4장 데이터베이스와 테이블

1 MySQL 데이터베이스

2 테이블의 생성

3 테이블 데이터 처리(삽입, 변경, 삭제) 작업

4 보안 설정

4.1 MySQL 데이터베이스 생성

4.4.1 데이터베이스 및 테이블 만들기

[실습 따라하기]

① MySQL접속

```
C:₩> mysql -u root -p
```

② MySQL 새로운 데이터베이스("haksa") 생성

```
mysql> create database haksa;
Query OK, 1 row affected (0.00 sec)
```

③ 생성된 Database 확인

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| haksa              |
| information_schema |
| my_database        |
| mysql              |
| performance_schema |
| sakila             |
| sys                |
| world              |
+--------------------+
8 rows in set (0.01 sec)
mysql>
```

④ 생성된 Database 사용하기 위해 데이터베이스 변경

```
mysql> use haksa;
Database changed
mysql>
```

2) 인사테이블("insa") 생성 및 데이터 입력

[실습 따라하기]

```
mysql> create table insa(
    -> bunho int(1) auto_increment,
    -> name char(8) not null,
    -> e_name char(4),
    -> town char(6) not null,
    -> primary key(bunho)
    -> );
Query OK, 0 rows affected, 1 warning (0.75 sec)

mysql> insert into insa values('1','홍길동', 'Hong', '서울');
mysql> insert into insa values('2','제갈공명', 'Je', '부산');
mysql> insert into insa values('3','순자', 'Soon', '대구');
mysql> insert into insa values(4,'이순신', 'Lee',  '대전');
mysql> insert into insa values(null,'연개소문', 'Yean', '서울');
mysql> insert into insa values(null,'강감찬', NULL, '부산');
mysql> insert into insa values(null,'최영', '', '광주');
mysql> insert into insa(name,e_name,town) values('계백', 'Gae', '서울');
```

데이터 입력 시 e_name은 null값을 허용하기 때문에
강감찬의 경우 NULL로 최영의 경우는 공백(space)로 출력

4.1.2 Commit/Rollback 작업

1. Commit : 변경된 데이터를 데이터베이스에 적용시킨다.
2. Rollback : 변경된 데이터를 취소. 직전에 Commit이 수행된 시점까지 취소

①"INSA" 테이블 질의

```
mysql> select * from insa;
+-------+----------+--------+------+
| bunho | name     | e_name | town |
+-------+----------+--------+------+
|     1 | 홍길동   | Hong   | 서울 |
|     2 | 제갈공명 | Je     | 부산 |
|     3 | 순자     | Soon   | 대구 |
|     4 | 이순신   | Lee    | 대전 |
|     5 | 연개소문 | Yean   | 서울 |
|     6 | 강감찬   | NULL   | 부산 |
|     7 | 최영     |        | 광주 |
|     8 | 계백     | Gae    | 서울 |
+-------+----------+--------+------+
8 rows in set (0.00 sec)
```

표 4.3 INSA 테이블

BUNHO	NAME	E_NAME	TOWN
1	홍길동	Hong	서울
2	제갈공명	Je	부산
3	순자	Soon	대구
4	이순신	Lee	대전
5	연개소문	Yean	서울
6	강감찬		부산
7	최영		광주
8	계백	Gae	서울

② 주의사항 : MySQL은 명령어를 실행하면 자동(Default)으로 Commit을 하게 되어 있다. 우선 AutoCommit를 하지 않도록 한다.

```
mysql> set autocommit = 0;
```

③ "INSA"테이블의 내용 변경 : 번호 4번 도시(TOWN)을 한산도로 변경

```
mysql> update insa
    -> set town = '한산도'
    -> where bunho = 4;
```

- 변경된 "INSA"테이블 질의

```
mysql> select * from insa;
+-------+----------+--------+--------+
| bunho | name     | e_name | town   |
+-------+----------+--------+--------+
|     1 | 홍길동   | Hong   | 서울   |
|     2 | 제갈공명 | Je     | 부산   |
|     3 | 순자     | Soon   | 대구   |
|     4 | 이순신   | Lee    | 한산도 |
|     5 | 연개소문 | Yean   | 서울   |
|     6 | 강감찬   | NULL   | 부산   |
|     7 | 최영     |        | 광주   |
|     8 | 계백     | Gae    | 서울   |
+-------+----------+--------+--------+
8 rows in set (0.00 sec)
```

④ 변경된 데이터 복구작업 : Rollback

```
mysql> rollback;
Query OK, 0 rows affected (0.03 sec)
```

⑤ "INSA" 테이블 질의

```
mysql> select * from insa;
+-------+----------+--------+------+
| bunho | name     | e_name | town |
+-------+----------+--------+------+
|     1 | 홍길동   | Hong   | 서울 |
|     2 | 제갈공명 | Je     | 부산 |
|     3 | 순자     | Soon   | 대구 |
|     4 | 이순신   | Lee    | 대전 |
|     5 | 연개소문 | Yean   | 서울 |
|     6 | 강감찬   | NULL   | 부산 |
|     7 | 최영     |        | 광주 |
|     8 | 계백     | Gae    | 서울 |
+-------+----------+--------+------+
8 rows in set (0.00 sec)
```

⑥ "INSA" 테이블 내용 변경 : TOWN이 부산인 데이터를 "인천"로 변경

```
mysql> update insa
    -> set town = '인천'
    -> where town = '부산';
```

⑦ “INSA” 테이블 내용 데이터베이스에 저장 : Commit

```
mysql> commit;
Query OK, 0 rows affected (0.00 sec)
```

⑧ 변경된 데이터 복구작업 : Rollback(복구가 되지 않음)

```
mysql> rollback;
Query OK, 0 rows affected (0.00 sec).
```

⑨ “INSA” 테이블 질의

```
mysql> select * from insa;
+-------+----------+--------+------+
| bunho | name     | e_name | town |
+-------+----------+--------+------+
|     1 | 홍길동   | Hong   | 서울 |
|     2 | 제갈공명 | Je     | 인천 |
|     3 | 순자     | Soon   | 대구 |
|     4 | 이순신   | Lee    | 대전 |
|     5 | 연개소문 | Yean   | 서울 |
|     6 | 강감찬   | NULL   | 인천 |
|     7 | 최영     |        | 광주 |
|     8 | 계백     | Gae    | 서울 |
+-------+----------+--------+------+
8 rows in set (0.00 sec)
```

4.1.3 Savepoint/Truncate 작업

1. Savepoint는 변경된 지점(저장점)의 위치를 저장
2. Savepoint로 저장점을 저장하고 INSERT, DELETE, UPDATE작업을 수행 후 Rollback to 저장점을 수행하면 그 위치까지 다시 복구

① "INSA" 테이블 변경 작업 : 번호 2의 도시(TOWN)을 "여수"로 변경

```
mysql> update insa
    -> set town = '여수'
    -> where bunho = 2;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

② Savepoint "AA" 지정

```
mysql> savepoint aa;
Query OK, 0 rows affected (0.00 sec)
```

③ 번호 3번 행 삭제 : DELETE 작업

```
mysql> delete from insa
    -> where bunho = 3;
Query OK, 1 row affected (0.00 sec).
```

④ “INSA” 테이블 질의

```
mysql> select * from insa;
+-------+----------+--------+------+
| bunho | name     | e_name | town |
+-------+----------+--------+------+
|     1 | 홍길동   | Hong   | 서울 |
|     2 | 제갈공명 | Je     | 여수 |
|     4 | 이순신   | Lee    | 대전 |
|     5 | 연개소문 | Yean   | 서울 |
|     6 | 강감찬   | NULL   | 인천 |
|     7 | 최영     |        | 광주 |
|     8 | 계백     | Gae    | 서울 |
+-------+----------+--------+------+
7 rows in set (0.00 sec)
```

⑤ “INSA” 테이블 Savepont “AA”까지 복구

```
mysql> rollback to aa;
Query OK, 0 rows affected (0.00 sec)
```

① 작업을 수행한 후 Savepont “AA”를 지정했기 때문에 ①에서 변경 작업한 내용은 복구되지 않고, ③ 작업 이후만 복구

⑥ “INSA” 테이블 질의

```
mysql> select * from insa;
+-------+----------+--------+------+
| bunho | name     | e_name | town |
+-------+----------+--------+------+
|     1 | 홍길동   | Hong   | 서울 |
|     2 | 제갈공명 | Je     | 여수 |
|     3 | 순자     | Soon   | 대구 |
|     4 | 이순신   | Lee    | 대전 |
|     5 | 연개소문 | Yean   | 서울 |
```

⑦ Truncate 작업 : “INSA”테이블의 삭제 처리 (모든 행이 삭제 처리됨)

```
mysql> truncate table insa;
Query OK, 3 rows affected (0.08 sec)
```

⑧ Truncate 작업 후 “INSA”테이블 복구 (복구가 되지 않음)

```
mysql> rollback;
Query OK, 0 rows affected (0.00 sec).
```

⑨ “INSA” 테이블 질의

```
mysql> select * from insa;
Empty set (0.00 sec)
```

Truncate 작업의 “INSA”테이블 삭제 처리시 복구가 되지 않는다.

⑩ “INSA” 테이블 칼럼정보 “INSA” 테이블

```
mysql> desc insa;
+--------+---------+------+-----+---------+----------------+
| Field  | Type    | Null | Key | Default | Extra          |
+--------+---------+------+-----+---------+----------------+
| bunho  | int(1)  | NO   | PRI | NULL    | auto_increment |
| name   | char(8) | NO   |     | NULL    |                |
| e_name | char(4) | YES  |     | NULL    |                |
| town   | char(6) | NO   |     | NULL    |                |
+--------+---------+------+-----+---------+----------------+
4 rows in set (0.03 sec)
```

4.2 SQL 데이터형(Data Type)

SQL 명령문의 그룹

① DDL : 데이터 정의 언어(Data Definition Language)
- 데이터베이스 객체인 테이블, 인덱스, 뷰 등의 구성에 영향을 주는 명령문
- DBA(데이터베이스관리자)만이 권한을 갖는다.
- CREATE TABLE, CREATE INDEX, CREATE VIEW, DROP TABLE, DROP INDEX, ALTER TABLE, RENAME

② DML : 데이터 조작 언어(Data Manipulation Language)
-테이블의 내용을 변경하고 질의하는 명령문
- SELECT, INSERT, UPDATE, DELETE, COMMIT, ROLLBACK, SAVEPOINT, LOCK

③ DCL : 데이터 제어 언어(Data Control Language)
- 유저관리와 데이터 보호와 관련된 명령문
- GRANT, ROVOKE, CONNECT, DISCONNECT

4.2.1 SQL 데이터형

1) 숫자 데이터형

① 정수 데이터형(INT) : int 데이터형은 정수형 데이터(0, 음수, 양수)를 저장
ex) INT(n)

② 실수 데이터형(FLOAT) : 실수 데이터형은 소수점을 포함하여 값을 저장
ex) FLOAT(N, M)

2) 문자 데이터형(CHAR, VARCHAR, BLOB)

① CHAR 데이터형

-1바이트에서 255바이트까지의 고정 길이 문자열을 저장
-정의된 저장공간보다 입력 데이터가 짧으면 나머지 공간은 공백(SPACE)으로
-정의된 길이보다 입력 데이터가 길면 길이에 맞게 잘린 데이터가 입력
-테이블 생성시 저장할 데이터의 최대크기로 정의해야만 데이터의 손실 방지

② VARCHAR 데이터형

-정의된 저장공간보다 긴 문자열이 입력되면 VARCHAR에서는 에러 값을 리턴
-최대로 정의할 수 있는 데이터의 길이는 255바이트까지 저장
-메모 등의 다양한 길이의 데이터에 적절하고, 가변적인 길이의 문자열을 저장
-VARCHAR가 CHAR보다 검색 속도가 훨씬 느리다.

③ BLOB ,TEXT 데이터형
-BLOB와 TEXT는 65,535 이상의 거대한 텍스트 데이터를 저장할 때 사용
-BLOB는 검색시 대소문자를 구분
-TEXT는 대소문자의 구분이 없이 검색

3) 날짜 데이터형

-날짜 및 시간 데이터를 저장하기 위해서 Date 데이터형을 제공
-SYSDATE이라는 함수를 사용해서 현재 OS의 날짜를 조회

```
mysql> select now();
+---------------------+
| now()               |
+---------------------+
| 2019-01-31 16:42:34 |
+---------------------+
1 row in set (0.00 sec)
```

-select는 산술 계산의 결과나 날짜 등을 볼 수 있다.
-위의 실행결과는 현재 오늘 시스템 날짜가 2019년 01월 31일인 경우
-Date형은 B. C. 4712년 1월 1일~A. D. 9999년 12월 31일까지 범위 값 저장

4) 바이너리(binary) 데이터형

-음성, 화상(이미지), 동영상과 같은 데이터를 저장하기 위해서 바이너리 데이터형으로 RAW와 LONG RAW 데이터형을 사용
-제약점으로는 내장함수를 사용할 수 없다.

① RAW 데이터형 : 이진형 데이터를 255바이트까지 사용, 저장 공간의 제한점 때문에 많이 사용하지 않는다.
② LONG RAW 데이터형 : 이진형 데이터를 2GB까지 수용
③ BLOB 데이터형 : 이진형 데이터를 4GB까지 수용

4.2.2 NULL 값

-NULL 값은 “값이 알려져 있지 않다” 또는 “값이 존재하지 않는다”라는 의미
-NULL 값은 다른 NULL 값과 결코 일치하지 않는다.
-NOT NULL의 의미는 모든 행에서 해당 열은 특정한 값으로 채워져 있어야 한다는 것을 의미
- NOT NULL로 정의된 칼럼은 NULL값을 사용할 수 없다.

4.3 학사관리 예제 만들기

4.3.1 데이터베이스 및 사용자 계정 생성

① MySQL 접속하기

```
C:₩Users₩solero>mysql -u root -p
Enter password: *******
Welcome to the MySQL monitor.  Commands end with ; or ₩g.
Your MySQL connection id is 13
Server version: 8.4.0 MySQL Community Server - GPL
Copyright (c) 2000, 2024, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '₩h' for help. Type '₩c' to clear the current input statement.
mysql>
```

② MySQL 새로운 데이터베이스 생성

```
create database haksa;
```

③ 생성된 Database 확인

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| haksa              |
| information_schema |
| my_database        |
| mysql              |
| performance_schema |
| sakila             |
| sys                |
| world              |
+--------------------+
8 rows in set (0.01 sec)
```

④ 사용자 생성 (sol)

```
mysql> create user sol@localhost identified by 'sol123';
```

⑤ 권한부여

```
mysql> grant all privileges on haksa.* to sol@localhost;
```

새로 생성된 사용자(sol)와 암호(sol1234)로
새로 생성된 데이터베이스(haksa)로 로그인 하기

```
C:₩Users₩solero> mysql -u sol -p haksa
Enter password: ******
Welcome to the MySQL monitor.  Commands end with ; or ₩g.
Your MySQL connection id is 14
Server version: 8.4.0 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '₩h' for help. Type '₩c' to clear the current input statement.

mysql>
```

4.3.2 테이블 생성

-테이블 생성 방법 : 명령어(Command) 방식, MySQL Query Browser 방식
-학사("haksa")데이터베이스는 전체 9개의 테이블로 구성

1. 학사관리 테이블 생성(SQL 명령어 사용)

```
# 학과테이블
create table department(
dept_code int(2) Not null,          #학과번호
dept_name char(30) Not null,        #학과명
dept_ename varchar(50),             #학과영문이름
create_date date default null,      #학과생성날짜
primary key (dept_code)
)engine = innoDB;
```

```
# 학적(학생신상)테이블
create table student(
stu_no char(10) Not null,           #학번
stu_name char(10) Not null,         #학생이름
stu_ename varchar(30),              #영문이름
dept_code int(2) Not null,          #학과코드
grade int(1) Not null,              #학년
class int (1) Not null,             #반
juya char(2),                       #주야구분(예시 : 주, 야)
birthday varchar(8) Not null,       #생년월일 (예시 : 19880912)
gender varchar(1) not null,         #성별(예시 : 남자(1,3,5), 여자(2,4,6))
post_no varchar(5) Not null,        #우편번호
address varchar(100),               #주소
tel1 varchar(3),                   #집전화 지역
tel2 varchar(4),                   #집전화 국
tel3 varchar(4),                   #집전화 번호
mobile varchar(14),                #휴대전화번호
primary key (stu_no),
constraint s_dp_fk foreign key(dept_code)  #외래키 학과 테이블의 학과코드
references department(dept_code)
) engine = innoDB;
```

```
# 수강신청
create table attend(
stu_no char(10) Not null,                  #학번
att_year char(4) Not null,                 #수강년도
att_term int(1) Not null,                  # 수강학기
att_isu int(1) Not null,                   #이수구분
sub_code char(5) Not null,                 #과목코드
prof_code char(4) Not null,                #교수코드
att_point int(1) Not null,                 #이수학점
att_grade int(3) default '0',              #취득점수
att_div char(1) default 'N' Not null,      #수강신청구분
att_jae char(1) default '1',               #재수강 구분 1(본학기 수강), 2(재
   수강), 3(계절학기 수강)
att_date date Not null,                    #수강처리일자
primary key (stu_no, att_year, att_term, sub_code, prof_code, att_jae)
) engine = innoDB;
```

등록금테이블

```
create table fee(
stu_no varchar(10) Not null,              #학번
fee_year varchar(4) Not null,             #등록년도
fee_term int(1) Not null,                 #등록학기
fee_enter int(7),                         #입학금
fee_price int(7) Not null,                #등록금(수업료)
fee_total int(7) Default '0' Not null, #등록금총액=입학금+수업료
jang_code char(2) Null,                   #장학코드
jang_total int(7),                        #장학금액
fee_pay int(7) Default '0' Not null,   #납부총액=등록금총액-장학금액
fee_div char(1) Default 'N' Not null,  #등록구분
fee_date date Not null,                   #등록날짜
primary key (stu_no, fee_year, fee_term)
) engine = innoDB;
```

```
# 성적테이블
create table score(
stu_no char(10) Not null,                    #학번
sco_year char(4) Not null,                   #성적취득년도
sco_term int(1) Not null,                    #학기
req_point int(2),                            #신청학점
take_point int(2),                           #취득학점
exam_avg float(2,1),                         #평점평균
exam_total int(4),                           #백분율 총점
sco_div char(1),                             #성적구분
sco_date date,                               #성적처리일자
primary key (stu_no, sco_year, sco_term)
) engine = innoDB;
```

```
# 교과목테이블
create table subject(
sub_code char(5) Not null,              #과목번호
sub_name varchar(50) Not null,          #과목명
sub_ename varchar(50),                  #영문과목명
create_year char(4),                    #개설년도
primary key (sub_code)
)engine = innoDB;

# 교수테이블
create table professor(
prof_code char(4) Not null,             #교수번호
prof_name char(10) Not null,            #교수명
prof_ename varchar(30),                 #교수영문이름
create_date date default null,          #교수임용날짜
primary key (prof_code)
)engine = innoDB;
```

동아리테이블

```
create table circle(
cir_num int(4) Not null auto_increment,  #동아리가입번호
cir_name char(30) Not null,              #동아리명
stu_no char(10) Not Null,                #학번
stu_name char(10) Not Null,              #이름
president char(1) default '2' Not null,  #동아리회장(0), 부회장(1), 회원(2)
primary key (cir_num)
)engine = innoDB;
```

도로명 우편번호테이블

```
create table post(
post_no varchar(6) Not null,          #구역번호          1 신우편번호
sido_name varchar(20) Not null,       #시도명            2
sido_eng varchar(40) Not null,        #시도영문          3
sigun_name varchar(20) Not null,      #시군구명          4
sigun_eng varchar(40) Not null,       #시군구영문        5
rowtown_name varchar(20) Not null,    #읍면              6
rowtown_eng varchar(40) Not null,     #읍면영문          7
road_code varchar(12),                #도로명코드        8 (시군구코드(5)+
   도로명번호(7))
```

```
road_name varchar(80),                  #도로명              9
road_eng varchar(80),                   #도로영문명          10
underground_gubun varchar(1),           #지하여부            11 (0 : 지상,
   1 : 지하, 2 : 공중)
building_bon int(5),                    #건물번호본번        12
building_boo int(5),                    #건물번호부번        13
management_no varchar(25) Not null,     #건물관리번호        14
baedal varchar(40),                     #다량배달처명        15 (NULL)
town_building varchar(200),             #시군구용 건물명     16
row_code varchar(10) Not null,          #법정동코드          17
row_dongname varchar(20),               #법정동명            18
ri_name varchar(20),                    #리명                19
administration_name varchar(40),        #행정동명            20
mountain_gubun varchar(1),              #산여부        21 (0 : 대지, 1 : 산)
bungi int(4),                           #지번본번(번지)      22
town_no varchar(2),                     #읍면동일련번호      23
ho int(4),                              #지번부번(호)        24
gu_post_no varchar(6),                  #구 우편번호         25 (NULL)
post_seq varchar(3),                    #우편일련번호        26 (NULL)
primary key (management_no)
)engine = innoDB;
```

➢ UPLOAD 할 파일 경로 변경

```
mysql> SHOW VARIABLES LIKE 'secure_file_priv';
+------------------+-----------------------------------------------+
| Variable_name    | Value                                         |
+------------------+-----------------------------------------------+
| secure_file_priv | C:₩ProgramData₩MySQL₩MySQL Server 8.4₩Uploads₩ |
+------------------+-----------------------------------------------+
1 row in set, 1 warning (0.00 sec)
```

Upload 할 폴더를 C:₩SQL로 변경

C:₩ProgramData₩MySQL₩MySQL Server 8.4₩my.ini을 변경
secure-file-priv="C:/ProgramData/MySQL/MySQL Server 8.4/Uploads" 을
secure-file-priv="C:/SQL" 로 변경 한 후, 작업관리 서비스에서
MySQL84 서비스를 다시 시작 해준다

```
mysql> SHOW VARIABLES LIKE 'secure_file_priv';
```

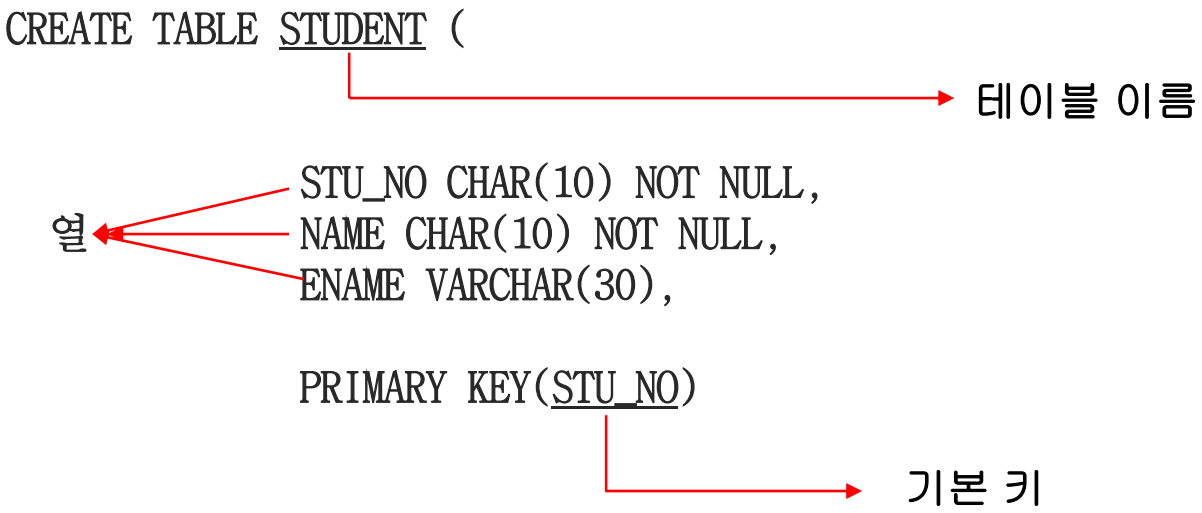
```
mysql -u root -p
Enter password: *******

mysql> SHOW VARIABLES LIKE 'secure_file_priv';
+------------------+---------+
| Variable_name    | Value   |
+------------------+---------+
| secure_file_priv | C:₩SQL₩ |
+------------------+---------+
1 row in set, 1 warning (0.01 sec)
```

- table.sql 파일을 이용한 TABLE 생성

홈페이지에서 다운로드 받은 SQL 파일을 C:₩SQL 이라는 폴더로 복사한다.

```
mysql> ₩. C:₩sql₩table.sql
Query OK, 0 rows affected (0.26 sec)
```

➢ 테이블에는 3개의 속성인 이름, 열, 기본 키를 정의



-테이블의 기본 키는 모든 값이 오직 한번만 나타난 열(또는 열의 조합)이다.
-기본 키는 무결성 규칙의 특별한 자료형이다.
-기본 키가 없다면 중복된 값이 입력될 수 있으므로 문제를 발생
-기본 키를 지원하지 않는 제품은 각 기본 키에 대한 유일한 인덱스를 생성

➢ 인덱스를 생성하는 명령문

```
CREATE INDEX 인덱스이름 on 테이블(칼럼_);
CREATE  INDEX STU_PRIM ON
        STUDENT (STU_NO);
CREATE  INDEX ATT_PRIM ON
        ATTEND (STU_NO);
CREATE  INDEX FEE_PRIM ON
        STUDENT(STU_NO);
CREATE  INDEX SUB_PRIM
        SUBJECT(SUB_CODE);
CREATE  INDEX SCO_PRIM ON
        STUDENT(STU_NO);
CREATE  INDEX PRO_PRIM ON
        PROFESSOR(PROF_CODE);
CREATE  INDEX POS_PRIM ON
        POST(POST_NO);
CREATE  INDEX CIR_PRIM ON
        CIRCLE(CIR_NUM);
CREATE  INDEX DEP_PRIM ON
        DEPARTMENT(DEPT_CODE);
```

4.3.3 테이블에 데이터 삽입

```
# DEPARTMENT 입력
INSERT INTO DEPARTMENT VALUES (10,'간호학과','Dept. of Nersing','1991-02-01');

# STUDENT 학생 테이블 입력
INSERT INTO STUDENT VALUES ('20141001', '박도상', 'Park Do-Sang', 40, 4, 1, '주',
'19960116','1','01066','101동 203호','02','744','6126','010-0611-9884','1996');

# SUBJECT 입력
INSERT INTO SUBJECT VALUES ('4001', '데이터베이스 응용', 'Database Application', '2002');

# PROFESSOR 입력
INSERT INTO PROFESSOR VALUES ('4001','정진용','Jung jin-yong','1995-09-01');

# ATTEND 입력
INSERT INTO ATTEND VALUES ('20141001','2014',1,3,4001,'4002',3, 99,'Y','1','2014-03-05');

# FEE 입력
INSERT INTO FEE VALUES ('20141001','2014',1,500000,3000000,3500000,01,500000,3000000,'Y','2014-02-18');

# SCORE 입력
INSERT INTO SCORE VALUES ('20141001','2014',1,18,18,4.5,580,'Y','2014-08-10');

# CIRCLE 입력
INSERT INTO CIRCLE VALUES (1,'컴맹탈출','20141001','박도상','0');
```

➢ insert.sql 파일을 이용한 TABLE 생성

insert.sql 파일을 C:₩SQL 이라는 폴더로 복사
mysql> ₩. c:₩sql₩insert.sql
Query OK, 1 row affected (0.02 sec)

➢ 우편번호 테이블 우정사업본부 홈페이지에서 다운로드

홈페이지에서 다운로드 받은 zipcode의 txt파일을 C:₩SQL₩ZIP_CODE₩ 폴더로 복사한다.

```
mysql> load data infile 'C:/SQL/zip_code/mysql-post.txt' into table post fields terminated by '|';
Query OK, 17 rows affected (0.01 sec)
Records: 17  Deleted: 0  Skipped: 0  Warnings: 0

mysql> load data infile 'C:/SQL/zip_code/build_seoul.txt' into table post fields terminated by '|';
Query OK, 564915 rows affected (21.48 sec)
Records: 564915  Deleted: 0  Skipped: 0  Warnings: 0
```

4.3.4 질의 테이블

SELECT 명령문은 테이블로부터 데이터를 검색할 때 사용

[예제 4-1] STUDENT 테이블로부터 성별이 남자인 각 학생의 학번, 이름, 영문 이름, 학년, 성별을 영문이름 순서로 출력하라.

```
mysql> select stu_no, stu_name, stu_ename, grade, gender
    -> from student
    -> where gender = 1 or gender = 3 or gender = 5
    -> order by stu_ename;
+----------+-------------+-----------------+-------+--------+
| stu_no   | stu_name    | stu_ename       | grade | gender |
+----------+-------------+-----------------+-------+--------+
| 20201002 | 강감찬      | Gang Gam-Chan   |     1 | 3      |
| 20191002 | 홍길동      | Hong Gil-Dong   |     1 | 3      |
| 20181001 | 장수인      | Jang Soo-In     |     2 | 1      |
| 20191005 | 김할리      | Kim Hal-Li      |     1 | 5      |
| 20191001 | 김유신      | Kim Yoo-Shin    |     1 | 3      |
| 20201001 | 김영호      | Kim Young-Ho    |     1 | 3      |
| 20181003 | 이상진      | Lee Sang-Gin    |     2 | 1      |
| 20191004 | 이순신      | Lee Sun-Shin    |     1 | 3      |
| 20141001 | 박도상      | Park Do-Sang    |     4 | 1      |
| 20191008 | 연개소문    | Yean Gae-So-Moon|     1 | 3      |
+----------+-------------+-----------------+-------+--------+
10 rows in set (0.00 sec)
```

-FROM 다음에는 질의를 원하는 테이블을 지정 : FROM STUDENT
-WHERE 다음에는 원하는 조건을 기술
-성별이 남자인 학생을 찾는 조건 :
WHERE gender = 1 or gender = 3 or gender = 5
성별 gender 컬럼에서 1900~1999년생 까지 남자(1), 여자(2)이고
2000년생 이후에 태어난 남자(3), 여자(4)이며,
외국인은 남자(5), 여자(6)으로 되어있다.
-SELECT는 탐색하고자 하는 열을 선택
학생의 학번, 이름, 영문이름, 학년, 성별을 선택하여 출력
SELECT STU_NO, STU_NAME, STU_ENAME, GRADE, GENDER
-ORDER BY 다음에는 출력될 데이터가 정렬되는 방법을 지정
ORDER BY STU_NAME asc ; 오름차순(ascending)이 생략되어 있음
내림차순일 때는 desc(descending)을 기술

● SELECT 명령문의 결과로 나타나는 것은 SQL에 의해서 결정
① 열의 폭은 열의 자료형에 의해서 결정
② 표제의 이름은 SELECT 명령문에서 사용한 열의 이름과 동일하게 사용
③ 열에 있는 값이 출력될 때 문자형 데이터는 왼쪽 정렬로 출력
④ 수치 값은 오른쪽 정렬로 출력
⑤ 출력 결과 테이블의 각 열 사이에는 1 칸의 공백이 존재
⑥ NULL 값은 NULL또는 공백(SPACE)으로 출력 : INSERT할 때 입력한 값(NULL, 공백)에 따라 출력

[예제 4-2] 학년이 1학년이고 성별이 여자인 각 학생의 학번과 이름을 출력하는데, 출력 순서는 학번 내림차순이다.

```
mysql> select stu_no, stu_name, gender
    -> from student
    -> where grade = 1
    -> and gender = 2 or gender = 4 or gender = 6
    -> order by stu_no desc;
+----------+-------------+--------+
| stu_no   | stu_name    | gender |
+----------+-------------+--------+
| 20191009 | 유하나      | 4      |
| 20191007 | 신안나      | 6      |
| 20191006 | 최에스터    | 6      |
| 20191003 | 고혜진      | 4      |
+----------+-------------+--------+
4 rows in set (0.00 sec)
```

[예제 4-3] 교과목 테이블에 관한 모든 정보를 출력하라.

```
mysql> select * from subject;
+----------+-----------------------------+-----------------------------------------+-------------+
| sub_code | sub_name                    | sub_ename                               | create_year |
+----------+-----------------------------+-----------------------------------------+-------------+
| 4001     | 데이터베이스 응용           | Database Application                    | 2002        |
| 4002     | 웹사이트 구축               | Web Site Construction                   | 2003        |
| 4003     | 소프트웨어공학              | Software Engineering                    | 2003        |
| 4004     | 웹프로그래밍                | Web Programming                         | 2004        |
| 4005     | 컴퓨터구조                  | Computer Structure                      | 2001        |
| 4006     | 정보처리실무                | Information Process Practical business  | 2001        |
| 4007     | UML                         | Unified Modeling Language               | 2005        |
| 4008     | 운영체제                    | Operating System                        | 2002        |
| 4009     | 객체지향프로그래밍          | Object Oriented Programming             | 2003        |
| 4010     | 윈도우즈 프로그래밍         | Windows Programming                     | 2006        |
| 4011     | 자바프로그래밍              | Java Programming                        | 2006        |
| 4012     | 파이썬 프로그래밍           | Python Programming                      | 2019        |
| 4013     | 스크래치 프로그래밍         | Scratch Programming                     | 2019        |
+----------+-----------------------------+-----------------------------------------+-------------+
13 rows in set (0.00 sec)
```

교과목(FROM SUBJECT)에 대하여 모든 열 값(SELECT *)을 출력하는 명령문

* 문자는 모든 열(“ALL COLUMNS”)을 나타내는 문자

4.3.5 행의 갱신과 삭제

UPDATE 명령문은 행에 있는 열의 값을 변경할 때 사용
DELETE 명령문은 테이블에서 완전히 행을 삭제할 때 사용

[예제 4-4] 교과목 중 운영체제의 생성년도를 2006년으로 변경하라.

```
mysql> UPDATE  SUBJECT
    -> SET  CREATE_YEAR = '2006'
    -> WHERE  SUB_NAME ='운영체제';
```

-UPDATE SUBJECT : 교과목 테이블을 변경하라.
-WHERE SUB_NAME = '운영체제' : 조건은 교과목명이 운영체제인 것을 변경
-SET CREATE_YEAR = '2006' : 변경할 내용은 생성년도를 2006년으로 변경
SET 다음에는 새로운 값으로 변경될 열을 지정한다.
이 때 값은 기존에 존재하는 값에 관계없이 변경된다.

[예제 4-5] 교과목 테이블에서 교과목코드, 교과목명, 교과목영문이름, 생성년도를 출력하라.

```
mysql> select sub_code, sub_name, sub_ename, create_year
    -> from subject;
+----------+--------------------+-------------------------------------+-------------+
| sub_code | sub_name           | sub_ename                           | create_year |
+----------+--------------------+-------------------------------------+-------------+
| 4001     | 데이터베이스 응용  | Database Application                | 2002        |
| 4002     | 웹사이트 구축      | Web Site Construction               | 2003        |
| 4003     | 소프트웨어공학     | Software Engineering                | 2003        |
| 4004     | 웹프로그래밍       | Web Programming                     | 2002        |
| 4005     | 컴퓨터구조         | Computer Structure                  | 2001        |
| 4006     | 정보처리실무       | Information Process Practical business | 2001        |
| 4007     | UML                | Unified Modeling Language           | 2005        |
| 4008     | 운영체제           | Operating System                    | 2006        |
| 4009     | 전자상거래 실무    | Electronic Commerce                 | 2003        |
| 4010     | 윈도우즈 프로그래밍 | Windows Programming                 | 1998        |
| 4011     | 자바프로그래밍     | Java Programming                    | 1999        |
| 4012     | 파이썬 프로그래밍  | Python Programming                  | 2019        |
| 4013     | 스크래치 프로그래밍 | Scratch Programming                 | 2019        |
+----------+--------------------+-------------------------------------+-------------+
13 rows in set (0.05 sec)
```

교과목 코드 4008번인 운영체제의 생성년도가 2006년으로 변경됨

삭제처리(DELETE) 작업은 테이블의 내용을 제거할 때 행(ROW) 단위로 이루어진다.
만약에 “SAMPLE"이라는 테이블의 모든 행을 삭제한다면 다음과 같다.

```
DELETE FROM SAMPLE;
```

위의 예제는 “SAMPLE" 테이블의 모든 행을 삭제처리 하나 테이블 자체가 없어진 것은 아니다.

테이블의 명세표, 인덱스, 부여된 권한 등 환경테이블 정보 자체를 완전히 없애버릴 때는 DROP명령어를 다음과 같이 사용한다.

```
“DROP TABLE SAMPLE"
```

[예제 4-6] 과목명(SUB_NAME)이 UML인 과목을 삭제하라.

```
mysql> delete
    -> from subject
    -> where sub_name = 'UML';
Query OK, 1 row affected (0.06 sec)

mysql> select * from subject;
```

sub_code	sub_name	sub_ename	create_year
4001	데이터베이스 응용	Database Application	2002
4002	웹사이트 구축	Web Site Construction	2003
4003	소프트웨어공학	Software Engineering	2003
4004	웹프로그래밍	Web Programming	2002
4005	컴퓨터구조	Computer Structure	2001
4006	정보처리실무	Information Process Practical business	2001
4008	운영체제	Operating System	2006
4009	전자상거래 실무	Electronic Commerce	2003
4010	윈도우즈 프로그래밍	Windows Programming	1998
4011	자바프로그래밍	Java Programming	1999
4012	파이썬 프로그래밍	Python Programming	2019
4013	스크래치 프로그래밍	Scratch Programming	2019

```
12 rows in set (0.05 sec)
```

교과목 코드 4007번인 UML과목이 삭제됨

[예제 4-7] 교과목 중 운영체제의 생성년도를 2002년으로 변경하라.

```
mysql> UPDATE SUBJECT
    -> SET CREATE_YEAR = '2002'
    -> WHERE SUB_NAME = '운영체제';
```

[예제 4-8] 교과목 테이블에 교과목코드(4007), 교과목명(UML), 교과목영문이름(Unified Modeling Language), 생성년도(2005)인 새로운 행을 삽입하라.

```
mysql> INSERT INTO SUBJECT VALUES (
    -> '4007', 'UML',  'Unified Modeling Language' , '2005');
```

교과목테이블의 원래의 데이터로 복원하기 위하여 UPDATE, INSERT 명령을 수행해 보았다.

◆ 질의(Query) 처리의 최적화

```
mysql> SELECT  *
    -> FROM  SUBJECT
    -> WHERE  CREATE_YEAR = '2002';
```

-생성년도(CREATE_YEAR)가 2002년이면 결과 테이블에 출력
-만약 테이블이 몇 개의 행만을 가지고 있다면 SQL은 아주 빠르게 작업을 수행
-수십만 개의 행을 가지고 있다면 각 행을 조사하기 때문에 많은 시간을 요구
-인덱스를 정의하게 되면 처리 속도가 훨씬 빠르게 수행

➢ 인덱스(index)는 열 또는 열의 조합으로 정의

```
CREATE          INDEX CREATEIX ON
                SUBJECT (CREATE_YEAR)
```

-SUBJECT 테이블에서 CREATE_YEAR 열에 대한 CREATEIX 인덱스를 정의
-인덱스는 WHERE 조건 절을 만족하는 행을 보기 위해서만 사용
-테이블에서 조건에 만족하는 행의 검색은 훨씬 더 빨라진다.
-인덱스 CREATEIX는 행의 직접 접근을 제공

➢ 인덱스를 사용하는 중요한 점

·인덱스는 SELECT 명령문의 처리를 최적화(빠르게 처리)하기 위해서 사용

·인덱스에 관한 내용은 SELECT 명령문에서 명확하게 표현하지는 않는다.

·명령문을 처리할 때 SQL은 이미 존재하는 인덱스가 사용될 것인지 선택한다.

·인덱스는 언제든지 생성되거나 삭제될 수 있다.

·행을 추가하거나 삭제, 갱신이 발생할 때 테이블에 관련된 인덱스를 유지한다.

·SELECT 명령문의 처리 시간을 감소시킨다는 것을 의미

·갱신 명령문(INSERT, UPDATE, DELETE 등)의 시간은 증가

-SQL은 최적화 처리를 위해서 이러한 유일한 인덱스를 사용

-유일한 인덱스(Unique index)는 또 다른 기능을 가지고 있는데, 이는 중복된 값이 없는 특별한 열 또는 열의 조합을 보장해야 한다.

4.3.6 뷰(View)

-뷰(Views) 테이블은 필요에 따라 사용자가 재 정의하여 생성해주는 테이블
-뷰(Views) 테이블은 어떤 기억 공간을 차지하지는 않는다.
-뷰는 유도된(derived) 또는 가상(virtual) 테이블이라고도 한다.
-뷰는 실제 데이터 행을 가지고 있는 것처럼 동작하지만 데이터 행은 없음

-뷰가 주로 사용되는 경우

- 반복되는 명령문이나 루틴(routine)을 간단히 사용하고자 할 때,
- 테이블의 출력 방법을 재구성하고자 할 때,
- 여러 단계에서 SELECT 명령문이 사용될 때,
- 데이터를 보호하고자 할 때

[예제 4-9] 학적 테이블의 학번, 이름, 생년월일, 나이를 출력하라.

```
mysql> select stu_no, stu_name, birthday "생년월일",
    -> year(now()) -  substring(birthday, 1, 4) + 1 "나이"
    -> from student;
```

stu_no	stu_name	생년월일	나이
20141001	박도상	19960116	24
20161001	박정인	19970403	23
20181001	장수인	19990209	21
20181002	정인정	19990315	21
20181003	이상진	19990819	21
20181004	김유미	19990207	21
20191001	김유신	20001007	20
20191002	홍길동	20000402	20
20191003	고혜진	20000307	20
20191004	이순신	20000222	20
20191005	김할리	20010418	19
20191006	최에스터	20021003	18
20191007	신안나	20011214	19
20191008	연개소문	20000615	20
20191009	유하나	20000921	20
20201001	김영호	20010811	19
20201002	강감찬	20010312	19

17 rows in set (0.00 sec)

열의 제목 중에서 생년월일(BIRTHDAY)를 출력할 때 “생년월일”로 출력한다. NOW()함수는 현재의 년,월,일,시간을 나타내는 함수이고 YEAR()함수는 년도를 나타내는 함수이다.
따라서 NOW()함수로 현재의 날짜를 추출한 다음 다시 YEAR()함수로 년도만을 추출한다. 나이는 현재의 연도에서 출생년도를 감산하여 1를 더한 값으로 얻을 수 있다.(예를들면 현재의 날짜가 2019년 2월 6일)일 때 2000년생의 나이를 계산하면 다음과 같다.

나이=현재 연도-출생년도+1

2019-2000+1=20

STUDENT 테이블의 출생년도를 계산하여 AGES라는 VIEW 테이블을 작성

[예제 4-10] 학적 테이블의 학번, 이름, 나이로 구성된 AGES 뷰 테이블을 생성하라.

```
mysql> create view ages(stu_no, stu_name, age) as
    -> select stu_no, stu_name, year(now()) - substring(birthday, 1, 4) + 1
    -> from student;
Query OK, 0 rows affected (0.14 sec)
```

```
mysql> select * from ages;
+----------+-------------+------+
| stu_no   | stu_name    | age  |
+----------+-------------+------+
| 20141001 | 박도상      |   24 |
| 20161001 | 박정인      |   23 |
| 20181001 | 장수인      |   21 |
| 20181002 | 정인정      |   21 |
| 20181003 | 이상진      |   21 |
| 20181004 | 김유미      |   21 |
| 20191001 | 김유신      |   20 |
| 20191002 | 홍길동      |   20 |
| 20191003 | 고혜진      |   20 |
| 20191004 | 이순신      |   20 |
| 20191005 | 김할리      |   19 |
| 20191006 | 최에스터    |   18 |
| 20191007 | 신안나      |   19 |
| 20191008 | 연개소문    |   20 |
| 20191009 | 유하나      |   20 |
| 20201001 | 김영호      |   19 |
| 20201002 | 강감찬      |   19 |
+----------+-------------+------+
17 rows in set (0.01 sec)
```

-AGES 뷰 테이블은 데이터베이스에 저장되지는 않지만 SELECT 명령문 (또 다른 명령문)이 실행되는 순간에 유도된다.

-뷰의 사용은 기억 공간을 사용하는 부가적인 비용이 없고, 다른 테이블에 이미 저장된 데이터로 구성

4.4 보안설정

4.4.1 ROOT 사용자의 데이터 보안

-데이터베이스에 저장된 데이터는 부정확한 사용이나 잘못된 사용에 대비하여 보호되어야 한다.
-GRANT라는 명령문을 사용하여 사용자의 접근을 등록할 수 있다.

1) root 패스워드를 “12345”로 설정하기

```
mysql> use mysql;
Database changed

mysql> update user set password=password('12345') where user = 'root';
Query OK, 1 row affected (0.09 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> flush privileges;
Query OK, 0 rows affected (0.08 sec)
```

2) 슈퍼 유저[Root] 패스워드 변경 2가지

➢ SET PASSWORD 사용하기

-SET PASSWORD 문을 사용하여, root의 패스워드를 12345로 변경하는 예제
-flush privileges; 명령을 사용하지 않아도 바로 적용

```
mysql> set password for root@localhost = password('12345');
Query OK, 0 rows affected (0.05 sec)
```

➢ UPDATE 문으로 user 테이블 수정하기

-UPDATE 문을 사용하여 mysql 시스템 데이터베이스 안의 user 테이블을 수정
-flush privileges; 명령을 주어야 적용

```
mysql> update user set password=password('12345')
    -> where user='root';
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```

➢ PASSWORD() 함수

-PASSWORD()는 해독할 수 없는 암호화 방식

-base64 인코딩 방식은 디코딩이 가능하지만, pasword()는 해독 함수가 존재하지 않는다.

-for 루프 등을 돌려서 추측할 수 있으므로, 비밀번호는 최소 8자 이상으로 설정

-비밀번호는 길이가 길어지면, 해독 시간이 늘어나기 때문

```
mysql> select password('12345');
+-------------------------------------------+
| password('12345')                         |
+-------------------------------------------+
| *00A51F3F48415C7D4E8908980D443C29C69B60C9 |
+-------------------------------------------+
1 row in set (0.00 sec)
```

4.4.2 사용자 생성 및 권한 부여

➢ CREATE 문으로 user 생성하기

[형식] create user 사용자명 identified by '비밀번호';

① CREATE 문을 사용하여 새로운 사용자 “choi”를 암호(Password) “choi123” 으로 생성해 보자.
② CREATE 문을 사용하여 새로운 사용자 “lee@localhost”를 암호(Password) “lee123” 으로 생성해 보자.

```
① mysql> create user choi identified by 'choi123';
   Query OK, 0 rows affected (0.00 sec)

② mysql> create user lee@localhost identified by 'lee123';
   Query OK, 0 rows affected (0.16 sec)
```

-테이블의 "user" 테이블에서 새로이 생성된 사용자를 확인하기 위해 호스트명, 사용자, 비밀번호를 출력해보자.

```
mysql> select host, user, password from user;
+-----------+------+-------------------------------------------+
| host      | user | password                                  |
+-----------+------+-------------------------------------------+
| localhost | root | *FAAFFE644E901CFAFAEC7562415E5FAEC243B8B2 |
| %         | choi | *C6A35C53FC3460B0050486E1FB5A9D6B916407FA |
| localhost | lee  | *694E6C3F78B6E0EE7CCC45C949CD225769EE7CA6 |
+-----------+------+-------------------------------------------+
3 rows in set (0.00 sec)
```

사용자 “lee”는 host명이 “localhost”이므로 localhost로 접속이 가능
사용자 “choi”의 경우는 host명이 “%”이므로 원격에서 접속이 가능

➢ 사용자 권한부여

[형식1] grant all privileges on 데이터베이스명.* to 사용자명;

[형식2] grant 부여할 권한 SQL명령문 on 데이터베이스명.* to 사용자명;

① mysql> grant select, insert, update, delete on haksa.* to lee@localhost;
Query OK, 0 rows affected (0.00 sec)
② mysql> grant all privileges on haksa.* to choi;
Query OK, 0 rows affected (0.02 sec)
③ mysql> grant all privileges on *.* to lee@localhost;
Query OK, 0 rows affected (0.00 sec)

①사용자 “lee”에게 “haksa” DB를 select, insert, update, delete 권한 부여
②사용자 “choi”에게 “haksa” DB를 관리할 수 있는 모든 권한 부여
③사용자 “lee”는 모든 DB를 모든 권한을 가지고 관리할 수 있도록 권한 부여

➢ 사용자 생성 및 권한부여를 동시에 처리

[일반형식] grant priv_type [(column_list)] [, priv_type [(column_list)] ...]
on tbl_name | * | *.* | db_name.* to user_name [identified by 'password']
[, user_name [identified by 'password'] ...] [with grant option]

[형식1] grant all privileges on DB명.* to DB계정명@localhost identified by '비밀번호' with grant option;
[형식2] grant all privileges on DB명.* to DB계정명 identified by '비밀번호' with grant option;

[형식1]의 경우는 localhost에서 'DB계정명' 이라는 사용자를 등록한 경우
[형식2]의 경우는 localhost 아닌 원격에서 접속시 호스트를 % 로 해준 경우

① mysql> grant all privileges on haksa.* to kim@localhost identified by 'kim123' with grant option;
Query OK, 0 rows affected (0.00 sec)
② mysql> grant all privileges on *.* to han identified by 'han123' with grant option;
Query OK, 0 rows affected (0.00 sec)
①사용자 “kim”, password는 “kim123” 생성, “haksa” DB에 모든 권한 부여
②사용자 “han”, password는 “han123” 생성, 모든 DB에 모든 권한 부여

"user" 테이블에서 새로이 생성된 사용자를 확인하기 위해 호스트명, 사용자, 비밀번호를 출력해보자.

```
mysql> select host, user, password from user;
+-----------+------+-------------------------------------------+
| host      | user | password                                  |
+-----------+------+-------------------------------------------+
| localhost | root | *FAAFFE644E901CFAFAEC7562415E5FAEC243B8B2 |
| localhost | kim  | *DFC0E7A43E14F96552318E8651E32B546FE91C59 |
| localhost | lee  | *694E6C3F78B6E0EE7CCC45C949CD225769EE7CA6 |
| %         | choi | *C6A35C53FC3460B0050486E1FB5A9D6B916407FA |
| %         | han  | *C4B2087D966138F5AB4470F41E7BEEFDC593157D |
+-----------+------+-------------------------------------------+
5 rows in set (0.00 sec)
```

새로이 생성된 사용자의 데이터베이스 권한을 확인하기 위해 “db”테이블에서 호스트명, DB명, 사용자, select 권한만을 출력해보자.

```
mysql> select host, db, user, select_priv from db;
+-----------+-------+------+-------------+
| host      | db    | user | select_priv |
+-----------+-------+------+-------------+
| localhost | haksa | lee  | Y           |
| %         | haksa | choi | Y           |
| localhost | haksa | kim  | Y           |
+-----------+-------+------+-------------+
3 rows in set (0.00 sec)
```

➢ 사용자 권한 회수

[일반형식] revoke priv_type [(column_list)] [, priv_type [(column_list)] ...]
on tbl_name | * | *.* | db_name.* from user_name [, user_name ...]
[형식] revoke SQL명령문 on DB명.* from '해당유저이름';

① mysql> revoke select on haksa.* from choi@'%';
Query OK, 0 rows affected (0.00 sec)
② mysql> revoke select, update on haksa.* from lee@'localhost';
Query OK, 0 rows affected (0.00 sec)
① 사용자 "choi"을 "haksa" 데이터베이스에서 select할 수 있는 권한 회수
② 사용자 "lee"을 "haksa" 데이터베이스에서 select, update 권한 회수

mysql> flush privileges;

사용자 "kim"과 "lee"의 권한이 회수되었는지 확인해 보자.
mysql> select host, db, user, select_priv, update_priv from db;

host	db	user	select_priv	update_priv
localhost	haksa	lee	N	N
%	haksa	choi	N	Y

2 rows in set (0.00 sec)

➢ 사용자 삭제

[형식1] drop user '해당유저이름';
[형식2] delete from user where user='해당유저이름';
[형식3] delete from db where user='해당유저이름';

[형식1]은 “user” 테이블과 "db" 테이블에서 해당유저를 완전히 삭제
[형식2]는 “user” 테이블에서 사용자를 삭제하는 경우
[형식3]은 "db" 테이블에서 해당유저에게 부여된 데이터베이스의 권한을 삭제

사용자의 데이터베이스를 권한을 알 수 있는 "db" 테이블의 정보를 알아보자.

```
mysql> desc db;
+-----------------------+---------------+------+-----+---------+-------+
| Field                 | Type          | Null | Key | Default | Extra |
+-----------------------+---------------+------+-----+---------+-------+
| Host                  | char(60)      | NO   | PRI |         |       |
| Db                    | char(64)      | NO   | PRI |         |       |
| User                  | char(16)      | NO   | PRI |         |       |
| Select_priv           | enum('N','Y') | NO   |     | N       |       |
| Insert_priv           | enum('N','Y') | NO   |     | N       |       |
| Update_priv           | enum('N','Y') | NO   |     | N       |       |
| Delete_priv           | enum('N','Y') | NO   |     | N       |       |
             :                    :                    :

| Drop_priv             | enum('N','Y') | NO   |     | N       |       |
| Execute_priv          | enum('N','Y') | NO   |     | N       |       |
+-----------------------+---------------+------+-----+---------+-------+
20 rows in set (0.00 sec)
```

"db"테이블에서 호스트명, DB명, 사용자, select 권한만을 출력해보자.

```
mysql> select host, db, user, select_priv from db;
+-----------+-------+------+-------------+
| host      | db    | user | select_priv |
+-----------+-------+------+-------------+
| localhost | haksa | lee  | Y           |
| %         | haksa | choi | Y           |
| localhost | haksa | kim  | Y           |
+-----------+-------+------+-------------+
3 rows in set (0.00 sec)
```

① mysql> drop user kim@localhost;
② mysql> delete from user where user='lee';
③ mysql> delete from db where user='lee';

①의 경우는 사용자 "kim"을 완전히 삭제 처리한 경우
②의 경우는 "user"테이블에서 사용자 "lee"를 삭제한 경우
③의 경우는 "db"테이블에서 사용자 "lee"를 삭제한 경우

```
mysql> select host, db, user, select_priv from db;
+-----------+-------+------+-------------+
| host      | db    | user | select_priv |
+-----------+-------+------+-------------+
| %         | haksa | choi | Y           |
+-----------+-------+------+-------------+
1 rows in set (0.00 sec)
```