Cursores

Sitio: Agencia de Aprendizaje a lo largo de la Vida
Curso: Administración de Base de Datos 1° G
Libro: Cursores

Imprimido por: Eduardo Moreno
Día: martes, 22 de octubre de 2024, 19:20

Tabla de contenidos

Cursores

¿Qué son y para qué sirven?

Cuando trabajamos dentro de los procedimientos podemos construir consultas que arrojen varias filas y columnas y tenemos que acceder a determinados datos. La consulta deja de ser estática para empezar a recorrerla fila a fila.

Existe un elemento llamado cursor que nos permite almacenar el resultado de la consulta para examinarla.

Los cursores tienen las siguientes características:

- *"Asensitive"*: el servidor puede o no hacer una copia de su tabla resultado.
- *"Read-only"*: no puede modificarse el resultado
- *"Nonscrollable"*: se accede a las filas de a una y solo hacia adelante

Podemos pensar al cursor como una caja con tapa que contiene un número determinado de fichas. Si queremos manipular lo que contiene lo primero que hacemos es declarar que es una caja y determinamos el contenido. Luego abrimos la caja y leemos la primera ficha. Para acceder a la segunda, descartamos la primera, así sucesivamente hasta llegar a la última ficha y como vemos el fondo ya sabemos que no hay más y cerramos la caja.

Este mecanismo es lo que hacemos con el cursor, las palabras resaltadas en el párrafo anterior se traducen luego a acciones escritas en lenguaje SQL.

En los próximos capítulos veremos como manipulamos a esta estructura.

Cómo trabajar con un cursor

1. Tomemos cada una de las acciones mencionadas en los siguientes ejemplos.

“declarar que es una caja” .

2. El cursor se declara de la siguiente manera. No olvidemos que el cursor almacena una sentencia *select.*

```
declare nombre_cursor cursor for  << Sentencia select >> ;
```

“abrimos la caja”

3. La instrucción *open* ejecuta la sentencia *select* asociada al cursor y lo carga con el resultado.

```
open nombre_cursor;
```

Lectura del cursor - “leemos la primera ficha”

1. Para la lectura usamos la instrucción *fetch.* La consulta que tiene el cursor proyecta una o varias columnas y la lectura la realiza por filas, por eso, el *fetch* lee las columnas de la fila y debe almacenarlas en variables para que después las usemos en el procedimiento.

2. Si el *select* es *select codsocio*, nombre, apellido from socio la lectura cuenta con 3 variables, una para cada columna. Obviamente estas variables están declaradas en el procedimiento y deben ser del mismo tipo de datos que el atributo proyectado.

```
fetch nombre_cursor into  Variable1, Variable2, ......  ;
```

Veamos un ejemplo de código

1. Las declaraciones de las variables que se usan en el procedimiento deben ir antes de la declaración del cursor y del *handler.*

2. La consigna es simplemente declarar un cursor cuyo *select* proyecta dos atributos de la tabla del *from.*

3. Se deben declarar dos variables para almacenar el *id* que es entero y el dato que es un *varchar*, estas declaraciones están antes de la declaración del cursor.

4. Se declara la variable *var* para usarla en el *handler* por defecto decimos que es falso, de esa manera cuando no haya más filas que leer o la consulta select sea *empty* se cambia el valor a verdadero.

5. Para ingresar al ciclo *while* se debe leer antes, si *var = false* ingresa al ciclo realiza lo que solicita la consigna, es decir, lo que nosotros programamos y vuelve a leer.

¿Por qué? Porque si no vuelve a leer quedamos encerrados en ese ciclo y nunca llegamos al final del procedimiento.

```
delimiter //
CREATE PROCEDURE EjemploModelo()
Begin
    /* var es la variable para el HANDLER */
    Declare var boolean Default False;

    Declare a Int;
    Declare b Varchar(16);
    /* Declaración del CURSOR
                    ***********   */
    Declare cur1 CURSOR FOR SELECT id, dato FROM tabla;
    /* Declaración del Handler
                     *********    */
    Declare Continue HANDLER For Not Found Set var = True;

    Open cur1;
    Fetch cur1 Into a, b;

    While var = False do

        /* Realiza el proceso que solicita la consigna */

        Fetch cur1 into a, b; /* vueñve a leer para poder salir del ciclo */

    End While;

    Close cur1;
End//
```

Fin de lectura y cierre - “vemos el fondo”

En el ejemplo a medida que quitamos fichas de la caja nuestros ojos pueden detectar si nos quedan o no, pero, ¿Cómo lo maneja el sistema gestor? El no mira si el cursor tiene filas aún por tratar o no.

Cuando se acaban las filas ocurre la condición de *No Data* ¿Condición?

Se llaman “condiciones” a situaciones que ocurren durante la ejecución de una rutina. Estas situaciones pueden tener diferentes causas. Por ejemplo, el fin de datos del cursor.

Para poder tratar las condiciones se define un elemento llamado “ *Handler*”. El *handler* se invocará cuando ocurra la condición y debe estar declarado en el procedimiento después de la declaración del cursor.

```
declare handler_action HANDLER for condition_value;
```

Puede ser CONTINUE o EXIT

Tiene diferentes valores

- Para un *handler continue*, continúa la rutina actual tras la ejecución del comando del *handler.*
- Para un *handler exit* , termina la ejecución del comando compuesto *Begin … End* actual.
- La *condition_value* que nos interesa es *no found*.

En la creación de los procedimientos con uso de cursores se emplea el *handler continue*.

¿Cómo saber si ocurre la condición?

En la declaración del *handler* le asignamos a una variable previamente declarada un valor, entonces, dentro del código con un *ciclo while o ciclo loop* evaluamos a la variable y si tiene ese último valor asignado quiere decir que se produjo la condición declarada. Esta variable tiene el tipo de dato booleano la declaramos por defecto en falso y cuando se produce la condición pasamos su valor al otro estado posible que es el verdadero.

Por último, nos queda “cerramos la caja”

Al terminar un cursor debemos cerrarlo.

```
close nombre_cursor;
```

Luego, liberá el *result-set* sobre las filas en las que está posicionado el cursor.

Por último, dejá las estructuras de datos internas listas para usarse de nuevo, pero no se pueden hacer *fetch* hasta que se abra nuevamente el cursor.