Proyecto final SQL
Base de datos para Gestión de Torneos de Videojuegos

Índice

Sistema de Gestión de Torneos de Videojuegos

Introducción:

El presente informe tiene como objetivo presentar y analizar el diseño y la utilidad de un Sistema de Gestión de Torneos de Videojuegos. Este sistema se enfoca en la administración y organización de información relacionada con jugadores, equipos, juegos, torneos y partidas de videojuegos. Proporciona una plataforma centralizada para la gestión de torneos, lo que facilita la organización y promoción de competiciones de videojuegos.

Entidades del Sistema:

Jugador: Representa a un jugador participante en los torneos. Contiene información relevante como ID, nombre, apellido, fecha de nacimiento, país y detalles de contacto.

Equipo: Esta entidad almacena información sobre los equipos que compiten en los torneos. Incluye ID, nombre del equipo, fecha de creación, país de origen y los jugadores asociados a dicho equipo.

Juego: Representa un videojuego disponible para disputarse en los torneos. Contiene datos como ID, nombre del juego, desarrollador, género y plataformas compatibles.

Torneo: Esta entidad almacena la información sobre los torneos organizados. Incluye ID, nombre del torneo, fechas de inicio y finalización, juego asociado, premios ofrecidos, entre otros detalles relevantes.

Partida: Representa las partidas individuales o encuentros dentro de un torneo. Contiene información sobre el ID de la partida, el torneo en el que se juega, los jugadores o equipos participantes, los resultados y la duración de la partida.

Registro de Puntuaciones: Esta entidad almacena los registros de puntuaciones obtenidas por los jugadores en las diferentes partidas y juegos. Incluye información sobre el ID del registro, el jugador, el juego, la puntuación obtenida y la fecha y hora en que se registró.

Problemática:

El creciente auge de los deportes electrónicos y la competición en línea ha llevado a un aumento significativo en la cantidad de torneos de videojuegos. La gestión manual de estos torneos puede resultar compleja y propensa a errores, especialmente cuando se trata de un gran número de jugadores, equipos y partidas. Además, la falta de una plataforma centralizada para organizar y administrar la información puede llevar a una experiencia desorganizada y poco atractiva para los participantes.

Solución Propuesta:

El Sistema de Gestión de Torneos de Videojuegos ofrece una solución integral para superar los desafíos asociados con la organización de torneos. Proporciona una plataforma unificada para gestionar jugadores, equipos, juegos, torneos y partidas de videojuegos de manera eficiente y organizada.

Utilidad y Beneficios:

Facilita la Organización: La plataforma centralizada del sistema simplifica la organización de torneos, permitiendo una gestión eficiente de las diferentes etapas y aspectos del evento.

Mejora la Experiencia del Participante: Los jugadores y equipos pueden registrarse fácilmente en los torneos, acceder a la información relevante y recibir actualizaciones en tiempo real sobre los resultados y horarios.

Automatización de Procesos: El sistema automatiza tareas repetitivas como la generación de clasificaciones y resultados, lo que reduce la carga de trabajo para los organizadores.

Generación de Clasificaciones: El sistema es capaz de generar automáticamente clasificaciones y tablas de posiciones, lo que permite a los participantes conocer su rendimiento y posición en el torneo.

Promoción y Difusión: La plataforma puede utilizarse para promocionar los torneos y atraer a una audiencia más amplia, lo que aumenta la visibilidad y el interés en los eventos.

Acceso a Estadísticas y Datos Históricos: El sistema permite almacenar datos históricos de torneos anteriores, lo que brinda a los organizadores y jugadores acceso a estadísticas y análisis para mejorar su desempeño.

Conclusiones:

El Sistema de Gestión de Torneos de Videojuegos es una herramienta fundamental para la organización y promoción efectiva de competiciones de videojuegos. Al proporcionar una plataforma centralizada, automatizada y de fácil acceso, el sistema mejora la experiencia tanto para los organizadores como para los participantes. Permite una gestión eficiente de los torneos y proporciona valiosa información para mejorar la calidad y competitividad de los eventos. Con esta solución, los torneos de videojuegos pueden alcanzar un nuevo nivel de profesionalismo y atraer a una audiencia más amplia, impulsando así el crecimiento y desarrollo de los deportes electrónicos.

DESCRIPCION DE TABLAS:

JUGADOR	EQUIPO	JUEGO	TORNEO	PARTIDA	REGISTRO DE PUNTUACIONES
id_jugador (int)	id_equipo (int)	id_juego (int)	id_torneo (int)	id_partida (int)	id_puntuacion (int)
nombre (varchar)	nombre_equipo (varchar)	nombre (varchar)	nombre_torneo (varchar)	id_torneo (int)	id_Jugador (int)
apellido (varchar)	fecha_de_creacion (date)	desarrollador (varchar)	fecha_de_inicio (date)	id_equipo (int)	id_Juego (int)
fecha_de_nacimiento (date)	pais (varchar)	genero (varchar)	fecha_de_finalizacion (date)	resultados (varchar)	puntuacion_obtenida (int)
informacion_de_contacto (varchar)	id_torneo(int)	plataforma (varchar)	premios (varchar)	duracion (varchar)	fecha_y_hora (date)
pais (varchar)		id_torneo(int)	id_Juego(int)		
id_equipo (int)					
COLOR CLAVE FORANEA (FK)					
COLOR CLAVE PRIMARIA (PK)					
COLOR ENTIDADES (Tablas)					

DIAGRAMA ER:

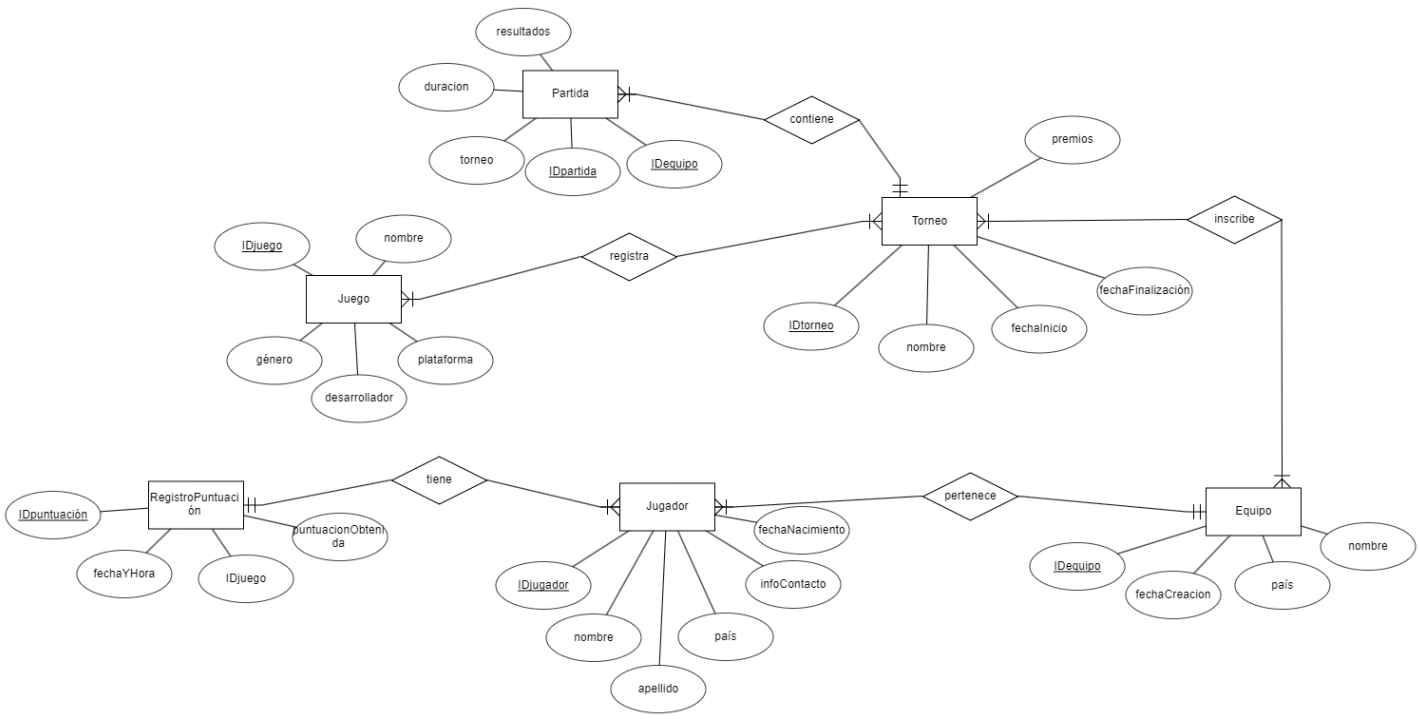
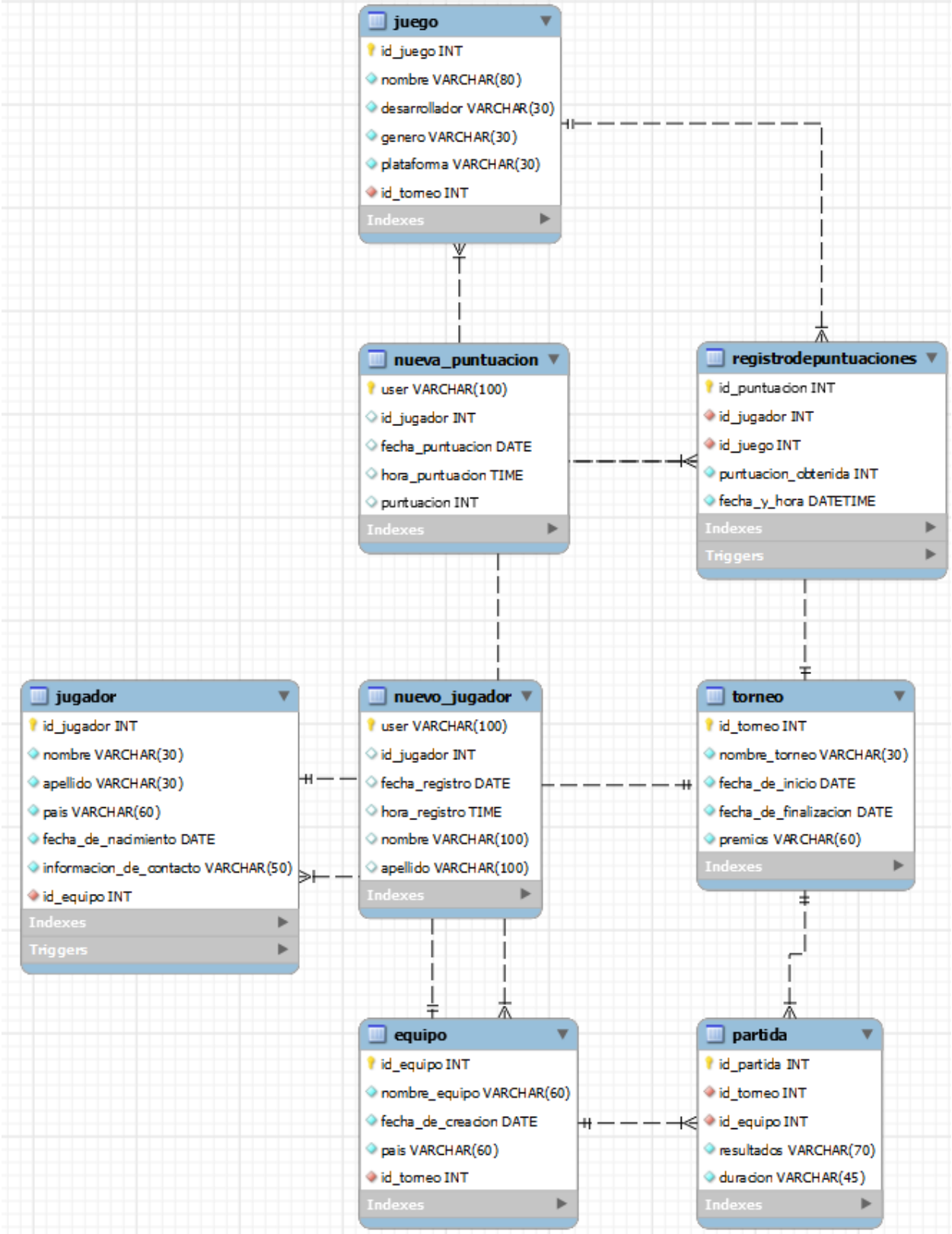


DIAGRAMA ER - WB:



VISTAS

Presione Ctrl + Click para redirigirse al Script en GitHub

El objetivo de estas vistas es proporcionar consultas predefinidas que permitan obtener información específica y relevante de las tablas relacionadas en la base de datos del Sistema de Gestión de Torneos de Videojuegos.

1. Vista puntuacionesDeJugadores

```
CREATE VIEW puntuacionesDeJugadores AS
  (SELECT id_jugador, puntuacion_obtenida

  FROM registrodepuntuaciones

  WHERE fecha_y_hora LIKE ('2023-07-12 17:30:00'));
```

Definición:

La vista `puntuacionesDeJugadores` se crea a partir de la tabla `registrodepuntuaciones`. Filtra los registros para obtener las puntuaciones obtenidas por los jugadores en un momento específico del tiempo, en este caso, el 12 de julio de 2023 a las 17:30:00.

Utilidad:

Esta vista permite obtener rápidamente las puntuaciones de los jugadores registradas en el instante mencionado, lo que puede ser útil para realizar análisis, seguimiento de rendimiento y evaluación de los resultados obtenidos por los jugadores en ese momento exacto.

2. Vista duracionDePartida:

```
CREATE VIEW duracionDePartida AS

   (SELECT id_partida, duracion
```

```
FROM partida

WHERE resultados LIKE ('%A Definir%'));
```

Definición:

La vista `duracionDePartida` se crea a partir de la tabla `partida`. Filtra los registros para obtener la duración de las partidas que aún tienen resultados pendientes de definir, identificados por la cadena "%A Definir%" en la columna "resultados".

Utilidad:

Esta vista permite obtener una lista de las partidas cuyos resultados todavía no han sido determinados, lo que puede ser útil para la planificación y organización de las siguientes etapas del torneo, o para identificar partidas pendientes de resolución.

3. Vista equiposDeLosJugadores:

```
CREATE VIEW equiposDeLosJugadores AS

(SELECT nombre, apellido, id_equipo

FROM jugador);
```

Definición:

La vista `equiposDeLosJugadores` se crea a partir de la tabla `jugador`. Esta vista permite obtener los nombres, apellidos e identificadores de equipo de todos los jugadores registrados en la base de datos.

Utilidad:

Esta vista proporciona una lista completa de todos los jugadores junto con la información de sus respectivos equipos, lo que facilita el seguimiento y la identificación de los jugadores asociados a cada equipo.

4. Vista premioTorneo:

```
CREATE VIEW premioTorneo AS

(SELECT nombre_torneo, premios

FROM torneo);
```

Definición:

La vista `premioTorneo` se crea a partir de la tabla `torneo`. Filtra los registros para obtener los nombres de los torneos y sus respectivos premios.

Utilidad:

Esta vista proporciona una lista de los torneos y sus premios asociados, lo que permite conocer rápidamente qué premios se ofrecen en cada torneo. Es especialmente útil para los jugadores y equipos interesados en participar en torneos con premios atractivos.

5. Vista plataformasDeJuegos:

```
CREATE VIEW plataformasDeJuegos AS
    (SELECT nombre, plataforma
    FROM juego);
```

Definición:

La vista `plataformasDeJuegos` se crea a partir de la tabla `juego`. Proporciona una lista de los nombres de los juegos y sus respectivas plataformas.

Utilidad:

Esta vista permite obtener información sobre los juegos y las plataformas en las que están disponibles. Es útil para identificar qué juegos son compatibles con diferentes plataformas y para facilitar la selección de juegos para los torneos en función de las preferencias de los jugadores y la disponibilidad de las plataformas.

FUNCIONES

Presione Ctrl + Click para redirigirse al Script en GitHub

Estas funciones tienen como objetivo contar el número de puntuaciones en la tabla `registrodepuntuaciones` que son mayores o menores a un valor específico, dependiendo de la función. A continuación, la explicación detallada de cada una de las funciones:

1. Función contarPuntuacionesMayoresA:

```
CREATE FUNCTION contarPuntuacionesMayoresA(cantidadDePuntos INT)

RETURNS INT

NO SQL

BEGIN

    DECLARE contador INT;

    SET contador = (SELECT COUNT(*) FROM registrodepuntuaciones WHERE puntuacion_obtenida > cantidadDePuntos);


   RETURN contador;

END //
```

Esta función tiene el objetivo de contar el número de puntuaciones en la tabla `registrodepuntuaciones` que son mayores a un valor específico (`cantidadDePuntos`), que es el parámetro que se pasa a la función.

2. Función contarPuntuacionesMenoresA:

```
CREATE FUNCTION contarPuntuacionesMenoresA(cantidadDePuntos INT)

RETURNS INT
```

```
NO SQL

BEGIN

    DECLARE contador INT;
    SET contador = (SELECT COUNT(*) FROM registrodepuntuaciones WHERE puntuacion_obtenida <
cantidadDePuntos);



    RETURN contador;
END //
```

Esta función es similar a la anterior, pero en lugar de contar puntuaciones mayores a un valor, cuenta puntuaciones menores a un valor específico (`cantidadDePuntos`).

En resumen, estas funciones permiten obtener el número de puntuaciones mayores o menores a un valor dado, proporcionando una forma reutilizable de realizar estas consultas específicas en la tabla `registrodepuntuaciones`.

STORED PROCEDURES

Presione Ctrl + Click para redirigirse al Script en GitHub

Estos procedimientos realizan consultas y proporcionan resultados para obtener información sobre los equipos y jugadores en el proyecto. A continuación, detallo cada uno de los procedimientos y explico el beneficio que aportan al proyecto:

1. Procedimiento insertarEquipo:

```
CREATE PROCEDURE insertarEquipo(IN nombreEquipo VARCHAR(100), IN fechaCreacion DATE, IN pais VARCHAR(100))

BEGIN

    -- Insertar el nuevo equipo con el id_torneo igual a 1

     INSERT INTO equipo (nombre_equipo, fecha_de_creacion, pais, id_torneo)

     VALUES (nombreEquipo, fechaCreacion, pais, 1);



    -- Obtener el id_equipo generado durante la inserción

    SELECT LAST_INSERT_ID() AS id_equipo;

END //
```

Este procedimiento tiene como objetivo insertar un nuevo equipo en la tabla `equipo` con los datos proporcionados como parámetros de entrada (`nombreEquipo, fechaCreacion, pais`). A continuación, obtiene el `id_equipo` generado durante la inserción y lo devuelve como resultado.
Beneficios:

- **Reutilización**: Al encapsular la lógica de inserción en un procedimiento, se puede reutilizar fácilmente en diferentes partes del proyecto sin necesidad de repetir el código de inserción en cada lugar donde se necesite agregar un equipo.
- **Seguridad**: Al usar un procedimiento almacenado, se evita la exposición directa de la tabla y se limita el acceso a la base de datos solo a través del procedimiento, lo que puede mejorar la seguridad del sistema.

- **Mantenimiento**: Si en el futuro es necesario cambiar la lógica de inserción o agregar validaciones adicionales, solo se necesita modificar el procedimiento en lugar de realizar cambios en todas las partes del código donde se realiza la inserción.

2. Procedimiento ordenarTabla:

```
CREATE PROCEDURE ordenarTabla(IN nombreTabla VARCHAR(50), IN nombreCampo VARCHAR(50), IN
orden VARCHAR(10))
BEGIN

    -- Verificar si el parámetro de orden es válido

    IF (orden <> 'ASC' AND orden <> 'DESC') THEN

        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'El parámetro de orden debe ser "ASC" o "DESC".';

    ELSE
        -- Construir y ejecutar la consulta dinámica

        SET @sql = CONCAT('SELECT * FROM ', nombreTabla, ' ORDER BY ', nombreCampo, ' ', orden);

        PREPARE stmt FROM @sql;

        EXECUTE stmt;

        DEALLOCATE PREPARE stmt;

    END IF;

END //
```

Este procedimiento tiene como objetivo ordenar los registros de una tabla específica (`nombreTabla`) según un campo (`nombreCampo`) y un orden (`orden`) específicos. Los parámetros de entrada permiten que el procedimiento sea más versátil, ya que puede ser utilizado para ordenar diferentes tablas y campos según la necesidad.
Beneficios:

- **Reutilización:** Al encapsular la lógica de ordenación en un procedimiento, se puede reutilizar fácilmente para ordenar diferentes tablas en diferentes partes del proyecto, lo que evita duplicación de código.
- **Flexibilidad:** Al usar parámetros de entrada, el procedimiento se adapta para ordenar diferentes tablas y campos, lo que proporciona una solución más flexible y versátil para las necesidades de ordenación del proyecto.
- **Validación de parámetros:** El procedimiento realiza una validación para asegurarse de que el parámetro de orden (`orden`) sea "ASC" o "DESC". Si se proporciona un valor incorrecto, se

generará una excepción, lo que garantiza que se utilicen valores válidos y evita errores de ejecución.

En resumen, estos procedimientos almacenados aportan beneficios en términos de reutilización de código, seguridad, mantenimiento y flexibilidad, lo que puede mejorar la eficiencia y la calidad del proyecto al trabajar con la base de datos de manera más estructurada y segura.

TRIGGERS

Presione Ctrl + Click para redirigirse al Script en GitHub

1. **tr_add_new_score:**

Este trigger se ejecutará automáticamente antes de insertar un registro en la tabla registrodepuntuaciones.
Para cada nuevo registro insertado en registrodepuntuaciones, el trigger almacenará la información en la tabla nueva_puntuacion.

```
CREATE TRIGGER tr_add_new_score
BEFORE INSERT ON registrodepuntuaciones
for each row
insert INTO nueva_puntuacion (user,
id_jugador,fecha_puntuacion,hora_puntuacion,puntuacion)
        VALUES (USER(),NEW.id_jugador,CURDATE(),CURTIME(),new.puntuacion_obtenida);
```

Este trigger controlara la fecha y hora exactas en las que se registra la puntuación de un jugador, que es diferente al campo fecha_y_hora de la tabla registrodepuntuaciones, en el cual se ingresa la fecha y hora en la que se obtuvo la puntuación. Además, registra el usuario que realizo la operación, el id del jugador y su puntuación obtenida en esta nueva tabla nueva_puntuacion.

2. **tr_add_new_player:**

Este trigger se ejecutará automáticamente después de insertar un registro en la tabla jugador.
Para cada nuevo registro insertado en jugador, el trigger almacenará la información en la tabla nuevo_jugador.

```
CREATE TRIGGER tr_add_new_player
AFTER INSERT ON jugador
for each row
insert INTO nuevo_jugador (user, id_jugador,fecha_registro,hora_registro,nombre,apellido)
        VALUES (USER(),NEW.id_jugador, CURDATE(),CURTIME(),new.nombre,new.apellido);
```

En resumen, cada vez que se inserta un nuevo registro en la tabla jugador, el trigger tr_add_new_player captura el usuario actual, la fecha y hora actual, el id_jugador, el nombre y el apellido del nuevo jugador y los guarda en la tabla nuevo_jugador.
Esto es útil para mantener un historial de los nuevos jugadores registrados en la base de datos.

CREACIÓN DE TABLAS

Presione Ctrl + Click para redirigirse al Script en GitHub

INSERCIÓN DE DATOS

Presione Ctrl + Click para redirigirse al Script en GitHub

CREACIÓN DE USUARIOS

Presione Ctrl + Click para redirigirse al Script en GitHub

1. **Usuario solo lectura:**

-Comando para crear al usuario 'usuario_solo_lectura' con permisos de solo lectura sobre todas las tablas

```
CREATE USER 'usuario1_solo_lectura'@'localhost' IDENTIFIED BY 'usuario1lectura';
```

Comando para otorgarle permisos de solo lectura a 'usuario_solo_lectura' en todas las tablas de la base de datos 'basedatos'

```
GRANT SELECT ON tournament_tracker. * TO 'usuario1_solo_lectura'@'localhost';
```

2. **Usuario lectura y modificación:**

Comando para crear al usuario 'usuario_lectura_modificacion_insercion' con permisos de lectura, inserción y modificación sobre todas las tablas

```
CREATE USER 'usuario2_modificacion'@'localhost' IDENTIFIED BY 'usuario2modificacion';
```

Comando para otorgarle permisos de lectura, inserción y modificación a 'usuario_modificacion' en todas las tablas de la base de datos 'basedatos'

```
GRANT SELECT, INSERT, UPDATE ON tournament_tracker. * TO
'usuario2_modificacion'@'localhost';
```

REPOSITORIO EN GITHUB

A continuación, dejo el enlace del repositorio con todo el proyecto, donde encontraran todos los scripts y pdfs de las entregas semanales y las Pre-Entregas.

Link del Repositorio

HERRAMIENTAS UTILIZADAS

- **GitHub:** Plataforma de desarrollo de software basada en la nube que se utilizó para gestionar el proyecto entero.
- **MySQL Workbench:** Herramienta de desarrollo y administración de bases de datos
- **ERDPlus:** Página web que se utilizó para el diseño del Diagrama Entidad Relación de la base de datos.
- **Excel:** Se utilizó para la descripción de las tablas y todos sus campos.