TEMA 12: EXCEPCIONES Y CURSORES.

1. Excepciones.

Se considera una excepción cualquier error que pueda ocurrir a lo largo de la ejecución de un programa. Si una excepción no es tratada, provocará la terminación anormal del programa en el que se produzca. Sin embargo, si lo que se desea es que se informe del error o excepción al usuario y que se pueda continuar con la ejecución del programa, entonces será necesario tratar la excepción correspondiente mediante un manejador de errores o *handler*.

Veamos un ejemplo al respecto. Creemos un procedimiento que se encargue de añadir un nuevo pedido a la tabla *Pedido* de nuestra base de datos *Pedidos*. Este procedimiento recibirá como parámetros la referencia del pedido que se desea añadir y su fecha:

```
delimiter //
create procedure InsertarPedido (refe char(5), fecha date)
begin
    insert into Pedido values (refe, fecha);
end;//
```

Ejecutemos a continuación este procedimiento dos veces:

```
mysql> call InsertarPedido ('P0005', '2019-11-09');//
Query OK, 1 row affected (0.05 sec)

mysql> call InsertarPedido ('P0005', '2019-11-09');//
ERROR 1062 (23000): Duplicate entry 'P0005' for key 'PRIMARY'
```

La primera vez que ejecutamos el procedimiento funciona correctamente, por lo que nos añade el pedido con referencia P0005 a la tabla *Pedido*. En la segunda llamada al procedimiento, como intentamos añadir de nuevo un pedido con la misma referencia, se produce un error o excepción y el programa finaliza anormalmente mostrándonos una descripción del error en pantalla. Si queremos que cuando se produzca la excepción, en lugar de terminar anormalmente el programa y mostrarnos una descripción del error predeterminada en inglés, el programa termine normalmente y muestre un mensaje confeccionado por nosotros, entonces deberemos tratar la excepción creando un manejador o *handler* para ella.

Podemos crear un manejador para la excepción 1062, que es la que se produce cuando se incumple una restricción de clave primaria o de unicidad. Vamos a borrar el procedimiento que acabamos de crear y crear uno con el mismo nombre que en caso de que se produzca la excepción 1062 muestre un mensaje de aviso al usuario indicando que ya existe un pedido con la referencia pasada como parámetro. En este procedimiento crearemos un manejador o *handler* para la excepción 1062, de manera que si se produce dicha excepción, se asigne a la

variable *duplicado* el valor 1. Esta es una variable booleana que debemos declarar anteriormente con valor inicial 0, ya que suponemos optimistamente que no se va a producir un duplicado. En caso de que se produzca dicha excepción, se asignará a la variable *duplicado* el valor 1. Por ello, en este caso, mostramos un mensaje de error al usuario. En caso contrario, indicaremos que el pedido ha sido añadido a la base de datos.

```
mysql> drop procedure InsertarPedido;//
Query OK, 0 rows affected (0.00 sec)

create procedure InsertarPedido (refe char(5), fecha date)
begin
declare duplicado bool default 0;
declare continue handler for 1062 set duplicado = 1;
insert into Pedido values (refe, fecha);
if duplicado = 1 then
    select concat ('Ya existe un pedido con la referencia ',refe) Error;
else
    select concat ('Añadido pedido con referencia ',refe) Mensaje;
end if;
end//
```

Ahora, si llamamos a este procedimiento solicitando la inserción de un pedido con una referencia no repetida, no habrá ningún problema y no aparecerá ningún mensaje de error. En caso contrario, se mostrará el mensaje de error diseñado por nosotros. Probémoslo a continuación:

```
mysql> call InsertarPedido ('P0006', '2019-11-11');//
+--------------------------------------+
| Mensaje                              |
+--------------------------------------+
| Añadido pedido con referencia P0006  |
+--------------------------------------+
1 row in set (0.04 sec)

Query OK, 0 rows affected (0.05 sec)

mysql> call InsertarPedido ('P0006', '2019-11-14');//
+-------------------------------------------+
| Error                                     |
+-------------------------------------------+
| Ya existe un pedido con la referencia P0006 |
+-------------------------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
```

La instrucción empleada para crear un manejador o *handler* de excepciones presenta el siguiente formato:

```
DECLARE {CONTINUE|EXIT} HANDLER FOR
Condición acciones_manejador;

Condición:
{SQLSTATE 'valor_estado_SQL' | código_error_MySQL | NOT FOUND}
```

Como se puede observar, después de la palabra DECLARE se debe indicar si se trata de un manejador de tipo CONTINUE o EXIT:

- Con un manejador del tipo CONTINUE, tras el levantamiento de la excepción, la ejecución del programa continúa en la siguiente instrucción.
- Con un manejador del tipo EXIT, después de producirse la excepción, la ejecución del bloque en el que se ha producido la excepción finaliza, pasando la ejecución al bloque externo dentro del mismo programa, o bien, si es el bloque principal, se devuelve la ejecución al programa externo que invocó al procedimiento o función.

Se debe indicar después HANDLER FOR y a continuación la condición por la que se produce la excepción, que puede tomar varios formatos, entre ellos:

- SQLSTATE 'valor_estado_SQL': Es un código alfanumérico de 5 caracteres que lleva asociado cada posible error en MySQL.
- código_error_MySQL: Es un código numérico de 4 cifras que lleva asociado cada posible error en MySQL.
- NOT FOUND: Hace referencia a estados SQL que comienzan por '02'.

Se muestran en la siguiente tabla algunas de las excepciones que se pueden producir en MySQL, indicando por cada una de ellas su código de error, su valor de estado y una descripción. Están resaltadas en amarillo las excepciones más relevantes.

Código de error	Valor de estado	Descripción
1005	HY000	No se puede crear la tabla indicada
1006	HY000	No se puede crear la base de datos indicada
1007	HY000	No se puede crear la base de datos indicada. La base de datos ya existe.
1008	HY000	No se puede borrar la base de datos indicada. La base de datos no existe.
1009	HY000	Error eliminando base de datos.
1040	08004	Demasiadas conexiones
1044	42000	Acceso denegado para el usuario indicado sobre la base de datos indicada.
1045	28000	Acceso denegado para el usuario indicado con la contraseña indicada.
1046	3D000	No seleccionada base de datos.
1047	08S01	Comando desconocido
1048	23000	La columna indicada no puede tomar valor nulo
1049	42000	Base de datos desconocida
1050	42S01	La tabla indicada ya existe
1051	42S02	Tabla desconocida
1052	23000	Columna ambigua

1054	42S22	Columna desconocida en la tabla indicada
1055	42000	La columna indicada no está en GROUP BY
1056	42000	No se puede agrupar sobre el campo indicado
1057	42000	La misma sentencia contiene funciones de resumen y atributos
1062	23000	Entrada duplicada para el atributo clave indicado
1068	42000	Definida clave primaria duplicada
1090	42000	No se pueden eliminar todas las columnas con ALTER TABLE; use DROP TABLE
1102	42000	Nombre de base datos incorrecta
1103	42000	Nombre de tabla incorrecta
1106	42000	Nombre de procedimiento desconocido
1107	42000	Incorrecto número de parámetros para el procedimiento indicado.
1108	42000	Parámetros incorrectos para el procedimiento indicado.
1215	HY000	No se puede añadir una restricción de clave ajena.
1216	HY000	No se puede añadir o eliminar una fila hija porque falla una restricción de clave ajena
1217	HY000	No se puede añadir o eliminar una fila padre porque falla una restricción de clave ajena
1231	42000	A la variable indicada no se le puede asignar el valor indicado
1242	21000	La subconsulta devuelve más de una fila.
1280	42000	Nombre de índice indicado incorrecto.
1329	02000	Cero filas (ningún dato) recibido, seleccionado o procesado
1348	HY000	La columna indicada no es modificable
1451	23000	No se puede eliminar o modificar una fila padre porque falla una restricción de clave ajena
1452	23000	No se puede insertar o modificar una fila hija porque falla una restricción de clave ajena

En el ejemplo expuesto con anterioridad, en el que se creó el procedimiento *InsertarPedido*, se creó un manejador para la excepción con código de error 1062, excepción que se produce cuando se intenta asignar un valor duplicado a un atributo con valor único. También se podría haber especificado en la instrucción de declaración del manejador en vez del código de error 1062 el valor de estado '23000', que es el que corresponde a esta excepción.

Creemos otro ejemplo de procedimiento con manejo de excepciones. Se trata de un procedimiento que recibe el código de un artículo y nos mostrará su descripción en caso de que exista un artículo con el código recibido como parámetro. En caso de que no exista

ningún artículo con dicho código, se mostrará un mensaje como el siguiente: ‘No existe ningún artículo con el código XXXXX’. Hemos de tener en cuenta que para obtener la descripción de un artículo a partir de su código tendremos que emplear una instrucción SELECT … INTO, de manera que si esta instrucción no nos devuelve ninguna fila por no haber ningún artículo con el código buscado, se producirá la excepción con código de error 1329 y valor de estado ‘02000’. Podremos especificar cualquiera de estos dos valores en la instrucción de declaración del manejador. Optamos por especificar el valor de estado. El procedimiento nos quedará como sigue:

```
delimiter //
create procedure MostrarDescri (codar char(5))
begin
declare encontrado bool default 1;
declare descri varchar(30);
declare continue handler for sqlstate '02000' set encontrado = 0;
select DesArt into descri from Articulo where codart = codar;
if encontrado = 0 then
    select concat ('No existe ningún artículo con el código ', codar) ERROR;
else
    select descri Descripción;
end if;
end//
```

Ahora si llamamos a este procedimiento pasándole el código de un artículo existente, nos mostrará su descripción; en caso contrario, se nos mostrará el mensaje de error que hemos especificado. Probémoslo:

```
mysql> call MostrarDescri ('A0043');//
+----------------+
| Descripción    |
+----------------+
| Bolígrafo azul |
+----------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> call MostrarDescri ('A0299');//
+-----------------------------------------------+
| ERROR                                         |
+-----------------------------------------------+
| No existe ningún artículo con el código A0299 |
+-----------------------------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

2. Cursores.

Hasta ahora todas las consultas que hemos creado en nuestros procedimientos y funciones nos devolvían una sola fila y esto no era casualidad. Esto se debe a que hemos empleado lo que se llaman cursores implícitos. Y es que estos cursores permiten solo manejar una fila. De hecho, si en los programas que hemos creado hasta ahora una consulta nos devolviera varias filas, se produciría una excepción que, al no ser tratada, provocaría la terminación anormal del programa.

Por todo esto, si queremos ejecutar consultas que devuelvan varias filas, debemos usar cursores explícitos, que vamos a tratar a continuación.

Para poder utilizar un cursor explícito, al igual que para poder utilizar una variable, es necesario declararlo. Un cursor se declara de acuerdo con la siguiente sintaxis:

```
DECLARE Nombre_cursor CURSOR FOR Sentencia_select;
```

Por su parte, para utilizar un cursor, será necesario, en primer lugar, abrirlo. Para ello, se utilizará la siguiente instrucción:

```
OPEN Nombre_cursor;
```

Al abrir un cursor, se ejecuta la sentencia *select* que se asignó al mismo en la declaración y se almacenan los resultados de la ejecución en estructuras internas de memoria.

Para acceder a la información almacenada en el cursor, es decir, a los datos resultantes de la ejecución de la sentencia select, es necesario usar la instrucción siguiente:

```
FETCH Nombre_cursor into Lista_variables;
```

donde *Lista_variables* puede ser una sola variable o varias variables separadas por comas. Habrá que escribir tantas variables separadas por comas como elementos aparezcan en la cláusula *select*. Esta/s variable/s tendrá/n que estar previamente declarada/s.

La instrucción *fetch* recupera una de las filas y pasa automáticamente a la siguiente fila de las resultantes de la ejecución de la sentencia *select*. Si al ejecutar una orden *fetch*, esta no devuelve datos, o lo que es lo mismo, si una vez leídas todas las filas del cursor, se pretende leer otra fila, se produce el error o excepción NOT FOUND (no encontrado). Si este error o excepción no es tratado, se producirá la terminación anormal del subprograma ejecutado.

Una vez empleado el cursor, es necesario cerrarlo con la siguiente instrucción:

```
CLOSE Nombre_cursor;
```

Para probar todo esto comencemos creando una tabla *Pedido2* copia de *Pedido*, pero sin datos. Para ello, ejecutemos las siguientes órdenes:

```
mysql> create table pedido2 as select * from pedido;
Query OK, 4 rows affected (0.34 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> delete from pedido2;
Query OK, 4 rows affected (0.05 sec)
```

Creemos a continuación un procedimiento que muestre por cada pedido su referencia y fecha mediante el empleo de un cursor. Primero lo aplicaremos sobre la tabla *Pedido* y luego sobre la tabla *Pedido2*. Comenzaremos creando un procedimiento que recupere solo una de las filas de la tabla *Pedido*. Para ello, declararemos un cursor, a continuación lo abriremos, seleccionaremos una fila y mostraremos su contenido. Finalmente cerraremos el cursor.

```
delimiter //
create procedure VerPedido()
begin
declare refe char(5);
declare fecha date;
declare c cursor for select refped, fecped from pedido;
open c;
fetch c into refe, fecha;
select concat ('Referencia: ', refe, ' Fecha: ', fecha) "Datos pedidos";
close c;
end//
```

Si ejecutamos este procedimiento, obtendremos el siguiente resultado.

```
mysql> call VerPedido();//
+-----------------------------------+
| Datos pedidos                     |
+-----------------------------------+
| Referencia: P0001 Fecha: 2018-02-16 |
+-----------------------------------+
1 row in set (0.00 sec)
```

Como podemos observar, nos muestra los datos del primer pedido almacenado en la tabla *Pedido*. Ahora eliminemos este procedimiento y creemos uno con el mismo nombre, pero que consulte la tabla *Pedido2*, copia de *Pedido* pero sin datos. Una vez creado, ejecutémoslo:

```
mysql> drop procedure VerPedido; //
Query OK, 0 rows affected (0.00 sec)

create procedure VerPedido()
begin
declare refe char(5);
declare fecha date;
declare c cursor for select refped, fecped from pedido2;
open c;
fetch c into refe, fecha;
select concat ('Referencia: ', refe, ' Fecha: ', fecha) "Datos pedidos";
close c;
end; //
```

```
mysql> call VerPedido();//
ERROR 1329 (02000): No data - zero rows fetched, selected, or processed
```

Como podemos observar, el procedimiento ha finalizado anormalmente pues se ha producido un error o excepción, concretamente la excepción con el número 1329, que indica que no se han encontrado datos. Esto es lo que ocurrirá cuando ejecutemos una orden *fetch* que no devuelva datos.

Pues bien, si trabajamos con cursores es porque deseamos que se recorran varias filas de una o varias tablas y que se muestre información referente a esas filas. Dado que con una orden *fetch* recuperamos los datos de una fila de un cursor, deberemos emplear varias órdenes *fetch* para recorrer las diversas filas resultado de ejecutar una consulta. Pues bien, utilizaremos un bucle *while* para ejecutar varias órdenes *fetch*. Siempre que empleemos un cursor, después de abrirlo, recuperaremos la primera fila del mismo con una orden *fetch* y luego realizaremos un cierto tratamiento sobre las filas recuperadas dentro de un bucle *while* que tendrá como condición el que la orden *fetch* que se acaba de ejecutar no haya provocado la excepción NOT FOUND.

Pues bien, para poder trabajar con excepciones tenemos que declarar lo que se llama un manejador de errores o *handler*. Para declarar un manejador para la excepción NOT FOUND debemos emplear la siguiente sintaxis:

```
DECLARE CONTINUE HANDLER FOR NOT FOUND SET NombreVariable = valor;
```

De esta manera, si al ejecutar una orden *fetch* se produce la excepción NOT FOUND, se asignará a la variable *NombreVariable* el valor indicado. Esta variable es necesario haberla declarado previamente. Normalmente lo que haremos es declarar una variable booleana llamada *fin* con valor inicial 0 y obtendremos los datos correspondientes a cada una de las filas del resultado de la ejecución de la sentencia *select* con una orden *fetch* dentro del bucle mientras que *fin* sea 0. Al declarar el manejador, indicaremos que cuando se produzca la excepción NOT FOUND, se asigne a la variable *fin* el valor 1. De esta manera, cuando una orden *fetch* no encuentre datos, se asignará a la variable *fin* el valor 1 y ya nos saldremos del bucle porque han sido tratadas todas las filas.

De esta manera, el procedimiento *VerPedido* correcto nos quedará con el siguiente código:

```
mysql> drop procedure VerPedido;//
Query OK, 0 rows affected (0.00 sec)
```

```
create procedure VerPedido()
begin
declare refe char(5);
declare fecha date;
declare fin bool default 0;
declare c cursor for select refped, fecped from pedido;
declare continue handler for not found set fin = 1;
open c;
fetch c into refe, fecha;
while fin = 0 do
   select concat ('Referencia: ', refe, ' Fecha: ', fecha) "Datos pedidos";
   Fetch c into refe, fecha;
end while;
close c;
end//

mysql> call verpedido();//
+-------------------------------------+
| Datos pedidos                       |
+-------------------------------------+
| Referencia: P0001 Fecha: 2018-02-16 |
+-------------------------------------+
1 row in set (0.00 sec)

+-------------------------------------+
| Datos pedidos                       |
+-------------------------------------+
| Referencia: P0002 Fecha: 2018-02-18 |
+-------------------------------------+
1 row in set (0.00 sec)

+-------------------------------------+
| Datos pedidos                       |
+-------------------------------------+
| Referencia: P0003 Fecha: 2018-02-23 |
+-------------------------------------+
1 row in set (0.00 sec)

+-------------------------------------+
| Datos pedidos                       |
+-------------------------------------+
| Referencia: P0004 Fecha: 2018-02-25 |
+-------------------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.02 sec)
```

Al trabajar con cursores puede ocurrir que no nos sea posible indicar en la sentencia *select* asociada al cursor los términos exactos de la sentencia, sino que tengamos que utilizar una o varias variables. Pues bien, estas variables reciben el nombre de variables de acoplamiento. Para usarlas, será necesario declararlas como cualquier otra variable y luego utilizarlas en la sentencia *select* asociada al cursor. Si estas variables aparecen definidas como parámetros del procedimiento o función, entonces no es necesario declararlas explícitamente con *declare*.

Por ejemplo, para mostrar para los pedidos en los que se solicita un determinado artículo identificado por su código (parámetro del procedimiento), el código del pedido y el número de unidades solicitadas del artículo, podemos crear el siguiente procedimiento:

```
create procedure VerArticuloPedido (art CHAR(5))
begin
declare refe char(5);
declare cant int;
declare fin bool default 0;
declare c cursor for select refped, cantart from lineapedido where codart =
                     art;
declare continue handler for not found set fin = 1;
open c;
fetch c into refe, cant;
while fin = 0 do
   select concat ('Pedido: ', refe, ' Unidades: ', cant) "Datos línea pedido";
   fetch c into refe,cant;
end while;
close c;
end//

mysql> call VerArticuloPedido('A0043');//
+----------------------------+
| Datos                      |
+----------------------------+
| Pedido: P0001 Unidades: 10 |
+----------------------------+
1 row in set (0.00 sec)

+---------------------------+
| Datos                     |
+---------------------------+
| Pedido: P0002 Unidades: 5 |
+---------------------------+
1 row in set (0.00 sec)

+---------------------------+
| Datos                     |
+---------------------------+
| Pedido: P0004 Unidades: 5 |
+---------------------------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)
```

Como vemos, en la declaración del cursor se ha incluido una variable de acoplamiento (*art*) que es un parámetro del procedimiento. De esta manera, el resultado de la ejecución del procedimiento será diferente en función del artículo que reciba como parámetro el procedimiento.