SQL Lab 4

1. Stored function

1) 宣告 function

```
delimiter //
create function dept_count (dept_name varchar(20))
  returns int
  begin
  declare d_count int;
  select count(*) into d_count
  from instructor
  where instructor.dept_name = dept_name;
  return d_count;
  end ;
  //
delimiter ;
```

請執行

```
  select dept_name, budget
  from department
  where dept_count (dept_name ) > 1
```

請說明此 function 的作用.

此 function 可算出輸入系名為 dept_name 的老師人數 因此上述 SQL 查詢會列出老師人數多於 1 人的系及其預算值

2) 宣告 procedure

```
delimiter //
create procedure dept_count_proc (in dept_name varchar(20))
          begin
      select count(*)
      from instructor
      where instructor.dept_name = dept_count_proc.dept_name;
          end ;
//
delimiter ;
```

```
call dept_count_proc('Physics');
```

請說明此 procedure 的作用.

此 procedure 可算出輸入系名為 dept_name 的老師人數 因此上述 SQL 查詢會列出物理系的老師人數

2. Trigger

1)

```
CREATE TABLE account (acct_num INT, amount DECIMAL(10,2));
CREATE TRIGGER ins_sum BEFORE INSERT ON account
FOR EACH ROW SET @sum = @sum + NEW.amount;

SET @sum = 0;
INSERT INTO account VALUES(137,100),(141,50),(97,-30);
SELECT @sum AS 'Total inserted';
```

觀察上述 insert 指令執行後 select 的結果，請說明此 trigger ins_sum 的作用

此 trigger 在對 account 執行 insert 前，加總新增資料的 amount 值

2)

```
CREATE TABLE test1(a1 INT);
CREATE TABLE test2(a2 INT);
CREATE TABLE test3(a3 INT NOT NULL AUTO_INCREMENT PRIMARY KEY);
CREATE TABLE test4(
   a4 INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
   b4 INT DEFAULT 0
);

delimiter //

CREATE TRIGGER testref BEFORE INSERT ON test1
   FOR EACH ROW
   BEGIN
      INSERT INTO test2 SET a2 = NEW.a1;
      DELETE FROM test3 WHERE a3 = NEW.a1;
      UPDATE test4 SET b4 = b4 + 1 WHERE a4 = NEW.a1;
   END
//

delimiter ;

INSERT INTO test3 (a3) VALUES
   (NULL), (NULL), (NULL), (NULL), (NULL),
   (NULL), (NULL), (NULL), (NULL), (NULL);

INSERT INTO test4 (a4) VALUES
   (0), (0), (0), (0), (0), (0), (0), (0), (0), (0);
```

先觀察記錄下 test1, test2, test3, 及 test4 中的值,

test1 為空, test2 為空, 因為有 AUTO_INCREMENT 指令
test3 為((1),(2),(3),(4),(5),(6),(7),(8),(9),(10)),
test4 為((1,0),(2,0),(3,0),(4,0),(5,0),(6,0),(7,0),(8,0),(9,0),(10,0))

接下來執行

```
INSERT INTO test1 VALUES (1), (3), (1), (7), (1), (8), (4), (4);
```

請觀察 test1, test2, test3, 及 test4 中的值有何變化

test1 為((1),(3),(1),(7),(1),(8),(4),(4))
test2 為((1),(3),(1),(7),(1),(8),(4),(4))
test3 為((2),(5),(6),(9),(10))
test4 為((1,3),(2,0),(3,1),(4,2),(5,0),(6,0),(7,1),(8,1),(9,0),(10,0))

請說明此 trigger 的作用

對 test1 新增前，也同時對 test2 新增相同的值
若 test3 中有此新增值，則會被刪除（因此 test3 中 1, 3, 4, 7, 8 會被刪除）
test4 中會將 test1 中新增值的新增次數統計在 b4 欄位中，因此 1 被新增 3 次，3 被新增 1 次，4 被新增 2 次，7 被新增 1 次，8 被新增 1 次

3. 請寫一個 trigger，當新增一個學生修課資料(takes)，這個 trigger 會自動更新這個學生對應的總學分數 tot_cred

```
delimiter //
CREATE TRIGGER increase_cred AFTER INSERT ON takes
FOR EACH ROW
BEGIN
IF NEW.grade is not null and NEW.grade <> 'F'
THEN UPDATE student
SET tot_cred = tot_cred +
(select credits from course where course.course_id= NEW.course_id)
WHERE id = NEW.id;
END IF;
END; //
delimiter ;
```

4. 請寫一個 trigger，當刪除一個學生修課資料(takes)，這個 trigger 會自動更新這個學生對應的總學分數 tot_cred

```
delimiter //
CREATE TRIGGER decrease_cred AFTER DELETE ON takes
FOR EACH ROW
BEGIN
IF OLD.grade is not null and OLD.grade <> 'F'
THEN UPDATE student
SET tot_cred = tot_cred -
(select credits from course where course.course_id= OLD.course_id)
WHERE id = OLD.id;
END IF;
END; //
delimiter ;
```

5. 請寫一個 trigger，當修改一個學生修課資料(takes)，這個 trigger 會自動更新這個學生對應的總學分數 tot_cred

```
delimiter //
CREATE TRIGGER update_cred AFTER UPDATE ON takes
REFERENCING NEW ROW AS nrow REFERENCING OLD ROW AS orow
FOR EACH ROW
BEGIN
IF nrow.grade <> 'F' AND nrow.grade IS NOT NULL AND
( orow.grade = 'F' OR orow.grade IS NULL )
SET tot_cred = tot_cred +
( SELECT credits FROM course WHERE course.course_id = nrow.course_id )
WHERE student.ID = nrow.ID ;
END IF;
END; //
delimiter ;
```

或

```
delimiter //
CREATE TRIGGER update_cred AFTER UPDATE ON takes
FOR EACH ROW
BEGIN
IF OLD.id=NEW.id THEN IF OLD.grade is null and NEW.grade is not null and
NEW.grade <> 'F'
THEN UPDATE student SET tot_cred = tot_cred +
(select credits from course where course.course_id= OLD.course_id)
WHERE id = OLD.id;
ELSEIF OLD.grade = 'F' and NEW.grade is not null and NEW.grade <> 'F'
THEN UPDATE student SET tot_cred = tot_cred +
(select credits from course where course.course_id= OLD.course_id)
WHERE id = OLD.id;
ELSEIF OLD.grade is not null and OLD.grade <> 'F' and NEW.grade ='F'
THEN UPDATE student SET tot_cred = tot_cred -
(select credits from course where course.course_id= OLD.course_id)
WHERE id = OLD.id;
END IF;
END IF;
END; //
delimiter ;
```