Working with MySQL cursor

First, declare a cursor by using the `DECLARE` statement:

DECLARE cursor_name CURSOR FOR SELECT_statement;

The cursor declaration must be after any variable declaration. If you declare a cursor before the variable declarations, MySQL will issue an error. A cursor must always associate with a `SELECT` statement.

Next, open the cursor by using the `OPEN` statement. The `OPEN` statement initializes the result set for the cursor, therefore, you must call the `OPEN` statement before fetching rows from the result set.

OPEN cursor_name;

Then, use the `FETCH` statement to retrieve the next row pointed by the cursor and move the cursor to the next row in the result set.

FETCH cursor_name INTO variables list;

After that, check if there is any row available before fetching it.

Finally, deactivate the cursor and release the memory associated with it using the `CLOSE` statement:

CLOSE cursor_name;

It is a good practice to always close a cursor when it is no longer used.

When working with MySQL cursor, you must also declare a `NOT FOUND` handler to handle the situation when the cursor could not find any row.

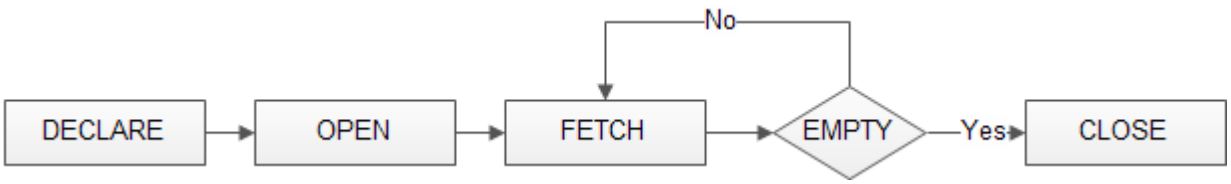
Because each time you call the `FETCH` statement, the cursor attempts to read the next row in the result set. When the cursor reaches the end of the result set, it will not be able to get the data, and a condition is raised. The handler is used to handle this condition.

To declare a **`NOT FOUND` handler**, you use the following syntax:

DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;

The **`finished`** is a variable to indicate that the cursor has reached the end of the result set. Notice that the handler declaration must appear after variable and cursor declaration inside the stored procedures.

The following diagram illustrates how MySQL cursor works.



CursorExample :

CREATE PROCEDURE p25 (OUT return_val INT)

BEGIN DECLARE a,b INT;

DECLARE cur_1 CURSOR FOR SELECT s1 FROM t;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET b = 1;

OPEN cur_1;

REPEAT FETCH cur_1 INTO a;

UNTIL b = 1

END REPEAT;

CLOSE cur_1;

SET return_val = a;

END;//

```
mysql> CALL p25(@return_val)//
 Query OK, 0 rows affected (0.00 sec)
mysql> SELECT @return_val//
+-------------+
| @return_val |
+-------------+
     | 5 |
+-------------+
1 row in set (0.00 sec)
```

2)Develop a cursor that creates an email list of all employees in the employees table.

```
DELIMITER $$
CREATE PROCEDURE createEmailList (
    INOUT emailList varchar(4000)
)
BEGIN
    DECLARE finished INTEGER DEFAULT 0;
    DECLARE emailAddress varchar(100) DEFAULT "";

    -- declare cursor for employee email
    DECLARE curEmail
        CURSOR FOR
            SELECT email FROM employees;

    -- declare NOT FOUND handler
    DECLARE CONTINUE HANDLER
        FOR NOT FOUND SET finished = 1;
```

```
    OPEN curEmail;

    getEmail: LOOP
        FETCH curEmail INTO emailAddress;
        IF finished = 1 THEN
            LEAVE getEmail;
        END IF;
        -- build email list
        SET emailList = CONCAT(emailAddress,";",emailList);
    END LOOP getEmail;
    CLOSE curEmail;

END$$
DELIMITER ;
```

Mysql> SET @emailList = "";

Mysql> CALL createEmailList(@emailList);
Mysql> SELECT @emailList;

There are two types of cursors:

An ***IMPLICIT*** **cursor** is automatically declared by Oracle every time an SQL statement is executed. The user will not be aware of this happening and will not be able to control or process the information in an implicit cursor.

An ***EXPLICIT*** **cursor** is defined by the program for any query that returns more than one row of data. That means the programmer has declared the cursor within the PL/SQL code block.

Table 8.1 ■ Explicit Cursor Attributes

Cursor Attribute	Syntax	Explanation
%NOTFOUND	cursor_name%NOTFOUND	A Boolean attribute that returns TRUE if the previous FETCH did not return a row, and FALSE if it did.
%FOUND	cursor_name%FOUND	A Boolean attribute that returns TRUE if the previous FETCH returned a row, and FALSE if it did not.
%ROWCOUNT	cursor_name%ROWCOUNT	# of records fetched from a cursor at that point in time.
%ISOPEN	cursor_name%ISOPEN	A Boolean attribute that returns TRUE if cursor is open, FALSE if it is not.