« 13.6.5.8 WHILE Syntax
13.6.6.1 Cursor CLOSE Syntax »

13.6.6. Cursors

[+/-]

MySQL supports cursors inside stored programs. The syntax is as in embedded SQL. Cursors have these properties:

- Asensitive: The server may or may not make a copy of its result table
- Read only: Not updatable
- Nonscrollable: Can be traversed only in one direction and cannot skip rows

Cursor declarations must appear before handler declarations and after variable and condition declarations.

Example:

```
CREATE PROCEDURE curdemo()
BEGIN
  DECLARE done INT DEFAULT FALSE;
  DECLARE a CHAR(16);
  DECLARE b, c INT;
  DECLARE cur1 CURSOR FOR SELECT id,data FROM test.t1;
  DECLARE cur2 CURSOR FOR SELECT i FROM test.t2;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

  OPEN cur1;
  OPEN cur2;

  read_loop: LOOP
    FETCH cur1 INTO a, b;
    FETCH cur2 INTO c;
    IF done THEN
      LEAVE read_loop;
    END IF;
    IF b < c THEN
      INSERT INTO test.t3 VALUES (a,b);
    ELSE
      INSERT INTO test.t3 VALUES (a,c);
    END IF;
  END LOOP;

  CLOSE cur1;
  CLOSE cur2;
END;
```

Copyright © 1997, 2013, Oracle and/or its affiliates. All rights reserved. Legal Notices

Previous / Next / Up / Table of Contents

User Comments

Posted by Roland Bouman on February 23 2006 2:44pm	[Delete] [Edit]

Hi all, as beat pointed out, LOOP is the only construct that I found useful for traversing cursors. Both WHILE and REPEAT lead to problems. See http://rpbouman.blogspot.com/2005/09/why-repeat-and-while-are-usually-not.html for examples:

Nesting cursors is possible, you just need to take care of a few things. You can either put the inner cursor in it's own BEGIN..END block, duplicating the code for the handler and the loop control, or you can reset your loop control variable inside the inner loop once the cursor is exhausted. I won't post an example here, as this would soon clutter the page. Just check it out here, it's got all the examples:
http://rpbouman.blogspot.com/2005/10/nesting-mysql-cursor-loops.html

Posted by Alexander Pelov on March 15 2006 6:54pm	[Delete] [Edit]

Hello everyone,

I comment the examples given on this page.

I had several working stored functions that when called from a "fetching" cursor loop didn't work (exited prematurely). The problem was that the functions were issuing SELECT statements which sometimes returned empty resultsets. This in turn executed the declared HANDLER, which interrupted the function(!) and set the variable 'done' to true -> after the first itteration the loop was acting as if the end of the data was hit.

The solution I found is somewhat cumbersome (but works!) - I've redeclared the HANDLER in each of the functions :

/* Disable DATA NOT FOUND handlers from calling functions */
DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' BEGIN END;

Posted by charles rogers on May 9 2007 1:55pm	[Delete] [Edit]

There are really helpful comments posted here and throughout the manual. PLEASE, everyone, when you post do not forget to mention the VERSION of MySQL you were using when you encountered your issue. Without the VERSION information, the report of a bug or the description of a technical feature or a problem is lessened in value to the community.

Thanks!

Posted by Horga Marius on November 17 2007 10:55am	[Delete] [Edit]

it is possible to nest 2 cursor an find the number of rows for each cursor. We can use SQL_CALC_FUND_ROWS in the cursor.
See examples.

```
DELIMITER $$

DROP PROCEDURE IF EXISTS `igla`.`twoCursor` $$
CREATE PROCEDURE `igla`.`twoCursor` ()
BEGIN
declare idFact, idMaster, idSlave, indexM, indexS int default 0;

declare masterCursor cursor for Select SQL_CALC_FOUND_ROWS Factura_ID from facturi;
declare slaveCursor cursor for Select SQL_CALC_FOUND_ROWS iesire_id from iesiri where factura_ID
=idFact;

open masterCursor;

set idMaster = (Select FOUND_ROWS());

while indexM<idMaster do
fetch masterCursor into idFact;

open slaveCursor;

set idSlave = (Select FOUND_ROWS());

while indexS<idSlave do
-- do some hear

set indexS= indexS+1;
end while;

close slaveCursor;
set indexM=indexM+1;
end while;

close masterCursor;

END $$

DELIMITER ;
```

Posted by Dipak Patel on November 29 2007 8:27pm	[Delete] [Edit]

How you can use nested loop,here each dept has one or more majors.

```
/*
Write an SP that retrieves from your CS442 database all the departments and the majors offered by each department, and insert the results in the MESSAGES table. Here are detailed instructions:
Store a string composed of the department ID and the major description in the MESSAGES table (separate the two with an asterisk (*) symbol by using the concat function). Your MESSAGES table should contain 16 records and look similar to the following:
BUS*Marketing
COM*Communications
CS*Computer Networking
```

```
*/
-- Drop Procedure
DROP PROCEDURE IF EXISTS get_major_info_for_all_dept;

DELIMITER //
-- Create procedure get_major_info_for_all_dept()
CREATE PROCEDURE get_major_info_for_all_dept()

BEGIN

DECLARE dept_id VARCHAR(3) ;
DECLARE major_description VARCHAR(50) ;
DECLARE concat_description VARCHAR(60);
DECLARE flag1 VARCHAR(5) DEFAULT 'START' ;
DECLARE flag2 VARCHAR(5) DEFAULT 'START' ;

-- Declare cursor for outside looping purpose to get all departments from the dept table
DECLARE getDept CURSOR FOR
SELECT deptid
FROM dept
ORDER BY deptid;
--Declare cursor for getting each and every major description of each department.
DECLARE getMajorsFromEachDept CURSOR FOR
SELECT description
FROM major
WHERE deptid = dept_id;

-- Record not found means for inner loop no more major record left for that dept_id and for outer loop no
more record left for dept-id
DECLARE CONTINUE HANDLER FOR NOT FOUND SET flag1 = 'END';

-- Open cursor
OPEN getDept;

-- loop to extract each major of each department and insert into messages table
-- until no record left.

WHILE flag1<>'END' DO
-- to get deptid
FETCH getDept INTO dept_id;

IF flag1 <> 'END' THEN

OPEN getMajorsFromEachDept;

SET flag2= flag1;
WHILE flag1 <> 'END' DO
FETCH getMajorsFromEachDept INTO major_description;
IF flag1 <> 'END' THEN
SET concat_description = CONCAT(dept_id,'*',major_description);
INSERT INTO messages
VALUES (concat_description);
END IF;
END WHILE;
-- free memory
CLOSE getMajorsFromEachDept;
```

```
SET flag1 = flag2;
END IF;
END WHILE;
-- Loop end.

-- free memory
CLOSE getDept;

--- Display all records of Messages Table---------
SELECT * FROM messages;

END;
//

DELIMITER ;
```

Purvi

Posted by Aaron Diers on March 6 2008 9:15pm	[Delete] [Edit]

Just as a response to that last comment, wouldn't it be easier just to do this?

```
CREATE PROCEDURE get_major_info_for_all_dept()
INSERT INTO messages
SELECT CONCAT(dept.deptid, '*', major.description)
FROM dept JOIN major ON dept.deptid = major.dept_id
```

Posted by steffen hirsch on June 18 2008 6:14pm	[Delete] [Edit]

hello folks,

you cannot reopen a cursor, if you changed rows hold by the cursor after scrolling through it for the first time!!

I'm using mysql 5.1.22-rc

regards,
s

Posted by Serdar S. Kacar on March 10 2009 2:04pm	[Delete] [Edit]

Are you looking for an alternative cursor traversal code template in which there is no loop control variable (like "done") and no label (like "MyLabel:") ?

Take a look at "Simple Cursor Traversal 2"
http://forge.mysql.com/tools/tool.php?id=186

Posted by Dominik Egert on May 19 2010 7:30pm	[Delete] [Edit]

This method is stored function safe, even when nested into each other. (Like Function A calls Function B; both using cursors)

```
-- Initialise cursor
OPEN cursor40201;

-- USE BEGIN-END handler for cursor-control within own BEGIN-END block
BEGIN
```

```
DECLARE EXIT HANDLER FOR NOT FOUND BEGIN END;

-- Loop cursor throu users of group
LOOP

-- Get next value
FETCH cursor40201 INTO ;

-- Add update useracls
IF (SELECT libcrm_user_updateeffective(idUserU0204)) <=> NULL THEN

-- Return error
RETURN CONCAT('ERROR 404: iIDUser(', idUserU0204, '), unknown. - NOT FOUND');

END IF;

END LOOP;

END;

-- Release cursor
CLOSE cursorU0204;
```

Posted by David Bergan on February 23 2012 10:00pm	[Delete] [Edit]

In response to Alexander Pelov's situation...

The problem is that when one of your SELECT INTO queries within the loop generates 0 results... it trips the CONTINUE HANDLER and sets "done = TRUE". The easiest way around this is to simply add the line "SET done = FALSE ;" immediately before the FETCH command(s) like so...

```
read_loop: LOOP
-- This is the line that fixes the problem
SET done = FALSE ;

FETCH cur1 INTO a, b;
FETCH cur2 INTO c;

IF done THEN
LEAVE read_loop;
END IF;
IF b < c THEN
INSERT INTO test.t3 VALUES (a,b);
ELSE
INSERT INTO test.t3 VALUES (a,c);
END IF;

-- This is the kind of line that would cause the problem.
-- SELECT INTO returns 0 records, triggering the CONTINUE HANDLER and setting done to TRUE (which
we don't want)
SELECT id INTO a FROM test.t1 WHERE 1 = 2 ;
END LOOP;
```

This way, "done" is reset to false from whatever else went on during the loop and the "IF done THEN" check could only be true when FETCH runs out of records.

Posted by Brent Roady on May 9 2012 8:22pm	[Delete] [Edit]

It should be noted that the local variable names used in FETCH [cursor] INTO must be different than the variable names used in the SELECT statement defining the CURSOR. Otherwise the values will be NULL.

In this example,

```
DECLARE a VARCHAR(255);
DECLARE cur1 CURSOR FOR
SELECT a FROM table1;
FETCH cur1 INTO a;
```

the value of a after the FETCH will be NULL.

This is also described here: http://bugs.mysql.com/bug.php?id=28227

Add your own comment.

ORACLE © 2013, Oracle Corporation and/or its affiliates