Cursores

Introducción a cursores

Definición

Una base de datos relacional como MySQL está orientada a conjuntos de manera natural por ejemplo una instrucción SELECT regresa un conjunto de datos sin embargo muchas veces es necesario utilizar no el enfoque de conjunto de datos sino de registros para realizar ciertas operaciones no complejas es donde se implementan los cursores. Un cursor es un objeto de la Base de Datos usado por las aplicaciones para manipular los datos fila a fila en lugar de hacerlo en bloques de filas como lo hacen los comandos SQL normales.

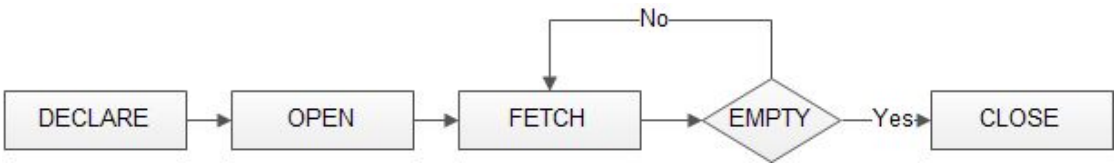
Un cursor puede ser usado internamente en la base de datos para realizar alguna operación registro a registro, dentro de un disparador o de un Procedimiento Almacenado, como así también ser utilizado en una aplicación externa, como por ejemplo una conexión JDBC o ADO.Net. Cada vez que se realiza una iteración sobre un ResultSet se está generando un cursor.

El punto de entrada de un cursor será siempre una instrucción SQL. A partir de los resultados que devuelve una instrucción SQL se podrá iterar sobre ella.

Ciclo de vida

Para poder trabajar con Cursores debemos realizar los siguientes pasos , :

- Declarar el Cursor con la instrucción DECLARE .
- Abrir el Cursor con la la instrucción OPEN
- Declarar una variable para manejar
- Recuperar las filas desde el Cursor con la instrucción FETCH .
- Procesar las filas obtenidas.
- Cerrar el Cursor con la instrucción CLOSE .



Modelo de cursores

En base a lo descrito con anterioridad, vamos a analizar parte por parte del ciclo de vida del cursor como debe implementarse en MySQL.

1) Declaración

```
DECLARE <nombre cursor> CURSOR
FOR
<sentencia sql>;
```

En esta etapa estamos declarando dentro de un procedimiento (que puede ser almacenado o no) una variable de tipo CURSOR para luego ser utilizada con posterioridad. Supongamos que queremos declarar un cursor para iterar sobre los resultados de una lista de nombres, apellidos y números de documento cuya nacionalidad sea Argentina :

```
DECLARE CUR_PERSONAS CURSOR FOR SELECT
ID_PERSONA,NOMBRE,APELLIDO,DNI FROM PERSONAS; WHERE
NACIONALIDAD = 'ARGENTINA';
```

2) Declarar un NOT_FOUND handler

Para poder verificar que el cursor ha traído datos, debemos declarar una variable que sea modificada con TRUE|FALSE según el resultado de la última operación.

Para esto debemos declarar una variable booleana y luego asignarle a esa variable booleana la tarea de cambiar su valor en base a la última operación.

```
DECLARE vFinished INTEGER DEFAULT 0;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET vFinished = 1;
```

Cuando una operación FETCH no encuentre datos, la variable vFinished cambiará a 1, por lo cual es sumamente importante inicializar esta en 0.

3) Apertura

Cuando declaramos el valor solamente estamos indicando que existe una variable de tipo cursor pero no la estamos aun utilizando de ninguna manera. Para poder comenzar a utilizarla debemos abrir el cursor con la siguiente sintaxis :

OPEN <nombre cursor>;

En nuestro caso de ejemplo sería :

OPEN CUR_PERSONAS;

4) Recuperación de datos

Este paso constará de recorrer los resultados del cursor, la instrucción FETCH permitirá efectuar dicha operación. Las filas leídas podrán copiarse a variables utilizando la sentencia INTO en combinación con la sentencia FETCH, por ejemplo la sentencia:

```
FETCH CUR_PERSONAS INTO
vId,vNombre,vApellido,vDni;
```

Tomará la siguiente fila de resultados del cursor y lo alojará en las variable vId vNombre vApellido y vDni.
Un detalle a comentar es que en la sentencia **INTO** (como puede verse en el ejemplo anterior) el mapeo entre columnas del cursor y variables se realizará implícitamente, asignándose la primera columna a la primera variable, la segunda columna a la segunda variable y así sucesivamente. Esto implica que deberán crearse tantas variables como columnas se definan en la declaración del cursor y las mismas deberán ubicarse en el mismo orden que se encuentran definidas las columnas en la sentencia **SELECT** de la declaración.

Como cada sentencia **FETCH** leerá un registro, una pregunta interesante que podríamos hacernos es, ¿de qué manera podremos saber si existe un próximo o previo registro, o si hemos llegado al límite (ya sea superior o inferior)?. La respuesta a esto es que el valor de la variable FINISHED que declaramos anteriormente cambiara a 1 y podremos tomar ese valor para chequear esto.

De esta manera, para poder recorrer en forma completa el contenido de nuestro cursor CUR_PERSONAS la instrucción seria la siguiente :

```
get_personas :  LOOP

        FETCH CUR_PERSONAS INTO   vId,vNombre,vApellido,vDni;
        IF vFinished = 1 THEN
                LEAVE get_personas;
        END IF;
        SELECT CONCAT(vId,vNombre);
END LOOP get_personas;
```

5) Cierre de cursor

En el cierre del cursor se liberarán los registros tomados por el mismo. Una vez que el cursor es cerrado ya no podrá recorrerse el conjunto de resultados hasta que el mismo sea reabierto, la sentencia CLOSE cerrará un cursor abierto y la sintaxis puede verse a continuación

```
CLOSE  <nombre cursor>;
```