
	TEMA 9: Gestión de errores.- EXCEPCIONES (HANDLER)- CURSORES
	

Una excepción es el aviso de un error que se está produciendo durante la ejecución de un trozo de código. Durante la ejecución de procedimientos almacenados pueden darse errores que hagan que el procedimiento falle. Mysql permite 'capturar' esos errores y tratarlos mediante programación permitiendo que el procedimiento continúe su ejecución. Para ello deberemos declarar un handler (que podemos traducir literalmente como "manejador" de excepciones) de la siguiente manera:

DECLARE tipo_handler HANDLER FOR condición[,...]

lista_sentencias

donde **tipo_handler** toma valor de: **[CONTINUE | EXIT]**

CONTINUE: la ejecución del programa continua

EXIT: la ejecución termina

y condición toma valor de **error_code | SQLSTATE [VALUE]**

error_code y **SQLSTATE** son valores que los encontramos en el mensaje de error que queremos controlar. Basta con ejecutar el código hasta esperar a que aparezca el mensaje de error, y entonces apuntar dicho código para introducirlo en el manejador, o bien, consultar en la ayuda de MySQL el listado de todos los mensajes de error susceptibles de ser captirados y controlados por los manejadores: <https://dev.mysql.com/doc/refman/5.7/en/server-error-reference.html>

Un caso típico podría ser cuando intentamos dar de alta (INSERT) una fila con una clave primaria que ya existe en la tabla. Al intentar hacer esto, el gestor 'rompe' la ejecución del procedimiento almacenado en la línea del INSERT e informa del problema. Otro ejemplo sería intentar modificar una tabla que no existe, etc. Lo que hace Mysql es 'declarar' primero cuales son los errores que se van a capturar y cuáles serán las instrucciones que se deben ejecutar en caso de que se produzca. Esto hará que el procedimiento almacenado continúe la ejecución con la siguiente línea a la que provocó el error.



TEMA 9: Gestión de errores.- EXCEPCIONES (HANDLER)- CURSORES

-- EJEMPLO DE CONTROL DE ERRORES AL INSERTAR EN UNA TABLA UNA FILA CUYA CLAVE YA EXISTE---

/* CREAMOS UNA BD PARA HACER LOS EJEMPLOS Y NOS CONECTAMOS A ELLA*/

```
DROP DATABASE IF EXISTS EJEMPLOS_HANDLER;  
CREATE DATABASE EJEMPLOS_HANDLER;  
USE EJEMPLOS_HANDLER;
```

/* CREAMOS UNA TABLA E INSERTAMOS ALGUNOS REGISTROS DE PRUEBA*/

/* CREAMOS EL PROCEDIMIENTO QUE INSERTA UNA NUEVA FILA. DECLARAMOS UN MANEJADOR DE ERRORES HANDLER PARA CONTROLAR QUE NO INSERTEMOS VALORES DE PK QUE YA EXISTEN*/

```
DELIMITER //  
DROP PROCEDURE IF EXISTS INSERTAR_VALORES_REPETIDOS//  
CREATE PROCEDURE INSERTAR_VALORES_REPETIDOS(PAR1 INT)  
BEGIN  
    DECLARE error boolean default false;  
    DECLARE CONTINUE HANDLER FOR SQLSTATE '23000'  
    BEGIN  
        SET error=true;  
    END;  
    INSERT INTO TABLA1 VALUES (PAR1);  
    IF (error) THEN  
        SELECT 'Clave primaria duplicada. No se puede insertar la fila' AS MENSAJE;  
    ELSE  
        SELECT 'Fila añadida' AS MENSAJE;  
    END IF;  
END//  
DELIMITER ;
```

EJECUTO EL PROCEDIMIENTO INSERTANDO UN VALOR QUE YA ESTÁ EN LA TABLA.

```
CALL INSERTAR_VALORES_REPETIDOS(1);
```

MENSAJE

Clave primaria duplicada. No se puede insertar la fila

EJECUTO EL PROCEDIMIENTO INSERTANDO UN VALOR QUE NO ESTÁ EN LA TABLA.

```
CALL INSERTAR_VALORES_REPETIDOS(10);
```

MENSAJE

Fila añadida



TEMA 9: Gestión de errores.- EXCEPCIONES (HANDLER)- CURSORES

TEMA 9: Gestión de errores.- EXCEPCIONES (HANDLER)- CURSORES

Ejemplo 2: Control de errores cuando se intenta insertar una fila en una tabla que no existe.

```
DELIMITER //
DROP PROCEDURE IF EXISTS INSERTAR_TABLA_PRUEBA1//
CREATE PROCEDURE INSERTAR_TABLA_PRUEBA1(PAR1 INT)
BEGIN
    DECLARE error boolean default false; /*DECLARO UNA VARIABLE PARA CHEQUEAR EL POSIBLE ERROR*/
    DECLARE CONTINUE HANDLER FOR 1146 /* ES EL ERROR CUANDO LA TABLA REFERIDA NO EXISTE*/
    BEGIN
        SET error=true; /* SI OCURRE EL ERROR SE CAPTURA Y LA VARIABLE ERROR TOMA EL VALOR TRUE*/
    END;
    INSERT INTO TABLA_PRUEBA VALUES (PAR1);
    IF error THEN
        SELECT CONCAT('LA TABLA TABLA_PRUEBA NO EXISTE EN LA BASE DE DATOS') AS MENSAJE;
    ELSE
        SELECT 'Fila añadida' AS MENSAJE;
    END IF;
END//
DELIMITER ;
```

LLAMAMOS AL PROCEDIMIENTO CON UNA TABLA QUE NO TENEMOS EN LA BASE DE DATOS.

```
CALL INSERTAR_TABLA_PRUEBA1(4);
```

MENSAJE

LA TABLA TABLA_PRUEBA NO EXISTE EN LA BASE DE DATOS

EL MISMO EJEMPLO CAMBIANDO EL ERROR DEVUELTO POR EL CÓDIGO SQLSTATE

```
DELIMITER //
DROP PROCEDURE IF EXISTS INSERTAR_TABLA_PRUEBA2//
CREATE PROCEDURE INSERTAR_TABLA_PRUEBA2(PAR1 INT)
BEGIN
    DECLARE error boolean default false;
    DECLARE CONTINUE HANDLER FOR SQLSTATE '42S02'
    BEGIN
        SET error=true;
    END;
    INSERT INTO TABLA_PRUEBA VALUES (PAR1);
    IF (error) THEN
        SELECT CONCAT('LA TABLA TABLA_PRUEBA NO EXISTE EN LA BASE DE DATOS') AS MENSAJE;
    ELSE
        SELECT 'Fila añadida' AS MENSAJE;
    END IF;
END//
```

```
CALL INSERTAR_TABLA_PRUEBA2(4);
```

MENSAJE

LA TABLA TABLA_PRUEBA NO EXISTE EN LA BASE DE DATOS

TEMA 9: Gestión de errores.- EXCEPCIONES (HANDLER)- CURSORES

SI EN EL MISMO PROCEDIMIENTO UTILIZAMOS EXIT HANDLER SE CAPTURARÁ EL ERROR Y SE FORZARÁ A FINALIZAR EL PROCEDIMIENTO (NO SE MOSTRARÁN LOS MENSAJES)*/

Esto no haría falta si pones en el cuerpo del handler el mensaje de error y abajo del todo el mensaje de fila añadida porque es un EXIT

```

DELIMITER //
DROP PROCEDURE IF EXISTS INSERTAR_TABLA_PRUEBA3//
CREATE PROCEDURE INSERTAR_TABLA_PRUEBA3(PAR1 INT)
BEGIN
  DECLARE error boolean default false;
  DECLARE EXIT HANDLER FOR SQLSTATE '42S02'
  BEGIN
    SET error=true;
  END;
  INSERT INTO TABLA_PRUEBA VALUES (PAR1);
  IF (error) THEN
    SELECT CONCAT('LA TABLA TABLA_PRUEBA NO EXISTE EN LA BASE DE DATOS') AS MENSAJE;
  ELSE
    SELECT 'Fila añadida' AS MENSAJE;
  END IF;
END//

CALL INSERTAR_TABLA_PRUEBA3(4);
  
```

118 16:49:57 CALL INSERTAR_TABLA_PRUEBA3(4) 0 row(s) affected

CAPTURE EL ERROR Y FINALIZA. EN CAMBIO SI CAMBIAMOS POR UNA TABLA QUE SI EXISTA, INSERTA LA FILA SIN PROBLEMAS

Mensaje
Fila añadida

AHORA UTILIZAR EXIT HANDLER PERO PERMITIR QUE EL PROC. MUESTRE UN MENSAJE DE ERROR

```



DELIMITER //
DROP PROCEDURE IF EXISTS INSERTAR_TABLA_PRUEBA4//
CREATE PROCEDURE INSERTAR_TABLA_PRUEBA4(PAR1 INT)
BEGIN
  DECLARE error_tabla_no_existe condition for 1146;
  DECLARE exit HANDLER FOR error_tabla_no_existe select 'LA TABLA NO EXISTE' as MENSAJE;
  INSERT INTO TABLA_PRUEBA VALUES (PAR1);
  SELECT 'FILA AÑADIDA' AS MENSAJE;
END//

CALL INSERTAR_TABLA_PRUEBA4(2);
  
```

Mensaje
LA TABLA NO EXISTE

Mensaje
FILA AÑADIDA

SI LA TABLA EXISTE, AL LLAMAR AL PROCEDIMIENTO:

	<h1>TEMA 9: Gestión de errores.- EXCEPCIONES (HANDLER)- CURSORES</h1>
	

Ejemplo 3: Control de errores cuando se intenta insertar una fila en una tabla y el valor del campo hace referencia a un registro de la tabla referenciada que no existe. (Error de integridad referencial provocado por una FK).

```
CREATE TABLE TABLA1 (CAMPO1 INT PRIMARY KEY);
```

/ TENEMOS LA TABLA1 CON ALGUNOS CAMPOS*/*

CAMPO:
1
2
3
4

```
SELECT * FROM TABLA1;
```

/ CREAMOS TABLA2 VACÍA CON UNA FK A TABLA1, DE FORMA QUE LAS 2 TABLAS ESTÉN RELACIONADAS*.*

```
CREATE TABLE TABLA2 (CAMPO1 INT PRIMARY KEY, FOREIGN KEY (CAMPO1) REFERENCES TABLA1 (CAMPO1) );
-- CREAMOS TABLA2 VACÍA CON UNA FK A TABLA1, DE FORMA QUE LAS 2 TABLAS ESTÉN RELACIONADAS
```

```
INSERT INTO TABLA2 VALUES (8);
```

Message
Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`ejemplos_handler`.`tabla2`, CONSTRAINT `tabl..`

/ INTENTAMOS INSERTAR UN VALOR EN LA TABLA TABLA2, Y ESE VALOR NO SE ENCUENTRA EN LA TABLA1, POR LO TANTO NOS DARÁ UN ERROR!!! NOS QUEDAMOS CON EL ERROR PORQUE LO UTILIZAREMOS EN EL HANDLER QUE CONTROLARÁ EL ERROR EN EL PROCEDIMIENTO. CREAMOS UNA TABLA EN LA QUE VAMOS A IR ALMACENANDO A MODO DE LOG LOS ERRORES QUE NOS VAN SALIENDO. SOLO TIENE UN CAMPO DE TEXTO Y AL CREARLA ESTARÁ INICIALMENTE VACÍA*/*



```
CREATE TABLE TABLA_ERRORES (TEXTO VARCHAR(200));
```

```
DELIMITER //
DROP PROCEDURE IF EXISTS PROCEDIMIENTO_CON_HANDLER//
CREATE PROCEDURE PROCEDIMIENTO_CON_HANDLER(PAR INT)
BEGIN
    DECLARE EXIT HANDLER FOR 1452
    -- EXIT SIGNIFICA QUE CUANDO TERMINE LA EJECUCIÓN DEL HANDLER SE SALE DEL PROCEDIMIENTO ALMACENADO;
    -- SI PONEMOS CONTINUE EN LUGAR DE EXIT, LA EJECUCIÓN DEL PROCEDIMIENTO ALMACENADO CONTINUARÍA.
    BEGIN
        INSERT INTO TABLA_ERRORES VALUES
        (CONCAT('EN LA FECHA:', ' ', CURRENT_DATE(), ' ERROR DE CLAVE FORÁNEA PARA EL VALOR:', ' ', PAR,
        ' NO EXISTE EL DATO EN LA TABLA REFERENCIADA'));
    END;
    INSERT INTO TABLA2 VALUES (PAR);
END; //;
```

```
CALL PROCEDIMIENTO_CON_HANDLER(6);
```

```
SELECT * FROM TABLA2;
SELECT * FROM TABLA_ERRORES;
```

TEXTO
EN LA FECHA: 2019-05-12 ERROR DE CLAVE FORÁNEA PARA EL VALOR: 6 NO EXISTE EL DATO EN LA TABLA REFERENCIADA

	TEMA 9: Gestión de errores.- EXCEPCIONES (HANDLER)- CURSORES
	

Cursores (CURSOR)

Los cursores son estructuras temporales de almacenamiento auxiliar muy útiles cuando se construyen procedimientos sobre bases de datos. Para crear un cursor es necesario declararlo y definir la consulta select que lo llenará de valores:

```
DECLARE nombre_cursor CURSOR FOR sentencia_select
```

- La sentencia SELECT no puede contener INTO.
- Los cursores no son actualizables.
- Se recorren en un sentido, sin saltos.
- Se deben declarar antes de los handlers y después de la declaración de variables

Ejemplo:

```
DECLARE cur1 CURSOR FOR select dni from profesores;
```

Manipulación de cursores



- **OPEN** nombre_cursor -- Para abrir el cursor que se haya definido
- **FETCH** nombre_cursor **INTO** nombre_variable --Para desplazarnos por el cursor. Si no existen más registros disponibles, ocurrirá una condición de Sin Datos con el valor SQLSTATE 02000. Se debe definir una excepción para detectar esta condición
- **CLOSE** nombre_cursor -- Para cerrar el cursor

EJEMPLO DE CURSOR:

CREAMOS UNA BASE DE DATOS PARA PRACTICAR Y UNA TABLA SENCILLA:

```
CREATE DATABASE EJEMPLO_CURSOR;
USE EJEMPLO_CURSOR;
DROP TABLE IF EXISTS TABLA1;
CREATE TABLE TABLA1(ID1 INT PRIMARY KEY, DATO1 INT);
```

EL CURSOR RECORRE LA TABLA, CHEQUEA LOS VALORES Y EN FUNCIÓN DE ESE VALOR INSERTA EN LA TABLA AUXILIAR (TABLA_RESULTADOS) UN VALOR U OTRO.

	<h1>TEMA 9: Gestión de errores.- EXCEPCIONES (HANDLER)- CURSORES</h1>
 <p>UNION EUROPEA Fondo Social Europeo "Una manera de hacer Europa"</p>	

CREAMOS UNA TABLA AUXILIAR PARA INSERTAR FILAS EN FUNCIÓN DE LA CONDICIÓN:

```
DROP TABLE IF EXISTS TABLA_RESULTADO;
CREATE TABLE TABLA_RESULTADO(ID_R INT, DATO_R INT);
```

```
DELIMITER //
DROP PROCEDURE IF EXISTS CURSORES1//
CREATE PROCEDURE CURSORES1()
BEGIN
    DECLARE FIN_TABLA INT DEFAULT 0;
    DECLARE VAR_ID1,VAR_DATO1 INT;
    DECLARE CUR1 CURSOR FOR SELECT ID1,DATO1 FROM TABLA1;
    DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET FIN_TABLA = 1; /* UNA VEZ QUE TODAS LAS FILAS HAN SIDO
    PROCESADAS O LA SENTENCIA FETCH SE POSICIONA EN UNA FÍLA NO EXISTENTE EL SISTEMA DEVUELVE EL SQLSTATE 02000*/
    OPEN CUR1;
    REPEAT
        FETCH CUR1 INTO VAR_ID1,VAR_DATO1;
        IF NOT FIN_TABLA THEN
            IF VAR_DATO1 > 10 THEN INSERT INTO TABLA_RESULTADO VALUES (VAR_DATO1,VAR_ID1);
            ELSE INSERT INTO TABLA_RESULTADO VALUES (0,0);
            END IF;
        END IF;
    UNTIL FIN_TABLA
    END REPEAT;
    CLOSE CUR1;
END;
```

LLAMAMOS AL CURSOR:

```
CALL CURSORES1();
```

COMPROBAMOS EL CONTENIDO DE LA TABLA_RESULTADO:

ID_R	DATO_R
50	1
0	0
0	0

EJEMPLO 2: EJEMPLO DE PROCEDIMIENTO CON 2 CURSORES: CADA CURSOR RECORRE UNA TABLA, SE COMPARAN LOS DATOS Y SE INSERTA EN LA TABLA 3 DEPENDIENDO DEL RESULTADOS DE LA COMPARACIÓN.

CREAMOS LA TABLA3 PARA GUARDAR LOS RESULTADOS:

```
DROP TABLE IF EXISTS TABLA3;
CREATE TABLE TABLA3(ID3 INT PRIMARY KEY, DATO3 INT);
```

ESTA TABLA INICIALMENTE ESTÁ VACÍA.

INTRODUCIMOS VALORES EN TABLA1, TABLA2.

TEMA 9: Gestión de errores.- EXCEPCIONES (HANDLER)- CURSORES

```
INSERT INTO TABLA1 VALUES(1,7);
INSERT INTO TABLA1 VALUES(2,8);
INSERT INTO TABLA1 VALUES(3,9);
INSERT INTO TABLA2 VALUES(10,11);
INSERT INTO TABLA2 VALUES(20,22);
INSERT INTO TABLA2 VALUES(30,33);
```

```
DELIMITER //
DROP PROCEDURE IF EXISTS CURSORES2//
CREATE PROCEDURE CURSORES2()
BEGIN
    DECLARE FIN_TABLA INT DEFAULT 0; /*VARIABLE QUE CONTROLA EL FIN DE LA TABLA*/
    DECLARE VAR_ID1, VAR_ID2, VAR_DATO1 INT; /*VARIABLES PARA GUARDAR LOS DATOS*/
    DECLARE CUR1 CURSOR FOR SELECT ID1, DATO1 FROM TABLA1; /*CURSOR QUE RECORRE TABLA1*/
    DECLARE CUR2 CURSOR FOR SELECT ID2 FROM TABLA2; /*CURSOR QUE RECORRE TABLA2*/
    DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET FIN_TABLA = 1;
    /* UNA VEZ QUE TODAS LAS FILAS HAN SIDO PROCESADAS O LA SENTENCIA FETCH SE POSICIONA EN UNA
    FILA NO EXISTENTE EL SISTEMA DEVUELVE EL SQLSTATE 02000*/

    OPEN CUR1; OPEN CUR2; /*ABRIMOS LOS 2 CURSORES*/
    REPEAT
        FETCH CUR1 INTO VAR_ID1, VAR_DATO1;
        FETCH CUR2 INTO VAR_ID2;
        IF NOT FIN_TABLA THEN
            IF VAR_DATO1 > VAR_ID2 THEN
                INSERT INTO TABLA3 VALUES (VAR_DATO1, VAR_ID1);
            ELSE INSERT INTO TABLA3 VALUES (VAR_DATO1, VAR_ID2);
            END IF;
        END IF;
    UNTIL FIN_TABLA
    END REPEAT;
    CLOSE CUR1;
    CLOSE CUR2;
END;
//
```

CALL CURSORES2();

PARA LOS DATOS QUE TENEMOS EN TABLA1 Y TABLA2,

ID1	DATO1	ID2	DATO2
1	50	10	11
2	8	20	22
3	9	30	33
NULL	NULL	NULL	NULL

Y

ID3	DATO3
8	20
9	30
50	1
NULL	NULL

ESTE ES EL RESULTADO DE LA TABLA 3: