Bases de Datos

Unidad 8:

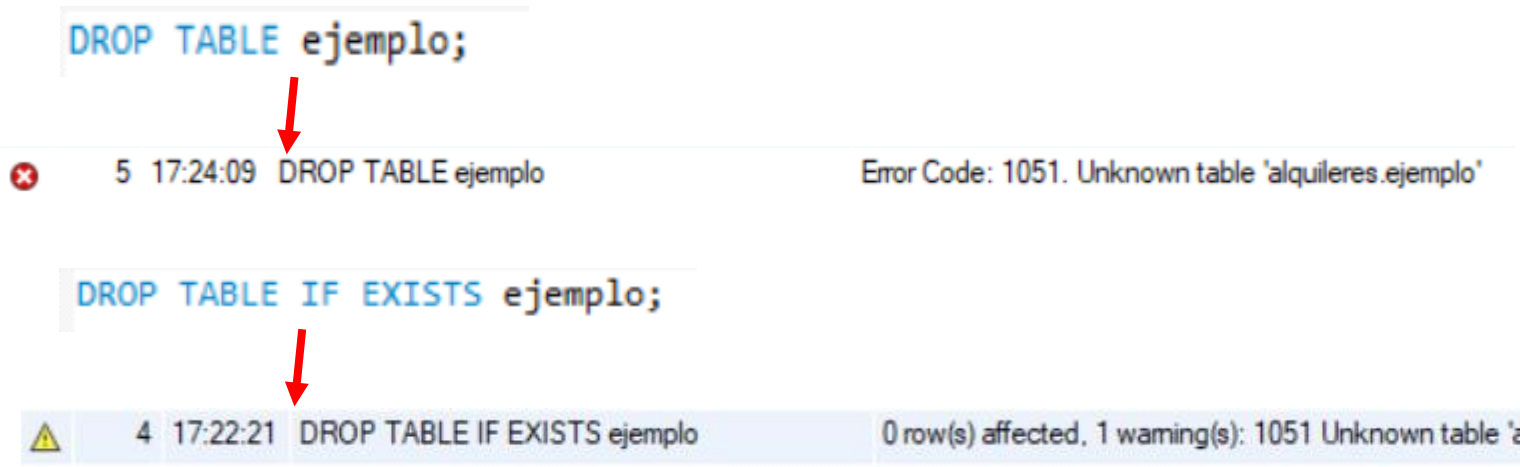
Programación de bases de datos

Sesión 8

Los mensajes de error

Cuando una instrucción se envía a ejecución al servidor, puede ocurrir:

- Que la ejecute correctamente (no hay error)
- Que genere un error. No se ha podido ejecutar.
- Que genere un riesgo o *warning*.



Los mensajes de error

Si una instrucción de una rutina almacenada (procedimiento, función o trigger) **produce un ERROR**:

- Se produce una excepción y se aborta la ejecución de la rutina.
- Las instrucciones anteriores a la que produjo el error quedan hechas.

Si una instrucción de una rutina almacenada (procedimiento, función o trigger) **produce un WARNING**:

- Continúa la ejecución de la rutina normalmente.

CÓDIGOS DE ERROR

Siempre que el servidor ejecuta una instrucción, devuelve al cliente un **código de error.** Además, se envía un mensaje con información sobre el error.

```
mysql> DROP TABLE ejemplo;
ERROR 1051 (42S02): Unknown table 'alquileres.ejemplo'
```

Se ha devuelto el código de error **1051** y un mensaje descriptivo del error.

Cuando una instrucción se ha ejecutado sin error, el servidor responde con **código de error cero.**

```
mysql> UPDATE contratos SET kini=kini+1 WHERE numcontrato=1;
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

La instrucción se ejecutó correctamente. El código de error recibido fue 0. El servidor envío un mensaje descriptivo de lo realizado.

CÓDIGOS DE ERROR Y ESTADOS DE ERROR SQL

A un código de error le corresponde un solo estado de error (**SQLSTATE**). A un estado de error pueden corresponder varios códigos de error, en su caso, esos códigos corresponden a errores distintos pero de un mismo tipo.

Por ejemplo, en la ejecución:

```
mysql> DROP TABLE ejemplo;
ERROR 1051 (42S02): Unknown table 'alquileres.ejemplo'
```

Vemos que él código de error **1051** se corresponde con **SQLSTATE '42S02'.**

Cada SGBD usa sus propios códigos de error. Los **SQLSTATE** son **comunes** a todos los SGBD relacionales.

Un código de error es un número entero. Un SQLSTATE es un código de cinco caracteres.

CÓDIGOS DE ERROR Y ESTADOS DE ERROR SQL

Ejemplo de unos pocos códigos de error MySQL, con el SQLSTATE y mensaje asociados.

- Error: 1048 SQLSTATE: 23000 (ER_BAD_NULL_ERROR)

  Mensaje: La columna '%s' no puede ser nula

- Error: 1049 SQLSTATE: 42000 (ER_BAD_DB_ERROR)

  Mensaje: Base de datos desconocida '%s'

- Error: 1050 SQLSTATE: 42S01 (ER_TABLE_EXISTS_ERROR)

  Mensaje: La tabla '%s' ya existe

- Error: 1051 SQLSTATE: 42S02 (ER_BAD_TABLE_ERROR)

  Mensaje: Tabla '%s' desconocida

- Error: 1052 SQLSTATE: 23000 (ER_NON_UNIQ_ERROR)

  Mensaje: La columna: '%s' en %s es ambigua

CAMBIAR IDIOMA DE MENSAJES DE ERROR

Puedes cambiar el idioma en que el servidor devuelve los mensajes de error, por ejemplo, a español. Para ello tienes que editar el fichero:

C:\ProgramData\MySQL\MySQL Server 5.7**my.ini**

Y añadir detrás del inicio del bloque [mysqld]:

language="spanish"

Si reinicias el servidor, desde ahora todos los mensajes de error los verás en español. Podrías incluso modificar el texto de los mensajes editando el archivo:

C:\Program Files\MySQL\MySQL Server 5.7\share**errmsg-utf8.txt**

7.- Manipuladores de error

MySQL permite usar manipuladores de errores o **handlers** que sirven para indicar como debe responder el servidor MySQL, en procedimientos y funciones, a situaciones de error.

Un **manipulador de error** tiene un **nombre** y una sentencia que será ejecutada tras ocurrir una determinada condición de **error o un warning**.

Con los manipuladores de errores intentaremos controlar errores para que el procedimiento, la función o el trigger termine o continúe ejecutándose de una forma adecuada.

Ya hemos visto anteriormente que:

- Cuando se produce un error en la ejecución de una instrucción **MySQL,** el servidor devuelve la descripción y dos códigos diferentes para el error:
  - **MySQL error**: un código de error (numérico) que es exclusivo del sistema gestor de base de datos.
  - **SQLSTATE**: una cadena de cinco caracteres que está estandarizada, es decir, es independiente del sistema gestor de base de datos. Estemos trabajando con MySQL, Oracle o SQL Server, los SQLSTATE coinciden.

- Normalmente, todo código de error MySQL tiene asociado un SQLSTATE.
- No todos los códigos de error MySQL tienen su equivalente en código SQLSTATE.
- **HY000** es un código SQLSTATE para propósitos generales que devuelve MySQL cuando su código de error no tiene asociado un código SQLSTATE.

Como funciona un handler o manipulador de error

- Los HANDLER o manipuladores de error son invocados al producirse un error o una condición de error que hayamos declarado con anterioridad

- Existen tres métodos de definir o declarar un handler:

  - Con **código de error** de MySQL
  - Con **código SQLSTATE** ANSI (standard para todos los SGBD relacionales).
  - Como **nombre de condición**, por ejemplo SQLWARNING, SQLEXCEPTION Y NOT FOUND

Existen numerosos códigos de error y sus equivalentes SQLSTATE, que podemos consultar en la documentación oficial de MySQL.

Declaración de un handler

Los **handlers** se declaran después de las variables y los cursores. La sintaxis para declararlos es la siguiente:

DECLARE tipo_de_handler HANDLER FOR condicion_del_handler

Donde **tipo_de_handler** puede ser:
- **CONTINUE**: indica que la ejecución de la rutina debe seguir.
- **EXIT**: indica que la ejecución de la rutina debe culminar.

En **condicion_del_handler** podemos usar alguno de los siguientes valores:
- **SQLSTATE**: códigos de errores independiente de la plataforma.
- **MySQL error**: códigos de errores exclusivos de **MySQL**.
- **SQLWARNING**: abreviación para los códigos **SQLSTATE** que comienzan con 01.
- **NOT FOUND**: abreviación para los códigos **SQLSTATE** que comienzan con 02.
- **SQLEXCEPTION**: abreviación para el resto de códigos **SQLSTATE**.
- nombre de una condición declarada previamente.

7.- Manipuladores de error

Ejemplo 1: Control de finalización de un cursor mediante un handler asociado a la condición de error NOT FOUND o al valor de estado ‘02000’

```
CREATE PROCEDURE ejemplo()
BEGIN
  -- declaramos las variables antes de los cursores
  DECLARE bContinuar BOOLEAN DEFAULT true;
  DECLARE a CHAR(20);
  -- declaramos un cursor con su respectiva consulta
  DECLARE cursor1 CURSOR FOR SELECT nombre FROM usuarios;
 -- declaramos un manejador, bcontinuar sera false cuando se produzca la condicion SQLSTATE 02000,
  DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET bContinuar = false;
  OPEN cursor1;-- se abre el cursor declarado
    WHILE bContinuar DO
              FETCH cursor1 INTO a;
      -- mientras no lleguemos el final de los registros, continuamos
              IF bContinuar THEN
                        -- aqui lo que tengas que hacer ..
              END IF;
END WHILE;
 CLOSE cursor1; -- cerramos el cursor
END//
```

7.- Manipuladores de error

Ejemplo 2: En el procedimiento **votar** de la base de datos **concursomusica,** se recibe el usuario que vota y el número de canción votada. El procedimiento inserta el voto, contabiliza el voto del usuario y contabiliza el voto a la canción. El voto corresponde al día actual. El proceso se debe realizar dentro de una transacción y se debe controlar que no se duplique la PK en votos y que la canción y el usuario existan.

```
CREATE PROCEDURE votar(IN usu VARCHAR(10), IN numc integer)
BEGIN
declare errorfk tinyint default 0;
declare errorpk tinyint default 0;
DECLARE CONTINUE HANDLER FOR 1452 set errorfk=1;
DECLARE CONTINUE HANDLER FOR 1062 set errorpk=1;
start transaction;
update canciones set total_votos=total_votos+1 where numCancion=numc;
update usuarios set numerovotos=numerovotos+1 where user=usu;
insert into votos (fecha,usuario,cancion) values (curdate(), usu, numc);
if errorfk=0 and errorpk=0 then
                commit;
else
                rollback;
                 if errorfk=1 then select 'error, no existe usuario o cancion';
                else select 'error, el susuario ya ha votado en la fecha';
                end if;
end if;
end//
```

7.- Manipuladores de error

Ejemplo 3: El procedimiento anterior realizado con la condición EXIT HANDLER.

```
CREATE PROCEDURE votar2(IN usu VARCHAR(10), IN numc integer)
BEGIN
DECLARE EXIT HANDLER FOR 1452 BEGIN
                rollback;
                select 'error, no existe usuario o cancion';
end;
DECLARE EXIT HANDLER FOR 1062 BEGIN
                rollback;
                select 'error, el usuario ya ha votado en la fecha';
end;
start transaction;
update canciones set total_votos=total_votos+1 where numCancion=numc;
update usuarios set numvotos=numvotos+1 where user=usu;
insert into votos (fecha,usuario,cancion) values (curdate(), usu, numc);
 commit;
select 'OK, se ha registrado el voto';
End //
```

7.- Manipuladores de error

Ejemplo 4: Realizar un procedimiento que obtiene los datos de los dos automóviles más caros de cada marca. Hay que realizarlo con cursores. Se controlará la finalización del proceso cuando el cursor ya no tenga datos que leer.

```
CREATE PROCEDURE ejemplo4()   BEGIN
declare existe_fila boolean  default true;
declare mar varchar(15);
declare curMarcas cursor for select distinct marca from automoviles;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET existe_fila=false;
DROP TEMPORARY TABLE IF EXISTS informe;
create temporary table  informe (
                marca varchar(15),
                matricula char(7),
                modelo varchar(20),
                precio numeric(5,2));
open curMarcas;
fetch curMarcas into mar;
while existe_fila do
                insert into informe select mar,matricula,modelo,precio from automoviles where
marca=mar order by precio desc limit 2;
                fetch curMarcas into mar;
end while;
select * from informe;
close curMarcas;
END//
```

7.- Manipuladores de error

En el anterior ejemplo, en lugar de

```
DECLARE CONTINUE HANDLER FOR NOT FOUND SET existe_fila=false;
```

Podríamos haber usado:

```
DECLARE CONTINUE HANDLER FOR 1329 SET existe_fila=false;
```

O también

```
DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET existe_fila=false;
```

O incluso, **declarar una condición de error** y usarla en la declaración del handler.

```
DECLARE cursor_finalizado CONDITION FOR 1329;
DECLARE CONTINUE HANDLER FOR cursor_finalizado SET existe_fila=false;
```

LEVANTAR EXCEPCIONES

MySQL permite provocar o levantar una excepción dentro de un procedimiento, función o trigger. Al producirse una excepción se podrá producir la finalización de la rutina.

La instrucción para levantar una excepción o un error es **SIGNAL** cuya sintaxis es la siguiente:

```
SIGNAL SQLSTATE valor_estado
    SET MESSAGE_TEXT = 'mensaje de error', MYSQL_ERRNO = numero_error;
```

7.- Manipuladores de error

Ejemplo: Realiza un procedimiento insertaVoto en la base de datos concursoMusica que recibe los datos de un voto a insertar y:
-Si el usuario ya ha dado 10 votos, no permite la inserción, rechazándola con un mensaje de error apropiado. Si aún no ha dado esos 10 votos, se añade el nuevo voto en la tabla votos con la fecha actual. Si se ha podido añadir el voto, incrementa el total de votos dados por el usuario y el total de votos recibidos por la canción.

```
CREATE PROCEDURE insertaVoto(IN usu VARCHAR(15),IN nc INT)
BEGIN
DECLARE nv INT;
   SELECT numvotos INTO nv FROM usuarios WHERE user=;
   IF nv>=10 THEN
      SIGNAL SQLSTATE '47000'  SET MESSAGE_TEXT =‘El usario ya ha dado el límite
de 10 votos', MYSQL_ERRNO = 1800;
  ELSE
      INSERT INTO votos (usuario,cancion,fecha) VALUES (usu, nc, curdate());
      UPDATE usuarios SET numvotos=numvotos+1 WHERE user=usu;
      UPDATE canciones SET total_votos = total_votos+1 WHERE numcancion=nc;
 END IF;
END
```