# Cursores

# Índice

# 1. Cursores

Os cursores **serven para procesar fila a fila o resultado dunha consulta SELECT, dentro dos programas almacenados**. Cando a sentenza SELECT devolve unha única fila, pódense asignar os valores das columnas a variables utilizando a sentenza SELECT ... INTO, e non é necesario o uso de cursores.

Estas estruturas só deben utilizarse cando non se teña outra forma de manipular conxuntos de datos cunha soa sentenza.

Cada SXBDR ten a súa forma de declarar un cursor e incluso os seus propios tipos de cursores (Oracle PL/SQL, Transact SQL, etc).

MySQL permite utilizar  cursores dende a versión 5.0. Os cursores en MySQL son de só lectura, é dicir, pódese ler a información que conteñen, pero non se pode escribir neles.

Outra característica dos cursores en MySQL é que a información que conteñen só pode ser procesada de maneira secuencial fila a fila, empezando pola primeira fila, e non é posible acceder a unha fila directamente sen pasar polas anteriores. A cada fila pódeselle aplicar un bloque de sentenzas SQL. Para utilizar cursores en MySQL hai que seguir catro pasos:

- Declaración.
- Apertura.
- Lectura.
- Peche.

## 1.1 Sentenza DECLARE ... CURSOR

A sentenza DECLARE ... CURSOR permite declarar un cursor, asignarlle un nome, e relacionalo cunha sentenza SELECT. Sintaxe:

```
DECLARE nome_cursor CURSOR FOR sentenza_SELECT
```

O cursor debe declarase (definirse) antes de usarse. No proceso de declaración non se recuperan datos, simplemente asóciase a unha sentenza SELECT. A declaración dos cursores ten que ir sempre despois da declaración de variables e de condicións de erro, pero antes da declaración de manipuladores de erro. Nun programa almacenado pódense declarar máis dun cursor.

## 1.2 Sentenza OPEN

A sentenza **OPEN permite abrir un cursor para poder procesar a información resultante da consulta SELECT**. Sintaxe:

```
OPEN nome_cursor
```

A operación de apertura ten que facerse despois de declarar o cursor e antes de facer ningunha operación con el. Esta operación recupera os datos que resultan de executar a sentenza SELECT que ten asociada.

Cando se abre o cursor, créase un punteiro que sinala á primeira fila de resultados. Este punteiro é utilizado polo sistema para sinalar a fila que se vai a ler cando se execute unha operación de lectura.

## 1.3 Sentenza FETCH

A sentenza **FETCH permite ler as filas obtidas pola consulta SELECT** asociada a un cursor. Cada vez que se executa esta sentenza, móvense os valores das columnas correspondente a unha fila do conxunto de resultados, a un conxunto de variables e avanza o punteiro á fila seguinte. Sintaxe:

```
FETCH nome_cursor INTO nome_variable [, nome_variable ] …
```

Despois da palabra INTO teñen que especificarse tantas variables como columnas devolve a sentenza SELECT e teñen que estar definidas previamente co mesmo tipo de dato que a columna da que vai recibir o valor.

A lectura de resultados é sempre secuencial. A primeira vez que se executa unha sentenza FETCH, despois de abrir o cursor, accédese á primeira fila obtida pola consulta, a segunda vez que se executa accédese á segunda fila, e así sucesivamente.

Para poder recorrer todas as filas que devolve a sentenza SELECT é necesario executar un número variable de veces a sentenza FETCH, en función do número de filas que se devolvan. Para facer isto, o lóxico é utilizar unha estrutura repetitiva (WHILE ou REPEAT) e incluír a sentenza FETCH dentro do bucle, que debe finalizar cando xa non queden filas do resultado por ler.

Cando xa non quedan filas por ler no cursor, e se executa unha sentenza FETCH, prodúcese un erro asociado ao código SQLSTATE '02000', ou ben á condición de erro 'NOT FOUND', polo que para establecer a condición para que finalice a execución do bucle hai que definir un manipulador asociado a ese código de erro.

Exemplo de manipulador para controlar o final do bucle:

```
declare continue handler for SQLSTATE '02000' set vFinal = 1;
```

Esta sentenza de declaración de manipulador indica que se ocorre o erro con código SQLSTATE '02000' non finalice a execución do programa almacenado (por ser tipo 'continue'), e que se lle asigne o valor 1 á variable *vFinal*, que ten que estar previamente declarada. Esta variable pode ser utilizada para establecer a condición para finalizar o bucle, é dicir, o bucle debe finalizar cando a variable tome o valor 1.

Outra forma de declarar o manipulador que produce o mesmo resultado cá anterior é:

```
declare continue handler for NOT FOUND set vFinal = 1;
```

## 1.4 Sentenza CLOSE

A sentenza **CLOSE permite pechar un cursor e limpar os espazos de memoria que utiliza**.

```
CLOSE nome_cursor
```

O cursor péchase automaticamente no momento que finaliza o programa almacenado que o contén, aínda que non se execute a sentenza CLOSE. Mentres o programa non finalice, un cursor pode ser pechado e aberto as veces que se necesite.

## 1.5 Utilización de cursores

O proceso de utilización de cursores require facer unha serie de operacións que son obrigatorias sempre, e polo tanto pódese facer un esqueleto coas sentenzas SQL necesarias para utilizar cursores.

Cando se utiliza MySQL Workbench para editar os procedementos almacenados que utilizan cursores, pódese crear un fragmento de código (*snippet*) e gardalo para utilizalo cada vez que se escribe o código dun cursor.

Exemplo de contido do snippet esqueleto_cursor:

```
begin
  -- Declaración de variables
  declare vFinal integer default 0;
  /* aquí decláranse tamén as variables para almacenar os datos contidos
  nas columnas da consulta*/
  -- Declaración do cursor
  declare nomeCursor cursor for
    sentenza_SELECT;
  -- Declaración do manipulador para controlar o final da lectura do cursor
  declare continue handler for sqlstate '02000' set vFinal=1;
  -- Abrir cursor: recupéranse os datos da consulta
  open nomeCursor;
  -- Bucle para procesar os datos asociados ao cursor, ata que non queden filas
  /*Sábese que non quedan filas coando se produce o erro código sqlstate '02000
    nese momento asígnaselle á variable vFinal o valor 1*/
  repeat
  -- Lectura dos datos dunha fila do cursor e asignación a variables
   fetch nomeCursor into lista_variables;
   if vFinal = 0 then
          -- Aquí van as sentenzas para procesar unha fila do cursor
   end if;
  -- Condición de final da execución do bucle: cando vFinal tome o valor 1
  until vFinal = 1 end repeat;
  -- Peche do cursor, para liberar memoria
  close nomeCursor;
end
```

Exemplo de utilización de cursores:

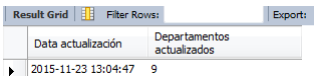
```
/*
-----------------------------------------------------------------------------
NOME RUTINA: traballadores.sp_demoCursor1 (procedemento)

TAREFA A AUTOMATIZAR:     - Utilizando un cursor, que conta os empregados agrupados por
                          departamento, actualizar a columna depEmpregados (número de
                          empregados do departamento) na táboa departamento no caso que
                          o número de empregados contados non coincida co contido actual
                          da columna depEmpregados.
                          - Contar o número de departamentos que foron actualizados.
PARAMETROS REQUERIDOS:    - Non require parámetros.
RESULTADOS PRODUCIDOS:    - Mostrar na pantalla a data na que se fai a actualización, e
                          o número de departamentos que foron actualizados.
                          - Columna depEmpregados actualizada na táboa departamento.
-----------------------------------------------------------------------------
*/
```

```
drop procedure traballadores.sp_demoCursor1;
delimiter //
create procedure traballadores.sp_demoCursor1()
begin
  -- Declaración de variables:
  -- vDepartamento e vContador almacenan os valores das columnas da consulta
  declare vDepartamento, vContador integer;
  -- vFinal utilízase para controlar o momento que non hai máis filas
  -- vFModificados é un contador dos departamentos actualizados
  declare vFinal, vModificados integer default 0;
  -- Declaración de cursores
  declare cursorEmpregados cursor for
   select empDepartamento,count(*)
          from traballadores.empregado
          group by empDepartamento;
  -- Declaración do manipulador para controlar o final da lectura do cursor
  declare continue handler for sqlstate '02000' set vFinal=1;
  -- Abrir cursor: recupéranse os datos da consulta
  open cursorEmpregados;
  -- Bucle para procesar os datos asociados ao cursor, ata que non queden filas
  repeat
    -- Lectura dos datos dunha fila do cursor e asignación a variables
   fetch cursorEmpregados into vDepartamento, vContador;
   if vFinal = 0 then
     -- Procesamento dunha fila do cursor
    if vContador <> (select depEmpregados
                         from traballadores.departamento
                         where depNumero=vDepartamento)
    then
        -- contar departamento modificado
        set vModificados = vModificados + 1;
        -- modificar os datos do departamento
        update traballadores.departamento set depEmpregados=vContador
          where depNumero=vDepartamento;
    end if;
   end if;
 -- Condición de final da execución do bucle: cando vFinal tome o valor 1
 until vFinal = 1 end repeat;
-- Peche do cursor, para liberar memoria
  close cursorEmpregados;
-- Mostra número de departamentos modificados
  select now() as 'Data actualización', vModificados as 'Departamentos actualizados';
end
//
delimiter ;
```

Este procedemento almacenado utiliza o cursor *cursorEmpregados* asociado a unha consulta sobre a táboa *empregado* que devolve o número do departamento e o contador de empregados que traballan nese departamento. Cando se le unha fila do cursor, compárase o valor da columna *depEmpregados* da táboa *departamento* co valor que devolve o contador da consulta asociada ao cursor. Se non coinciden os valores, actualízase o contido da columna *depEmpregados* co resultado do contador, e contabilízase a actualización feita empregando unha variable tipo contador. Ao finalizar o proceso, móstrase a data e hora do sistema e o número de departamentos que foron actualizados.

Pódense executar as seguintes sentenzas para facer unha proba do funcionamento do procedemento e ver o resultado na zona *Result Grid* en Workbench.

```
update traballadores.departamento set depEmpregados=0;
call sp_demoCursor1();
```




### 1.5.1 Cursores e sentenzas preparadas

A combinación do uso de cursores e de sentenzas preparadas proporciona aos administradores de bases de datos unha ferramenta moi potente para automatizar tarefas propias da administración.

Exemplo de cursor combinado con sentenzas preparadas:

```
/* ----------------------------------------------------------------------------
NOME RUTINA: utilidades.analizartaboas (procedemento)

TAREFA A AUTOMATIZAR:     - Analizar todas as táboas dunha base de datos que se pasa
                            como parámetro, utilizando a sentenza:
                            'analyze table nome_táboa'
PARAMETROS REQUERIDOS:    - IN: nome da base de datos
RESULTADOS PRODUCIDOS:    - Executar a sentenza 'analyze table nome_táboa' para cada
                            táboa.
                          - Saída por pantalla do resultado da análise.

----------------------------------------------------------------------------
*/
drop procedure if exists utilidades.analizartaboas;
delimiter //
create procedure utilidades.analizartaboas(pBaseDatos varchar(64))
begin
  -- Declaración de variables:
  -- vTaboa almacena os nomes da táboas que se obteñen na consulta
  declare vTaboa varchar(64) character set utf8;
  -- vFinal utilízase para controlar o momento que non hai máis filas
  declare vFinal integer default 0;
  -- Declaración de cursores
  declare cursorTaboas cursor for
    select table_name
    from information_schema.tables
    where table_schema=pBaseDatos;
  -- Declaración de manipuladores:
  -- Declaración do manipulador para controlar o final da lectura do cursor
  declare continue handler for sqlstate '02000' set vFinal=1;
  -- Abrir cursor: recupéranse os datos da consulta
  open cursorTaboas;
  -- Comprobar se existe a base de datos
  if (select count(*)
          from information_schema.schemata
          where schema_name = pBaseDatos) = 0
  then
    select 'A base de datos non existe';
  else
  -- No caso de que exista a base de datos:
  -- Bucle para procesar os datos asociados ao cursor, ata que non queden filas
```

```
    repeat
      -- Lectura dos datos dunha fila do cursor e asignación a variables
      fetch cursorTaboas into vTaboa;
      if vFinal = 0 then
     -- Procesamento dunha fila do cursor: analizar a táboa
            set @sentenza = concat('analyze table ',pBaseDatos,'.',vTaboa);
            prepare analizar from @sentenza;
            execute analizar;
      end if;
    -- Condición de final da execución do bucle: cando vFinal tome o valor 1
    until vFinal = 1 end repeat;
    -- borrado da sentenza preparada, para liberar memoria
    drop prepare analizar;
    -- Peche do cursor, para liberar memoria
    close cursorTaboas;
  end if;
end
//
delimiter ;
```

Este tipo de procedementos almacenados utilizan a información do dicionario de datos para poder automatizar operacións sobre obxectos das bases de datos. No caso do exemplo execútase a sentenza ANALYZE TABLE para facer un mantemento dos índices e mellorar o rendemento das consultas. A execución desta sentenza produce o bloqueo da táboa mentres se fai o análise, e mostra como saída unha mensaxe informando do estado da operación. Como a sintaxe da sentenza non admite utilizar variables para poñer o nome da táboa, hai que recorrer ás sentenzas preparadas.

Exemplo de execución do procedemento almacenado:

```
call utilidades.analizartaboas('traballadores');
```