SQL

DCL – Data Control Language

Database User

- 데이터베이스 사용자
  - 데이터베이스에 접속하여 데이터를 이용하기 위해 접근하는 모든 사람
  - 이용 목적에 따라 데이터베이스 관리자(Database Administrator: DBA), 최종 사용자, 응용프로그래머로 구분
  - MySQL 설치시 기본으로 root 계정이 생성
    - root 계정은 데이터베이스에 대한 모든 권한을 가지므로 반드시 비밀번호를 입력하여 임의의 접속으로부터 데이터베이스를 보호해야 한다

사용자 관리

- Syntax
  - 사용자 생성 : CREATE USER username IDENTIFIED BY password;
  - 비밀번호 변경 : ALTER USER username IDENTIFIED BY password;
  - 사용자 삭제 : DROP USER username;

- 주의사항
  - 일반적으로 DBA의 일
  - 사용자를 생성하려면 CREATE USER 권한 필요
  - root 계정은 데이터베이스 전체에 대한 모든 권한을 가지므로 실제로는 사용자를 생성하고 데이터베이스에 제한된 접근 권한을 주어 실수 혹은 오작동으로 인한 데이터의 손상 혹은 오용을 막아야 한다

사용자 관리

- [연습] bituser 사용자를 생성하고 비밀번호를 변경해 봅니다

```
mysql> CREATE USER 'testuser'@'localhost' IDENTIFIED BY 'test';
Query OK, 0 rows affected (0.05 sec)
```

- [연습] 새로 생성한 user1 계정으로 MySQL 데이터베이스에 접속해 봅니다.

```
mysql -utestuser -ptest
```

- [연습] 현재 접속한 계정의 정보를 확인해 봅니다.

```
mysql> SELECT CURRENT_USER;
mysql> SELECT USER();
```

권한(Privilege)과 롤(Role)
: Privileges

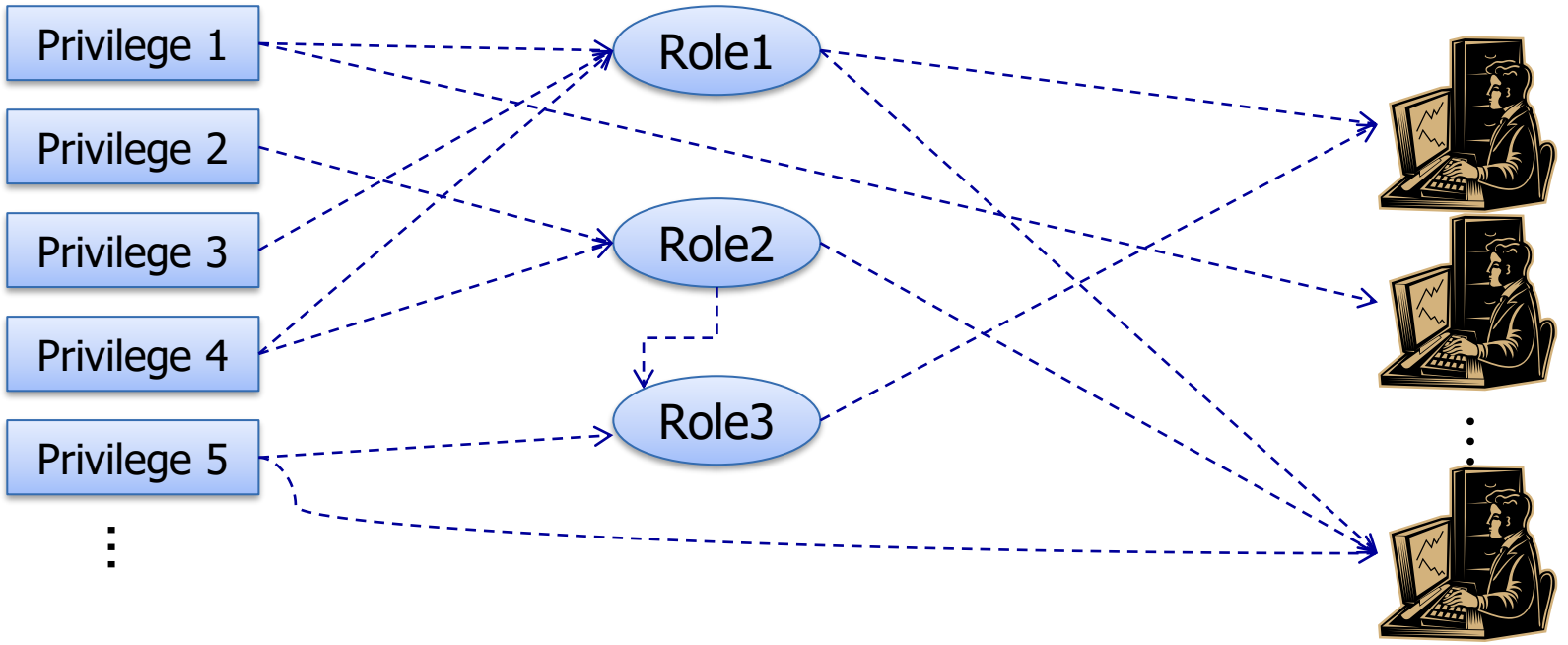
- 권한(Privilege)
  - 사용자가 특정 SQL 문을 실행하거나 특정 정보에 접근할 수 있는 권리
  - 사용자는 작업에 요구되는 관련 권한에 대한 허가(GRANT)를 받아야 한다
  - 종류
    - 시스템 권한
    - 스키마 객체 권한
    - 참고: https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html

  - 권한의 확인 : SHOW GRANTS

```
mysql> SHOW GRANTS;
mysql> SHOW GRANTS FOR 'testuser'@'localhost';
mysql> SHOW GRANTS FOR CURRENT_USER();
```

권한(Privilege)과 롤(Role)

- 롤(Role)
  - 권한을 쉽게 관리하기 위하여 특정한 종류별로 묶어놓은 그룹



GRANT / REVOKE

- 권한/롤을 부여(GRANT)하거나 회수(REVOKE)
- Syntax (System Privileges)

```
GRANT [system_priv|role [,system_priv|role ...]
    ON *.* TO {user [,user ...]|role|PUBLIC}
    [WITH ADMIN OPTION];
```

```
REVOKE [system_priv|role [,system_priv|role ...] ON *.*
    FROM {user [,user ...]|role|PUBLIC};
```

- Syntax (Object Privileges)

```
GRANT {[object_priv [,object_priv ...]|ALL}
    ON object TO {user [,user ...]|role|PUBLIC}
    [WITH ADMIN OPTION];
```

```
REVOKE {[object_priv [,object_priv ...]|ALL} ON object
    FROM {user [,user ...]|role|PUBLIC};
```

- GRANT, REVOKE 문의 효력을 발휘하려면 flush privileges;

GRANT / REVOKE

- 시스템 권한: 관리자로 수행

```
GRANT CREATE ON *.* TO 'testuser'@'localhost';
```

```
REVOKE CREATE ON *.* FROM 'testuser'@'localhost';
```

- 스키마 객체 권한

```
GRANT SELECT ON employees.* TO 'testuser'@localhost;
```

```
REVOKE SELECT ON employees.* FROM 'testuser'@localhost;
```

- WITH GRANT OPTION
  - 해당 권한을 받은 사용자가 다시 제3자에게 권한을 부여할 수 있도록 하는 옵션
  - 권한을 REVOKE 하면 해당 사용자가 3자에게 부여한 권한들도 함께 회수됨

```
GRANT select ON hr.* TO user2@localhost
WITH GRANT OPTION;
```

ROLE

- ROLE을 생성한 후 Role에 Privilege를 Grant하여 Role 관리
  - 주로 DBA가 하는 작업

```
mysql> CREATE ROLE OBSERVER;
Query OK, 0 rows affected (0.10 sec)
mysql> GRANT SELECT ON employees.* TO OBSERVER;
Query OK, 0 rows affected (0.08 sec)
```

- 특정 Role을 사용자에게 Grant / Revoke

```
mysql> GRANT OBSERVER TO 'testuser'@'localhost';
```

- Role에 의해 부여된 Grant 확인

```
mysql> SHOW GRANTS FOR CURRENT_USER USING OBSERVER;
```

```
mysql> SHOW GRANTS FOR OBSERVER;
```

ROLE

- 현재 사용자의 ROLE을 확인해 봅시다

```
mysql> SELECT CURRENT_ROLE();
+----------------+
| CURRENT_ROLE() |
+----------------+
| NONE           |
+----------------+
1 row in set (0.00 sec)
```

- ROLE의 활성화(Activation)

```
mysql> SET ROLE OBSERVER;
```

- Role에 의해 부여된 Grant 확인

```
mysql> SHOW GRANTS FOR CURRENT_USER USING OBSERVER;
```

SQL

DDL – Data Definition Language Basic

DDL 요약

- 데이터베이스 관련
  - **CREATE DATABASE :** 데이터베이스 생성
  - **SHOW DATABASES :** 가용한 데이터베이스 목록 확인
  - **DROP DATABASE :** 데이터베이스 삭제
- 테이블 관련
  - **CREATE TABLE :** 테이블 생성
  - **ALTER TABLE :** 테이블 관련 변경
  - **DROP TABLE :** 테이블 삭제
  - **RENAME :** 이름 변경
  - **TRUNCATE :** 테이블의 모든 데이터 삭제
  - **COMMENT :** 테이블에 설명 추가

데이터베이스 생성

- CREATE DATABASE 문 이용
- Syntax

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
    [create_option] ...

create_option: [DEFAULT] {
    CHARACTER SET [=] charset_name
  | COLLATE [=] collation_name
  | ENCRYPTION [=] {'Y' | 'N'}
}
```

  - 일반적으로는 데이터베이스 명만 명시해 주어 생성
  - MySQL에서는 DATABASE와 SCHEMA를 같은 용어로 간주
    - 데이터베이스를 만든다는 것은 테이블을 포함, 데이터베이스 객체들의 집합을 저장하는 저장 공간을 만드는 것이다
  - DEFAULT 키워드와 함께 입력하지 않았을 때의 초기값을 지정할 수 있음
  - create_option을 부여하여 데이터베이스의 인코딩 등을 지정할 수 있음

데이터베이스 생성/삭제

- 일반적인 데이터베이스의 생성

```
CREATE DATABASE test_db;
```

- 데이터베이스에 특정 인코딩 부여

```
CREATE DATABASE test_db
    DEFAULT CHARACTER SET utf8
    COLLATE utf8_general_ci;
```

- 데이터베이스의 삭제

```
DROP DATABASE test_db;
```

  - 데이터베이스를 삭제하면 데이터베이스 내의 테이블 등 모든 데이터베이스 객체들도 함께 삭제되니 주의

MySQL의 데이터 유형

TINYINT(M)	부호 있는 수는 -128~127까지, 부호 없는 수는 0~255까지 표현. 1 바이트
SMALLINT(M)	부호 있는 수는 -32768~32767까지, 부호 없는 수는 0~65535까지 표현. 2 바이트
MEDIUMINT(M)	부호 있는 수는 -8388608~8388607까지, 부호 없는 수는 0~16777215까지 수를 표현. 3 바이트
INT(M) or INTEGER(M)	부호 있는 수는 -2147483648~2147483647까지, 부호 없는 수는 0~4294967295까지 . 4 바이트
BIGINT(M)	부호있는 수는 -9223372036854775808 ~ 9223372036854775808 부호 없는 수는 0~18446744073709551615
FLOAT(M,D)	부동 소수점을 나타낸다. 언제나 부호 있는 수임. (-3.402823466E+38~3.402823466E+38)
DOUBLE(M,D)	2배 정밀도를 가진 부동 소수점. (-1.79769313486231517E+308~6931348623157E+308)
DATE	날짜를 표현하는 타입. '9999-12-31'. 3 바이트
DATETIME	날짜와 시간을 같이 나타내는 타입. '9999-12-31 23:59:59'. 8 바이트
TIMESTAMP	'1970-01-01 00:00:00' 부터 2037년까지 나타낼 수 있다. 4 바이트

MySQL의 데이터 유형

TIME	시간을 나타낸다. '-839:59:59' 부터 '838:59:59'까지 나타낼 수 있다.
YEAR	년도를 나타낸다. 1901년부터 2155년, 0000년을 나타낼 수 있다.
CHAR(M)	고정 길이를 갖는 문자열을 저장할 수 있다. M은 1 부터 255 까지 이다.
VARCHAR(M)	CHAR는 고정 길이인 반면 VARCHAR는 가변 길이이다.
TINYBLOB, TINYTEXT	255개의 문자를 저장할 수 있다.
BLOB,TEXT	63,535개의 문자를 저장할 수 있다.
MEDIUMBLOB, MEDIUMTEXT	16,777,215개의 문자를 저장할 수 있다.
LONGBLOB, LONGTEXT	4,294,967,295(4기가)개의 문자를 저장할 수 있다.

테이블 생성

- CREATE TABLE 문 이용
- Syntax

```
CREATE TABLE [schema.]table_name
    (column datatype [NULL|NOT NULL][DEFAULT expr][AUTO_INCREMENT]
                     [column_constraints],
        ......,
    [PRIMARY KEY(column_name)]
    [table_constraints]);
```

  - 데이터 형 이외에도 속성값의 빈 값 허용 여부는 NULL 또는 NOT NULL로 설정
  - DEFAULT 키워드와 함께 입력하지 않았을 때의 초기값을 지정할 수 있음
  - 입력하지 않고 자동으로 1씩 증가하는 번호를 위한 AUTO_INCREMENT

테이블 생성

- 테이블명, 컬럼명, 데이터 타입 등 정의

- [연습] 오른쪽 DDL 쿼리를 이용, book 테이블을 만들어 봅시다
- DESC 명령을 이용, 생성된 테이블이 원하는 구조대로 만들어졌는지 확인해 봅니다

```
CREATE TABLE book (
    book_id INT(10),
    title VARCHAR(50),
    author VARCHAR(20),
    pub_date DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

book_id	title	author	pub_date
1	토지	박경리	2005-03-12
2	슬램덩크	다케이코	2006-04-05
…	…	…	…

Subquery를 이용한 테이블 생성

- Subquery의 결과와 동일한 테이블 생성됨
- 질의 결과 레코드들이 포함됨
- NOT NULL 제약 조건만 상속됨

```
CREATE TABLE  account_employees
AS (
    SELECT  *
        FROM  employees
     WHERE  job_id = 'FI_ACCOUNT'
);
```

Naming Rules

- 테이블, 컬럼 등의 이름 명명 규칙
  - 문자로 시작
  - 64자 이내 (Alias는 256자 이내)
  - A-Z, a-z, 0-9, _, $
  - 동일 데이터베이스 내 다른 객체의 이름과 겹치지 않아야 함
    (다른 데이터베이스의 객체와는 같을 수도 있음)

ALTER TABLE

- 컬럼 추가 (ADD)
  - `ALTER TABLE book ADD (pubs VARCHAR2(50));`
- 컬럼 수정 (MODIFY)
  - `ALTER TABLE book MODIFY title VARCHAR2(100);`
- 컬럼 삭제 (DROP)
  - `ALTER TABLE book DROP author;`

기타 테이블 관련 명령

- 테이블 삭제 (DROP TABLE)
  - `DROP TABLE book;`
- 데이터 삭제 (TRUNCATE TABLE)
  - `TRUNCATE TABLE book;`
- Comment
  - `ALTER TABLE book COMMENT 'this is comment';`
- RENAME
  - `RENAME TABLE book TO article;`
- ROLLBACK 대상이 아님

Constraint(제약조건)

- Database 테이블 레벨에서 특정한 규칙을 설정함
- 예상치 못한 데이터의 손실이나 일관성을 어기는 데이터의 추가, 변경 등을 예방

- 종류
  - NOT NULL
  - UNIQUE
  - PRIMARY KEY / FOREIGN KEY
  - CHECK
  - DEFAULT

제약조건 정의

- Syntax
  - CREATE TABLE 테이블명 (
    컬럼명 DataType [DEFAULT 기본값][컬럼 제약 조건],
    컬럼명 DataType [DEFAULT 기본값][컬럼 제약 조건],
    …
    [테이블 제약조건] …);
  - 컬럼 제약 조건 : [CONSTRAINT 이름] constraint_type
  - 테이블 제약 조건 : [CONSTRAINT 이름] constraint_type(column, …)

제약조건

- NOT NULL
  - NULL 값을 허용하지 않음
  - 컬럼 형태로만 제약조건 정의할 수 있음(테이블 제약 조건 불가

```
CREATE TABLE book (
    book_id INTEGER NOT NULL
);
```

- UNIQUE
  - 중복된 값을 허용하지 않음 (NULL은 들어올 수 있음)
  - 복합 컬럼에 대해서도 정의 가능
  - 자동적으로 인덱스 생성

```
CREATE TABLE book (
    book_id INTEGER,
    UNIQUE(book_id)
);
```

제약조건

- PRIMARY KEY
  - NOT NULL + UNIQUE (인덱스 자동 생성)
  - 테이블 당 하나만 나올 수 있음
  - 복합 컬럼에 대하여 정의 가능 (순서 중요)

```
CREATE TABLE book (
   ...
   PRIMARY KEY (book_id)
);
```

- CHECK
  - 임의의 조건 검사 조건식이 참이어야 변경 가능
  - 동일 테이블의 컬럼만 이용 가능

```
CREATE TABLE book (
    rate INTEGER CHECK (rate IN (1,2,3,4,5))
);
```

실습

- [연습] book 테이블 생성 (제약조건 포함)
  - 제약조건을 포함하여 book 테이블을 생성해 봅시다

```
1. 컬럼 타입

   author_id : INTEGER
   author_name : VARCHAR(100)
   author_desc : VARCHAR(500)

2. 제약조건

   author_id : PRIMARY_KEY
   author_name : NOT NULL
```

실습

- [연습] author 테이블 생성
  - book 테이블의 author 컬럼을 author 테이블로 분리, 관리하고자 합니다.
  - 다음 조건으로 author 테이블을 생성해 봅시다

```
1. 컬럼 타입

   author_id : INTEGER
   author_name : VARCHAR(100)
   author_desc : VARCHAR(500)

2. 제약조건

   author_id : PRIMARY_KEY
   author_name : NOT NULL
```

실습

- [연습] book 테이블 변경
  - book 테이블의 author 컬럼을 삭제해 봅니다
  - book 테이블에 author_id 컬럼을 추가합니다
    - author 테이블의 author_id 컬럼과 같은 형식(INTEGER)으로 지정합니다

제약조건

- FOREIGN KEY
  - 참조 무결성 제약
  - 일반적으로 REFERENCE 테이블의 PK를 참조
  - REFERENCE 테이블에 없는 값은 삽입 불가
  - REFERENCE 테이블의 레코드 삭제시 동작
    - ON DELETE CASCADE : 해당하는 FK를 가진 참조행도 삭제
    - ON DELETE SET NULL : 해당하는 FK를 NULL로 바꿈

```
CREATE TABLE book (
    …
    author_id INTEGER,
    FOREIGN KEY (author_id)
    REFERENCE author(id)
    ON DELETE SET NULL
);
```

ADD / DROP CONSTRAINTS

- 제약조건 추가
  - ALTER TABLE 테이블명 ADD CONSTRAINT …
  - NOT NULL은 추가 못함

```
ALTER TABLE book ADD CONSTRAINT c_book_fk
   FOREIGN KEY (author_id) REFERENCES author(author_id);
```

- 제약조건 삭제
  - ALTER TABLE 테이블명 DROP 제약조건 제약조건명
  - PRIMARY KEY의 경우 FK 조건이 걸린 경우에는 CASCADE로 삭제해야 함

```
ALTER TABLE book DROP FOREIGN KEY c_book_fk;

ALTER TABLE author DROP PRIMARY KEY CASCADE;
```

  - UNIQUE 제약은 DROP INDEX로 삭제