ENTREGA FINAL : CREACION DE UNA BASE DE DATOS

Micaela Flores

introducción:
Este proyecto se basa en la organización de datos de un club.
Nos encontramos con un club que necesita que se registren y guarden datos de socios, jugadores, aranceles y empleados.

INDICE

OBJETIVOS :

Teniendo en cuenta la introducción ¿Cómo llevaremos a cabo esta tarea?
¿En que datos se hará hincapié?

Veamos..

En primer lugar crearemos una Base de datos.

En segundo lugar vamos a crear nuestra primer tabla, en este caso será la tabla deportes. Acá definiremos los tipos de deporte que tenemos para poder identificar jugadores y empleados del club.

Luego vamos a generar una tabla en la cual vamos a contabilizar a los jugadores de cada deporte en cuestión, ya sea futbol, hockey o vóley.
Cada una de estas disciplinas tendrá una tabla en la cual vamos a generar un identificador para cada uno de los jugadores, las tiras en las cuales se desempeñan, los deportes que realizan, los aranceles que abonan y por ultimo adjuntaremos datos personales de cada uno de ellos.

Con los socios también crearemos una tabla, en la misma se va a plasmar los datos personales de cada uno, un numero de identificación único y también podremos saber si realiza un deporte o no, para identificar cual es su arancel a pagar.

Hablado de aranceles, este club tiene 3 tipos y necesita una tabla para poder ser identificado:
Tiene un arancel para socio, otro para jugadores y otro para jugadores y socios.

Así mismo, el club tiene empleados, y para ellos también se necesita una organización que este ala altura
¿Cuáles serán las tablas a realizar en un principio?:

Al momento 12 personas tiene un rol de subcomisión, entonces:

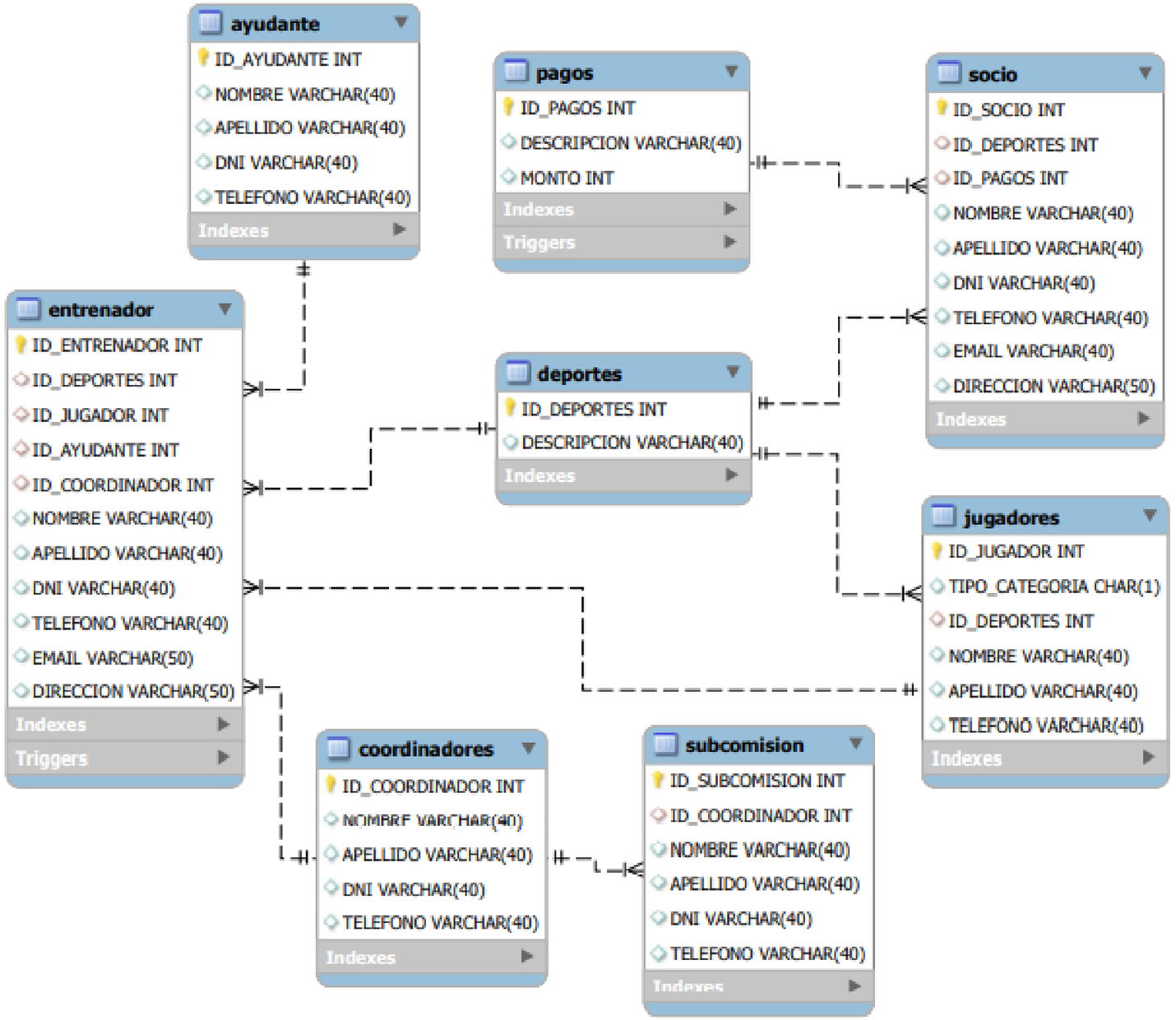
Crearemos la tabla subcomisión con los datos correspondientes para identificar a estas personas que se hacen cargo de ver que tareas lleva a cargo el coordinador de cada deporte.

Los Coordinadores de los que hablamos mas arriba, tienen deportes a cargo, por lo cual es necesaria una tabla para identificar que coordinador se hace cargo de cada deporte y de cada entrenador, por supuesto también necesitaremos los datos personales de cada uno de ellos.

Los entrenadores de este club se van sumando todo el tiempo, como son muchos, necesitamos una tabla en la cual los identifiquemos de manera unica, donde tengamos sus datos personales y también podremos identificar sus deportes y categorías correspondientes.

Por ultimo cada entrenador tiene ayudantes para poder realizar su trabajo de manera optima.
¿necesitamos una tabla también para estas personas? Claro!
En esta tabla tendremos datos personales de cada ayudante, el deporte en el que están trabajando y con que entrenador.

Estas tablas pueden modificarse todo el tiempo o incluso podemos compilarlas para tener distintos resultados, pero eso lo veremos mas adelante.



DER

Diagrama de entidad - relacion

ENTRENADOR

CLAVE	COLUMNAS	TIPO	NOT NULL	NOTAS
PK	ID_ENTRENADOR	INT	TRUE	NUMERO IDENTIFICADOR DEL ENTRENADOR
FK	ID_DEPORTE	INT		NUMERO QUE IDENTIFICA EL DEPORTE QUE DIRIGE
FK	ID_JUGADOR	INT		NUMERO IDENTIFICADOR DEL JUGADOR
FK	ID_AYUDANTE	INT		AYUDANTE DEL ENTRENADOR
FK	ID_COORDINADOR	INT		COODINADOR QUE SE ENCARGA DE ESTE ENTRENADOR
	NOMBRE	VARCHAR		NOMBRE DEL ENTRENADOR
	APELLIDO	VARCHAR		APELLIO DEL ENTRENADOR
	DNI	VARCHAR		DNI DEL ENTRENADOR
	TELEFONO	VARCHAR		TELEFONO DEL ENTRENADOR
	EMAIL	VARCHAR		EMAIL DEL ENTRENADOR
	DIRECCION	VARCHAR		DIRECCION DEL ENTRENADOR

SUBCOMISION

CLAVE	COLUMNAS	TIPO	NOT NULL	NOTAS
PK	ID_SUBCOMISION	INT	TRUE	NUMERO IDENTIFICADOR DE LA PERSONA
FK	ID_COORDINADOR	INT		NUMERO IDENTIFICADOR DEL COORDINADOR DEL CUAL SE HACE CARGO
	NOMBRE	VARCHAR		NOMBRE DE LA PERSONA DE SUBCOMISION
	APELLIDO	VARCHAR		APELLIDO DE LA PERSONA DE SUBCOMISION
	DNI	VARCHAR		DNI DE LA PERSONA DE SUBCOMISION
	TELEFONO	VARCHAR		TELEFONO DE LA PERSONA DE SUBCOMISION

CREACION DE TABLAS Y VALORES

AYUDANTE

CLAVE	COLUMNAS	TIPO	NOT NULL	NOTAS
PK	ID_AYUDANTE	INT	TRUE	NUMERO IDENTIFICADOR DEL AYUDANTE
	NOMBRE	VARCHAR		NOMBRE DEL AYUDANTE
	APELLIDO	VARCHAR		APELLIO DEL AYUDANTE
	DNI	VARCHAR		DNI DEL AYUDANTE
	TELEFONO	VARCHAR		TELEFONO DEL AYUDANTE

COORDINADORES

CLAVE	COLUMNAS	TIPO	NOT NULL	NOTAS
PK	ID_COORDINADOR	INT	TRUE	NUMERO IDENTIFICADOR DE COORDINADOR
	NOMBRE	VARCHAR		NOMBRE DEL COORDINADOR
	APELLIDO	VARCHAR		APELLIO DEL COORDINADOR
	DNI	VARCHAR		DNI DEL COORDINADOR
	TELEFONO	VARCHAR		TELEFONO DEL COORDINADOR

SOCIO

CLAVE	COLUMNAS	TIPO	NOT NULL	NOTAS
PK	ID_SOCIO	INT	TRUE	NUMERO DE SOCIO IDENTIFICADOR
FK	ID_DEPORTE	INT		TIPO DE DEPORTE
	NOMBRE	TEXT		NOMBRE DEL SOCIO
	APELLIDO	TEXT		APELLIO DEL SOCIO
	DNI	INT		DNI DEL SOCIO
	TELEFONO	INT		TELEFONO DEL SOCIO
	EMAIL	VARCHAR		EMAIL DEL SOCIO
	DIRECCION	VARCHAR		DIRECCION DEL SOCIO

JUGADORES

CLAVE	COLUMNAS	TIPO	NOT NULL	NOTAS
PK	ID_JUGADOR	INT	TRUE	NUMERO IDENTIFICADOR DE JUGADOR
	TIPO_CATEGORIA	CHAR		TIPO DE CATEGORIA EN LA QUE JUEGA (A,B,C)
FK	ID_DEPORTE	INT		NUMERO QUE IDENTIFICA EL TIPO DE DEPORTE QUE REALIZA

DEPORTES

CLAVE	COLUMNAS	TIPO	NOT NULL	NOTAS
PK	ID_DEPORTE	INT	TRUE	ID
	DESCRIPCION	VARCHAR		TIPO DE DEPORTE

PAGOS

CLAVE	COLUMNAS	TIPO	NOT NULL	NOTAS
PK	ID_PAGOS	INT	TRUE	NUMERO DE PAGO IDENTIFICADOR
	DESCRIPCION	VARCHAR		TIPO DE PAGO
	MONTO	INT		PRECIO DE LOS ARANCELES

TRIGGERS-REG_ENTRENADORES2

CLAVE	COLUMNAS	TIPO	NOT NULL	NOTAS
	ID_ENTRENADOR	INT	TRUE	NUMERO IDENTIFICADOR DEL ENTRENADOR
	ID_DEPORTES	INT		NUMERO IDENTIFICADOR DEL TIPO DE DEPORTE
	ID_JUGADOR	INT		NUMERO IDENTIFICADOR DEL JUGADOR
	ID_AYUDANTE	INT		NUMERO IDENTIFICADOR DEL AYUDANTE
	ID_COORDINADOR	INT		NUMERO IDENTIFICADOR DE SU COORDINADOR
	NOMBRE	VARCHAR		NOMBRE DEL ENTRENADOR
	APELLIDO	VARCHAR		APELLIDO DEL ENTRENADOR
	DNI	VARCHAR		DNI DEL ENTRENADOR
	TELEFONO	VARCHAR		TELEFONO DEL ENTRENADOR
	EMAIL	VARCHAR		EMAIL DEL ENTRENADOR
	DIRECCION	VARCHAR		DIRECCION DEL ENTRENADOR

TRIGGERS-REG_PAGOS

CLAVE	COLUMNAS	TIPO	NOT NULL	NOTAS
	ANTERIOR_ID_PAGOS	INT	TRUE	NUMERO IDENTIFICADOR DE EMPLEADO
	ANTERIOR_DESCRIPCION	INT		TIPO DE DEPORTE
	ANTERIOR_MONTO	INT		NUMERO DE SOCIO
	NUEVO_ID_PAGOS	TEXT		NOMBRE DEL EMPLEADO
	NUEVO_DESCRIPCION	TEXT		APELLIO DEL EMPLEADO
	NUEVO_MONTO	INT		DNI DEL EMPLEADO
	USUARIO	INT		TELEFONO DEL EMPLEADO
	F_MODIFICACION	VARCHAR		EMAIL DEL EMPLEADO

SEGUIMOS EJECUTANDO NUESTRAS TAREAS PARA CUMPLIR CON EL OBJETIVO:

Como dijimos antes, nuestra BD puede cambiar todo el tiempo según cual sean nuestras necesidades.
Para las consultas que queramos hacer, cambios, modificaciones o concatenaciones vamos a utilizar :

Vistas

Funciones

Stored procedures

Triggers

Vistas

VAMOS A PONERNOS EN CONTEXTO.. UNA VISTA SQL ES BÁSICAMENTE UNA TABLA VIRTUAL QUE SE GENERA A PARTIR DE LA EJECUCIÓN DE UNA O MÁS CONSULTAS SQL, APLICADA SOBRE UNA O MÁS TABLAS.

VISTA VW_SOCIO

En este ejemplo, podemos ver como se crean las vistas. Está, particularmente fue creada para encontrar los emails de todos los socios que tiene en club cuyo inicio de email sea con S.

```
-- AQUI VAMOS A PODER VER EL EMAIL DE LOS SOCIOS QUE COMIENZAN CON S
create or replace view vw_socio as  (
select email
from socio
where email like 's%'
);

select * from vw_socio -- CUANDO EJECUTEMOS ESTE SELECT PODREMOS VERIFICAR SI LA VISTA SE REALIZO CORRECTAMENTE
SELECT* FROM SOCIO -- EJECUANDO ESTE SELECT PODREMOS VERIFICAR SI LA INFORMACION ES CORRECTA
```

VISTA VW_SUBCOMISION

Esta vista esta creada para identificar que personas de subcomisión de hacen cargo del coordinador de voley.

```
-- AQUI VAMOS A PODER VER EL NUMERO DE IDENTIFICACION DE QUIENES CONFORMAN
-- EL GRUPO DE SUBCOMISION QUE TIENEN QUE HACERSE CARGO DEL COORDINADOR NUMERO 3. ESTE NUMERO CORRESPONDE AL COORDINADOR DE VOLEY.
create or replace view vw_subcomision as  (
select ID_SUBCOMISION
from subcomision
where id_coordinador like '3'
);

select * from vw_subcomision-- CUANDO EJECUTEMOS ESTE SELECT PODREMOS VERIFICAR SI LA VISTA SE REALIZO CORRECTAMENTE
SELECT* FROM SUBCOMISION -- EJECUANDO ESTE SELECT PODREMOS VERIFICAR SI LA INFORMACION ES CORRECTA
```

Funciones

LAS FUNCIONES CUSTOMIZADAS PERMITEN PROCESAR Y MANIPULAR DATOS DE FORMA PROCEDURAL Y EFICIENTE. DICHOS DATOS SON ENVIADOS A TRAVÉS DE UNO O MÁS PARÁMETROS, AL MOMENTO DE INVOCAR LA FUNCIÓN Y RETORNANDO UN ÚNICO RESULTADO.

FUNCION - FN_ENTRENADOR

En este ejemplo, podemos ver como se crean las funciones. Esta función esta particularmente creada para encontrar el apellido del entrenador que estamos buscando. Esto mismo se puede realizar con nombre, numeros de identificacion, informacion personal, etc

```
delimiter //
create function FN_entrenador (D int) -- CREAMOS UNA FUNCION CON NOMBRE Y LA DECLARAMOS
returns varchar (100)
deterministic
begin
declare consulta varchar (100);
set consulta =( select (apellido ) from entrenador where D = id_entrenador); -- EN ESTA FUNCION VAMOS A CONSULATR
                                                                              -- EL APELLIDO DEL ENTRENADOR CUYO ID SEA 5
return consulta;
end//
DELIMITER;

Select fn_entrenador  (5) as 'apellido del entrenador'; -- EN ESTE SELECT DECLARAMOS EL NUMERO DEL ENTRENADOR QUE VAMOS A BUSCAR
-- Y TAMBIEN ESTAMOS GENERANDO UN ALIAS LLAMADO "APELLIDO DEL ENTRENADOR"
-- EL CUAL VA A APARECER COMO TITULO EN LA RESPUESTA A MI CONSULTA
```

F U N C I O N E S

Esta función fue creada para contabilizar la cantidad de entrenadores de futbol, hockey y voley que tienen en el club

```
DELIMITER &&
CREATE FUNCTION total_entrenadors () -- CREAMOS UNA FUNCION CON NOMBRE
RETURNS INT
DETERMINISTIC
BEGIN
DECLARE  TOTAL_TABLA INT;
SELECT COUNT(id_entrenador) INTO TOTAL_TABLA FROM entrenador; -- EN ESTA FUNCION VAMOS A CONTAR A TRAVES DE LOS ID EL NUMERO
                                                              -- DE ENTRENADORES QUE TENEMOS EN TOTAL EN TODOS LOS DEPORTES
RETURN TOTAL_TABLA;
END&&
DELIMITER ;

SELECT total_entrenadors (); -- CUANDO EJECUTEMOS ESTE SELECT PODREMOS VERIFICAR SI FUNCION SE EJECUTO CORRECTAMENTE
```

Stored Procedures

UN STORED PROCEDURE O PROCEDIMIENTO ALMACENADO REPRESENTA UN CONJUNTO DE SENTENCIAS ALMACENADO FÍSICAMENTE EN UNA DB, CREADO PARA CUMPLIR TAREAS ESPECÍFICAS.
PERMITE TAMBIÉN ESTABLECER NIVELES DE SEGURIDAD Y MANIPULAR OPERACIONES COMPLEJAS O EXTENSAS DEL LADO DEL SERVIDOR, EVITANDO UN IDA Y VUELTA DE DATOS QUE TERMINE SOBRECARGANDO UNA RED O SERVIDOR.

STORED PROCEDURE

En este ejemplo, podemos ver como se crean los stored procedures. Este sp esta creado para identificar a los jugadores que juegan en tira A, tira B y tira C.

```
DELIMITER !!
CREATE PROCEDURE valida_query (consulta varchar(100))
BEGIN

        if consulta like '%A%' then
            select "jugadores de categoria A" as result;
        ELSEif consulta like '%B%' then
            select "jugadores de categoria B" as result;
        ELSEif consulta like '%C%' then
            select "jugadores de categoria c" as result;
        else
            select * from jugadores;
        end if;

END !!
DELIMITER ;

call valida_query('A');
call valida_query('B');
call valida_query('C');
select* from jugadores
```

```sql
delimiter $$
create procedure SP_Editar_coordinadores (
in id int,
in NOM varchar(40),
in APE varchar(40),
in DNI varchar(40),
in TEL varchar(40)
)
begin
update coordinadores
set  id_coordinador = id,
 NOMBRE = NOM,
 APELLIDO = APE,
 DNI = DNI,
 TELEFONO = TEL
where id_coordinador = id;

end $$

call SP_Editar_coordinadores (3,'ANALIA','PEREZ',34987640 ,8876403422);
SELECT* FROM COORDINADORES
```

STORED PROCEDURES

Este sp fue creado para modificar al coordinador que había anteriormente por el actual, con todos sus datos correspondientes

Triggers

DEFINIMOS COMO TRIGGER A UN CONJUNTO DE SENTENCIAS O PROGRAMA ALMACENADO EN EL SERVIDOR (DE DB) CREADO PARA EJECUTARSE (DISPARARSE) DE FORMA AUTOMÁTICA, CUANDO UNO O MÁS EVENTOS DE DML ESPECÍFICOS OCURREN EN LA DB.
EL TRIGGER SE DESPIERTA Y EJECUTA SUS SENTENCIAS EN EL MOMENTO EN QUE UNA OPERACIÓN DE DML (INSERT, UPDATE Y DELETE) ASOCIADA AL DISPARADOR APARECE.

T R I G G E R S

En este ejemplo, podemos ver una de las maneras en las cuales se puede actualizar el precio de los pagos de las cuotas

```
CREATE TABLE reg_pagos(
ANTERIOR_ID_PAGOS INT ,
ANTERIOR_DESCRIPCION VARCHAR (40),
ANTERIOR_MONTO INT,
NUEVO_ID_PAGOS INT ,
NUEVO_DESCRIPCION VARCHAR (40),
NUEVO_MONTO INT,
USUARIO VARCHAR (15),
F_MODIFICACION DATE
);

delimiter !!
CREATE TRIGGER ACT_PRECIOS_BU BEFORE UPDATE ON PAGOS
FOR EACH ROW
begin
INSERT INTO reg_pagos (ANTERIOR_ID_PAGOS, ANTERIOR_DESCRIPCION, ANTERIOR_MONTO, NUEVO_ID_PAGOS, NUEVO_DESCRIPCION, NUEVO_MONTO, USUARIO, F_MODIFICACION)
VALUES
(OLD.ID_PAGOS, OLD.DESCRIPCION, OLD.MONTO, NEW.ID_PAGOS, NEW.DESCRIPCION, NEW.MONTO, CURRENT_USER(), NOW());
end !!

delimiter ;

UPDATE PAGOS SET MONTO=MONTO+20
WHERE  ID_PAGOS = 1;

SELECT* FROM PAGOS;
SELECT* FROM reg_pagos;
```

TREIGGERS

Este trigger fue creado para sumar un nuevo entrenador al grupo de trabajo. Aquí podemos ver como es la inserción de datos.

```
create table reg_entrenadores2 (
ID_ENTRENADOR INT ,
ID_DEPORTES INT,
ID_JUGADOR INT,
ID_AYUDANTE INT,
ID_COORDINADOR INT,
NOMBRE VARCHAR (40),
APELLIDO VARCHAR (40),
DNI VARCHAR (40),
TELEFONO VARCHAR (40),
EMAIL VARCHAR (50),
DIRECCION VARCHAR (50),
insertado datetime
);

delimiter ##
create trigger ENTRENADOR_AI2
after insert on entrenador
for each row
begin
insert into reg_entrenadores2 (id_entrenador, ID_DEPORTES, id_jugador,  id_ayudante, id_coordinador, nombre, apellido, dni, telefono, email, direccion, insertado)
values (NEW.id_entrenador, NEW.id_deportes,  NEW.id_jugador,  NEW.id_ayudante,  NEW.id_coordinador, NEW.nombre, NEW.apellido, NEW.dni, NEW.telefono,  NEW.email, NEW.direccion, now());
end ##
delimiter ;

insert into entrenador (id_entrenador, ID_DEPORTES, id_jugador,  id_ayudante, id_coordinador, nombre, apellido, dni, telefono, email, direccion)
values (28, 1, 50, 12, 3, 'camilo', 'zapi', 67543896, 2276543988, 'zapican@hotmail.com', 'españa 2309');


select * from reg_entrenadores2;
select * from entrenador;
```