

Proyecto de Bases de Datos: Campeonatos de Tenis



Daniel Verano Moreno 1º DAW Mañana

Índice

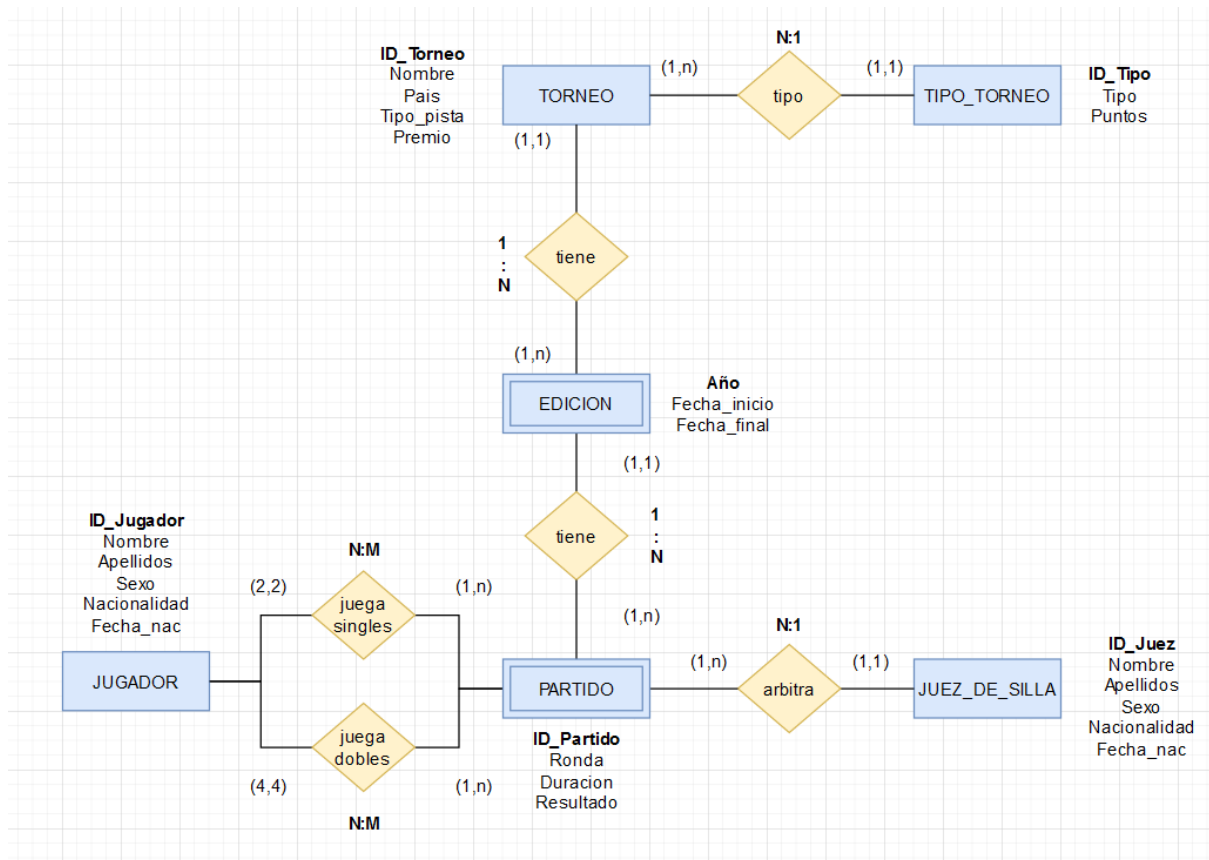
1. Descripción del proyecto
2. Modelo Entidad Relación
3. Modelo Relacional
4. Carga masiva de datos
5. Realización de consultas
6. Funciones
7. Procedimientos
8. Triggers
9. Valoración personal

1. Descripción del proyecto

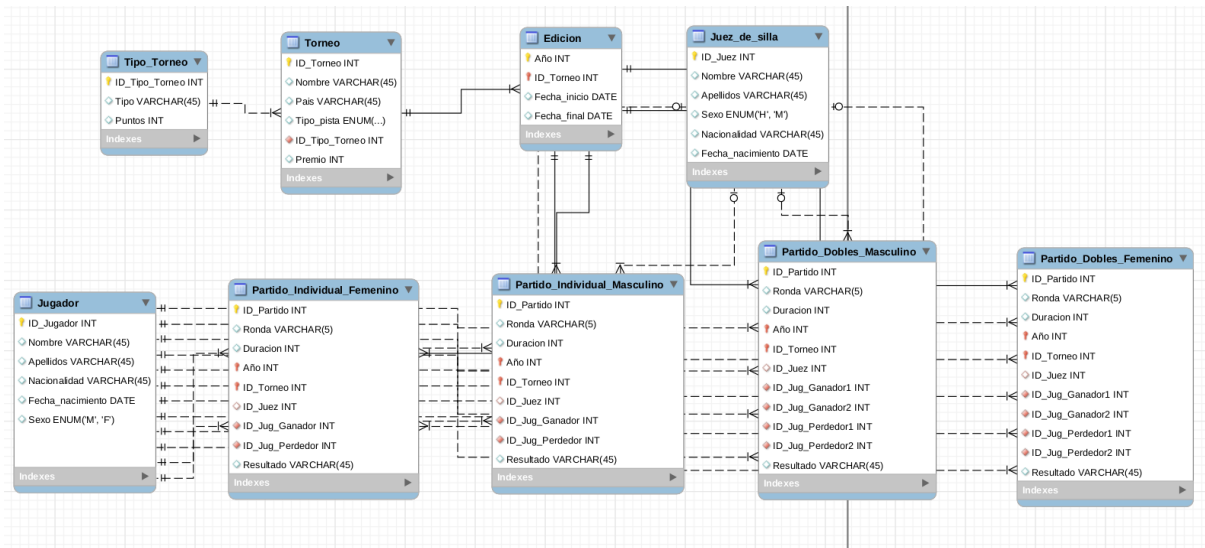
En este proyecto vamos a representar un modelo sobre una base de datos de diferentes campeonatos de tenis, explicando cuáles son sus diferentes entidades, los atributos y cómo se relacionan entre sí.

- Tenemos una entidad **Torneo** la cual tiene su identificador, nombre del torneo, país en el que se juega, el tipo de pista del torneo (que puede ser dura, hierba o tierra batida), la categoría a la que pertenece el torneo y el premio.
- Cada torneo pertenece a una **categoría**, dependiendo de la relevancia del torneo, que puede ser desde un Grand Slam, Masters Finals, Masters 1000, ATP 500, ATP 250, Challenger, etc. Cada categoría tiene una serie de puntos a repartir entre los mejores jugadores del torneo.
- Cada torneo tiene su propia **edición**, que normalmente es de una vez al año, con una fecha de inicio y una fecha de finalización.
- Cada edición del torneo tiene muchos **partidos**, los cuales tienen un identificador, la ronda del partido, su duración, que la guardaremos en minutos, y un resultado, que nos indica el resultado de todos los sets del partido.
- Por otro lado, la tabla Partido se dividirá entre partidos **individuales** y partidos **dobles**, en los que en los individuales aparecerán la ID del jugador ganador y la ID del jugador perdedor. En los partidos dobles aparecerán cuatro ID de jugadores, dos pertenecientes al equipo ganador y otros dos que pertenecen al equipo perdedor.
- Los **jugadores** pueden participar en muchos partidos, de los cuales guardaremos su identificador, nombre, apellidos, sexo, nacionalidad y fecha de nacimiento.
- También guardaremos la información de los **jueces de silla** que participan en cada partido, que será un identificador, nombre, apellidos, sexo, nacionalidad y fecha de nacimiento.

2. Modelo Entidad Relación



3. Modelo relacional

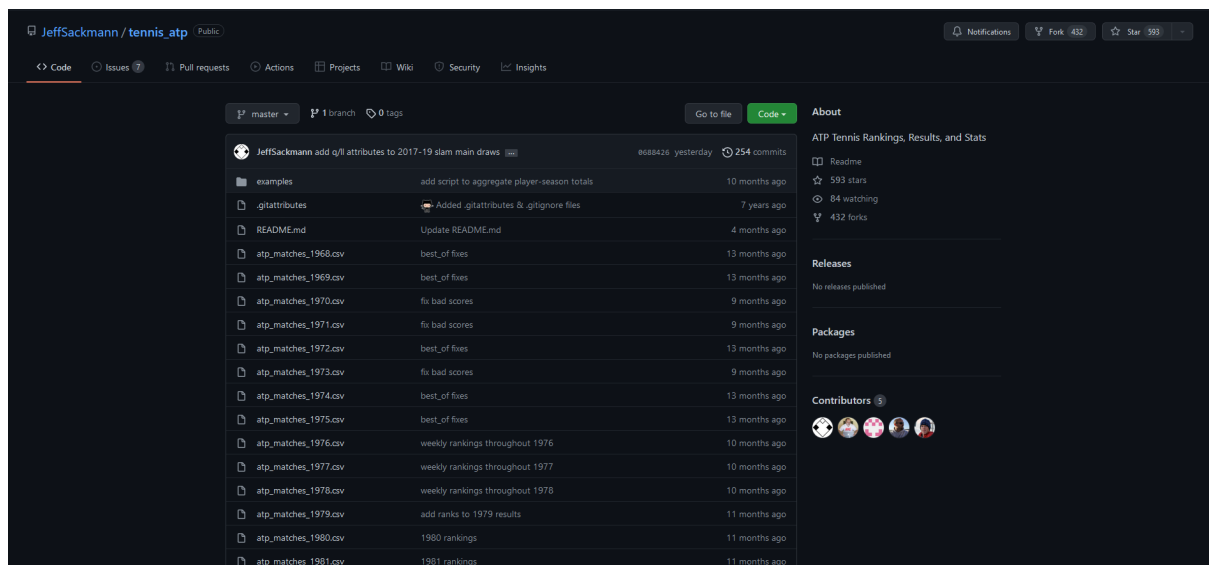


4. Carga masiva de datos

Para la carga de datos he utilizado las siguientes páginas:

- https://github.com/JeffSackmann/tennis_atp
- https://github.com/JeffSackmann/tennis_wta

Se trata de dos repositorios en Github tanto para los jugadores masculinos como femeninos en el cual aparecen muchos archivos .csv ordenados por el año en el cual se jugaron los partidos, además de estar ordenados por partidos individuales y dobles. También podemos encontrar archivos ordenados por otras categorías, como los torneos Challenger y Futures.



Principalmente tuve dos problemas a la hora de cargar los datos. En primer lugar, en los archivos .csv de estos repositorios hay una columna llamada 'tourney_id' la cual está formada por el año en el que se jugaron los partidos seguido de un guión más un identificador del torneo.

En mi modelo, las tablas de Partido tienen como clave primaria tres columnas, el ID del partido, el año en el que se jugó el partido y el ID del torneo.

Por esta razón, tuve que dividir la columna 'tourney_id' en dos, una para el año y otra para el ID del torneo. Por otro lado, también tuve que modificar el ID del torneo para que coincidieran con el ID del torneo en mi modelo, ya que eran diferentes.

1463 lines (1463 sloc) | 324 KB

RawBlame

Search this file...

1	tourney_id	tourney_name	surface	draw_size	tourney_level	tourney_date	match_num	winner_id	winner_seed	winner_entry	winner_name
2	2020-8888	Atp Cup	Hard	24	A	20200106	300	104925			Novak Djokovic
3	2020-8888	Atp Cup	Hard	24	A	20200106	299	105138			Roberto Bautista Agut
4	2020-8888	Atp Cup	Hard	24	A	20200106	298	104925			Novak Djokovic
5	2020-8888	Atp Cup	Hard	24	A	20200106	297	105583			Dusan Lajovic
6	2020-8888	Atp Cup	Hard	24	A	20200106	296	104745			Rafael Nadal
7	2020-8888	Atp Cup	Hard	24	A	20200106	295	105138			Roberto Bautista Agut
8	2020-8888	Atp Cup	Hard	24	A	20200106	294	104925			Novak Djokovic
9	2020-8888	Atp Cup	Hard	24	A	20200106	293	105583			Dusan Lajovic
10	2020-8888	Atp Cup	Hard	24	A	20200106	292	106421			Daniil Medvedev
11	2020-8888	Atp Cup	Hard	24	A	20200106	291	111575			Karen Khachanov
12	2020-8888	Atp Cup	Hard	24	A	20200106	290	105554			Daniel Evans
13	2020-8888	Atp Cup	Hard	24	A	20200106	289	106401			Nick Kyrgios
14	2020-8888	Atp Cup	Hard	24	A	20200106	288	105676			David Goffin
15	2020-8888	Atp Cup	Hard	24	A	20200106	287	105138			Roberto Bautista Agut
16	2020-8888	Atp Cup	Hard	24	A	20200106	286	104925			Novak Djokovic
17	2020-8888	Atp Cup	Hard	24	A	20200106	285	105332			Benoit Paire
18	2020-8888	Atp Cup	Hard	24	A	20200106	284	104925			Novak Djokovic
19	2020-8888	Atp Cup	Hard	24	A	20200106	283	105583			Dusan Lajovic
20	2020-8888	Atp Cup	Hard	24	A	20200106	282	104925			Novak Djokovic
21	2020-8888	Atp Cup	Hard	24	A	20200106	281	105583			Dusan Lajovic
22	2020-8888	Atp Cup	Hard	24	A	20200106	280	104731			Kevin Anderson
23	2020-8888	Atp Cup	Hard	24	A	20200106	279	104468			Gilles Simon
24	2020-8888	Atp Cup	Hard	24	A	20200106	278	104792			Gael Monfilis
25	2020-8888	Atp Cup	Hard	24	A	20200106	277	105332			Benoit Paire

Una vez realizadas estas modificaciones y eliminado las columnas que no necesitaba, ya tenía los datos listos para ser cargados.

	A	B	C	D	E	F	G	H	I
1	ID_Partido	Ronda	Duracion	Año	ID_Torneo	ID_Juez	ID_Jug_Ganador	ID_Jug_Perdedor	Resultado
2	300	F	115	2020	29		104925	104745	6-2 7-6(4)
3	299	F	97	2020	29		105138	105583	7-5 6-1
4	298	SF	167	2020	29		104925	106421	6-1 5-7 6-4
5	297	SF	108	2020	29		105583	111575	7-5 7-6(1)
6	296	SF	133	2020	29		104745	200282	4-6 7-5 6-1
7	295	SF	81	2020	29		105138	106401	6-1 6-4
8	294	QF	160	2020	29		104925	133430	4-6 6-1 7-6(4)
9	293	QF	99	2020	29		105583	200000	6-4 6-2
10	292	QF	142	2020	29		106421	106043	6-4 4-6 6-3
11	291	QF	98	2020	29		111575	105550	6-2 7-6(4)
12	290	QF	203	2020	29		105554	200282	7-6(4) 4-6 7-6(2)
13	289	QF	72	2020	29		106401	111815	6-2 6-2
14	288	QF	143	2020	29		105676	104745	6-4 7-6(3)
15	287	QF	90	2020	29		105138	106293	6-1 6-4
16	286	RR	91	2020	29		104925	104792	6-3 6-2
17	285	RR	125	2020	29		105332	105583	6-2 6-7(6) 6-4
18	284	RR	139	2020	29		104925	104731	7-6(5) 7-6(6)
19	283	RR	135	2020	29		105583	144750	3-6 7-6(4) 6-3
20	282	RR	72	2020	29		104925	106426	6-3 6-3
21	281	RR	93	2020	29		105583	111797	6-2 7-6(3)
22	280	RR	153	2020	29		104731	105332	2-6 7-6(1) 7-6(5)
23	279	RR	135	2020	29		104468	144750	2-6 6-2 6-2
24	278	RR	79	2020	29		104792	106426	6-3 7-5
25	277	RR	132	2020	29		105332	111797	6-7(3) 6-3 6-3
26	276	RR	74	2020	29		104731	106426	6-0 6-3
27	275	RR	81	2020	29		144750	111797	6-4 6-4
28	274	RR	127	2020	29		104745	106415	7-6(4) 6-4
29	273	RR	79	2020	29		105138	104424	6-2 6-4
30	272	RR	109	2020	29		104745	105932	6-3 7-5
31	271	RR	72	2020	29		105138	106223	6-0 6-0
32	270	RR	73	2020	29		104745	104655	6-2 6-1
33	269	RR	57	2020	29		105138	208364	6-1 6-2
34	268	RR	66	2020	29		106415	105332	6-3 6-3

El segundo problema que tuve fue que los datos de los jugadores estaban divididos por sexo, por un lado los masculinos y por otro los femeninos. Primero tuve que crear dos tablas para cada sexo, pero después al intentar unirlos, me di cuenta de que había ID que coincidían en las dos tablas. Como las ID tenían un rango de 100000 a 200000, hice un UPDATE de las ID a una de las tablas:

```
UPDATE Jugador_Femenino  
SET ID_Jugador = ID_Jugador + 100000 ;
```

Antes de ejecutar esta sentencia, tuve que actualizar las claves foráneas del ID de los jugadores en las tablas Partido, para que tuvieran una actualización en cascada.

```
ALTER TABLE Partido_Individual_Femenino ADD FOREIGN KEY (ID_Jug_Ganador)  
REFERENCES Jugador_Femenino(ID_Jugador);
```

```
ALTER TABLE Partido_Individual_Femenino ADD FOREIGN KEY (ID_Jug_Perdedor)  
REFERENCES Jugador_Femenino(ID_Jugador);
```

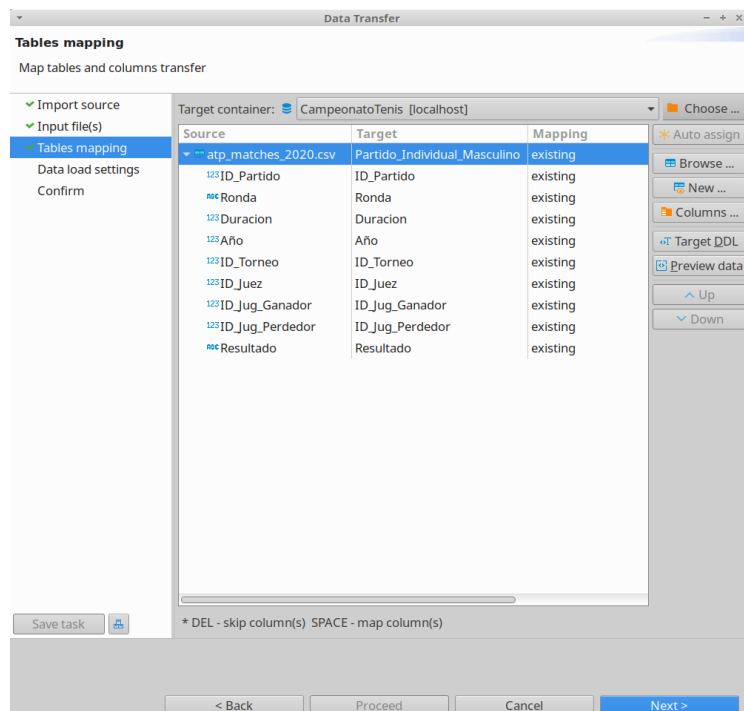
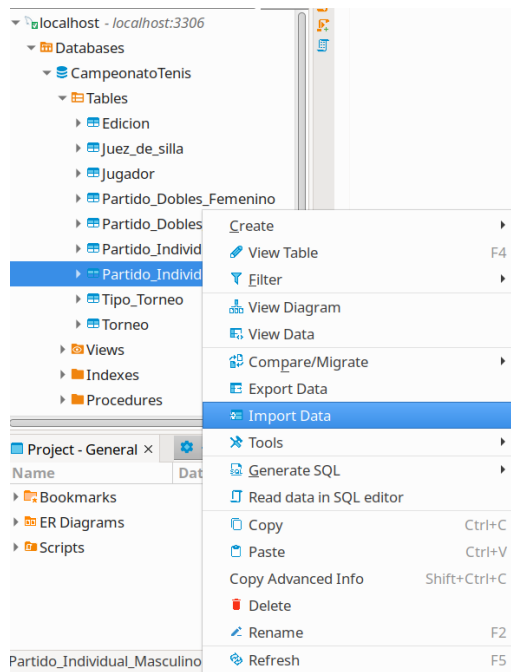
De esta manera, ya no había ID que coincidieran en ambas tablas, por lo tanto ya podía unirlos con un UNION.

```
CREATE TABLE Jugador as  
select * from Jugador_Masculino jm  
union  
select * from Jugador_Femenino jf ;
```

Una vez unidos todos los jugadores, añadí una columna para el sexo y la actualicé teniendo en cuenta el intervalo de las ID de los jugadores.

```
alter table Jugador add column sexo enum('M','F');  
  
update Jugador  
set sexo = 'M'  
WHERE ID_Jugador >= 100000 AND ID_Jugador < 300000;  
  
update Jugador  
set sexo = 'F'  
where sexo IS NULL;
```

Por último, cargué los datos en las tablas a través de DBeaver con la opción de 'Import data'.



	ID_Partido	Ronda	Duración	Año	ID_Torneo	ID_Juez	ID_Jug_Ganador	ID_Jug_Perdedor	Resultado
1	100	R128	122	2,020	1	[NULL]	104,745	106,198	6-2 6-3 6-0
2	100	R128	118	2,020	4	[NULL]	104,925	106,000	6-1 6-4 6-1
3	101	R128	139	2,020	1	[NULL]	105,643	105,311	6-3 6-4 7-6(3)
4	101	R128	167	2,020	4	[NULL]	106,378	122,330	2-6 7-5 7-5 6-0
5	101	F	97	2,020	36	[NULL]	200,005	200,282	6-1 7-6(4)
6	102	R128	125	2,020	1	[NULL]	105,376	111,153	7-6(1) 6-3 4-6 6-0
7	102	R128	154	2,020	4	[NULL]	111,581	105,311	6-2 7-5 2-6 6-1
8	102	SF	168	2,020	36	[NULL]	200,282	105,777	7-6(4) 6-7(3) 6-4
9	103	R128	157	2,020	1	[NULL]	105,807	106,075	6-4 3-6 6-1 7-6(2)
10	103	R128	131	2,020	4	[NULL]	105,526	124,079	6-0 7-5 6-4
11	103	SF	192	2,020	36	[NULL]	200,005	105,554	4-6 7-6(7) 6-4
12	104	R128	133	2,020	1	[NULL]	106,401	132,283	6-2 7-6(3) 7-6(1)
13	104	R128	201	2,020	4	[NULL]	105,807	106,034	4-6 6-3 1-6 6-3 6-3
14	104	QF	58	2,020	36	[NULL]	200,282	106,218	6-3 6-0
15	105	R128	109	2,020	1	[NULL]	104,468	104,655	6-1 6-3 6-3
16	105	R128	157	2,020	4	[NULL]	106,283	105,155	3-6 6-2 7-5 6-3
17	105	QF	0	2,020	36	[NULL]	105,777	105,683	W/O
18	106	R128	119	2,020	1	[NULL]	144,707	106,034	6-4 6-1 6-2
19	106	R128	186	2,020	4	[NULL]	105,575	105,943	7-6(3) 4-6 6-4 6-4
20	106	QF	160	2,020	36	[NULL]	105,554	111,575	3-6 7-6(7) 6-4
21	107	R128	170	2,020	1	[NULL]	111,575	106,408	4-6 6-4 7-6(4) 6-3
22	107	R128	230	2,020	4	[NULL]	105,449	104,545	6-7(5) 6-3 6-7(5) 6-3 7-6(3)
23	107	QF	95	2,020	36	[NULL]	144,750	200,005	6-3 7-6(2)
24	108	R128	104	2,020	1	[NULL]	104,792	104,229	6-1 6-4 6-2

5. Realización de consultas

Consulta 1: Partidos ganados por Jugador

```
CREATE VIEW PartidosGanadosHombres AS
SELECT j.Nombre , j.Apellidos , COUNT(*) as partidos_ganados
FROM Partido Individual_Masculino pim
INNER JOIN Jugador j
ON pim.ID_Jug_Ganador = j.ID_Jugador
GROUP BY j.Nombre , j.Apellidos
ORDER BY 3 DESC ;

SELECT * FROM PartidosGanadosHombres ;
```

	Nombre	Apellidos	partidos_ganados
1	Novak	Djokovic	41
2	Andrey	Rublev	41
3	Alexander	Zverev	28
4	Daniil	Medvedev	28
5	Stefanos	Tsitsipas	28
6	Rafael	Nadal	27
7	Diego	Schwartzman	25
8	Dominic	Thiem	25
9	Ugo	Humbert	24
10	Milos	Raonic	23

```
CREATE VIEW PartidosGanadosMujeres AS
SELECT j.Nombre , j.Apellidos , COUNT(*) as partidos_ganados
FROM Partido Individual_Femenino pif
INNER JOIN Jugador j
ON pif.ID_Jug_Ganador = j.ID_Jugador
GROUP BY j.Nombre , j.Apellidos
ORDER BY 3 DESC ;

SELECT * FROM PartidosGanadosMujeres ;
```

	Nombre	Apellidos	partidos_ganados
1	Elise	Mertens	32
2	Elena	Rybakina	29
3	Aryna	Sabalenka	28
4	Simona	Halep	23
5	Garbine	Muguruza	23
6	Sofia	Kenin	23
7	Ons	Jabeur	21
8	Petra	Kvitova	20
9	Anett	Kontaveit	20
10	Maria	Sakkari	20

Consulta 2: Nombres de los jugadores y resultado del partido más largo del 2020

```
SELECT j.Nombre , j.Apellidos , j2.Nombre , j2.Apellidos , pim.Duracion , pim.Resultado
FROM Partido_Individual_Masculino pim
INNER JOIN Jugador j
ON pim.ID_Jug_Ganador = j.ID_Jugador
INNER JOIN Jugador j2
ON pim.ID_Jug_Perdedor = j2.ID_Jugador
WHERE pim.Duracion = (
    SELECT MAX(p2.Duracion)
    FROM Partido_Individual_Masculino p2
);
```

	Nombre	Apellidos	Nombre	Apellidos	Duracion	Resultado
1	Lorenzo	Giustino	Corentin	Moutet	365	0-6 7-6(7) 7-6(3) 2-6 18-16

Consulta 3: Partidos agrupados por el número de sets jugados

```
CREATE VIEW PartidosPorSets AS
SELECT LENGTH(Resultado) - LENGTH(REPLACE(Resultado, '-', '')) as num_sets , COUNT(*) partidos
FROM Partido_Individual_Masculino pim
GROUP BY num_sets
ORDER BY 2 DESC ;

SELECT * FROM PartidosPorSets ;
```

	num_sets	partidos
1	2	606
2	3	552
3	4	104
4	5	75
5	0	6
6	1	5

Consulta 4: Jugadoras femeninas agrupadas por nacionalidad que han ganado algún partido

```
SELECT j.Nacionalidad , COUNT(*)
FROM Partido_Individual_Femenino pif
INNER JOIN Jugador j
ON pif.ID_Jug_Ganador = j.ID_Jugador
GROUP BY j.Nacionalidad
ORDER BY 2 DESC ;
```

	Nacionalidad	partidos_ganados
1	USA	162
2	CZE	92
3	RUS	84
4	BLR	61
5	ROU	50
6	FRA	49
7	KAZ	49
8	BEL	45
9	UKR	43
10	CHN	41
11	ESP	39

Consulta 5: Partidos en los que ha habido algún roscó (6-0 ó 0-6)

```
SELECT COUNT(*)
FROM Partido_Individual_Masculino pim
WHERE pim.Resultado LIKE '%6-0%' OR pim.Resultado LIKE '%0-6%';

SELECT COUNT(*)
FROM Partido_Individual_Femenino pif
WHERE pif.Resultado LIKE '%6-0%' OR pif.Resultado LIKE '%0-6%';
```

	123 COUNT(*)
1	90

	123 COUNT(*)
1	104

6. Funciones

Primera función: Calcular la ronda máxima que ha ganado un jugador en un torneo y año determinado

```
DELIMITER $$

CREATE FUNCTION calcularRondaMaxima(codJugador INT, idTorneo INT, anyo INT) RETURNS VARCHAR(5)
DETERMINISTIC

BEGIN
    DECLARE ronda VARCHAR(5) DEFAULT '';

    SELECT pim.Ronda INTO ronda
    FROM Partido_Individual_Masculino pim
    WHERE pim.ID_Jug_Ganador = codJugador AND
    pim.ID_Torneo = idTorneo AND
    pim.Año = anyo
    ORDER BY pim.ID_Partido DESC LIMIT 1;

    RETURN ronda;

END$$

DELIMITER ;
```

The screenshot shows a database IDE interface. At the top, a SQL query is entered: `SELECT calcularRondaMaxima (104745,2,2020) ronda;`. Below the query editor, the results are displayed in a table with one row and one column labeled 'ronda', containing the value 'F'. The interface includes a sidebar with icons for Grid, Text, and Record views, and a bottom toolbar with buttons for Save, Cancel, Script, and various navigation and execution tools.

Segunda función: Calcular los puntos que corresponden a una ronda para un torneo

```
CREATE FUNCTION calcularPuntosRonda(idTorneo INT, ronda VARCHAR(5)) RETURNS INT
DETERMINISTIC
```

```
BEGIN
```

```
    DECLARE puntos INT DEFAULT 0;
    DECLARE puntosTorneo INT DEFAULT 0;

    SELECT tt.Puntos INTO puntosTorneo
    FROM Torneo t
    INNER JOIN Tipo_Torneo tt
    ON t.ID_Tipo_Torneo = tt.ID_Tipo_Torneo
    WHERE t.ID_Torneo = idTorneo ;

    CASE ronda
        WHEN 'F' THEN SET puntos = puntosTorneo ;
        WHEN 'SF' THEN SET puntos = puntosTorneo * 0.6;
        WHEN 'QF' THEN SET puntos = puntosTorneo * 0.36;
        WHEN 'R16' THEN SET puntos = puntosTorneo * 0.18;
        WHEN 'R32' THEN SET puntos = puntosTorneo * 0.09;
        WHEN 'R64' THEN SET puntos = puntosTorneo * 0.045;
        WHEN 'R128' THEN SET puntos = puntosTorneo * 0.0225;
    ELSE
        SET puntos = 0;
    END CASE;
```

```
RETURN puntos;
```

```
END$$
```

```
SELECT calcularPuntosRonda (1, 'F'); -- Australian Open
```

results 1 ×

SELECT calcularPuntosRonda (1, 'F') | Enter a SQL expression to filter results (use Ctrl+Space)

123	calcularPuntosRonda (1, 'F')
1	2,000

```
SELECT calcularPuntosRonda (22, 'SF'); -- Cincinnati
```

results 1 ×

SELECT calcularPuntosRonda (22, 'SF') | Enter a SQL expression to filter results (use Ctrl+Space)

123	calcularPuntosRonda (22, 'SF')
1	600

Tercera función: Calcular los puntos de un jugador

```
CREATE FUNCTION calcularPuntosJugador(codJugador INT) RETURNS INT DETERMINISTIC
BEGIN
    DECLARE total_puntos INT DEFAULT 0;
    DECLARE torneo_actual INT DEFAULT 0;
    DECLARE ronda_actual VARCHAR(5) DEFAULT '';
    DECLARE salida INT DEFAULT FALSE;
    DECLARE curl CURSOR FOR
        SELECT pim.ID_Torneo , calcularRondaMaxima (codJugador, pim.ID_Torneo, pim.Año)
        FROM Partido_Individual_Masculino pim
        WHERE pim.ID_Jug_Ganador = codJugador OR pim.ID_Jug_Perdedor = codJugador
        GROUP BY pim.ID_Torneo , pim.Año
        ORDER BY pim.ID_Torneo ;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET salida = TRUE;

    OPEN curl;

    leer_cursor: LOOP
        FETCH curl INTO torneo_actual , ronda_actual;

        IF (salida) THEN
            LEAVE leer_cursor;
        END IF;

        SET total_puntos = total_puntos + calcularPuntosRonda (torneo_actual, ronda_actual);
    END LOOP;


    CLOSE curl;

    RETURN total_puntos;

END$$
```

```
SELECT calcularPuntosJugador (104925);
```

Results 1 ×

SELECT calcularPuntosJugador (104925) |  Enter a SQL expression

123	calcularPuntosJugador (104925)	
1		7,470

7. Procedimientos

Primer procedimiento: Actualizar los puntos de cada jugador

```
DELIMITER $$

CREATE PROCEDURE actualizarPuntosJugadores()

BEGIN

    UPDATE Rankings
    SET puntos = calcularPuntosJugador (ID_Jugador);

END$$

DELIMITER ;

CALL actualizarPuntosJugadores ;
```

Resultado:

Rankings <small>Enter a SQL expression to filter results (use Ctrl+Space)</small>			
Grid	ID_Jugador	puntos	
1	104,925	7,470	
2	106,233	4,640	
3	104,745	4,300	
4	106,421	4,278	
5	100,644	3,245	
6	126,094	2,965	
7	105,138	2,355	
8	126,774	2,200	
9	106,043	2,160	
10	105,807	1,665	
11	105,683	1,613	
12	105,583	1,350	
13	111,575	1,260	
14	106,426	1,200	
15	105,554	1,193	

Segundo procedimiento: Calcular los ganadores de cada torneo para un año e insertarlos en la tabla Ganadores

Primero creamos la tabla Ganadores que almacenará los ganadores para cada torneo:

```
CREATE TABLE Ganadores (
    id INT PRIMARY KEY AUTO_INCREMENT,
    anyo INT,
    torneo VARCHAR(45),
    nombre VARCHAR(45),
    apellido VARCHAR(45)
);

DELIMITER $$

CREATE PROCEDURE insertarGanadores(anyo INT)
BEGIN
    DECLARE salida INT DEFAULT FALSE;
    DECLARE nom_jug VARCHAR(100) DEFAULT '';
    DECLARE apel_jug VARCHAR(100) DEFAULT '';
    DECLARE cod_jug INT DEFAULT 0;
    DECLARE id_torneo VARCHAR(100) DEFAULT '';
    DECLARE nom_torneo VARCHAR(100) DEFAULT '';
    DECLARE cur1 CURSOR FOR
        SELECT pim.ID_Torneo , pim.ID_Jug_Ganador
        FROM Partido_Individual_Masculino pim
        WHERE pim.Ronda = 'F' AND pim.Año = anyo
        ORDER BY pim.ID_Torneo ;
    DECLARE cur2 CURSOR FOR
        SELECT pif.ID_Torneo , pif.ID_Jug_Ganador
        FROM Partido_Individual_Femenino pif
        WHERE pif.Ronda = 'F' AND pif.Año = anyo
        ORDER BY pif.ID_Torneo ;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET salida = TRUE;

    OPEN cur1;

    leer_cursor: LOOP
        FETCH cur1 INTO id_torneo , cod_jug ;

        IF (salida) THEN
            LEAVE leer_cursor ;
        END IF;

        SELECT j.Nombre , j.Apellidos INTO nom_jug , apel_jug
        FROM Jugador j
        WHERE j.ID_Jugador = cod_jug ;
```



```

SELECT t.Nombre INTO nom_torneo
FROM Torneo t
WHERE t.ID_Torneo = id_torneo ;

INSERT INTO Ganadores (anyo, torneo, nombre, apellido)
VALUES (anyo, nom_torneo, nom_jug, apel_jug);
END LOOP;

CLOSE cur1;
SET salida = FALSE;

OPEN cur2;

leer_cursor: LOOP
    FETCH cur2 INTO id_torneo , cod_jug ;

    IF (salida) THEN
        LEAVE leer_cursor ;
    END IF;

    SELECT j.Nombre , j.Apellidos INTO nom_jug , apel_jug
    FROM Jugador j
    WHERE j.ID_Jugador = cod_jug ;

    SELECT t.Nombre INTO nom_torneo
    FROM Torneo t
    WHERE t.ID_Torneo = id_torneo ;

    INSERT INTO Ganadores (anyo, torneo, nombre, apellido)
    VALUES (anyo, nom_torneo, nom_jug, apel_jug);
END LOOP;

CLOSE cur2;

END$$

DELIMITER ;

```

Resultado:

	id	anyo	torneo	nombre	apellido	
1	1	2,020	Australian Open	Novak	Djokovic	
2	2	2,020	Roland Garros	Rafael	Nadal	
3	3	2,020	US Open	Dominic	Thiem	
4	4	2,020	Doha	Andrey	Rublev	
5	5	2,020	Adelaide	Andrey	Rublev	
6	6	2,020	Auckland	Ugo	Humbert	
7	7	2,020	Cordoba	Cristian	Garin	
8	8	2,020	Montpellier	Gael	Monfils	
9	9	2,020	Pune	Jiri	Vesely	
10	10	2,020	Buenos Aires	Casper	Ruud	
11	11	2,020	New York	Kyle	Edmund	
12	12	2,020	Delray Beach	Reilly	Opelka	
13	13	2,020	Rotterdam	Gael	Monfils	
14	14	2,020	Marseille	Stefanos	Tsitsipas	
15	15	2,020	Rio	Cristian	Garin	
16	16	2,020	Acapulco	Rafael	Nadal	
17	17	2,020	Dubai	Novak	Djokovic	
18	18	2,020	Santiago	Thiago	Seyboth Wild	
19	19	2,020	Cincinnati	Novak	Djokovic	
20	20	2,020	Kitzbuhel	Miomir	Kecmanovic	
21	21	2,020	Rome	Novak	Djokovic	
22	22	2,020	Hamburg	Andrey	Rublev	
23	23	2,020	Cologne 1	Alexander	Zverev	
24	24	2,020	Sardinia	Laslo	Djere	

	id	anyo	torneo	nombre	apellido
36	36	2,020	Roland Garros	Iga	Swiatek
37	37	2,020	US Open	Naomi	Osaka
38	38	2,020	Doha	Aryna	Sabalenka
39	39	2,020	Adelaide	Ashleigh	Barty
40	40	2,020	Auckland	Serena	Williams
41	41	2,020	Acapulco	Heather	Watson
42	42	2,020	Dubai	Simona	Halep
43	43	2,020	Cincinnati	Victoria	Azarenka
44	44	2,020	Rome	Simona	Halep
45	45	2,020	St Petersburg	Kiki	Bertens
46	46	2,020	Brisbane	Karolina	Pliskova
47	47	2,020	Shenzhen	Ekaterina	Alexandrova
48	48	2,020	Hobart	Elena	Rybakina
49	49	2,020	Hua Hin	Magda	Linette
50	50	2,020	Lyon	Sofia	Kenin
51	51	2,020	Monterrey	Elina	Svitolina
52	52	2,020	Palermo	Fiona	Ferro
53	53	2,020	Lexington	Jennifer	Brady
54	54	2,020	Prague	Simona	Halep
55	55	2,020	Istanbul	Patricia Maria	Tig
56	56	2,020	Strasbourg	Elina	Svitolina
57	57	2,020	Ostrava	Aryna	Sabalenka
58	58	2,020	Linz	Aryna	Sabalenka

Tercer procedimiento: Mostrar los enfrentamientos entre dos jugadores

DELIMITER \$\$

CREATE PROCEDURE mostrarEnfrentamientos(jugador1 INT, jugador2 INT)

BEGIN

```

SELECT *
FROM Partido_Individual_Masculino pim
WHERE pim.ID_Jug_Ganador IN (jugador1, jugador2) AND
pim.ID_Jug_Perdedor IN (jugador1, jugador2);

```

END\$\$

DELIMITER ;

CALL mostrarEnfrentamientos (104745, 104925);

	ID_Partido	Ronda	Duracion	Año	ID_Torneo	ID_Juez	ID_Jug_Ganador	ID_Jug_Perdedor	Resultado
1	300	F	115	2,020	29	[NULL]	104,925	104,745	6-2 7-6(4)
2	1,701	F	161	2,020	2	[NULL]	104,745	104,925	6-0 6-2 7-5

8. Triggers

Primer trigger: Actualizar las victorias y derrotas de cada jugador al insertar un partido

```
ALTER TABLE Rankings ADD COLUMN victorias INT DEFAULT 0;
ALTER TABLE Rankings ADD COLUMN derrotas INT DEFAULT 0;

DELIMITER $$

CREATE TRIGGER actualizarVictoriasYDerrotas
AFTER INSERT ON Partido_Individual_Masculino FOR EACH ROW
BEGIN
    UPDATE Rankings
    SET victorias = victorias + 1
    WHERE ID_Jugador = NEW.ID_Jug_Ganador ;

    UPDATE Rankings
    SET derrotas = derrotas + 1
    WHERE ID_Jugador = NEW.ID_Jug_Perdedor ;

END$$

DELIMITER ;

INSERT INTO Partido_Individual_Masculino
VALUES(1702, 'F', 248, 2021, 2, NULL, 104925, 126774, '6-7 2-6 6-3 6-2 6-4');
```

Properties Data ER Diagram					
Rankings ID_Jugador = 104925 OR ID_Jugador = 126774					
Grid		123 ID_Jugador	123 puntos	123 victorias	123 derrotas
	1	104,925	7,470	1	0
Text	2	126,774	2,200	0	1

Segundo trigger: Comprobar que al insertar un partido no haya dos finales para un mismo torneo

```
DELIMITER $$

CREATE TRIGGER dosFinales
BEFORE INSERT ON Partido_Individual_Masculino FOR EACH ROW

BEGIN

    DECLARE existe BOOL;


    SELECT COUNT(*) INTO existe
    FROM Partido_Individual_Masculino pim
    WHERE pim.Año = NEW.Año AND
    pim.ID_Torneo = NEW.ID_Torneo AND
    pim.Ronda = 'F';

    IF (existe) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No pueden haber dos finales para un mismo torneo';
    END IF;

END$$

DELIMITER ;

INSERT INTO Partido_Individual_Masculino
VALUES(1702, 'F', 161, 2021, 2, NULL, 104745, 104925, '6-0 6-2 7-5');
```

Results 1	Output
INSERT INTO Partido_Individual_Masculino VAL Enter a SQL expression to filter results (us	
 SQL Error [1644] [45000]: No pueden haber dos finales para un mismo torneo	

9. Valoración personal

Considero que ha sido un proyecto interesante ya que lo he realizado sobre un tema que me gusta personalmente y además me ha servido para repasar todo lo que se ha dado en el módulo de bases de datos.

Por otro lado, el proyecto me ha ayudado a tener en cuenta los posibles problemas que podemos encontrarnos, sobre todo a la hora de cargar los datos cuando ya hemos hecho el paso a tablas.

Pienso que la mejor opción para cargar los datos es cargarlos a través de un archivo .csv para que el programa genere las tablas automáticamente, y luego crear todas las relaciones necesarias (claves primarias y foráneas) una vez tengamos los datos cargados.