

COMP2350/COMP6350 Database Systems

Practical – Weeks 11 & 12

Objective

- To define a cursor in a stored procedure. [1]
- To understand error handling in MySQL stored procedures. [2]
- To understand SIGNAL statement and how to raise error conditions within a stored procedure.
- To create the first trigger and understand the building blocks of a trigger. [2]

Sample Database Schema

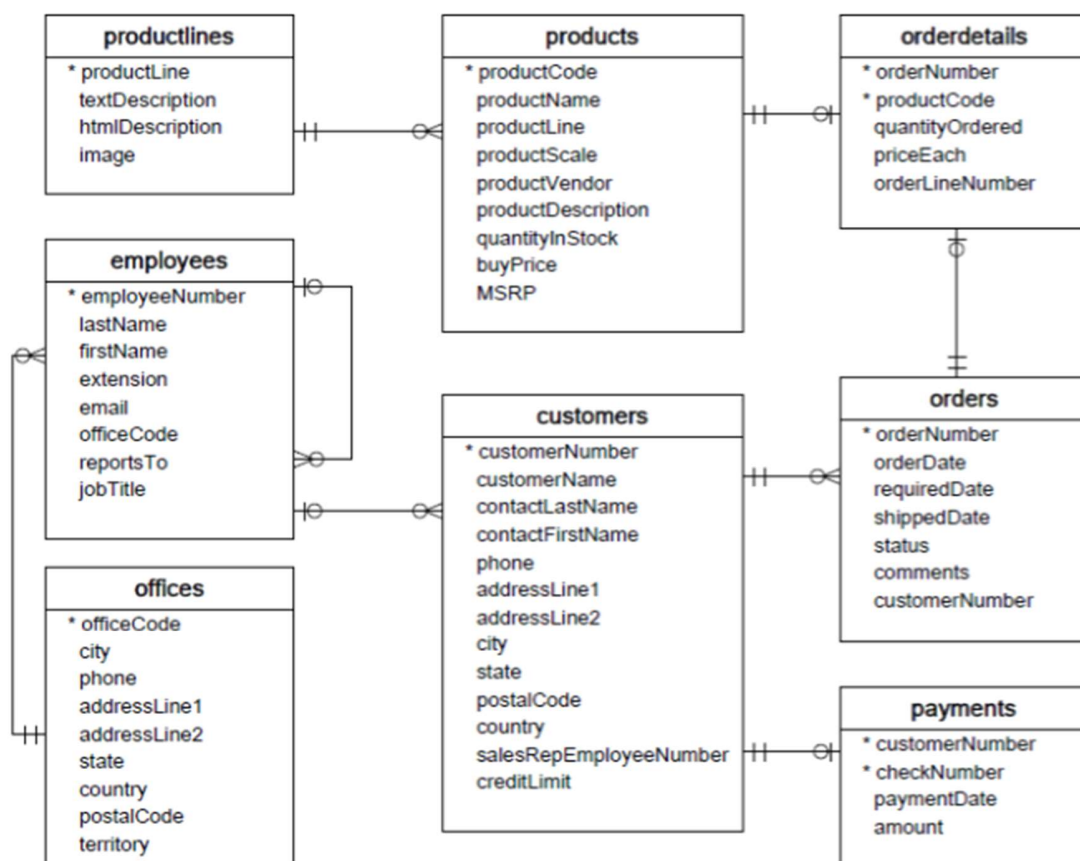


Figure 1: Sample Database Schema Diagram


MySQL Cursor

1. Creating a stored procedure with a cursor

- Run MySQL Workbench and then open a connection to the database server.
- Open a New Query tab in WorkBench environment and type in the following stored procedure into the query interface as shown in Figure 2.

```
1  DELIMITER //
2  • DROP PROCEDURE IF EXISTS country_list //
3  -- This procedure returns a comma separated list of customers countries
4  • CREATE PROCEDURE country_list (INOUT c_list TEXT)
5  BEGIN
6      DECLARE cnt varchar(255);
7      DECLARE finished INT DEFAULT 0;
8      -- Declaring the Cursor
9      DECLARE countries CURSOR FOR
10         SELECT DISTINCT country FROM customers;
11      -- Declaring error handler
12      DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;
13      -- Opening the cursor
14      OPEN countries;
15      WHILE finished != 1 DO
16          -- Retrieving each row from cursor
17          FETCH countries INTO cnt;
18          SET c_list = CONCAT(c_list, cnt, ', ');
19      END WHILE;
20      -- Closing the cursor
21      CLOSE countries;
22  END
23  //
24  DELIMITER ;
```

Figure 2: A stored procedure with a cursor

- Execute the stored procedure by pressing CTRL+SHIFT+ENTER or clicking on .

2. Calling the stored procedure

```
26 • SET @c_list = '';
27 • CALL country_list (@c_list);
28 • SELECT @c_list;
```

Figure 3: Statements to execute the stored procedure


MySQL Trigger

1. Creating the first trigger

- Run MySQL Workbench and then open a connection to the database server.
- Open a New Query tab in WorkBench environment and type in the following SQL statements into the query interface as shown in Figure 4.

```
1  -- Create products_audit table that keeps sensitive financial data
2  DROP TABLE IF EXISTS products_audit;
3  CREATE TABLE products_audit(
4      audit_id int auto_increment primary key,
5      productCode varchar(15) not null,
6      quantityInStock smallint(6),
7      buyPrice decimal(10,2),
8      MSRP decimal(10,2),
9      updateUser varchar(20) NOT NULL,
10     updateTime DATETIME NOT NULL);
11  -- Then create the trigger that inserts log into product_audit table on any update on products table
12  DELIMITER //
13  DROP TRIGGER IF EXISTS log_products_update //
14  CREATE TRIGGER log_products_update
15      BEFORE UPDATE ON products
16      FOR EACH ROW
17  BEGIN
18      INSERT INTO products_audit (productCode, quantityInStock, buyPrice, MSRP, updateUser, updateTime)
19          VALUES (OLD.productCode, OLD.quantityInStock, OLD.buyPrice, OLD.MSRP,
20                  current_user, NOW());
21  END
22  //
```

Figure 4: Script creating appropriate table and the trigger

- First, we write SQL statements to create a table, **products_audit**, that keeps track of changes on sensitive data (Line 2 – 10).
- Second, we define a trigger, **log_products_update**, that inserts log records into **products_audit** table in case of any update on products table (Line 14 - 22).
- Now select Line 1 – 22 and execute the statements by pressing CTRL+SHIFT+ENTER or clicking on .
- This will create appropriate table and the trigger. The trigger will be stored in the database as shown in Figure 5.

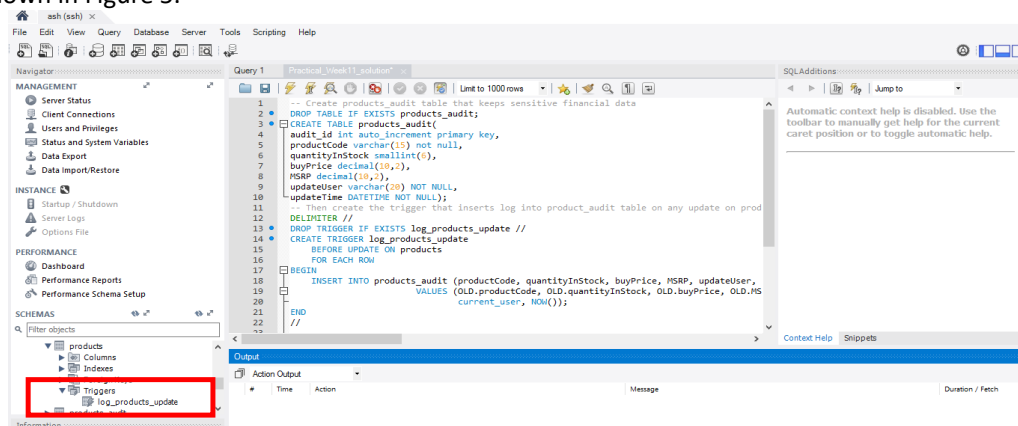


Figure 5: Trigger created and stored within database

2. Verifying the first trigger

- The trigger, **log_products_update**, will be fired when the product table gets updated.
- Figure 6 shows an excerpt of the products table from the sample database.

| productCode | productName | productLine | productSi | productVendor | productDescription | quantityIn | buyPrice | MSRP |
|-------------|--------------------------------|--------------|-----------|------------------------|---|------------|----------|--------|
| S10 1678 | 1969 Harley Davidson Ultima... | Motorcvcles | 1:10 | Min Lin Diecast | This replica features working kickstand, front su... | 7933 | 48.81 | 95.70 |
| S10 1949 | 1952 Alpine Renault 1300 | Classic Cars | 1:10 | Classic Metal Creat... | Turnable front wheels; steering function; detail... | 7305 | 98.58 | 214.30 |
| S10 2016 | 1996 Moto Guzzi 1100i | Motorcvcles | 1:10 | Highway 66 Mini Cl... | Official Moto Guzzi logos and inscriptions, saddle b... | 6625 | 68.99 | 118.94 |
| S10 4698 | 2003 Harley-Davidson Eagle ... | Motorcvcles | 1:10 | Red Start Diecast | Model features, official Harley Davidson logos a... | 5582 | 91.02 | 193.66 |
| S10 4757 | 1972 Alfa Romeo GTA | Classic Cars | 1:10 | Motor City Art Clas... | Features include: Turnable front wheels; steeri... | 3252 | 85.68 | 136.00 |
| S10 4962 | 1962 Lancia Delta 16V | Classic Cars | 1:10 | Second Gear Diecast | Features include: Turnable front wheels; steeri... | 6791 | 103.42 | 147.74 |
| S12 1099 | 1968 Ford Mustang | Classic Cars | 1:12 | Autoart Studio Des... | Hood, doors and trunk all open to reveal highlv ... | 68 | 95.34 | 194.57 |

Figure 6: Current products table

- Execute the following SQL statements to update the products table first and then display the updated products table as shown in Figure 7.

```

26 • UPDATE products
27   SET buyPrice = '50.81'
28   WHERE productCode = 'S10_1678';
29
30 • SELECT * FROM products;
31
32
33
34

```

| productCode | productName | productLine | productSi | productVendor | productDescription | quantityIn | buyPrice | MSRP |
|-------------|--------------------------------|--------------|-----------|------------------------|---|------------|----------|--------|
| S10 1678 | 1969 Harley Davidson Ultima... | Motorcvcles | 1:10 | Min Lin Diecast | This replica features working kickstand, front su... | 7933 | 50.81 | 95.70 |
| S10 1949 | 1952 Alpine Renault 1300 | Classic Cars | 1:10 | Classic Metal Creat... | Turnable front wheels; steering function; detail... | 7305 | 98.58 | 214.30 |
| S10 2016 | 1996 Moto Guzzi 1100i | Motorcvcles | 1:10 | Highway 66 Mini Cl... | Official Moto Guzzi logos and inscriptions, saddle b... | 6625 | 68.99 | 118.94 |
| S10 4698 | 2003 Harley-Davidson Eagle ... | Motorcvcles | 1:10 | Red Start Diecast | Model features, official Harley Davidson logos a... | 5582 | 91.02 | 193.66 |
| S10 4757 | 1972 Alfa Romeo GTA | Classic Cars | 1:10 | Motor City Art Clas... | Features include: Turnable front wheels; steeri... | 3252 | 85.68 | 136.00 |
| S10 4962 | 1962 Lancia Delta 16V | Classic Cars | 1:10 | Second Gear Diecast | Features include: Turnable front wheels; steeri... | 6791 | 103.42 | 147.74 |
| S12 1099 | 1968 Ford Mustang | Classic Cars | 1:12 | Autoart Studio Des... | Hood, doors and trunk all open to reveal highlv ... | 68 | 95.34 | 194.57 |

Figure 7: After performing an update – products table

- Check the records in the **products_audit** table to verify the execution of the trigger as depicted in Figure 8.

```

32 • SELECT * FROM products_audit;

```

| audit_id | productCode | quantityInStock | buyPrice | MSRP | updateUser | updateTime |
|----------|-------------|-----------------|----------|-------|--------------|---------------------|
| 1 | S10 1678 | 7933 | 48.81 | 95.70 | ma20175097@% | 2017-10-16 02:36:32 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL |

Figure 8: The trigger executes while updating products and as a result a new record inserted into products_audit table

Task Specifications:

1. Write a stored procedure that takes an employee number as input and print a message showing the employee name and who S/he reports to. The procedure prints an error message in case the employee number doesn't exist in the database.
2. Given the following stored procedure.

```
1  DELIMITER //
2  DROP PROCEDURE IF EXISTS FindCustomersByCountry //
3  CREATE PROCEDURE FindCustomersByCountry (IN country_name TEXT)
4  BEGIN
5      DECLARE cNumber INT;
6      DECLARE cName TEXT;
7      SELECT customerNumber, customerName INTO cNumber, cName
8      FROM customers
9      WHERE country = country_name;
10     SELECT cNumber AS 'Customer Number', cName AS 'Customer Name';
11 END
12 //
13 DELIMITER ;
```

Figure 9: Stored Procedure for Question 1

- a) What output would you expect by calling the procedure with the following statement?

CALL FindCustomersByCountry('Australia');
 - b) Verify your answer by executing and calling the procedure.
3. Rewrite the stored procedure given in Question 2 (see Figure 9) using a cursor. You may use the following template:

```
CREATE PROCEDURE FindCustomersByCountryWithCursor (IN country_name TEXT)
BEGIN
    ...
    DECLARE customers_country_rec CURSOR FOR
        SELECT ...
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET .....
    OPEN customers_country_rec;
    ...
    CLOSE customers_country_rec;
END
//
```

4. Test your procedure in Question 3 by printing the customer number and name for those customers who live in Australia.
5. Write a stored procedure that updates the credit limit of all the customers who live in a given country. The procedure implements the following business rules in updating the credit limit of a customer.

"If the current credit limit is greater than or equal to 100000 then credit limit will be increased by 25% of the current limit. If the current credit limit is in between 50000 and 100000 then credit limit will be increased by 15% of the current limit. For all other cases, credit limit will be increased by 5% of the current limit."

6. Execute and verify your procedure in Question 4 for customers living in USA.
7. Write a trigger to enforce the business rule that the minimum credit limit for a customer is \$500. The trigger must raise an appropriate error and gives informative feedback in case the credit limit of a customer is set to an amount less than the minimum.

You may use the following template:

```
CREATE TRIGGER creditlimit_check
... on customers
FOR EACH ROW
BEGIN
...
END
//
```

8. Write an appropriate SQL statement to verify the trigger.
9. Assume, the organization has recently adopted a new business rule as follows:

*"Total amount payable for an order must be more than or equal to 2000 dollars. Otherwise the order will not be shipped to the customer and order status will remain **Pending**. As soon as the customer buys some more products under the same order which causes the total amount payable to reach the minimum limit (2000 dollars), order status will be changed to **Confirmed**."*

- a) Write a trigger to implement the aforesaid business rule.
- b) Verify the correctness of your trigger by writing appropriate SQL statement.