**Bases de Datos Relacionales - Tarea 8**
**Eduardo David González Flores**
**1515489**

# VISTAS

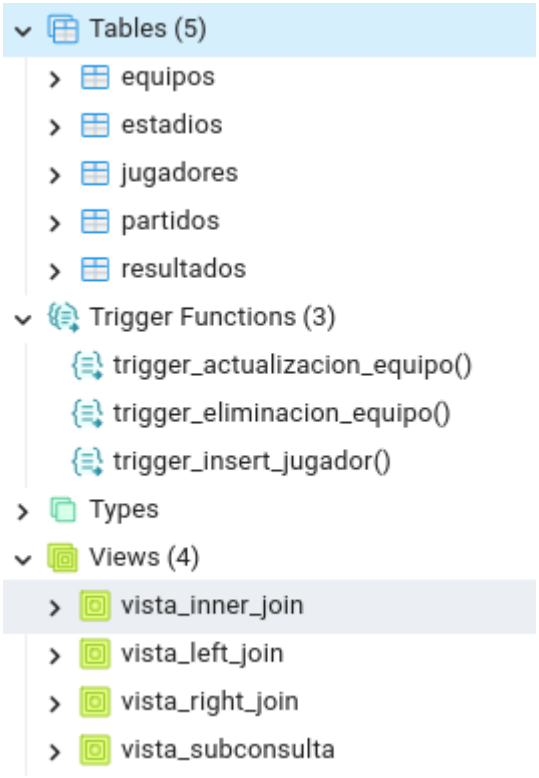
```sql
-- Vista con inner join: Muestra el nombre del equipo y el nombre de los jugadores asociados
-- Beneficio: Proporciona una vista consolidada de los equipos y sus jugadores, facilitando consultas que requieran información conjunta de ambas tablas.

CREATE OR REPLACE VIEW vista_inner_join AS
SELECT e.nombre AS nombre_equipo, j.nombre AS nombre_jugador
FROM equipos e
INNER JOIN jugadores j ON e.id = j.equipo;

-- Vista con left join: Muestra el nombre del equipo y la cantidad total de jugadores asociados
-- Beneficio: Permite obtener una visión general de los equipos y el número de jugadores que tienen asociados, incluso aquellos equipos que no tienen jugadores.

CREATE OR REPLACE VIEW vista_left_join AS
SELECT e.nombre AS nombre_equipo, COUNT(j.id) AS total_jugadores
FROM equipos e
LEFT JOIN jugadores j ON e.id = j.equipo
GROUP BY e.nombre;

-- Vista con right join: Muestra el nombre de los jugadores y el nombre de los equipos asociados
-- Beneficio: Proporciona una vista que muestra todos los jugadores y el equipo al que están asociados, incluso aquellos jugadores que no tienen un equipo asignado.

CREATE OR REPLACE VIEW vista_right_join AS
SELECT j.nombre AS nombre_jugador, e.nombre AS nombre_equipo
FROM jugadores j
RIGHT JOIN equipos e ON e.id = j.equipo;

-- Vista con subconsulta: Muestra el nombre del equipo, la ciudad y el nombre de los jugadores asociados a los equipos que hayan tenido partidos a partir del 1 de enero de 2023
-- Beneficio: Permite obtener una lista de equipos con sus respectivas ciudades y los jugadores asociados, limitados a aquellos equipos que hayan tenido partidos a partir de una fecha específica.

CREATE OR REPLACE VIEW vista_subconsulta AS
SELECT e.nombre AS nombre_equipo, e.ciudad, j.nombre AS nombre_jugador
FROM equipos e
INNER JOIN jugadores j ON e.id = j.equipo
WHERE e.id IN (
 SELECT p.equipo_local
 FROM partidos p
 WHERE p.fecha >= '2023-01-01'::date
);
```

# TRIGGERS

```
-- TRIGGER DE INSERCIÓN
-- Descripción: Este trigger se activa después de insertar un nuevo registro en la tabla
"jugadores".
-- Beneficios: Permite realizar acciones adicionales automáticamente después de insertar un
jugador, como la actualización de campos relacionados o la generación de registros adicionales
en otras tablas.

CREATE OR REPLACE FUNCTION trigger_insert_jugador()
RETURNS TRIGGER AS
$$
BEGIN
 -- Acciones adicionales a realizar después de la inserción del jugador
 -- Ejemplo: Actualizar el número total de jugadores en la tabla "equipos"
 UPDATE equipos
 SET total_jugadores = total_jugadores + 1
 WHERE id = NEW.equipo;

 RETURN NEW;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER insercion_jugador
AFTER INSERT ON jugadores
FOR EACH ROW
EXECUTE FUNCTION trigger_insert_jugador();


-- TRIGGER DE ACTUALIZACIÓN
-- Descripción: Este trigger se activa después de actualizar un registro en la tabla
"equipos".
-- Beneficios: Permite realizar acciones adicionales automáticamente después de actualizar un
equipo, como la verificación de restricciones o la actualización de datos relacionados en
otras tablas.

CREATE OR REPLACE FUNCTION trigger_actualizacion_equipo()
RETURNS TRIGGER AS
$$
BEGIN
 -- Acciones adicionales a realizar después de la actualización del equipo
 -- Ejemplo: Verificar si el equipo ha cambiado de conferencia y actualizar la tabla
"partidos" en consecuencia
 IF OLD.conferencia <> NEW.conferencia THEN
   UPDATE partidos
   SET conferencia_local = NEW.conferencia
   WHERE equipo_local = NEW.id;
 END IF;

 RETURN NEW;
END;
```

```
$$
LANGUAGE plpgsql;

CREATE TRIGGER actualizacion_equipo
AFTER UPDATE ON equipos
FOR EACH ROW
EXECUTE FUNCTION trigger_actualizacion_equipo();


-- TRIGGER DE ELIMINACIÓN
-- Descripción: Este trigger se activa antes de eliminar un registro en la tabla "equipos".
-- Beneficios: Permite realizar acciones adicionales antes de eliminar un equipo, como la
eliminación de registros relacionados o la verificación de restricciones.

CREATE OR REPLACE FUNCTION trigger_eliminacion_equipo()
RETURNS TRIGGER AS
$$
BEGIN
 -- Acciones adicionales a realizar antes de la eliminación del equipo
 -- Ejemplo: Eliminar los jugadores asociados al equipo que se está eliminando
 DELETE FROM jugadores
 WHERE equipo = OLD.id;

 RETURN OLD;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER eliminacion_equipo
BEFORE DELETE ON equipos
FOR EACH ROW
EXECUTE FUNCTION trigger_eliminacion_equipo();
```

# EVIDENCIAS




# TEMA PROYECTO INTEGRADOR

**TEMA: Integración de SQL con Python y AWS**

# Referencia

## Tarea 8

1. Crear vistas (`VIEW`) sobre consultas significativas, recurrentes, etc. que
   a. [1 punto] incluyan un `JOIN` (al menos una)
   b. [1 punto] incluyan un `LEFT JOIN` (al menos una)
   c. [1 punto] incluyan un `RIGHT JOIN` (al menos una)
   d. [1 punto] incluyan una sub consulta (al menos una)
2. [2 puntos] Investigar y crear al menos un disparador (`TRIGGER`) de inserción, actualización o eliminación
3. Guarda tus consultas como archivo SQL en tu repositorio
4. [4 puntos] Explicar qué hace cada vista y disparador que utilizas y qué beneficios para tu BD tiene crearlos en un archivo PDF o MD que subas a tu repositorio
5. Elegir tema para Proyecto Integrador de Aprendizaje

## Proyecto Integrador de Aprendizaje

1. Elige un tema relacionado a la materia y distinto del resto de tus compañeros, por ejemplo:
   a. Instalación de SGBD
   b. Tutorial de software para manipular SGBD
   c. Funciones para cadenas
   d. Minería de textos
   e. Series de tiempo
   f. Funciones para fechas
   g. Usuarios y permisos
   h. Integraciones con Python
   i. NO-SQL
   j. Integración con aplicaciones móviles o web
2. Prepara una clase de 10 a 20 min
3. [10 puntos] Grábala y publícala en Youtube (o exponla durante clase)
4. [5 puntos] Da retroalimentación honesta y significativa de las clase de los demás compañeros en el archivo correspondiente
5. [5 puntos] Calificación del profesor