



SESSION #4

# 데이터베이스 객체 작성과 삭제 Part 1

By EDU Team 2nd (Jaehoon)



# 목차 소개

## Part. 1

Chapter 01 데이터베이스 객체란?

Chapter 02 DDL 구문 (작성 · 삭제 · 변경 · 제약)

- 2.1 DDL구문 (작성)
- 2.2 DDL구문 (삭제)
- 2.3 DDL구문 (변경)
- 2.4 DDL구문 (제약)

## Part. 2

Chapter 03 기본키 · 인덱스

Chapter 04 뷰(View)

# Chapter 1



# Chapter 1.0 데이터베이스 객체란?

## 데이터베이스 객체

테이블이나 뷰, 인덱스 등 데이터베이스 내에 정의하는(define)하는 모든 것을 말한다.

여기서 객체는 데이터베이스 내에 **실체**를 가지는 어떤 것이다.

→ 테이블, 뷰, 인덱스

그럼 실체를 가지지 않는 것은?

→ SQL 명령어



# Chapter 1.0 데이터베이스 객체란?

## 데이터베이스 객체

“친구야 식탁에서 사과를 가져와.”

mysql>> SELECT apple FROM food\_table ;

명령어 → 실체X

열 → 객체 X

테이블 → 객체O

하지만 명명규칙은 유사함

# Chapter 1.0 데이터베이스 객체란?

## ■ 데이터베이스 객체 (명명규칙)

- 1) 기존이름이나 예약어와 중복하지 않는다.
- 2) 숫자로 시작할 수 없다.
- 3) 언더스코어(\_) 이외의 기호는 사용할 수 없다.
- 4) 한글을 사용할 때는 더블쿼트 (MySQL에서는 백쿼트)로 둘러싼다.
- 5) 시스템이 허용하는 길이를 초과하지 않는다.



# Chapter 1.0 데이터베이스 객체란?

## 스키마(Scheme)

데이터베이스의 구조와 제약조건에 관해 전반적인 명세를 기술한 것이다.

개체의 특성을 나타내는 속성과 속성들의 집합으로 이루어진 개체, 개체 사이에 존재하는 관계에 대한 정의와 이들이 유지해야할 제약조건들을 기술한 것이다.

즉, 데이터베이스 내에서 어떤 구조로 데이터에 저장되어야 하는지를 나타내는 데이터베이스 구조를 스키마라고 한다.

# 객체의 이름이 같아도 스키마가 서로 다르다면 상관이 없다.

# 데이터베이스에 테이블을 작성해서 구축해 나가는 작업을 '스키마 설계'라고 한다.

# 어떤 것이 스키마가 되는지는 데이터베이스 제품에 따라 달라진다.

MySQL의 경우에는 CREATE DATABASE 명령으로 작성한 데이터베이스가 스키마가 된다.

# 테이블 안에는 열을 정의할 수 있고 스키마 안에는 테이블을 정의할 수 있다.

# 스키마처럼 이름이 충돌하지 않도록 기능하는 그릇을 '네임스페이스(namespace)'라고 한다.



# Chapter 1.0 데이터베이스 객체란?

## ■ 수정사항

에일리어스(alias)라고도 불리는 별명은 영어, 숫자, 한글 등으로 지정할 수 있습니다. 단, 별명을 한글로 지정하는 경우에는 여러 가지로 오작동하는 경우가 많으므로 더블쿼트(MySQL에서는 백쿼트)로 둘러싸서 지정합니다. 이 룰은 데이터베이스 객체의 이름에 ASCII 문자 이외의 것을 사용할 경우에 해당합니다.

더블쿼트 → “  
싱글쿼트 → ‘  
백쿼트 → `

<https://stackoverflow.com/questions/11321491/when-to-use-single-quotes-double-quotes-and-back-ticks-in-mysql>

<https://teamtreehouse.com/community/single-or-double-quotes-in-mysql>



# Chapter 2



## Chapter 2.0 DDL 구문

### ■ DDL (Data Define Language)

DB를 정의하는 명령어이다.

대표적으로 CREATE, DROP, ALTER TRUNCATE가 해당된다.



## Chapter 2.1 DDL 구문 (작성)

### 테이블 작성 구문 이해하기

```
CREATE TABLE sample62(
  a INTEGER NOT NULL,
  b VARCHAR(30),
  c DATE );
```

CREATE TABLE 테이블명

컬럼명 자료형 [DEFAULT 기본값] [NULL | NOT NULL] ,  
# 컬럼명 작성시에는 명명 규칙을 지키도록 한다.

CHAR나 VARCHAR 같은 문자열형으로 지정할 때는 최대길이를 괄호로 묶어줘야 한다.

TYPE	사용되는 바이트	예제
CHAR(n)	정확히 n (<=255)	CHAR(5) 'Hello'는 5바이트 사용 CHAR(50) 'Hello'는 50바이트 사용
VARCHAR(n)	최대 n 까지(<=65535)	VARCHAR(100)'Hello'는 5바이트 사용 VARCHAR(5) 'Hello'는 5바이트 사용

TYPE	사용되는 바이트	속성
TINYTEXT(n)	최대 n (<=255)	문자열로 취급
TEXT(n)	최대 n (<=65535)	문자열로 취급
MEDIUMTEXT(n)	최대 n (<=16777215)	문자열로 취급
LONGTEXT(n)	최대 n (<=4294967295)	문자열로 취급

# VARCHAR는 각 값의 크기를 추적할 수 있는 약간의 오버헤드가 필요하기 때문에 모든 데이터의 크기가 비슷하다면 CHAR이 더 효율적이다..



## Chapter 2.2 DDL 구문 (작성)

### 테이블 정보

NOT NULL 조건을 걸었기 때문이다.

```
mysql> DESC sample62;
```

Field	Type	Null	Key	Default	Extra
no	int(11)	NO		NULL	
a	varchar(30)	YES		NULL	
b	date	YES		NULL	

```
3 rows in set (0.01 sec)
```

DEFAULT 값을 따로 지정하지 않으면 NULL로 지정된다.



## Chapter 2.2 DDL 구문 (작성)

### 테이블에 데이터 입력해보기

```
mysql> INSERT INTO sample62 VALUES(1, 'SQL세션', '2018-05-01');  
Query OK, 1 row affected (0.04 sec)
```

```
mysql> SELECT * FROM sample62 ;
```

no	a	b
1	SQL세션	2018-05-01

```
1 row in set (0.00 sec)
```



## Chapter 2.2 DDL 구문 (작성)

### 테이블에 데이터 입력해보기

```
mysql> INSERT INTO sample62 VALUES(1, '텍스트123456789012345678901234567890', '2018-01-01') ;  
ERROR 1406 (22001): Data too long for column 'a' at row 1
```

텍스트 길이가 30이 넘으니까 에러가 발생한다 !



## Chapter 2.2 DDL 구문 (삭제)

### 테이블 삭제 구문 이해하기

**DROP TABLE** sample62 ;

# 많은 데이터베이스가 SQL명령을 실행할 때 확인을 요구하지 않기 때문에 주의해야한다.  
# DROP TABLE은 물리 삭제!

```
mysql> DROP TABLE sample62 ;  
Query OK, 0 rows affected (0.01 sec)  
  
mysql> DESC sample62 ;  
ERROR 1146 (42S02): Table 'sample.sample62' doesn't exist
```



## Chapter 2.2 DDL 구문 (삭제)

### 데이터 행 삭제 구문 이해하기

```
DELETE FROM sample62 WHERE no = 1 ;
```

# 테이블 정의는 그대로 둔 채 데이터만 삭제한다.

# 여기서 WHERE 조건을 지정하지 않으면 테이블의 모든 행이 삭제된다.

# DELETE 명령은 행 단위로 여러 가지 내부처리가 일어나므로 삭제할 행이 많으면 처리속도가 상당히 늦어진다.

```
TRUNCATE TABLE sample62 ;
```

# TRUNCATE TABLE 구문은 삭제할 행을 지정할 수 없지만, 모든 행을 삭제해야 할 때 빠른 속도로 삭제할 수 있다.





## Chapter 2.2 DDL 구문 (삭제)

### 데이터 행 삭제 구문 이해하기

**DELETE** FROM sample62 **WHERE** no = 1 ;

```
mysql> SELECT * FROM sample62 ;
+----+-----+-----+
| no | a      | b      |
+----+-----+-----+
| 1  | SQL세션 | 2018-05-01 |
| 2  | PHP세션 | 2018-05-02 |
| 3  | JAVA세션 | 2018-05-03 |
+----+-----+-----+
3 rows in set (0.00 sec)

mysql> DELETE FROM sample62 WHERE no = 1 ;
Query OK, 1 row affected (0.01 sec)
```



```
mysql> SELECT * FROM sample62 ;
+----+-----+-----+
| no | a      | b      |
+----+-----+-----+
| 2  | PHP세션 | 2018-05-02 |
| 3  | JAVA세션 | 2018-05-03 |
+----+-----+-----+
2 rows in set (0.00 sec)
```



## Chapter 2.2 DDL 구문 (삭제)

### 데이터 행 삭제 구문 이해하기

**TRUNCATE TABLE** sample62 ;

# 테이블을 없애는 DROP과 달리 모든 행을 지우기만 하는 것! (테이블은 남아 있음)

```
mysql> TRUNCATE TABLE sample62 ;  
Query OK, 0 rows affected (0.04 sec)
```



```
mysql> SELECT * FROM sample62 ;  
Empty set (0.00 sec)
```

# 테이블이 아예 없을 때의 에러

```
ERROR 1146 (42S02): Table 'sample.sample62' doesn't exist
```



## Chapter 2.3 DDL 구문 (변경)

### 테이블 변경 구문 이해하기

#### ALTER TABLE 테이블명 변경명령

# ALTER TABLE 명령을 사용하면 테이블에 저장되어 있는 데이터는 그대로 남긴 채 구성만 변경을 할 수 있다.  
# ALTER TABLE로 할 수 있는 일은 크게 두 가지가 있다.



열 추가 / 열 삭제 / 열 변경

제약 추가 / 제약 삭제



## Chapter 2.3 DDL 구문 (변경)

### 테이블 변경 구문 이해하기 – 열 추가

**ALTER TABLE** 테이블명 **ADD** 열 정의

여기서의 열정의는 CREATE TABLE에서 사용하는 각 열에 대한 정의를 입력할 때와 동일하게 진행하면 된다.

- # 열 이름과 자료형을 지정하고 필요에 따라 기본값과 NOT NULL 제약을 지정한다.
- # 만일 기존 데이터 행이 존재하는 경우에 새로 열을 추가한다면 추가한 열의 값이 NULL이 된다.  
다만, 기본 값이 지정 되어있는 경우에는 기본 값으로 데이터가 채워진다.



## Chapter 2.3 DDL 구문 (변경)

### 테이블 변경 구문 이해하기 – 열 추가

**ALTER TABLE sample62 ADD newcol INTEGER ;**

새로 만들 칼럼명   열 정의 – 데이터 타입

```
mysql> ALTER TABLE sample62 ADD newcol INTEGER;
Query OK, 0 rows affected (0.45 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> DESC sample62;
```

Field	Type	Null	Key	Default	Extra
no	int(11)	NO		NULL	
a	varchar(30)	YES		NULL	
b	date	YES		NULL	
newcol	int(11)	YES		NULL	

```
4 rows in set (0.00 sec)
```



## Chapter 2.3 DDL 구문 (변경)

### 테이블 변경 구문 이해하기 – 열 속성 변경

**ALTER TABLE** 테이블명 **MODIFY** 열 정의

여기서의 열 정의 역시 CREATE TABLE에서 사용하는 각 열에 대한 정의를 입력할 때와 동일하게 진행하면 된다.

# MODIFY 구문으로 열 이름은 변경할 수 없지만, 자료형이나 기본 값, NOT NULL 제약 등의 속성은 변경할 수 있다.

# 만약에 기존 데이터 행이 존재하는 경우에는, 속성 변경에 따라 데이터 역시 변환이 된다. 예를 들어, 자료형을 변경한다면 테이블에 들어간 데이터의 자료형 역시 바뀌게 된다.

→ 물론 처리 과정에서 에러가 발생하면 명령은 수행되지 않는다.



## Chapter 2.3 DDL 구문 (변경)

### 테이블 변경 구문 이해하기 – 열 속성 변경

**ALTER TABLE** sample62 **MODIFY** newcol **VARCHAR(20)** ;

```
mysql> ALTER TABLE sample62 MODIFY newcol VARCHAR(20);
Query OK, 0 rows affected (0.15 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> DESC sample62;
```

Field	Type	Null	Key	Default	Extra
no	int(11)	NO		NULL	
a	varchar(30)	YES		NULL	
b	date	YES		NULL	
newcol	varchar(20)	YES		NULL	

데이터 타입이 바뀜

Field	Type	Null	Key	Default	Extra
no	int(11)	NO		NULL	
a	varchar(30)	YES		NULL	
b	date	YES		NULL	
newcol	int(11)	YES		NULL	

4 rows in set (0.00 sec)



## Chapter 2.3 DDL 구문 (변경)

### 테이블 변경 구문 이해하기 – 열 이름 변경

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

열 이름을 변경할 때에는 `MODIFY`가 아닌 `CHANGE`를 사용한다.  
`CHANGE`는 열 이름 뿐만 아니라 열 속성도 변경할 수 있다.





## Chapter 2.3 DDL 구문 (변경)

### 테이블 변경 구문 이해하기 – 열 이름 변경

**ALTER TABLE** sample62 **CHANGE** newcol c **VARCHAR(10)** ;

```
mysql> ALTER TABLE sample62 CHANGE newcol c VARCHAR(10);
Query OK, 0 rows affected (0.14 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> DESC sample62 ;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| no    | int(11)       | NO   |     | NULL    |       |
| a     | varchar(30)   | YES  |     | NULL    |       |
| b     | date          | YES  |     | NULL    |       |
| c     | varchar(10)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
4 rows in set (0.02 sec)
```

열 이름 및 데이터 타입이 바뀜

Field	Type	Null	Key	Default	Extra
no	int(11)	NO		NULL	
a	varchar(30)	YES		NULL	
b	date	YES		NULL	
newcol	varchar(20)	YES		NULL	

4 rows in set (0.02 sec)



## Chapter 2.3 DDL 구문 (변경)

### 테이블 변경 구문 이해하기 – 열 이름 변경

**ALTER TABLE sample62 CHANGE c cc VARCHAR(15) NOT NULL DEFAULT 'ORACLE' ;**

```
mysql> ALTER TABLE sample62 CHANGE c cc VARCHAR(15) NOT NULL DEFAULT 'ORACLE' ;
```

```
Query OK, 0 rows affected (0.03 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> DESC sample62 ;
```

Field	Type	Null	Key	Default	Extra
no	int(11)	NO		NULL	
a	varchar(30)	YES		NULL	
b	date	YES		NULL	
cc	varchar(15)	NO		ORACLE	

4 rows in set (0.00 sec)

Field	Type	Null	Key	Default	Extra
no	int(11)	NO		NULL	
a	varchar(30)	YES		NULL	
b	date	YES		NULL	
c	varchar(10)	YES		NULL	

4 rows in set (0.02 sec)



## Chapter 2.3 DDL 구문 (변경)

### 테이블 변경 구문 이해하기 – 열 삭제

`ALTER TABLE` 테이블명 `DROP` 열 이름

DROP 뒤에 삭제하고 싶은 열 이름을 지정한다.  
만일 테이블에 존재하지 않는 열을 지정할 경우 에러가 발생한다.



## Chapter 2.3 DDL 구문 (변경)

### 테이블 변경 구문 이해하기 – 열 이름 변경

**ALTER TABLE** sample62 **DROP cc ;**

```
mysql> ALTER TABLE sample62 DROP cc ;  
Query OK, 0 rows affected (0.03 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> DESC sample62 ;
```

cc 열이 삭제되었음을 확인할 수 있다.

Field	Type	Null	Key	Default	Extra
no	int(11)	NO		NULL	
a	varchar(30)	YES		NULL	
b	date	YES		NULL	

```
3 rows in set (0.00 sec)
```



# 중간 복습



# 📎 중간 복습

## 테이블 생성하기

```
CREATE TABLE sample99(
→ song_rank INT NOT NULL,
→ song_name TEXT NOT NULL,
→ song_genre CHAR(10),
→ artist_name VARCHAR(20)
→ );
```

Field	Type	Null	Key	Default	Extra
song_rank	int(11)	NO		NULL	
song_name	text	NO		NULL	
song_genre	char(10)	YES		NULL	
artist_name	varchar(20)	YES		NULL	

4 rows in set (0.02 sec)

여기서 **ALTER TABLE**의 **MODIFY**구문을 사용해서

- 1) song\_genre의 Type을 TEXT로 바꾸고 NULL 제약을 NOT NULL로 바꿔보자.
- 2) artist\_name의 NULL 제약을 NOT NULL로 바꿔보자.

# 중간 복습

## 테이블 변경 구문

**ALTER TABLE sample99 MODIFY song\_genre TEXT NOT NULL ;**  
**ALTER TABLE sample99 MODIFY artist\_name VARCHAR(20) NOT NULL ;**

```
mysql> DESC sample99 ;
```

Field	Type	Null	Key	Default	Extra
song_rank	int(11)	NO		NULL	
song_name	text	NO		NULL	
song_genre	text	NO		NULL	
artist_name	varchar(20)	NO		NULL	

4 rows in set (0.00 sec)



```
mysql> SELECT * FROM sample99 ;
```

song_rank	song_name	song_genre	artist_name
1	거짓말	댄스	빅뱅
2	소확행	포크	스탠딩에그
3	좋다	락	데이브레이크

3 rows in set (0.00 sec)

**데이터를 추가해보자**

**주의→**

```
mysql> ALTER TABLE sample99 MODIFY artist_name NOT NULL;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'NOT NULL' at line 1
```

# 📎 중간 복습

## 데이터 삽입 구문

```
mysql> INSERT INTO sample99 VALUES
  -> (1, '거짓말', '댄스', '빅뱅'),
  -> (2, '소확행', '포크', '스탠딩에그'),
  -> (3, '좋다', '락', '데이브레이크');
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

```
mysql> SELECT * FROM sample99 ;
```

song_rank	song_name	song_genre	artist_name
1	거짓말	댄스	빅뱅
2	소확행	포크	스탠딩에그
3	좋다	락	데이브레이크

```
3 rows in set (0.00 sec)
```

열을 추가하고 싶은 경우에는  
어떻게 하면 될까? (INT type)



new\_col

NULL

NULL

NULL



# 중간 복습

## 테이블 행 추가 구문

```
mysql> ALTER TABLE sample99 ADD new_col INT ;
Query OK, 0 rows affected (0.10 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> DESC sample99;
```

Field	Type	Null	Key	Default	Extra
song_rank	int(11)	NO		NULL	
song_name	text	NO		NULL	
song_genre	text	NO		NULL	
artist_name	varchar(20)	NO		NULL	
new_col	int(11)	YES		NULL	

```
5 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM sample99 ;
```

song_rank	song_name	song_genre	artist_name	new_col
1	거짓말	댄스	빅뱅	NULL
2	소확행	포크	스탠딩 에그	NULL
3	좋다	락	데이브레이크	NULL

```
3 rows in set (0.00 sec)
```

**ALTER TABLE sample99 ADD new\_col INT ;**

열 이름을 user\_score로 하고,  
Type을 FLOAT로 하고 싶다면  
어떻게 입력해야할까?

# 중간 복습

## 테이블 행 추가 구문

**ALTER TABLE** sample99 **CHANGE** new\_col user\_score **FLOAT** ;

```
mysql> ALTER TABLE sample99 CHANGE new_col user_score FLOAT ;
Query OK, 3 rows affected (0.09 sec)
Records: 3 Duplicates: 0 Warnings: 0
```

```
mysql> DESC sample99;
```

Field	Type	Null	Key	Default	Extra
song_rank	int(11)	NO		NULL	
song_name	text	NO		NULL	
song_genre	text	NO		NULL	
artist_name	varchar(20)	NO		NULL	
user_score	float	YES		NULL	

5 rows in set (0.01 sec)

**UPDATE**와 **CASE** 구문을 활용하여 각각의 user\_score에 다른 점수를 입력해보자.

**참고 사이트 →**

<http://www.seobangnim.com/zbxe/604897>

# 📎 중간 복습

## 다중 업데이트 구문

```
mysql> UPDATE sample99 SET user_score = CASE song_rank
  -> WHEN 1 THEN 99
  -> WHEN 2 THEN 85
  -> WHEN 3 THEN 70
  -> ELSE 0
  -> END
  -> WHERE song_rank IN (1, 2, 3) ;
Query OK, 3 rows affected (0.01 sec)
Rows matched: 3  Changed: 3  Warnings: 0
```

```
mysql> SELECT * FROM sample99 ;
```

song_rank	song_name	song_genre	artist_name	user_score
1	거짓말	댄스	빅뱅	99
2	소확행	포크	스탠딩에그	85
3	좋다	락	데이브레이크	70

```
3 rows in set (0.00 sec)
```

**user\_score**를 **평균** 몇 점을 부여했는지  
**함수 구문을 사용하여** 계산해보자.



## 중간 복습

### 산술 함수 구문

```
mysql> SELECT * FROM sample99 ;
```

song_rank	song_name	song_genre	artist_name	user_score
1	거짓말	댄스	빅뱅	99
2	소확행	포크	스탠딩 에그	85
3	좋다	락	데이브레이크	70

```
3 rows in set (0.00 sec)
```

```
mysql> SELECT AVG(user_score) FROM sample99 ;
```

AVG(user_score)
84.66666666666667

```
1 row in set (0.00 sec)
```

**복습 끝 !**



## Chapter 2.4 DDL 구문 (제약)

### 제약

테이블에 제약을 설정함으로써 저장될 데이터를 제한할 수 있다.

ALTER TABLE 구문으로 제약을 지정하거나 변경할 수 있지만 보통은 CREATE TABLE로 테이블을 작성할 때 제약을 정의한다.

```
mysql> CREATE TABLE sample631(  
  -> a INT NOT NULL,  
  -> b INT NOT NULL UNIQUE  
  -> c VARCHAR(30)  
  -> );  
Query OK, 0 rows affected (0.05 sec)
```

# 열에 대해 정의하는 제약은 '열 제약'

```
mysql> CREATE TABLE sample632(  
  -> no INT NOT NULL,  
  -> sub_no INT NOT NULL,  
  -> name VARCHAR(30),  
  -> PRIMARY KEY (no, sub_no)  
  -> );  
Query OK, 0 rows affected (0.05 sec)
```

# 한 개의 제약으로 복수의 열에 제약을 설정하는 '테이블 제약'



## Chapter 2.4 DDL 구문 (제약)

### 제약

테이블 제약에도 이름을 붙일 수 있다.

# 제약에 이름을 붙이면 나중에 관리하기가 쉬워진다고 한다.

```
mysql> CREATE TABLE sample632(
-> no INT NOT NULL,
-> sub_no INT NOT NULL,
-> name VARCHAR(30),
-> CONSTRAINT pkey_sample PRIMARY KEY (no, sub_no)
-> ) CONSTRAINT 제약이름
Query OK, 0 rows affected (0.05 sec)
```

```
mysql> DESC sample632 ;
```

Field	Type	Null	Key	Default	Extra
no	int(11)	NO	PRI	NULL	
sub_no	int(11)	NO	PRI	NULL	
name	varchar(30)	YES		NULL	

3 rows in set (0.02 sec)



## Chapter 2.4 DDL 구문 (제약)

### ■ 제약 추가

기존 테이블에도 나중에 제약을 추가할 수 있다! 다만, 제약을 추가할 때 기존의 행을 검사하여 추가할 제약을 위반하는 데이터가 있으면 에러가 발생한다.

Ex) 테이블에 이미 NULL 값이 존재하는 열에 NOT NULL 제약을 설정하려 할 경우.

**# 제약을 추가할 대상이 열 제약 / 테이블 제약인가에 따라서 사용하는 구문이 다르다.**



## Chapter 2.4 DDL 구문 (제약)

### 제약 추가 – 열 제약

열 제약을 추가할 경우에는 ALTER TABLE로 열 정의를 변경할 수 있다.

**ALTER TABLE sample631 MODIFY c VARCHAR(30) NOT NULL ;**

```
mysql> DESC sample631;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| a     | int(11)       | NO   |     | NULL    |       |
| b     | int(11)       | NO   | PRI | NULL    |       |
| c     | varchar(30)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.02 sec)

mysql> ALTER TABLE sample631 MODIFY c VARCHAR(30) NOT NULL ;
Query OK, 0 rows affected (0.08 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

Field	Type	Null	Key	Default	Extra
a	int(11)	NO		NULL	
b	int(11)	NO	PRI	NULL	
c	varchar(30)	NO		NULL	





## Chapter 2.4 DDL 구문 (제약)

### 제약 추가 – 테이블 제약

테이블 제약은 ALTER TABLE의 ADD 하부명령으로 추가할 수 있다.

**ALTER TABLE** sample631 **ADD CONSTRAINT** pkey\_sample631 **PRIMARY KEY(a)** ;

```
mysql> DESC sample631
-> ;
```

Field	Type	Null	Key	Default	Extra
a	int(11)	NO		NULL	
b	int(11)	NO	PRI	NULL	
c	varchar(30)	NO		NULL	

3 rows in set (0.00 sec)

```
mysql> ALTER TABLE sample631 ADD CONSTRAINT pkey_sample631 PRIMARY KEY(a) ;
Query OK, 0 rows affected (0.08 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> DESC sample631 ;
```

Field	Type	Null	Key	Default	Extra
a	int(11)	NO	PRI	NULL	
b	int(11)	NO	UNI	NULL	
c	varchar(30)	NO		NULL	

3 rows in set (0.00 sec)



## Chapter 2.4 DDL 구문 (제약)

### 제약 삭제 – 열 제약

열 제약을 삭제할 경우에는 추가할 때와 동일하게 열 정의를 변경하는 방법을 사용합니다.

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

```
mysql> ALTER TABLE sample631 MODIFY c VARCHAR(30);
Query OK, 0 rows affected (0.09 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> DESC sample631
-> ;
```

Field	Type	Null	Key	Default	Extra
a	int(11)	NO	PRI	NULL	
b	int(11)	NO	UNI	NULL	
c	varchar(30)	YES		NULL	

3 rows in set (0.00 sec)

NOT NULL 제한을 제외하여 추가할 때와 동일하게 그대로 작성한 것이다.



## Chapter 2.4 DDL 구문 (제약)

### ■ 제약 삭제 – 테이블 제약

테이블 제약을 삭제할 경우에는 ALTER TABLE의 DROP 구문을 사용하여 삭제합니다.  
단, 삭제할 때에는 제약명을 지정합니다. (단, MySQL에서는 적용 X)

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



## Chapter 2.4 DDL 구문 (제약)

### ■ 제약 삭제 – 기본키

기본키는 테이블당 하나만 설정할 수 있기 때문에 따로 제약명을 지정할 필요없이 바로 DROP 구문을 사용하면 된다.

```
ALTER TABLE sample631 DROP PRIMARY KEY ;
```



# 과제

## 과제

# 1. 다음 시간 퀴즈 준비하기. (단답형 주관식 1문제 : 30초)

퀴즈 예시) 다음 결과를 만들기 위해서 쿼리문의 빈칸을 채우시오.

```
mysql> SELECT * FROM sample21;
```

no	name	birthday	address
1	박준용	1976-10-18	대구광역시 수성구
2	김재진	NULL	대구광역시 동구
3	홍길동	NULL	서울특별시 마포구

3 rows in set (0.00 sec)



```
mysql> SELECT * FROM sample21  name;
```

no	name	birthday	address
2	김재진	NULL	대구광역시 동구
1	박준용	1976-10-18	대구광역시 수성구
3	홍길동	NULL	서울특별시 마포구

3 rows in set (0.00 sec)

*Growth  
Hackers*

# Thank you