DAY 2

SQL

1. DML (Data Manipulation Language)

- INSERT
- SELECT
- UPDATE
- DELETE
- 트렌젝션의 영향을 받는다 트렌젝션을 걸 수 있다.

2. DDL (Data Definition Language)

- CREATE
- DROP
- ALTER
- 트렌젝션을 못 건다.

3. DCL (Data Control Language)

- GRANT
- REVOKE
- DENY

4. Transaction (트렌젝션)

- 작업의 최소단위
- 임의로 설정할 수 있다.
- 여러가지 SQL문을 엮어서 하나의 트렌젝션으로 설정할 수 있다.

WHY?? 왜 Transaction을 설정할까??

트렌젝션을 설정하면 ACID라고 불리는 특성이 부여된다.

그냥 SQL문을 작성했을때 나타나는 오류의 여지들을 DBMS가 ACID라고 불리는

특성을 부여해서 자동으로 막아준다.

그렇다면 ACID는 무엇일까??

ACID

- **A (Atomicity)**
  - **원자성**
    - **All or Noting**
- **C (Consistency)**
  - **일관성**
- **I (Isolation)**
  - **독립성**
    - 하나의 트렌젝션이 사용하고있는 데이터는 다른 트렌젝션이 사용하지 못한다.
- **D (Durability)**
  - **영속성**
    - Transaction이 정상적으로 종료되면 해당결과가 2차 저장소에 영구적으로 저장된다.

MySQL Data Type

1. 숫자

- **INT**
  - **4byte**
  - -21억 ~ +21억

```
ALTER TABLE testtbl2 AUTO_INCREMENT = 100; -- 현재 auto_incremetn값을 설정할 수 있다.
INSERT INTO testtbl2(userName,age) VALUES('아이유',30); -- id: 100으로 설정된다.

set @@auto_increment_increment = 5;
INSERT INTO testtbl2(userName,age) VALUES('김다미',30); -- id: 100으로 설정된다.
```

- **BIGINT**
  - **8byte**
  - -900경 ~ +900경
- **FLOAT**
  - **4byte**

- 소수점표현 실수를 근사값으로 계산한다.

- **Double**
  - **8byte**
  - 소수점표현 실수를 근사값으로 계산한다.
- **DECIMAL**
  - 전체 자릿수와 소수점자릿수를 정할 수 있다.

2. 문자

- **CHAR**
  - 고정길이 (1 ~ 255)
  - INSERT, DELETE와 같은 연산을 수행할 때 VARCHAR보다 빠르다.

```
SELECT  CHAR_LENGTH('abcde'),
    CHAR_LENGTH('홍길동'), -- 문자열의 개수를 샐때는 CHAR_LENGTH를 써야한다.
    LENGTH('abcde'),
    LENGTH('홍길동'); -- LENGTH는 byte단위를 샌다 즉 9가 나옴.

SELECT CONCAT ('소리없는', '아우성'), -- 그냥 문자열 연결
    CONCAT_WS('-','2022','02','20'); -- 문자열을 '-'를 이용해서 연결한다.

SELECT FORMAT(1234567.1415234,3); -- 3자리마다 ',' 찍으면서 소수점 아래 3자리까지 반올림하여 표시한다.

SELECT TRIM('          소리없는 아우성                    '), -- 여백 제거
    REPLACE('이것은 소리없는 아우성','소리','양심'),
        SUBSTRING('이것은 소리없는 아우성',3,5); -- 3번째 부터 5글자 가지고 온다. DMBS는 첨자가 1부터 시작한다.

-- 날짜에 관련된 내장함수

SELECT CURDATE(), -- 현재 날짜만 가지고온다.
    NOW(), -- 현재 날짜 + 시,분,초까지 가지고온다.
        YEAR(CURDATE()); -- 현재 년도만 가지고온다.
```

- **VARCHAR**
  - 가변길이 (1 ~ 65535)
- **LONGTEXT**
  - 최대 4G까지 저장가능.(책 한권, 영화 등등)
- **LONGBLOB**
  - **BLOB (Binary Large OBject)**
  - 4G까지 저장가능

```
INSERT INTO movieTBL VALUES(1,'쉰들러 리스트',
  LOAD_FILE('C:/MySQL/Schindler.txt'),
    LOAD_FILE('C:/MySQL/Schindler.mp4')
    );
SELECT * FROM movieTBL;
DELETE FROM movieTBL;
-- 환경설정이 안되어있기 때문에 정상적으로 동작하지 않는다.
-- LOAD_FILE의 크기, 어떤 FOLDER에서 LOADING하는지를 설정해주어야 한다.

SELECT movie_film
FROM movieTBL
WHERE movie_id =1
INTO DUMPFILE 'C:/MySQL/video.output.mp4'
```

- LONGCLOB (문자열)

JOIN

- **Inner Join**
  - **일반적으로 Join 이라고하면 Inner Join을 지칭한다.**

```
USE sqldb;

-- 구매 테이블에서 'JYP'라는 아이디를 가진 사람이 구매한 물건을
-- 발송하기 위해서 이름과 주소가 필요하다. 이 정보는 usertbl에 존재한다.
-- 두 테이블을 결합해서 결과를 알아내야 한다.
-- 이것이 바로 JOIN!!!!!

-- 표준 방식
SELECT buyTBL.userID, name, addr
FROM buyTBL B -- AS와 같이 한칸 띄고 입력하면 입력한 문자로 줄여서 쓸 수 있다.
  INNER JOIN userTBL U -- 이제 userTBL 대신 U를 사용하면된다.
    ON buyTBL.usERID = userTBL.userID
WHERE buyTBL.USerID = 'JYP';

-- 비표준 방식
SELECT buyTBL.userID, name, addr
FROM buyTBL, userTBL
WHERE buyTBL.userID = USERTBL.UserID;
```

- **Outer Join**
  - **Join 조건을 만족하지 않는 행도 포함 시킨다.**
    - **LEFT**
    - **RIGHT**
    - **FULL**

```
-- OUTER JOIN
-- usertbl과 buytbl을 이용해서 다음을 구해보아요!
-- 전체 회원의 구매기록을 조회하세요. 단, 구매기록이 없는 회원도
```

```
-- 출력되어야 해요!

SELECT U.userID, U.name, B.prodName, U.addr
FROM usertbl U
  LEFT OUTER JOIN buytbl B
  ON U.userID = B.userID
ORDER BY U.userID;

-- 한번도 구매한 적이 없는 회원의 목록을 출력하세요.
SELECT U.userID, U.name, B.prodName, U.addr
FROM usertbl U
  LEFT OUTER JOIN buytbl B
  ON U.userID = B.userID
WHERE B.prodName IS NULL -- IS NULL 구문을 이용해서 구매목록이 없는사람을 출력한다.
ORDER BY U.userID;
```

- **Self Join**
- **Cross Join (X)**
    - **dummy data를 생성할 때 사용한다.**