를 삭제하시오.

```
DROP VIEW view_사원;
```



## 5. 뷰 삭제하기

### 확인문제

뷰를 생성할 때 반드시 필요한 키워드를 모두 선택하시오.

CREATE, VIEW, SELECT, JOIN, AS, FROM, WHERE, WITH, CHECK, OPTION

### 정답

CREATE, VIEW, AS, SELECT, FROM

# Section 02

**뷰를 통한 데이터 삽입**

# 1. 데이터 삽입 조건

## ■ 데이터 삽입 조건

- 삽입할 대상의 컬럼 명시
- NOT NULL 제약조건
- PRIMARY KEY 제약조건
- CHECK 제약조건
- WITH CHECK OPTION 조건
- 함수, 수식, JOIN, GROUP BY, UNION 등 사용 여부

# 1. 데이터 삽입 조건

- [예제 9-8] [예제 9-3]에서 작성한 'view\_사원\_여' 뷰를 사용하여 다음 레코드를 삽입하시오. 오류가 발생한다면 이유를 생각해보시오
  - 이름: 황여름, 전화번호: (02)587-4989, 입사일: 2023-02-10, 주소: 서울시 강남구 청담동 23-5, 성별: 여

```
INSERT INTO view_사원_여(이름, 전화번호, 입사일, 주소, 성별)
VALUES('황여름','(02)587-4989','2023-02-10','서울시 강남구 청담동 23-5','여');
```

```
Error Code: 1423. Field of view '한빛무역.view_사원_여' underlying table doesn't have a
default value
```

# 1. 데이터 삽입 조건

- [예제 9-8] [예제 9-3]에서 작성한 'view\_사원\_여' 뷰를 사용하여 다음 레코드를 삽입하시오. 오류가 발생한다면 이유를 생각해보시오
  - 오류 해결1

```
CREATE OR REPLACE VIEW view_사원_여
AS
SELECT 사원번호
      ,이름
      ,집전화 AS 전화번호
      ,입사일
      ,주소
      ,성별
FROM 사원
WHERE 성별 = '여';
```

## 1. 데이터 삽입 조건

- [예제 9-8] [예제 9-3]에서 작성한 'view\_사원\_여' 뷰를 사용하여 다음 레코드를 삽입하시오. 오류가 발생한다면 이유를 생각해보시오
  - 오류 해결2

```
ALTER VIEW view_사원_여
AS
SELECT 사원번호
      ,이름
      ,집전화 AS 전화번호
      ,입사일
      ,주소
      ,성별
FROM 사원
WHERE 성별 = '여';
```

# 1. 데이터 삽입 조건

- [예제 9-8] [예제 9-3]에서 작성한 'view\_사원\_여' 뷰를 사용하여 다음 레코드를 삽입하시오. 오류가 발생한다면 이유를 생각해보시오

- 사원번호: E12, 이름: 황여름, 전화번호: (02)587-4989, 입사일: 2023-02-10, 주소: 서울시 강남구 청담동 23-5, 성별: 여

```
INSERT INTO view_사원_여(사원번호, 이름, 전화번호, 입사일, 주소, 성별)
VALUES('E12', '황여름', '(02)587-4989', '2023-02-10', '서울시 강남구 청담동 23-5', '여');
```

- 뷰를 통한 레코드 삽입 확인

```
SELECT *
FROM view_사원_여
WHERE 사원번호 = 'E12';
```

## ▶ 실행결과

사원번호	이름	전화번호	입사일	주소	성별
E12	황여름	(02)587-4989	2023-02-10	서울시 강남구 청담동 23-5	여

# 1. 데이터 삽입 조건

- [예제 9-8] [예제 9-3]에서 작성한 'view\_사원\_여' 뷰를 사용하여 다음 레코드를 삽입하시오. 오류가 발생한다면 이유를 생각해보시오
  - 뷰를 사용하여 레코드를 추가하였지만 실제로는 사원 테이블에 레코드가 삽입됨

```
SELECT *  
FROM 사원  
WHERE 사원번호 = 'E12';
```

## ▶ 실행결과

사원번호	이름	영문이름	직위	성별	생일	입사일	주소	도시	지역	집전화	상사번호	부서번호
E12	황여름	NULL	NULL	여	NULL	2023-02-10	서울시 강남구 청담동 23-5	NULL	NULL	(02)587-4989	NULL	NULL



# 1. 데이터 삽입 조건

- [예제 9-9] [예제 9-2]에서 작성한 'view\_제품별주문수량합' 뷰를 사용하여 레코드를 삽입하시오. 오류가 난다면 이유를 생각해보시오
  - 제품명: 단짠 새우깡, 주문수량합: 250

```
INSERT INTO view_제품별주문수량합  
VALUES('단짠 새우깡', 250);
```

Error Code: 1471. The target table view\_제품별주문수량합 of the INSERT is not insertable-into

## 2. WITH CHECK OPTION

### ■ WITH CHECK OPTION

- 뷰에서 사용하는 옵션 중 하나
- 특정 뷰에 대해 INSERT 또는 UPDATE 작업을 수행할 때 해당 작업으로 인해 뷰에서 정의한 조건을 만족하지 않는 행이 생성되거나 수정되는 것을 방지함
- WITH CHECK OPTION을 사용하면 뷰를 통해 데이터를 수정하거나 삽입할 때 뷰의 조건을 만족하는 데이터만 작업이 가능함
- 즉, 뷰에서 사용된 조건식을 기반으로 데이터의 일관성을 보장하기 위한 제약조건
- 형식

```
CREATE [OR REPLACE] VIEW 뷰명  
AS  
SELECT문  
WITH CHECK OPTION;
```

## 2. WITH CHECK OPTION

- [예제 9-10] 'view\_사원\_여' 뷰를 사용하여 '남' 사원 정보를 추가하고, 결과를 확인하시오.

- 사원번호: E13, 이름: 강겨울, 입사일: 2023-02-10, 주소: 서울시 성북구 장위동 123-7, 성별: 남

```
INSERT INTO view_사원_여(사원번호, 이름, 입사일, 주소, 성별)
VALUES('E13','강겨울','2023-02-10','서울시 성북구 장위동 123-7','남');
```

- 삽입된 레코드를 뷰로 확인하기

```
SELECT *
FROM view_사원_여
WHERE 사원번호 = 'E13';
```

### ▶ 실행결과

사원번호	이름	전화번호	입사일	주소	성별

- ✓ 삽입은 정상적으로 이루어졌으나 검색되지 않음

## 2. WITH CHECK OPTION

- [예제 9-10] 'view\_사원\_여' 뷰를 사용하여 '남' 사원 정보를 추가하고, 결과를 확인하시오.
  - 테이블에서 레코드를 확인하기

```
SELECT *  
FROM 사원  
WHERE 사원번호 = 'E13';
```

### ▶ 실행결과

사원번호	이름	영문이름	직위	성별	생일	입사일	주소	도시	지역	집전화	상사번호	부서번호
E13	강겨울	NULL	NULL	남	NULL	2023-02-10	서울시 성북구 장위동 123-7	NULL	NULL	NULL	NULL	NULL

✓ 테이블에는 존재함

## 2. WITH CHECK OPTION

- [예제 9-10] 'view\_사원\_여' 뷰를 사용하여 '남' 사원 정보를 추가하고, 결과를 확인하시오.
  - 해결방안

```
CREATE OR REPLACE VIEW view_사원_여
AS
SELECT 사원번호
      ,이름
      ,집전화 AS 전화번호
      ,입사일
      ,주소
      ,성별
FROM 사원
WHERE 성별 = '여'
WITH CHECK OPTION;
```

## 2. WITH CHECK OPTION

- [예제 9-10] 'view\_사원\_여' 뷰를 사용하여 '남' 사원 정보를 추가하고, 결과를 확인하시오.

- 새로운 '남' 사원을 추가해보기

✓ 사원번호: E14, 이름: 유봄, 성별: 남

```
INSERT INTO view_사원_여(사원번호, 이름, 성별)
VALUES('E14','유봄','남');
```

Error Code: 1369. CHECK OPTION failed '한빛무역.view\_사원\_여'

✓ 이름: 황여름, 성별: 남

```
UPDATE view_사원_여
SET 성별 = '남'
WHERE 이름 = '황여름';
```

Error Code: 1369. CHECK OPTION failed '한빛무역.view\_사원\_여'

## 2. WITH CHECK OPTION

### 확인문제

‘view\_고객’ 뷰를 통해 ‘서울특별시’ 고객의 정보에 대해서만 삽입하고자 한다. 빈칸을 채워 문장을 완성하시오.

```
CREATE OR REPLACE VIEW view_고객
AS
SELECT 고객번호
      ,고객회사명
      ,도시
FROM 고객
WHERE 도시 = '서울특별시'
 ;
```

### 정답

WITH CHECK OPTION

# Section 03

인덱스



# 1. 인덱스의 개념

## ■ 인덱스(Index)

- 데이터베이스 테이블에서 특정 컬럼이나 컬럼의 집합에 대한 검색 성능을 향상시키기 위해 사용되는 자료구조
- 데이터베이스의 성능 향상을 위해 중요한 역할을 수행함

## ■ 인덱스의 장점

- 검색 속도 향상
- 정렬 및 순서 유지
- 중복 제거
- 조인 성능 향상
- 쿼리 성능 향상

## 2. 인덱스 종류와 주의사항

### ■ 기본 인덱스(Primary Index)

- 클러스터형 인덱스(Clustered Index)라고도 함
- 테이블의 기본키 컬럼에 대해 행 데이터가 오름차순으로 정렬되는 인덱스
- 기본 인덱스는 테이블당 한 개만 생성할 수 있음
- 기본 인덱스는 사전과 유사한 방식으로 저장됨



그림 9-2 기본 인덱스와 유사한 사전

## 2. 인덱스 종류와 주의사항

### ■ 보조 인덱스(Secondary Index)

- 기본 인덱스 이외 인덱스로, 비클러스터형 인덱스(Non-Clustered Index)라고도 함
- 보조 인덱스는 책의 찾아보기 페이지와 유사함



찾아보기	
<b>1</b>	<b>E</b>
객체 262, 267, 308, 316	다중 상속 336, 338
객체 구현 328	다형성 24
객체명 매서드명() 279	단항 연산자 129
객체명 속성명 278	대·소문자 83, 166
객체 변수 308	대입 연산자 67, 73, 117, 126, 129
객체의 속성 278	더하기(+) 108, 113
객체지향 24	데스크톱 응용 프로그램 24
객체지향 프로그래밍 406	데이터형 80, 142
객체지향 프로그래밍 언어 22	동(EAST) 485

그림 9-3 보조 인덱스와 유사한 색인

## 2. 인덱스 종류와 주의사항

- 인덱스 생성 시 주의사항
  - 쿼리 패턴 분석
  - 테이블 크기와 메모리 제한
  - 업데이트 비용 주의
  - 복합 인덱스
  - 중복 및 NULL 값

### 3. MySQL 인덱스의 자료구조

#### ■ B-트리 인덱스

- 가장 일반적으로 사용되는 인덱스 형식으로 좌우 자식 간에 항상 균형을 이루는 균형 트리(Balanced Tree)임
- B-트리에서 노드(Node)는 블록(Block) 또는 페이지(Page)라고 부름
- M차 B-트리
  - ✓ 최대 M개의 자식 노드를 가질 수 있는 B-트리
- 각 노드의 키들은 좌우로 다른 노드를 가리키는 포인터를 가지고 있음
  - ✓ 좌측 포인터는 키보다 작은 데이터를 가진 노드를 가리키고,  
우측 포인터는 키보다 큰 데이터를 가진 노드를 가리킴
- B-트리 인덱스는 테이블 데이터를 정렬하여 B-트리 형태로 구성하며, 각 노드는 정렬된 값의 범위를 나타냄
- 이러한 구조로 인해 효율적인 범위 검색과 정렬이 가능함

### 3. MySQL 인덱스의 자료구조

#### ■ B-트리 인덱스

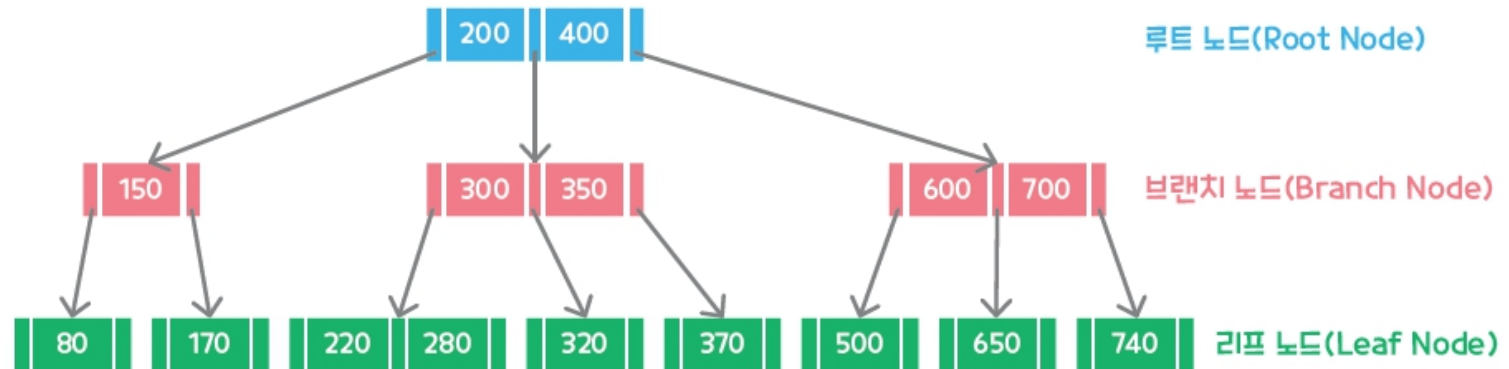


그림 9-4 3차 B-트리의 형태

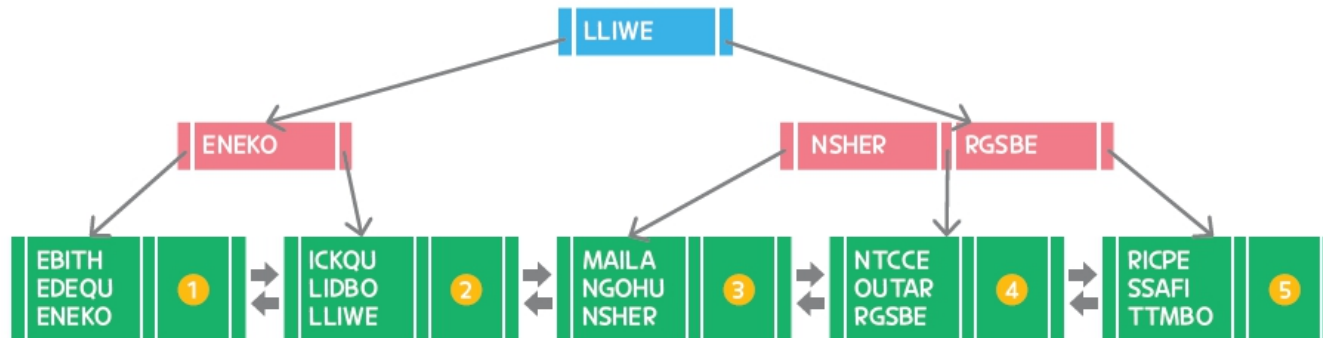
### 3. MySQL 인덱스의 자료구조

#### ■ MySQL의 B-트리 인덱스

- 데이터는 루트 노드부터 탐색이 시작되고, 정렬된 인덱스키를 따라서 리프 노드까지 도달함
- 기본 인덱스는 리프 노드에 키가 포함된 행의 모든 데이터를 포함하므로 추가적인 데이터 조회가 필요하지 않음
- 반면 보조 인덱스는 먼저 인덱스키와 쌍이 되는 기본키 값을 찾은 후 기본 인덱스에서 검색함

### 3. MySQL 인덱스의 자료구조

#### ■ MySQL의 B-트리 인덱스



	고객번호	고객회사명	담당자명	담당자직위	주소	마일리지
1	EBITH	그린로더스	홍소현	마케팅 과장	남동구 간석동 264-11	81479
	EDEQU	에이원카프	박은빈	회계 과장	중구 신흥동 75-3	139
	ENeko	염앤에스푸드	윤석영	영업 사원	서구가좌 1동 7	1258
2	ICKQU	인터비스	정대광	회계 과장	서구 연희동 400-16	857
	LIDBO	진흥 상사	송아린	대표 이사	부산진구 당감 3동 611-3	2819
	LLIWE	태산무역	박소영	영업 과장	부산진구 양정 1동 462-77	806
3	MAILA	그린월드푸드	이소연	영업 과장	중구 자양동 34-16	428
	NGOHU	케이티에스씨	오유진	영업 사원	중구 범일 3동 14-1	7795
	NSHER	숨인티내셔널	정다경	영업 과장	중구 도원동 507-16	120
4	NTCCE	코스리아 무역	유석희	마케팅 과장	남동구 간석 3동 270-8	112
	OUTAR	엘에이플러스	문인수	영업 사원	남구 연수동 208-16	6851
	RGSBE	탑루드코리아	박건희	영업 과장	서구 도마동 110-6	57746
5	RICPE	뉴가든상사	김지호	영업 사원	금정구 청룡동 168-14	1177
	SSAFI	대동 무역	정수환	회계 과장	중구 마산동 250-1	975
	TTMB0	컬러월드푸드	안수인	회계 과장	사하구 신평동 701-29	7329

그림 9-5 고객번호를 기본키로 한 기본 인덱스의 형태



### 3. MySQL 인덱스의 자료구조

#### ■ MySQL의 B-트리 인덱스

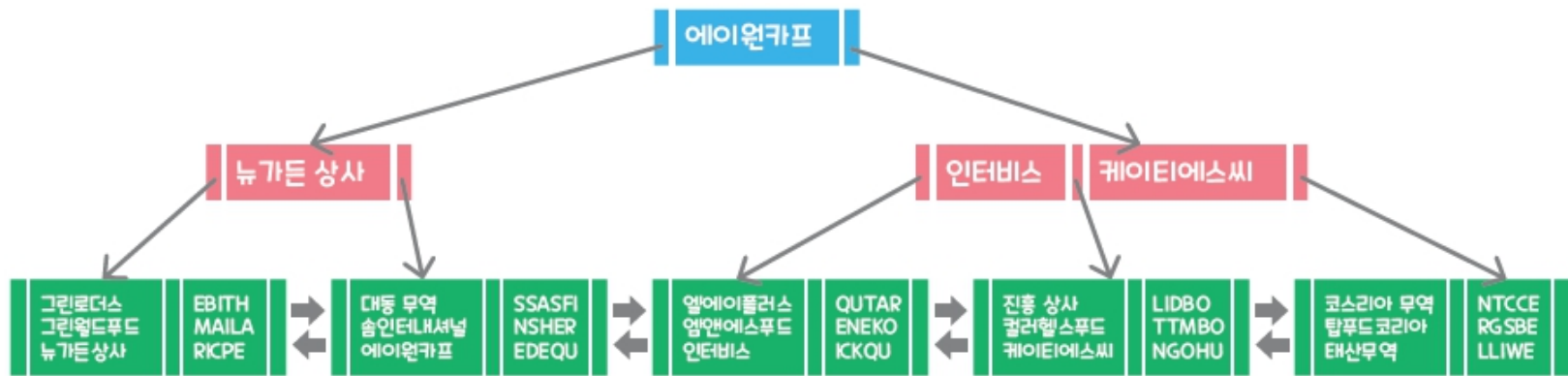


그림 9-6 고객회사명을 인덱스키로 한 보조 인덱스의 형태

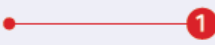
## 4. 인덱스 생성과 사용

### ■ 인덱스 생성하기

- 형식1

```
CREATE [UNIQUE] INDEX 인덱스명 ON 테이블명(컬럼명);
```

- 형식2: 테이블을 생성하면서 인덱스를 함께 생성

```
CREATE TABLE 테이블명  
(  
    컬럼1 데이터타입  
    , 컬럼2 데이터타입  
    , 컬럼3 데이터타입  
    , 컬럼4 데이터타입  
    , PRIMARY KEY(컬럼1, 컬럼2, 컬럼3)  
    , INDEX 인덱스명 (컬럼4)   
);
```

## 4. 인덱스 생성과 사용

### ■ 인덱스 생성하기

- 테이블에 인덱스를 추가하는 형식

```
ALTER TABLE 테이블명 ADD INDEX 인덱스명 (컬럼명);
```

- 테이블에서 인덱스를 삭제하는 형식

```
ALTER TABLE 테이블명 DROP INDEX 인덱스명 (컬럼명);
```

- 테이블에 걸려있는 인덱스를 확인하는 형식

```
SHOW INDEX FROM 테이블명;
```

## 4. 인덱스 생성과 사용

### ■ 인덱스 사용 시 고려 사항

- 인덱스가 다중 컬럼으로 구성되어 있다면 옵티마이저는 인덱스의 가장 왼쪽 접두사(Leftmost Prefix)를 사용해서 컬럼을 찾게 됨
- LIKE 비교에서 인수가 상수 문자열인 경우에는 인덱스를 사용할 수 있으나, '%'와 같은 와일드카드 문자로 시작하는 경우에는 인덱스를 사용할 수 없음
- 인덱스의 가장 왼쪽 접두사가 반드시 모든 AND 그룹에서 사용되어야 인덱스를 사용할 수 있음

## 4. 인덱스 생성과 사용

### ■ [예제 9-11] 날씨 테이블과 인덱스를 생성하시오.

- 컬럼: 년도 INT, 월 INT, 일 INT, 도시 VARCHAR(20), 기온 NUMERIC(3,1), 습도 INT
- 기본키: 년도 + 월 + 일 + 도시
- 보조 인덱스: 기온, 도시

```
CREATE TABLE 날씨
(
  년도 INT
,월 INT
,일 INT
,도시 VARCHAR(20)
,기온 NUMERIC(3,1)
,습도 INT
,PRIMARY KEY(년도, 월, 일, 도시)
,INDEX 기온인덱스(기온)
,INDEX 도시인덱스(도시)
);
```

## 4. 인덱스 생성과 사용

### 확인문제

[예제 9-11]의 날씨 테이블에서 데이터를 검색할 때 인덱스를 사용한다면 ○, 아니면 ×를 표시하십시오.

번호	조건	INDEX 사용 여부
①	WHERE 년도 = 2023 AND 월 = 6;	
②	WHERE 년도 = 2023 OR 월 = 6;	
③	WHERE 년도 = 2023 AND 일 > 1;	
④	WHERE 기온 BETWEEN 10 AND 20;	
⑤	WHERE 월 = 6 AND 일 > 1;	
⑥	WHERE 기온 >= 10 AND 습도 >= 20;	
⑦	WHERE 기온 >= 10 OR 습도 >= 20;	
⑧	WHERE 도시 = '서울';	
⑨	WHERE 도시 LIKE '서울%';	
⑩	WHERE 도시 LIKE '%서울%';	
⑪	WHERE 년도 = 2023 AND 월 = 6 AND 일 > 1 AND 도시 = '서울';	
⑫	WHERE 년도 = 2023 AND 월 = 6 AND 일 > 1 AND 도시 LIKE '서울%';	

### 정답

① ○, ② ×, ③ ○, ④ ○, ⑤ ×, ⑥ ○, ⑦ ×, ⑧ ○, ⑨ ○, ⑩ ×, ⑪ ○, ⑫ ○

# Section 04

옵티마이저

# 1. 옵티마이저의 개념

## ■ 옵티마이저의 개념

- MySQL의 옵티마이저는 데이터베이스에서 가장 효율적인 방식으로 쿼리를 처리할 수 있도록 SQL 쿼리의 실행 계획을 결정하는 역할을 함
- 옵티마이저는 테이블 크기, 인덱스 품질, 통계 정보, 시스템 리소스 사용량 등의 다양한 정보와 통계를 기반으로 쿼리 실행 계획을 결정함



## 2. EXPLAIN과 EXPLAIN ANALYZE

### ■ EXPLAIN

- 쿼리의 실행 계획을 확인하기 위해 사용됨
- 쿼리를 실행하기 전에 실행 계획을 추출하여 반환하는데, 반환된 실행 계획은 테이블 스캔, 인덱스 스캔, 조인 순서, 필터링 방법 등과 같이 어떻게 쿼리를 실행할 것인지에 대한 정보를 제공함
- EXPLAIN을 사용하여 쿼리의 실행 계획을 확인하면 비효율적인 작업을 찾아내어 개선할 수 있으며, 쿼리의 성능 이슈를 파악하고 최적화할 수 있음
- 형식

```
EXPLAIN [FORMAT = TREE | JSON]  
SELECT문;
```

## 2. EXPLAIN과 EXPLAIN ANALYZE

- [예제 9-12] 주문건수가 많은 고객 순으로 고객회사명별 주문건수를 보이는 쿼리의 실행 계획을 확인하시오

```
EXPLAIN FORMAT = TREE
```

```
SELECT 고객회사명  
      ,COUNT(*) AS 주문건수  
FROM 고객  
INNER JOIN 주문  
ON 고객.고객번호 = 주문.고객번호  
GROUP BY 고객회사명  
ORDER BY COUNT(*) DESC;
```

### ▶ 실행결과

```
-> Sort: `주문건수` DESC  
-> Table scan on <temporary>  
-> Aggregate using temporary table  
-> Nested loop inner join (cost=342.40 rows=752)  
-> Filter: (`주문`.`고객번호` is not null) (cost=79.20 rows=752)  
-> Table scan on 주문 (cost=79.20 rows=752)  
-> Single-row index lookup on 고객 using PRIMARY (고객번호=`주문`.`고객번호`) (cost=0.25 rows=1)
```

## 2. EXPLAIN과 EXPLAIN ANALYZE

### ■ EXPLAIN ANALYZE

- EXPLAIN ANALYZE를 사용하면 실행 계획 및 실행한 쿼리에 대한 통계를 함께 확인할 수 있음
- 형식

```
EXPLAIN ANALYZE  
SELECT문;
```

## 2. EXPLAIN과 EXPLAIN ANALYZE

- [예제 9-13] 주문건수가 많은 고객 순으로 고객회사명별로 주문건수를 보이는 쿼리에 대해 실행 계획 및 실행 결과에 대한 통계를 확인하시오

### EXPLAIN ANALYZE

```
SELECT 고객회사명
      ,COUNT(*) AS 주문건수
FROM 고객
INNER JOIN 주문
ON 고객.고객번호 = 주문.고객번호
GROUP BY 고객회사명
ORDER BY COUNT(*) DESC;
```

### ▶ 실행결과

```
-> Sort: `주문건수` DESC (actual time=12.545..12.559 rows=88 loops=1)
  -> Table scan on <temporary> (actual time=0.002..0.021 rows=88 loops=1)
    -> Aggregate using temporary table (actual time=12.404..12.452 rows=88 loops=1)
      -> Nested loop inner join (cost=342.40 rows=752) (actual time=3.925..9.286 rows=751 loops=1)
        -> Filter: (`주문`.`고객번호` is not null) (cost=79.20 rows=752) (actual time=3.770..5.411 rows=751 loops=1)
          -> Table scan on 주문 (cost=79.20 rows=752) (actual time=3.767..5.255 rows=751 loops=1)
            -> Single-row index lookup on 고객 using PRIMARY (고객번호=`주문`.`고객번호`) (cost=0.25 rows=1) (actual time=0.005..0.005 rows=1 loops=751)
```

# 점검문제

## 문제 1

다음 실행결과와 같이 피벗 형식으로 결과가 보이도록 뷰를 작성하시오.

### ▶ 실행결과

도시	대표 이사	영업	마케팅	회계
서울특별시	11	23	10	3
광명시	0	2	0	0
구리시	0	0	1	0
김해시	0	1	0	0
평택시	0	1	0	0
인천광역시	0	4	2	1
대전광역시	0	2	0	1
상주시	0	1	0	0
전주시	0	1	0	0
합계	0	0	0	0