SQL Practice 3

1. Create database university;
2. Show databases;
3. Use university;
4. Create table…
5. Show tables;
6. Describe course;
7. Insert into …

(1) 執行

Select *

From ***course*** **natural left outer join** ***prereq***;

結果中 prereq_id = null 的資料代表什麼?

- 沒有先修課程要求的課程資料

(2) 執行

Select *

From ***course*** **natural join** ***prereq***;

結果和(1) 結果的差異為何?

- 只有具有先修要求的課程資料會被列出來

(3) 執行

Select *

From ***course*** **natural right outer join** ***prereq***;

其結果和(2)結果相同嗎? 這隱含什麼意思?

- 相同. 表示 ***prereq*** 表中的 course_id 都存在 course 表中

(4) 請自行新增一筆資料到 ***prereq***, 使(3)運算的結果和(2)的結果會不同, 寫出此一新增資料, 說明它的特性.

- 例如新增 ('BIO-201', 'BIO-301')
  但是 prereq 中的 course_id 若有設成 foreign key, 則無法新增
  以避免此課程編號實際不存在 course 中的不合理現象

(5) 執行

select *

from *course* **inner join** *prereq* **on**

course.course_id = prereq.course_id;

結果和(2) 結果的差異為何? (請觀察欄位)

- course_id 欄位不會像 **natural join** 的結果只保留一個, 而會留下有 2 個 course_id 欄位

(6) 執行

select *

from *section* **left outer join** *teaches* **using** (*course_id*);

select *

from *section* **natural left outer join** *teaches*;

這兩個查詢在執行 join 時, 檢查的條件有何差異?

- 對 2 個 SQL 查詢, 都是先對 *section* 跟 *teaches* 做 Cartesian product, 但第一個 SQL 的檢查條件只用 using 後面指定的欄位相等: section.course_id = teaches.course_id
  第二個 SQL 的檢查條件則用共同欄位相等:
  section.year = teaches.year and section.semester = teaches.semester and section.course_id = teaches.course_id and section.sec_id = teaches.sec_id

顯示的欄位有何差異?

- 第一個 SQL 的顯示欄位為 *section* 的欄位, 加上 *teaches* 的欄位, 但 using 的欄位不重複顯示(7+5-1 所以有 11 個)
  第二個 SQL 的顯示欄位為 *section* 的欄位, 加上 *teaches* 的欄位, 所有共同欄位不重複顯示(7+5-4 所以有 8 個)

(7) Write a SQL query to

Display a list of all instructors, showing their ID, name, and the number of sections that they have taught.

(Make sure to show the number of sections as 0 for instructors who have not taught any section. Your query should use an outer join.)

select ID, name, count(course_id)
from *instructor* **natural left outer join** *teaches*
group by ID, name;

8. Create View

(1)

create view *dept_total_salary*(*dept_name*, *total_salary*) **as**
select *dept_name*, **sum**(*salary*)
from *instructor*
group by *dept_name*;

執行 select * from *dept_total_salary*;

insert into instructor **values** ('10301', 'KOH', 'Comp. Sci.', '30000');

再執行 select * from *dept_total_salary*; 結果有何改變?

- Comp. Sci 系的 *total_salary* 欄位會增加 30000

(2) 觀察新增前後 *instructor* 表中的改變

create view *faculty* **as**
select *ID*, *name*, *dept_name*
from *instructor*;

執行 select * from *instructor*;
執行 select * from *faculty*;

insert into *faculty* **values** ('630765', 'Green ', 'Music ');

執行 select * from *faculty*;
執行 select * from *instructor*;

Instructor 新增的資料有哪個欄位是空值? 為什麼?

- salary 欄位是空值, 因為透過 view 的對應新增, 未提供 salary 欄位

(3) 觀察新增前後 *instructor* 表中的改變

create view *history_instructors* **as**
select *
from *instructor*
where *dept_name*= 'History';

執行 select * from *instructor*;
執行 select * from *history_instructors*;

insert into *history_instructors* *values* ('25566', 'Brown', 'Biology', 100000);

執行 select * from *instructor*;
執行 select * from *history_instructors*;

思考為什麼新增的資料從 *history_ instructors* 找不到?

- 因為新增時給定的系名是'Biology'(這裡有問題, 為什麼生物系的老師要新增到歷史系的老師中), 實際是新增到 instructors, 而 history_instructors 只是一個 view. 透過查詢 instructors 中 *dept_name*= 'History' 的資料, 所以感覺新增了, 卻從 history_instructors 找不到

9. Transction

```
CREATE TABLE customers(a int, b char(20), index(a)) ENGINE=InnoDB;
# BEGIN 的方法
BEGIN;
INSERT INTO customers VALUES(10, 'abc');
COMMIT;    # 此 BEGIN Transaction 已在此結束
SELECT * FROM customers;
INSERT INTO customers VALUES(11, 'aaa');
ROLLBACK;  # 此命令是無作用的, 已經不在 BEGIN 的範圍了, 資料會自動 COMMIT 進去.
SELECT * FROM customers;

# SET AUTOCOMMIT 的方法
# (注意: 其它 Connection 並不會因為這 Connection 設定而不 自動 COMMIT)
SET AUTOCOMMIT=0;
INSERT INTO customers VALUES(15, 'def');
ROLLBACK;
INSERT INTO customers VALUES(16, 'ggg');
COMMIT;
SELECT * FROM customers;
```

回復每筆交易都會自動 COMMIT 的狀態
SET AUTOCOMMIT=1;

10.Foreign key 設定 (需啟動 Innodb)

```
CREATE TABLE parent (
        id INT NOT NULL,
        PRIMARY KEY (id)
) ENGINE=INNODB;

CREATE TABLE child(
        id INT,
        parent_id INT,
        PRIMARY KEY (id),
       FOREIGN KEY (parent_id)
              REFERENCES parent(id)
              ON update CASCADE
              ON delete CASCADE
      ) ENGINE=INNODB;

insert into parent values(100);
insert into parent values(200);
insert into child values(001,100);
insert into child values(002,100);
insert into child values(003,200);
```

(1) 請觀察 insert into child values(004, 500); 後的錯誤訊息
執行 delete from parent where id = 100; 後兩個表中的內容有何改變?

- 執行 insert into child values(001, 500); 因為 child 中的 primary key 為 id, id 重複而無法加入
- 若執行 insert into child values(004, 500); 因為 child 中設 foreign key 到 parent(id), 500 不存在 parent 中而無法執行
- 執行 delete from parent where id = 100; 因為 parent(id)為 child 的 foreign key, 而 child 中有資料的 parent id 為 100, 且指定 on delete cascade 的處理方式, 因此 parent 表中及 child 表中 parent id 為 100 的資料都會刪除.

(2) 執行 update parent set id = 300 where id=200; 後兩個表中的內容有何改變?
Parent 中 id =200 被改成 300, 而 child 中 parent_id 為 200 也被自動改成 300

(3) 根據 Lab 中 ddl.sql 中的宣告，若刪除 department 中一筆資料，會影響到哪些表中的資料進行那些變動?

例如刪除 department 中 dept_name 為'Music'者

Course 的 dept_name 若為刪除的'Music'，會被設為 null

Instructor 的 dept_name 若為刪除的'Music'者，會被設為 null

Student 的 dept_name 若為刪除的'Music'者，會被設為 null