CURSORES en MySQL - Definición

(https://dev.mysql.com/doc/refman/8.0/en/cursors.html)

Un cursor es una estructura que se utiliza dentro de procedimientos, funciones o disparadores que permite recorrer un conjunto de filas devueltas por una consulta y que permite "procesar" con cada fila en consecuencia.

Un cursor en MySQL es solo de lectura, ya que no puede actualizar los datos de la tabla subyacente a través del cursor.

Trabajando con cursores en MySQL

En primer lugar, se tiene que declarar un cursor utilizando la sentencia **DECLARE**:

DECLARE nombre_cursor CURSOR FOR consulta_(SELECT …..);

La declaración del cursor debe ser después de cualquier declaración de variables. Si se declara un cursor antes de la declaración de variables, MySQL emitirá un error. Un cursor siempre debe estar asociado a una sentencia **SELECT**.

A continuación, se abre el cursor utilizando la sentencia **OPEN**. La sentencia **OPEN** inicializa el conjunto de resultados para el cursor, por lo tanto, usted debe llamar a la sentencia **OPEN** antes de capturar las filas del conjunto de resultados.

OPEN nombre_cursor;

Una vez "abierto" el cursor utilizaremos la sentencia **FETCH** movernos a través de cada uno de los registros y poder recuperar la información almacenada. La información se deberá asignar en variables y de esa forma podremos trabajar con dicha información.

Si no existen más registros disponibles, ocurrirá una condición de Sin Datos con el valor **SQLSTATE 02000 (que será deberá ser gestionada por un handler *)**.

FETCH nombre_cursor INTO nombre_variable;

Finalmente, se llama la sentencia **CLOSE** para desactivar el cursor y liberar la memoria asociada a ella de la siguiente manera:

CLOSE nombre_cursor;

Cuando ya no se utiliza el cursor, se deber cerrar dicho cursor.

(*) Cuando se trabaja con cursores en **MySQL**, también debe declarar un **handler o manejador habitualmente de tipo NOT FOUND** para gestionar la situación cuando el cursor no pudo encontrar ninguna fila. Debido a que cada vez que se llama a la instrucción **FETCH**, el cursor se intenta leer la siguiente fila del conjunto de resultados. Cuando el cursor llega al final del conjunto de resultados, no va a ser capaz de obtener los datos y provoca un **evento**. El **handler** que se utiliza para manejar esta situación es.

Para declarar el evento **`NOT FOUND`** , se utiliza la siguiente sintaxis:

```
DECLARE CONTINUE HANDLER FOR NOT FOUND SET terminado = 1;
```

Donde **terminado** es una variable para indicar que el cursor ha alcanzado el final del conjunto de resultados. Observe que la declaración del controlador debe aparecer después de variable y declaración de cursor dentro de los procedimientos almacenados.

Ejemplo de un cursor en MySQL

Vamos a desarrollar un procedimiento almacenado que crea una lista de correo electrónico de todos los empleados de la tabla empleados en la base de datos MySQL.

empleados		
ID	**nombre**	**email**
1	Antonio Valls Aquino	aquinoantonio@hotmail.com
2	Juan Perez Gomez	gomezjuan@hotmail.com
3	Pedro Cruz Lopez	lopezpedro@hotmail.com

En primer lugar, tendremos que declarar algunas variables auxiliares, un cursor para recorrer los **email** de la tabla **empleado** , y un handler de tipo **`NOT FOUND para gestionar el final del cursor`**:

```
DECLARE v_finished INTEGER DEFAULT 0;

DECLARE v_email varchar(255) DEFAULT "";

 -- declare cursor for employee email

        DECLARE email_cursor CURSOR FOR SELECT email FROM empleado;

 -- declare NOT FOUND handler

        DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished = 1;

 --a continuación, abrimos el email_cursor utilizando la sentencia OPEN:

        OPEN email_cursor;
```

-- Después, creamos un bucle (while) para recorrer cada uno de los registros de la consulta realizada anteriormente y almacenada en nuestra variable de tipo cursor:

```
obtener_email: LOOP
    -- recorremos registro por registro hasta que no tenga más registros
    -- al final se activara el HANDLER FOR NOT FOUND , el cual actualiza la variable
    -- v_finished en 1 automáticamente y terminara el ciclo en la siguiente comparación
  FETCH email_cursor INTO v_email;
   IF v_finished = 1 THEN
      LEAVE obtener_email;
   END IF;
   -- concantemos en un parámetro (email_lista)  de tipo inout el valor del email
   SET email_lista = CONCAT(v_email ,";", email_lista);
END LOOP obtener_email;
```

-- Finalmente, cerramos el cursor utilizando la sentencia CLOSE:

```
CLOSE email_cursor;
```

Codigo Completo en MySql:

```
DELIMITER $$
CREATE PROCEDURE construir_email_lista (  INOUT email_lista VARCHAR(1000) )
 BEGIN
   DECLARE v_finished INTEGER DEFAULT 0;
   DECLARE v_email VARCHAR(255) DEFAULT "";

   DECLARE email_cursor CURSOR FOR SELECT nombre, email FROM empleado;
   DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished = 1;

   OPEN email_cursor;
   obtener_email: LOOP
         FETCH email_cursor INTO variable_nombre, v_email;
         IF v_finished = 1 THEN
            LEAVE obtener_email;
         END IF;
         /* SET email_lista = CONCAT(v_email,";",email_lista);*/ /* concateno al final*/

          SET email_lista = CONCAT(email_lista,";", v_email); /* concateno al principio*/
      END LOOP obtener_email;
   CLOSE email_cursor;

  /* quito el ; del principio*/

   select SUBSTRING(email_lista, 2)  into email_lista;

   select substr(email_lista, 2,length(email_lista)-1)  into email_lista;
   END$$
DELIMITER ;
```

```
create schema empleados;

use empleados;

create table empleados( id int auto_increment primary key, nombre varchar(25), email varchar(50));



insert into empleados values('0','empleado1', 'email1'); insert into empleados values('0','empleado2', 'email2');insert into empleados values('0','empleado3', 'email3');

insert into empleados values('0','empleado7', 'email7');insert into empleados values('0','empleado8', 'email8');insert into empleados values ('0','empleado9', 'email9');

insert into empleados values('0','empleado4', 'email4');insert into empleados values('0','empleado5', 'email5');insert into empleados values('0','empleado6', 'email6');

select * from empleados;

DELIMITER $$

drop procedure if exists construir_email_lista$$

CREATE PROCEDURE construir_email_lista (  INOUT email_lista VARCHAR(1000) )

  BEGIN

    DECLARE v_finished INTEGER DEFAULT 0;

    DECLARE v_email VARCHAR(255) DEFAULT "";

   DECLARE email_cursor CURSOR FOR SELECT  email FROM empleados;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished = 1;

    OPEN email_cursor;

    obtener_email: LOOP

          FETCH email_cursor INTO v_email;

          IF v_finished = 1 THEN

             LEAVE obtener_email;

          END IF;

          /*SET email_lista = CONCAT(v_email,";",email_lista);*/

          SET email_lista = CONCAT(email_lista,";",v_email);

       END LOOP obtener_email;

    CLOSE email_cursor;

    select SUBSTRING(email_lista, 2)  into email_lista;

    select substr(email_lista, 2,length(email_lista)-1)  into email_lista;

    END$$

DELIMITER ;
```

Prueba de codigo:

```
SET @email_lista = "";
CALL construir_email_lista(@email_lista);
SELECT @email_lista;
```