

## 4. 기본 SQL 작성하기-DDL

# SQL 명령어

- DB에서는 모든 작업을 **SQL문**을 이용해 작업한다.
- SQL명령어는 3가지 그룹으로 나눈다. **DDL, DML, DCL**

## **DDL(Data Definition Language) : 데이터 정의어**

객체 생성, 변경, 삭제 명령어들  
create, alter, drop

## **DML(Data Manipulation Language) : 데이터 조작어**

데이터를 처리하는 언어(테이블에 있는 데이터 조작하는 언어)  
select, insert, update, delete

## **DCL(Data Control Language) : 데이터 제어어**

객체 권한 부여, 데이터 보안, 무결성 회복 등에 사용하는 언어.  
commit, rollback, grant, revoke

# DDL SQL명령어 - 생성, 삭제, 수정

## 생성 명령어

**CREATE** DATABASE [TABLE, USER, VIEW, INDEX, .....]

## 삭제 명령어

**DROP** DATABASE [TABLE, USER, VIEW, INDEX, .....]

## 수정 명령어

**ALTER** TABLE [USER, VIEW, INDEX, .....]

SQL문은 대소문자 구별 안함 – 대문자로 입력해도 다 소문자로 저장됨.

# 데이터베이스 생성, 삭제

```
CREATE DATABASE 데이터베이스명;
```

```
DROP DATABASE 데이터베이스명;
```

```
CREATE DATABASE testDB;  
SHOW DATABASES; 데이터베이스 목록 확인
```

```
DROP DATABASE testDB;
```

**주의!!** 데이터베이스에 저장된 모든 정보가 삭제

# 테이블

- 관계형 데이터베이스에서는 정보를 **테이블 형태로 보관**

The diagram illustrates a database table structure. A table with four columns and five rows is shown. The first row is the header, and the following four rows are data. Annotations include: '고객 릴레이션' (Customer Relation) pointing to the entire table; '속성(컬럼명)' (Attribute/Column Name) pointing to the header row; '컬럼' (Column) pointing to the '주소록' column; and '행(튜플)' (Row/Tuple) pointing to the data rows. A red box highlights the header row, and a purple box highlights the '주소록' column.

고객 릴레이션			
속성(컬럼명)			
고객번호	이름	전화번호	주소록
0001	김은미	001-145-1111	서울 마포구
0002	나윤희	001-111-1111	부산 동래구
0003	이연수	001-112-1111	경북 대구
0004	조서윤	001-124-1111	대전 동구

# 데이터정의어(DDL:DATA DEFINITION LANGUAGE)

테이블 정의 (설계된 테이블명세서로 실제 테이블로 정의)

**CREATE TABLE** 테이블\_이름( 값을 꼭 입력해야 한다.  
 {속성\_이름 데이터\_타입 [NOT NULL] [AUTO\_INCREMENT]}  
 { [PRIMARY KEY(컬럼명),] 기본키 자동증가  
 [UNIQUE(컬럼명),] 속성의 모든 값이 유일한 값  
 [DEFAULT 값] 값이 입력되지 않을 때 저장되는 값  
 [FOREIGN KEY(컬럼명) REFERENCES 참조테이블(컬럼명)] 외래키  
 );

제약조건

{ } 의 의미 : 반복가능, 즉 여러 번 쓸 수 있음.

[ ] : 생략가능. 필요하면 쓰고, 필요 없으면 안 써도 됨.

그 외에는 필수로 써야 함.

# 제약조건 - 데이터 무결성을 지키기 위한 조건

- **NOT NULL** - NULL값을 허용하지 않음
- **AUTO\_INCREMENT** - 자동으로 값 증가, 이 속성을 지정한 열은 반드시 primary key나 unique로 지정해야 한다.
- **UNIQUE** - 컬럼에 같은 값을 허용하지 않음, NULL은 허용
- **PRIMARY KEY** - 각 튜플을 구별하는 컬럼(**NOT NULL, UNIQUE**)
- **FOREIGN KEY** - 다른 테이블의 값을 참조할 때 사용하는 키
- **DEFAULT**- 값이 없을 경우 입력되는 값

department			employee			FOREIGN KEY
dept_no	dept_name	location	emp_no	name	department	
1	영업부	대구	1001	구창민	1	
2	인사부	서울	1002	김민서	1	
3	총무부	대구	1003	이은영	2	
4	기획부	서울	1004	한성일	2	

# 데이터 타입

데이터 타입의 종류가 30개 정도 됨. 모두 외울 필요 없고 자주 사용하는 것만 보기, 나머지는 필요할 때 인터넷 검색

데이터타입	바이트수	설명
<b>tinyint</b>	1	작은 정수이지만, <u>논리형으로 사용</u>
<b>int</b>	4	정수
float	4	소수점 아래 7자리까지 표현(근사치)
<b>double</b>	8	소수점 아래 15자리까지 표현(근사치)
decimal(m,[d])	5~17	전체 자릿수(m)와 소수점 이하 자릿수(d)를 가진 숫자형. 예)decimal(5,2)는 전체자릿수 5자리, 소수점 이하 2자리로. (근사치가 아니고 정확한 수치)
char(n)	1~255	고정길이 문자형(insert/update시 성능이 더 좋음)
<b>varchar(n)</b>	1~65535	가변길이 문자형(공간을 효율적으로 사용 가능)
<b>date</b>	3	날짜 'yyyy-mm-dd'형식
<b>datetime</b>	8	날짜와 시간 'yyyy-mm-dd hh:mm:ss'형식



## 예) 고객 테이블 만들기

<u>고객번호</u>	이름	전화번호	주소록
0001	김은미	001-145-1111	서울 마포구
0002	나윤희	001-111-1111	부산 동래구
0003	이연수	001-112-1111	경북 대구
0004	조서윤	001-124-1111	대전 동구

- 각 컬럼의 **데이터형**을 생각해 보자
- 각 컬럼의 **제약조건**을 생각해 보자
  - 기본키가 되는 컬럼을 생각해 보자
  - null값 허용 여부
  - default 값 여부
  - AUTO\_INCREMENT(1씩 자동 증가)

```
CREATE TABLE customer(  
  no int NOT NULL AUTO_INCREMENT,  
  name varchar(20) NOT NULL,  
  tel varchar(30) NOT NULL,  
  addr varchar(50) DEFAULT '강남구' ,  
  PRIMARY KEY(no),  
  UNIQUE(tel),  
);
```

# 도서관 데이터베이스 만들기

```
MariaDB [(none)]> create database librarydb;  
Query OK, 1 row affected (0.005 sec)
```

```
MariaDB [(none)]> show databases;
```

```
+-----+  
| Database |  
+-----+  
| information_schema |  
| librarydb          |  
| mysql              |  
| performance_schema |  
| test               |  
+-----+
```

```
5 rows in set (0.009 sec)
```

```
MariaDB [(none)]> use librarydb;
```

```
Database changed
```

```
MariaDB [librarydb]>
```

# 테이블 만들기

## 회원 테이블 만들기 - 이전 수업 테이블 명세서를 보고 만드세요.

```
MariaDB [librarydb]> create table member(mem_id int primary key auto_increment, name varchar(20) not null,  
grade varchar(1) not null default 'A', tel varchar(15) not null, addr varchar(20) );  
Query OK, 0 rows affected (0.025 sec)
```

```
MariaDB [librarydb]> desc member;
```

Field	Type	Null	Key	Default	Extra
mem_id	int(11)	NO	PRI	NULL	auto_increment
name	varchar(20)	NO		NULL	
grade	varchar(1)	NO		A	
tel	varchar(15)	NO		NULL	
addr	varchar(20)	YES		NULL	

```
5 rows in set (0.028 sec)
```

# 테이블 만들기

## 도서 테이블 - 이전 수업 테이블 명세서를 보고 만드세요.

```
MariaDB [librarydb]> create table book(book_id int auto_increment, title varchar(20) not null, writer varchar(20) not null, company varchar(20) not null, booking tinyint not null default false, primary key(book_id) );
Query OK, 0 rows affected (0.027 sec)
```

```
MariaDB [librarydb]> desc book;
```

Field	Type	Null	Key	Default	Extra
book_id	int(11)	NO	PRI	NULL	auto_increment
title	varchar(20)	NO		NULL	
writer	varchar(20)	NO		NULL	
company	varchar(20)	NO		NULL	
booking	tinyint(4)	NO		0	

```
5 rows in set (0.026 sec)
```

# 테이블 만들기

## 대여 테이블 - 이전 수업 테이블 명세서를 보고 만드세요.

```
MariaDB [librarydb]> create table rent(mem_id int, book_id int, rent_date date not null, return_date date, primary key(mem_id, book_id), foreign key(mem_id) references member(mem_id), foreign key(book_id) references book(book_id) );
Query OK, 0 rows affected (0.031 sec)
```

```
MariaDB [librarydb]> desc rent;
```

Field	Type	Null	Key	Default	Extra
mem_id	int(11)	NO	PRI	NULL	
book_id	int(11)	NO	PRI	NULL	
rent_date	date	NO		NULL	
return_date	date	YES		NULL	

```
4 rows in set (0.029 sec)
```

# 테이블 만들기

## 테이블 제약조건 보기

- 제약조건도 테이블에 저장되어 있음.
- information\_schema 데이터베이스의 table\_constraints 테이블에서 table\_name 필드에 테이블명을 주고 조회.

```
>use information_schema;  
>select * from table_constraints where table_name='rent';
```

```
MariaDB [information_schema]> select * from table_constraints where table_name='rent';
```

CONSTRAINT_CATALOG	CONSTRAINT_SCHEMA	CONSTRAINT_NAME	TABLE_SCHEMA	TABLE_NAME	CONSTRAINT_TYPE
def	librarydb	PRIMARY	librarydb	rent	PRIMARY KEY
def	librarydb	rent_ibfk_1	librarydb	rent	FOREIGN KEY
def	librarydb	rent_ibfk_2	librarydb	rent	FOREIGN KEY

```
3 rows in set (0.006 sec)
```

```
MariaDB [information_schema]> select * from table_constraints where table_name='member';
```

CONSTRAINT_CATALOG	CONSTRAINT_SCHEMA	CONSTRAINT_NAME	TABLE_SCHEMA	TABLE_NAME	CONSTRAINT_TYPE
def	librarydb	PRIMARY	librarydb	member	PRIMARY KEY

```
1 row in set (0.001 sec)
```

# 테이블 만들기

## 테이블 제약조건 보기

- 기본키와 외래키를 만들면 index가 생성되기때문에 index를 조회해도 된다.

```
>show index from rent;  
>show index from member;
```

```
MariaDB [librarydb]> show index from rent;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment
rent	0	PRIMARY	1	mem_id	A	0	NULL	NULL		BTREE		
rent	0	PRIMARY	2	book_id	A	0	NULL	NULL		BTREE		
rent	1	book_id	1	book_id	A	0	NULL	NULL		BTREE		

```
3 rows in set (0.004 sec)
```

```
MariaDB [librarydb]> show index from member;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment
member	0	PRIMARY	1	mem_id	A	0	NULL	NULL		BTREE		

```
1 row in set (0.002 sec)
```

# 테이블 변경하기 - ALTER

## 테이블명 변경

```
ALTER TABLE 테이블_이름 RENAME 변경할_테이블_이름;
```

## 테이블 구조 확인

```
DESC 테이블_이름;
```

## 테이블에 컬럼 수정

```
ALTER TABLE 테이블_이름 ADD 컬럼명 데이터형; // 추가
```

```
ALTER TABLE 테이블_이름 MODIFY 컬럼명 데이터형 [after | first 기준컬럼명];  
// 컬럼의 데이터형과 컬럼 위치 바꾸기 가능
```

```
ALTER TABLE 테이블_이름 DROP [COLUMN] 컬럼명; // 삭제
```

```
ALTER TABLE 테이블_이름 CHANGE 컬럼명 변경할컬럼명 데이터형;  
// 컬럼명과 데이터형 수정가능
```



# 테이블 변경하기

## 테이블명 변경

- rent 테이블의 테이블 이름을 rent\_t로 변경하기

```
MariaDB [information_schema]> use librarydb;
Database changed
MariaDB [librarydb]> alter table rent rename rent_t;
Query OK, 0 rows affected (0.015 sec)

MariaDB [librarydb]> show tables;
+-----+
| Tables_in_librarydb |
+-----+
| book                 |
| member               |
| rent_t               |
+-----+
3 rows in set (0.001 sec)

MariaDB [librarydb]>
```

# 테이블 변경하기

## 테이블에 컬럼 추가

- rent\_t 테이블에 아래 컬럼(필드)를 추가하세요.

필드명(test) 데이터타입(int)

```
MariaDB [librarydb]> desc rent_t;
```

Field	Type	Null	Key	Default	Extra
mem_id	varchar(15)	NO	PRI	NULL	
book_id	varchar(15)	NO	PRI	NULL	
rent_date	date	NO		NULL	
return_date	date	NO		NULL	

4 rows in set (0.012 sec)

```
MariaDB [librarydb]> alter table rent_t add test int;
```

Query OK, 0 rows affected (0.011 sec)  
Records: 0 Duplicates: 0 Warnings: 0

```
MariaDB [librarydb]> desc rent_t;
```

Field	Type	Null	Key	Default	Extra
mem_id	varchar(15)	NO	PRI	NULL	
book_id	varchar(15)	NO	PRI	NULL	
rent_date	date	NO		NULL	
return_date	date	NO		NULL	
test	int(11)	YES		NULL	

5 rows in set (0.013 sec)

# 테이블 변경하기

## 테이블 컬럼의 데이터타입 변경

- rent\_t 테이블의 test컬럼(필드) 데이터타입을 varchar(4)로 변경하세요.

```
MariaDB [librarydb]> alter table rent_t modify test varchar(4);
Query OK, 0 rows affected (0.049 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
MariaDB [librarydb]> desc rent_t;
```

Field	Type	Null
mem_id	varchar(15)	NO
book_id	varchar(15)	NO
rent_date	date	NO
return_date	date	NO
test	varchar(4)	YES

```
5 rows in set (0.012 sec)

MariaDB [librarydb]>
```

```
MariaDB [librarydb]> alter table rent_t change test test varchar(4);
Query OK, 0 rows affected (0.032 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [librarydb]> desc rent_t;
```

Field	Type	Null	Key	Default	Extra
mem_id	varchar(15)	NO	PRI	NULL	
book_id	varchar(15)	NO	PRI	NULL	
rent_date	date	NO		NULL	
return_date	date	NO		NULL	
test	varchar(4)	YES		NULL	

```
5 rows in set (0.012 sec)

MariaDB [librarydb]>
```

- **modify** : 필드의 데이터 타입, 위치 변경 가능
- **change** : 필드의 데이터 타입, 필드명 변경 가능

# 테이블 변경하기

## 테이블 컬럼의 이름 변경

- rent\_t 테이블의 test필드의 이름을 imsi로 변경하세요.

```
MariaDB [librarydb]> alter table rent_t change test imsi varchar(4);  
Query OK, 0 rows affected (0.033 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

```
MariaDB [librarydb]> desc rent_t;
```

Field	Type	Null	Key	Default	Extra
mem_id	int(11)	NO	PRI	NULL	
book_id	int(11)	NO	PRI	NULL	
rent_date	date	NO		NULL	
return_date	date	YES		NULL	
imsi	varchar(4)	YES		NULL	

```
5 rows in set (0.028 sec)
```

# 테이블 변경하기

## 테이블 컬럼의 위치 변경

- rent\_t 테이블의 imsi 필드를 book\_id 뒤로 위치를 변경하세요.

```
MariaDB [librarydb]> alter table rent_t modify imsi varchar(4) after book_id;  
Query OK, 0 rows affected (0.035 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

```
MariaDB [librarydb]> desc rent_t;
```

Field	Type	Null	Key	Default	Extra
mem_id	int(11)	NO	PRI	NULL	
book_id	int(11)	NO	PRI	NULL	
imsi	varchar(4)	YES		NULL	
rent_date	date	NO		NULL	
return_date	date	YES		NULL	

```
5 rows in set (0.027 sec)
```

# 테이블 변경하기

## 테이블 컬럼 삭제

- rent\_t 테이블의 imsi 필드를 삭제하세요.

```
MariaDB [librarydb]> alter table rent_t drop imsi;  
Query OK, 0 rows affected (0.026 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

```
MariaDB [librarydb]> desc rent_t;
```

Field	Type	Null	Key	Default	Extra
mem_id	int(11)	NO	PRI	NULL	
book_id	int(11)	NO	PRI	NULL	
rent_date	date	NO		NULL	
return_date	date	YES		NULL	

```
4 rows in set (0.028 sec)
```

# 테이블 삭제하기

## 정의된 테이블 삭제

```
DROP TABLE 테이블_이름;
```

**주의 !!** drop table 명령어는 테이블 안에 데이터가 있을 경우, 모든 데이터를 지우고 실행되므로 주의해서 사용해야 한다.

# 테이블 삭제하기

## 테이블 삭제

- test 테이블(t1 int 필드만 가짐)을 만드세요.
- test 테이블을 삭제하세요.

```
MariaDB [librarydb]> create table test(t1 int);  
Query OK, 0 rows affected (0.020 sec)
```

```
MariaDB [librarydb]> show tables;
```

Tables_in_librarydb
book
member
rent_t
test

```
4 rows in set (0.001 sec)
```

```
MariaDB [librarydb]> drop table test;  
Query OK, 0 rows affected (0.024 sec)
```

```
MariaDB [librarydb]> show tables;
```

Tables_in_librarydb
book
member
rent_t

```
3 rows in set (0.001 sec)
```