
고급 SQL
1학년 전유경

목차 Contents of List

1. 인덱스 Index

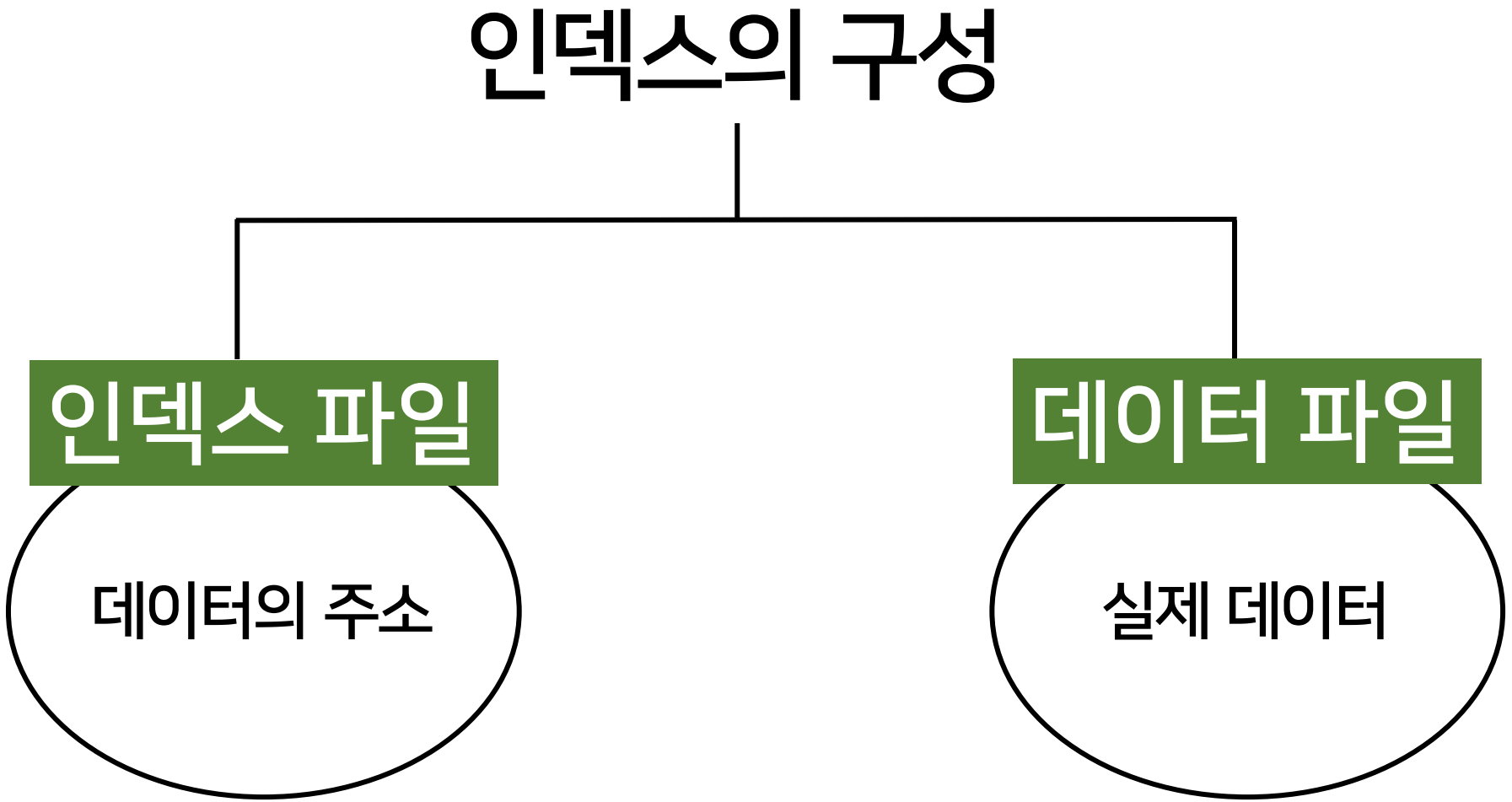
인덱스란?

→ **원하는 자료를 검색**하기 위해 사용하는 방법

왜?

→ **빠르고 효율적**

1. 인덱스 Index

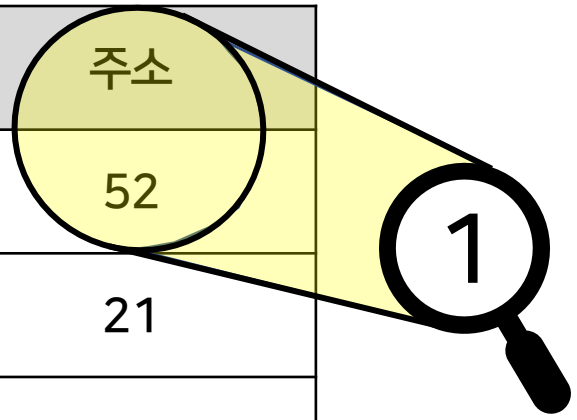


1. 인덱스 Index

데이터 검색하기

인덱스 파일

키 값	주소
Korea	52
Sweden	21
Japan	18
Canada	64
German	34



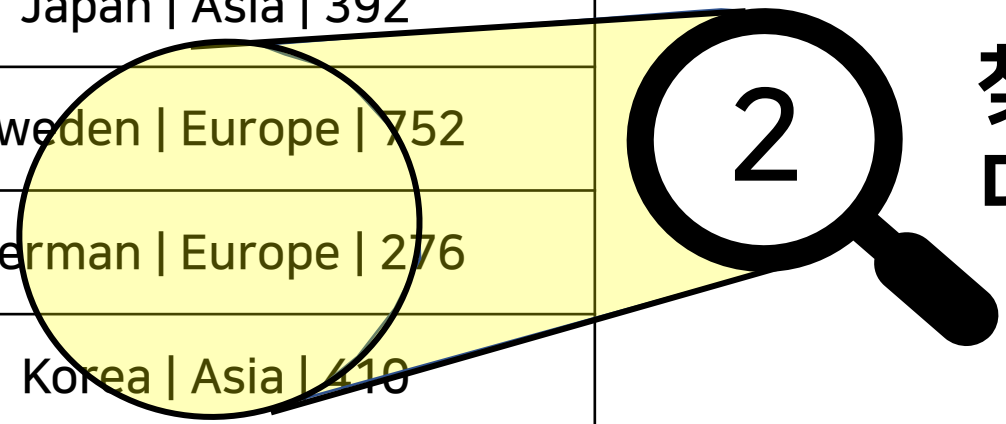
인덱스 파일에서 **데이터 주소** 찾기

1. 인덱스 Index

데이터 검색하기

데이터 파일

주소	데이터
18	Japan | Asia | 392
21	Sweden | Europe | 752
34	German | Europe | 276
52	Korea | Asia | 410
64	Canada | North America | 124



찾은 주소의 **데이터**를
데이터 파일에서 **검색**

1. 인덱스 Index

장점

- 데이터 검색 속도 향상
- 시스템 부하 감소
- 시스템 전체의 성능 향상

단점

- 추가 DB 공간의 필요
- 인덱스 생성 시간 소요
- 잦은 변경 작업으로 인한 성능 저하

1. 인덱스 Index

- DDL문을 통해 '생성/제거/수정' 조작이 가능
- 시스템에 의해 자동 관리됨

인덱스 선정 기준

카디널리티(Cardinality)가 높은 것

전체 행에 대한 특정 컬럼의 중복 수치

중복도가 ↑, 카디널리티 ↓

중복도가 ↓, 카디널리티 ↑

인덱스 생성

```
CREATE [UNIQUE] INDEX 인덱스_이름
ON 테이블_이름(속성_이름 [ASC|DESC])
[CLUSTER];
```

UNIQUE : 중복 설정

속성_이름 : 인덱스로 사용할 속성 값

ASC | DESC : 속성값을 오름/내림차순으로 정렬

CLUSTER : 인접된 튜플들을 물리적 그룹으로 묶어 저장

1. 인덱스 Index

```
mysql> CREATE UNIQUE INDEX stud_idx ON 학생(학번 ASC);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> SHOW INDEX FROM 학생;
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| 학생  |          0 | stud_idx |            1 | 학번        | A         |           5 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
1 row in set (0.00 sec)
```

인덱스 제거

ALTER TABLE 테이블_이름 DROP INDEX 인덱스_이름;

```
mysql> ALTER TABLE 학생
    -> DROP INDEX stud_idx;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> show index from 학생;
Empty set (0.00 sec)
```

뷰(View)란?

→ 하나 이상의 테이블로부터 유도되어 만들어진
가상 테이블 혹은 **임시 테이블**

→ 물리적으로 기억공간 차지 X
논리적 독립성 제공
보안성 향상

2. 뷰 View

뷰의 생성

CREATE VIEW 뷰_이름[(뷰_속성이름)]
AS SELECT 기본테이블 속성_이름
FROM 기본테이블_이름
[WHERE 조건]
[WITH CHECK OPTION]

↓ 기존 테이블

```
mysql> SELECT * FROM 학생;
+----------+--------+--------------+------+----------+
| 학번     | 성명   | 학과         | 학년 | 연락처   |
+----------+--------+--------------+------+----------+
| 91987654 | 고양이 | 실용음악과   |    4 | 6262-8585 |
| 92034567 | 홍길동 | 유아교육과   |    3 | 9933-7722 |
| 92109876 | 토마토 | 국제통상학과 |    2 | 1122-3344 |
| 92212345 | 아무개 | 경제학과     |    1 | 1234-5678 |
| 92213067 | 전유경 | 정보보호학과 |    1 | 8752-6207 |
+----------+--------+--------------+------+----------+
5 rows in set (0.00 sec)
```

```
mysql> CREATE VIEW 1학년연락처(학번, 이름, 연락처)
    -> AS SELECT 학번, 성명, 연락처
    -> FROM 학생
    -> WHERE 학년 = 1;
Query OK, 0 rows affected (0.01 sec)
```

→

```
mysql> SELECT * FROM 1학년연락처;
+----------+--------+----------+
| 학번     | 이름   | 연락처   |
+----------+--------+----------+
| 92212345 | 아무개 | 1234-5678 |
| 92213067 | 전유경 | 8752-6207 |
+----------+--------+----------+
2 rows in set (0.01 sec)
```

뷰의 삭제

```
DROP VIEW 뷰_이름 [RESTRICT | CASCADE];
```

RESTRICT : 삭제할 요소가 참조 중이면 삭제 하지 않음

CASCADE : 삭제할 요소가 참조 중이더라도 삭제, 연관된 요소들도 일괄적으로 삭제

```
mysql> DROP VIEW 1학년연락처;
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> SELECT * FROM 1학년연락처;
ERROR 1146 (42S02): Table 'db1.1학년연락처' doesn't exist
```

뷰의 특징

뷰가 정의된 기본 테이블이 제거 -> 뷰도 자동적으로 제거

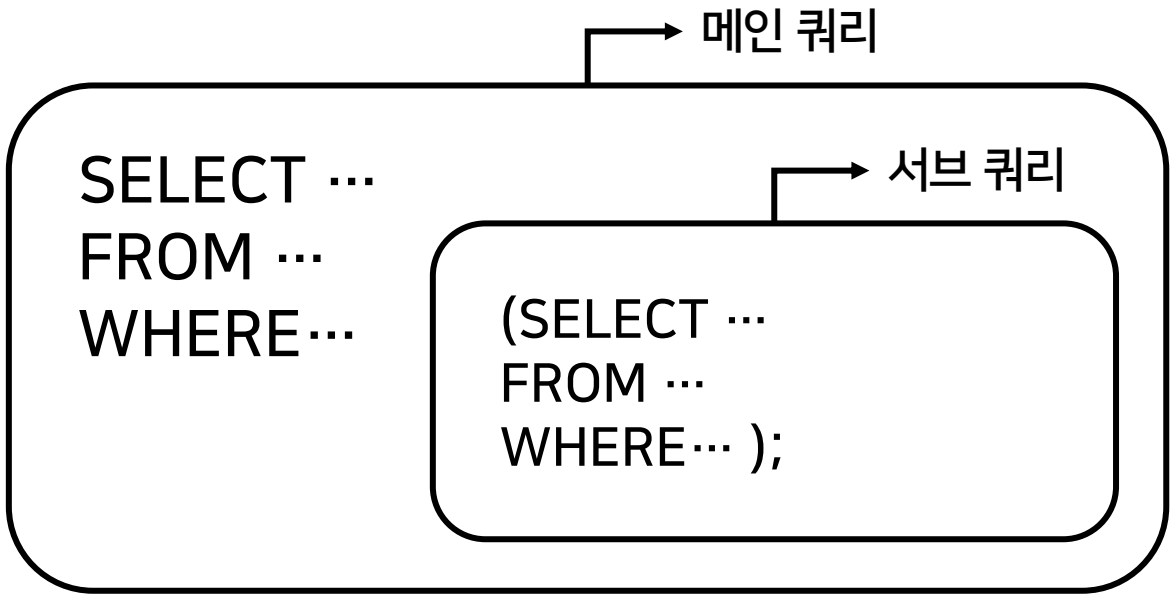
한 번 정의된 뷰는 변경 불가능하며 삭제한 후 다시 생성해야 함

뷰의 속성 중 기본 테이블의 기본키가 포함되어 있지 않으면 삽입, 삭제, 갱신 X

3. 서브 쿼리

서브 쿼리 = 부속 질의

하나의 SELECT문 안에 또 하나의 SELECT문이 포함되어 있는 것



3. 서브 쿼리

서브 쿼리의 결과 값은 메인 쿼리로 반환되어 사용

서브 쿼리가 반환하는 행의 수

1개 이하 → 단일 행 서브 쿼리

여러 개 → 다중 행 서브 쿼리

3. 서브 쿼리

단일 행 서브 쿼리

```
mysql> SELECT * FROM 성적
    -> WHERE 학과 = ( SELECT 학과 FROM 성적 WHERE 성명 = '전유경');
+----------+--------+--------------+------+------+
| 학번     | 성명   | 학과         | 학년 | 점수 |
+----------+--------+--------------+------+------+
| 92213067 | 전유경 | 정보보호학과 |    1 |   85 |
+----------+--------+--------------+------+------+
1 row in set (0.00 sec)
```

```
mysql> SELECT * FROM 성적
    -> WHERE 점수 < ( SELECT 점수 FROM 성적 WHERE 학년 = 4);
+----------+--------+--------------+------+------+
| 학번     | 성명   | 학과         | 학년 | 점수 |
+----------+--------+--------------+------+------+
| 92034567 | 홍길동 | 유아교육과   |    3 |   75 |
| 92109876 | 토마토 | 국제통상학과 |    2 |   50 |
| 92212345 | 아무개 | 경제학과     |    1 |   80 |
| 92213067 | 전유경 | 정보보호학과 |    1 |   85 |
+----------+--------+--------------+------+------+
4 rows in set (0.00 sec)
```

↓ 기존 테이블

```
mysql> SELECT * FROM 성적;
+----------+--------+--------------+------+------+
| 학번     | 성명   | 학과         | 학년 | 점수 |
+----------+--------+--------------+------+------+
| 91987654 | 고양이 | 실용음악과   |    4 |   95 |
| 92034567 | 홍길동 | 유아교육과   |    3 |   75 |
| 92109876 | 토마토 | 국제통상학과 |    2 |   50 |
| 92212345 | 아무개 | 경제학과     |    1 |   80 |
| 92213067 | 전유경 | 정보보호학과 |    1 |   85 |
+----------+--------+--------------+------+------+
5 rows in set (0.00 sec)
```

메인 쿼리의 WHERE 절에서는
단일 행 비교 연산자 사용

=,<>,>,=>,<,<=

다중 행 서브 쿼리

반드시 다중 행 연산자와 함께 사용

다중 행 연산자	설명
IN	메인 쿼리의 비교 조건이 서브 쿼리의 결과 중 하나라도 일치하면 참
ANY, SOME	… 서브 쿼리의 검색 결과와 하나 이상이 일치하면 참
ALL	… 서브 쿼리의 검색 결과와 모든 값이 일치하면 참
EXISTS	… 서브 쿼리의 결과 중에서 만족하는 값이 하나라도 존재하면 참

3. 서브 쿼리

↓ 기존 테이블

```
mysql> SELECT * FROM 성적;
+----------+--------+--------------+------+------+
| 학번     | 성명   | 학과         | 학년 | 점수 |
+----------+--------+--------------+------+------+
| 91987654 | 고양이 | 실용음악과   |    4 |   95 |
| 92034567 | 홍길동 | 유아교육과   |    3 |   75 |
| 92109876 | 토마토 | 국제통상학과 |    2 |   50 |
| 92212345 | 아무개 | 경제학과     |    1 |   80 |
| 92213067 | 전유경 | 정보보호학과 |    1 |   85 |
+----------+--------+--------------+------+------+
5 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM 성적
    -> WHERE 학과 IN ( SELECT 학과 FROM 성적 WHERE 점수 >= 80 );
+----------+--------+--------------+------+------+
| 학번     | 성명   | 학과         | 학년 | 점수 |
+----------+--------+--------------+------+------+
| 91987654 | 고양이 | 실용음악과   |    4 |   95 |
| 92212345 | 아무개 | 경제학과     |    1 |   80 |
| 92213067 | 전유경 | 정보보호학과 |    1 |   85 |
+----------+--------+--------------+------+------+
3 rows in set (0.00 sec)
```

4. 느낀 점

1.인덱스 구조인 B+- 트리, 클러스터드, 넌 클러스터드 알아보기

2.조인과 집합 연산자 공부

3.다음에는 더 꼼꼼히 공부하고 준비하기

감사합니다