



# Bases de datos

Act02. Ut07 Construcción de  
guiones. Parte 2

Francisco José García Cutillas | 1FPGS\_DAM



## Índice

Parte 1. ....	3
Ejercicio 1 .....	3
Ejercicio 2 .....	3
Parte 2. ....	4
Ejercicio 3 .....	4
Ejercicio 4 .....	5
Ejercicio 5 .....	7
Ejercicio 6 .....	9
Ejercicio 7 .....	11
Parte 3 .....	13
Ejercicio 8 .....	13
Ejercicio 9 .....	14
Ejercicio 10 .....	15
Ejercicio 11 .....	17
Ejercicio 12 .....	18

## Parte 1.

### Funciones NO base de datos

#### Ejercicio 1

Crea un procedimiento al que se le pasen 2 enteros por parámetro y tengo 3 parámetros de salida que son números aleatorios generados que se encuentran entre el primer y segundo parámetro.

```
DELIMITER $$
drop procedure if exists p_genera_aleatorio; $$
create procedure p_genera_aleatorio (in min int, in max int, out al1 int, out al2 int, out al3 int)
begin
    select round((rand() * (max - min)) + min) into al1;
    select round((rand() * (max - min)) + min) into al2;
    select round((rand() * (max - min)) + min) into al3;
end; $$

call p_genera_aleatorio (-10, 30, @sal1, @sal2, @sal3);
select @sal1 aleatorio1, @sal2 aleatorio2, @sal3 aleatorio3;
```

	aleatorio1	aleatorio2	aleatorio3
▶	14	29	-6
	aleatorio1	aleatorio2	aleatorio3
▶	14	18	18
	aleatorio1	aleatorio2	aleatorio3
▶	4	16	-3

#### Ejercicio 2

Llamada al procedimiento anterior desde una función y devuelve la suma de los 3.

```
DELIMITER $$
drop function if exists f_aleatorio_suma; $$
create function f_aleatorio_suma (min int, max int) returns int
begin
    declare val1 int;
    declare val2 int;
    declare val3 int;

    call p_genera_aleatorio (min, max, val1, val2, val3);

    return (val1 + val2 + val3);
end; $$
```

```
select f_aleatorio_suma (5, 20) suma_aleatorios;
```

	suma_aleatorios
▶	26
	suma_aleatorios
▶	46
	suma_aleatorios
▶	28

## Parte 2.

Para realizar los siguientes apartados debes utilizar la Base de Datos se utilizará el script que se encuentra dentro de la unidad del aula virtual de la liga de baloncesto. Todos los ejercicios que se mencionan a continuación deben realizar la devolución de errores SIN UTILIZAR manejadores, es decir, sin utilizar HANDLER.

Tú debes ser el que detecte que control de errores debemos hacer.

### Ejercicio 3

Crea un procedimiento que dado el código de dos equipos nos devuelva en dos parámetros de salida el código y el nombre del equipo que tiene mejor puntuación. En caso de que lleven los mismos puntos debe devolver el primero alfabéticamente.

```
DELIMITER $$
drop procedure if exists p_mejor_puntuacion; $$
create procedure p_mejor_puntuacion (in cod_eq1 int, in cod_eq2 int, out codigo int, out nombre_eq varchar(200))
begin
    if (cod_eq1 not in (select id_equipo from equipo)) then
        set nombre_eq = concat('ERROR. El equipo ', cod_eq1, ' no se encuentra en la base de datos');
        set codigo = -1;
    elseif (cod_eq2 not in (select id_equipo from equipo)) then
        set nombre_eq = concat('ERROR. El equipo ', cod_eq2, ' no se encuentra en la base de datos');
        set codigo = -1;
    else
        select id_equipo, nombre into codigo, nombre_eq
        from equipo
        where id_equipo in (cod_eq1, cod_eq2)
        order by puntos desc, nombre
        limit 1;
    end if;
end; $$
```

```
call p_mejor_puntuacion (7, 5, @sal1, @sal2);
select @sal1 id_equipo, @sal2 nombre_equipo;
```

	id_equipo	nombre_equipo
▶	7	BALONCESTO MURCIA

```
call p_mejor_puntuacion (0, 5, @sal1, @sal2);
select @sal1 id_equipo, @sal2 nombre_equipo;
```

	id_equipo	nombre_equipo
►	-1	ERROR. El equipo 0 no se encuentra en la base de datos

```
call p_mejor_puntuacion (1, 15, @sal1, @sal2);
select @sal1 id_equipo, @sal2 nombre_equipo;
```

	id_equipo	nombre_equipo
►	-1	ERROR. El equipo 15 no se encuentra en la base de datos

## Ejercicio 4

Crea dos funciones que utilicen el procedimiento anterior, una que dado el código de los dos equipos devuelva el código del mejor equipo y otra que dado los dos códigos devuelva el nombre del mejor equipo (en términos de puntuación).

```
DELIMITER $$
drop function if exists f_mejor_codigo; $$
create function f_mejor_codigo (cod_eq1 int, cod_eq2 int) returns varchar (200)
begin
    declare salida_cod varchar(10);
    declare salida_nombre varchar (200);

    call p_mejor_puntuacion (cod_eq1, cod_eq2, salida_cod, salida_nombre);

    if (salida_cod = -1) then
        return salida_nombre;
    else
        return salida_cod;
    end if;
end; $$
```

```
select f_mejor_codigo (2,4) mejor_equipo;
```

	mejor_equipo
►	4

```
select f_mejor_codigo (2,9) mejor_equipo;
```

	mejor_equipo
►	ERROR. El equipo 9 no se encuentra en la base de datos

```
select f_mejor_codigo (11,7) mejor_equipo;
```

	mejor_equipo
►	ERROR. El equipo 11 no se encuentra en la base de datos

```

DELIMITER $$
drop function if exists f_mejor_nombre; $$
create function f_mejor_nombre (cod_eq1 int, cod_eq2 int) returns varchar (200)
begin
    declare salida_cod varchar(10);
    declare salida_nombre varchar (200);

    call p_mejor_puntuacion (cod_eq1, cod_eq2, salida_cod, salida_nombre);

    if (salida_cod = -1) then
        return salida_nombre;
    else
        return salida_nombre;
    end if;
end; $$

```

```
select f_mejor_nombre (1,5) mejor_equipo;
```

	mejor_equipo
►	GRAN CANARIA

```
select f_mejor_nombre (-5,5) mejor_equipo;
```

	mejor_equipo
►	ERROR. El equipo -5 no se encuentra en la base de datos

## Ejercicio 5

Crea una función que devuelva OK si todo ha ido bien o el mensaje de error y que realice lo siguiente: Pasando por parámetro el código del equipo local, código del equipo visitante, resultado del equipo local, resultado del equipo visitante y código de árbitro inserte los datos del partido correspondiente de la base de datos y actualice los puntos que tiene el equipo en su tabla de tal manera que si ha ganado el partido se suma 2 puntos y si ha empatado se suma 1 punto (la fecha del partido se considera la fecha actual).

```
DELIMITER $$
drop function if exists f_inserta_partido; $$
create function f_inserta_partido (cod_local int, cod_visitante int, res_loc int, res_vis int, cod_arbitro int) returns varchar (200)
> begin
    declare cont int;

    select count(*) into cont
    from partido
    where elocal = cod_local
    and evisitante = cod_visitante;

    if(cont > 0) then
        return 'ERROR. Estos equipos ya han jugado';
    elseif (cod_local not in (select id_equipo from equipo)) then
        return 'ERROR. El equipo local no existe';
    elseif (cod_visitante not in (select id_equipo from equipo)) then
        return 'ERROR. El equipo visitante no existe';
    elseif (res_loc < 0) then
        return 'ERROR. El resultado del local no puede ser negativo';
    elseif (res_vis < 0) then
        return 'ERROR. El resultado del visitante no puede ser negativo';
    elseif (cod_arbitro not in (select arbitro from partido)) then
        return 'ERROR. El árbitro no existe';
    else
        insert into partido values (cod_local, cod_visitante, concat(res_loc, '-', res_vis), curdate(), cod_arbitro);
    if(res_loc > res_vis) then
        update equipo
        set puntos = puntos + 2
        where id_equipo = cod_local;
    elseif (res_vis > res_loc) then
        update equipo
        set puntos = puntos + 2
        where id_equipo = cod_visitante;
    else
        update equipo
        set puntos = puntos + 1
        where id_equipo in (cod_local, cod_visitante);
    end if;
    return 'OK';
end if;
end; $$
```

```
select f_inserta_partido (2, 1, 100, 90, 5) info;
```

	info
▶	OK

```
select *
from partido;
```

	elocal	evisitante	resultado	fecha	arbitro
▶	2	1	100-90	2023-05-10 00:00:00	5

```
select *
from equipo;
```

id_equipo	nombre	ciudad	web	puntos
1	REGAL BARCELONA	BARCELONA	<a href="https://www.fcbarcelona.es/es/baloncesto">https://www.fcbarcelona.es/es/baloncesto</a>	10
2	REAL MADRID	MADRID	<a href="https://www.realmadrid.com/baloncesto">https://www.realmadrid.com/baloncesto</a>	11

```
select f_inserta_partido (8, 3, 100, 100, 3) info;
```

info
▶ ERROR. El equipo local no existe

```
select f_inserta_partido (7, 10, 100, 100, 3) info;
```

info
▶ ERROR. El equipo visitante no existe

```
select f_inserta_partido (7, 6, -100, 100, 3) info;
```

info
▶ ERROR. El resultado del local no puede ser negativo

```
select f_inserta_partido (7, 6, 100, -100, 3) info;
```

info
▶ ERROR. El resultado del visitante no puede ser negativo

```
select f_inserta_partido (7, 6, 100, 100, 13) info;
```

info
▶ ERROR. El árbitro no existe

```
select f_inserta_partido (7, 7, 100, 100, 3) info;
```

info
▶ ERROR. Un equipo no se puede enfrentar a sí mismo



## Ejercicio 6

Crea una función que reciba por parámetro el código de un equipo y devuelva en una única cadena el resultado de todos los partidos que ha jugado como local (separados por punto y coma). De tal manera que si se pasa como parámetro el código 2 su resultado sería: 90-90;65-145. La manera de recorrer los partidos debe ser OBLIGATORIAMENTE con un cursor.

```
DELIMITER $$
drop function if exists f_partidos_local; $$
create function f_partidos_local (codigo int) returns varchar (100)
begin
    declare res varchar (20);
    declare salida varchar (100);
    declare i int;
    declare cont int;
    declare existe int;
    declare partidos cursor for
        select resultado
        from partido
        where elocal = codigo;

    set i = 0;

    select count(*) into cont
        from partido
        where elocal = codigo;

    select count(*) into existe
        from equipo
        where id_equipo = codigo;

    if (existe = 0) then
        set salida = concat('El equipo ', codigo, ' no existe');
    elseif (cont = 0) then
        set salida = concat('El equipo ', codigo, ' no ha jugado ningún partido como local');
    else
        open partidos;
        partidos_loop: while (i < cont) do
            fetch partidos into res;
            if (salida is null) then
                set salida = res;
            else
                set salida = concat(salida,'; ', res);
            end if;
            set i = i + 1;
        end while partidos_loop;
        close partidos;
    end if;
    return salida;
end; $$
```

```
select f_partidos_local (3) res_partidos_local;
```

	res_partidos_local
▶	88-77; 91-88

```
select f_partidos_local (9) res_partidos_local;
```

	res_partidos_local
▶	El equipo 9 no existe

Ejemplo tras borrar el único partido como local del equipo 7:

```
select f_partidos_local (7) res_partidos_local;
```

	res_partidos_local
▶	El equipo 7 no ha jugado ningún partido como local

## Ejercicio 7

Crea una función que pasando un equipo y número de jugador debe actualizarlo y debe pasar a ser el capitán del equipo. Solo debe haber un capitán por equipo. Esta función debe devolver OK si todo ha ido correctamente y mensaje de error en todo caso.

```
DELIMITER $$
drop function if exists f_actualiza_jugador; $$
create function f_actualiza_jugador (cod_eq int, cod_jug int) returns varchar (100)
begin
    declare cont_eq int;
    declare cont_cap int;
    declare cap int;

    select count(*) into cont_eq
    from jugador
    where id_jugador = cod_jug
    and num_equipo = cod_eq;

    select count(*) into cont_cap
    from jugador
    where id_jugador = cod_jug
    and id_capitan = cod_jug;

    if (cod_eq not in (select id_equipo from equipo)) then
        return concat('El equipo ', cod_eq, ' no existe');
    elseif (cod_jug not in (select id_jugador from jugador)) then
        return concat('El jugador ', cod_jug, ' no existe');
    elseif (cont_eq = 0) then
        return concat('El jugador ', cod_jug, ' no juega en el equipo ', cod_eq);
    elseif (cont_cap = 1) then
        return concat('El jugador ', cod_jug, ' ya es capitán del equipo ', cod_eq);
    else
        update jugador
        set id_capitan = cod_jug
        where num_equipo = cod_eq;
    end if;
    return 'OK';
end; $$
```

```
select f_actualiza_jugador (1,8) info;
```

	info
►	OK

```
select *
from jugador
where num_equipo = 1;
```

id_jugador	nombre	apellido	puesto	id_capitan	fecha_alta	salario	num_equipo	altura
1	JUAN CARLOS	NAVARRO	ESCOLTA	8	2010-01-10	130000	1	NULL
8	VÍCTOR	SADA	BASE	8	2012-01-01	80000	1	1.9
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
select f_actualiza_jugador (1,8) info;
```

info
------

El jugador 8 ya es capitán del equipo 1
---

```
select f_actualiza_jugador (8,8) info;
```

info
------

El equipo 8 no existe
-----------------------

```
select f_actualiza_jugador (5,21) info;
```

info
------

El jugador 21 no existe
-------------------------

```
select f_actualiza_jugador (5,2) info;
```

info
------

El jugador 2 no juega en el equipo 5
--------------------------------------

## Parte 3

Para realizar los siguientes apartados debes utilizar la Base de Datos se utilizará el script que se encuentra dentro de la unidad en el aula virtual de la liga de baloncesto. Todos los ejercicios que se mencionan a continuación deben realizar la devolución de errores UTILIZANDO manejadores, es decir, debes utilizar HANDLER.

Tú debes ser el que detecte que control de errores debemos hacer.

### Ejercicio 8

Realiza una función denominada `f_total_salarios` que devuelva el total de los salarios del equipo de baloncesto cuyo código se pasa por parámetro (no es necesario utilizar ningún cursor).

Como hemos comentado en clase, en este caso no era necesario usar el Handler.

```
DELIMITER $$
drop function if exists f_total_salarios; $$
create function f_total_salarios (cod_eq int) returns decimal
> begin
    declare total decimal;
    select ifnull(sum(salario), '0') into total
      from jugador
     where num_equipo = cod_eq;

    return total;
end; $$
```

```
select f_total_salarios (3) total_salarios;
```

	total_salarios
▶	141000

```
select f_total_salarios (30) total_salarios;
```

	total_salarios
▶	0

## Ejercicio 9

Realiza una función denominada `f_abreviatura` que devuelva los tres primeros caracteres del apellido de cada jugador. Deben estar ordenados de tal manera que en primer lugar salgan los que más salario tienen y cada abreviatura debe estar separada por una coma. Ejemplo: NAV, REY, CAB, ...

```
DELIMITER $$
drop function if exists f_abreviatura; $$
create function f_abreviatura () returns varchar (500)
> begin
    declare salida varchar (500);
    declare temp varchar (20);
    declare final boolean;
    declare cursor_ab cursor for
        select substr(apellido, 1, 3)
            from jugador
            order by salario desc;
    declare continue handler for not found set final = 1;

    set final = 0;

    open cursor_ab;
>   apellidos_loop: loop
        fetch cursor_ab into temp;
>   if (final = 1) then
        set final = 0;
        close cursor_ab;
        leave apellidos_loop;
    else
>   if (salida is null) then
        set salida = temp;
    else
        set salida = concat(salida, ', ', temp);
-   end if;
-   end if;
-   end loop;
-   return salida;
end; $$
```

```
select f_abreviatura() apellido_abreviado;
|
| apellido_abreviado
|
| NAV, REY, CAB, LLU, ROJ, REY, CLA, SAD, TEL, SAN, SUÁ, SAV, HET, MAR, AGU, SOK
```

## Ejercicio 10

Realiza un procedimiento p\_recorta\_salario que recorte el salario un 20% a los empleados cuyo nombre empiece por la letra que recibe como parámetro.

```
DELIMITER $$
drop procedure if exists p_recorta_salario; $$
create procedure p_recorta_salario (in letra char)
begin
    declare cod_jugador int;
    declare in_nombre char;
    declare sal decimal;
    declare sal_recorte decimal;
    declare final boolean;
    declare recortes cursor for
        select id_jugador, substr(nombre, 1, 1), salario
            from jugador;
    declare continue handler for not found set final = 1;

    set final = 0;

    open recortes;
loop_jugadores: loop
    fetch recortes into cod_jugador, in_nombre, sal;
    if (final != 1) then
        if (in_nombre = letra) then
            set sal_recorte = sal - (sal * 0.2);
            update jugador
                set salario = sal_recorte
                where id_jugador = cod_jugador;
        end if;
    else
        set final = 1;
        leave loop_jugadores;
        close recortes;
    end if;
end loop;
end; $$
```

```
select substr(nombre, 1, 1) inicial, salario
    from jugador;
```

inicial	salario
C	60000
C	105000
F	120000
F	60000
J	104000
M	70000
O	41500
P	47000
R	51000
R	53000
S	100000
S	60000
S	99000
V	72000
V	64000
X	95000

```
call p_recorta_salario ('v');
```

inicial	salario
C	60000
C	105000
F	120000
F	60000
J	104000
M	70000
O	41500
P	47000
R	51000
R	53000
S	100000
S	60000
S	99000
V	57600
V	51200
X	95000



## Ejercicio 11

**Realiza un procedimiento que devuelva el total de los salarios del equipo que se pase por parámetro.**

En este caso, al igual que en el 8, no haría falta la comprobación con el Handler.

```
DELIMITER $$
drop procedure if exists p_total_salarios; $$
create procedure p_total_salarios (in cod_eq int, out total decimal)
> begin
```

```
    select ifnull(sum(salario), '0') into total
        from jugador
        where num_equipo = cod_eq;
```

```
end; $$
```

```
call p_total_salarios (5, @sal1);
select @sal1 total_salarios_equipo;
```

	total_salarios_equipo
▶	155000

## Ejercicio 12

Realiza una función que devuelva el nombre de todos los equipos y entre paréntesis el total de los salarios de sus jugadores. Deberás utilizar un cursor para recorrer los equipos (no hace falta para los jugadores) y utilizarás el procedimiento anterior para el total del salario. Todo debe devolverse en una única cadena ordenada por orden alfabético de equipo. El resultado puede tener el formato: Baloncesto Murcia(xxxx)-Cai Zaragoza(xxxx)-Caja Laboral(xxxx)-....

```
DELIMITER $$
drop function if exists f_salarios_cadena; $$
create function f_salarios_cadena () returns varchar (500)
begin
    declare salida varchar (500);
    declare cod_eq int;
    declare nom_eq varchar (30);
    declare total_salario decimal;
    declare final boolean;
    declare cur_equipos cursor for
        select id_equipo, nombre
        from equipos;
    declare continue handler for not found set final = 1;

    set final = 0;
    open cur_equipos;
loop_equipos: loop
    fetch cur_equipos into cod_eq, nom_eq;
    if (final = 1) then
        set final = 0;
        close cur_equipos;
        leave loop_equipos;
    else
        call p_total_salarios (cod_eq, total_salario);
        if (salida is null) then
            set salida = concat(nom_eq, '(', total_salario, ')');
        else
            set salida = concat(salida,'-',nom_eq, '(', total_salario, ')');
        end if;
    end if;
end loop;
return salida;
end; $$
```

```
select f_salarios_cadena() salarios_equipos;
```

salarios_equipos
BALONCESTO MURCIA(140500)-CAI ZARAGOZA(205000)-CAJA LABORAL(130000)-GRAN CANARIA(155000)-REAL MADRID(280000)-REGAL BARCELONA(210000)-VALENCIA BASKET(141000)