



BASE DE DATOS I

POR: MARCO ANTONIO CALLE VAQUIATA



DETALLE EL PROBLEMA

Diseñe un sistema de Base de Datos Relacional utilizando el gestor de Base de Datos SQL Server teniendo como premisa el uso de buenas prácticas en diseño de la base de datos aplicados al siguiente escenario. Una comunidad de estudiantes de la nación UNIFRANZ de nombre los UNIFRANZITOS desea implementar un nuevo sistema para poder administrar los CAMPEONATOS DE FÚTBOL de todas las sedes. Es decir crear un campeonato en donde puedan participar todas las sedes, en el campeonato pueden inscribirse tanto categoría varones y mujeres.

UNIFRANZITOS

Problema

Se tiene como contexto un **CAMPEONATO DE FÚTBOL** en el cual se tiene 3 entidades principales el **campeonato** como tal, los **equipos** que participaran en el campeonato y en donde cada equipo tendrá una cantidad de **jugadores**.

En tal sentido se deberá crear las siguientes tablas.

- campeonato
- equipo
- jugador

Detalle de las tablas.

campeonato

id_campeonato => cadena de 12 caracteres y ademas llave primaria
nombre_campeonato => una cadena de 30 caracteres que no acepta valores nulos
sede => una cadena de 20 caracteres que no acepta valores nulos

equipo

id_equipo => cadena de 12 caracteres y ademas llave primaria
nombre_equipo => una cadena de 30 caracteres, que no acepta valores nulos
categoría => esta columna recibe valores como (varones o mujeres), que no acepta valores nulos
id_campeonato => llave foreign key relacionado con la tabla campeonato

jugador

id_jugador => cadena de 12 caracteres y ademas llave primaria
nombres => una cadena de 30 caracteres, que no acepta valores nulos
apellidos => una cadena de 50 caracteres, que no acepta valores nulos
ci => una cadena de 15 caracteres (ejem: 8997899LP), que no acepta valores nulos
edad => un valor numérico, que no acepta valores nulos
id_equipo => llave foreign key relacionado con la tabla equipo

DETALLE EL PROBLEMA

Dado el detalle explicado en la parte inicial de este documento debería generar una base de datos similar al siguiente.

Los registros de cada tabla deberían quedar de la siguiente forma



tabla campeonato		
id_campeonato	nombre_campeonato	sede
camp-111	Campeonato Unifranz	El Alto
camp-222	Campeonato Unifranz	Cochabamba

tabla equipo			
id_equipo	nombre_equipo	categoria	id_campeonato
equ-111	Google	VARONES	camp-111
equ-222	404 Not found	VARONES	camp-111
equ-333	girls unifranz	MUJERES	camp-111

tabla jugador					
id_jugador	nombres	apellidos	ci	edad	id_equipo
jug-111	Carlos	Villa	8997811LP	19	equ-222
jug-222	Pedro	Salas	8997822LP	20	equ-222
jug-333	Saul	Araj	8997833LP	21	equ-222
jug-444	Sandra	Solis	8997844LP	20	equ-333
jug-555	Ana	Mica	8997855LP	23	equ-333

LO PONEMOS EN CODIGO EN SQL SERVER

SQLQuery1.sql - DE...4\Marco Calle (51))" -> X

```
create database HIT04
use HIT04
CREATE TABLE CAMPEONATO
(
    ID_CAMPEONATO VARCHAR(12) PRIMARY KEY NOT NULL,
    NOMBRE_DEL_CAMPEONATO VARCHAR (30) NOT NULL,
    SEDE VARCHAR (30) NULL
);

CREATE TABLE EQUIPO
(
    ID_EQUIPO VARCHAR(12) PRIMARY KEY NOT NULL,
    ID_CAMPEONATO VARCHAR (12) NOT NULL,
    NOMBRE_DE_EQUIPO VARCHAR (30) NOT NULL,
    CATEGORIA VARCHAR (30) NULL,
    FOREIGN KEY (ID_CAMPEONATO) REFERENCES CAMPEONATO (ID_CAMPEONATO )
);
```

```
CREATE TABLE JUGADOR
(
    ID_JUGADOR VARCHAR (12) PRIMARY KEY NOT NULL,
    NOMBRES VARCHAR (30),
    APELLIDOS VARCHAR (30),
    CI VARCHAR (30),
    EDAD INTEGER,
    ID_EQUIPO VARCHAR(12) NOT NULL,
    FOREIGN KEY (ID_EQUIPO) REFERENCES EQUIPO(ID_EQUIPO)
);

INSERT INTO CAMPEONATO(ID_CAMPEONATO,NOMBRE_DEL_CAMPEONATO,SEDE)
VALUES('CAMP-111','CAMPEONATO UNIFRANZ','EL ALTO')
INSERT INTO CAMPEONATO(ID_CAMPEONATO,NOMBRE_DEL_CAMPEONATO,SEDE)
VALUES('CAMP-222','CAMPEONATO UNIFRANZ','COCHABAMBA')
INSERT INTO CAMPEONATO(ID_CAMPEONATO,NOMBRE_DEL_CAMPEONATO,SEDE)
VALUES('CAMP-333','CAMPEONATO UNIFRANZ','PANDO')
```


CONTINUAMOS

“

```
INSERT INTO EQUIPO(ID_EQUIPO,NOMBRE_DE_EQUIPO,CATEGORIA,ID_CAMPEONATO)
VALUES('EQUIP-111','GOOGLE','VARONES','CAMP-111')
INSERT INTO EQUIPO(ID_EQUIPO,NOMBRE_DE_EQUIPO,CATEGORIA,ID_CAMPEONATO)
VALUES('EQUIP-222','404 NOT FOUND ','VARONES','CAMP-111')
INSERT INTO EQUIPO(ID_EQUIPO,NOMBRE_DE_EQUIPO,CATEGORIA,ID_CAMPEONATO)
VALUES('EQUIP-333','GIRLS UNIFRANZ ','MUJERES','CAMP-111')
INSERT INTO JUGADOR(ID_JUGADOR,NOMBRES,APELLIDOS,CI,EDAD,ID_EQUIPO)
VALUES('JUG-111','CARLOS','VILLAS','8997811LP',19,'EQUIP-222')
  INSERT INTO JUGADOR(ID_JUGADOR,NOMBRES,APELLIDOS,CI,EDAD,ID_EQUIPO)
VALUES('JUG-222','PEDRO','SALAS','8997822LP',20,'EQUIP-222')
  INSERT INTO JUGADOR(ID_JUGADOR,NOMBRES,APELLIDOS,CI,EDAD,ID_EQUIPO)
VALUES('JUG-333','SAUL','ARAJ','8997833LP',21,'EQUIP-222')
  INSERT INTO JUGADOR(ID_JUGADOR,NOMBRES,APELLIDOS,CI,EDAD,ID_EQUIPO)
VALUES('JUG-444','SANDRA','SOLIS','8997844LP',20,'EQUIP-333')
  INSERT INTO JUGADOR(ID_JUGADOR,NOMBRES,APELLIDOS,CI,EDAD,ID_EQUIPO)
VALUES('JUG-555','ANA','MICA','8997855LP',23,'EQUIP-333')
```

Manejo de conceptos

Muestra un ejemplo de DDL

- **CREATE**, se usa para crear una base de datos, tabla, vistas, etc.
- **ALTER**, se utiliza para modificar la estructura, por ejemplo añadir o borrar columnas de una tabla.
- **DROP**, con esta sentencia, podemos eliminar los objetos de la estructura, por ejemplo un índice o una secuencia.



```
create database HIT04  
use HIT04
```

Muestra un ejemplo de DML


- **SELECT**, esta sentencia se utiliza para realizar consultas sobre los datos.
- **INSERT**, con esta instrucción podemos insertar los valores en una base de datos.
- **UPDATE**, sirve para modificar los valores de uno o varios registros.
- **DELETE**, se utiliza para eliminar las filas de una tabla.



```
SELECT*  
FROM EQUIPO AS EQ  
right JOIN CAMPEONATO AS CA  
ON EQ.ID_EQUIPO = EQ.ID_CAMPEONATO;
```


Para que sirve INNER JOIN

Combina los registros de dos tablas si hay valores coincidentes en un campo común.



```
SELECT*  
FROM JUGADOR AS ju  
inner JOIN EQUIPO AS eq  
ON ju.ID_EQUIPO = eq.ID_EQUIPO;
```

Defina que es una función de agregación

Las **funciones de agregación en SQL** nos permiten efectuar operaciones sobre un conjunto de resultados, pero devolviendo un único valor agregado para todos ellos. Es decir, nos permiten obtener medias, máximos, etc... sobre un conjunto de valores

Liste funciones de agregación que conozca.

- **COUNT:** devuelve el número total de filas seleccionadas por la consulta.
- **MIN:** devuelve el valor mínimo del campo que especifiquemos.
- **MAX:** devuelve el valor máximo del campo que especifiquemos.
- **SUM:** suma los valores del campo que especifiquemos. Sólo se puede utilizar en columnas numéricas.
- **AVG:** devuelve el valor promedio del campo que especifiquemos. Sólo se puede utilizar en columnas numéricas.

Mencione algunas funciones propias de SQL-Server

UPPER() – Convierte un campo a **mayúsculas**


LOWER() – Convierte un campo a **minúsculas**

MID() – Extrae caracteres de un campo de texto

LEN() – Devuelve la **longitud** de un campo de texto

NOW() – Devuelve la **hora y fecha** actuales del sistema

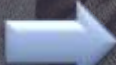
FORMAT() – Da formato a un **formato** para mostrarlo



```
select LOWER(eq.NOMBRE_DE_EQUIPO)
from EQUIPO as eq
```

Para qué sirve la función CONCAT en SQL-Server

Para unir dos o más cadenas en una, use la **función** **CONCAT()** con la siguiente sintaxis: **CONCAT** (input_string1, input_string2 [, input_stringN]); **CONCAT()** toma dos o hasta 255 cadenas **de** entrada y las une en una. Requiere al menos dos cadenas **de** entrada.



```
SELECT CONCAT (ju.NOMBRES, ''
,ju.APELLIDOS, '',ju.EDAD) as nombres_y_apellidos_edad
FROM JUGADOR AS ju
inner JOIN EQUIPO AS eq
ON ju.ID_EQUIPO = eq.ID_EQUIPO
WHERE JU.EDAD > 20
```


¿Muestra un ejemplo del uso de COUNT?

La primera **definición** de recuento en el diccionario es sumar o verificar para determinar la suma; enumerar.

Otra **definición** de conteo es recitar los números en orden ascendente hasta e incluyendo. El conteo también suele pasar por alto para tener en cuenta o incluir.



```
select count(ju.NOMBRES)
from JUGADOR as ju
```

Muestra un ejemplo del usos de AVG

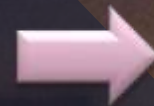
La función **AVG** devuelve el promedio de un conjunto de números. El esquema es SYSIBM. Una expresión que devuelve un conjunto de valores numéricos incluidos booleanos. La función **AVG** omite cualquier valor nulo que pueda contener la expresión.



```
select AVG(ju.EDAD)
from JUGADOR as ju
```

Muestra un ejemplo del uso de MIN-MAX

Las sentencias **SQL MIN** y **MAX** nos ayudan a obtener los valores mínimos y máximos de una columna en una tabla específica de **SQL Server**. **SQL MIN** y **MAX** se puede usar con tipos numéricos y de fecha. **SELECT MIN** devuelve el valor **mínimo** que se encuentra en una columna.



```
select MAX(ju.EDAD)
from JUGADOR as ju
```

```
select MIN(ju.EDAD)
from JUGADOR as ju
```

MANEJO DE CONSULTAS

Mostrar que jugadores que formen parte del equipo equ-333

```
SELECT*  
FROM JUGADOR AS ju  
inner JOIN EQUIPO AS eq  
ON ju.ID_EQUIPO = eq.ID_EQUIPO  
WHERE eq.ID_EQUIPO = 'EQUIP-333'
```

Crear una función que permita saber cuántos jugadores están inscritos.

- La función debe llamarse Crear una función que permita saber cuántos jugadores están inscritos.
- La función debe llamarse F1_CantidadJugadores(())

CONTINUAMOS

```
CREATE FUNCTION F1_CantidadJugadores()  
RETURNS VARCHAR (30) AS  
BEGIN  
DECLARE @MOSTRAR VARCHAR (30);  
SELECT @MOSTRAR= COUNT(JU.NOMBRES)  
FROM JUGADOR AS JU  
RETURN @MOSTRAR  
END;
```

```
SELECT dbo.F1_CantidadJugadores()
```

Crear una función que permita saber cuántos jugadores inscritos y que sean de la categoría varones o mujeres.

- La función debe llamarse

F2_CantidadJugadoresParam()

- La función debe recibir un parámetro “Varones” o “Mujeres”

```
CREATE FUNCTION F2_CantidadJugadoresParam(@SEXO VARCHAR (30))  
RETURNS VARCHAR (30) AS  
BEGIN  
DECLARE @MOSTRAR VARCHAR (30);  
SELECT @MOSTRAR= COUNT(JU.NOMBRES)  
FROM JUGADOR AS JU  
INNER JOIN EQUIPO AS EQ  
ON JU.ID_EQUIPO=EQ.ID_EQUIPO  
WHERE EQ.CATEGORIA=@SEXO  
RETURN @MOSTRAR  
END;
```

```
SELECT dbo.F2_CantidadJugadoresParam('VARONES')
```

Crear una función que obtenga el promedio de las edades mayores a una cierta edad.

- La función debe llamarse F3_PromedioEdades()
- La función debe recibir como parámetro 2 valores.
- La categoría. (Varones o Mujeres)
- La edad con la que se comparara (21 años ejemplo)
- Es decir mostrar el promedio de edades que sean de una categoría y que sean mayores a 21 años

```
CREATE FUNCTION F3_PromedioEde(@CATA VARCHAR (30), @ED INT)
RETURNS VARCHAR (30) as
BEGIN
DECLARE @MOSTRAR VARCHAR (30);
SELECT @MOSTRAR= AVG(JU.EDAD)
FROM JUGADOR AS JU
INNER JOIN EQUIPO AS EQ
ON JU.ID_EQUIPO=EQ.ID_EQUIPO
WHERE EQ.CATEGORIA=@CATA AND JU.EDAD>@ED
RETURN @MOSTRAR
END;

SELECT dbo.F3_PromedioEde('VARONES',20) as promedio
```


Crear una función que permita concatenar 3 parámetros.

- La función debe llamarse F4_ConcatItems() 6

- La función debe de recibir 3 parámetros.

- La función debe de concatenar los 3 valores.

- Para verificar la correcta creación de la función debe mostrar lo siguiente.

- Mostrar los nombres de los jugadores, el nombre del equipo y la sede concatenada, utilizando la función que acaba de crear.

```
CREATE FUNCTION F9_ConcatItems(  
    @PAR1 VARCHAR(80),  
    @PAR2 VARCHAR(80),  
    @PAR3 VARCHAR(80)  
)  
    RETURNS VARCHAR(80) AS  
    BEGIN  
        DECLARE @RESPUESTA VARCHAR(80)  
        DECLARE @CONCATENADO1 VARCHAR(80)  
        DECLARE @CONCATENADO2 VARCHAR(80)  
        DECLARE @CONCATENADO3 VARCHAR(80)  
        SET @CONCATENADO1 = CONCAT(' NOMBRES DE LOS JUGADORES: ',@PAR1);  
        SET @CONCATENADO2 = CONCAT(' NOMBRE DE EQUIPO: ',@PAR2);  
        SET @CONCATENADO3 = CONCAT(' SEDE: ',@PAR3);  
        SET @RESPUESTA = @CONCATENADO1 + @CONCATENADO2 +@CONCATENADO3  
  
        RETURN @RESPUESTA;  
    END;  
SELECT DBO.F9_ConcatItems( JUG.NOMBRES, EQUI.NOMBRE_DE_EQUIPO, CAMP.SEDE)  
FROM JUGADOR AS JUG  
INNER JOIN EQUIPO AS EQUI ON JUG.ID_EQUIPO = EQUI.ID_EQUIPO  
INNER JOIN CAMPEONATO AS CAMP ON EQUI.ID_CAMPEONATO = CAMP.ID_CAMPEONATO
```

El objetivo es generar una función que retorne una cadena con la serie de la fibonacci.

- La función solo recibe el valor N.
- Comportamiento esperado

```
CREATE FUNCTION Fibonacci(@max int)
RETURNS @numbers TABLE(number int)
AS
BEGIN
    Declare @n1 int = 0,@n2 int =1,@i int=0,@temp int
    Insert Into @numbers Values(@n1),(@n2)
    WHILE (@i<=@max-2)
    BEGIN
        Insert Into @numbers Values(@n2+@n1)
        set @temp = @n2
        Set @n2 = @n2 + @n1
        Set @n1 = @temp
        Set @i += 1
    END
    RETURN
END
Select * from dbo.Fibonacci(5)
```



150 %

Results

	number
1	0
2	1
3	1
4	2
5	3
6	5

The background of the image features a close-up of autumn leaves, likely Japanese maple, in shades of purple, red, and orange. A diagonal line splits the image from the top-left to the bottom-right. The area to the left of this line is a solid dark purple, while the area to the right shows the detailed texture of the leaves. The word "GRACIAS!!!" is centered across the diagonal line in a white, bold, serif font.

GRACIAS!!!