Procedimientos Almacenados en MySQL. Cursores.

Cursores en MySQL

Introducción al cursor MySQL

Para manejar un conjunto de resultados dentro de un procedimiento almacenado, se utiliza un cursor. Un cursor le permite iterar en un conjunto de filas devueltas por una consulta y procesar cada fila individualmente.
Un cursor MySQL es de solo lectura, no desplazable y no sensible.

- Solo lectura (Read-only): no puede actualizar los datos en la tabla subyacente a través del cursor.
- No desplazable (Non-scrollable): solo puede recuperar filas en el orden determinado por la instrucción SELECT. No puede recuperar filas en el orden inverso. Además, no puede omitir filas o saltar a una fila específica en el conjunto de resultados.
- Asensible (Asensitive): existen dos tipos de cursores: cursor .ªsensiblez cursor insensible. Un cursor .ªsensible.ªpunta a los datos reales, mientras que un cursor ïnsensibleüsa una copia temporal de los datos. Un cursor .ªsensible"funciona más rápido que un cursor insensible porque no tiene que hacer una copia temporal de los datos. Sin embargo, cualquier cambio que se realice en los datos desde otras conexiones afectará los datos que está utilizando un cursor sensible, por lo tanto, es más seguro si no actualiza los datos que está utilizando un cursor sensible. El cursor MySQL es .ªsensible".

Puede usar cursores MySQL en procedimientos y funciones almacenados y disparadores. Trabajando con el cursor MySQL

- Primero, declare un cursor usando la declaración DECLARE:

```
DECLARE cursor_name CURSOR FOR SELECT_statement;
```

La declaración del cursor debe ser posterior a cualquier declaración de variable. Si declara un cursor antes de las declaraciones de variables, MySQL emitirá un error. Un cursor siempre debe asociarse con una instrucción SELECT.

- A continuación, abra el cursor utilizando la instrucción OPEN. La instrucción OPEN inicializa el conjunto de resultados para el cursor, por lo tanto, debe llamar a la instrucción OPEN antes de recuperar filas del conjunto de resultados.

```
OPEN cursor_name;
```

- Luego, use la instrucción FETCH para recuperar la siguiente fila apuntada por el cursor y mover el cursor a la siguiente fila del conjunto de resultados.

```
FETCH cursor_name INTO variables list;
```

- Después de eso, verifique si hay alguna fila disponible antes de buscarla.
- Finalmente, desactive el cursor y libere la memoria asociada con él usando la instrucción CLOSE:

```
CLOSE cursor_name;
```

Es una buena práctica cerrar siempre un cursor cuando ya no se usa.
Cuando trabaje con cursores MySQL, debe declarar un controlador (handler) NOT FOUND para manejar la situación en la que el cursor no encuentre más filas de datos.
Cada vez que llama a la instrucción FETCH, el cursor intenta leer la siguiente fila del conjunto de resultados. Cuando el cursor llega al final del conjunto de resultados, no podrá obtener los datos y se genera una condición. El controlador se utiliza para manejar esta condición.
Para declarar un controlador NOT FOUND, hacemos:

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

finished es una variable que usaremos para indicar que el cursor ha llegado al final del conjunto de resultados. Observe que la declaración del controlador debe aparecer después de la declaración de variables y cursores dentro de los procedimientos almacenados.
El siguiente diagrama ilustra cómo funciona el cursor MySQL.

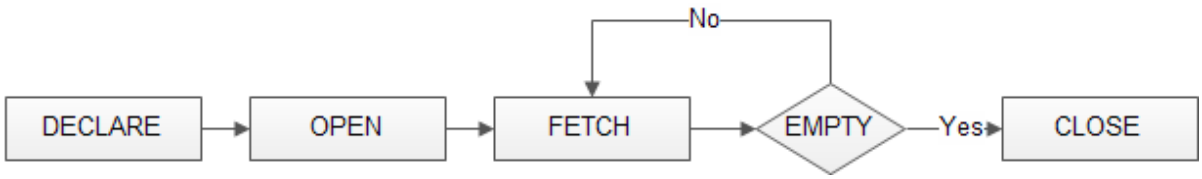

Figura 1: Diagrama de flujo para el funcionamiento de un cursor

Ejemplo de cursor de MySQL

Desarrollaremos un procedimiento almacenado que cree una lista de correo electrónico de todos los empleados en la tabla de employees en la base de datos de muestra .

- Primero, declare algunas variables, un cursor para recorrer los correos electrónicos de los empleados y un controlador NOT FOUND :

```
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;
```

- Luego, abra el cursor usando la instrucción OPEN

```
OPEN  curEmail;
```

- Posteriormente, repita la lista de correos electrónicos y concatene todos los correos electrónicos donde cada correo electrónico esté separado por un punto y coma (;):

```
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;
```

- Después, dentro del bucle, usamos la variable finished para verificar si se ha recuperado o no un correo electrónico de la lista para terminar el bucle.
- Finalmente, cierre el cursor usando la instrucción CLOSE:

```
CLOSE  email_cursor;
```

El procedimiento almacenado createEmailList es el siguiente:

```
 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  ;
```

Puede probar el procedimiento almacenado createEmailList utilizando el siguiente script:

```
SET  @emailList =  "" ;
CALL  createEmailList(@emailList);
SELECT  @emailList;
```