西北工業大學

NORTHWESTERN POLYTECHNICAL UNIVERSITY

Lab report

Name : ABID ALI
Student_No : 2019380141

Experiment 5

Experiment No:5

Trigger, Procedure and Function

Goal:

To practice how to create the trigger/procedure/function.

Content:

1. For SPJ_ MNG database, create and execute the following procedures.

(1) Create a stored procedure with no parameters named jsearch1. The function of the procedure is: when the procedure is executed, it returns all the information of Beijing suppliers in table S. Call the procedure and verify the results.

(2) Create a procedure with input parameters named jsearch2. The function of the procedure is: when you enter the city name of a supplier (such as Beijing), it will return all the information of the specific supplier. Call the procedure and verify the results.

(3) Create a procedure (or a function) jsearch3 with input and output parameters. The function of this procedure/function is: when a supplier number (input parameter SNO) is entered, the name of the supplier (output parameter SNAME) will be returned. Call the procedure/function and verify the results.

(4) Create a procedure jsearch4 using cursors, call the procedure and verify the result after. The function of the procedure is: when a project number JNO is input, the names (SNAMEs) of all suppliers supplying the engineering parts will be returned. These supplier names are spliced into a string and separated by comma ','.
For example: input: J2, output: 'JINGYI, SHENGXIN, WEIMIN'.

(5) View the text information of the procedures jsearch1 and jsearch2.

(6) View the basic status information of the procedures jsearch1 and jsearch2.

(7) Delete the procedure jsearch1.

2. Create and execute the following triggers for student database: (40 points in total)

(1) Delete the foreign key constraint on the SC table and create INSERT trigger named insert_s for the table SC. The function of the trigger is: when the user inserts a record into the SC table, if the inserted CNO value is not the existing value of CNO in table C, the user will be prompted a message "the data not in table C cannot be inserted" and the insertion of the data will be prevented; if the inserted SNO value is not an existing value of SNO in table S, the user will be prompted that "the data not in table S cannot be inserted" and the insertion of the data will be prevented. After the trigger created successfully, insert some records into the SC table to verify whether the trigger works normally. (5 points)

(2) Create a DELETE trigger named dele _ s1 for table S. The function of the trigger is: to prompt the user a message of "the data in this table cannot be deleted" and prevent the user from deleting the data in table S. After the trigger is created successfully, delete some record in table S to verify whether the trigger works normally. (5 points)

(3) Create a DELETE trigger named dele _ s2 for the table S, the function of the trigger is to delete the his course selection record in SC table when deleting a student's record in S table. After the trigger is created successfully, delete some records in the table S and verify whether the trigger works normally (confirm whether the relevant data of S table and SC table are both deleted). (5 points)

(4) Create an UPDATE trigger for table S named update_s, the function of this trigger is to prohibit updating the contents of the column "sdept" in table S (make the updating be failed and prompt a message of "the column of 'sdept' cannot be updated). After the trigger is created successfully, update the content of the "sdept" column in the table S to verify whether the trigger works normally. (5 points)

(5) Delete update_s trigger. (5 points)

(6) Design a before update trigger and after update trigger, and compare the difference between the two triggers. (5 points)

(7) Create a new curriculum score statistics table CAVGGRADE (CNO, snum, examsnum, avggrade), for the columns are indicating the course number, the number of students who choose the course, the number of students taking the exam, and the average score of the course respectively. Use a trigger to achieve the following functions: when insert, delete or update a person's score of SC table, automatically update the records in table CAVGGRADE correspondingly. Note that if the grade of a student in SC table is NULL, it indicates that the student has not taken the exam, and it is not necessary to calculate the average score. However, when grade is 0, that is, the score of the student is 0, the average grade needs to be calculated. (10 points)

3. Create an table employee (EID, ename, salary) for the employees, assuming there are 1000 employee data in the table, complete the following requirements. (total 20 points, 10 points for each question)

(1) In order to automatically generate 1000 employee data, a user-defined function GENERATEEID is created to automatically generate employee ID. The employee ID is required to be an 8-digit number. The higher four digits indicate the current year in which the employee data is inserted, and the last four digits increase in the order from 0001 to 9999. For example, the first record inserted in the year of 2015 is 20150001, and all 1000 employee IDs are 20150001-20151000. Call this function to insert 1000 pieces of data automatically. (Note:when inserting data that the employee name can be any value , and the value of salary is between 2000 and 5000)

(2) The company plans to increase the salary of each employee according to certain rules. Please use cursor to create a procedure and execute the procedure to complete the salary adjustment. The salary increase rules are as follows:

- The salary below 3000 yuan, with a monthly increase of 300 yuan;
- The salary between 3000-4000 yuan, with a monthly increase of 200 yuan;
- The salary more than or equal to 4000 yuan, with a monthly increase of 50 yuan;

1(1)

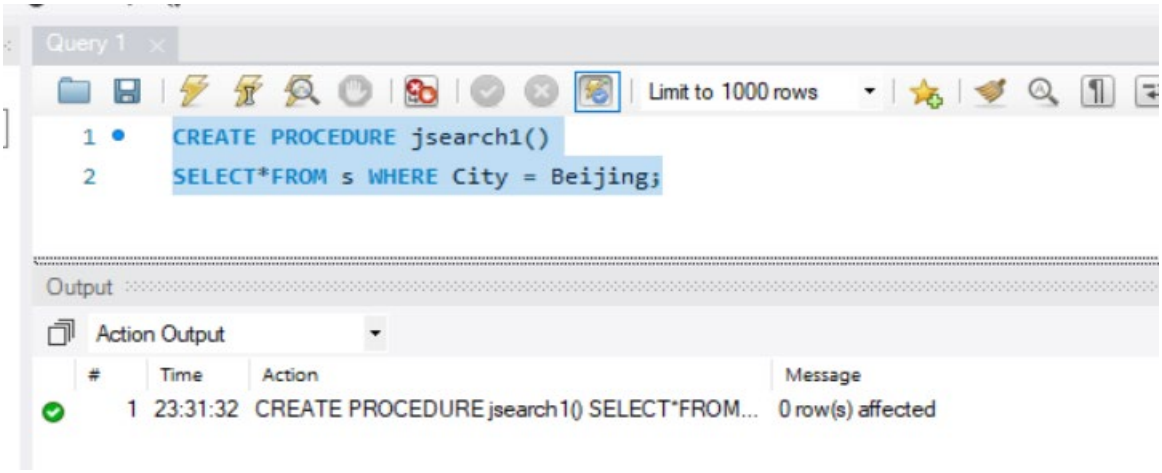
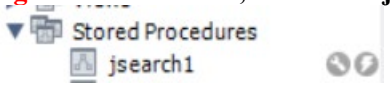


Fig: Creating Procedure jsearch1

SNO	SNAME	STATUS	CITY
S2	SHENGXI	10	Beijing
S3	DONGFANGHONG	30	Beijing

Fig: We can see that,Procedure jsearch1



1(2)

Fig:The table was called jsearch2

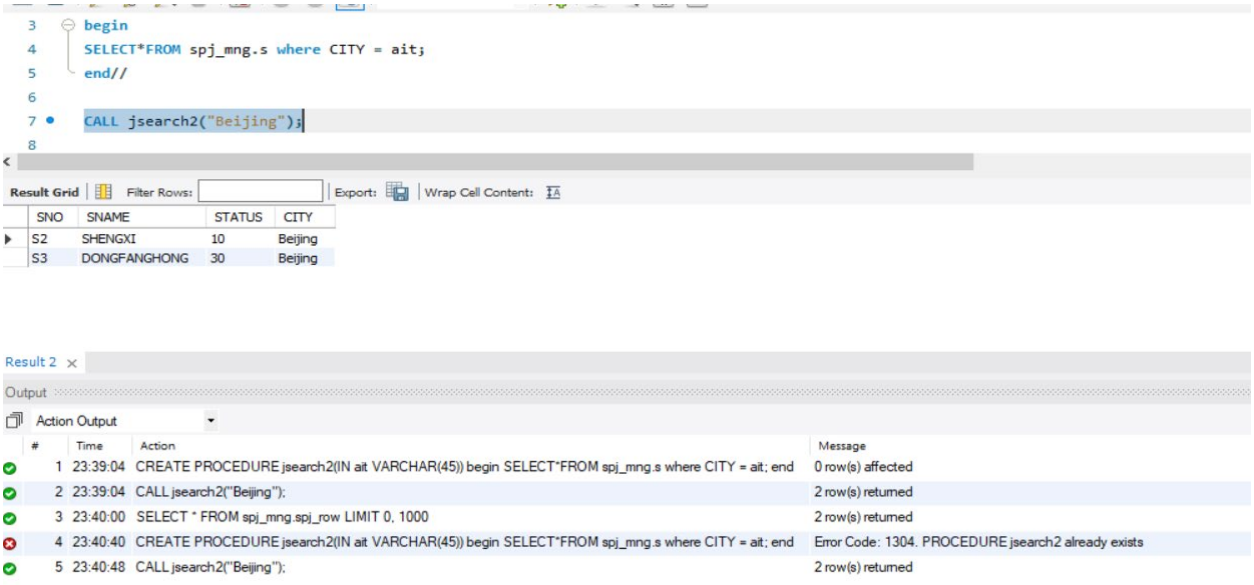


Fig: Creating Procedure jsearch2

Code:

```
DELIMITER //
CREATE PROCEDURE jsearch2(IN ait VARCHAR(45))
begin
SELECT*FROM spj_mng.s where CITY = ait;
end//

CALL jsearch2("Beijing");
```

SNO	SNAME	STATUS	CITY
S2	SHENGXI	10	Beijing
S3	DONGFANGHONG	30	Beijing

Fig: Creating procedure jserach2 with the following input parameters.

1(3)

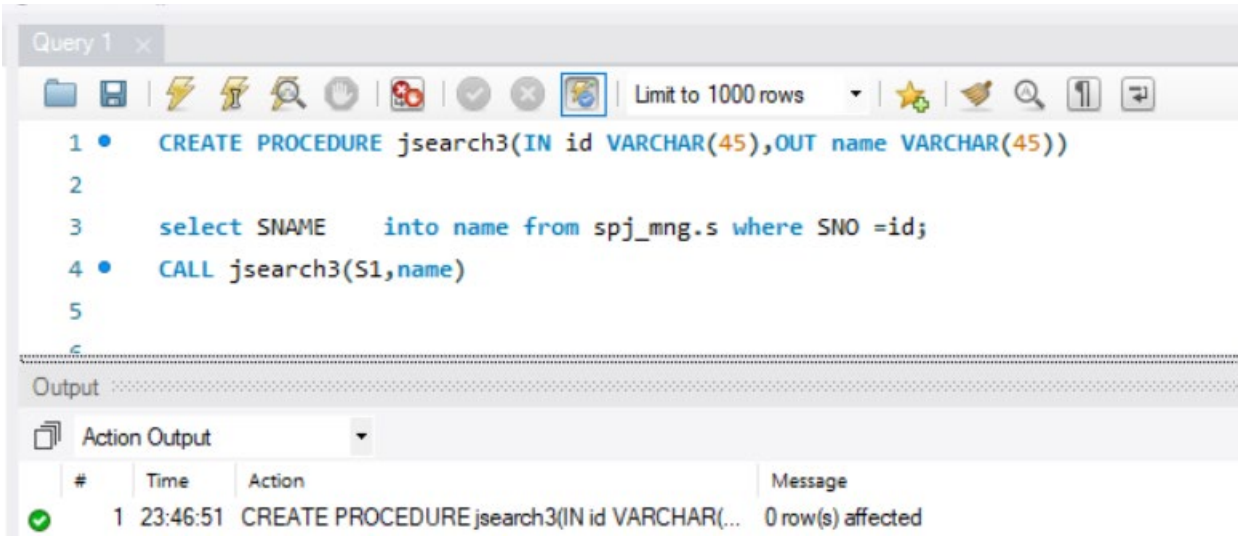


Fig: Creating procedure jsearch3 with input

SNAME
SHENGXI

1(4)

```
1  SELECT SNAME FROM   j;
2  DECLARE
3      CURSOR jsearch4 IS
4        SELECT SNAME FROM   j
5        WHERE  JNO = J1;
```

Fig: Creating the jsearch4 procedure using cursor.

```
JNAME
SANJIAN
```

```
1  SHOW CREATE PROCEDURE jsearch1;
2  SHOW CREATE PROCEDURE jsearch2;
3
4
```

Output

Action Output

#	Time	Action	Message
1	23:53:37	SHOW CREATE PROCEDURE jsearch1	1 row(s) returned
2	23:53:37	SHOW CREATE PROCEDURE jsearch2	1 row(s) returned

Fig: Procedure is showing

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Procedure	sql_mode	Create Procedure	character_set_c
jsearch2	STRICT_TRANS_TABLES,NO_ENGINE_SUBSTIT...	CREATE DEFINER=`root`@`localhost` PROCE...	utf8mb4

Result Grid

Form Editor

Fig : Showing the procedure status

```
1  DROP PROCEDURE jsearch1;
```

Deleting the procedure1

2

(1)

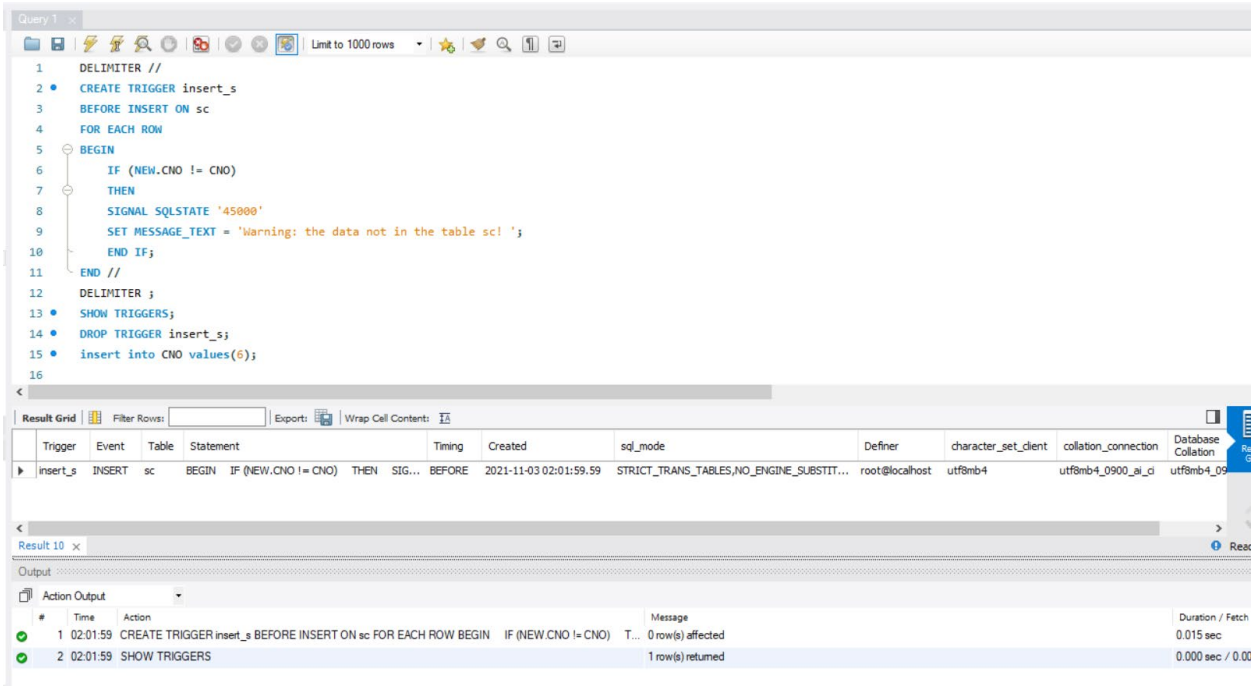
```
1  ALTER TABLE student.sc
2  CHANGE COLUMN CNO CNO INT NULL DEFAULT NULL;
3  ALTER TABLE student.sc;
```

Output

Action Output

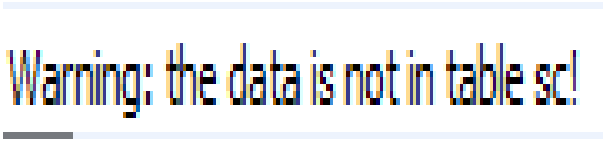
#	Time	Action	Message
1	00:39:53	ALTER TABLE student.sc CHANGE COLUMN CN...	6 row(s) affected Records: 6 Duplicates: 0 Warnin...
2	00:39:53	ALTER TABLE student.sc	0 row(s) affected

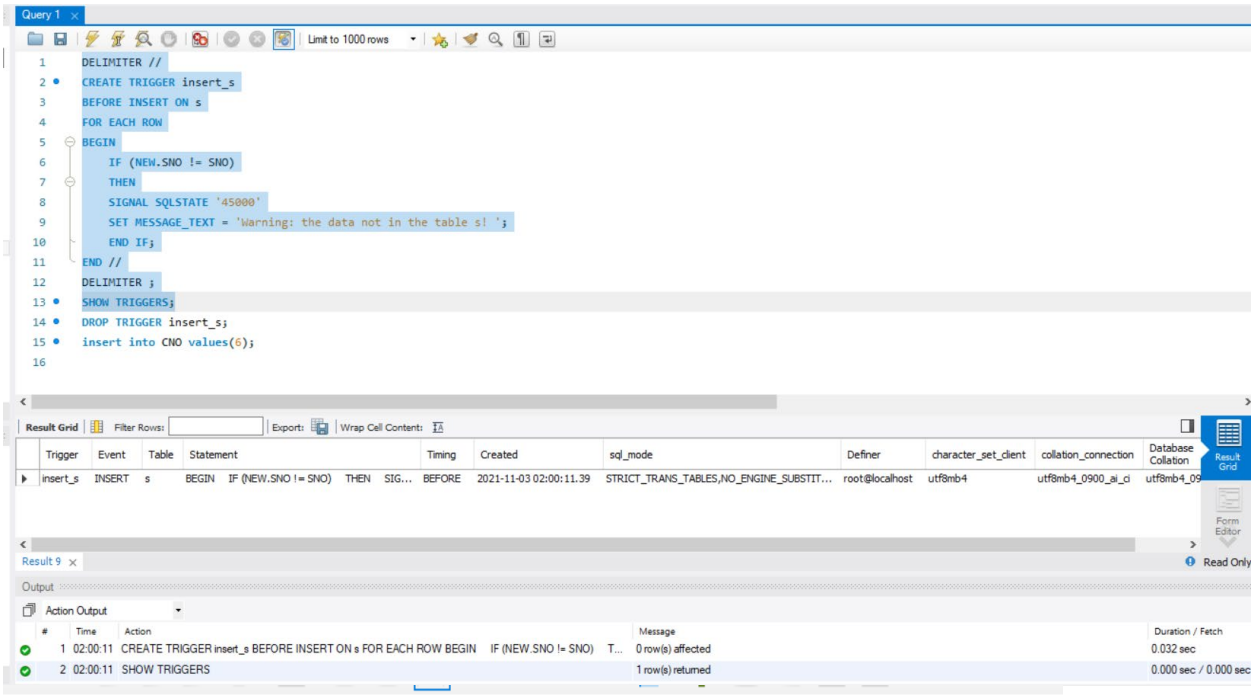
Fig: Drop the foreign key constraints in the “SC” table in student database




```
DELIMITER //
CREATE TRIGGER insert_s
BEFORE INSERT ON sc
FOR EACH ROW
BEGIN
   IF (NEW.CNO != CNO)
   THEN
   SIGNAL SQLSTATE '45000'
   SET MESSAGE_TEXT = 'Warning: the data not in the table sc! ';
   END IF;
END //
DELIMITER ;
SHOW TRIGGERS;
```

Creating trigger to prevent the insertion in table “SC”.






```
DELIMITER $$
CREATE TRIGGER insert_s
BEFORE INSERT ON s
FOR EACH ROW
BEGIN
  IF NEW.SNO != SNO()
  THEN
   SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'Warning: the data is not in table s!';
  END IF;
END$$
   DELIMITER ;
```

```
DELIMITER //
CREATE TRIGGER insert_s
BEFORE INSERT ON s
FOR EACH ROW
BEGIN
  IF (NEW.SNO != SNO())
  THEN
  SIGNAL SQLSTATE '45000'
  SET MESSAGE_TEXT = 'Warning: the data not in the table s! ';
  END IF;
END //
DELIMITER ;
SHOW TRIGGERS;
```

Warning: the data is not in table s!

(2)

Creating trigger to prevent the insertion in ”S” table and show the prevent

```
1    DELIMITER $$
2    CREATE TRIGGER dele_s1
3    BEFORE DELETE ON s
4    FOR EACH ROW
5    BEGIN
6      IF s=""
7      THEN
8       SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'Warning: the data in this table cannot be deleted!';
9      END IF;
10   END$$
11      DELIMITER ;
```

Now creating the trigger to prevent the deletion in “S” and show the message.

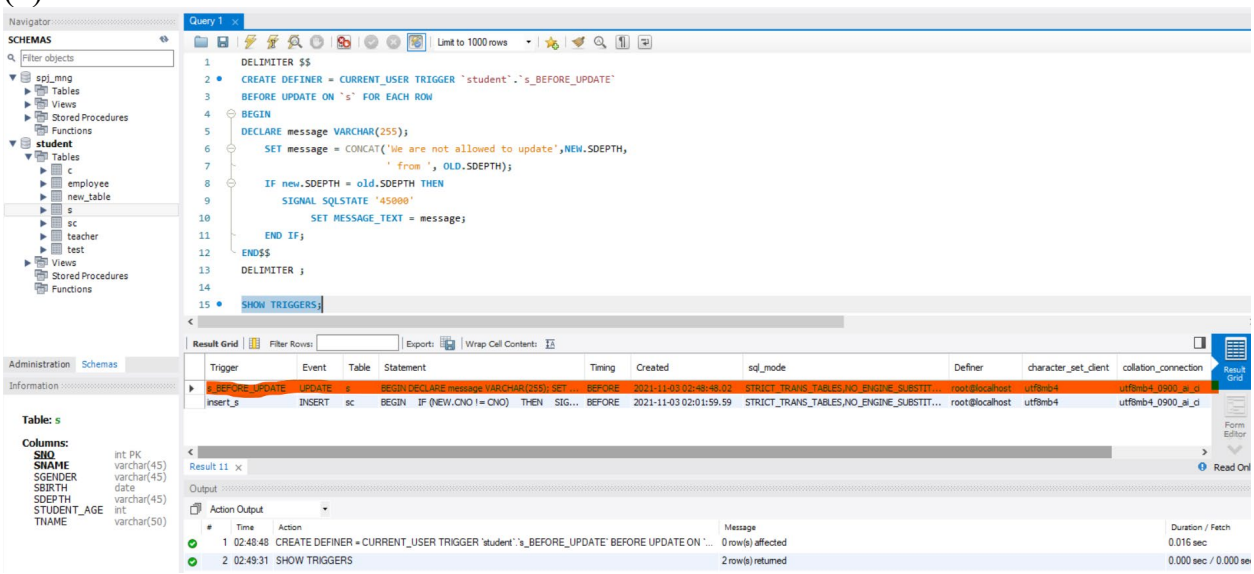
Warning: the data in this table cannot be deleted

3

```
1        DELIMITER $$
2        CREATE TRIGGER dele_s2
3        AFTER DELETE ON s
4        FOR EACH ROW
5        BEGIN
6            DELETE FROM sc
7              WHERE s.SNO = old.SNO;
8        END$$
9              DELIMITER ;
```

Now Creating the trigger that works after deletion.

(4)

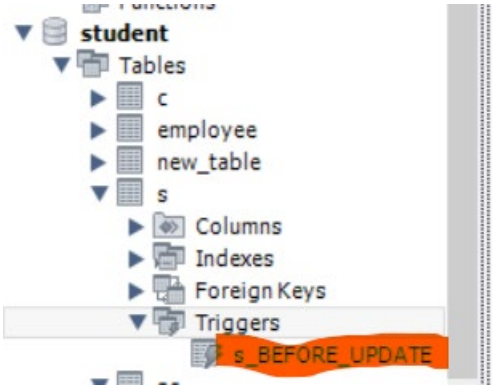


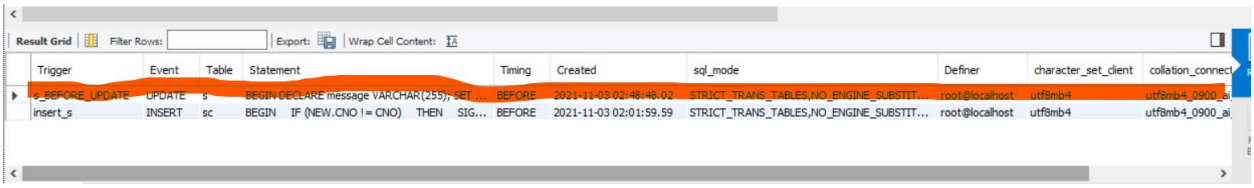

```
DELIMITER $$
CREATE DEFINER = CURRENT_USER TRIGGER `student`.`s_BEFORE_UPDATE`
BEFORE UPDATE ON `s` FOR EACH ROW
BEGIN
DECLARE message VARCHAR(255);
        SET message = CONCAT('We are not allowed to update',NEW.SDEPTH,
                ' from ', OLD.SDEPTH);
        IF new.SDEPTH = old.SDEPTH THEN
    SIGNAL SQLSTATE '45000'
       SET MESSAGE_TEXT = message;
        END IF;
END$$
DELIMITER ;
```

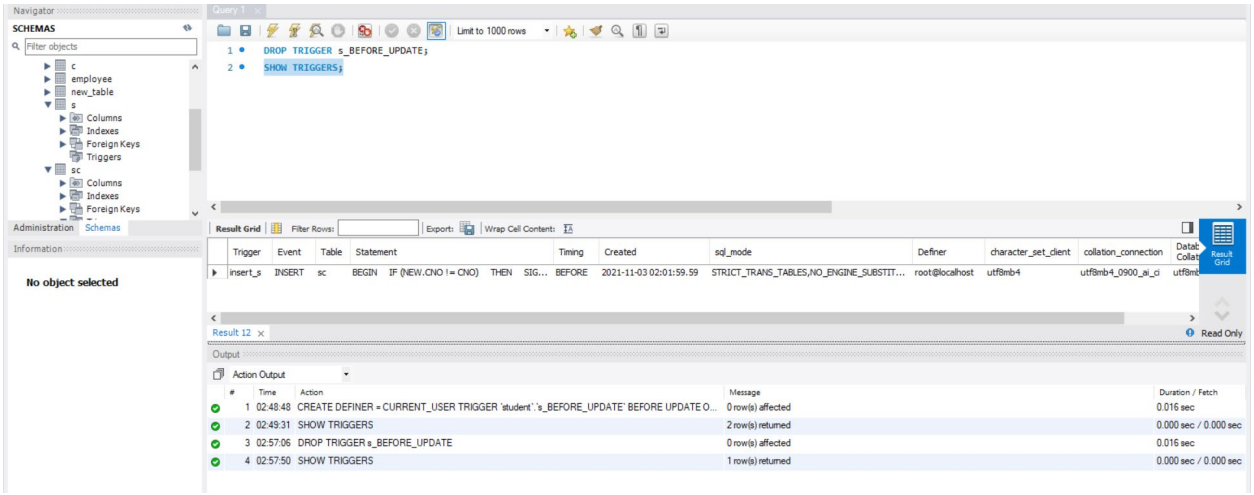
Now creating the trigger that prevents “S” table to update.

```
Warning: we are not allowed to update 200 from 700
```

(5)







(6)

```
1 •   CREATE DEFINER = CURRENT_USER TRIGGER `student`.`s_AFTER_UPDATE`
2       AFTER UPDATE ON `s` FOR EACH ROW
3     BEGIN
4       update s
```

Now create a after update trigger and compare it with previous trigger is that the before trigger works before update to prevent or tells a specific message and after update triggers works after update.

(7)

```
1   CREATE TABLE `student`.`cavggrade` (
2     `CNO` INT NOT NULL,
3     `snum` VARCHAR(45) NULL,
4     `examsnum` VARCHAR(45) NULL,
5     `avggrade` VARCHAR(45) NULL,
6     PRIMARY KEY (`CNO`));
7
```

Creating table “CAVGGRADE”

```
1 •   CREATE DEFINER = CURRENT_USER TRIGGER
2     `student`.`cavggrade_AFTER_UPDATE`
3     AFTER UPDATE ON `sc` FOR EACH ROW
4     BEGIN
5       update student.cavggrade
6       avg
```

Calculating average.

3(1)

```
1  CREATE TABLE spj_mng.employee(
2   `EID` INT NOT NULL DEFAULT 20150001,
3    `ename` varchar(45) NULL,
4    `salary` varchar(45) NULL,
5    PRIMARY KEY (`EID`));
```

Output

Action Output

#	Time	Action	Message
1	01:05:09	CREATE TABLE spj_mng.employee(`EID` INT NO...	0 row(s) affected

```
1  ALTER TABLE `spj_mng`.`employee`
2  CHANGE COLUMN `EID` `EID` INT NOT NULL AUTO_INCREMENT;
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	01:09:08	ALTER TABLE `spj_mng`.`employee` CHANGE CO...	0 row(s) affected Records: 0 Duplicates: 0 Warnin...	0.062 sec

Creating table name “employee” with auto insertion data in “EID” that starts from 20150001-20151000.

EID	ename	salary
20150001	s	1000
20150002	d	1000
20150003	h	3000
20150004	g	4000
20150005	r	4500

We can see the above table has “EID” that starts from 20150001-20151000 it is done through auto insertion.Also,8 digits EID is noticeable and the value of salary is between 2000 and 5000.

3(2)

```
1  SELECT salary FROM   employee;
2  DECLARE
3      CURSOR employee_cur IS
4        SELECT salary FROM   employee
5        FOR UPDATE;
6      incr_sal NUMBER;
7  BEGIN
8      FOR employee_rec IN employee_cur LOOP
9          IF employee_rec.salary < 3000 THEN incr_sal := +300;
10         ELSE IF employee_rec.salary > 4000 THEN incr_sal := +50;
11           ELSE incr_sal := +200;
12         END IF;
13         UPDATE employee
14         SET    salary = salary + salary * incr_sal
15         WHERE  CURRENT OF employee_cur;
16     END LOOP;
17 END;
```

Now increase the salaries of the employees If the salary less than 3000 yuan then 300 yuan increase, if the salary is between 3000-4000 yuan then 200 yuan increase and if the salary is greater than 4000 yuan then 50 yuan increase.

EID	ename	salary
20150001	s	1300
20150002	d	1300
20150003	h	3200
20150004	g	4050
20150005	r	4550

Problems:

I was not familiar with the concepts of create the trigger/procedure/function. I was having difficulties understanding those concepts and implementing those using mysql was little bit complicated for me.

Solution:

To solve these problems which I faced during doing this practical, I took help from internet especially YouTube,StackOverflow and W3school to get information about these errors for the solution. I also asked the teacher to help me understand them. And provided instructions helped to solve some of my errors during the experiment.
Then, successfully managed to solved the problem that I faced.

Summary:

In conclusion, this was a good learning experience in learning SQL concepts. I put SQL concepts into implementation that enhanced my skills and that open a new scope in my life about Database field. This experience will be definitely helpful for upcoming experiments and in real life application in further projects.

Attachments:

References:

1) https://www.w3schools.com/

2) https://stackoverflow.com/

3) https://youtube.com/

4) https://www.youtube.com/watch?v=PMhn9KnWU0I

5) https://www.youtube.com/watch?v=AH0-eAVWpG4