DEFENSA HITO 3 - TAREA FINAL

Nombre

Beymar Edy Mamani Mamani

Consigna

- 1. Diseño de base de datos.

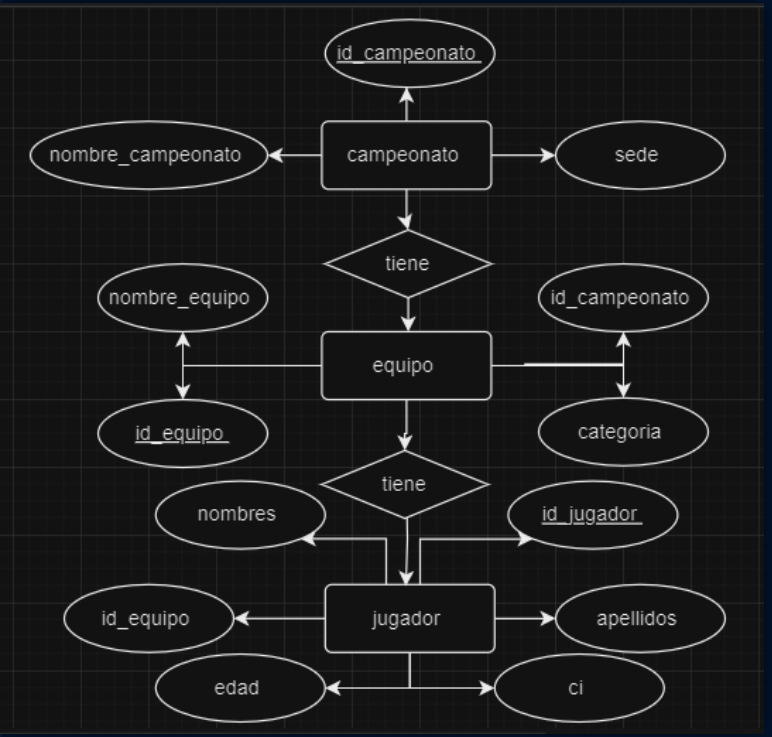
```
CREATE DATABASE UNIFRANSITOS
USE UNIFRANSITOS
CREATE TABLE campeonato (
    id_campeonato VARCHAR(12) PRIMARY KEY,
    nombre_campeonato VARCHAR(30) NOT NULL,
    sede VARCHAR(20) NOT NULL
);
CREATE TABLE equipo (
    id_equipo VARCHAR(12) PRIMARY KEY,
    nombre_equipo VARCHAR(30) NOT NULL,
    categoria VARCHAR(20) NOT NULL,
    id_campeonato CHAR(12) NOT NULL,
    FOREIGN KEY (id_campeonato) REFERENCES
campeonato(id_campeonato)
);
CREATE TABLE jugador (
    id_jugador VARCHAR(12) PRIMARY KEY,
    nombres VARCHAR(30) NOT NULL,
    apellidos VARCHAR(50) NOT NULL,
    ci VARCHAR(15) NOT NULL,
    edad INTEGER NOT NULL,
    id_equipo VARCHAR(12)NOT NULL,
    FOREIGN KEY (id_equipo) REFERENCES equipo(id_equipo)
);
```

```
INSERT INTO campeonato (id_campeonato,
nombre_campeonato, sede) VALUES ('camp-111','Campeonato
Unifranz', 'El Alto'),('camp-222', 'Campeonato Unifranz',
'Cochabamba');

INSERT INTO equipo (id_equipo, nombre_equipo, categoria,
id_campeonato) VALUES ('equ-111', 'google', 'varones', 'camp-
111'),('equ-222', '404 not found', 'varones', 'camp-111'),('equ-
333', 'girls unifranz', 'varones', 'camp-111');

INSERT INTO jugador (id_jugador, nombres, apellidos, ci, edad,
id_equipo) VALUES  ('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');
```

- 2. Manejo de conceptos
- 2.1. Adjuntar el diagrama E-R GENERADO



2.2. ¿Qué es DDL y DML, adicionalmente muestra un ejemplo en la base de datos UNIFRANZITOS?

DDL se utiliza para definir la estructura de la base de datos, como la creación, modificación y eliminación de tablas, índices y restricciones

DML se utiliza para manipular los datos dentro de la base de datos, como la inserción, actualización, eliminación y recuperación de registros.

Ejemplo

```
CREATE TABLE jugador (
    id_jugador VARCHAR(12) PRIMARY KEY,
    nombres VARCHAR(30) NOT NULL,
    apellidos VARCHAR(50) NOT NULL,
    ci VARCHAR(15) NOT NULL,
    edad INT NOT NULL,
    id_equipo VARCHAR(12),
    FOREIGN KEY (id_equipo) REFERENCES equipo(id_equipo)
);
```

```
INSERT INTO jugador (id_jugador, nombres, apellidos, ci, edad, id_equipo) VALUES ('jug-777', 'Juan', 'Perez', '8997877LP', 22, 'equ-111');
```

2.3. ¿Qué significa PRIMARY KEY y FOREIGN KEY?

PRIMARY KEY: La clave primaria (PRIMARY KEY) es una restricción que garantiza que un campo o conjunto de campos en una tabla sea único y que no contenga valores nulos. Se utiliza para identificar de manera única cada registro en la tabla.

FOREIGN KEY: La clave foránea (FOREIGN KEY) es una restricción que establece una relación entre dos tablas, en la que los valores en una columna de una tabla (tabla hija) coinciden con los valores en una columna de otra tabla (tabla padre). Esto garantiza la integridad referencial de la base de datos.

2.4. ¿Qué es una TABLA y el uso de IDENTITY?

Una tabla en una base de datos relacional es una estructura que organiza los datos en filas y columnas, cada fila representa un registro y cada columna representa un atributo o campo de datos.

IDENTITY: IDENTITY es una propiedad que se puede aplicar a una columna de una tabla para que esta columna se autogenere automáticamente con valores numéricos únicos a medida que se insertan nuevos registros en la tabla, es útil para generar identificadores únicos automáticamente.

2.5. ¿Para qué se utiliza la cláusula WHERE?

WHERE se utiliza en consultas SQL para filtrar filas de una tabla que cumplan con una condición específica. Solo las filas que cumplan con la condición especificada en la cláusula WHERE se incluirán en el resultado de la consulta.

2.6. ¿Para qué se utiliza la instrucción INNER JOIN?

INNER JOIN se utiliza para combinar filas de dos o más tablas en función de un campo común entre ellas. Solo se incluirán las filas que tengan coincidencias en ambas tablas. Esto se utiliza para relacionar datos de diferentes tablas en una consulta.

2.7.1. Ejemplo de INNER JOIN:

Suponemos obtener una lista de jugadores y sus equipos en UNIFRANZITOS:

```
SELECT jugador.nombres, jugador.apellidos, equipo.nombre_equipo
FROM jugador
INNER JOIN equipo ON jugador.id_equipo = equipo.id_equipo;
```

2.8.1. Ejemplo de LEFT JOIN:
Suponemos obtener una lista de todos los equipos y los jugadores asociados en UNIFRANZITOS:

SELECT equipo.nombre_equipo, jugador.nombres, jugador.apellidos FROM equipo LEFT JOIN jugador ON equipo.id_equipo = jugador.id_equipo;

2.9.1. Ejemplo de RIGHT JOIN:
Suponemos obtener una lista de todos los jugadores y los equipos a los que están asociados en UNIFRANZITOS:

SELECT jugador.nombres, jugador.apellidos, equipo.nombre_equipo FROM jugador RIGHT JOIN equipo ON jugador.id_equipo = equipo.id_equipo;

2.10. Crear 3 tablas y crear una consulta SQL que muestra el uso de INNER JOIN.

```
CREATE TABLE Clientes (
ClienteID INT PRIMARY KEY,
Nombre VARCHAR(50) );
CREATE TABLE Productos (
ProductoID INT PRIMARY KEY,
NombreProducto VARCHAR(50) );
CREATE TABLE Pedidos (
 PedidoID INT PRIMARY KEY,
ClienteID INT, ProductoID INT,
Cantidad INT );
INSERT INTO Clientes (ClienteID, Nombre) VALUES (1, 'Juan'),(2,
'Maria'),(3, 'Carlos');
INSERT INTO Productos (ProductoID, NombreProducto) VALUES
(101, 'Producto A'),(102, 'Producto B'), (103, 'Producto C');
INSERT INTO Pedidos (PedidoID, ClienteID, ProductoID, Cantidad)
VALUES (1001, 1, 101, 3), (1002, 1, 102, 2), (1003, 2, 101, 1), (1004,
3, 103, 5);
```

```
SELECT Clientes.Nombre, Productos.NombreProducto, Pedidos.Cantidad
FROM Clientes
INNER JOIN Pedidos ON Clientes.ClienteID = Pedidos.ClienteID
INNER JOIN Productos ON Pedidos.ProductoID =Productos.ProductoID;
```

3. Manejo de consultas

3.1. Mostrar jugadores que son del equipo equ-222:

```
SELECT nombres, apellidos
FROM jugador
WHERE id_equipo = 'equ-222';
```

Carlos Villa

3.2. Mostrar jugadores (nombres, apellidos) que juegan en la sede de El Alto:

```
SELECT nombres, apellidos
FROM jugador
INNER JOIN equipo ON jugador.id_equipo = equipo.id_equipo
WHERE equipo.sede = 'El Alto';
```

Carlos Villa
Pedro Salas
Saúl Araj

3.3. Mostrar jugadores mayores o igual a 21 años que sean de la categoría VARONES:

```
SELECT nombres, apellidos
FROM jugador
INNER JOIN equipo ON jugador.id_equipo = equipo.id_equipo  WHERE
jugador.edad >= 21 AND equipo.categoria = 'varones';
```

Saúl Araj

3.4. Mostrar a todos los estudiantes en donde su apellido empiece con la letra S:

```
SELECT nombres, apellidos
FROM jugador
WHERE apellidos LIKE 'S%';
```

- Sandra Solis
- Saúl Araj

3.5. Mostrar qué equipos forman parte del campeonato camp-111 y además sean de la categoría MUJERES:

```
SELECT equipo.nombre_equipo
FROM equipo
 INNER JOIN campeonato ON equipo.id_campeonato = campeonato.id_campeonato
WHERE campeonato.id_campeonato = 'camp-111' AND equipo.categoria = 'mujeres';
```

Ninguna

3.6. Mostrar el nombre del equipo del jugador con id_jugador igual a jug-333:

```
SELECT equipo.nombre_equipo
FROM equipo
INNER JOIN jugador ON equipo.id_equipo = jugador.id_equipo
WHERE jugador.id_jugador = 'jug-333';
```

404 not found

3.7. Mostrar el nombre del campeonato del jugador con id_jugador igual a jug-333:

```
SELECT campeonato.nombre_campeonato
FROM campeonato
INNER JOIN equipo ON campeonato.id_campeonato = equipo.id_campeonato
INNER JOIN jugador ON equipo.id_equipo = jugador.id_equipo
WHERE jugador.id_jugador = 'jug-333';
```

Campeonato Unifranz

3.8. Crear una consulta SQL que maneje las 3 tablas de la base de datos:

```
SELECT campeonato.nombre_campeonato, equipo.nombre_equipo,
jugador.nombres, jugador.apellidos FROM campeonato INNER JOIN equipo
ON campeonato.id_campeonato = equipo.id_campeonato INNER JOIN
jugador ON equipo.id_equipo = jugador.id_equipo;
```

3.9. ¿Qué estrategia utilizaría para determinar cuántos equipos inscritos hay?

Podemos utilizar la función de agregación COUNT en una consulta
Ejemplo:

```
SELECT COUNT(DISTINCT id_equipo) AS equipos_inscritos
FROM equipo;
```

3.10. ¿Qué estrategia utilizaría para determinar cuántos jugadores pertenecen a la categoría VARONES o Categoría MUJERES?

```
SELECT COUNT(*) AS total_varones FROM jugador INNER JOIN equipo ON
jugador.id_equipo = equipo.id_equipo WHERE equipo.categoria = 'varones';
SELECT COUNT(*) AS total_mujeres FROM jugador INNER JOIN equipo ON
jugador.id_equipo = equipo.id_equipo WHERE equipo.categoria = 'mujeres';
```

GRACIAS