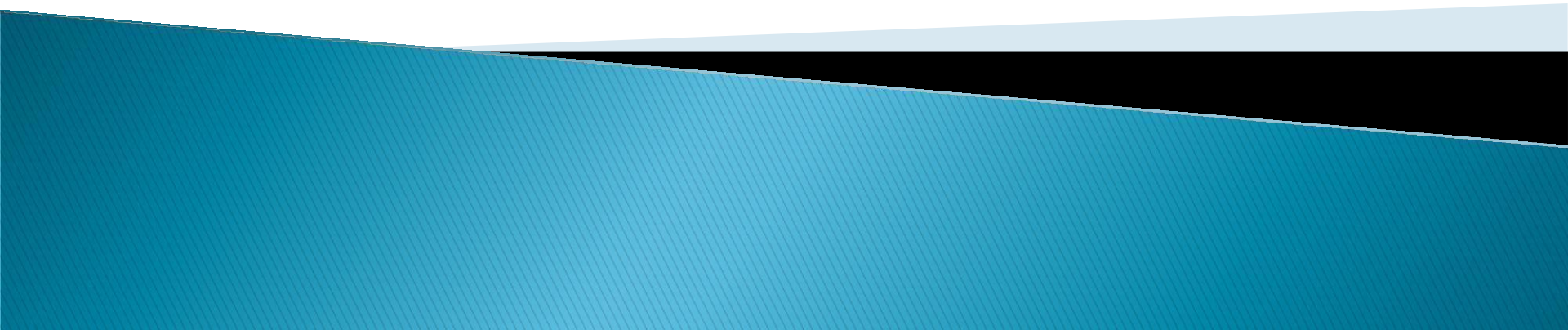


Bases de datos

Capítulo 6

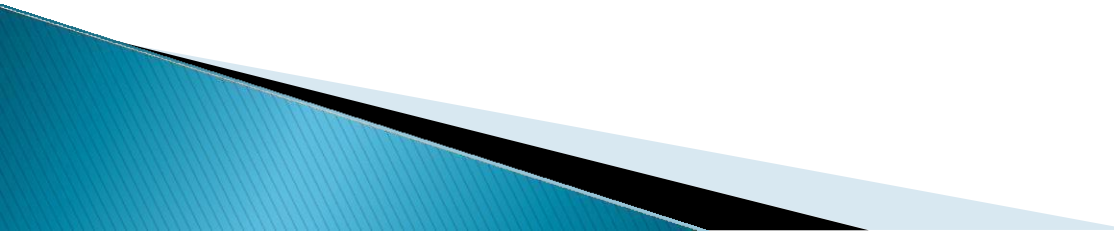
Cursores y Excepciones



Uso de cursores

Un cursor es una estructura de datos que se utiliza para recorrer *fila a fila* el conjunto de resultados de una consulta (SELECT).

Características de los cursores:

- **Son de sólo lectura:** se utilizan para leer los datos que provienen de una consulta SELECT pero no permiten modificar los datos de la tabla.
 - **De acceso secuencial:** recorren los resultados desde la primera fila a la última. No se puede saltar a una fila directamente.
 - Pueden usarse dentro de un **procedimiento, función o trigger**.
- 

Uso de cursores

El manejo de un cursor implica estas operaciones:

- DECLARACIÓN DEL CURSOR: Definir el nombre del cursor y la consulta `SELECT` que va a devolver el conjunto de resultados.
- APERTURA DEL CURSOR: Para poder procesar el conjunto de resultados de la sentencia `SELECT`.
- LECTURA DEL CURSOR: Se debe guardar la información de una fila en variables locales y pasar a la siguiente fila del conjunto de resultados. La lectura va en un **bucle** que se ejecuta hasta que se terminan de leer todas las filas.
- CIERRE DEL CURSOR: Liberar de la memoria el cursor.

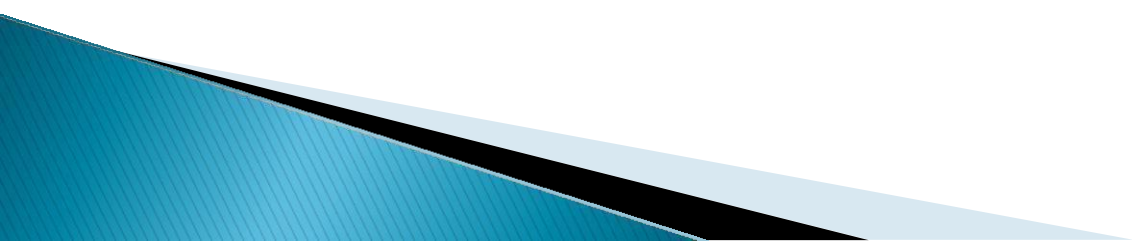
Declaración del cursor

DECLARE nombre_cursor **CURSOR FOR** sentencia_SELECT

El comando SELECT no puede tener una cláusula INTO .

IMPORTANTE

Los cursores deben declararse antes de declarar los handlers, y las variables deben declararse antes de declarar cursores o handlers.



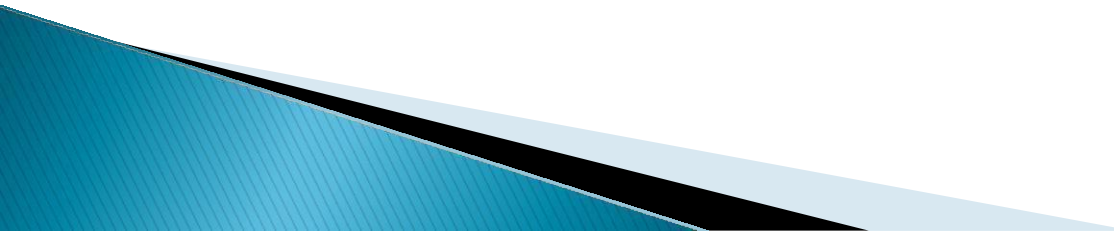
- Apertura de un cursor

OPEN nombre_cursor

- Cierre de un cursor

CLOSE nombre_cursor

Es posible trabajar con más de un cursor en el mismo procedimiento. Por eso indicamos su nombre al abrirlo o cerrarlo.




Sentencia de cursor FETCH

FETCH nombre_cursor **INTO** var1 [, var2] ...

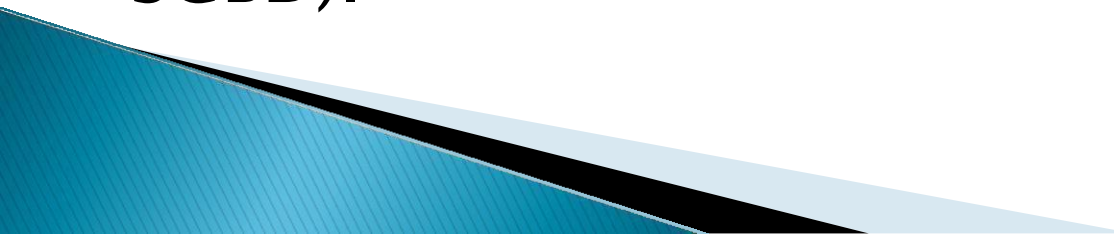
Este comando trata el siguiente registro (si existe), usando el cursor abierto que se especifique, y avanza el puntero del cursor.

Asigna a determinadas variables el valor de los campos del registro donde se encuentre el cursor en ese momento para así poder trabajar con esos valores. Habrá tantas variables como columnas tengamos en la sentencia SELECT del cursor.



Los Handlers (manejadores)

Cuando se produce un error en MySQL el servidor devuelve la descripción y dos códigos diferentes para el error:

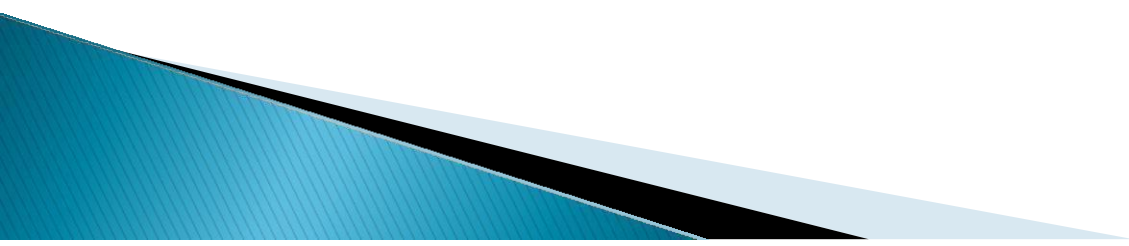
- 1.– **MySQL error**: un código de error (numérico) que es exclusivo de ese sistema gestor de base de datos.
 - 2.– **SQLSTATE**: una cadena de cinco caracteres que está estandarizada(independiente del SGBD).
- 

Los handlers permiten controlar determinados errores debido a excepciones dentro de un procedimiento.

DECLARE {CONTINUE| EXIT} **HANDLER FOR**
condición[,...] sentencia

Si una de estas condiciones ocurre la sentencia especificada se ejecuta.

- **CONTINUE** :continúa la ejecución del programa.
- **EXIT**: termina la ejecución del procedimiento.



Condiciones predefinidas MySQL

- **SQLWARNING:** representa todos los códigos SQLSTATE que empiezan por '01'.
- **NOT FOUND:** representa todos los códigos SQLSTATE que comienzan por '02'.
- **SQLEXCEPTION:** representa todos los códigos SQLSTATE que no comienzan por '00', '01' o '02'.
(equivaldría a la parte ELSE de una sentencia CASE)

Ejemplos:

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

Esta sentencia pone la variable *terminado* a 1 cuando salta la excepción '02000' (indica que no hay más filas disponibles en el cursor)

```
DECLARE EXIT HANDLER FOR SQLEXCEPTION  
BEGIN  
    ROLLBACK;  
    SELECT 'Ocurrió un error. Procedimiento  
terminado';  
END;
```

10

Declarar variables

DECLARE terminado INT DEFAULT 0;

DECLARE v_usuario VARCHAR(30);

DECLARE v_email VARCHAR(30);

2

Declarar cursos

DECLARE cursor CURSOR FOR SELECT usuario,email
FROM usuarios;

3

Declarar Handler

DECLARE CONTINUE HANDLER FOR SQLSTATE '02000'
SET terminado = 1;

OPEN cursor;  Abrir Cursor

FETCH cursor INTO v_usuario,v_email;

WHILE terminado = 0 DO

INSERT INTO copia_usuarios (usuario,email)

VALUES (v_usuario,v_email);

FETCH cursor INTO v_usuario,v_email;

END WHILE;

CLOSE cursor;

```
CREATE TABLE ActualizacionLimiteCredito(  
Fecha DATETIME, CodigoCliente INTEGER, Incremento  
NUMERIC(15,2)  
) engine=innodb;
```

delimiter //

```
CREATE PROCEDURE IncrementaLimCredito (IN  
porcentaje INTEGER)  
BEGIN
```

```
    DECLARE TotalPedidos,Credito,Incremento  
    NUMERIC(15,2);
```

```
    DECLARE Cliente, Terminado INTEGER DEFAULT 0;
```



#cursor para recorrer clientes

```
DECLARE curClientes CURSOR FOR  
SELECT LimiteCredito,CodigoCliente FROM Clientes;
```

#Al terminar de recorrerse, se activará la variable terminado

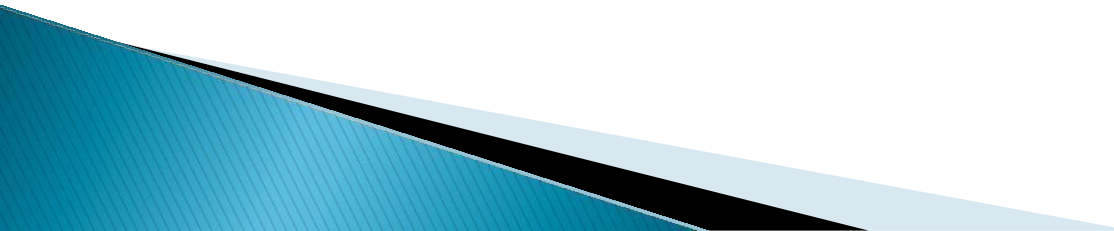
```
DECLARE CONTINUE HANDLER FOR SQLSTATE '02000'  
SET Terminado = 1;
```

#Si ocurre alguna excepción se producirá un rollback

```
DECLARE EXIT HANDLER FOR NOT FOUND rollback;  
DECLARE EXIT HANDLER FOR SQLEXCEPTION rollback;
```



```
OPEN curClientes;  
START TRANSACTION;  
FETCH curClientes INTO Credito,Cliente; #primer  
cliente  
WHILE NOT Terminado DO  
    SELECT SUM(Cantidad*PrecioUnidad) INTO  
    TotalPedidos  
    FROM DetallePedidos  
    NATURAL JOIN Pedidos WHERE YEAR(FechaPedido)  
    BETWEEN 2008 AND 2010 AND  
    Pedidos.CodigoCliente=Cliente;
```



IF TotalPedidos IS NOT NULL THEN #Si hay
pedidos

SET Incremento=TotalPedidos*Porcentaje/100;

UPDATE Clientes SET

LimiteCredito=LimiteCredito+Incremento

WHERECodigoCliente=Cliente;

INSERT INTO ActualizacionLimiteCredito

VALUES(now(),Cliente,Incremento);

END IF;

FETCH curClientes INTO Credito,Cliente;

#siguiente cliente

END WHILE;

COMMIT;

END//

delimiter;

