Stored Procedures Triggers for MySQL

Content was adapted from the following sources:

- Presentation slides and figures from the course textbook.

References/Tutorials

- https://www.mysqltutorial.org/mysql-stored-procedure-tutorial.aspx
  - Stored Procedures
    - Creating, removing, altering, listing, variables, parameters (In, out, inout)
    - Statements (While loop, Repeat loop, If, Case, exceptions, and SELECT and cursors (for processing rows))
    - External communication (parameters, write into a file, use print with mssg, tables)
  - Stored Functions (Creating, removing, listing, querying about)
  - Security implications

References/Tutorials

- Examples and Scripts re Triggers and Stored Procedures
  - In lecture slides
  - SQL scripts for the textbook DB and triggers:

    Content -> How-to -> SQL Scripts -> Triggers.sql

  - Example
    - *Content -> How-to -> SQL Scripts -> Triggers Example.pdf*

Procedures / Triggers … Invocation

- No conditional execution or loop constructs to invoke TRIGGERS
  - Trigger is either fired based on a particular event (e.g., before a tuple is inserted into a particular table or after tuple is inserted)
  - Stored procedures and functions can be invoked from triggers, programs (i.e., any program that can connect to the DB and execute SQL statements (e.g, MySQL Workbench, application programs)
- Persistent Storage Module (PSM)
  - A block of code containing SQL statements and procedural extensions
  - Stored and executed on the DB server
- Procedural language SQL (PL/SQL) … Oracle … in the textbook and on Bspace in: *Content -> Lecture Slides*
  - SQL Script for the textbook DB can be found on Bspace in: *Content -> How-to -> SQL Scripts*
- MySQL language is similar – covered here
  - Executed as a unit
  - Triggers
  - Stored Procedures and Stored Functions

Procedural SQL based on MySQL

- Has loops and conditionals
- Data types
- Example

```
DECLARE productCount INT DEFAULT 0;
SELECT COUNT(*)
INTO productCount
FROM products;
```

MySQL DATA TYPES

DATE TYPE	SPEC	DATA TYPE	SPEC
CHAR	String (0 - 255)	INT	Integer (-2147483648 to 214748-3647)
VARCHAR	String (0 - 255)	BIGINT	Integer (-9223372036854775808 to 9223372036854775807)
TINYTEXT	String (0 - 255)	FLOAT	Decimal (precise to 23 digits)
TEXT	String (0 - 65535)	DOUBLE	Decimal (24 to 53 digits)
BLOB	String (0 - 65535)	DECIMAL	"DOUBLE" stored as string
MEDIUMTEXT	String (0 - 16777215)	DATE	YYYY-MM-DD
MEDIUMBLOB	String (0 - 16777215)	DATETIME	YYYY-MM-DD HH:MM:SS
LONGTEXT	String (0 - 4294967295)	TIMESTAMP	YYYYMMDDHHMMSS
LONGBLOB	String (0 - 4294967295)	TIME	HH:MM:SS
TINYINT	Integer (-128 to 127)	ENUM	One of preset options
SMALLINT	Integer (-32768 to 32767)	SET	Selection of preset options
MEDIUMINT	Integer (-8388608 to 8388607)	BOOLEAN	TINYINT(1)

Stored Procedure Declarations

- Stored Procedure Declaration:

```
CREATE PROCEDURE GetTotalOrder()
BEGIN
   DECLARE totalOrder INT DEFAULT 0;
   SELECT COUNT(*)   INTO totalOrder  FROM orders;
....
```

Stored Procedure Declarations

- Stored Procedure Declaration
  - Processed by DB
  - Interaction with a DBMS is through individual statements that end with a semicolon ... as individual statements => Problem: How to tell DBMS to execute a group of statements that define a procedure?

https://www.mysqltutorial.org/mysql-stored-procedure/mysql-delimiter/

```
DELIMITER $$
CREATE PROCEDURE GetTotalOrder()
BEGIN
   DECLARE totalOrder INT DEFAULT 0;
   SELECT COUNT(*)   INTO totalOrder  FROM orders;
END$$
DELIMITER ;
```

Procedural SQL – Example

```
DELIMITER $$
DECLARE
W_P1 NUMBER(3) := 0;
W_P2 NUMBER(3) := 10;
W_NUM NUMBER(2) := 0;
BEGIN
   WHILE W_P2 < 300 LOOP
            SELECT COUNT(P_CODE) INTO W_NUM
            FROM PRODUCT
            WHERE P_PRICE BETWEEN W_P1 AND W_P2;
            DBMS_OUTPUT .PUT_LINE('There are ' || W_NUM || ' Products with price between ' || W_P1 || ' and ' || W_P2);
            W_P1 := W_P2 + 1;
             W_P2 := W_P2 + 50;
   END LOOP;
END;
DELIMITER ;
```

Note: In MySQL: Use SELECT (…);

Triggers

- Procedural SQL code automatically invoked by RDBMS when given data manipulation event occurs or when invoked by an application program, or command line
- Parts of a trigger definition
  - Triggering timing: indicates when trigger's code executes
  - Triggering event: statement that causes the trigger to execute
    - Triggering level: row-level only for MySQL … for Oracle PL/SQL also table-level
- Trigger action based on conditional DML predicates
  - Actions depend on the type of DML statement that fires the trigger
    - INSERT, DELETE, UPDATE

Triggers – Create in MySQL

```
DELIMITER $$
CREATE TRIGGER TRG_PRODUCT_REORDER
AFTER INSERT ON line
FOR EACH ROW
BEGIN
        UPDATE PRODUCT
        SET P_REORDER = 1
        WHERE P_QOH <=  P_MIN;
END; $$
DELIMITER ;
Show triggers;
```

Note that DELIMITER statement is used to temporarily alter the delimiter from ";" to say "$$" so that the whole procedure would be "processed" by the DB Server.

```
95   DELIMITER $$
96   CREATE TRIGGER AFTER_INSERT_LINE
97   AFTER INSERT on line
98   FOR EACH ROW
99   BEGIN
100    UPDATE   PRODUCT
101    SET      P_REORDER = 1
102    WHERE    P_QOH <=  P_MIN;
103  END; $$
104
105  DELIMITER ;
106  Show triggers;
107
```

100% 15:106

Result Grid | Filter Rows: Search | Export:

Trigger	Event	Table	Statement	Timing
lineInsert	INSERT	LINE	BEGIN UPDATE invoice SET inv_amo…	AFTER
AFTER_INSERT_LI…	INSERT	LINE	BEGIN UPDATE PRODUCT SE…	AFTER
lineDelete	DELETE	LINE	BEGIN UPDATE invoice SET inv_amo…	AFTER

Triggers – Management in MySQL

- show triggers; … to list triggers
- drop trigger lineInsert; … delete trigger w name lineInsert
- To alter/modify a trigger … delete it and then create a new one
- SHOW CREATE TRIGGER lineInsert; … to show the create statement for a trigger

```
29  SHOW CREATE TRIGGER lineInsert;
```

100% 17:29

Result Grid | Filter Rows: Search | Export:

Trigger	sql_mode	SQL Original Statement	char
▶ lineInsert	ONLY_FULL_GROUP_BY,STRICT_TRANS_TA…	CREATE DEFINER=`root`@`localhost` TRIGGE…	utf8

Triggers … continued

- Triggers can refer to “new” and “old” values. An ”old” value cannot be changed.
- Old and new values … due to manipulating DB values in memory … Stored in a DB

```
110
111  DELIMITER $$
112  CREATE TRIGGER BEFORE_INSERT_LINE
113  BEFORE INSERT ON PRODUCT
114  FOR EACH ROW
115  BEGIN
116     IF NEW.P_QOH <= NEW.P_MIN THEN
117        SET NEW.P_REORDER := 1;
118     ELSE
119        SET NEW.P_REORDER := 0;
120     END IF;
121  END; $$
122
123  DELIMITER ;
124  Show triggers;
125
```

100% 15:124

Result Grid | Filter Rows: Search | Export:

Trigger	Event	Table	Statement	Timing
lineInsert	INSERT	LINE	BEGIN UPDATE invoice…	AFTER
AFTER_INSERT_LINE	INSERT	LINE	BEGIN UPDATE PR…	AFTER
lineDelete	DELETE	LINE	BEGIN UPDATE invoice…	AFTER
BEFORE_INSERT_LI…	INSERT	PRODUCT	BEGIN IF NEW.P_QO…	BEFO…

SQL Plus

```
SQL> CREATE OR REPLACE TRIGGER TRG_PRODUCT_REORDER
  2  BEFORE INSERT OR UPDATE OF P_QOH, P_MIN ON PRODUCT
  3  FOR EACH ROW
  4  BEGIN
  5     IF :NEW.P_QOH <= :NEW.P_MIN THEN
  6        :NEW.P_REORDER := 1;
  7     ELSE
  8        :NEW.P_REORDER := 0;
  9     END IF;
 10  END;
 11  /

Trigger created.

SQL>
```

Triggers … continued

- Trigger to update the product quantity on hand

```
127
128  DELIMITER $$
129  CREATE TRIGGER BEFORE_INSERT_LINE
130  BEFORE INSERT ON LINE
131  FOR EACH ROW
132  BEGIN
133     UPDATE     PRODUCT
134     SET        P_QOH = P_QOH + NEW.LINE_UNITS
135     WHERE      PRODUCT.P_CODE007 = NEW.P_CODE;
136  END; $$
137
138  DELIMITER ;
139  Show triggers;
```

100% 11:139

Result Grid | Filter Rows: Search | Export:

Trigger	Event	Table	Statement	Timing	Created
lineInsert	INSERT	LINE	BEGIN…	AFTER	2019-03-16 12:1
AFTER_INSERT_LINE	INSERT	LINE	BEGIN…	AFTER	2019-03-16 12:4
lineDelete	DELETE	LINE	BEGIN…	AFTER	2019-03-16 12:1
BEFORE_INSERT_LI…	INSERT	PRODUCT	BEGIN…	BEFO…	2019-03-16 13:2

SQL Plus

```
SQL> CREATE OR REPLACE TRIGGER TRG_LINE_PROD
  2  AFTER INSERT ON LINE
  3  FOR EACH ROW
  4  BEGIN
  5     UPDATE PRODUCT
  6     SET P_QOH = P_QOH - :NEW.LINE_UNITS
  7     WHERE PRODUCT.P_CODE = :NEW.P_CODE;
  8  END;
  9  /

Trigger created.

SQL>
```

Triggers … continued

- MySQL …. Trigger to update the customer balance (CUS_BALANCE in CUSTOMER TABLE)

```
155
156  DELIMITER $$
157  CREATE TRIGGER AFTER3_INSERT_LINE
158  AFTER INSERT ON LINE
159  FOR EACH ROW
160  BEGIN
161     DECLARE   W_CUS      CHAR(4);
162     DECLARE   W_TOT      DECIMAL (6,2);
163     SELECT  CUS_CODE INTO W_CUS
164     FROM     INVOICE
165     WHERE   INVOICE.INV_NUMBER007 = NEW.INV_NUMBER007;
166     SET W_TOT = NEW.LINE_UNITS * NEW.LINE_PRICE;
167     UPDATE      CUSTOMER
168     SET         CUS_BALANCE = W_TOT
169     WHERE       CUS.CUS_CODE007 = W_CUS;
170  END; $$
171
172  DELIMITER ;
173  Show triggers;
```

100% 15:173

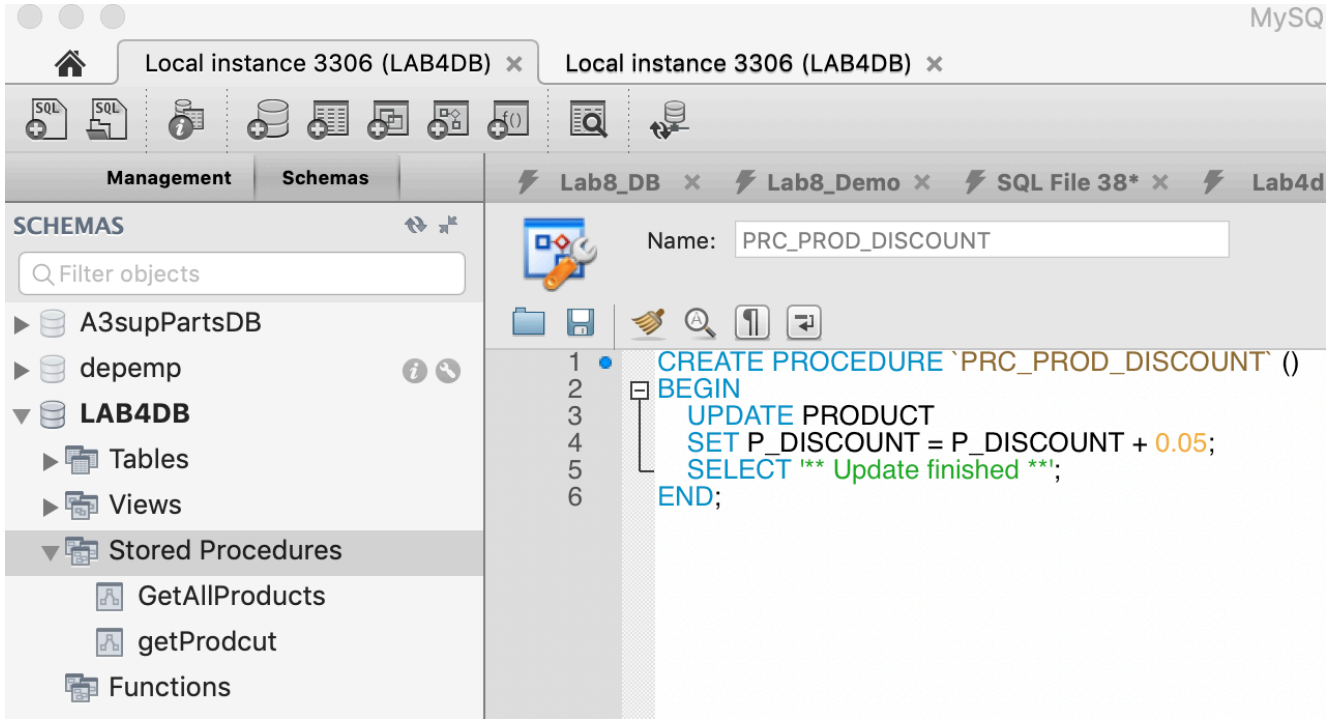
Result Grid | Filter Rows: Search | Export:

Trigger	Event	Table	Statement	Timing	Created
lineInsert	INSERT	LINE	BEGIN…	AFTER	2019-03-16 12:15:30.40
AFTER_INSERT_LINE	INSERT	LINE	BEGIN…	AFTER	2019-03-16 12:45:39.97
AFTER2_INSERT_LI…	INSERT	LINE	BEGIN…	AFTER	2019-03-16 13:47:47.94
AFTER3_INSERT_LI…	INSERT	LINE	BEGIN…	AFTER	2019-03-16 14:00:47.12
lineDelete	DELETE	LINE	BEGIN…	AFTER	2019-03-16 12:12:40.33
BEFORE_INSERT_LI…	INSERT	PRODUCT	BEGIN…	BEFO…	2019-03-16 13:24:32.42

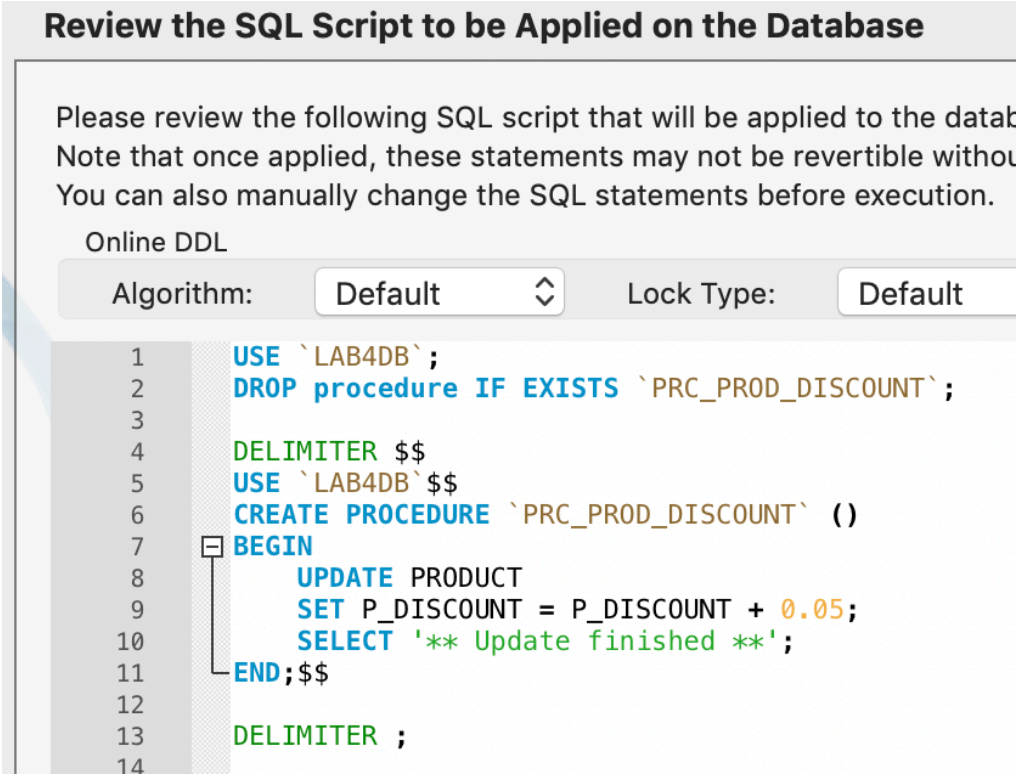
Stored Procedures - Example

- Assign an additional 5 percent discount for all products when the quantity on hand is more than or equal to twice the minimum quantity.

Right Click on Stored Procedures to code



After clicking on Apply button



Stored Procedures – Managing in MySQL

- To list stored procedures:
  - http://www.mysqltutorial.org/lis ting-stored-procedures-in-mysql-database.aspx

  - Use: SHOW PROCEDURE STATUS [LIKE 'pattern' | WHERE expr];
  - SHOW PROCEDURE STATUS; … to list all store procedures
  - SHOW PROCEDURE STATUS WHERE db = ‘TEXTDB’; … to list procedures in a DB schema
  - SHOW PROCEDURE STATUS WHERE name LIKE '%product%’; … to list procedures with pattern in names
  - SHOW CREATE PROCEDURE getProduct; … to display procedure’s code

Stored Procedures - Example

To call a procedure in MySQL

To call a procedure in MySQL

```
17  SHOW PROCEDURE STATUS WHERE name LIKE '%product%';
18
19  SHOW CREATE PROCEDURE getProdcut;
20
21  CALL GetAllProducts();
22  CALL getProdcut ('13-Q2/P2');
```

00% 32:22

Result Grid Filter Rows: Search Export:

P_CODE007	P_DESCRIPT	P_INDATE	P_QOH	P_MIN	P_PRICE	P_DISCOUNT	V_CODE	P_REO
13-Q2/P2	7.25-in. pwr. saw blade	2015-12-13 00:00:00	32	40	14.99	0.05	21344	1

Stored Procedures – Example with Parameter in MySQL

- Same example, but the discount is passed as a parameter – in MySQL

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `getProdcut`(IN productCode varchar (10))
BEGIN
    SELECT *
    FROM PRODUCT
    WHERE P_CODE007 = productCode;
END
```

```
SHOW PROCEDURE STATUS WHERE name LIKE '%product%';

SHOW CREATE PROCEDURE getProdcut;

CALL GetAllProducts();

CALL getProdcut ('13-Q2/P2');
```

Triggers Calling Stored Procedures in MySQL

- Triggers can call stored procedures --MySQL Examples
- Example: Procedure to update reorder status invoked by triggers

Procedure

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `updateReorder`(P_CODE_IN varchar(10))
BEGIN
   DECLARE  W_QOH     int;
   DECLARE  W_MIN     int;

   SELECT  P_QOH INTO W_QOH
   FROM    PRODUCT
   WHERE   P_CODE007 = P_CODE_IN;

   SELECT  P_MIN INTO W_MIN
   FROM    PRODUCT
   WHERE   P_CODE007 = P_CODE_IN;

   IF (W_QOH <  W_MIN) THEN
      UPDATE  PRODUCT
      SET     P_REORDER = 1
      WHERE   PRODUCT.P_CODE007 = P_CODE_IN;
   ELSE
      UPDATE  PRODUCT
      SET     P_REORDER = 0
      WHERE   PRODUCT.P_CODE007 = P_CODE_IN;
   END IF;
END
```

Triggers Calling Stored Procedures in MySQL … continued

- Procedure to update reorder status invoked by triggers
  - The procedure should also update QOH in product by the difference in update (using NEW.LINE_UNIITS and OLD.LINE_UNITS

```
CREATE ` PROCEDURE `updateReorder`(P_CODE_IN varchar(10))
BEGIN
   DECLARE  W_QOH                int;
   DECLARE  W_MIN                int;

   SELECT   P_QOH INTO W_QOH
   FROM     PRODUCT
   WHERE    P_CODE007 = P_CODE_IN;

  SELECT    P_MIN INTO W_MIN
   FROM     PRODUCT
   WHERE    P_CODE007 = P_CODE_IN;

            IF (W_QOH <  W_MIN) THEN
                     UPDATE     PRODUCT
                     SET                    P_REORDER = 1
                     WHERE      PRODUCT.P_CODE007 = P_CODE_IN;
            ELSE
                     UPDATE     PRODUCT
                     SET                    P_REORDER = 0
                     WHERE      PRODUCT.P_CODE007 = P_CODE_IN;
            END IF;
END
```

```
1  CREATE DEFINER=`root`@`localhost` PROCEDURE `updateReorder`(P_COD
2  BEGIN
3     DECLARE  W_QOH     int;
4     DECLARE  W_MIN     int;
5
6     SELECT  P_QOH INTO W_QOH
7     FROM    PRODUCT
8     WHERE   P_CODE007 = P_CODE_IN;
9
10    SELECT  P_MIN INTO W_MIN
11    FROM    PRODUCT
12    WHERE   P_CODE007 = P_CODE_IN;
13
14    IF (W_QOH <  W_MIN) THEN
15       UPDATE  PRODUCT
16       SET     P_REORDER = 1
17       WHERE   PRODUCT.P_CODE007 = P_CODE_IN;
18    ELSE
19       UPDATE  PRODUCT
20       SET     P_REORDER = 0
21       WHERE   PRODUCT.P_CODE007 = P_CODE_IN;
22    END IF;
23 END
```

Triggers Calling Stored Procedures in MySQL … continued

After Insert on Line Trigger to update reorder

```
DELIMITER $$
CREATE TRIGGER after_line_insert
   AFTER INSERT ON line
   FOR EACH ROW
BEGIN
   DECLARE W_PRODUCT_CODE varchar (10);
            SET W_PRODUCT_CODE = NEW.P_CODE;
   CALL updateReorder (W_PRODUCT_CODE);
END$$
DELIMITER ;

show triggers;
```

```
34  DELIMITER $$
35  CREATE TRIGGER after_line_insert
36     AFTER INSERT ON line
37     FOR EACH ROW
38  BEGIN
39     DECLARE W_PRODUCT_CODE varchar (10);
40     SET W_PRODUCT_CODE = NEW.P_CODE;
41     CALL updateReorder (W_PRODUCT_CODE);
42  END$$
43  DELIMITER ;
44
45  show triggers;
```

0% 15:45

esult Grid | Filter Rows: Search | Edit: | Export/Import:

P_CODE007	P_DESCRIPT	P_INDATE	P_QOH	P_MIN	P_PRICE	P_DISCOUNT	V_CODE	P_REORDER
13-Q2/P2	7.25-in. pwr. saw blade	2015-12-13 00:00:00	32	40	14.99	0.05	21344	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Triggers Calling Stored Procedures in MySQL … continued

After Update on Line Trigger to update reorder

```
46
47  DELIMITER $$
48  CREATE TRIGGER after_line_update
49     AFTER UPDATE ON line
50     FOR EACH ROW
51  BEGIN
52     DECLARE W_PRODUCT_CODE varchar (10);
53     SET W_PRODUCT_CODE = NEW.P_CODE;
54     CALL updateReorder (W_PRODUCT_CODE);
55  END$$
56  DELIMITER ;
57
58  show triggers;
```

0% 15:58

esult Grid | Filter Rows: Search | Export:

Trigger	Event	Table	Statement	Timing	Create
after_line_insert	INSERT	LINE	BEGIN...	AFTER	2019-(
after_line_update	UPDATE	LINE	BEGIN...	AFTER	2019-(

```
DELIMITER $$
CREATE TRIGGER after_line_update
   AFTER update ON line
   FOR EACH ROW
BEGIN
   DECLARE W_PRODUCT_CODE varchar (10);
            SET W_PRODUCT_CODE = NEW.P_CODE;
   CALL updateReorder (W_PRODUCT_CODE);
END$$
DELIMITER ;

show triggers;
```

Questions and Answers (Q/A)