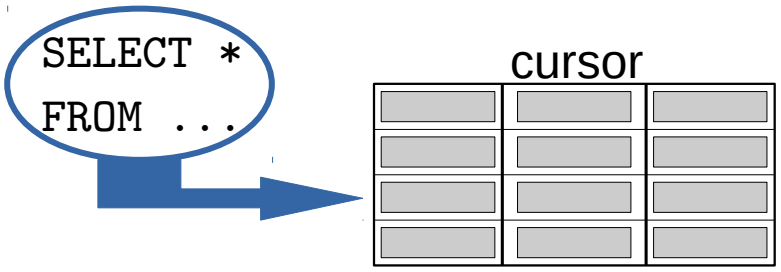
Cursores

Un cursor es una estructura que existe en la base de datos para recoger el resultado de una consulta. Estamos habituados a que el resultado de una consulta se muestre por pantalla, pero el cursor (al igual que un ResultSet en Java) almacena este resultado para su posterior procesamiento.

La siguiente figura esquematiza como se almacena el resultado de una consulta en un cursor:



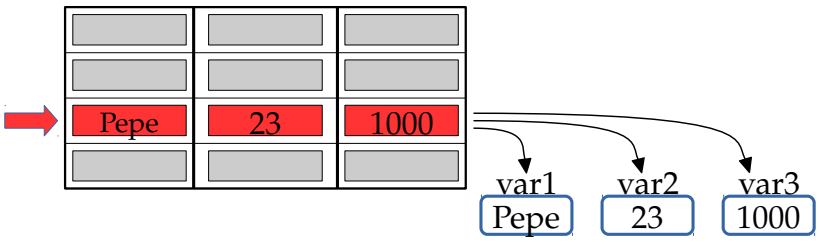
Los cursores se utilizan habitualmente en un procedimiento almacenado. La forma de declarar un cursor es:

```
DECLARE nombreDelCursor CURSOR FOR
SELECT ...
```

Veamos un ejemplo: supongamos que me interesa guardar en un cursor el numemp, nombre y edad de lo empleados que trabajan en la oficina 12. Para ello tendremos vamos a declarar un cursos al que llamaremos empeados12, de la forma:

```
DECLARE empleados12 CURSOR FOR
SELECT numemp, nombre, edad
FROM Empleados
WHERE oficina = 12;
```

¿Cómo se usa un cursor? El cursor funciona de manera análoga a un ResultSet. En cada momento solo podremos trabajar con uno de los registros que almacena el cursor. Pasaremos los datos del registro activo a una serie de variables. Estas variables tienen que ser de un tipo acorde a los campos del registro.



Esta figura nos servirá para explicar detalladamente el uso de un cursor. En un momento dato, los valores del registro activo (que solo puede ser uno), se pasan a las variables var1, var2 y var3. El primer campo del registro es un VARCHAR, por lo tanto var1 tendrá que ser de tipo VARCHAR. El según y tercer campo del registro son de tipo INT, por lo tanto, var2 y var3 tendrán que ser de tipo INT.

Una vez que hemos procesado los valores de las variables, pasaremos al siguiente registro. Y de nuevo, pasaremos los valores de los campos del siguiente registro activo, a las variables para su procesamiento.
Este comportamiento hace que los cursores tengan una serie de restricciones:

- Los cursores son solo de lectura. La información siempre pasa de los registros a las variables.
- No se puede hacer scrolling. Es decir, cuando hemos acabado con un registro, pasaremos al registro siguiente. Así continuamente, hasta que hayamos procesado todos los registros del cursor. En ningún momento podremos volver atrás, a un registro ya procesado.

La instrucción para pasar los valores de un registro a variables es:

```
FETCH nombreCursor INTO variable1, variable2, ...
```

El trabajo que realiza FETCH es doble:

1. En primer lugar se sitúa en el siguiente registro. Si es la primera vez que se utiliza se coloca en el primer registro del cursor.
2. Lee los valores del registro actual y los asigna a las variables.

Para que FETCH funciona correctamente, necesitaremos utilizar tantas variables como campos tengan los registros del cursor.

¿Cuándo acabamos de procesar un cursor? Los cursores se van procesado registro a registro, hasta que no existen más registros. O dicho de otra forma, hasta que llegamos al último.
En el momento que estamos situados justo en el último registro y ejecutamos una instrucción FETCH, esta hará que salte una especie de "excepción". Las excepciones de MySQL están muy lejos de la complejidad y la sofisticación de las de Java. Una excepción de MySQL tan solo cambia el valor de una variable. El valor de esta variable puede pasar, por ejemplo, de 0 a 1. Y es lo que nos permitirá salir del bucle donde se procesan los registros del cursor.

Cuando no existen más registros que procesar en un cursor, se activa la "excepción" (o error) 02000 (cero dos mil). La forma de indicar que se modifique una variable cuando se produce un error es:

```
DECLARE CONTINUE HANDLER FOR SQLSTATE 'númeroDelError'
SET variable = valor;
```

En nuestro caso usaremos la variable "done", para indicar que el cursor se ha procesado entero (algo así como: "¡Hecho!, ¡Listo!"

La variable "done" deberá estar declarada previamente:

```
DECLARE done INT DEFAULT 0;
```

y el manejador de errores queda:

```
DECLARE CONTINUE HANDLER FOR SQLSTATE '02000'
SET done = 1;
```

En resumen:

1. Vamos a procesar tranquilamente el cursor registro a registro.
2. El procesamiento del cursor estará, evidentemente, dentro de un bucle.
3. El bucle estará iterando mientras done sea 0.

4. En algún momento, saltará el error (o excepción) 02000, que significa que hemos terminado con el cursor. Lo que hace que "done" tome el valor 1.
5. Cuando "done" es 1, salimos del bucle.

El algoritmo anterior se puede describir mediante el código:

```
REPEAT
      FETCH nombreCursos INTO variable1, variable2, ...
      . . . //procesamos las variables
UNTIL done
END REPEAT;
```

Recordemos que un bucle REPEAT-UNTIL itera mientras la condición (en este caso "done") sea falsa (done es 0).

Al igual que un fichero, que una conexión de base de datos o que un ResultSet; es necesario abrir un cursor siempre antes de su uso. Y es necesario cerrarlo después de haber terminado con él.
Las instrucciones que permiten esto son:

```
OPEN nombreCursor;
CLOSE nombreCursor;
```

La instrucción OPEN ejecuta la consulta asociada al cursor, carga el resultado de la consulta en el cursor y sitúa al primer registro como registro activo.
La instrucción CLOSE, cierra el cursor, lo que hace que sea imposible volver a utilizarlo. Si deseamos volver a usarlo, tendremos que realizar una nueva apertura.

Otra cosa a tener en cuenta con los cursores, es que son bastante quisquillosos, a la hora del orden de las declaraciones. Que serán siempre:

1. Primero las variables necesarias.
2. A continuación se declaran los cursores.
3. Y por último, se declaran los manejadores (HANDLER).

Teniendo en cuenta las distintas restricciones a la hora de declarar, veamos un plantilla que se puede utilizar para facilitar el uso de cursores:

```
-- declaramos las variables
DECLARE variables ...
DECLARE done INT DEFAULT 0;

-- declaramos el cursor
DECLARE nombreCursor CURSOR FOR
SELECT ... ;

-- declaramos el manejador de errores
DECLARE CONTINUE HANDLER FOR SQLSTATE '02000'
      SET done = 1;

--  abrimos el cursor para su uso
OPEN nombreCursor;
```

```
-- bucle para procesar los registros
REPEAT
      FETCH nombreCursor INTO variable...;
            IF not done THEN   -- el IF es fundamental para no procesar
                               -- nada cuando estamos detrás del último
                               -- registro (ya se ha activado done=1)
                  -- procesamiento de las variables
                  ...
            END IF;
UNTIL done
END REPEAT;
-- cerramos el cursor
CLOSE nombreCursor;
```