Module 14: MySQL Condition Handlers and Cursors

Condition Handler:

- DECLARE … HANDLER is used to

- They are also referred as condition / error / exception handler.
- The syntax is:
  - **DECLARE {CONTINUE | EXIT } HANDLER**
  - **FOR {mysql_error_code | SQLSTATE sqlstate_code | named_condition}**
  - **handler_actions;**

- **Condition Handler has three clauses:**

- **1) Handler Type:**
  - **CONTINUE** will continue the block of code when an error occurs.
  - **EXIT** will exit the current block of code when an error occurs.
  - Note: in either case, handler_actions are run before EXIT or CONTINUE takes place

- **2) Handler Conditions:**
  - MySQL error code: are set of error codes that are unique to MySQL server.
  - SQLSTATE error code: are defined by ANSI standard and hence database independent.
  - Note: some SQLState codes are associated with many MySQL error code

- 3) **Handler Actions:**
  - Statements to perform if error occurs.
  - Multiple statements should be written in BEGIN and END clause.

Named Conditions:

- They are used for improving the readability of handlers by defining condition declaration, which associates a MySQL error code or SQLSTATE code with meaningful name that we can use in handler declarations.

- The syntax is:
  - **DECLARE condition_name CONDITION FOR {SQLSTATE sqlstate_code | MySQL_error_code}**

- Named Conditions must be declared

SIGNAL:

- □ It's a good programming practice to prevent errors by checking the parameters before they are used to make sure they are valid. This is called data validation.
- □ If the data isn't valid we can raise an error using SIGNAL statement.
- □ The syntax is:
    - □ **SIGNAL SQLSTATE sqlstate_value**
    - □ **[SET MESSAGE_TEXT = message] [,MYSQL_ERRNO = mysql_error_number]]**
- □ The SIGNAL statement is followed by SQLSTATE keyword, followed by SQLSTATE code.
- □ We can optionally include a SET statement that sets a message and MySQL error code for the error.
- □ If the raised error is not handled den we get the following error message:
    - □ ERROR 1644 (23002): Unhandled user-defined exception condition

Stored Routines – MCQ -3

Q1) what is the output of following code snippet assuming table 'tt66' does not exist?

```
create procedure tp10()
begin
select 'start ' as MSG;
declare continue handler for 1146
select 'issue ' as msg;
select * from tt66;
select 'stop ' as MSG;
end $$
```

Options:

A. start issue B. issue

C. start issue stop D. code does not compile

Solution:

Q2) what is the output of following code snippet assuming table 'tt66' does not exist?

```
create procedure tp3()
begin
declare exit handler for 1146
select 'issue ' as msg;
select 'donald ' as MSG;
select * from tt66;
select 'duck ' as MSG;
end $$
```

Options:

A. donald B. issue

C. donald issue duck D. donald issue

Solution:

CURSOR:

- → Cursor enables us to

- Because of this row orientation, cursors are often used in loops that fetch and process a row with each iteration of the loop.
- Cursor Properties:
  - They are read-only, they cannot be used to modify tables.
  - They are not scrollable, they can only advance through result set row by row.

Working with CURSOR:

- **1) Declare CURSOR:**
  - To use a cursor in a stored routine we declare cursor statement that names the cursor and associate it with a SELECT statement that produces a result set.
  - **Syntax: DECLARE cursor_name CURSOR FOR select_statement.**

- **2) Open CURSOR:**
  - Open executes the SELECT statement associated with the cursor.
  - **Syntax: OPEN cursor_name**

- **3) Fetch from CURSOR:**
  - It fetches the next row of an open cursors result set.
  - **Syntax: FETCH cursor_name INTO var_name [, var_name]**

- **4) Close the CURSOR:**
  - Once the processing is done then we need to close the cursor.
  - **Syntax: CLOSE cursor_name**

While do:

- WHILE Statement provides the entry controlled looping feature in MySQL.
- The syntax is:
  - **WHILE boolean_expression DO**
    - **Statement_1;**
    - **[Statement_2;] …**
  - **END WHILE;**
- The conditional boolean expression is evaluated and the loop terminates if the condition is not true. Otherwise, the statement list within the loop executes, control transfers back to the beginning and the expression is tested again.

Repeat Until:

- □ REPEAT UNTIL Statement provides the exit controlled looping feature in MySQL.
- □ The syntax is:
  - □ **REPEAT**
    - □ **Statement_1;**
    - □ **[Statement_2;] …**
  - □ **UNTIL boolean_condition**
  - □ **END REPEAT;**
- □ The statement within the loop are executed and then the boolean_condition is evaluated. If the expression is true the loop terminates else it begins again.

Stored Routines – MCQs – 4

Q1) What is the correct order of working with cursor?

Options:

A. declare, fetch and close.   B. declare, open, fetch and close.
C. fetch, declare and close.   D. fetch, open and close

Solution:

Q2) Which two are true regarding cursor?

Options:

A. cursors can only read data from table.

B. cursor can read and modify data from table.

C. cursor are not scrollable.

D. cursor are scrollable.

Solution:

Q3) what is the result of following code snippet:

```
declare i int default 1;
declare sum int default 0;
while i <= 3
      set sum = sum + i;
      set i = i + 1;
end while;
select sum;
```

Options:

A. code will not compile   B. 3
C. 6   D. 10-

Solution:

Q4) what is the result of following code snippet:

```
declare i int default 1;
while i <= 5 do
        select i;
        i = i + 1;
end while;
end $$
```

Options:

A. code will not compile   B. 1 2 3 4 5

C. 1 2 3 4   D. 1 2 3 4 5 6

Solution:

Q5) what is the result of following code snippet:

```
declare i int default 1;
repeat
        select i;
        set i = i + 1;
until i = 3;
end repeat;
```

Options:

A. 1 2   B. 1 2 3

C. 1 2 3 4   D. code will not compile

Solution: