# world 데이터베이스에는 테이블이 3개가 있다.

각각의 테이블에 저장된 데이터를 모두 읽어서 화면에 출력하세요.

# 수도와 함께 국가 이름을 검색합니다.

Use world;

Select a.name, b.name

From city a, country b

Where

# 국가 코드라 'KOR'인 도시를 찾아 인구수를 역순으로 표시하세요.

도시명, 인구수

# 미국(USA)에 있는 모든 도시의 이름을 검색합니다.

USE WORLD;

SELECT a.NAME, a.population, b.name

FROM city a, country b

WHERE a.countrycode = 'usa'

and a.countrycode = b.code;

use world;

SELECT a.name, b.name

FROM country a, city b

WHERE code='usa'

and a.code = b.countrycode;

3. 한국 수도 찾기

use world;

SELECT a.name, b.name

FROM country a, city b

where a.code = 'kor'

and a.capital = b.id;

# 중국에서 사용되는 모든 언어의 이름을 검색합니다.

use world;

SELECT a.name, b.language

FROM country a, countrylanguage b

where a.code = 'chn'

and a.code = b.countrycode;

# 인구가 100만 명 이상인 유럽 국가의 이름을 검색합니다.

use world ;

SELECT name, Population, continent

FROM country

where continent='europe'

and population >=1000000;

use world;

SELECT a.name, b.language, a.Continent, a.population

FROM country a, countrylanguage b

WHERE continent = 'Europe'

and a.code = b.countrycode

and population >= 1000000;

# 1900년 이후 독립한 국가의 이름을 검색합니다.

use world;

SELECT name, IndepYear

FROM country

where IndepYear>=1900;

# 영어가 공용어가 아니고 국민총생산(GNP)이 1,000보다 큰 국가의 이름을 검색합니다.

use world;

SELECT a.name, b.language, gnp

FROM country a, countrylanguage b

where b.language != 'Engish'

and a.code = b.countrycode

and gnp >=1000

order by gnp asc;

# 세계에서 인구가 가장 많은 상위 5개 도시의 이름과 인구를 검색합니다. 결과를 인구별로 내림차순으로 정렬합니다.

use world;

SELECT a.name, b.name

FROM country a, city b, coun

where city

order by population desc;

use world;

select name, population

from city

order by population desc

limit 5

컴퓨터 공학과 교수님들 목록

use university;

SELECT \*

FROM instructor

where dept\_name = 'Comp. Sci.' ;

use university;

컴공 예산

SELECT I.\*, d.budget

FROM instructor I, department d,

where I.dept\_name = 'Comp. Sci.'

and I.dept\_name = d.dept\_name;

컴공 학생들 목록 학점 최대 최소 평균

-- select max(tot\_cred)

-- select min(tot\_cred)

-- select dept\_name, avg(tot\_cred) as avg\_credit

select dept\_name, count(\*) as cnt

from student

-- where dept\_name = 'Comp. Sci.';

group by dept\_name

order by cnt;

가장 많은 교수님이 소속된 학과명은?

use university;

SELECT dept\_name, count(\*) as cnt

FROM instructor

group by dept\_name

order by cnt desc;

학과별 과목수 → 가장 많은 과목 학과

학과별 학점수 → 가장 많은 학점 학과

SELECT s.\*, c.title

FROM section s, course c

where s.course\_id = c.course\_id

limit 10

연도별/학기별 개설과목 수

SELECT s. year, s.semester, count(\*) as cnt

FROM section s

group by year, semester;

2008년도 spring 학기에 network을 수강한 학생 목록은?

SELECT \*, count(\*) cnt

FROM takes

where year = 2008

and semester = 'spring'

and course\_id = 349

and sec\_id = 1 ;

SELECT \*

select count(\*)

FROM takes t, course c

where c.title = 'networking'

and c.course\_id = t.course\_id

and year = 2008

and t.semester = 'spring';

SELECT t.\*

FROM takes t, course c

where c.title like 'net%'

and c.course\_id = t.course\_id

and year = 2008

and t.semester = 'spring';

1. 직원 id가 el인 직원을 검색해보세요

use warehouse;

SELECT \*

FROM emp

where eno='e1';

2. 직원명이 Finzi인 직원을 검색해 보세요.

use warehouse;

select \*

from emp

where ename='Finzi';

3. 급여가 $40,000 이상인 직원을 검색해보세요

use warehouse;

select \*

from emp

where salary>=40

use pnu;

SELECT c.교과구분, c.sno, s.주관학과명, s.sname, i.교수명, c.학점

FROM course c, subject s, instructor i

where c.교과구분 = '교양선택' --전공기초, 전공선택, 전공필수, 교양필수, 일반선택 등등

and c.sno = s.sno

AND c.ino = i.ino;

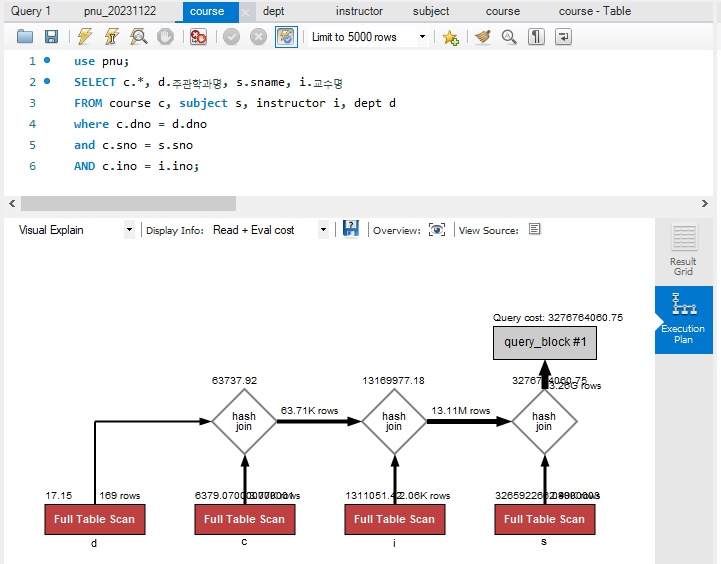
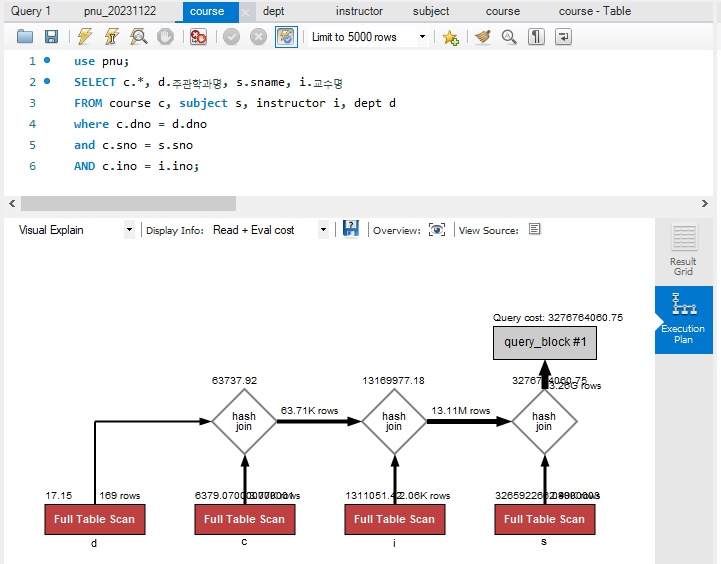
Dept, ino -> alter table : sno, ino에 pk 체크

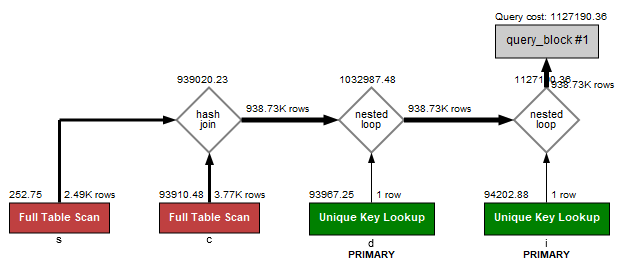
SELECT sno, count(\*) cnt

FROM pnu.subject

group by sno

having count(\*) > 1;





SELECT sno, count(\*) cnt

FROM pnu.subject

group by sno

having count(\*) > 1;

SET sql\_safe\_updates = 0;

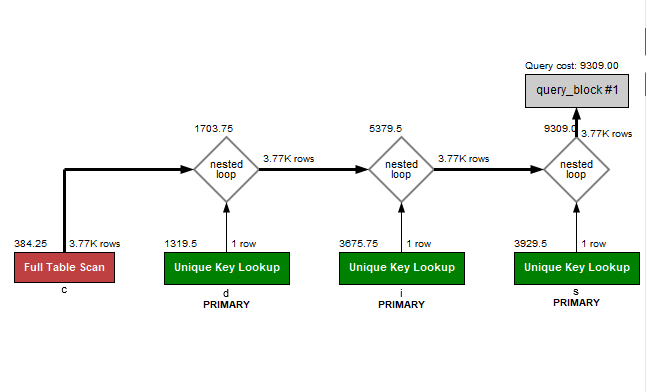
delete from `pnu`.`subject`

where sno in ('XA4000346', 'XA4000351');

-- where sno in ('XA4000343', 'ZE1000043');

set sql\_safe\_updates = 1;

subject -> alter table : sno에 pk 체크

그러면 싹 파랗게 바뀌던데 내 건 이렇게만 됨. 왜지

use pnu;

SELECT c.\*, d.주관학과명, s.sname, i.교수명

FROM course c, subject s, instructor i, dept d

where c.cno = 1

and c.dno = d.dno

and c.sno = s.sno

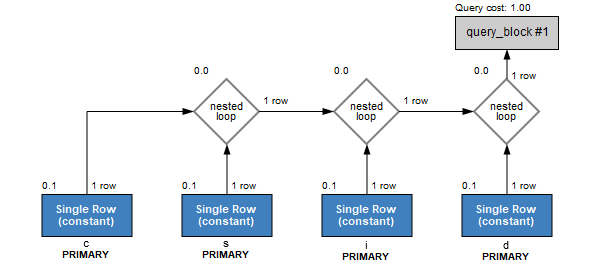
AND c.ino = i.ino;

Where c.cno = 1 추가하고

Course -> alter table : cno에 pk 체크

하니까 이렇게 싹 파랗게 바뀜

결과값도 다르게 나와야 이렇게 파랗게 바뀌는 거였음.



SELECT concat(d.주관학과명, '-', s.sname) as name, i.교수명, c.\*

아니면 LEft(d.주관학과명, 1) as name,

아니면 right(d.주관학과명, 2) as name,

아니면 length(d.주관학과명) as name, i.교수명, lower(c.sno) sno, c.\*

FROM course c, subject s,

dept d, instructor i

where c.cno = 1

and c.dno = d.dno

and s.sno = c.sno

and c.ino = i.ino;

SELECT reverse('010-1111-2222'), length(d.주관학과명) as name,i.교수명, lower(c.sno) sno, c.\*

FROM course c, subject s, dept d, instructor i

where c.cno = 1

and c.dno = d.dno

and s.sno = c.sno

and c.ino = i.ino;

use pnu;

SELECT c.sno, count(c.ino), group\_concat(i.교수명)

FROM course c, subject s, dept d, instructor i

where c.dno = d.dno

and s.sno = c.sno

and c.ino = i.ino

group by c.sno;

-- having count> 1;

Sakila

use sakila;

SELECT i.store\_id, i.film\_id, f.title, count(\*) cnt

FROM inventory i, film f

where store\_id = 1

and i.film\_id = f.film\_id

group by store\_id, film\_id

having count(\*) > 1

order by cnt desc;

# 각 카테고리에는 몇 편의 영화가 있나요?

# 'ACADEMY DINOSAUR'라는 영화를 대여한 고객의 이름은 무엇입니까?

# 가장 많은 작품에 출연한 배우는 누구인가요?

# 매장별 총 수익은 얼마나 되나요?

#한 번도 대여한 적이 없는 영화는 무엇인가요?

# 렌탈 비용을 가장 많이 지출한 고객은 누구인가요?

# 렌탈 건수 기준 상위 3명의 고객은 누구인가요?

# 대여료가 $2.00 미만인 영화는 몇 편이나 되나요

# 카테고리별 영화의 평균 대여 기간은 얼마나 되나요?

# 고객 ID 1과 고객 ID 2 모두가 대여한 영화를 나열합니다.

# 임대 건수가 가장 많은 달은 언제였나요?

# 결제를 가장 많이 처리한 직원은 누구인가요?

# 각 카테고리에는 몇 편의 영화가 있나요?

category -> category\_id, name

film\_category -> film\_id, category\_id

inventory -> film\_id

use sakila;

SELECT f.film\_id, c.category\_id, c.name, count(\*) cnt

FROM film\_category f, inventory i, category c

where c.category\_id = f.category\_id

and i.film\_id = f.film\_id;

group by f.category\_id

order by cnt desc; -> 했는데 안됨 group by 해야된대

select f.category\_id, c.name, count(\*) cnt

from film\_category f, category c

where f.category\_id = c.category\_id

group by f.category\_id

order by cnt desc;

use sakila;

SELECT c.category\_id, c.name, count(\*) cnt

FROM film\_category f, category c

where c.category\_id = f.category\_id

group by f.category\_id

order by cnt desc;

교수님 답

SELECT F.CATEGORY\_ID, C.NAME, COUNT(\*) CNT

FROM film\_category F, CATEGORY C

WHERE F.CATEGORY\_ID = C.CATEGORY\_ID

GROUP BY F.CATEGORY\_ID

ORDER BY CNT DESC;

# 'ACADEMY DINOSAUR'라는 영화를 대여한 고객의 이름은 무엇입니까?

rental -> customer\_id, inventory\_id

customer -> first\_name, last\_name, customer\_id

inventory -> film\_id, inventory\_id =1

film -> film\_id=1, title='ACADEMY DINOSAUR'

use sakila;

SELECT c.first\_name, c.last\_name, c.customer\_id, f.title

FROM film f, inventory i, customer c, rental r

where c.customer\_id = r.customer\_id

and r.inventory\_id = i.inventory\_id

and f.film\_id = i.film\_id

and f.title = 'ACADEMY DINOSAUR'

order by customer\_id;

# 가장 많은 작품에 출연한 배우는 누구인가요?

actor -> actor\_id, first\_name, last\_name

film\_actor -> actor\_id, film\_id

film -> film\_id

내 답

use sakila;

SELECT a.actor\_id, a.first\_name, a.last\_name, count(\*) cnt

FROM actor a, film\_actor x, film f

where a.actor\_id = x.actor\_id

and x.film\_id = f.film\_id

group by actor\_id

order by cnt desc

limit 1;

교수님 답

SELECT F.ACTOR\_ID, A.FIRST\_NAME, A.LAST\_NAME, COUNT(\*) CNT

FROM FILM\_ACTOR F, ACTOR A

WHERE F.ACTOR\_ID = A.ACTOR\_ID

GROUP BY F.ACTOR\_ID

ORDER BY CNT DESC

LIMIT 1;

# 매장별 총 수익은 얼마나 되나요?

0412, 0415

\*\*\* 필요한 경우 store, film, staff, customer 는 1 을 사용해 보세요!

-- # 2005년 5월 1일부터 2005년 5월 10일 사이에 발생한 모든 rental를 검색합니다.

SELECT \*

FROM RENTAL

WHERE rental\_date

BETWEEN str\_to\_date('2005-05-01 00:00:00', '%Y-%m-%d %H:%i:%s')

AND str\_to\_date('2005-05-01 23:59:59', '%Y-%m-%d %H:%i:%s');

-- 현재 날짜와 시간으로 새로운 rental 을 추가합니다.

INSERT INTO rental

(rental\_date,inventory\_id, customer\_id,staff\_id)

VALUES (now(), 1, 1, 1);

-- # 특정 rental id의 반납일을 7일 연장합니다.

UPDATE sakila.rental

SET return\_date = adddate(RETURN\_DATE, INTERVAL 7 DAY)

WHERE rental\_id = 16049;

SELECT \*

FROM sakila.rental

where rental\_id = 16049;

-- 각 rental에 대한 rental 기간을 일 단위로 표시합니다.

select rental\_date, return\_date, datediff(return\_date, rental\_date)

from rental;

-- # 2005년 주말에 이루어진 모든 rental 에 대해 rental 날짜를 다음 날로 업데이트합니다.

update rental

set return\_date = adddate(rental\_date, interval 1 day)

where dayofweek(rental\_date) in (1, 7);

-- # 2005년의 월별 rental 건수를 계산합니다.

select year(rental\_date), month(rental\_date), count(\*)

from rental

group by year(rental\_date), month(rental\_date);

# 2005년 6월 이후의 모든 rental를 찾아보세요.

select \*

from rental

where rental\_date >=str\_to\_date('2005-07-01 00:00:00','%Y-%m-%d %H:%i:%s');

search-ms:displayname=로컬%20디스크%20(C%3A)의%20검색%20결과&crumb=location:C%3A%5C\apache-tomcat-10.1.20-windows-x64

-- 이 프로시저는 MySQL에서 특정한 스토어(store), 인벤토리(inventory), 고객(customer), 직원(staff) 정보를 받아서 렌탈(rental)과 결제(payment)를 처리하는 프로시저입니다.

CREATE DEFINER = `root`@`%` PROCEDURE `sp\_rental\_with\_payment`(

-- 프로시저의 이름과 매개변수를 정의합니다.

IN store\_id INT, -- 스토어 ID를 입력으로 받습니다.

IN inventory\_id INT, -- 인벤토리 ID를 입력으로 받습니다.

IN customer\_id INT, -- 고객 ID를 입력으로 받습니다.

IN staff\_id INT -- 직원 ID를 입력으로 받습니다.

)

BEGIN

-- 프로시저의 본문을 시작합니다.

-- 예외 처리기를 정의합니다. SQL 예외나 경고가 발생할 경우에 대한 처리를 정의합니다.

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

-- 예외가 발생했을 때 롤백을 수행합니다.

ROLLBACK;

-- 예외에 대한 정보를 변수에 저장합니다.

GET DIAGNOSTICS CONDITION 1 @errno = MYSQL\_ERRNO, @sqlstate = RETURNED\_SQLSTATE, @text = MESSAGE\_TEXT;

-- 에러 메시지를 생성합니다.

SET @full\_error := CONCAT('ERROR ', @errno, ' (', @sqlstate, '): ', @text);

-- 에러 메시지를 출력합니다.

SELECT @full\_error;

-- 예외를 다시 발생시킵니다.

RESIGNAL;

END;

-- 예외 처리기를 정의합니다. SQL 경고나 NOT FOUND 상황에 대한 처리를 정의합니다.

DECLARE EXIT HANDLER FOR SQLWARNING, NOT FOUND

BEGIN

-- 예외가 발생했을 때 롤백을 수행합니다.

ROLLBACK;

-- 예외에 대한 정보를 변수에 저장합니다.

GET DIAGNOSTICS CONDITION 1 @`errno` = MYSQL\_ERRNO, @`sqlstate` = RETURNED\_SQLSTATE, @`text` = MESSAGE\_TEXT;

-- 에러 메시지를 생성합니다.

SET @full\_error := CONCAT('ERROR ', @`errno`, ' (', @`sqlstate`, '): ', @`text`);

-- 에러 메시지를 출력합니다.

SELECT @full\_error;

-- 예외를 다시 발생시킵니다.

RESIGNAL;

END;

-- 매개변수 값을 변수에 저장합니다.

SET @store\_id := store\_id;

SET @inventory\_id := inventory\_id;

SET @customer\_id := customer\_id;

SET @staff\_id := staff\_id;

-- 트랜잭션을 시작합니다.

START TRANSACTION;

-- rental 테이블에 새로운 렌탈 정보를 삽입합니다.

INSERT rental (rental\_date, inventory\_id, customer\_id, staff\_id)

SELECT now(), a.inventory\_id, @customer\_id, @staff\_id

FROM inventory a

WHERE a.store\_id = @store\_id and a.inventory\_id = @inventory\_id

AND NOT EXISTS (

SELECT r.inventory\_id

FROM rental r

WHERE r.inventory\_id = @inventory\_id

AND r.return\_date is null

);

-- 영향 받은 행의 수를 변수에 저장합니다.

SET @rows\_affected := ROW\_COUNT();

-- 영향 받은 행의 수가 0이면 에러를 발생시킵니다.

IF @rows\_affected = 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Error: 0 rows affected';

SELECT '@rows\_affected', @rows\_affected;

END IF;

-- 마지막으로 삽입된 rental 레코드의 ID를 변수에 저장합니다.

SET @last\_rental\_id = LAST\_INSERT\_ID();

-- 해당 inventory에 대한 렌탈 요금을 가져옵니다.

SELECT @rental\_rate := rental\_rate

FROM film a

WHERE a.film\_id = (select b.film\_id

from inventory b

where b.inventory\_id = @inventory\_id

group by b.inventory\_id);

-- 결제 정보를 payment 테이블에 삽입합니다.

INSERT INTO payment (customer\_id, staff\_id, rental\_id, amount, payment\_date)

VALUES (@customer\_id, @staff\_id, @last\_rental\_id, @rental\_rate, now());

-- 트랜잭션을 커밋하여 변경사항을 확정합니다.

COMMIT;

END;

Notification 테이블에

create event notification\_every

on schedule every 1 minute

do

insert into notification

(rental\_id, send\_date, customer\_id, channel\_type)

select rental\_id, now(), customer\_id, 'KakaoTalk'

from rental

where return\_date is null

and rental\_date between '2006-02-14' and '2006-02-15';

--

insert into sakila.payment

(customer\_id, staff\_id, rental\_id, amount, payment\_date)

values(1, 1, 76, 1.00, now());

select \*

from rental r left outer join payment p on

r.rental\_id = p.rental\_id

where payment\_id = null;

select \* from sakila.payment;

drop trigger book;

delimiter |

create trigger book after update on inventory for each row begin

if (old.reserved != new.reserved)

then

insert into book

(inventory\_id, customer\_id, book\_date, status)

values(old.inventory\_id, 1, curdate(), new.reserved);

end if;

end

|

delimiter ;

select \* from information\_schema.triggers;