SQL : 데이터베이스를 관리하기 위한 언어

DDL(Data Definition Language) : 데이터 정의어

- 데이터베이스나 테이블을 생성, 수정, 삭제

ex)학생 테이블, 과목 테이블, 교수 테이블을 생성

테이블 - 엔티티(Entity), 속성(Attribute)을 묶은 그룹

CREATE SCHEMA `university` DEFAULT CHARACTER SET utf8 ;

-> utf8 캐릭터셋의 university 스키마를 생성

DML(Data Manipulation Language) : 데이터 조작어

- 데이터를 검색, 추가, 삭제, 수정

CRUD(create, read, update, delete)

ex)학생1 정보 추가, 학생1 정보 수정, 학생1 정보 검색 등

기본키 - 객체를 식별할 수 있는 고유한 값

-> 중복불가, null값 사용불가 (not null)

read 명령어

select 속성명 from 테이블명 [where 조건(속성명=’속성값’)] || [order by 속성명 asc|desc]

필수요소 [선택 조건 : 있을 수도 있고 없을 수도 있다]

-> 있다면 위의 형태로 와야 한다

ex) where gender = ‘여’;

-> gender가 ‘여’인 객체만 출력

order by age asc;

-> age를 기준으로 오름차순 정렬

-> asc : 오름차순, desc : 내림차순

order by 미입력시 기본 asc 오름차순

select 속성명 : 결과에서 보여줄 속성들을 나열한다

select \* : 결과에서 모든 속성들을 나열한다

from 테이블명 : 해당 테이블에서 검색을 한다

ex)select \* from university.student;

데이터베이스 university에 있는 student 테이블에서 모든 객체의 모든 속성을 보여준다

->테이블명 뒤에 [where 조건]이 없는 경우 모든 객체를 의미한다

create 명령어

insert into 테이블명(컬럼1, 컬럼2, … 컬럼n)

values(컬럼값1, 컬럼값2, … 컬럼값n);

insert시 고려해야 할 사항들

1. 기본키의 속성은 생략할 수 없다 (null값을 가질 수 없기 때문에)
2. 기본키는 중복되면 안된다 (고유 식별자이기 때문에)
3. 각 속성의 제약 사항을 지켜야 한다(최대 글자수, 자료형)

예제) select \* from university.student where name = ‘논개’;

위 코드를 실행했을 때 나올 수 있는 값의 개수는? 3

1. 1개
2. 0개 또는 1개 (조건에서 비교하는 속성이 기본키일 경우)
3. 0개 또는 여러개(조건이 기본키가 아닌 속성인 경우)
4. 여러개

select \* from university.student where birthday is null;

null값의 속성을 검색하는 경우 = 이 아닌 is를 사용

select distinct 속성명 from 테이블명;

속성명 앞에 distinct를 입력하면 중복되는 결과를 제거하고 결과를 출력

->검색할 속성이 기본키이면 중복될 수 없기 때문에 사용하지 않아도 된다

where절의 연산자

and : where절에서 조건이 2개 이상인 경우 결과가 둘다 참인 경우에 사용

or : 조건이 2개 이상인 경우에서 결과가 하나만 참이어도 되는 경우에 사용

not : 결과가 거짓일 때 사용

-> != , <>와 유사

비교 연산자 <, >, <=, >=

-주로 정수형 값을 비교할 때 사용

select \* from student

where student\_grade = 1 or (student\_grade = 4 and student\_name = '임꺽정');

학년이 1인 학생들과 4학년이면서 이름이 임꺽정인 학생들의 모든 속성들을 나타낸다

like - 원하는 문자열이 있는지를 확인할 때 사용

->그냥 사용하면 =과 같지만 %를 추가하면 정확히 일치하지 않아도 값을 검색할 수 있다

where 속성명 like ‘a%’ : a로 시작하는 모든 문자열을 검색

where 속성명 like ‘%a’ : a로 끝나는 모든 문자열을 검색

where 속성명 like ‘%a% : a가 들어가는 모든 문자열을 검색

ex) SELECT \* FROM university.student where student\_name like '%마%';

-> student\_name테이블에서 ‘마’가 들어간 결과를 검색

update 테이블명

set 속성 = 수정할 값

where 수정할 대상의 기본키;

->중복된 다수의 대상이 함께 수정되는 걸 방지하기 위해 고유값을 가지는 기본키를 검색

where 기본키 >= 1;

-> 개체 전부를 한번에 수정

where 기본키 >= 1 and 변경할 대상

-> 전체 중에서 변경할 대상을 선택해서 한번에 수정;

delete from 테이블명

where 기본키; -> 기본키가 없을 경우 다른 컬럼값으로 사용 가능

->대상을 삭제

정렬

select \* from 테이블명 order by grade asc, age asc;

-> grade를 오름차순으로 정렬한 뒤 age를 오름차순으로 정렬

where절의 함수

-> 값이 정수형인 속성을 계산할 때 사용

count(\*), count(속성명) : 검색 결과의 개수를 알려주는 함수

ex)select count\* from 테이블명

-> 해당 테이블의 모든 객체수를 알려준다

sum(속성명) : 검색 결과의 합을 알려주는 함수

ex)select sum(grade) from 테이블명

->해당 테이블의 grade값의 합을 알려준다

avg(속성명 또는 수식) : 검색 결과의 평균을 알려주는 함수

ex)select avg(grade) from 테이블명

->해당 테이블의 grade값의 평균을 알려준다

min(속성명) : 검색 결과의 최소값을 알려주는 함수

ex)select min(grade) from 테이블명

->해당 테이블의 grade값의 최소값을 알려준다

max(속성명) : 검색 결과의 최대값을 알려주는 함수

ex)select max(grade) from 테이블명

->해당 테이블의 grade값의 최대값을 알려준다

join절

select 테이블명.속성명, 테이블명.속성명 나열한 테이블명.속성명들을 출력한다

from advise advise 테이블에서

join professor professor 테이블을 연결한 뒤

on

advise.advise\_professor\_num = professor.professor\_num;

adivse 테이블의 advise\_professor\_num과 professor테이블의 professor\_num의 기본키값이 같다면

select에서 선택한 속성명들을 모두 나타낸다

select 속성명 from 실행할 테이블명

join 불러올 테이블명

on 기준 테이블명.기본키or외래키 = 참조 테이블명.기본키or외래키

->select에서 여러 테이블을 참조할 시 join on, join on을 중복 사용한다

-> on 뒤에 where절을 추가해서 조건을 제한할 수 있다

select \* from 테이블1(외래키가 있는 테이블)

join 테이블2(테이블1과 연결된 기본키를 가진 테이블)

on 테이블1.외래키명 = 테이블2.기본키명

외래키를 통해 다른 테이블의 정보를 참조해서 가져올 필요가 없다면 굳이 외래키로 지정하지 않아도 된다

ex) 게시글의 작성자 id를 외래키로 지정할 경우 해당 id의 회원이 존재하는지 확인하지 않아도 되지만

id외의 작성자의 다른 정보를 참조해서 가져올 필요가 없기에 외래키로 지정하지 않는다

외래키의 외래키를 통해 테이블 속성 정보를 가져오는 방법

-> 직접적인 연결이 아닐 때

방법1)

select course.course\_class\_num,

classdetail.subject\_title,

course.course\_student\_num,

student.student\_name,

course.course\_total

from course

join (select \* from class join subject on class.class\_subject\_code = subject.subject\_code) as classdetail

on course.course\_class\_num = classdetail.class\_num

join student

on course.course\_student\_num = student.student\_num;

-> class의 외래키 subject\_code를 통해 subject 테이블을 참조할 때 join에 select문을 작성한 뒤

as를 통해 이름을 다시 붙여주고 붙여준 이름을 통해 on을 작성한다

방법2)

select course.course\_class\_num,

subject\_title,

course.course\_student\_num,

student.student\_name,

course.course\_total

from course

join student

on course.course\_student\_num = student.student\_num

join class

on course.course\_class\_num = class.class\_num

join subject

on subject.subject\_code = class.class\_subject\_code;

-> 방법2로 join절을 순서대로 사용해 테이블을 연결한 뒤 속성을 가져와도 결과는 같지만

방법2는 이어진 테이블의 모든 속성을 가져오기 때문에 테이블들의 속성이 많을 경우 불필요한 정보가 많아져

방법1을 사용해서 필요한 정보들만을 가져오는 게 데이터베이스의 크기가 클수록 더 효율적이다

left(‘sfkskfd’, 3) -> varchar의 값 중 앞에서 3자리만 가져온다

ex) ‘sfk’