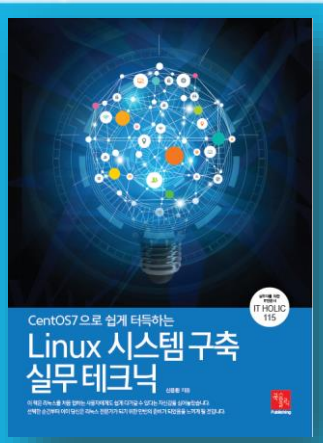
CHAPTER 13

데이터베이스 서버 구축 및 운영



Section 01 | DBMS의 개념과 역할

Section 02 | MariaDB 설치 및 운영

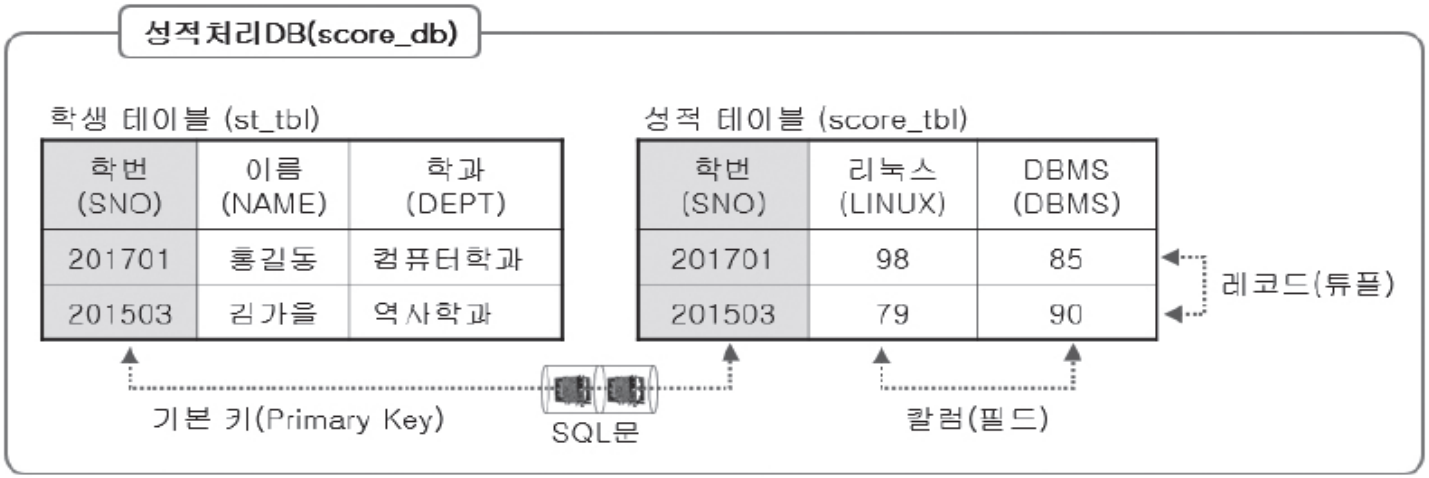
DBMS의 개념과 역할

- DBMS에 대해 이해하기

1. DBMS에 대한 개념과 역할을 이해합니다.
2. MariaDB 설치 및 접속 방법에 대해 실습합니다.
3. Oracle DB 설치 및 접속 방법에 대해 살펴봅니다.

DBMS의 이해

DBMS[DataBase Management System]는 데이터베이스를 직접 응용 프로그램들이 조작하는 것이 아니고 데이터베이스를 조작하는 별도의 소프트웨어를 DBMS(데이터베이스 관리 시스템)라고 합니다. 데이터베이스의 형태로 저장된 방대한 양의 각종 정보를 체계적으로 관리하는 기업용 소프트웨어와 인터넷의 발달과 함께 기업정보화가 급속도로 진행되면서 DBMS는 컴퓨터 운영체제[OS;Operating System]에 버금가는 중요한 소프트웨어로 인식되고 있습니다.



[그림 13-1] DBMS 개념도

필수 SQL 문

SQL[Structured Query Language]이란 데이터베이스를 사용할 때 데이터베이스에 접근할 수 있는 데이터베이스의 하부 언어를 의미하며 다른 표현으로는 구조화 질의어라고 합니다. 데이터베이스용 질의언어인 SQL 문에는 데이터 정의어[DDL;Data Definition Language]와 데이터 조작어[DML;Data Definition Language]를 포함하고 있습니다. SQL문과 관련하여 많은 내용들이 있지만 여기서는 꼭 필요한 요건들에 대해서만 살펴보도록 하겠습니다.

데이터베이스 관련 SQL문

먼저 데이터베이스의 생성, 삭제, 조회, 사용과 관련된 SQL문을 살펴보도록 하겠습니다. SQL문에서는 알파벳 대/소문자를 구별하지 않으며 각 문장의 끝에는 세미콜론(;)으로 끝내야 합니다. 그 이유는 한 문장의 명령은 여기까지라는 의미를 부여하기 위함입니다.

❶ 데이터베이스 조회 : show 명령으로 기존에 생성된 데이터베이스 목록을 출력합니다.

```
show databases;
```

❷ 데이터베이스 생성 : create 명령으로 새로운 데이터베이스를 생성합니다.

```
create database DB이름;
```

❸ 데이터베이스 지정 : use 명령으로 기존에 존재하는 여러 데이터베이스 중에서 작업할 데이터베이스를 선택합니다.

```
use DB이름;
```

❹ 데이터베이스 삭제 : drop 명령으로 특정 데이터베이스를 지정하여 삭제합니다. 이때 해당 DB에 존재하는 테이블도 같이 삭제되므로 주의해야 합니다.

```
drop database DB이름;
```

테이블 관련 SQL문

데이터베이스를 생성한 다음에는 테이블을 생성하는 과정으로 전개됩니다. 이 과정에서 DBMS가 수행되므로 선행되어야 할 부분은 DB생성이 우선이라는 점입니다. 생성된 DB를 삭제하게 되면 그와 관련된 모든 테이블도 함께 삭제됩니다. 테이블에 대해 조작하려면 먼저 소속되어 있는 데이터베이스와 관련됨을 면밀히 살펴봐야 합니다.

❶ 테이블 조회 : show 명령을 사용하여 기존에 생성된 테이블 목록을 출력합니다.

```
show tables;
```

❷ 테이블 생성 : create 명령으로 새로운 테이블을 생성합니다. 테이블을 생성할 때는 각 필드에 어떠한 데이터를 저장할 것인지에 대한 데이터 타입을 정해주어야 합니다. 테이블에서 사용되는 데이터 타입은 [표 13-1]과 같습니다.

[표 13-1] 테이블 필드의 데이터 타입

데이터 타입	의미
varchar(n)	개수 n개를 지정하여 가변성 문자열 지정 (메모리 효율성)
char(n)	개수 n개를 지정하여 고정성 문자열 지정 (메모리 비효율성)
int	정수형 숫자를 입력받을 필드의 데이터 타입 지정
float	실수형 숫자를 입력받을 필드의 데이터 타입 지정
date	날짜를 입력받을 필드의 데이터 타입 지정
time	시각을 입력받을 필드의 데이터 타입 지정

테이블을 생성할 때는 테이블 이름 다음에 괄호를 열고 필드명과 데이터 타입을 다음과 같은 형태로 지정합니다.

```
create table 테이블명(필드명1 데이터타입1, 필드명2 데이터타입2, …);
```

❸ 테이블 구조 조회 : explain 또는 desc 명령을 사용하여 테이블의 구조를 조회합니다.

```
explain 테이블명; 또는 desc 테이블명;
```

❹ 테이블 수정 : alter 명령을 사용하여 테이블의 구조와 필드명 등을 수정합니다.

```
alter table 테이블명 수정할 내용;
```

❺ 테이블 삭제 : drop 명령을 사용하여 테이블을 삭제합니다.

```
drop table 테이블명;
```

레코드 관련 SQL문

데이터베이스의 테이블에 데이터를 입력하거나 삭제할 때는 레코드 단위로 작업이 수행됩니다. 레코드를 삽입하거나 삭제, 수정하기 위해서 사용되는 SQL문에 대해 살펴보도록 하겠습니다.

❶ **레코드 삽입** : insert into 명령을 사용하여 지정한 테이블에 새로운 레코드를 추가합니다. 이 과정에서 주의해야 할 점은 테이블을 생성할 때 지정한 데이터 타입과 일치되도록 데이터를 입력해야 합니다.

```
insert into 테이블명 values(값1, 값2, …);
```

❷ **레코드 수정** : update 명령을 사용하여 기존 테이블에 저장되어 있는 레코드를 수정합니다.

```
update 테이블명 set 필드명1=수정할 값, 필드명2=수정할 값, … where 조건;
```

❸ **레코드 삭제** : delete 명령을 사용하여 기존 테이블에 저장된 레코드 중에서 지정한 레코드를 삭제합니다.

```
delete from 테이블명 where 조건;
```

❹ **레코드 조회** : select 명령을 사용하여 테이블에 저장된 레코드를 조회합니다.

```
select 필드명1, 필드명2, … from 테이블명 where 조건;
```

접근권한 부여

데이터베이스 작업은 주로 관리자가 DB를 생성하여 관리하지만 때로는 사용자에게 권한을 부여하는 경우도 있습니다. 예를 들어 제로보드와 같은 게시판을 설치하고자 할 때는 관리자가 일일이 데이터베이스를 관리하는 것보다는 해당 사용자에게 모든 권한을 부여하여 관리하도록 하는 것이 유지관리 측면에서 훨씬 더 효과적일 수 있습니다. 데이터베이스에 대한 접근권한은 grant명령을 사용하여 다음과 같이 권한을 부여합니다.

```
grant all privileges on DB명.* to 사용자계정@localhost identified by '암호';
```

[그림 13-1]에서 생성한 성적처리 데이터베이스인 score_db를 cskisa 사용자에게 관리자계정 암호 123456으로 지정하여 다음과 같이 접근권한을 부여하면 됩니다.

```
grant all privileges on score.db.* to cskisa@localhost identified by
'123456';
```

이상으로 DBMS와 SQL문에 대해서 간략하게 살펴봤습니다. DB를 처음 접하는 독자들에게는 다소 어렵게 느껴지는 부분이겠지만 실습을 통해 차츰 익숙해 질 용어들과 사용방법이니 너무 걱정하지 않아도 됩니다. SQL문에 대한 내용은 훨씬 다양하고 복잡한 기능을 제공하고 있지만 여기서는 꼭 필요한 내용만 다루었습니다. 이 정도 내용만 알고가도 우리가 사용할 DBMS를 운영하는 데에는 큰 불편함이 없을 것입니다.

MariaDB 설치 및 운영

- DBMS 설치 및 운영방법 이해하기

1. MariaDB 설치와 활성화 방법에 대해 살펴합니다.
2. 데이터베이스와 테이블 생성방법에 대해 실습합니다.
3. 테이블에 저장된 레코드 검색방법에 대해 이해합니다.

MariaDB 설치 및 활성화

MariaDB 서버는 예전에 사용하던 MySQL과 동일한 소스를 사용하여 호환성을 높였습니다. MariaDB에서의 명령은 MySQL과 정확한 대응 서비스를 제공하고 있기 때문에 점증적으로 MySQL에서 MariaDB로 전환하는 사용빈도가 높아지고 있습니다. 그리고 MariaDB는 기본적으로 CentOS 7에 설치된 MySQL의 드롭인 대체품이며 많은 스토리지 엔진을 제공하고 있기 때문에 MySQL에 비해 빠른 속도와 향상된 성능을 사용할 수 있게 되었습니다.

CentOS 7에 기본적으로 설치되어 있는 MariaDB는 5.5 버전이 포함되어 있으며 여기서는 CentOS 7에서 기본적으로 제공되는 MariaDB 5.5에 대해서만 다루기로 하겠습니다. 다음 예제를 통해 MariaDB 서버를 설치하는 과정과 MariaDB 서버를 활성화 하는 과정에 대해 살펴보도록 하겠습니다.

예제 13-1

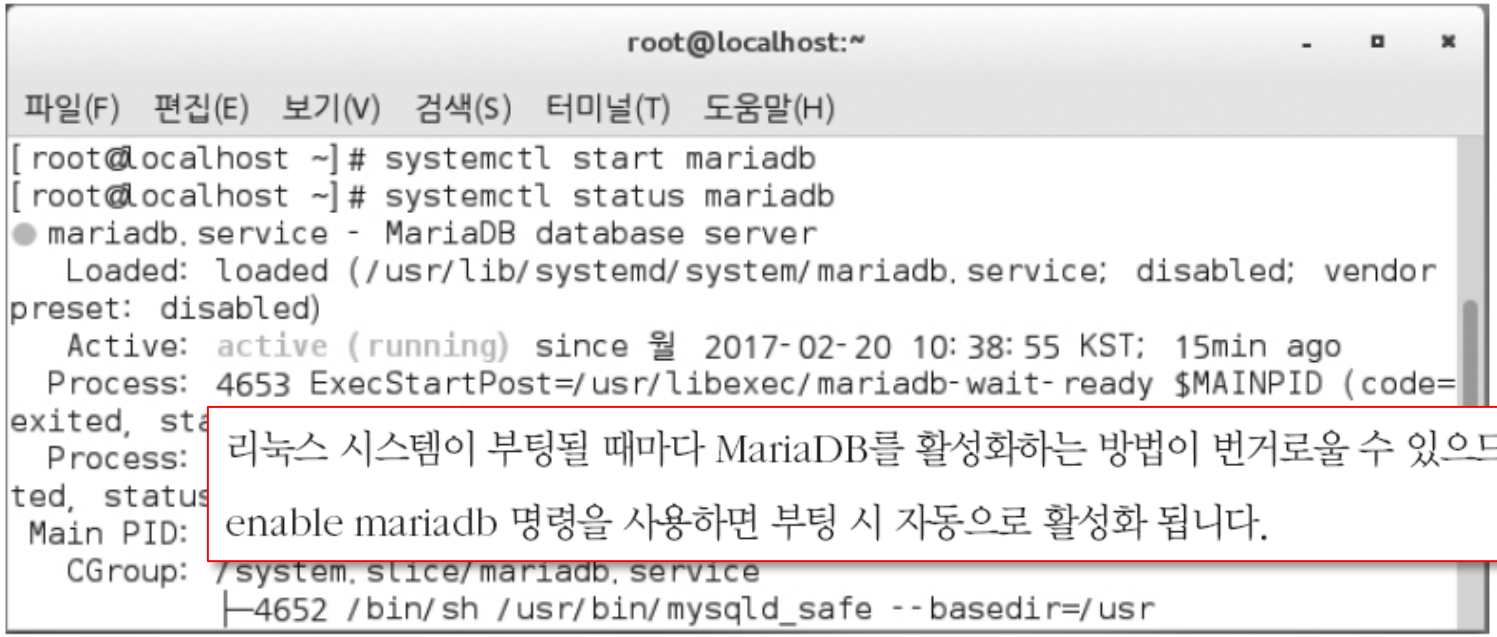
• **Step 01** | MariaDB를 설치하기 위해 터미널 창에서 다음과 같이 명령을 수행합니다.

```
# yum -y install mariadb-server mariadb
```

[그림 13-2] MariaDB 패키지 설치

- **Step 02** | MariaDB 활성화를 한 다음 상태를 확인하기 위해 터미널 창에서 다음과 같이 명령을 수행합니다.

```
# systemctl start mariadb
# systemctl status mariadb
```



리눅스 시스템이 부팅될 때마다 MariaDB를 활성화하는 방법이 번거로울 수 있으므로 systemctl enable mariadb 명령을 사용하면 부팅 시 자동으로 활성화 됩니다.

[그림 13-3] MariaDB 활성화 및 상태 확인

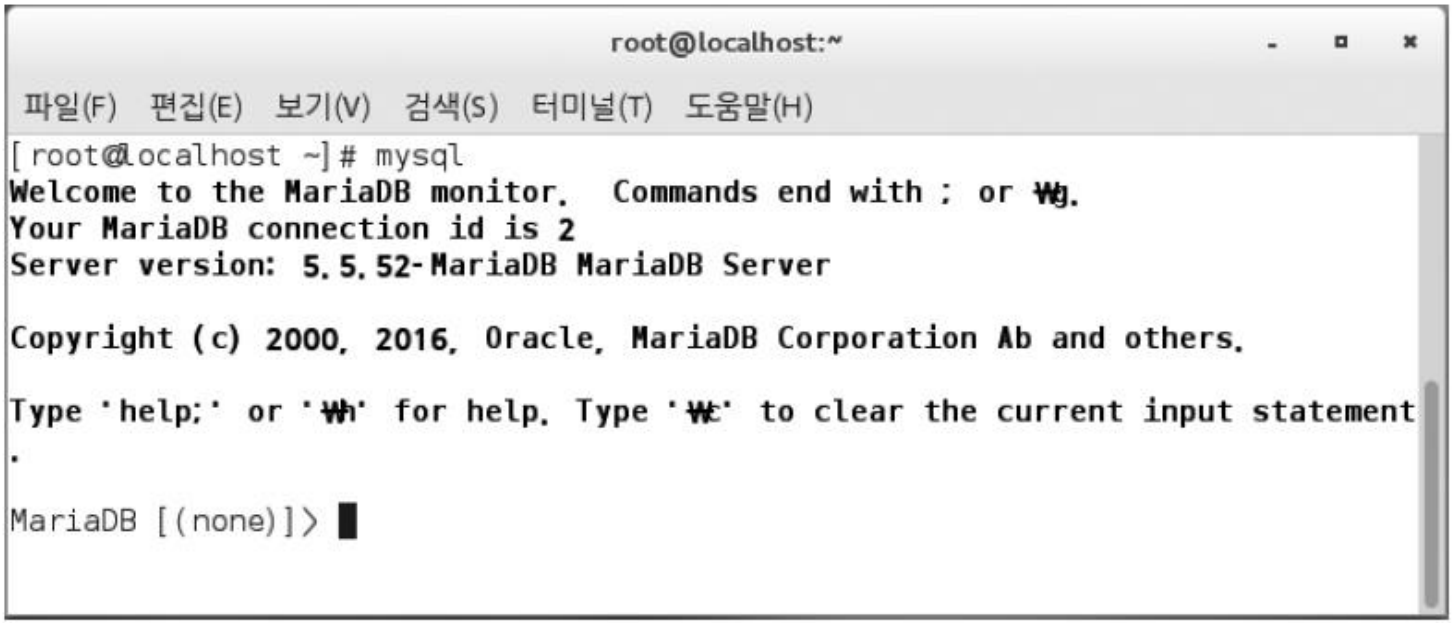
MariaDB 서버 접속

MariaDB에 대한 패키지 설치와 상태를 확인하였으니 이제는 본격적으로 MariaDB를 사용하는 과정에 대해 살펴보도록 하겠습니다. MariaDB 서버에 접속하는 과정을 다음 예제를 통해 자세히 살펴보도록 하겠습니다.

예제 13-2

MariaDB 서버에 접속하기 위해 터미널 창에서 다음과 같이 mysql명령을 수행합니다. 종료하려면 exit 명령을 입력하면 됩니다.

```
# mysql
```



[그림 13-4] MariaDB 서버 접속

데이터베이스 생성 및 활용

MariaDB에 접속한 다음에는 데이터베이스 생성과 활용방법에 대해 살펴보도록 하겠습니다. 데이터베이스의 생성은 create 명령을 사용하며 데이터베이스를 활용하기 위해 특정 데이터베이스를 지정하여 선택할 경우에는 use 명령을 사용하여 SQL문을 수행합니다.

터미널 창에서 접속된 MariaDB 서버 화면에서 여러 SQL문을 작성하다보면 화면을 깨끗이 지우고 싶을 때가 생깁니다. 이럴 때는 system clear 명령을 사용하여 화면을 깨끗하게 지우면 됩니다. 여기서는 데이터베이스의 생성과 활용하는 방법에 대해서만 다루기로 하겠습니다.

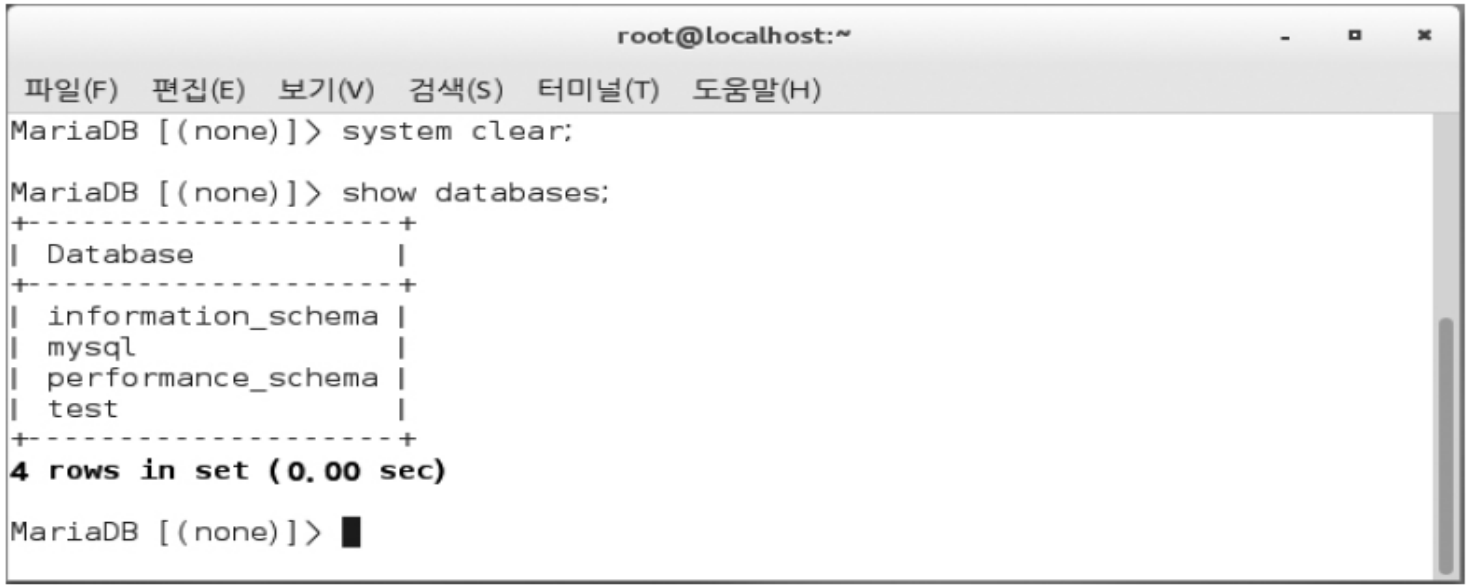
기존 데이터베이스 목록 확인

MariaDB 서버에 접속되었으면 기존에 생성되어 있는 데이터베이스 목록에 대해 살펴보도록 하겠습니다.

예제 13-3

MariaDB 서버에 존재하는 데이터베이스의 목록을 확인하기 위해 show 명령을 사용하여 다음과 같이 SQL문을 수행합니다. system clear 명령은 화면을 깨끗하게 지운다는 의미이며 SQL문의 끝에는 반드시 세미콜론(;)을 붙여야 합니다.

```
MariaDB[(none)]> system clear;          → 화면 지우기
MariaDB[(none)]> show databases;        → MariaDB 서버에 존재하는 모든 DB 출력
```



[그림 13-5] 데이터베이스 목록 확인

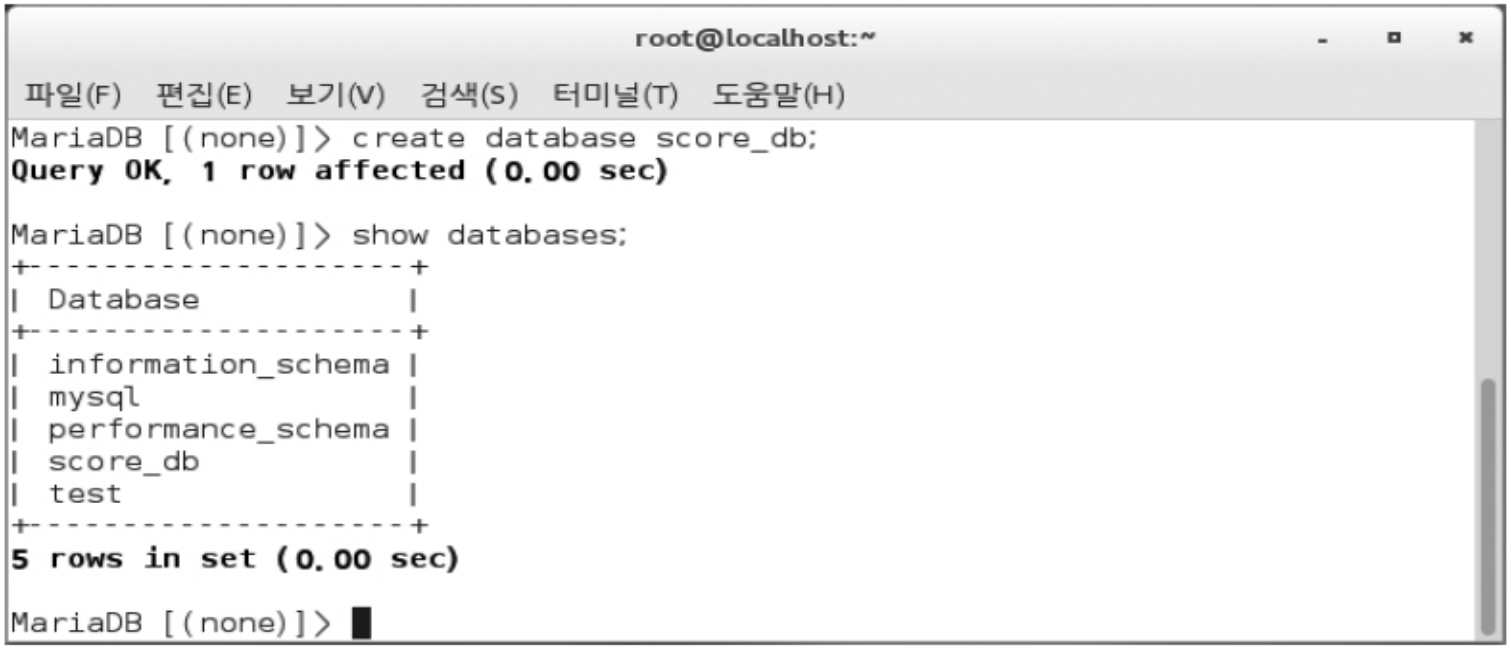
새로운 데이터베이스 생성

기존에 생성되어 있는 데이터베이스 목록을 확인했으니 이번에는 새로운 데이터베이스 score_db를 생성한 다음 생성된 데이터베이스가 목록에 존재하는지에 대해 살펴보도록 하겠습니다.

예제 13-4

create 명령으로 새로운 데이터베이스 score_db를 생성하기 위해 다음과 같이 SQL문을 수행합니다.

```
MariaDB[(none)]> create database score_db;   → 새로운 DB score_db 생성
MariaDB[(none)]> show databases;      → MariaDB 서버에 존재하는 모든 DB 출력
```



[그림 13-6] score_db 생성 후 확인

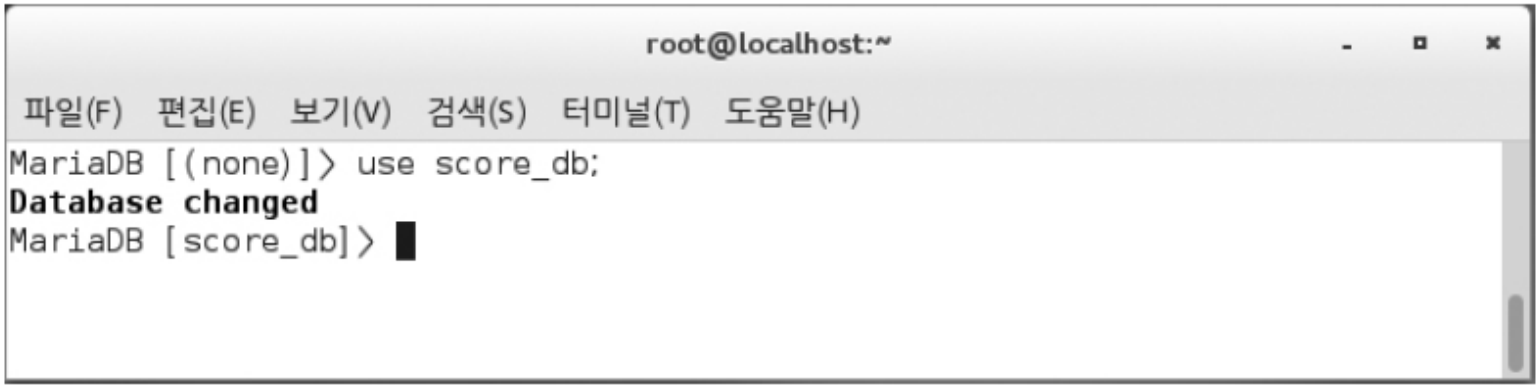
데이터베이스 선택하여 사용

실질적으로 데이터를 입력하기 위해 테이블을 생성하기 전에 앞에서 생성한 score_db 데이터베이스를 use 명령으로 선택해야 하는데 이 과정은 예제를 통해 살펴보도록 하겠습니다.

예제 13-5

use 명령으로 사용하고자 하는 데이터베이스 score_db를 선택하기 위해 다음과 같이 SQL문을 수행합니다.

```
MariaDB[(none)]> use score_db;   → 데이터베이스 score_db 선택
MariaDB[score_db]>
```



[그림 13-7] 데이터베이스 score_db 선택

MariaDB 설치 및 활성화

작업할 테이블 생성

데이터베이스 score_db를 선택한 다음에는 데이터를 입력할 테이블을 생성해야 합니다. 기존에 존재하는 테이블 목록을 먼저 살펴보도록 하겠습니다.

예제 13-6

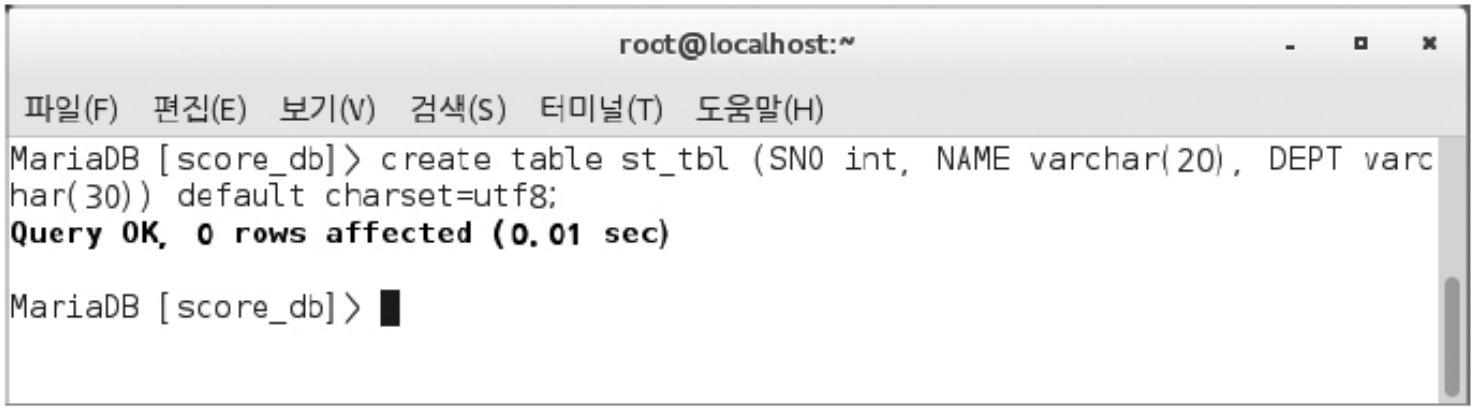
- **Step 01** | show 명령으로 사용하고자 하는 데이터베이스 score_db를 선택하기 위해 다음과 같이 SQL문을 수행합니다.

```
MariaDB[score_db]> show tables;   → 기존에 존재하는 테이블 목록 출력
```

[그림 13-8] 기존에 존재하는 테이블 목록 출력

- **Step 02** | 학생 테이블 st_tbl을 create 명령으로 생성합니다. 테이블에서 한글을 사용하고자 할 경우에는 글자가 깨지지 않도록 default charset=utf8을 추가해 다음과 같이 SQL문을 수행합니다.

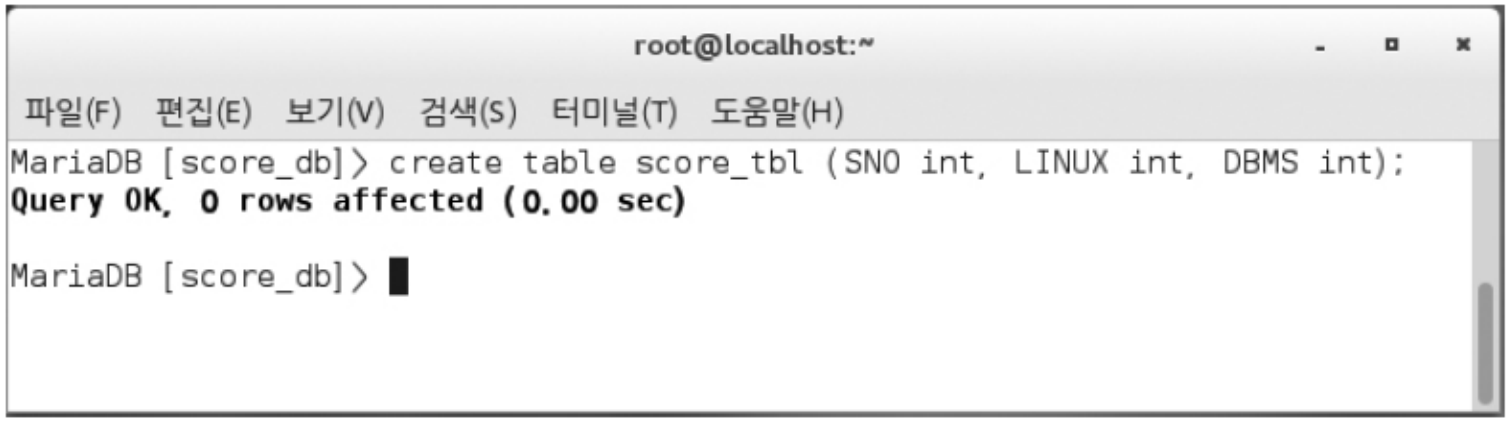
```
MariaDB[score_db]> create table st_tbl (SNO int, NAME varchar(20), DEPT
varchar(30)) default charset=utf8;
```



[그림 13-9] 학생 테이블 st_tbl 생성

- **Step 03** | 이번에는 성적 테이블 score_tbl을 create 명령으로 생성합니다. 테이블에서 한글을 사용하지 않을 경우에는 default charset=utf8 설정을 생략합니다.

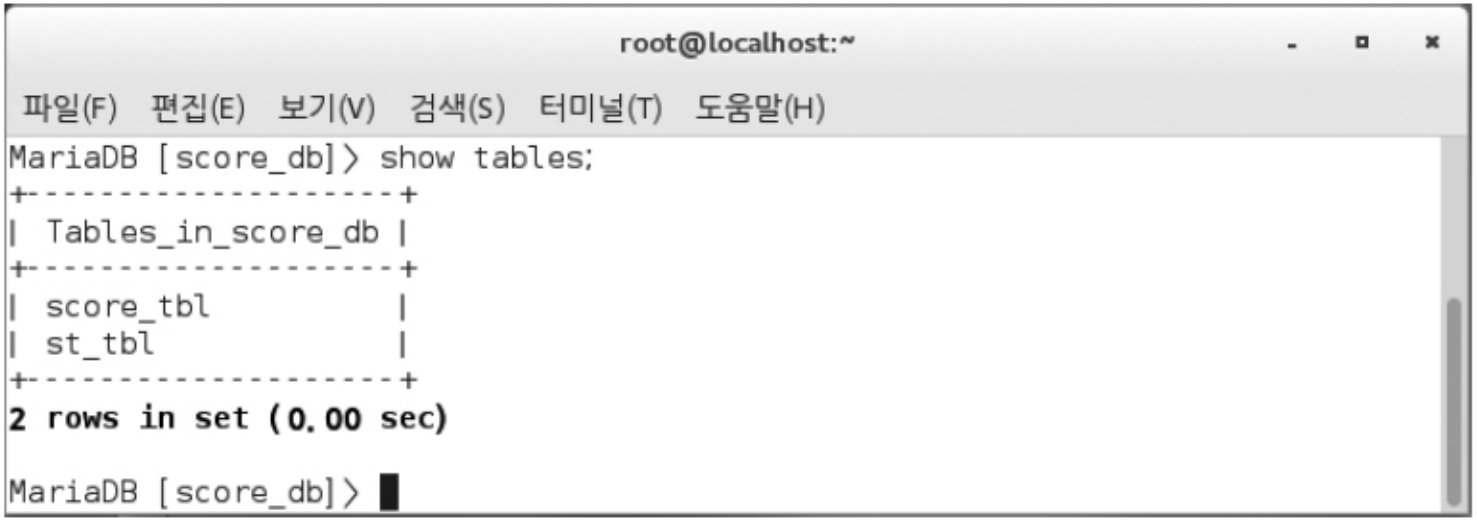
```
MariaDB[score_db]> create table score_tbl (SNO int, LINUX int, DBMS int);
```



[그림 13-10] 성적 테이블 score_tbl 생성

- **Step 04** | 지금까지 학생 테이블 st_tbl과 성적 테이블 score_tbl을 생성하였습니다. 생성한 테이블 2개가 존재하는지에 대해 확인합니다.

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

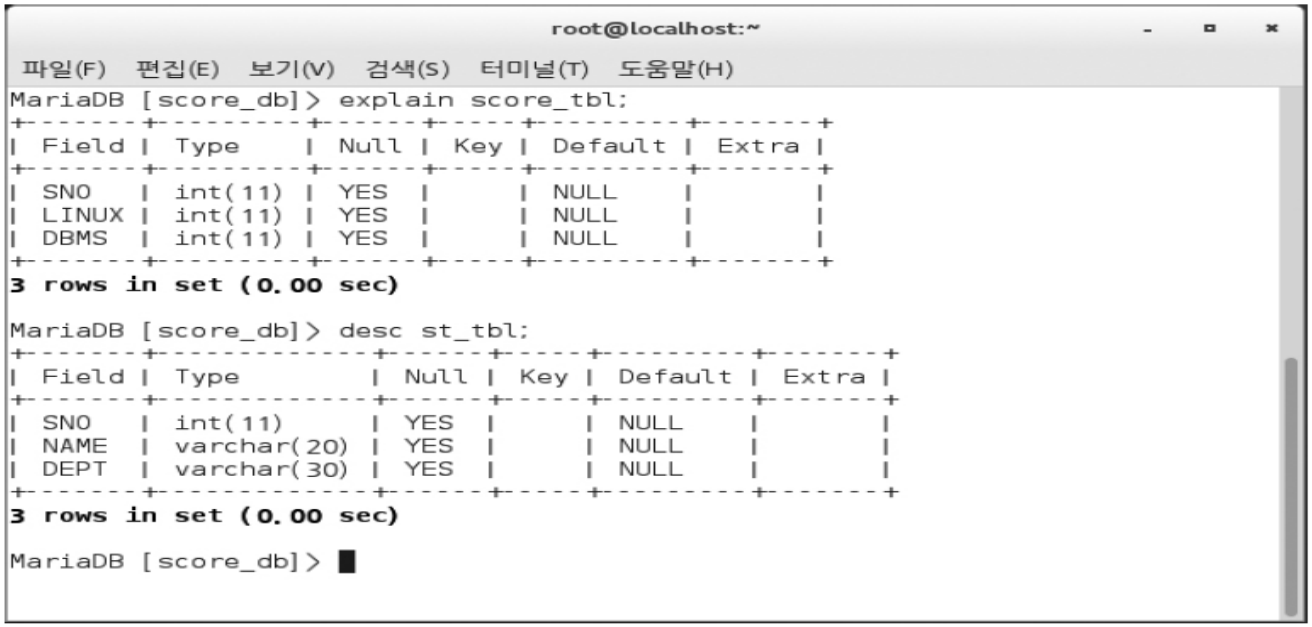


[그림 13-11] 데이터베이스 score_db에 존재하는 테이블 목록 출력

테이블의 구조 확인

- **Step 05** | 앞에서 생성한 학생 테이블 st_tbl과 성적 테이블 score_tbl의 구조를 확인하기 위해서는 explain 명령 또는 desc 명령을 사용합니다.

```
MariaDB[score_db]> explain score_tbl;   또는  desc score_tbl;
```



[그림 13-12] 테이블의 구조 확인

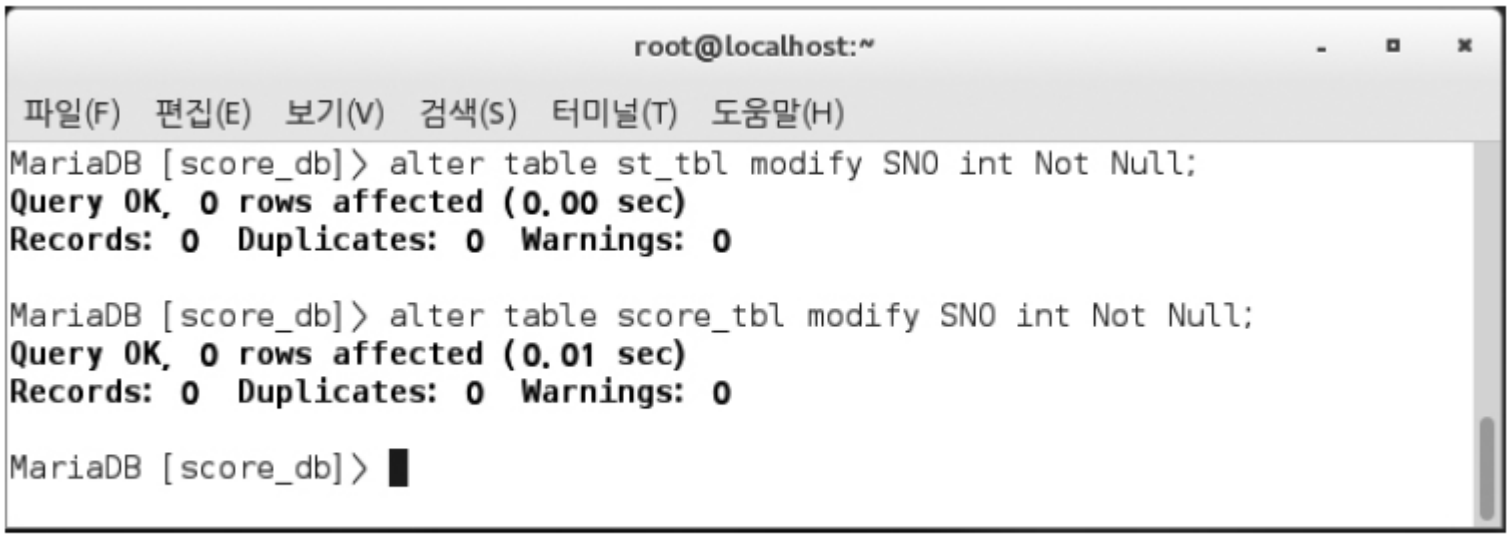
테이블의 필드에 기본 키 추가

테이블에는 여러 개의 필드가 생성되어 존재하게 됩니다. 이중에서 기본 키는 중복되지 않는 고유의 구분자를 의미합니다. 즉 학생들이 대학교에 입학하게 되면 학번이 부여되듯이 중복되지 않는 고유의 킷값을 의미합니다. 앞에서 생성한 학생 테이블과 성적 테이블의 학번 필드에 기본 키를 설정하도록 하겠습니다.

예제 13-7

- **Step 01** | 기본 키를 설정하기에 앞서 기본 키로 설정할 필드에 Null 값을 허용하지 않도록 하기 위해 st_tbl과 score_tbl의 SNO 필드에 다음과 같이 SQL문을 수행합니다.

```
MariaDB[score_db]> alter table st_tbl modify SNO int Not Null;
MariaDB[score_db]> alter table score_tbl modify SNO int Not Null;
```

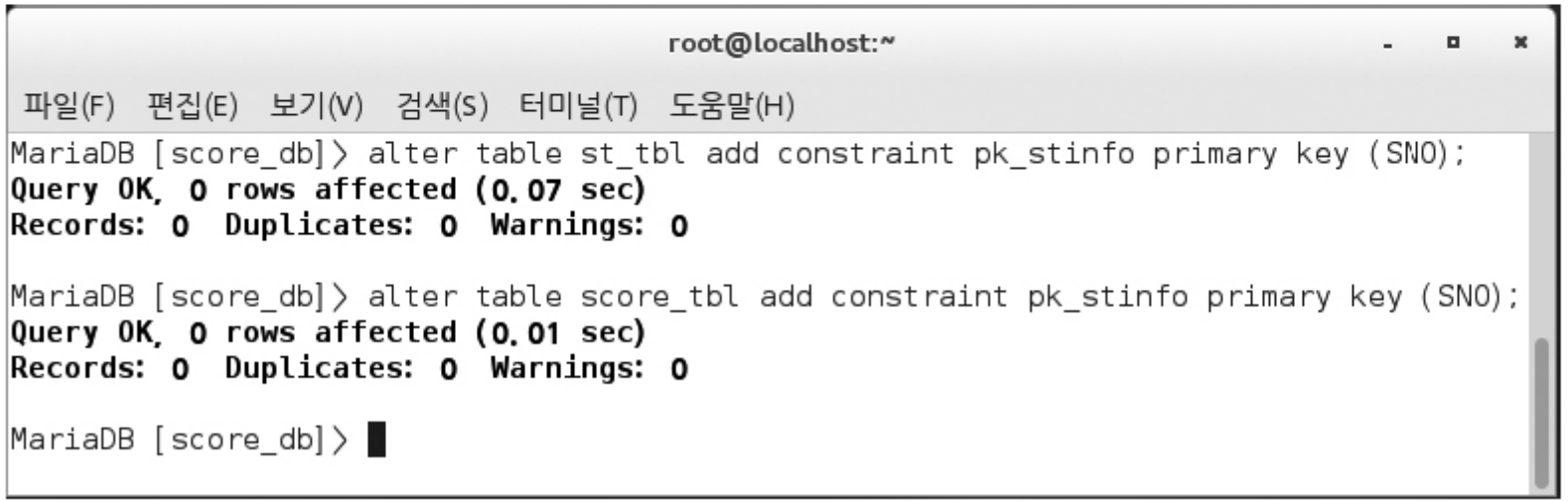


[그림 13-13] SNO 필드에 Null 값을 허용하지 않도록 설정

Null 값이라 함은 생략할 수 있음을 의미하기 때문에 기본 키에 저장되는 값은 생략하게 되면 중복되는 사례가 발생하게 되므로 생략해서는 안 됩니다. alter 명령은 테이블의 속성을 변경할 때 사용하는 명령이며 modify 이하에 있는 필드에 대해 속성을 변경할 때 사용됩니다.

- **Step 02** | SNO 필드에 Null 값을 허용하지 않도록 수행했으므로 이제 SNO 필드를 기본 키로 설정하도록 하겠습니다.

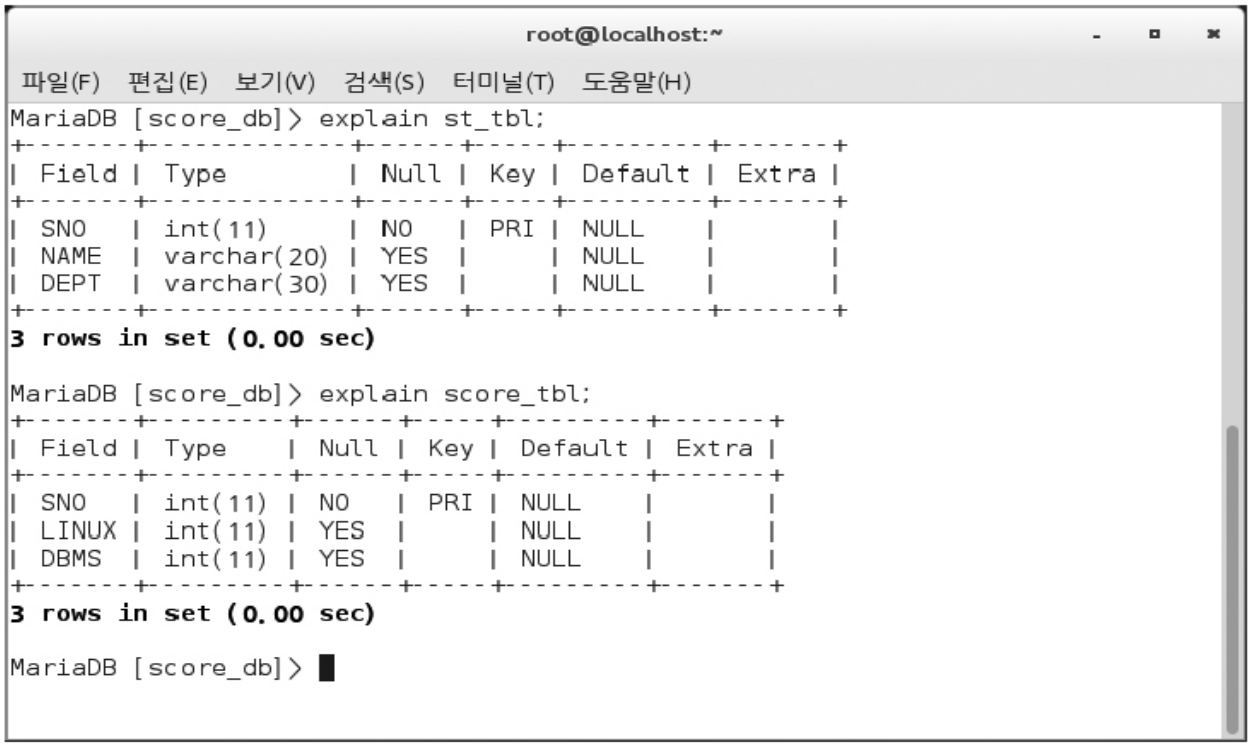
```
MariaDB[score_db]> alter table st_tbl add constraint pk_stinfo primary key (SNO);

MariaDB[score_db]> alter table score_tbl add constraint pk_stinfo primary key
(SNO);
```



[그림 13-14] 2개 테이블의 SNO 필드에 기본 키 설정

- **Step 03** | 학생 테이블 st_tbl과 성적 테이블 score_tbl의 SNO 필드의 속성을 각각 변경하였으니 테이블 구조를 explain 명령으로 확인합니다.

```
MariaDB[score_db]> explain st_tbl;   또는  desc st_tbl;
```



[그림 13-15] 2개의 테이블의 구조에서 기본 키 설정 확인

레코드 삽입과 삭제

레코드 삽입

레코드Record라 함은 한 사람에 대한 학번, 이름, 학과 등 여러 개의 필드로 구성되어 있는 하나의 행 단위를 의미합니다. 레코드 삽입은 학생 테이블 st_tbl과 성적 테이블 score_tbl에 대해 다음 표와 같이 각각 2개의 레코드씩 삽입하도록 하겠습니다.

[표 13-2] 학생 테이블(st_tbl)

SNO	NAME	DEPT
201701	홍길동	컴퓨터학과
201503	김가을	역사학과

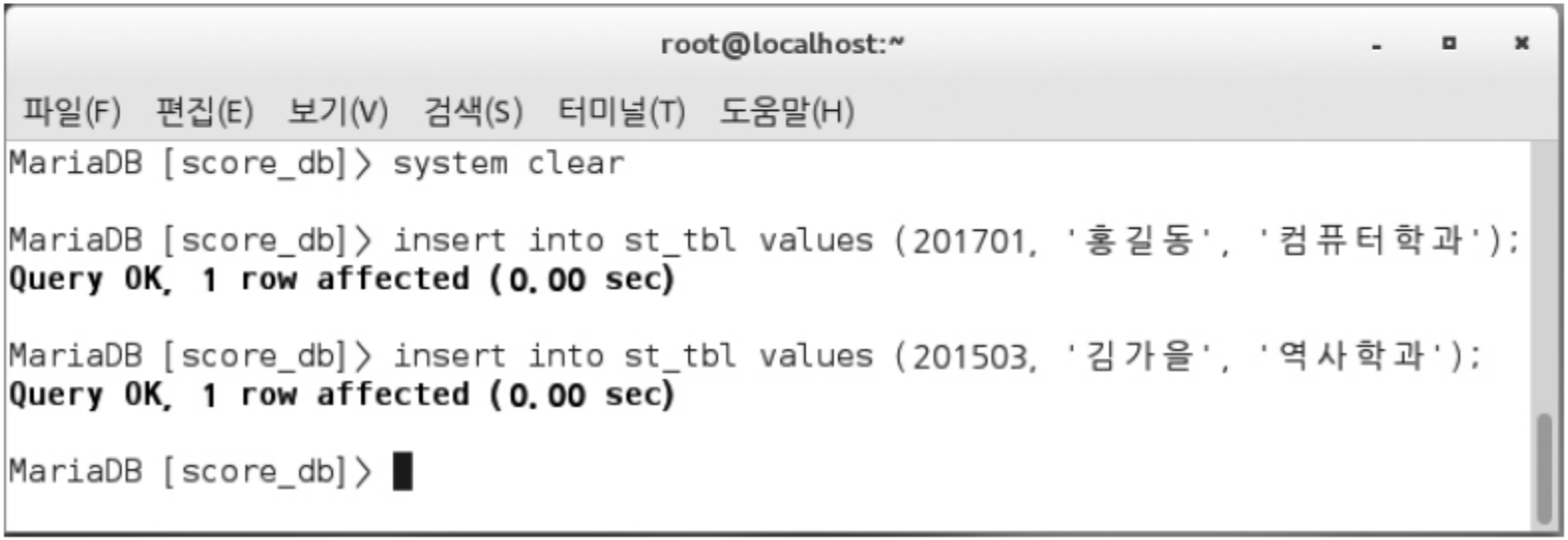
[표 13-3] 성적 테이블(score_tbl)

SNO	LINUX	DBMS
201701	98	85
201503	79	90

예제 13-8

- **Step 01** | 먼저 학생 테이블(st_tbl)에 레코드를 삽입하기 위해 다음과 같이 SQL문을 수행합니다.

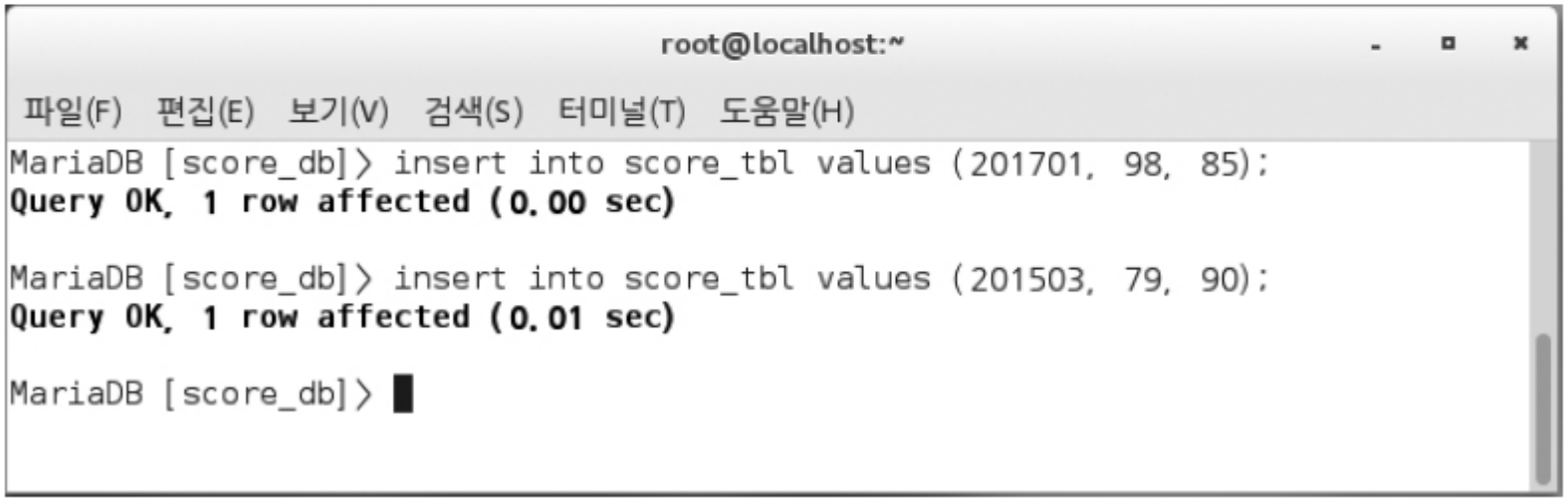
```
MariaDB[score_db]> insert into st_tbl values (201701, '홍길동', '컴퓨터학과');
MariaDB[score_db]> insert into st_tbl values (201503, '김가을', '역사학과');
```



[그림 13-16] 학생 테이블(st_tbl)에 2개의 레코드 삽입

- **Step 02** | 이번에는 성적 테이블(score_tbl)에 레코드를 삽입하기 위해 다음과 같이 SQL문을 수행합니다.

```
MariaDB[score_db]> insert into score_tbl values (201701, 98, 85);
MariaDB[score_db]> insert into score_tbl values (201503, 79, 90);
```



[그림 13-17] 성적 테이블(score_tbl)에 2개의 레코드 삽입

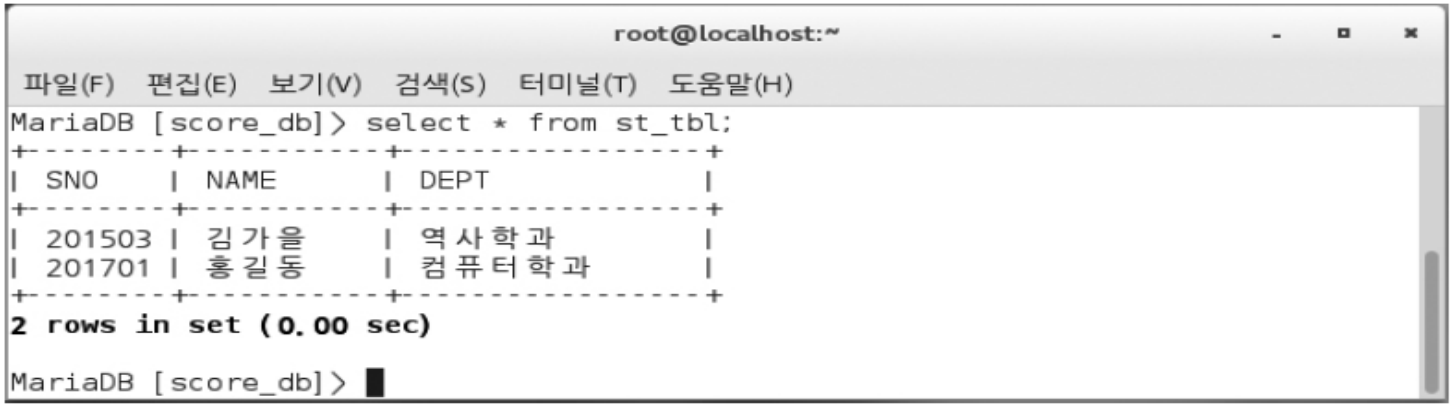
레코드 검색

앞에서 테이블에 삽입한 레코드가 정상적으로 입력되었는지를 검색하기 위해서 사용되는 명령은 select 명령입니다. 다음 예제를 통해 살펴보도록 하겠습니다.

예제 13-9

- **Step 01** | 먼저 학생 테이블(st_tbl)에 저장된 레코드를 검색하기 위해 다음과 같이 SQL문을 수행합니다.

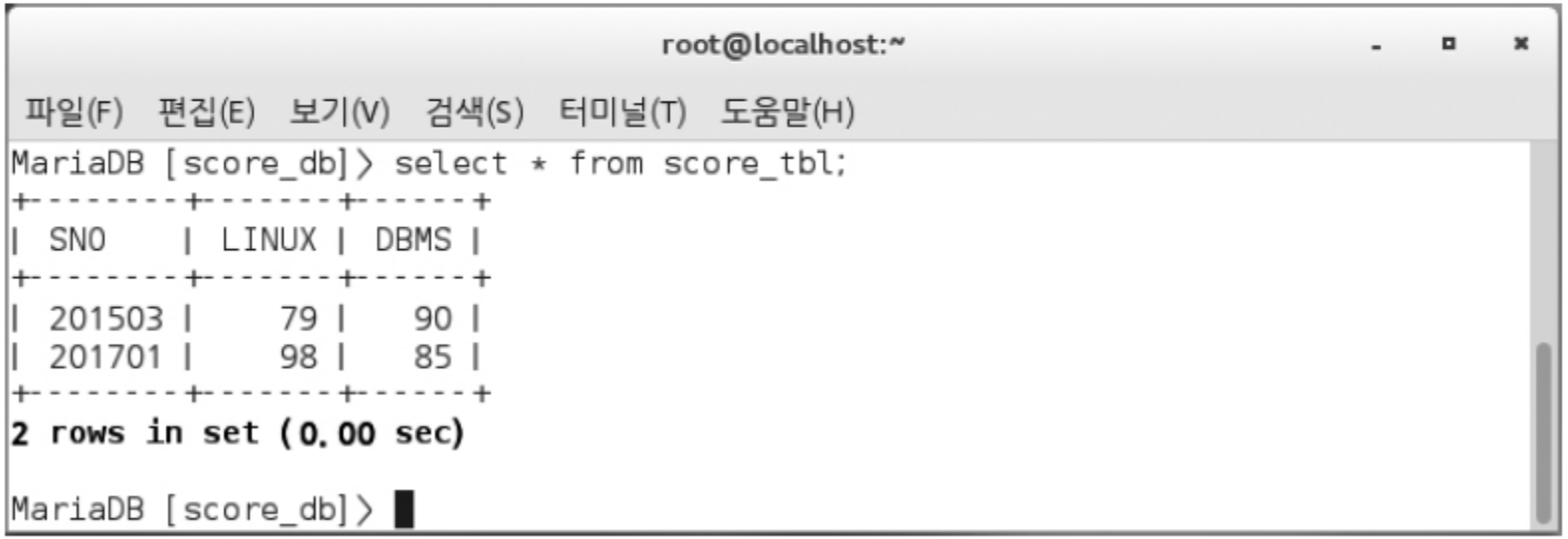
```
MariaDB[score_db]> select * from st_tbl;
```



[그림 13-18] 학생 테이블(st_tbl)에 삽입된 레코드 검색

- **Step 02** | 이번에는 성적 테이블(score_tbl)에 저장된 레코드를 검색하기 위해 다음과 같이 SQL문을 수행합니다.

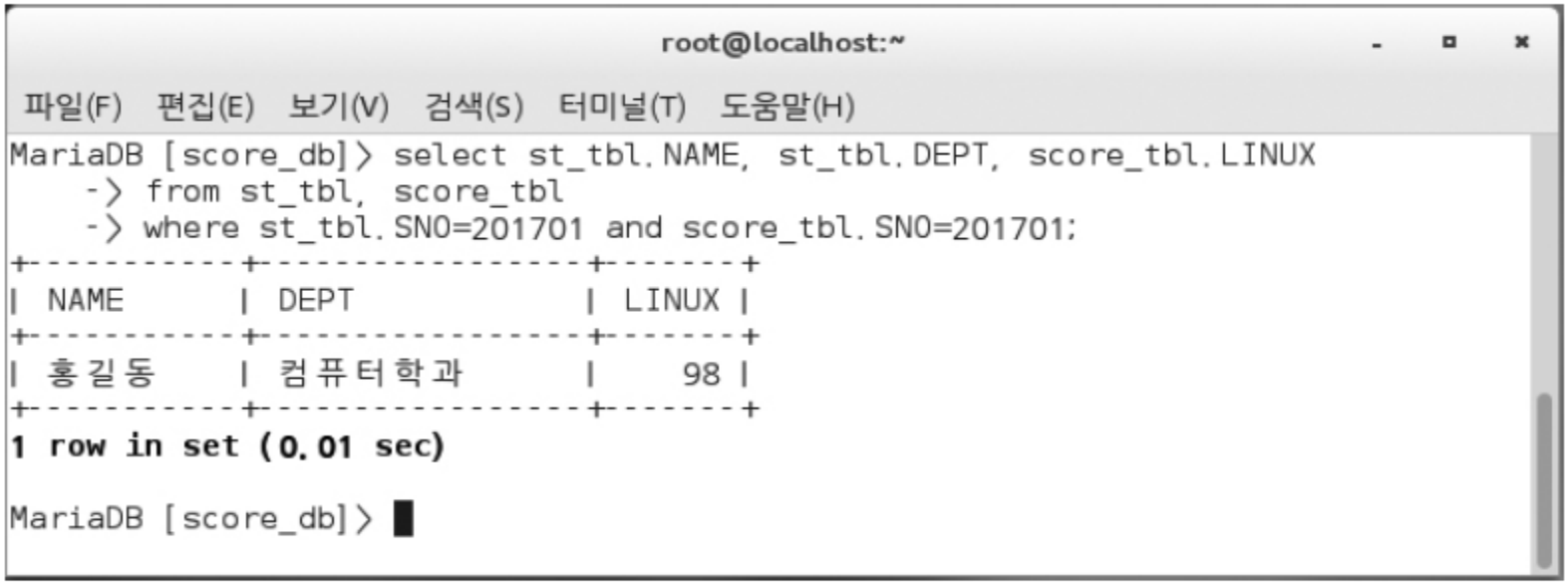
```
MariaDB[score_db]> select * from score_tbl;
```



[그림 13-19] 성적 테이블(score_tbl)에 삽입된 레코드 검색

- **Step 03** | 학번이 201701인 학생의 이름과 학과, 리눅스 성적을 한꺼번에 검색하기 위해 다음과 같이 SQL문을 수행합니다.

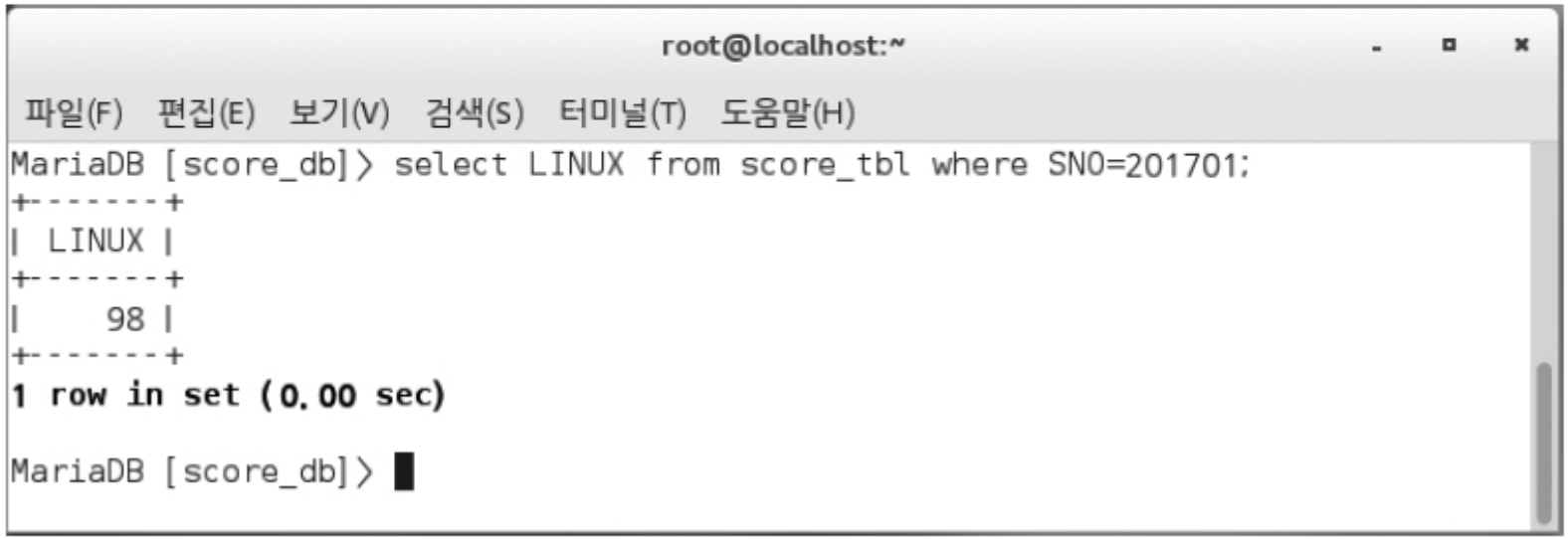
```
MariaDB[score_db]> select st_tbl.NAME, st_tbl.DEPT, score_tbl.LINUX
  -> from st_tbl, score_tbl
  -> where st_tbl.SNO=201701 and score_tbl.SNO=201701;
```



[그림 13-20] 학생 테이블과 성적 테이블의 조인검색

- **Step 04** | 이번에는 좀 더 간단하게 레코드를 검색하기 위해 학번이 201701인 학생의 리눅스 성적만을 검색하도록 하겠습니다.

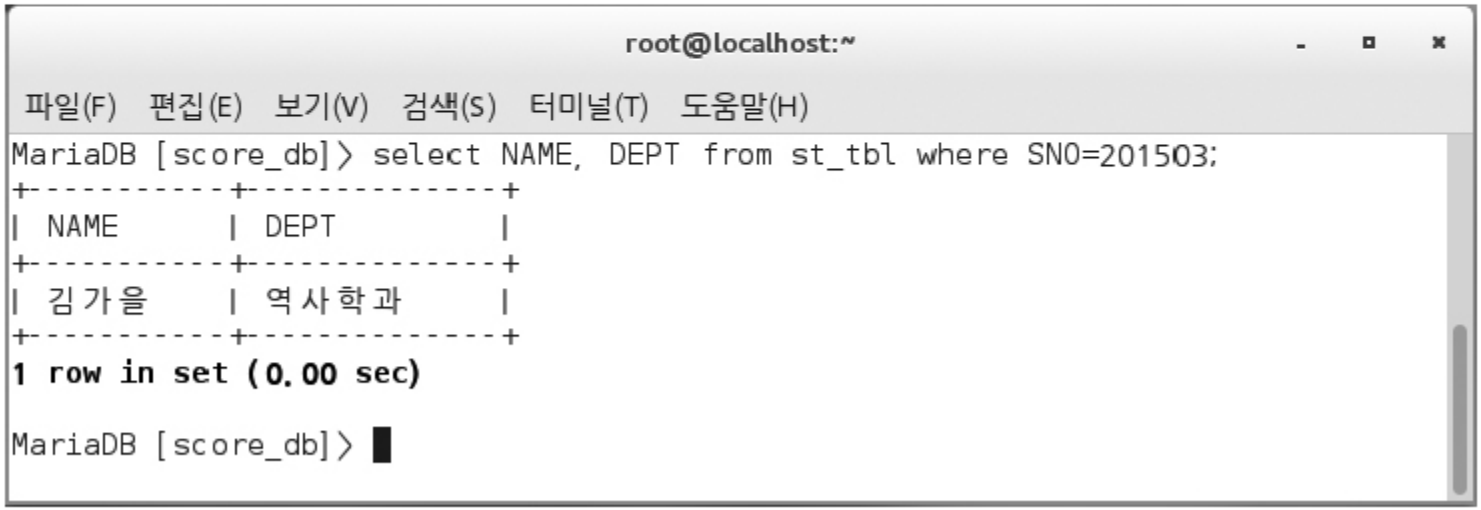
```
MariaDB[score_db]> select LINUX from score_tbl where SNO=201701;
```



[그림 13-21] 학번 201701 학생의 리눅스 성적 검색

- **Step 05** | 여기서는 학번이 201503인 학생의 이름과 학과를 검색하기 위해 다음과 같이 SQL 문을 수행합니다.

```
MariaDB[score_db]> select NAME, DEPT from st_tbl where SNO=201503;
```



[그림 13-22] 학번 201503 학생의 이름과 학과 검색

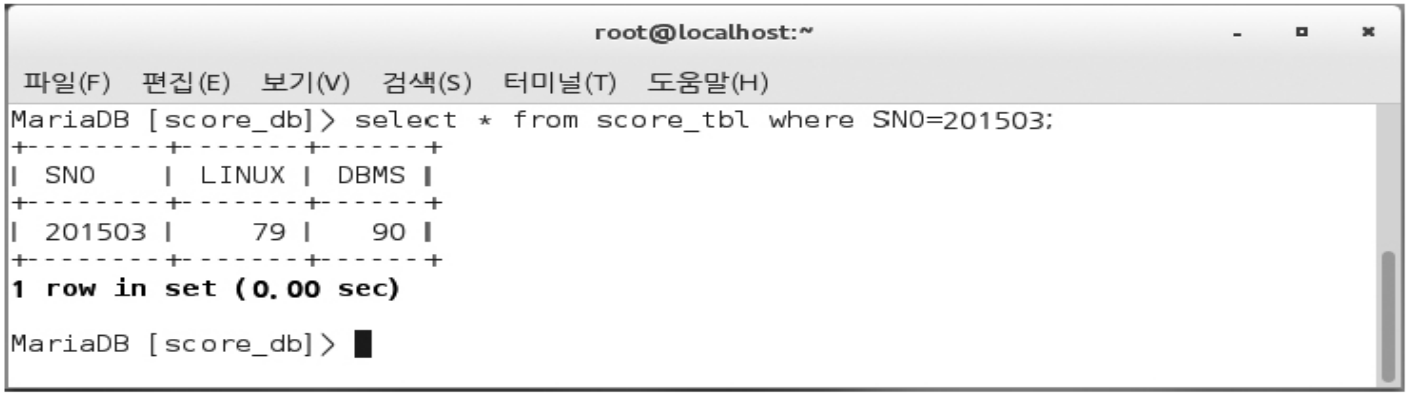
데이터 수정하기

이번에는 테이블에 저장되어 있는 데이터를 수정하는 방법에 대해 살펴보도록 하겠습니다. 데이터를 수정할 때는 update 명령을 사용합니다. 다음 예제에서 자세히 살펴보도록 하겠습니다.

예제 13-10

- **Step 01** | 먼저 성적 테이블(st_tbl)에 저장된 레코드 중에서 학번이 201503인 학생의 과목성적을 확인하기 위해 다음과 같이 SQL문을 수행합니다.

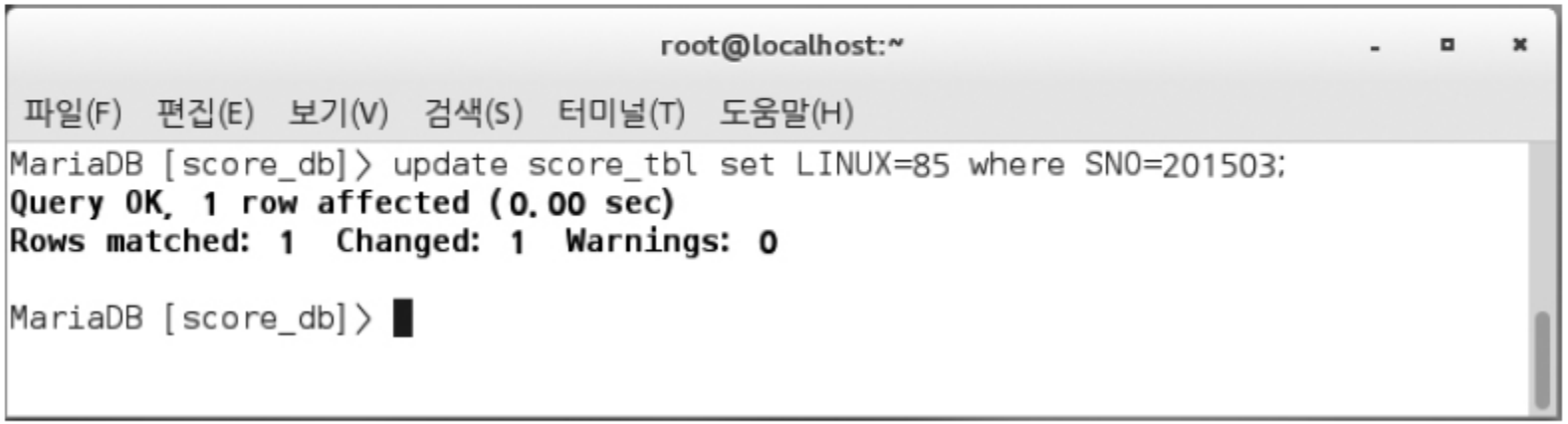
```
MariaDB[score_db]> select * from score_tbl where SNO=201503;
```



[그림 13-23] 학번 201503 학생의 성적현황

- **Step 02** | 성적 테이블(st_tbl)에 저장된 레코드 중에서 학번이 201503인 학생의 리눅스 과목 성적을 79점에서 85점으로 수정하려면 다음과 같이 SQL문을 수행합니다.

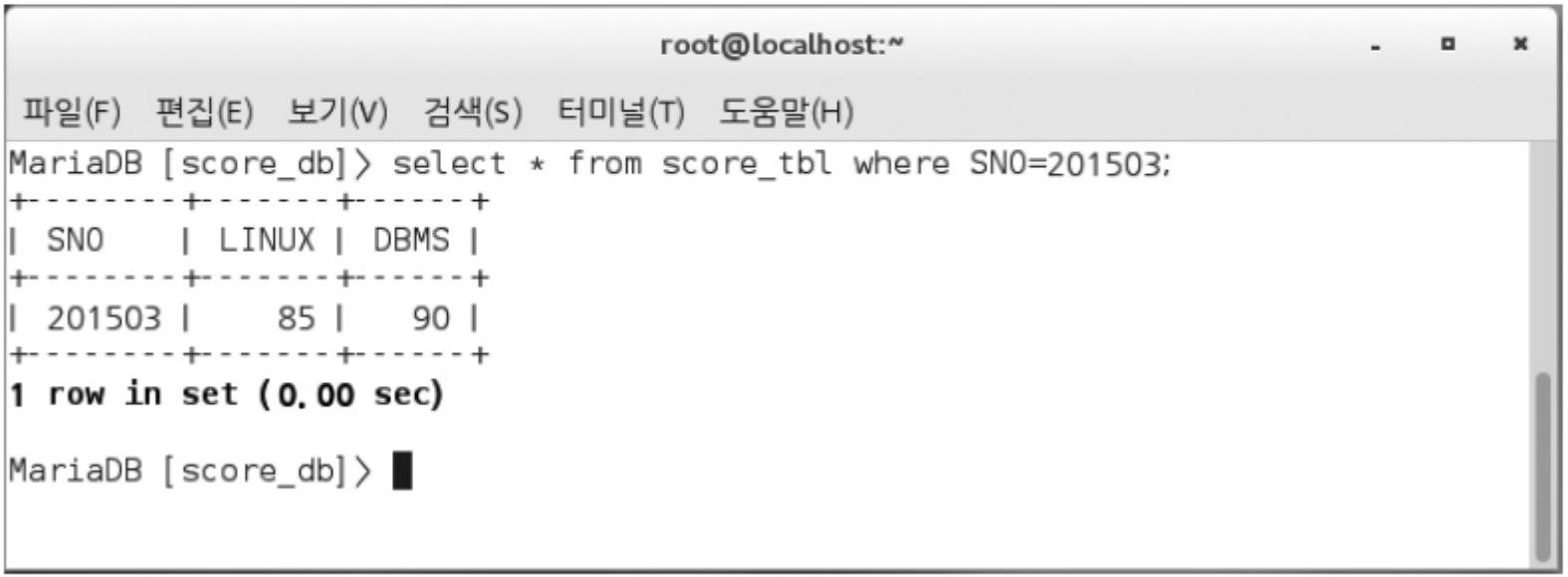
```
MariaDB[score_db]> update score_tbl set LINUX=85 where SNO=201503;
```



[그림 13-24] 학번 201503 학생의 리눅스 과목성적 수정

- **Step 03** | 학번이 201503인 학생의 수정한 리눅스 과목성적이 제대로 반영되었는지를 확인하기 위해 다음과 같이 SQL문을 수행합니다.

```
MariaDB[score_db]> select * from score_tbl where SNO=201503;
```

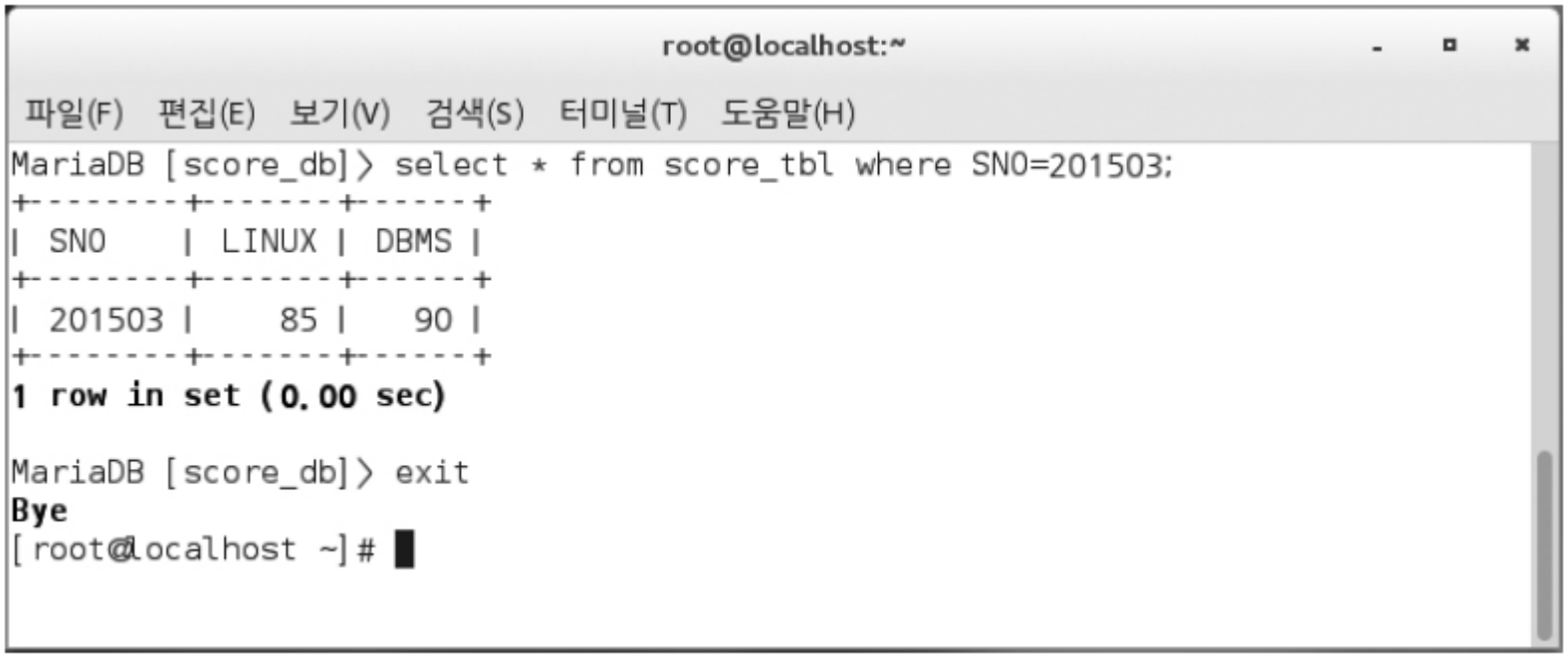


[그림 13-25] 학번 201503 학생의 수정된 성적현황

MariaDB 서버 접속종료

MariaDB 서버에 접속된 상태를 종료하려면 exit 명령을 사용하여 종료하면 됩니다.

```
MariaDB[score_db]> exit
```



[그림 13-26] MariaDB 접속종료

실습 13 다음 항목에서 주어진 지시사항을 수행하시오.

1. MariaDB 서버 접속하기
2. 새로운 데이터베이스 test13_db 생성하기
3. test13_db 선택하기
4. 테이블 addr_tbl 생성하기 → 아래 주소 테이블 참조
5. 테이블 addr_tbl의 구조 확인하기
6. 레코드 삽입하기 → 아래 주소 테이블 참조
7. 테이블에 삽입된 모든 레코드 검색하기
8. MariaDB 서버 접속 종료하기

■ 주소 테이블(addr_tbl)

NO(int)	NAME(varchar)	ADDR(varchar)
1001	이순신	서울특별시 중구 신당동 1850
1002	을지문덕	부산광역시 금정구 부산대학로 15

Chapter 13

최상의 노력에 따른 인고의 가치는 반드시 증명될 수 있습니다!

Thank You