



Invoice

Invoice_id
Customer_id
Order_id
Product_id
Date_time
Status
Total
Remark

UT10. SQL MODO PROGRAMACIÓN

Módulo: BASES DE DATOS

Curso 2022/2023. 1º DAM

Ruth Lospitao Ruiz



CONTENIDOS

- Introducción. Conceptos generales
- Variables, operadores, expresiones
- Bloques instrucciones de código
- Estructuras de control.
- Procedimientos y funciones almacenados.
- Cursores
- Manejo de errores
 - Introducción
 - Tipos de manejadores
 - Condición de manejador
 - Otras cuestiones



INTRODUCCIÓN Y EJEMPLOS



CONCEPTO

- En general, si una sentencia SQL falla dentro de un programa almacenado se produce una situación de error, se interrumpe la ejecución del programa en ese punto y finaliza salvo en el caso de que el programa que falla hubiera sido llamado por otro; en ese caso la ejecución continua por el programa que llamó a este programa que ha causado el error; ocurriría lo mismo si en lugar de un programa que es llamado desde otro programa nos encontráramos con un bloque interno dentro de otro bloque más externo; si se produjera error en el bloque anidado interno, la ejecución del programa se interrumpiría en el bloque interno para continuar por el externo.
- Una handler es un bloque de instrucciones SQL que se ejecuta cuando se verifica una condición tras una excepción (error) generada por el servidor.



SINTAXIS

El manejador puede ser de tres tipos:

- CONTINUE: La excepción o error generado no interrumpe el código del procedimiento.
- EXIT: Permite poner fin al bloque de instrucciones o programa en el que se genera la excepción.
- UNDO: No está soportada por el momento

```
DECLARE {CONTINUE | EXIT | UNDO} HANDLER FOR  
        {SQLSTATE sqlstate_code | MySQL error code | nombre_condición}  
        instrucciones_del_manejador
```

La condición de error del manejador puede expresarse también de 3 formas:

- Con un código estándar ANSI SQLSTATE.
- Con un código de error MySQL.
- Una expresión.



EJEMPLO 1A

- El siguiente procedimiento intenta insertar un registro que ya existe previamente. Es decir, se está intentando dar de alta un departamento con dep_no 10 cuando ese valor ya existe y es PK de la tabla

```
/*Ejemplo de uso de manejadores.  
Se intenta insertar un departamento que ya existe*/  
use curso;  
DROP PROCEDURE IF EXISTS handler1;  
DELIMITER //  
CREATE PROCEDURE handler1 () modifies sql data  
BEGIN  
    -- Se intenta insertar un registro que ya existe en la tabla  
    insert into departamentos values(10,'Otro Dpto', 'Madrid');  
END //  
DELIMITER ;  
call handler1();
```

Aparecerá un error con un número de error (1062) y una descripción (“entrada duplicada”) advirtiéndonos de tal situación; evidentemente no se realiza la inserción y la ejecución del programa se detiene y finaliza

Error Code: 1062. Duplicate entry '10' for key 'departamentos.PRIMARY'



EJEMPLO1B

- Para solucionar esta situación, insertar un valor PK que ya existe creamos un **handler** que lo capture mostrando un mensaje de error

```
/*Ejemplo de uso de manejadores.  
Se intenta insertar un departamento que ya existe para evitar el error anterior se crea un handler para ese código*/  
use curso;  
DROP PROCEDURE IF EXISTS handler1b;  
DELIMITER //  
CREATE PROCEDURE handler1b () modifies sql data  
BEGIN  
    DECLARE CONTINUE HANDLER FOR 1062  
        SELECT ('Num Dpto ya existe') as 'Aviso de error';  
  
    insert into departamentos values(10,'Otro Dpto', 'Madrid');  
END //  
DELIMITER ;  
call handler1b();
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	Aviso de error
▶	Num Dpto ya existe



EJEMPLO2A

- Procedimiento almacenado que recibe como parámetros de entrada los valores que se insertan en un nuevo registro

```
/*Ejemplo de uso de manejadores.  
Procedimiento que recibe como parámetros los valores del nuevo dpto que inserta*/  
  
• use curso;  
• DROP PROCEDURE IF EXISTS handler2a;  
  DELIMITER //  
• CREATE PROCEDURE handler2a (id int, nombre varchar(14), localidad varchar(10)) modifies sql data  
  BEGIN  
    DECLARE CONTINUE HANDLER FOR 1062  
      SELECT CONCAT ('Num Dpto ya existe') as 'Aviso de error';  
  
    insert into departamentos values(id, nombre, localidad);  
  END //  
  DELIMITER ;  
• call handler2a(10,'Otro Dpto', 'Madrid');  
• call handler2a(null,'Otro Dpto', 'Madrid');
```

Se ha controlado el error de insertar un dpto que ya existe, pero debemos controlar que el id de dpto tenga valor. En la última línea se produce un error porque estamos enviando un null



Error Code: 1048. Column 'DEP_NO' cannot be null



EJEMPLO2A

- Para solucionar el error anterior se agrega el código de error a otro handler

```
2  \ Procedimiento que recibe como parámetros los valores del nuevo dpto que inserta*/
3  • use cursoj;
4  • DROP PROCEDURE IF EXISTS handler2b;
5  DELIMITER //
6  • CREATE PROCEDURE handler2b (id int, nombre varchar(14), localidad varchar(10)) modifies sql data
7  BEGIN
8      DECLARE CONTINUE HANDLER FOR 1062 -- captura error de pk duplicada
9          SELECT CONCAT ('Num Dpto ', id , ' ya existe') as 'Aviso de error';
10     DECLARE CONTINUE HANDLER FOR 1048 -- captura error de pk null
11         SELECT ('Num Dpto no puede ser NULL') as 'Aviso de error';
12     insert into departamentos values(id, nombre, localidad);
13 END //
14 DELIMITER ;
15 • call handler2b(10,'Otro Dpto', 'Madrid');
16 • call handler2b(null,'Otro Dpto', 'Madrid');
17
```

Result Grid		Filter Rows: <input type="text"/>	Export: 	Wrap Cell Content: 
	Aviso de error			
▶	Num Dpto no puede ser NULL			



EJEMPLO 3

- Sobre el ejemplo anterior, añadimos un parámetro OUT que informe qué sucedió en el procedimiento almacenado

```
1  /*Ejemplo de uso de manejadores.*/
2  • use curso;
3  • DROP PROCEDURE IF EXISTS handler3a;
4  DELIMITER //
5  • CREATE PROCEDURE handler3a (id int, nombre varchar(14), localidad varchar(10), out resultado varchar(50)) modifies sql data
6  BEGIN
7      DECLARE CONTINUE HANDLER FOR 1062 -- captura error de pk duplicada
8          set resultado = CONCAT ('Num Dpto ', id , ' ya existe') ;
9      DECLARE CONTINUE HANDLER FOR 1048 -- captura error de pk null
10         set resultado = 'Num Dpto no puede ser NULL';
11         set resultado = 'ok';
12         insert into departamentos values(id, nombre, localidad);
13     END //
14     DELIMITER ;
15
16     DELIMITER //
17     • DROP PROCEDURE IF EXISTS handler3b;
18     CREATE PROCEDURE handler3b ()
19     BEGIN
20         DECLARE resul varchar(50);
21         call handler3a (10, 'Otro', 'Madrid', resul);
22         if resul = 'ok' then
23             select 'Registro dado de alta' as 'Inserción';
24         else
25             select resul as 'Error';
26         end if;
27     END //
28     DELIMITER ;
29     • call handler3b();
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

Error

Num Dpto 10 ya existe

SINTAXIS

- Volviendo otra vez a la declaración de un manejador, hemos visto que tiene 3 partes que pasaremos a ver a continuación más en detalle

1. Tipo de manejador: `CONTINUE` o `EXIT`.
2. Condición del manejador: `SQLSTATE sqlstate_code | MySQL error code | nombre_condición.`
3. Acciones o instrucciones que realiza el manejador.



TIPOS DE MANEJADOR



INTRODUCCIÓN

- **EXIT:** El programa que causa el error finaliza, devolviendo el control al programa que le llamó. Si se produce en un bloque interno, entonces el control de la ejecución pasa al externo.
- **CONTINUE:** La ejecución continúa por la siguiente línea de código que causó el error.
- En ambos casos el error es tratado por lo que la ejecución del programa puede considerarse correcta y antes de realizarse la opción CONTINUE o EXIT se ejecuta el conjunto de instrucciones asociados al manejador



EJEMPLO MANEJADOR EXIT

```
1  /*Ejemplo de uso de manejadores.*/
2  • use curso;
3  • DROP PROCEDURE IF EXISTS handler_exit1;
4  DELIMITER //
5  • CREATE PROCEDURE handler_exit1 (id int, nombre varchar(14), localidad varchar(10)) modifies sql data
6  BEGIN
7      DECLARE clave_repetida tinyint default 0;
8      BEGIN
9          DECLARE EXIT HANDLER FOR 1062
10             set clave_repetida = 1;
11             insert into departamentos values(id, nombre, localidad);
12             select concat ('Registro dado de alta: ', id, ' - ', nombre, ' - ', localidad) as 'Inserción';
13
14     END;
15     if clave_repetida = 1 then
16         select concat('Id dpto ', id, ' ya existe') as 'Aviso de error';
17     end if;
18
19 END //
20 DELIMITER ;
21 • call handler_exit1(10,'Otro', 'Madrid');
22 • call handler_exit1(150,'Otro', 'Madrid');
```

Dara como resultado el mensaje de error que Id dpto ya existe. La línea del insert no se realiza. Como el handler es de tipo EXIT, se interrumpe la ejecución del programa y sale del bloque interno. Si hubiera habido un solo bloque hubiera salido del programa. Por tanto, no llega a ejecutar la línea 12 y el programa sigue ejecutando por la línea 15

Dara como resultado la inserción de un nuevo departamento

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
Inserción			
▶ Registro dado de alta: 150 - Otro - Madrid			



EJEMPLO MANEJADOR CONTINUE

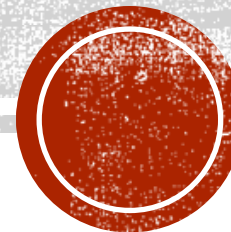
```
1  /*Ejemplo de uso de manejadores.*/
2  • use curso;
3  • DROP PROCEDURE IF EXISTS handler_continue1;
4  DELIMITER //
5  • CREATE PROCEDURE handler_continue1 (id int, nombre varchar(14), localidad varchar(10)) modifies sql data
6  BEGIN
7      DECLARE clave_repetida tinyint default 0;
8      BEGIN
9          DECLARE CONTINUE HANDLER FOR 1062
10             set clave_repetida = 1;
11             insert into departamentos values(id, nombre, localidad);
12             select concat ('Registro dado de alta: ', id, ' - ', nombre, ' - ', localidad) as 'Inserción';
13         END;
14         if clave_repetida = 1 then
15             select concat('Id dpto ', id, ' ya existe') as 'Aviso de error';
16         end if;
17     END //
18 DELIMITER ;
19 • call handler_continue1(10,'Otro', 'Madrid');
20 • call handler_continue1(151,'Otro', 'Madrid');
```

Dara como resultado el mensaje de error que Id dpto ya existe (igual que en caso anterior cuando se utilizó EXIT) que es tratado inmediatamente después en el manejador de error poniendo a 1 el valor de la variable clave_repetida. Por tanto, al ser tratado el error y tener clausula CONTINUE la ejecución continua por la línea 14 evaluando la confición (que es cierta) y mostrando el mensaje de error.

Dara como resultado la inserción de un nuevo departamento



CONDICIÓN DE MANEJADOR



SINTAXIS

- Has 3 formas posibles de indicar cuando debe ser invocado el conjunto de instrucciones del manejador de error para tratarlo.

- 1) Número de error de MySQL.
- 2) Código SQLSTATE estándar ANSI.
- 3) Condiciones de error con nombre



NÚMERO DE ERROR DE MYSQL

- Por ejemplo, el número de error 1062 está asociado en MySQL al intento de almacenar un registro de clave duplicada.

```
DECLARE CONTINUE HANDLER FOR 1062  
SET v_clave_repetida=1;
```

<https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html>

<https://dev.mysql.com/doc/mysql-errors/8.0/en/client-error-reference.html>

<https://dev.mysql.com/doc/mysql-errors/8.0/en/global-error-reference.html>



CÓDIGO SQLSTATE ESTÁNDAR ANSI

- A diferencia del anterior, SQLSTATE no es definido por MySQL sino que es un estándar y por tanto el mismo error tiene asociado el mismo SQLSTATE independientemente del gestor de bases de datos utilizado: MySQL, Oracle, DB2, Microsoft SQL Server. En los anteriores gestores de bases de datos el SQLSTATE 23000 está asociado al error de clave duplicada.
- Por tanto en MySQL el anterior manejador de error y el siguiente son similares

```
DECLARE CONTINUE HANDLER FOR SQLSTATE '23000'  
SET v_clave_repetida=1;
```

¿Cuál utilizar de los dos anteriores? En teoría el segundo permitiría una mayor portabilidad a otros gestores de bases de datos pero en la práctica los lenguajes de programación para Oracle y Microsoft SQL Server son incompatibles con el de MySQL por lo que no hace muy atractivo el uso de esta segunda opción (DB2 de IBM es “algo” compatible pues tanto DB2 como MySQL están basados en el estándar SQL:2003).

Además hay códigos SQLSTATE que no corresponden con un código de error de MySQL sino a varios (para estos casos se utiliza el SQLSTATE genérico 'HY000')

Por lo anteriormente expuesto seguiremos utilizando los códigos propios de error de MySQL

CONDICIONES DE ERROR CON NOMBRE

- Pueden ser: predefinidas o definidas por el usuario
- PREDEFINIDAS:
 - SQLEXCEPTION: es un subconjunto de los códigos SQLSTATE, representando todos los códigos que no comienzan ni por 01 ni por 02
 - SQLWARNING: es un subconjunto de los códigos SQLSTATE, representando todos los códigos que empiezan por 01
 - NOT FOUND: es un subconjunto de los códigos SQLSTATE, representando todos los códigos que empiezan por 02
- DEFINIDAS POR EL USUARIO
 - Facilitan lectura del código y por tanto el fácil mantenimiento de la aplicación.
 - Consiste en “bautizar” o darle un nombre a un código de error MySQL o SQLSTATE.



EJEMPLOS CONDICIONES ERROR PREDEFINIDAS

```
/* Si ocurre cualquier condición de error (salvo la excepción NOT FOUND tratada en el apartado de cursores) la variable v_error valdrá 1 y continuará la ejecución del programa */
```

```
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION  
    SET v_error=1;
```



EJEMPLOS CONDICIONES ERROR PREDEFINIDAS

/* Si ocurre cualquier condición de error (excepto la condición NOT FOUND) el procedimiento finaliza ejecutando antes la instrucción ROLLBACK (deshacer operaciones que se hubieran realizado) y otra de advertencia de la situación de error */

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



EJEMPLOS CONDICIONES ERROR PREDEFINIDAS

/ Si la instrucción FETCH en un cursor no recupera ninguna fila*/*

/ De 3 formas distintas: Condición de error con nombre, SQLSTATE y código de error de MySQL */*

```
DECLARE CONTINUE HANDLER FOR NOT FOUND  
    SET v_ultima_fila=1;
```



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

```
DECLARE CONTINUE HANDLER FOR 1329  
    SET v_ultima_fila=1;
```



EJEMPLO CONDICIÓN DEFINIDA POR USUARIO

```
4 • DROP PROCEDURE IF EXISTS handler_funcPreda;
5 DELIMITER //
6 • CREATE PROCEDURE handler_funcPreda ()
7 BEGIN
8
9     create table alumnos (id int primary key, nombre varchar(50));
10
11 END //
12 DELIMITER ;
13 • call handler_funcPreda(); -- Saltará el error 1050 que habrá que tratar
14
15
16 • DROP PROCEDURE IF EXISTS handler_funcPredb;
17 DELIMITER //
18 • CREATE PROCEDURE handler_funcPredb ()
19 BEGIN
20     declare tabla_Existe condition for 1050;
21     declare exit handler for tabla_Existe
22         select ('La tabla que intentas crear ya existe') as 'Aviso de error';
23     create table alumnos (id int primary key, nombre varchar(50));
24
25 END //
26 DELIMITER ;
27 • call handler_funcPredb();
```

<	
Result Grid	Filter Rows: <input type="text"/>
Export: 	Wrap Cell Content: 
Aviso de error	
▶ La tabla que intentas crear ya existe	



OTRAS CUESTIONES




ORDEN DE ACTUACIÓN DEL MANEJADOR

- Ante varias posibilidades de ejecutarse un manejador ¿Quién se ejecuta?
- Siempre el manejador asociado al error MySQL en primer lugar. Si éste no estuviera declarado y definido entonces se ejecutaría el manejador asociado al código SQLSTATE. En último lugar el manejador asociado al manejador SQLEXCEPTION.
- El orden de actuación de los manejadores puede facilitarnos una forma de trabajo en nuestros programas. Los errores de código MySQL más habituales podrán ser tratados en manejadores específicos y aquéllos que no sean considerados podrán ser atrapados en un manejador SQLEXCEPTION



EJEMPLO



```
3 • use curso;
4 • DROP PROCEDURE IF EXISTS handler_orden;
5 DELIMITER //
6 • CREATE PROCEDURE handler_orden ()
7 BEGIN
8     DECLARE EXIT HANDLER FOR SQLEXCEPTION
9         SELECT 'Ocurrió un error. Fin del programa';
10    declare exit handler for 1062
11        select 'Clave repetida. Código Mysql 1062';
12    declare exit handler for sqlstate '23000'
13        Select 'Ocurrió error SQL State 23000'; -- Error: SQLSTATE[23000]: Integrity constraint violation:
14    insert into departamentos values(10, 'otro', 'madrid');
15
16 END //
17 DELIMITER ;
18 • call handler_orden();
19
20
21
```

<	
Result Grid	Filter Rows: <input type="text"/>
Export:  Wrap Cell Content: 	
Clave repetida. Código Mysql 1062	
▶ Clave repetida. Código Mysql 1062	



EJEMPLO

```
3 • use curso;
4 • DROP PROCEDURE IF EXISTS handler_orden;
5 DELIMITER //
6 • CREATE PROCEDURE handler_orden ()
7 BEGIN
8     DECLARE EXIT HANDLER FOR SQLEXCEPTION
9         SELECT 'Ocurrió un error. Fin del programa';
10 /*declare exit handler for 1062
11     select 'Clave repetida. Código Mysql 1062';*/
12     declare exit handler for sqlstate '23000'
13         select 'Ocurrió error SQL State 23000'; -- Error: SQLSTATE[23000]: Integrity constraint violation:
14     insert into departamentos values(10, 'otro', 'madrid');
15
16 END //
17 DELIMITER ;
18 • call handler_orden();
19
20
21
```

<	
Result Grid	Filter Rows: <input type="text"/>
Export:  Wrap Cell Content: 	
Ocurrió error SQL State 23000	
▶	Ocurrió error SQL State 23000



EJEMPLO

```
1 • use curso;
2 • DROP PROCEDURE IF EXISTS handler_orden2;
3 DELIMITER //
4 • CREATE PROCEDURE handler_orden2 ()
5 BEGIN
6     DECLARE EXIT HANDLER FOR SQLEXCEPTION
7         SELECT 'Ocurrió un error. Fin del programa';
8     declare exit handler for 1062
9         select 'Clave repetida. Código Mysql 1062';
10    declare exit handler for sqlstate '23000'
11        Select 'Ocurrió error SQL State 23000'; -- Error: SQLSTATE[23000]: Integrity constraint violation:
12    BEGIN
13        insert into departamentos values(10, 'otro', 'madrid');
14    END;
15 END //
16 DELIMITER ;
17 • call handler_orden2();
18
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	Clave repetida. Código Mysql 1062
▶	Clave repetida. Código Mysql 1062



EJEMPLO

```
19 • DROP PROCEDURE IF EXISTS handler_orden3;
20 DELIMITER //
21 • CREATE PROCEDURE handler_orden3 () -- Si se atrapa en un bloque interno, ya no se propaga a la del bloque superior externo
22 BEGIN
23     DECLARE EXIT HANDLER FOR SQLEXCEPTION
24         SELECT 'Ocurrió un error. Fin del programa';
25     declare exit handler for 1062
26         select 'Clave repetida. Código Mysql 1062';
27     declare exit handler for sqlstate '23000'
28         Select 'Ocurrió error SQL State 23000'; -- Error: SQLSTATE[23000]: Integrity constraint violation:
29     BEGIN
30         DECLARE EXIT HANDLER FOR SQLEXCEPTION
31             SELECT 'Bloque interno. Ocurrió un error. Fin del programa';
32         declare exit handler for 1062
33             select 'Bloque interno.Clave repetida. Código Mysql 1062';
34         declare exit handler for sqlstate '23000'
35             Select 'Bloque interno. Ocurrió error SQL State 23000'; -- Error: SQLSTATE[23000]: Integrity constraint violation:
36
37         insert into departamentos values(10, 'otro', 'madrid');
38     END;
39 END //
40 DELIMITER ;
41 • call handler_orden3();
```

<	
Result Grid	Filter Rows: <input type="text"/>
Export:  Wrap Cell Content: 	
	Bloque interno.Clave repetida. Código Mysql 1062
▶	Bloque interno.Clave repetida. Código Mysql 1062



Invoice

Invoice_id
Customer_id
Order_id
Product_id
Date_time
Status
Total
Remark

UT10. SQL MODO PROGRAMACIÓN

Módulo: BASES DE DATOS

Curso 2022/2023. 1º DAM

Ruth Lospitao Ruiz

