Instrucciones condicionales

IF-ELSE

La sintaxis completa de esta instrucción está en [este enlace](https://dev.mysql.com/doc/refman/8.0/en/if.html).

Fijarse que el END IF debe llevar punto y coma si a continuación le siguen nuevas instrucciones.

[Interfaz de usuario gráfica, Texto, Aplicación, Teams

Descripción generada automáticamente](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mod_BD_Prog_ControlFlujo_1.jpg)

Imagen obtenida de [https://dev.mysql.com](https://dev.mysql.com/)]

Su funcionamiento es el mismo que en cualquier lenguaje de programación.

No debemos confundir esta sentencia con unas funciones de control de flujo que incorpora Mysql y que pueden ser empleadas dentro de una sentencia SELECT.

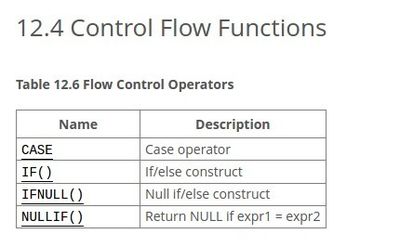
[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mod_BD_Prog_ControlFlujo_2.jpg)

Imagen obtenida de [https://dev.mysql.com](https://dev.mysql.com/)]

Por ejemplo, si queréis visualizar 'Desconocido' en un listado de atracciones cuando la fecha de inicio sea NULL, podrías poner la orden siguiente:

**SELECT** nombre, IFNULL(fecha\_inicio,'Desconocido')

**FROM** ATRACCIONES

**ORDER** **BY** nombre;

Disponéis de funciones:

* IF
* IFNULL
* CASE
* NULLIF

Vamos a crear un procedimiento de nombre artistas\_getPorNif que recupere los datos de un artista enviando un nif. En caso de que el artista no exista, devolverá la cadena 'NO ENCONTRADO' sin utilizar un parámetro de salida. La cadena que va a devolver tendrá el formato: artista>nif:nombre:apellidos-jefe>nif:nombre:apellidosEn caso de que no tenga jefe, el formato será: artista>nif:nombre:apellidos-jefe>SIN JEFE

Veremos que podemos emplear las funciones anteriores para simplificar el código...

Recordad que si un SELECT...INTO no se ejecuta, las variables mantienen sus valores por defecto...

USE CIRCO;

**DROP** **PROCEDURE** **IF** **EXISTS** artistas\_getPorNif;

**DELIMITER** $$

**CREATE** **PROCEDURE** artistas\_getPorNif(**IN** p\_nif char(9))

**COMMENT** 'Devuelve la información de un artista con el siguiente formato: artista>nif:nombre:apellidos-jefe>nombre:apellidos'

**BEGIN**

**DECLARE** v\_nifArt,v\_nifJefe char(9); *-- No ponemos default, por lo tanto el valor por defecto es NULL*

**DECLARE** v\_nombreArt,v\_nombreJefe varchar(45);

**DECLARE** v\_apellArt,v\_apellJefe varchar(100);

**DECLARE** v\_cadenaArtista varchar(400);

**DECLARE** v\_cadenaJefe varchar(400);

**SELECT** A.nif,A.nombre,A.apellidos,J.nif,J.nombre,J.apellidos

**INTO** v\_nifArt,v\_nombreArt,v\_apellArt,v\_nifJefe,v\_nombreJefe,v\_apellJefe

**FROM** ARTISTAS **as** A **LEFT** **OUTER** **JOIN** ARTISTAS **as** J

**ON** (A.nif\_jefe = J.nif)

**WHERE** A.nif=p\_nif;

**if** (v\_nifArt **IS** **NULL**) **THEN** *-- No hay un artista con ese NIF*

**SELECT** 'NO ENCONTRADO';

**ELSE**

**SET** v\_cadenaArtista=CONCAT('artista>',v\_nifArt,':',v\_nombreArt,':',v\_apellArt,'-jefe>');

**SET** v\_cadenaJefe = IFNULL(CONCAT(v\_nifJefe,':',v\_nombreJefe,':',v\_apellJefe),'SIN JEFE');

**SELECT** CONCAT(v\_cadenaArtista,v\_cadenaJefe);

**END** **IF**;

**END**$$

*Ejemplo de uso:*

**CALL** artistas\_getPorNif('11111111A'); *-- Artista con jefe*

**CALL** artistas\_getPorNif('22222222B'); *-- Artista sin jefe*

**CALL** artistas\_getPorNif('22222222X'); *-- No existe Artista*

Vamos a analizar un poco el código.

Lo primero es la sentencia que busca los datos del artista y el jefe. Esta sentencia es SQL que ya deberías saber.

Si se quisiera hacer de forma 'menos óptima', podríamos buscar primero los datos del artista y una vez guardado en una variable local el nif del jefe, comprobar si este era null para buscar o no los datos del jefe.

En la solución actual, hacemos uso del LEFT OUTER JOIN para que traiga todas las filas estén o no relacionadas con la tabla 'JEFES'.

* Una vez recuperados los datos en variables locales, comprobamos con un IF si el v\_nifArt es null. Fijarse que en la definición de la variable no tiene asignado valor por defecto y por tanto su valor por defecto es NULL. Si pusiéramos valor por defecto *, en el IF tendríamos que utilizar la comparación v\_nifArt='valor\_por\_defecto'.*
* En el caso de que encontrara al artista, tenemos que comprobar para crear la cadena de salida con el formato indicado, si tiene o no jefe. Podríamos emplear otra instrucción IF pero es mucho más cómodo emplear la función IFNULL. Si el jefe fuera null, la concatenación de cadenas daría NULL y por tanto dicha función devolvería el segundo parámetro (la cadena SIN JEFE).

**Nota aclaratoria:** Cuando se está desarrollando una aplicación, lo normal será que el control de que si existe un dato o no, lo programéis en el código de la aplicación y no en el procedimiento.

Es decir, en el ejercicio anterior, normalmente no se envían cadenas de tipo 'NO ENCONTRADO'. Se envían los datos con un SELECT:

**SELECT** A.nif,A.nombre,A.apellidos,J.nif,J.nombre,J.apellidos

**INTO** v\_nifArt,v\_nombreArt,v\_apellArt,v\_nifJefe,v\_nombreJefe,v\_apellJefe

**FROM** ARTISTAS **as** A **LEFT** **OUTER** **JOIN** ARTISTAS **as** J

**ON** (A.nif\_jefe = J.nif)

**WHERE** A.nif=p\_nif;

y es en la aplicación que recibe los datos en forma de conjunto de registros (RESULTSET) la que analiza los valores y muestra los mensajes de error al usuario...

Lo anterior sería correcto si el usuario va a llamar él mismo al procedimiento de forma directa.

**Nota aclaratoria:** Muchas veces, al escribir procedimientos almacenados necesitamos saber si una consulta ha encontrado algo o no.

En el ejemplo anterior lo supimos comprobando el valor variable v\_nifArt y mirando si tenía el valor por defecto NULL, queriendo esto decir que la sentencia SELECT no encontró ningún resultado.

No siempre vamos a querer guardar los datos del SELECT en variables locales y por lo tanto no tendríamos la opción anterior de comprobar si existe una determinada fila.

En otros gestores, como SQL SERVER, tenéis la [variable @@ROWCOUNT](https://docs.microsoft.com/es-es/sql/t-sql/functions/rowcount-transact-sql?view=sql-server-ver15) que devuelve el número de filas afectadas o leídas.

En Mysql, el equivalente a esa propiedad es la función [ROW\_COUNT()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_row-count), que devuelve el número de filas afectasdas, pero que sólo funciona con las sentencias UPDATE, DELETE y INSERT.

Existe otra que es la [función FOUND\_ROWS()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_found-rows) pero esta función en la versión 8.0 está deprecated. Quiero esto decir que será eliminada en versiones posteriores.

También podemos hacer uso de la sentencia SELECT COUNT(\*) dentro de un IF para determinar si existe o no un dato, antes de lanzar la orden SQL, de la forma:

**IF** (**SELECT** **COUNT**(\*) **FROM** TABLA **WHERE** condiciones)>0 **THEN**

*-- Código en caso de cumplir la condición*

**END** **IF**;

En este caso no queremos guardar nada de la consulta y sólo queremos saber si existe o no alguna fila (se podría emplear en el select cualquier expresión que devuelva una única fila y emplear cualquier tipo de comparación).

Pero si el uso que le queremos dar es el de recuperar información, si empleamos la forma anterior tendríamos que realizar otra vez la misma consulta, guardando la información que necesitemos.

En ese caso es mejor emplear el valor por defecto de la variabla que vamos a emplear para guardar el resultado de la consulta, para saber si existe o no (como hizimos antes):

**DECLARE** var varchar(50); *-- Valor por defecto null*

**SELECT** nombre

**FROM** TABLA

**WHERE** condiciones

**INTO** var;

**IF** (var **is** **null**) **THEN**

*-- Sabemos que no ha encontrado filas*

**END** **IF**;

CASE

END CASE debe llevar punto y coma si a continuación le siguen nuevas instrucciones.[Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico, Teams

Descripción generada automáticamente](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mod_BD_Prog_ControlFlujo_3.jpg)La instrucción CASE viene a ser como un IF-ELSEIF-ELSEIF-ELSEIF... pero de una forma mucho más legible.Veamos un ejemplo de uso.

Crea un procedimiento de nombre utilidades\_getDia al que se le pase como parámetro un número entre 1 y 7 y devuelva, en forma de parámetro de salida, Lunes, Martes, Miércoles,....

En caso de que el número esté fuera de rango debe devolver la cadena null.

USE CIRCO;

**DROP** **PROCEDURE** **IF** **EXISTS** utilidades\_getDia;

**DELIMITER** $$

**CREATE** **PROCEDURE** utilidades\_getDia(p\_numDia tinyint, **OUT** p\_nombreDia varchar(9))

**COMMENT** 'Devuelve en p\_nombreDia el nombre del día de la semana asociado al número indicado en p\_numDia siendo el 1 Lunes'

**BEGIN**

**CASE** p\_numDia

**WHEN** 1 **THEN**

**SET** p\_nombreDia = 'LUNES';

**WHEN** 2 **THEN**

**SET** p\_nombreDia = 'MARTES';

**WHEN** 3 **THEN**

**SET** p\_nombreDia = 'MIERCOLES';

**WHEN** 4 **THEN**

**SET** p\_nombreDia = 'JUEVES';

**WHEN** 5 **THEN**

**SET** p\_nombreDia = 'VIERNES';

**WHEN** 6 **THEN**

**SET** p\_nombreDia = 'SABADO';

**WHEN** 7 **THEN**

**SET** p\_nombreDia = 'DOMINGO';

**ELSE**

**SET** p\_nombreDia = **null**;

**END** **CASE**;

**END**$$

**DELIMITER** ;

Ejemplo de uso:

**CALL** utilidades\_getDia(3,@nombreDia); *-- Devuelve MIERCOLES*

**SELECT** @nombreDia;

**CALL** utilidades\_getDia(20,@nombreDia); *-- Devuelve NULL*

**SELECT** @nombreDia;

NOTA: Veremos más adelante que este es ejemplo en que sería más adecuado el uso de una función o no de un procedimiento almacenado.La sintaxis de CASE admite una variante cuando queremos tener un tipo de comparación que no sea 'igual', por ejemplo >= o <= o <>....

En ese caso, la cláusula CASE no lleva variable y es el las opciones WHEN donde se añade la variable junto con el tipo de comparación.

Aplicado a otro ejemplo donde se asigne 'SABADO','DOMINGO' o 'LABORABLE' en función del día:

**CASE**

**WHEN** p\_numDia>=1 **AND** p\_numDia<=5 **THEN**

**SET** p\_cadena = 'LABORABLE';

**WHEN** p\_numDia=6 **THEN**

**SET** p\_cadena = 'SABADO';

**WHEN** p\_numDia=7 **THEN**

**SET** p\_cadena = 'DOMINGO';

**END** **CASE**;

**Nota importante:** Si al ejecutar el procedimiento os aparece el error: 1339. Case not found for CASE statement

Esto es debido a que en Mysql el CASE **obliga** a que siempre exista una opción por la que deba ir. Es decir, la opción ELSE siempre debe de estar si existen valores de la variable que estáis usando, que no entren por ningún WHEN.

En ese caso, podéis poner: ELSE BEGIN END;

Existe una [función CASE](https://dev.mysql.com/doc/refman/8.0/en/control-flow-functions.html#operator_case) que se emplea para control de flujo y que tiene una sintaxis un poco diferente.

Este tipo de función se puede emplear en el interior de una sentencia SELECT de la forma:

**SELECT** nombre, **case** anhos

**when** 1 **then** 'Pequeño'

**when** 2 **then** 'Mediano'

**else** 'Grande'

**end** **as** anhos

**FROM** ANIMALES;