- 출처 : 나의 첫 SQL 수업

SQL 문법 종류

DDL

- Data Definition Language
- 테이블이나 관계의 구조를 생성, 삭제, 변경하는 데 사용
- 테이블/인덱스 등과 같은 데이터베이스 객체(Object)의 구조를 정의하는데 사용

명령어

- CREATE, ALTER, DROP, RENAME

DML

- Data Manipulation Language
- 테이블에 데이터를 조회, 추가, 삭제, 수정하는데 사용

명령어

- SELECT, INSERT, DELETE, UPDATE
- SELECT
  - 데이터베이스에 들어있는 데이터를 조회하거나 검색하기 위한 명령어
- INSERT, UPDATE, DELETE
  - 데이터베이스에 들어있는 데이터에 변형을 가하는 데 사용하는 명령어

DCL

- Data Control Language
- 데이터의 사용 권한을 관리하는 데 사용
- 데이터베이스에 접근하고 객체들을 사용하도록 권한을 부여하고 회수

명령어

- GRANT, REV

TCL

- Transaction Control Language
- 데이터 조작어(DML)에 의해 조작된 결과를 작업 단위(트랜잭션) 별로 제어하는데 사용

명령어

- BEGIN TRAN, COMMIT, ROLLBACK

테이블

- 데이터는 관계형 데이터베이스의 기본 단위인 테이블 형태로 저장
- 모든 데이터는 테이블에 저장되고, 테이블에서 원하는 자료를 조회할 수 있음
- 컬람과 행의 2차원 구조를 가지고 세로 방향을 컬럼(Column), 가로 방향을 행(Row)이라고 함
- 컬럼과 행이 겹치는 하나의 공간을 필드(Field)라고 함
- 정규화

```
테이블을 분할하여 데이터의 정합성을 확보하고, 불필요한 중복을 줄이는 과정
```

- 기본키

```
테이블에 존재하는 각 행을 한 가지 의미로 특정할 수 있는 1개 이상의 커럼
예를 들어 주문 테이블의 기본키로는 주문번호가 될 수 있으며, 고객 테이블의 기본키는 고객번호가 될수 있음
```

- 외래키

```
다른 테이블의 기본키로 사용되고 있는 관계를 연결하는 컬럼
주문 테이블의 고객번호 컬럼은 주문 테이블이 가지고 있는 FK이고, 고객번호 컬럼은 고객 테이블의 기본키
```

ERD

- Entity Relationship Diagram
- 테이블 간 서로의 상관 관계를 그림으로 도식화한 것

DDL

- DB를 구성하고 있는 다양한 객체(사용자, 테이블, 인덱스, 뷰, 트리거, 프로시저, 사용자 정의 함수등)를 정의/변경/제거하는 데 사용
- DDL은 물리적 DB 객체의 구조를 정의하는 데 사용

데이터 타입

- CHAR(L)

```
- 고정길이 문자열
- 고정길이보다 작은 값이 입력될 경우 그 차이만큼 공백으로 채움
```

- VARCHAR(L)

```
- 가변길이 문자열
- L값만큼의 최대 길이를 가짐
- L값보다 작을 경우 입력하는 값만큼만 공간을 차지
- 만약 VARCHAR(5)로 선언하고 'abc'를 입력했다면 그대로 'abc'가 저장
- CHAR 타입은 길이보다 짧은 문자열의 뒷부분을 공백으로 채우는데 비해 가변 길이인 VARCHAR 타입은 문자열 길이만큼만 저장
- VARCHAR형은 용량면에서 유리하고 CHAR형은 길이가 일정해 속도가 빠름
- 우편 번호나 주민등록번호 같이 길이가 일정한 필드에는 CHAR형을 쓰는게 유리
- 영화 제목이나 블로그의 댓글처럼 길이가 다른 문자열은 VARCHAR 타입이 적합
```

- NUMBER(L, D)

```
- 정수 및 실수를 저장
- L값은 전체 자리수, D값은 소수점 자리수
- NUMBER(12, 2)는 999999999999.99 까지 저장할수 있음
```

- Date

```
- 날짜와 시각정보를 저장
- 즉, 년월일시분초를 저장
```

테이블 만들기

- CREATE 구문

```
CREATE TABLE store
(
        name VARCHAR(12) NOT NULL
        .....
)
```

제약조건

- 기본키

```
- 테이블에 저장된 행들 중에서 특정 행을 고유하게 식별하기 위해서 사용
- 하나의 테이블에 단 하나의 기본키만 정의할 수 있다.
- 기본키 생성 시 DBMS는 유일 인덱스 Unique Index를 자동으로 생성한다.
- 기본키 칼럼에는 NULL 입력이 불가능하다
- 기본키는 UNIQUE 제약조건과 NOT NULL 제약조건을 만족해야 한다
(UNIQUE + NOT NULL)
```

- 고유키

```
- 테이블에 저장된 행들 중에서 특정 행을 고유하게 식별하기 위해 생성
- 기본키와의 차이점은 고유키는 NULL 입력이 가능하다는 것
- 고유키는 UNIQUE 제약조건만 만족하면 NULL 입력이 가능하다 (UNIQUE+NULL)
```

• NOT NULL

```
- 해당 칼럼에는 NULL 입력을 금지하므로 어떤 값이라도 입력되어야 함
- 이 칼럼은 항상 값을 필수적으로 받아야함
```

- CHECK

```
- 입력할 수 있는 값의 종류 혹은 범위를 제한
```

- 외래키

```
- 다른(부모 혹은 참조) 테이블의 기본키를 외래키로 지정하는 경우 생성
- 참조 무결성 제약조건이라고 함
```

• 기본값, 디폴트 값

```
- 해당 칼럼에 아무런 값도 입력하지 않았을 때 지정한 디폴트 값으로 데이터가 입력된다.
```

테이블 생성 예

```sql
CREATE TABLE `naver_day_stock` (
        `st_code` VARCHAR(10) NOT NULL,
        `st_date` DATE NOT NULL,
        `st_open` INT(11) NULL DEFAULT NULL,
        `st_high` INT(11) NULL DEFAULT NULL,
        `st_low` INT(11) NULL DEFAULT NULL,
        `st_close` INT(11) NULL DEFAULT NULL,
        `st_trade` BIGINT(20) NULL DEFAULT NULL,
```

```
        PRIMARY KEY (`st_code`, `st_date`)
);
```

ALTER

- ALTER TABLE은 컬럼 및 제약조건을 추가/수정/제거하는데 사용

이름 변경

컬럼 추가

```
ALTER TABLE table_name ADD COLUMN ex_column varchar(32) NOT NULL;
```

컬럼 제거

```
ALTER TABLE table_name DROP COLUMN ex_column;
```

컬럼 변경

- 컬럼명 변경하면서 NULL 및 NOT NULL 설정

```
-- type 변경
ALTER TABLE table_name MODIFY COLUMN ex_column varchar(16) NULL;
-- 한글 설정
ALTER TABLE table_name MODIFY COLUMN ex_column varchar(16) CHARACTER SET utf8mb4
COLLATE utf8mb4_general_ci NOT NULL
-- 이름 변경
ALTER TABLE table_name CHANGE COLUMN ex_column ex_column2 varchar(16) NULL;
```

컬럼 위치 변경

```
ALTER TABLE table_name MODIFY column_name data_type AFTER another_column_name;

-- ALTER TABLE users MODIFY age INT AFTER name;  -- age 컬럼을 name 컬럼 뒤로 이동
```

pk 생성

```
# pk 한개
ALTER TABLE table_name ADD PRIMARY KEY (column_name);

# pk가 다수
ALTER TABLE table_name ADD PRIMARY KEY (column1, column2, ...);
```

pk 제거

```
ALTER TABLE table_name DROP PRIMARY KEY;
```

외래키 제거

외래키 생성

테이블명 변경

```
ALTER TABLE table_name1 RENAME table_name2;
```

테이블 제거

DML

- Data Manipulation Language
- 데이터를 입력/수정/삭제/조회하는 역할을 함
- INSERT, UPDATE, DELETE, SELECT 문
- 입력, 수정, 삭제 구문 실행 후 영구적으로 저장(COMMIT)하거나 취소(ROLLBACK) 할 수 있음

삽입

```
-- 1
INSERT INTO 테이블명 (컬럼명1, 컬럼명2, ....) VALUES (vaule1, value2, ....);

-- 2
INSERT INTO 테이블명 VALUES (value1, value2, ....);

-- 3
INSERT IGNORE INTO 테이블명 VALUES (value1, value2, ....);
```

- 해당 컬럼명과 입력하는 값을 서로 1:1로 매핑해서 입력
- 데이터 타입이 문자형일 경우 single quotation을 이용하여 값을 입력, 숫자는 붙이지 않아도 됨
- 1번의 경우 primary key나 not null로 지정되어 있지 않은 컬럼은 default로 null 값이 입력된다.
- 컬럼의 순서는 테이블의 컬럼 순서와 일치하지 않아도 됨
- 2번의 경우 모든 컬럼에 데이터를 입력하는 경우로 컬럼의 순서대로 빠짐없이 데이터가 입력되어야 함
- 3번의 경우는 오류가 발생해도 무시하고 넘어가는 기능

갱신

삭제

조회

데이터 조회

조회에 사용되는 문법

- SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY를 학습
  - SELECT절은 최종 결과물로 확인하고 싶은 칼럼들의 이름을 쉼표(,)로 구분해 작성
  - FROM절은 조회할 데이터가 적재된 테이블의 이름을 작성
  - WHERE절은 FROM절에서 설정한 테이블의 데이터 중 특정한 데이터를 보기 위한 필터링 조건을 작성
  - GROUP BY절은 그룹화를 진행할 특정 기준을 작성
  - HAVING절은 그룹화 이후에 필터링 조건을 작성(반드시 GROUP BY와 함께 사용됨)
  - ORDER BY절은 조회한 결괏값을 오름차순과 내림차순으로 정렬할 기준을 작성

SQL 처리 순서	SQL 작성 순서	SELECT 구문 사용 예
5	1	SELECT shop_code, COUNT(*)
1	2	FROM store
2	3	WHERE country = 'JAPAN'
3	4	GROUP BY shop_code
4	5	HAVING count(*) >= 2
6	6	ORDER BY shop_code

- 처리순서

SELECT ~ FROM

- SELECT는 최종 결과물을 조회하고 싶은 컬럼을 쉼표로 구분해 작성
- SELECT절에 작성한 컬럼들의 순서는 조회하고 싶은 결과와 동일한 순서로 작성
- FROM절은 조회하고 싶은 데이터가 적재된 테이블명을 작성

```
-- 특정 컬럼만 조회
SELECT 컬럼1, 컬럼2, ..... FROM 테이블명
-- 모든 컬럼 조회
SELECT * FROM 테이블명
```

- "*"는 모든 컬럼을 보여달라는 의미

ALIAS(AS)

DISTINCT

- 특정 컬럼을 중복 제거하기 위해서 distinct 함수를 사용

- 중복 제거하고 이후의 값을 count 함수를 사용하여 집계

```
SELECT COUNT(DISTINCT column_name)
FROM table_name;
```

WHERE

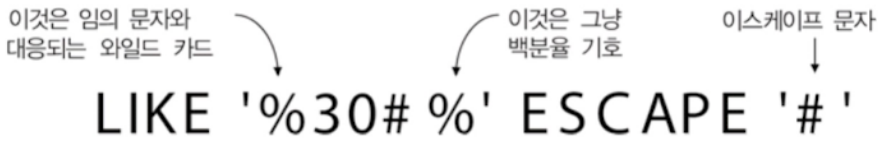
- 조건을 지정
- 조건을 지정하지 않으면 모든 레코드를 조회
- Database의 Table의 크기가 큰 데이터는 전체 조회를 할 경우 시스템의 부하를 일으킴

LIKE

- LIKE 연산자는 패턴으로 부분 문자열을 검색

기호

- %
  - 복수개의 문자와 대응한다. 도스의 *와 동일한 의미를 가지며 %자리에는 임의 개수의 임의 문자가 올 수 있다.
-
  - 하나의 문자와 대응한다. 도스의 ?와 동일한 의미를 가지며_자리에 하나의 임의 문자가 올 수 있다.
- '[ ]'
  - []안에 포함된 문자 리스트 중 하나의 문자와 대응한다.
- [^]
- 안에 포함된 문자 리스트에 포함되지 않은 하나의 문자와 대응한다.
- %문자 쓰기 - %를 와일드카드로 인식하기 때문에 아래와 같이 사용하면 % 문자 자체로 인식함



BETWEEN

CASE

- 로그 데이터 또는 업무 데이터로 저장된 코드 값을 그대로 집계에 사용하면 리포트의 가독성이 떨어짐
- 리포트를 작성할 때 변환하는 등의 작업이 필요

group by

- 특정 컬럼을 기준으로 집계 함수를 사용하여 count, sum, avg 등의 집계성 데이터를 추출할 때 사용

partition by

- parition이란 논리적으로 하나의 테이블이지만 실제로는 여러 개의 테이블로 분리해서 관리하는 기능을 의미
- group by와 집계 함수가 하는 역할이 비슷하지만

HAVING

- The `HAVING` clause was added to SQL because the `WHERE` keyword cannot be used with aggregate functions.
- 예제 1

```
SELECT column_name(s)
FROM  table_name_
WHERE  condition
GROUP BY column_name(s)
HAVING  condition
ORDER BY _olumn_name(s);
```

- 예제2

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5;
```

함수

- SQL 함수는 테이블의 컬럼을 가공해 결괏값을 생성
- SQL 함수는 관점에 따라 부르는 명칭과 종류가 조금씩 다름
- MySQL에서 데이터 형식의 종류는 약 30개 정도 된다.
- DBMS마다 같은 기능의 함수 이름이 동일한 것도 있지만 다른 이름으로 구현된 함수도 존재

문자열 함수

문자 붙이기

- CONCAT, CONCAT_WS(구분자, 문자열1, 문자열2,...)
- 문자열을 이어줌
- CONCAT(문자열1, 문자열2, ...)

- CONCAT(구분자, 문자열1, 문자열2, ...)

```
SELECT CONCAT_WS('/', '2025', '01', '01')
```

문자 찾기

- ELT(위치, 문자열1, 문자열2, ...)
- FIELD(찾을 문자열, 문자열1, 문자열2, ...)
- FIND_IN_SET(찾을 문자열, 문자열 리스트)
- INSTR(기준 문자열, 부분 문자열)
- LOCATE(부분 문자열, 기준 문자)

숫자 함수

날짜 함수

순위 함수

CASE문

데이터 집계

데이터 정렬