예제1)강의 테이블에는 과목 코드와 교수번호가 있고

현재 강의 테이블을 기준으로 아래와 같이 출력되도록 SQL문을 작성

->강의번호 과목명 교수번호 강의실 일정 년도 학기

SELECT

class.class\_num, subject.subject\_title, class.class\_professor\_num, class.class\_room,class.class\_schedule, class.class\_year, class.class\_term

FROM class

join subject

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

강의번호 과목코드 교수명 강의실 일정 년도 학기

SELECT class.class\_num, class.class\_subject\_code, professor.professor\_name, class.class\_room,class.class\_schedule, class.class\_year, class.class\_term

FROM class

join professor

on class.class\_professor\_num = professor.professor\_num;

SELECT

class.class\_num,

subject.subject\_title,

professor.professor\_name

FROM class

jOIN subject

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

join professor

ON class.class\_professor\_num = professor.professor\_num;

select에서 여러 테이블을 참조할 땐 join과 on을 중복해서 써준 뒤 마지막에만 ;을 붙인다

예제2)강의 테이블에서 과목명과 교수이름이 출력되도록 쿼리문을 작성

select class.class\_num, subject.subject\_title, professor.professor\_name

from class

join subject

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

join professor

on class\_professor\_num = professor.professor\_num;

교수명이 유관순인 강의만 출력되게 하는 경우 join on절 다음 where절을 입력

select class.class\_num, subject.subject\_title, professor.professor\_name

from class

join subject

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

join professor

on class\_professor\_num = professor.professor\_num

where professor.professor\_name = '유관순';

insert할 때 에러가 발생하는 경우

테이블에 외래키가 없는 경우는 쿼리문이 잘못되지 않는 한 에러가 거의 발생하지 않는다

- 기본키가 중복되게 입력된 경우

- Not Null로 체크한 속성에 Null값을 입력한 경우

- 자료형이 맞지 않게 입력한 경우

테이블에 외래키가 있는 경우(A테이블이 B테이블과 연결)

- 외래키에 입력한 속성값이 B테이블의 기본키값에 없는 경우

ex) 강의 테이블에 2019001이라는 교수번호를 입력했는데 교수번호에 2019001이 없는 경우

예제) a학생의 강의 정보를 출력

select course.course\_num, student.student\_num, student.student\_name, class.\*

from course

join class

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

join student

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

where student.student\_num = '201910101';

course의 수를 출력

select count(course.course\_num), student.student\_num, student.student\_name, class.\*

from course

join class

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

join student

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

where student.student\_num = '201910101';

테이블명이 중복되지 않는 경우 테이블명.이 생략 가능하다

예제) 성적들을 각각 비율로 환산하여 총점을 계산

select avg(course\_mid\*0.4 + course\_end\*0.4 + course\_attendance\*0.1 + course\_report\*0.1)

from course

join class

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

join student

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

where student.student\_num = '201910100';

course의 total을 수식으로 계산해서 변경하는 방법

update course

set course.course\_total =

course\_mid\*0.4 + course\_end\*0.4 + course\_attendance\*0.1 + course\_report\*0.1

where course\_num >= 1;

->where course\_class\_num = 2;

course\_class\_num이 2인 모든 course\_num들의 정보만 변경한다

자주 발생하는 에러

Error Code : 1054. Unknown …

-> 오타가 난 경우 컬럼을 찾을 수 없는 경우

1366 : incorrect integer value: … (정수형 타입으로 설정한 뒤 다른 자료형을 입력)

-> 자료형을 다르게 입력했을 경우

Cannot add or update a child row : a foreign key constraint fails

-> 없는 외래키를 입력했을 경우

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

예제) A학생의 평균을 출력

select course.course\_class\_num,

subject\_title,

course.course\_student\_num,

student.student\_name,

avg(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

where student\_name like '%니%';

예제) A학생의 평균 평점을 출력 (4.5 만점 기준)

select course.course\_student\_num,

student.student\_name,

avg(course.course\_total\*0.045) as average //또는 total / 100 \* 4.5

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

where student\_name like '%니%';

------------------

예제)신입생들의 학생수를 출력하는 쿼리문을 작성

select count(student\_num)

from student

where student\_grade = 1 and student\_num like '2019%';

예제)A학생의 2019년도 1학기 수강 과목들을 출력

select subject\_detail.subject\_title,

student.student\_num,

student.student\_name,

class.class\_year,

class.class\_term

from course

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

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

join student

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

join class

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

where student.student\_num = '201910100' and class.class\_year = '2019' and class.class\_term = 1;

예제)A학생이 수강한 전공 필수 과목들을 출력

select subject\_detail.subject\_title,

student.student\_num,

student.student\_name,

class.class\_year,

class.class\_term,

subject\_detail.subject\_type

from course

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

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

join student

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

join class

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

where student.student\_num = '201910100' and subject\_detail.subject\_type = '전공필수';

예제)A학생의 2019년도 1학기 수강과목들과 성적을 출력

select student.student\_num,

student.student\_name,

student.student\_grade,

class.class\_year,

class.class\_term,

course.course\_total

from course

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

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

join student

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

join class

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

where student.student\_num = '201910100' and class.class\_year = '2019' and class.class\_term = 1;

예제)A학생의 2019년도 1학기 수강 평균을 4.5점 만점 기준으로 환산해서 출력

select student.student\_num,

student.student\_name,

student.student\_grade,

class.class\_year,

class.class\_term,

avg(course.course\_total\*0.045) as '1학기 총점'

from course

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

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

join student

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

join class

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

where student.student\_num = '201910100' and class.class\_year = '2019' and class.class\_term = 1;

예제) 학번이 201910100인 학생이 수강한 수강목록과 학생 이름을 출력

방법1)

SELECT course\_class\_num as 강의번호,

student\_name as 학생이름,

course\_total as 총점

FROM course

join student

on course\_student\_num = student\_num

where course\_student\_num = 201910100;

방법2)

select student\_name, c.\*

from (select course\_class\_num, course\_total from course where course\_student\_num = 201910100) as c

join student

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

예제) 학번이 201910100인 학생이 수강한 목록과 학생 이름을 출력하는 쿼리를 join을 이용해 작성

-> 1.학생이름, 강의명, 총점

방법1)

SELECT student\_name as 학생이름,

class\_detail.subject\_title as 강의명,

course\_total as 총점

FROM course

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

on course\_class\_num = class\_num

join student

on course\_student\_num = student\_num

where course\_num >= 1 and course\_student\_num = 201910100;

방법2)

SELECT student\_name,

s.subject\_title,

c.course\_total

FROM (select course\_class\_num, course\_total, course\_student\_num

from course where course\_student\_num = 201910100) as c

join student

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

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

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

-> 2. 학생이름, 강의명, 교수이름, 총점

방법1)

SELECT student\_name,

s.subject\_title,

p.professor\_name,

c.course\_total

FROM (select course\_class\_num, course\_total, course\_student\_num

from course where course\_student\_num = 201910100) as c

join student

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

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

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

join (select \* from class join professor on class\_professor\_num = professor\_num) as p

on c.course\_class\_num = p.class\_num;

방법2)

SELECT student\_name,

subject\_title,

professor\_name,

c.course\_total

FROM (select \* from course where course\_student\_num = 201910100) as c

join student

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

join class

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

join subject

on class\_subject\_code = subject\_code

join professor

on class\_professor\_num = professor\_num;

예제) 트리거 생성

1.student 테이블에서 insert 이벤트가 발생하면 추가된 student\_num의 정보를 이용하여

graduation 테이블에 insert를 실행

drop trigger if exists input\_graduation;

delimiter //

create trigger input\_graduation

after insert on student

for each row

begin

declare r\_num int;

declare r\_year int;

set r\_year = new.student\_num / 100000;

if new.student\_major\_num is not null then

set r\_num = (select requirement\_num from requirement where

new.student\_major\_num = requirement\_major\_num and requirement\_year = r\_year);

end if;

insert graduation(graduation\_student\_num, graduation\_requirement\_num)

values(new.student\_num, r\_num);

end //

delimiter ;

2.course 테이블에 update 이벤트가 발생하면 course\_grade가 null이 아닐 때 (성적이 입력되면)

graduation 테이블의 총 이수학점(graduation\_total)을 업데이트한다

해당 학생이 졸업 요건 정보를 만족하면 graduation\_ok를 Y로 업데이트한다

3.subject의 학점(subject\_point)가 수정되면 해당 과목을 수강한 학생들의 총 이수학점을 수정한다

프로시저 생성

1.학생 번호가 주어지면 해당 학생의 졸업 가능 여부를 알려주는 프로시저