# 

[**Eventos**](#_eqi8i3j74qae) **1**

[**Triggers**](#_nh0qez9wvezq) **2**

[**Excepciones**](#_hu6he55au8q5) **2**

[**Expresiones regulares**](#_2d1el56q55vz) **2**

[**IF … ELSE**](#_v4r5dqehquv6) **5**

[**Case**](#_vy2y0i1uu5bl) **6**

[Sencillo](#_wh4m868hdsvf) 6

[De búsqueda](#_sz0qvmp7g0gn) 6

[En un select (Sencillo)](#_zicpfclvye5) 6

[En un select (Búsqueda)](#_y689s064ors) 6

[**5. Sentencia REPEAT – Bucles condicionales I**](#_i3jobnq8ep7k) **7**

[**6. Sentencia WHILE – Bucles condicionales II**](#_wi04ge1iwnhe) **7**

[**Sintaxis de los Manejadores**](#_219l5u3k9j79) **8**

# 

# Eventos

CREATE

[DEFINER = *user*]

EVENT

[IF NOT EXISTS]

*event\_name*

ON SCHEDULE *schedule*

[ON COMPLETION [NOT] PRESERVE]

[ENABLE | DISABLE | DISABLE ON SLAVE]

[COMMENT '*string*']

DO *event\_body*;

*schedule*: {

AT *timestamp* [+ INTERVAL *interval*] ...

| EVERY *interval*

[STARTS *timestamp* [+ INTERVAL *interval*] ...]

[ENDS *timestamp* [+ INTERVAL *interval*] ...]

}

*interval*:

*quantity* {YEAR | QUARTER | MONTH | DAY | HOUR | MINUTE |

WEEK | SECOND | YEAR\_MONTH | DAY\_HOUR | DAY\_MINUTE |

DAY\_SECOND | HOUR\_MINUTE | HOUR\_SECOND | MINUTE\_SECOND}

* AT ***timestamp*** se utiliza para un evento de una sola vez.
* EVERY 1 HOUR Cada hora

# Triggers

CREATE

[DEFINER = *user*]

TRIGGER [IF NOT EXISTS] *trigger\_name*

*trigger\_time* *trigger\_event*

ON *tbl\_name* FOR EACH ROW

[*trigger\_order*]

*trigger\_body*

*trigger\_time*: { BEFORE | AFTER }

*trigger\_event*: { INSERT | UPDATE | DELETE }

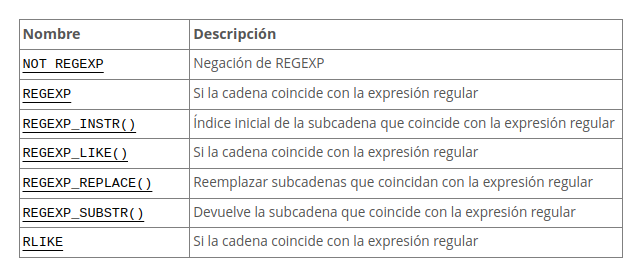
*trigger\_order*: { FOLLOWS | PRECEDES } *other\_trigger\_name*

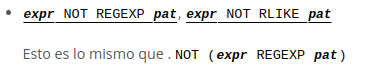
# Excepciones

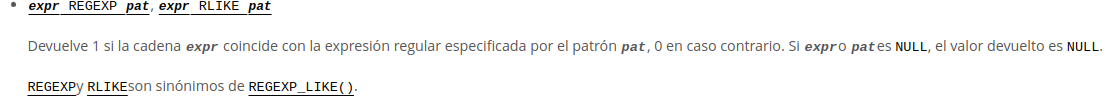
SIGNAL SQLSTATE '45000'

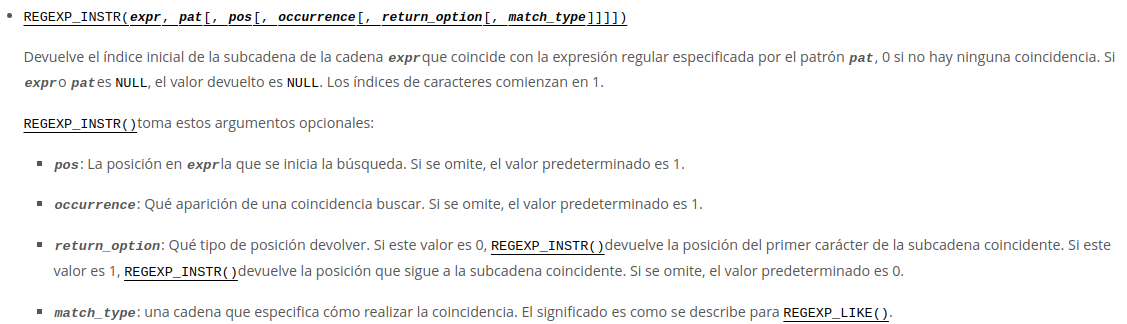
SET MESSAGE\_TEXT = 'An error occurred';

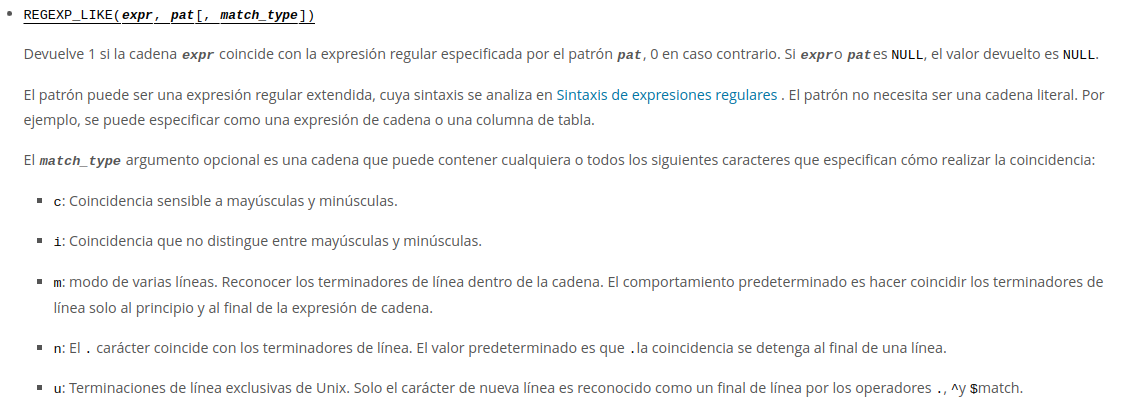
# Expresiones regulares

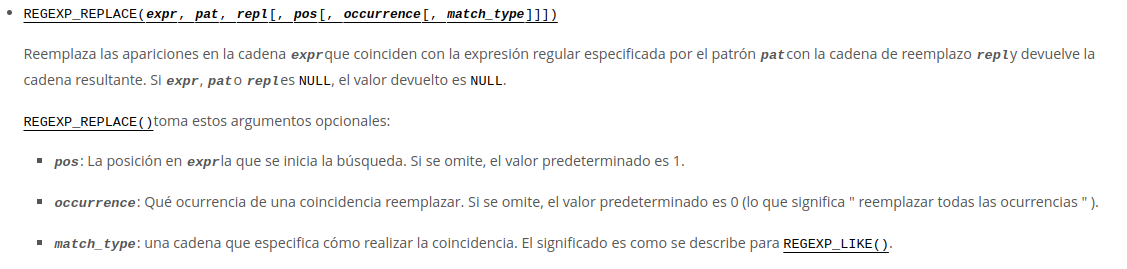


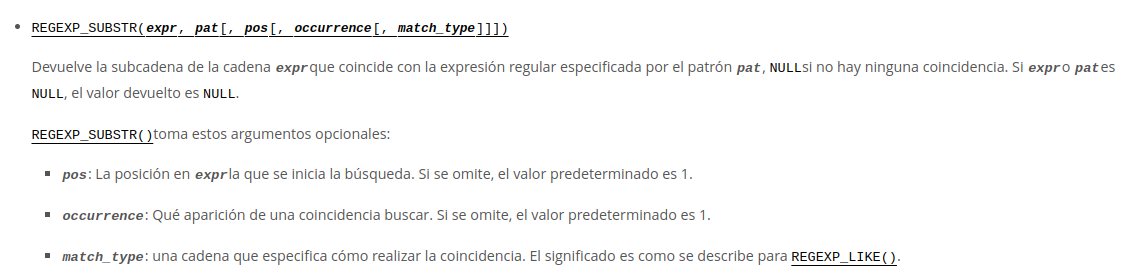


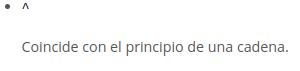


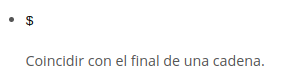


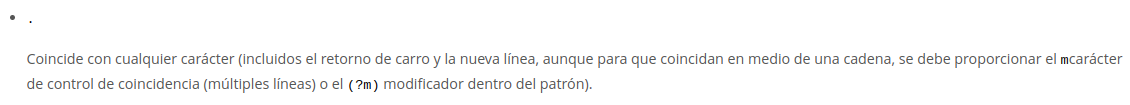


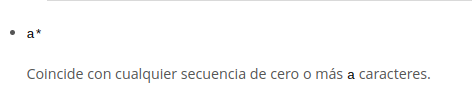


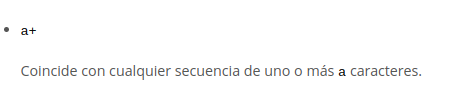


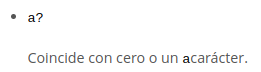


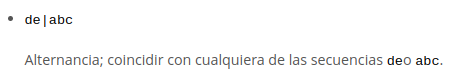


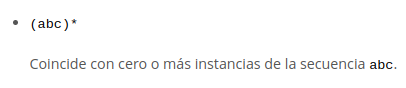


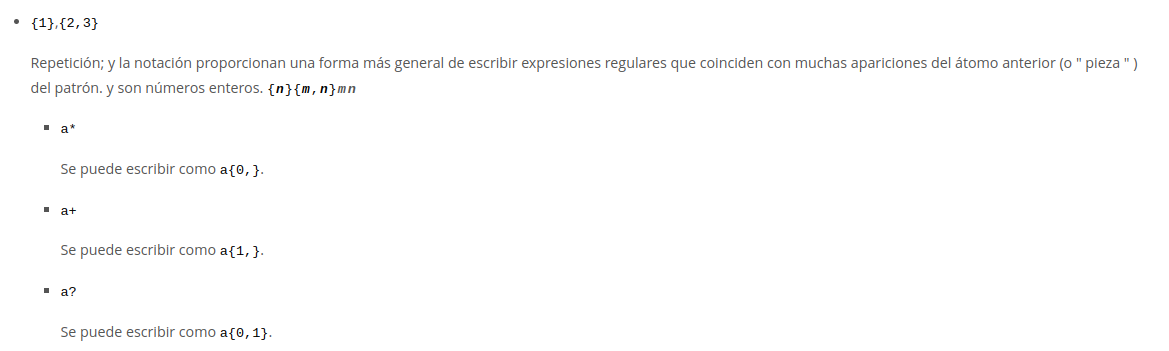


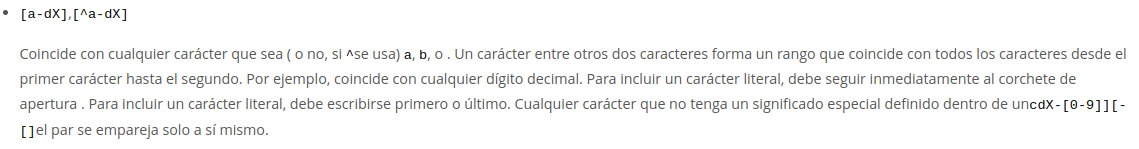


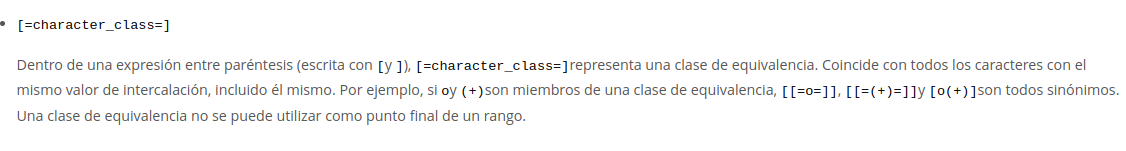


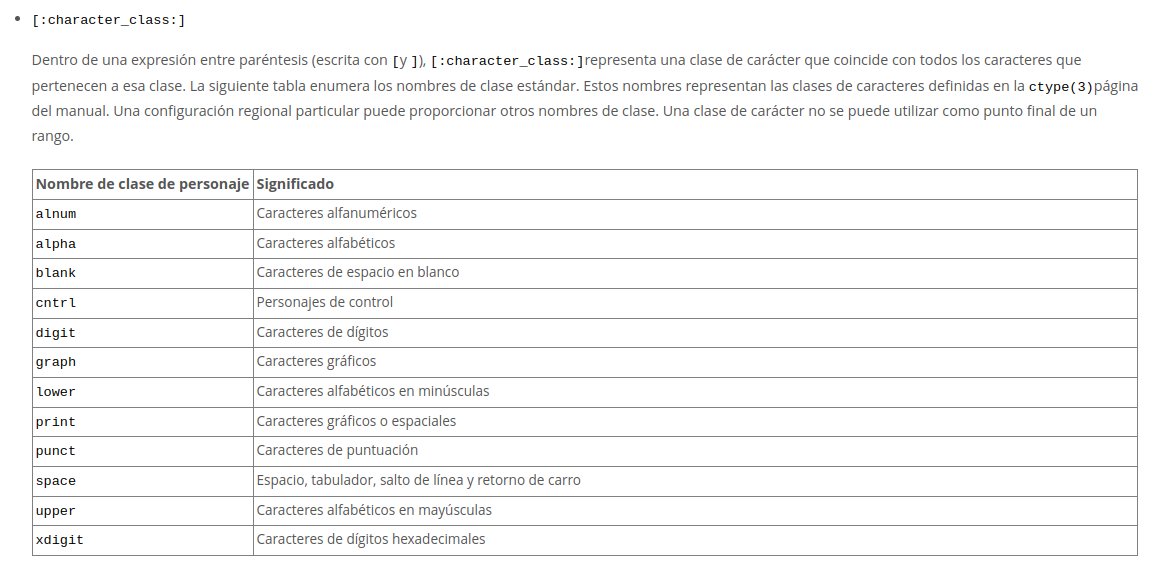












# 

# IF … ELSE

IF expresion\_booleana THEN

[BEGIN]

Sentencias SQL|Sentencias de control

[END]

[ELSE IF expresion\_booleana THEN

[BEGIN]

Sentencias SQL|Sentencias de control

[END]

]

[ELSE

[BEGIN]

Sentencias SQL|Sentencias de control

[END]

]

END IF [ ] /\* habrá 1 línea 'end if' por cada aparición de 'IF' (también else **if**)\*/

# Case

## Sencillo

CASE expresion

WHEN caso1 THEN rtdo1 - - cada rtdo. Puede ser un bloque de programa

WHEN caso2 THEN rtdo2

. . .

[ELSE rtdoN]

END CASE;

## De búsqueda

CASE

WHEN expres\_bool1 THEN rtdo1

WHEN expres\_bool2 THEN rtdo2

. . .

[ELSE rtdoN]

END CASE;

## En un select (Sencillo)

SELECT numem, nomem, ape1em, ape2em, - - leemos cuatro columnas de la tabla

CASE numhiem

WHEN 0 THEN ‘Cero hijos’

WHEN 1 THEN ‘Un hijo’

WHEN 2 THEN ‘Dos hijos’

. . .

WHEN 5 THEN ‘Cinco hijos’

ELSE ‘Mas de cinco hijos’

END as NUM\_HIJOS,

salarem, comisem - - leemos otras dos columnas de la tabla

FROM empleados;

## En un select (Búsqueda)

SELECT numem, nomem, ape1em, ape2em,

CASE

WHEN numhiem= 0 THEN ‘Cero hijos’

WHEN numhiem <= 3 THEN ‘Entre 1 y 3 hijos’

ELSE ‘Mas de tres hijos’

END AS NUM\_HIJOS,

salarem, comisem

FROM empleados

# 5. Sentencia REPEAT – Bucles condicionales I

La sentencia o conjunto de sentencias que aparezcan dentro del cuerpo de esta instrucción se repetirá **hasta que** se cumpla una condición.

Sintaxis

REPEAT

[BEGIN]

sentencias sql | sentencias de control

[END]

UNTIL condicion

END REPEAT;

# 

# 6. Sentencia WHILE – Bucles condicionales II

La sentencia o conjunto de sentencias que aparezcan dentro del cuerpo de esta instrucción se repetirá **mientras** se cumpla una condición.

Sintaxis

WHILE condicion DO

[BEGIN]

sentencias sql | sentencias de control

[END]

END WHILE

**Ejemplo//**

Duplicamos el precio de los productos hasta que la media del precio de dichos productos sea superior a 100.

Utilizando REPEAT

REPEAT

BEGIN

UPDATE productos

SET precio\_unidad = precio\_unidad\*2;

END

UNTIL (SELECT avg(precio\_unidad) FROM productos) >=100

END REPEAT

Utilizando WHILE

WHILE (SELECT avg(precio\_unidad) FROM productos) <100 DO

BEGIN

UPDATE productos

SET precio\_unidad = precio\_unidad\*2

END

END WHILE

# Sintaxis de los Manejadores

DECLARE {CONTINUE | EXIT} HANDLER FOR

{SQLSTATE cod\_estado\_sql | MySQL codigo\_error | nombre\_error\_usuario}

acciones\_error

CONTINUE | EXIT ==> Determina el tipo de manejador:

EXIT: Después de producirse el error saldremos del bloque de código donde se ha producido

CONTINUE: Después del error, continúa la ejecución por la siguiente instrucción a la que provocó el error.

SQLSTATE cod\_estado\_sql | MySQL codigo\_error | nombre\_error\_usuario ==>

Determina el tipo de error que activará el manejador: Estado SQL, error MySQL o error definido por el usuario.

**Ejemplo//**

Procedimiento que inserta un empleado nuevo, en caso de que la clave primaria esté duplicada, avisará e insertará en la tabla control\_errores una fila con el error producido y la fecha en la que se ha producido.

Supongamos que la estructura de control\_errores es:

control\_errores (cod\_control(INT PK), fec\_error (DATETIME), mensaje\_error (VARCHAR(100))

DELIMITER $$

CREATE PROCEDURE inserta\_empleado (cod\_emp INT, nombre VARCHAR(100), ….)

MODIFIES SQL DATA

BEGIN

DECLARE EXIT HANDLER FOR ‘1062’ | SQLSTATE ‘23000’

BEGIN

DECLARE control INT;

SET control = (select max(cod\_control)+1 FROM control\_errores);

INSERT INTO control\_errores

values (control,

current\_date, 'inserción clave duplicada en empleados');

END;

INSERT INTO empleados (cod\_emp, nombre, ….);

…