Chapter 6. 데이터베이스 객체 작성과 삭제

25강. 데이터베이스 객체

1) 데이터베이스 객체

테이블, 뷰, 인덱스 등 데이터베이스 내에 정의하는 모든 것을 일컫는 말

- 객체: 데이터베이스 내에 실체를 갖는 어떤 것
- 객체의 종류에 따라 데이터베이스에 저장되는 내용도 달라짐
- 객체는 이름을 갖는데, 이때 중복되지 않도록 함
- 이름 제약 사항
    - 기존 이름이나 예약어와 중복하지 않는다
    - 숫자로 시작할 수 없다
    - 언더스코어(_) 이외의 기호는 사용할 수 없다
    - 한글을 사용할 때 더블쿼트(MySQL에서는 백쿼트)로 둘러싼다
    - 시스템이 허용하는 길이를 초과하지 않는다
- 의미 없는 이름을 붙이지 않도록 한다
- 이름은 객체의 종류와는 관계없다

2) 스키마

스키마 안에 데이터베이스 객체가 만들어진다

⇒ 객체의 이름이 같아도 스키마가 다르면 상관없음

- 데이터베이스 객테는 '스키마 객체'라고 부르기도 함
- '스키마 설계' : 데이테베이스에 테이블을 작성하고 구축하는 작업
- DDL을 이용하여 스키마 정의
- 데이터베이스 제품에 따라 스키마가 되는 것이 다름

    - MySQL에서는 CREATE DATABASE로 작성한 데이터베이스
    - Oracle 등에서는 데이터베이스와 데이터베이스 사용자가 계층적 스키마가 됨
- '네임스페이스': 이름이 충돌하지 않도록 가능한 공간(스키마나 테이블)

26강. 테이블 작성 · 삭제 · 변경

1) 테이블 작성

```
CREATE TABLE 테이블명( #테이블 명 뒤에 괄호로 열 정의
    열 정의1, #열을 정의할 때 콤마(,)로 구분하여 연속해서 지정
    열 정의2,
    ...
)
```

- 열명은 명명규칙에 맞게 지정
- 자료형은 INTEGER, VARCHAR 등 지정. 이때 CHAR, VARCHAR은 최대 길이 지정
- 기본값은 DEFAULT로 지정하되 자료형에 맞게 기술, 생략 가능
- 열 NULL 허용 여부 지정 , 생략 시 허용
- 예제 6-1. CREATE TABLE로 테이블 작성하기

```
mysql> CREATE TABLE sample62(
    -> no INTEGER NOT NULL,
    -> a VARCHAR(30),
    -> b DATE);
Query OK, 0 rows affected (0.951 sec)

mysql> DESC sample62;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| no    | int         | NO   |     | NULL    |       |
| a     | varchar(30) | YES  |     | NULL    |       |
| b     | date        | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

2) 테이블 삭제

```
DROP TABLE 테이블명
```

- 삭제 시, 확인을 요구하지 않으니 실수로 삭제하지 않게 주의할 것

데이터 행 삭제

- DROP TABLE : 테이블 삭제 → 테이블의 모든 행도 함께 삭제
- DELECT : 테이블 정의는 그대로, 데이터만 삭제

  *WHERE 조건을 사용하지 않으면 모든 행 삭제
- **TRUNCATE TABLE** 테이블명(DDL): 모든 행 삭제

3) 테이블 변경

```
ALTER TABLE 테이블명 변경명령
```

- 테이블에 있는 데이터는 그대로 둔 채, 구성만 변경 관리 가능
- ALTER TABLE로 할 수 있는 일
  1. 열 추가, 삭제, 변경
  2. 제약 추가, 삭제

열 추가

```
ALTER TABLE 테이블명 ADD 열 정의 /*열 정의는 CREATE와 동일
                                  (열이름, 자료형, 기본값, NULL제약 등)*/
```

- 예제 6-2. ALTER TABLE로 테이블에 열 추가하기

```
mysql> ALTER TABLE sample62 ADD newcol INTEGER;
Query OK, 0 rows affected (1.504 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DESC sample62;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| no     | int         | NO   |     | NULL    |       |
| a      | varchar(30) | YES  |     | NULL    |       |
| b      | date        | YES  |     | NULL    |       |
| newcol | int         | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
```

- 기존 데이터행이 존재하면 추가한 열의 값이 모두 NULL
- NOT NULL 제약이 걸린 열을 추가할 때는 기본값을 지정해야 함

열 속성 변경

```
ALTER TABLE 테이블명 MODIFY 열 정의 /*열 정의는 CREATE와 동일
                                    (자료형, 기본값, NULL제약 등. 열 이름 제외)*/
```

- 예제 6-3. ALTER TABLE로 열 속성 변경하기

```
mysql> ALTER TABLE sample62 MODIFY newcol VARCHAR(20);
Query OK, 0 rows affected (1.277 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DESC sample62;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| no     | int         | NO   |     | NULL    |       |
| a      | varchar(30) | YES  |     | NULL    |       |
| b      | date        | YES  |     | NULL    |       |
| newcol | varchar(20) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
```

- 기존의 데이터 행이 존재하는 경우, 속성 변경에 따라 데이터 변환
- 처리과정에서 에러 발생 시, ALTER TABLE 명령 실행 X
- MODIFY은 MySQL과 Oracle에서 사용하는 하부 명령

열 이름 변경

ALTER TABLE 테이블명 CHANGE [기존 열 이름][신규 열 정의]

- 열이름 뿐만 아니라 열 속성도 변경 가능
- Oracle에서는 RENAME TO 하부명령 사용
- 예제 6-4. ALTER TABLE로 열 이름 변경하기

```
mysql> ALTER TABLE sample62 CHANGE newcol c VARCHAR(20);
Query OK, 0 rows affected (0.394 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DESC sample62;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| no    | int         | NO   |     | NULL    |       |
| a     | varchar(30) | YES  |     | NULL    |       |
| b     | date        | YES  |     | NULL    |       |
| c     | varchar(20) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

  - 교재에서는 제약사항을 안 붙였지만, 현재 syntax에러로 인해 제약사항 덧붙여 실행

열 삭제

ALTER TABLE 테이블명 DROP 열명

- 테이블에 존재하지 않는 열이 지정되면 에러 발생
- 예제 6-5. ALTER TABLE로 열 삭제하기

```
mysql> ALTER TABLE sample62 DROP c;
Query OK, 0 rows affected (1.150 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DESC sample62;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| no    | int         | NO   |     | NULL    |       |
| a     | varchar(30) | YES  |     | NULL    |       |
| b     | date        | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

4) ALTER TABLE로 테이블 관리

최대길이 연장

- 한 자리로 충분했던 용량이 시간이 지나며 부족해지는 케이스

```
ALTER TABLE sample MODIFY col VARCHAR(30)
```

- 반대로 저장공간을 줄이고 싶어 최대길이를 줄일 수도 있음. 그러나, 여러 문제가 존재
  - 기존의 행에 존재하는 데이터의 길이보다 작게 지정 불가능
  - 열의 최대길이를 줄여도 실제 저장공간이 늘어나는 경우는 적음

    (실제로 길이를 늘리는 경우는 많지만 줄이는 경우는 적음)

열 추가

- ADD 하부명령을 사용하여 열 추가 가능
- 테이블 정의가 잘 바뀌기 때문에 변경한 테이블에 행을 추가하는 INSERT명령 확인 필요

27강. 제약

1) 테이블 작성시 제약 정의

- 제약은 TABLE에 설정하는 것
- CREATE TABLE로 테이블을 작성할 때 제약도 같이 설정
  - NOT NULL 제약 등 하나의 열에 대해 설정하는 제약은 열을 정의할 때 함께 정의
- ALTER TABLE로 제약 지정, 변경 또한 가능
- 예제 6-6. 테이블 열에 제약 정의하기

```
mysql> CREATE TABLE sample631 (
    -> a INTEGER NOT NULL,
    -> b INTEGER NOT NULL UNIQUE,
    -> c VARCHAR(30));
Query OK, 0 rows affected (0.665 sec)
```

- **열 제약**: 열에 대해 정의하는 제약
- **테이블 제약**: 한 개의 제약으로 복수의 열에 제약을 설명하는 경우
- 예제 6-7. 테이블에 '테이블 제약' 정의하기

```
mysql> CREATE TABLE sample632(
    -> no INTEGER NOT NULL,
    -> sub_no INTEGER NOT NULL,
    -> name VARCHAR(30),
    -> PRIMARY KEY(no, sub_no));
Query OK, 0 rows affected (0.546 sec)
```

- **CONSTRAINT** 키워드 : 제약 이름 지정(가능하면 이름 지정하자!)
- 예제 6-8. '테이블 제약'에 이름 붙이기

```
CREATE TABLE sample632 (
   no INTEGER NOT NULL,
   sub_no INTEGER NOT NULL,
   name VARCHAR(30),
   CONSTRAINT pkey_sample PRIMARY KEY (no, sub_no)
);
```

2) 제약 추가

기존 테이블에 나중에 제약을 추가할 수 있다

열 제약 추가

- ALTER TABLE로 열 정의 변경
- 예제 6-9. 열 제약 추가하기

```
mysql> ALTER TABLE sample631 MODIFY c VARCHAR(30) NOT NULL;
Query OK, 0 rows affected (1.225 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

  - 기존 테이블을 변경할 경우 제약을 위반하는 데이터가 있는지 확인 필요
  - 만약 c열에 NULL값이 존재한다면 위의 명령은 에러 발생

테이블 제약 추가

- ADD 하부 명령으로 추가 가능
- 기본키는 테이블에 하나만 설정 가능하므로, 이미 기본키가 설정돼있을 경우 테이블에 추가로 기본키 작성 불가능
- 열 제약 추가와 마찬가지로 기존의 행 검사 필요
- 예제 6-10. 테이블 제약 추가하기

```
mysql> ALTER TABLE sample631 ADD CONSTRAINT pkey_sample PRIMARY KEY(a);
Query OK, 0 rows affected (2.319 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DESC sample631;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| a     | int         | NO   | PRI | NULL    |       |
| b     | int         | NO   | UNI | NULL    |       |
| c     | varchar(30) | NO   |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

3) 제약 삭제

- 예제 6-11. 열 제약 삭제하기

```
ALTER TABLE sample631 MODIFY c VARCHAR(30);
```

- 예제 6-12. 테이블 제약 삭제하기

```
ALTER TABLE sample631 DROP CONSTRAINT pkey_sample631;
```

- 예제 6-13. 테이블 제약 삭제하기

```
mysql> ALTER TABLE sample631 DROP PRIMARY KEY;
Query OK, 0 rows affected (1.172 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DESC sample631;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| a     | int         | NO   |     | NULL    |       |
| b     | int         | NO   | PRI | NULL    |       |
| c     | varchar(30) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.037 sec)
```

4) 기본키

- 예제 6-14. sample643 테이블 작성하기

```
mysql> CREATE TABLE sample643(
    -> p INTEGER NOT NULL,
    -> a VARCHAR(30),
    -> CONSTRAINT pkey_sample643 PRIMARY KEY(p));
Query OK, 0 rows affected (0.515 sec)
```

- 기본키로 지정할 열은 NOT NULL 제약 설정 필요
- 기본키는 대량의 키에서 원하는 데이터를 찾아낼 때 키가 되는 요소를 지정하는 키
- 테이블의 행 한 개를 특정할 수 있는 검색키
- 기본키로 설정된 열이 중복된 데이터 값을 가지면 안 됨
- 예제 6-15. sample643에 행 추가하기

```
mysql> INSERT INTO sample643 VALUES(1,'첫째줄');
Query OK, 1 row affected (0.244 sec)

mysql> INSERT INTO sample643 VALUES(2, '둘째줄');
Query OK, 1 row affected (0.150 sec)

mysql> INSERT INTO sample643 VALUES(3,'셋째째줄');
Query OK, 1 row affected (0.147 sec)
```

- 예제 6-16. sample643에 중복되는 행 추가하기

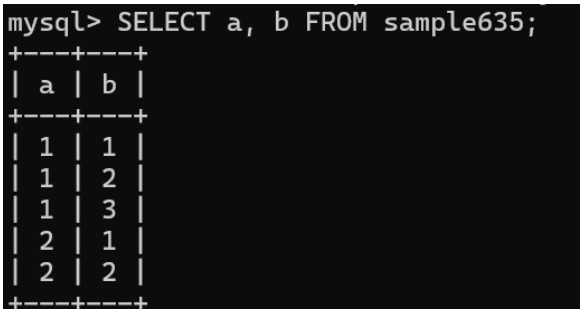
```
mysql> INSERT INTO sample643 VALUES(2,'넷째줄');
ERROR 1062 (23000): Duplicate entry '2' for key 'sample643.PRIMARY'
```

- 예제 6-17. sample643을 중복된 값으로 갱신하기

```
mysql> UPDATE sample643 SET p=2 WHERE p=3;
ERROR 1062 (23000): Duplicate entry '2' for key 'sample643.PRIMARY'
```

복수의 열로 기본키 구성하기

- 예제 6-18. a열과 b열로 이루어진 기본키



- a열과 b열을 기본키로 같이 지정할 경우, 중복되는 값 없음 → 기본키로 설정 가능

28강. 인덱스 구조

1) 인덱스

인덱스는 테이블에 붙여진 **색인**

- 인덱스의 역할: 검색속도 향상
- 테이블과는 별개로 독립된 데이터베이스 객체로 작성됨
- 대부분의 데이터베이스에서 테이블 삭제 시, 인덱스도 같이 삭제

2) 검색에 사용하는 알고리즘

- 인덱스에 쓰이는 대표적인 검색 알고리즘: **이진트리**, 해시

full table scan

- 인덱스가 지정되지 않은 테이블을 검색할 때 사용
- 테이블에 저장된 모든 값을 처음부터 차례대로 조사

binary search

- 차례로 나열된 집합에 대해 유효한 검색 방법
- 집합을 반으로 나누어 조사하는 방법
- 대량의 데이터 검색 시, 풀 테이블 스캔에 비해 이진 탐색이 빠름

binary tree

- 이진 탐색은 데이터가 미리 정렬되어 있어야 가능한 방법
- 테이블 데이터와 인덱스용 데이터가 별도로 저장되는데 이진 트리라는 데이터 구조로 작성
- 트리는 노드로 구성
    - 왼쪽은 작은 값, 오른쪽 가지는 큰 값으로 분기

3) 유일성

- 이진 트리에서는 집합 내의 중복하는 값을 가질 수 없음
- 기본키 제약은 이진 트리로 인덱스를 작성하는 데이터베이스가 많음

29강. 인덱스 작성과 삭제

1) 인덱스 작성

- 데이터베이스 제품에 따라 인덱스의 취급이 다름
    - Oracle, DB2 : 스키마 객체 → 스키마 내 이름 중복 불가
    - SQL Server, MySQL : 테이블 내의 객체 → 테이블 내 이름 중복 불가
- 인덱스를 작성할 때 해당 인덱스가 어느 테이블의 어느 열에 관한 것인지 지정할 필요

```
CREATE INDEX 인덱스명 ON 테이블명(열명1, 열명2, ... )
```

- 예제 6-19. 인덱스 작성하기

```
mysql> CREATE INDEX isample65 ON sample62(no);
Query OK, 0 rows affected (0.706 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

- 인덱스 작성 시 저장장치에 색인용 데이터 생성

2) 인덱스 삭제

```
DROP INDEX 인덱스명 #스키마 객체의 경우
DROP INDEX 인덱스명 ON 테이블명 #테이블 내 객체의 경우
```

- DROP 할 때는 다른 객체와 동일하게 이름만 지정
- 다만 테이블 내 객체로서 작성하는 경우 테이블 이름도 지정
- 인덱스는 테이블에 의존하는 객체. 고로, 테이블 삭제 시 인덱스도 자동 삭제
- 예제 6-20. 인덱스 삭제하기

```
mysql> DROP INDEX isample65 ON sample62;
Query OK, 0 rows affected (0.376 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

- SELECT 명령 시 인덱스를 사용하면 검색 속도 향상
- INSERT 명령 시 인덱스를 최신 상태로 갱신하기 위해 처리속도가 조금 떨어짐

3) EXPLAIN

```
EXPLAIN SQL명령
```

- 인덱스를 사용해 검색하는지 확인 하는 명령
- 뒤의 SQL명령은 실제 실행X, 어떤 상태로 실행되는지 설명만 해줌
- 표준 SQL은 아니지만 어떤 데이터베이스 제품이라도 이와 비슷한 명령 지원
- 예제 6-21. EXPLAIN으로 인덱스 사용 확인하기1 (MySQL)

```
EXPLAIN SELECT * FROM sample62 WHERE a='a';
```

- 예제 6-22. EXPLAIN으로 인덱스 사용 확인하기2 (MySQL)

```
EXPLAIN SELECT * FROM sample62 WHERE no>10;
```

  - a열을 사용하지 않도록 조건을 변경하면 인덱스를 사용할 수 없음

4) 최적화

실행계획

- 내부 처리에서는 SELECT 명령을 실행하기 전 시행됨
- 인덱스의 유무뿐만 아니라 인덱스를 사용할 것인지 여부에 대해서도 데이터베이스 내부의 최적화 처리를 통해 판단(인덱스 품질도 고려)
- '예' 또는 '아니오'값만 갖는 열은 이진트리가 되어 인덱스 효율이 떨어짐
  - 데이터의 종류가 적을수록 인덱스 효율이 떨어짐
  - 이렇게 인덱스의 품질을 고려하여 실행계획이 세워짐

30강. 뷰 작성과 삭제

1) 뷰

- 데이터베이스 객체(SELECT 명령은 객체X)
- 뷰를 정의할 때는 이름과 SELECT 명령을 지정
- 뷰 생성 후 SELECT 명령에서 뷰의 이름을 지정하면 참조할 수 있음
- 자주 사용하거나 복잡한 SELECT 명령을 뷰로 만들면 편리하게 사용 가능

가상 테이블

뷰는 '실체가 존재하지 않는' 테이블

- 테이블처럼 저장공간을 갖지 않음
- 따라서 SELECT 명령만 사용하는 것을 권장(다른 명령은 주의할 필요)

2) 뷰 작성과 삭제

뷰 작성

```
CREATE VIEW 뷰명 AS SELECT 명령
```

- AS 키워드는 생략 불가능

- 예제 6-23. 뷰 작성하기

```
mysql> CREATE VIEW sample_view67 AS SELECT * FROM sample54;
Query OK, 0 rows affected (0.370 sec)

mysql> SELECT * FROM sample_view67;
+------+------+
| no   | a    |
+------+------+
|    1 |  100 |
|    2 |  900 |
|    3 |   20 |
|    4 |   80 |
+------+------+
```

- 뷰는 필요에 따라 열 지정 가능, 이름 뒤에 괄호를 묶어 열을 나열

  CREATE VIEW 뷰명(열명1, 열명2, ...) AS SELECT 명령

- 열 지정 생략 시 자동적으로 뷰의 열 지정
- 열을 지정한 경우 SELECT 명령의 SELECT 구에 지정한 열보다 우선시 됨
- 열 이외 지정 불가, 자료형이나 제약 지정 불가
- 예제 6-24. 열을 지정해 뷰 작성하기

```
mysql> CREATE VIEW sample_view672(n, v, v2) AS
    -> SELECT no, a, a*2 FROM sample54;
Query OK, 0 rows affected (0.218 sec)

mysql> SELECT * FROM sample_view672 WHERE n=1;
+------+------+------+
| n    | v    | v2   |
+------+------+------+
|    1 |  100 |  200 |
+------+------+------+
1 row in set (0.021 sec)
```

3) 뷰의 약점

- 저장공간을 필요로 하지 않지만 CPU 자원을 사용
- 검색 뿐 아니라 정렬, 집계 등이 가능한데 이런 처리는 계산능력을 필요로 하여 CPU자원 사용

Materialized View

- 뷰의 약점: 뷰의 근원인 테이블의 데이터 양이 많으면 뷰의 처리속도도 떨어짐

  ⇒ 이를 회피하기 위한 것이 머티리얼라이즈드 뷰
- 일반 뷰와 차이
  - 일반 뷰: 데이터를 일시적으로 저장하고 쿼리가 실행 종료 시 함께 삭제
  - 머티리얼라이즈드 뷰: 테이블처럼 저장장치에 저장해두고 사용
- 처음 참조 시 데이터 저장, 이후 참조할 땐 이전 데이터 그대로 사용
- 테이블의 데이터가 변경된 경우, SELECT 명령을 재실행하여 데이터 다시 저장
- 변경 유무를 확인하여 재실행 하는 것은 RDBMS가 자동 실행
- MySQL 사용 불가, Oracle과 DB2에서만 사용 가능

함수 테이블

- 뷰의 약점: 부모 쿼리와 연관된 서브쿼리의 경우 뷰의 SELECT 명령 사용 불가

  ⇒ 이를 회피하기 위해 함수 테이블 사용
- 함수 테이블은 테이블을 결괏값으로 반환해주는 사용자 정의 함수
  - 인수의 값에 따라 WHERE 조건을 붙여 결괏값을 바꿀 수 있음

    → 서브쿼리처럼 동작 가능