Trabajo de prácticas BDGE

LUIS ARDÉVOL MESA, MIGUEL MATO MARTÍNEZ

Usando una conjunto de datos de partidos de tenis, se usarán distintos tipos de base de datos para realizar consultas acerca de los mismos. En concreto, se tratarán bases de datos relaciones y con datos agregados en PostgreSQL, bases de datos distribuidas con SQL usando CITUS Data, y dos tipos de bases de datos noSQL: documentales con MongoDB y en grafo con Neo4j.

1. BASES DE DATOS RELACIONALES (POSTGRESQL)

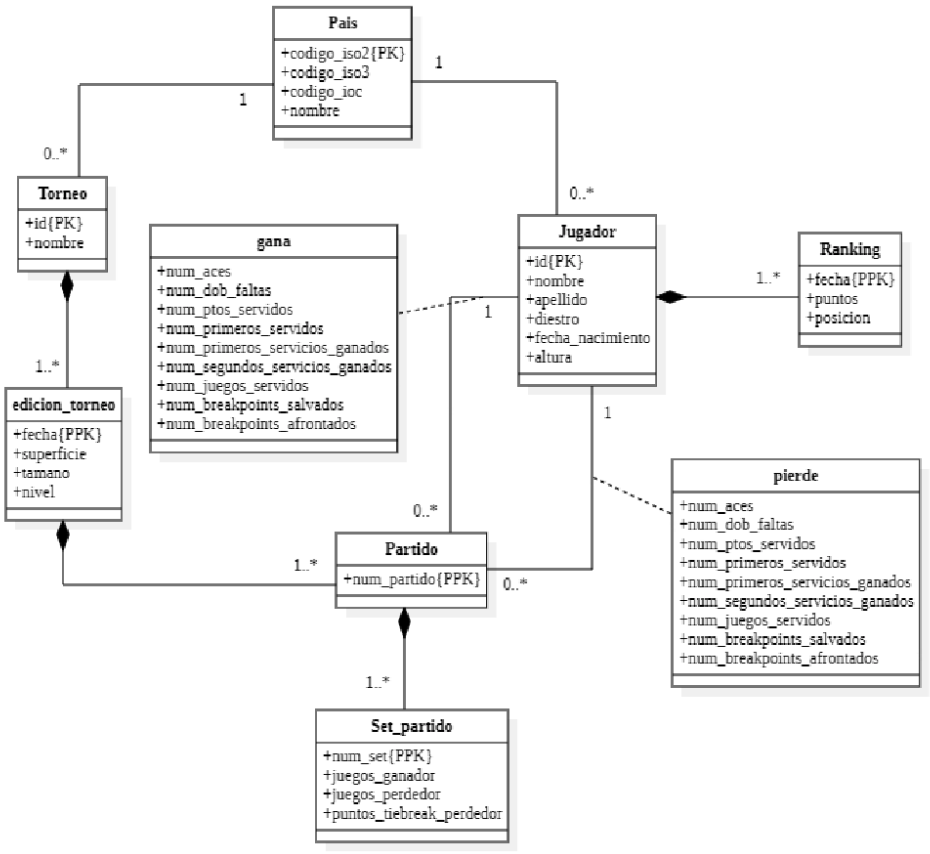

Fig. 1. Esquema de la base de datos de partidos de tenis.

En la figura **??** se muestra el esquema de la base de datos de partidos de tenis. Comenzaremos explicando este esquema de forma breve, pero haciendo especial hincapié en las relaciones entre las distintas tablas, ya que esto ayudará a comprender mejor las condiciones de *join* a usar en las consultas que usen este esquema. Destacamos que no se hará uso de la tabla `ranking`, por lo que no se comentará en el siguiente análisis.

La tabla `pais` contiene la información relativa a países, y se usará tanto para mostrar el país dónde se celebra un torneo como para referenciar el país de procedencia de un jugador; su clave primaria es `codigo_iso2`. La tabla `jugador` contiene la información de los jugadores, y su clave primaria es el `id` de cada jugador; el atributo `pais` referencia a `pais(codigo_iso2)`. La tabla `torneo` contiene la información de los torneos; su clave primaria es el `id` de cada torneo y el atributo `pais` referencia a `pais(codigo_iso2)`. La tabla `edicion_torneo` contiene la información de las ediciones de los torneos; su clave primaria es una combinación de los atributos `torneo` y `fecha`, donde el atributo `torneo` es una referencia a `torneo(id)`. La tabla `partido` contiene la información de los partidos disputados; su clave primaria es la combinación de los atributos `torneo`, `fecha` y `num_partido`, y tiene una clave foránea compuesta por los atributos `torneo` y `fecha`, que referencia a la clave primaria de `edicion_torneo`, `edicion_torneo(torneo, fecha)`. Además, los atributos `ganador` y `perdedor` son los `id` de los jugadores, por lo que puede enlazarse esta tabla con la de `jugador` mediante `jugador(id)`. Por último, la tabla `sets_partido` contiene la información de los sets de los partidos; su clave primaria es la combinación de los atributos `torneo`, `fecha`, `num_partido` y `num_set`, y tiene una clave foránea compuesta por los atributos `torneo`, `fecha` y `num_partido`, que referencia a la clave primaria de `partido`, `partido(torneo, fecha, num_partido)`.

Tras ver cómo se estructura la base de datos a usar, creamos la base de datos en PostgreSQL y cargamos los datos en ella. La estructura relacional se proporciona en el archivo `schema.sql`, y los datos en los archivos `pais.csv`, `jugador.csv`, `torneo.csv`, `edicion_torneo.csv`, `partido.csv`, `sets_partido.csv` y `ranking.csv`. Una vez creada la base de datos `tenis`, ejecutamos el archivo `schema.sql` con la definición de los esquemas de las tablas mediante el siguiente comando en la terminal:

```
psql -U alumnogreibd -d tenis -f
    /home/alumnogreibd/BDGE/datos/datos_tenis/schema.sql
```

Ahora, con los esquemas definidos (pero vacíos), cargamos los datos a partir de los archivos `.csv` con los siguientes comandos en terminal (es importante seguir el orden de carga de los datos debido a las dependencias entre las tablas):

```
psql -U alumnogreibd -d tenis -c "\copy pais from
    /home/alumnogreibd/BDGE/datos/datos_tenis/pais.csv csv"
psql -U alumnogreibd -d tenis -c "\copy jugador from
    /home/alumnogreibd/BDGE/datos/datos_tenis/jugador.csv csv"
psql -U alumnogreibd -d tenis -c "\copy torneo from
    /home/alumnogreibd/BDGE/datos/datos_tenis/torneo.csv csv"
psql -U alumnogreibd -d tenis -c "\copy edicion_torneo from
    /home/alumnogreibd/BDGE/datos/datos_tenis/edicion_torneo.csv csv"
psql -U alumnogreibd -d tenis -c "\copy partido from
    /home/alumnogreibd/BDGE/datos/datos_tenis/partido.csv csv"
psql -U alumnogreibd -d tenis -c "\copy sets_partido from
    /home/alumnogreibd/BDGE/datos/datos_tenis/sets_partido.csv csv"
psql -U alumnogreibd -d tenis -c "\copy ranking from
    /home/alumnogreibd/BDGE/datos/datos_tenis/ranking.csv csv"
```

Este orden es lógico por las dependencias entre tablas que comentamos anteriormente: `sets_partido` referencia a `partido`, por lo que es necesario cargar primero los datos de `partido` para poder cargar los de `sets_partido`. Este razonamiento aplica al resto de tablas: `partido` referencia a `edicion_torneo`, que referencia a `torneo`, que a su vez referencia a `pais`, que es referenciado por `jugador`.

En las siguientes consultas, los *join* se harán mediante *theta join* (*inner* por defecto). En caso de que se quiera

hacer otro tipo de *join*, se especificará de forma clara en la consulta. Esto considera el producto cartesiano de las tablas a unir y selecciona solo las tuplas que verifiquen el predicado (la condición o condiciones de *join*). Estos *join* se harán de forma general de forma implícita en la claúsula `where`, siguiendo los atributos referenciados entre las tablas. Además, trabajaremos siempre dentro del mismo esquema.

.1. Muestra todos los ganadores del torneo "Wimbledon" (Nombre, apellidos y año). Ordena el resultado por año.

Para esta consulta necesitamos la información de los jugadores, los partidos y los torneos, por lo que comenzamos en el `from` con el producto cartesiano de las tablas `jugador`, `partido` y `torneo` (especificando un alias para cada una de ellas). A continuación, especificamos las condiciones de *join* en el `where`, así como otras condiciones de selección:

- Condiciones de *join*:
  - El predicado `j.id = p.ganador` sirve para unir la tabla `jugador` con la tabla `partido` mediante el atributo `id` de la tabla `jugador` y el atributo `ganador` de la tabla `partido`, haciendo una selección únicamente de las tuplas que cumplan esta condición. Esto nos permite, para cada jugador, seleccionar solo los partidos en los que ha ganado.
  - El predicado `t.id = p.torneo` sirve para unir la tabla `torneo` con la tabla `partido` mediante el atributo `id` de la tabla `torneo` y el atributo `torneo` de la tabla `partido`, haciendo una selección únicamente de las tuplas que cumplan esta condición. Esto nos permite seleccionar solo los partidos que se han jugado en un torneo concreto.
- Condiciones de selección:
  - La condición `t.nombre = 'Wimbledon'` selecciona solo el torneo cuyo nombre es "Wimbledon". Al hacer el *join* con la tabla `torneo`, estamos seleccionando solo los partidos que se han jugado en el torneo "Wimbledon".
  - La condición `p.ronda = 'F'` selecciona solo los partidos que han sido finales.

Tras seleccionar las tuplas que nos interesan, usamos el `select` para obtener la proyección de las filas que nos interesan: en esta caso, nos quedamos con los atributos `nombre` y `apellido` de la tabla `jugador` y el año de la fecha del partido, que denotaremos por el alias "ano". Como `p.fecha` es del tipo `date`, este último atributo lo obtenemos mediante la función `extract()`, especificando que solo queremos el año de ese atributo (`year from p.fecha`). Tras esto, ordenamos el resultado por año con `order by ano`; el orden por defecto es ascendente. El código de la consulta se muestra a continuación, y el resultado se pueden ver en la figura **??**.

```
select j.nombre, j.apellido, extract(year from p.fecha) as ano
from jugador j, partido p, torneo t
where j.id = p.ganador
  and t.id = p.torneo
  and t.nombre = 'Wimbledon'
  and p.ronda = 'F'
order by ano
```

	nombre	apellido	ano
1	Michael	Stich	1.991
2	Andre	Agassi	1.992
3	Pete	Sampras	1.993
4	Pete	Sampras	1.994
5	Pete	Sampras	1.995
6	Richard	Krajicek	1.996
7	Pete	Sampras	1.997
8	Pete	Sampras	1.998
9	Pete	Sampras	1.999
10	Pete	Sampras	2.000
11	Goran	Ivanisevic	2.001
12	Lleyton	Hewitt	2.002
13	Roger	Federer	2.003
14	Roger	Federer	2.004
15	Roger	Federer	2.005
16	Roger	Federer	2.006
17	Roger	Federer	2.007
18	Rafael	Nadal	2.008
19	Roger	Federer	2.009
20	Rafael	Nadal	2.010
21	Novak	Djokovic	2.011
22	Roger	Federer	2.012
23	Andy	Murray	2.013
24	Novak	Djokovic	2.014
25	Novak	Djokovic	2.015
26	Andy	Murray	2.016
27	Roger	Federer	2.017
28	Novak	Djokovic	2.018
29	Novak	Djokovic	2.019
30	Novak	Djokovic	2.021
31	Novak	Djokovic	2.022
32	Carlos	Alcaraz	2.023

Fig. 2. Modelo relacional Postgresql, consulta 1.

.2. Muestra los años en los que Roger Federer ganó algún torneo de nivel Gran Slam (G) o Master 1000 (M). Para cada año, muestra el número de torneos y lista sus nombres (ordenados por la fecha de celebración). Ordena el resultado por el año

Para esta consulta necesitamos información de los jugadores, los partidos, los torneos y las ediciones de los torneos. Comenzamos en el `from` con el producto cartesiano de las tablas `jugador`, `partido`, `torneo` y `edicion_torneo` (especificando un alias para cada una de ellas). A continuación, especificamos las condiciones de *join* en el `where`, así como otras condiciones de selección:

- Condiciones de *join* que, en global, nos permitirán seleccionar las tuplas de un jugador específico que ganó un torneo específico en un año específico:

  – El predicado `j.id = p.ganador` sirve para unir la tabla `jugador` con la tabla `partido` mediante el atributo `id` de la tabla `jugador` y el atributo `ganador` de la tabla `partido`, haciendo una selección únicamente de las tuplas que cumplan esta condición. Esto nos permite, para cada jugador, seleccionar solo los partidos en los que ha ganado.

  – El predicado `t.id = p.torneo` sirve para unir la tabla `torneo` con la tabla `partido` mediante el atributo `id` de la tabla `torneo` y el atributo `torneo` de la tabla `partido`, haciendo una selección únicamente de las tuplas que cumplan esta condición. Esto nos permite seleccionar solo los partidos que se han jugado en un torneo concreto.

  – El predicado `t.id = et.torneo` sirve para unir la tabla `torneo` con la tabla `edicion_torneo` mediante el atributo `id` de la tabla `torneo` y el atributo `torneo` de la tabla `edicion_torneo`, haciendo una selección únicamente de las tuplas que cumplan esta condición. Esto nos permite seleccionar ediciones de torneos concretos.

  – El predicado `p.fecha = et.fecha` sirve para unir la tabla `partido` con la tabla `edicion_torneo` mediante el atributo `fecha` de la tabla `partido` y el atributo `fecha` de la tabla `edicion_torneo`, haciendo una selección únicamente de las tuplas que cumplan esta condición. Esto nos permite

seleccionar solo los partidos que se han jugado en una edición concreta de un torneo (concreto, por la condición anterior).

- Estamos interesados específicamente en torneos de nivel Gran Slam (G) o Master 1000 (M) y en las finales ganadas por Roger Federer (lo que indica que ganó el torneo) durante varios años (es decir, varias ediciones de torneos). Por tanto, las condiciones de selección son:
  - El predicado `p.ronda = 'F'` selecciona solo los partidos que han sido finales.
  - El predicado `j.nombre = 'Roger' and j.apellido = 'Federer'` selecciona solo los partidos en los que ha ganado "Roger Federer".
  - El predicado `et.nivel in ('G', 'M')` selecciona solo los torneos de nivel Gran Slam (G) o Master 1000 (M).

Con el `group by` agrupamos los resultados por el año en que se celebraron los torneos, para su posterior concatenación. Por último, obtenemos la proyección de los atributos que nos interesan: el año, que lo obtenemos extrayendo el año de la fecha de celebración del partido; el número de torneos ganados, que lo obtenemos contando el número de torneos distintos (usamos el `id` del torneo para diferenciar) en el resultado agrupado por año (`count(distinct t.id)`). Como última proyección, obtenemos una concantenación (con `string_agg()`) de esos torneos, separados por comas y ordenados por fecha de celebración de esa edición, en orden ascendente (`order by et.fecha`). El código de la consulta se muestra a continuación, y el resultado se pueden ver en la figura **??**.

```
select extract(year from et.fecha) as ano, count(distinct t.id) as numero_torneos,
       string_agg(t.nombre, ', ' order by et.fecha) as torneos
from jugador j, partido p, torneo t, edicion_torneo et
where j.id = p.ganador
  and t.id = et.torneo
  and p.torneo = t.id
  and p.fecha = et.fecha
  and p.ronda = 'F'
  and j.nombre = 'Roger'
  and j.apellido = 'Federer'
  and et.nivel in ('G', 'M')
group by ano
```

	ano	numero_torneos	torneos
1	2.002	1	Hamburg Masters
2	2.003	1	Wimbledon
3	2.004	6	Australian Open, Indian Wells Masters, Hamburg Masters, Wimbledon, Canada Masters, US Open
4	2.005	6	Miami Masters, Indian Wells Masters, Hamburg Masters, Wimbledon, US Open, Cincinnati Masters
5	2.006	7	Australian Open, Miami Masters, Indian Wells Masters, Wimbledon, US Open, Canada Masters, Madrid Masters
6	2.007	5	Australian Open, Hamburg Masters, Wimbledon, US Open, Cincinnati Masters
7	2.008	1	US Open
8	2.009	4	Roland Garros, Madrid Masters, Wimbledon, Cincinnati Masters
9	2.010	2	Australian Open, Cincinnati Masters
10	2.011	1	Paris Masters
11	2.012	4	Indian Wells Masters, Madrid Masters, Wimbledon, Cincinnati Masters
12	2.014	2	Cincinnati Masters, Shanghai Masters
13	2.015	1	Cincinnati Masters
14	2.017	5	Australian Open, Indian Wells Masters, Miami Masters, Wimbledon, Shanghai Masters
15	2.018	1	Australian Open
16	2.019	1	Miami Masters

Fig. 3. Modelo relacional Postgresql, consulta 2.

.3. Muestra los partidos de semifinales (ronda='SF') y final (ronda = 'F') del torneo de "Roland Garros" del 2018. Para cada partido muestra la ronda, el tipo de desenlace, el nombre y apellidos del ganador y el nombre y apellidos del perdedor y el resultado con el número de juegos del ganador y del perdedor en cada set, y opcionalmente en paréntesis el número de juegos del perdedor en el tie break

Para esta consulta necesitamos información de los partidos, los jugadores, los torneos y los sets de los partidos; como debemos distinguir entre el ganador y el perdedor, sin ningún tipo de ambigüedad, en el `from` usamos dos instancias de la tabla `jugador`, cada una con un alias distinto, por lo que estaremos haciendo el producto cartesiano de la tabla `jugador` consigo misma, así como con las tablas `partido`, `torneo` y `sets_partido` (especificando un alias para cada una de ellas). A continuación, especificamos las condiciones de *join* en el `where`, así como otras condiciones de selección:

- Condiciones de *join* que, en global, nos permitirán seleccionar las tuplas de partidos específicos de un torneo específico en una fecha específica:
  - El predicado `jg.id = p.ganador` sirve para unir la tabla `jugador` con la tabla `partido` mediante el atributo `id` de la tabla `jugador` y el atributo `ganador` de la tabla `partido`, haciendo una selección únicamente de las tuplas que cumplan esta condición. Esto nos permite, para cada jugador, seleccionar solo los partidos en los que ha ganado.
  - Del mismo modo, usamos el predicado `jp.id = p.perdedor` para eleccionar solo los partidos en los que ha perdido un determinado jugador.
  - El predicado `p.fecha = sp.fecha` sirve para unir la tabla `partido` con la tabla `sets_partido` mediante el atributo `fecha` de la tabla `partido` y el atributo `fecha` de la tabla `sets_partido`, haciendo una selección únicamente de las tuplas que cumplan esta condición. Esto nos permite seleccionar solo los sets de los partidos que se han jugado en una fecha concreta.
  - Como la fecha anterior no es unívoca para cada partido, necesitamos una condición adicional para obtener los sets de un partido determinado. Para ello, usamos el predicado `p.num_partido = sp.num_partido`, que une la tabla `partido` con la tabla `sets_partido` mediante el atributo `num_partido` de la tabla `partido` y el atributo `num_partido` de la tabla `sets_partido`, haciendo una selección únicamente de las tuplas que cumplan esta condición.
  - El predicado `t.id = p.torneo` sirve para unir la tabla `torneo` con la tabla `partido` mediante el atributo `id` de la tabla `torneo` y el atributo `torneo` de la tabla `partido`, haciendo una selección únicamente de las tuplas que cumplan esta condición. Esto nos permite seleccionar solo los partidos que se han jugado en un torneo concreto.
- Concretamente, estamos interesados en las semifinales (ronda='SF') y la final (ronda='F) del torneo de "Roland Garros" del 2018. Por tanto, las condiciones de selección son:
  - El predicado `p.ronda in ('SF', 'F')` selecciona solo los partidos que han sido semifinales o finales.
  - El predicado `t.nombre = 'Roland Garros'` selecciona solo los partidos que se han jugado en el torneo "Roland Garros".
  - El predicado `extract(year from p.fecha) = '2018'` selecciona solo los partidos que se han jugado en el año 2018.

Ahora, para cada combinación de `p.ronda`, `p.desenlace`, `jg.nombre`, `jg.apellido`, `jp.nombre` y `jp.apellido`, tendremos una fila para cada set, donde solo variará la información de los juegos ganados y perdidos. Por tanto, agrupamos por los atributos mencionados anteriormente y, en la proyección, concatenamos los resultados

de los juegos ganados y perdidos en cada set, así como el resultado final del partido, para lo que usamos la función `string_agg()`. Como también debemos especificar la puntuación del *tiebreak* (en caso de haberla), usamos un `case` para comprobar si el atributo `puntos_tiebreak_perdedor` es `null` o no: en caso que sea nulo, concatenamos una cadena vacia, en caso de no ser nulo, concatenamos la puntuación del *tiebreak*, puesta entre paréntesis. Si bien esta es una parte de la proyección, el resto incluye simplemente la ronda, el desenlace, la fecha y los nombres y apellidos de los jugadores concatenados (un atributo para el ganador y otro para el perdedor). El código de la consulta se muestra a continuación, y el resultado se pueden ver en la figura **??**.

```
select p.ronda, p.desenlace, jg.nombre || ' ' || jg.apellido AS ganador,
        jp.nombre || ' ' || jp.apellido AS perdedor,
        STRING_AGG(sp.juegos_ganador || '-' || sp.juegos_perdedor ||
                case
                  when sp.puntos_tiebreak_perdedor is not null then
                          '(' || sp.puntos_tiebreak_perdedor || ')'
                  else
                          ''
                end, ', ' order by sp.num_set) as resultado
from partido p, jugador jg, jugador jp, sets_partido sp, torneo t
where jg.id = p.ganador
  and jp.id = p.perdedor
  and p.fecha = sp.fecha
  and p.num_partido = sp.num_partido
  and t.id = p.torneo
  and p.ronda in ('SF', 'F')
  and t.nombre = 'Roland Garros'
  and extract(year from p.fecha) = '2018'
group by p.ronda, p.desenlace, jg.nombre, jg.apellido, jp.nombre, jp.apellido
```

	ronda	desenlace	ganador	perdedor	resultado
1	F	N	Rafael Nadal	Dominic Thiem	6-4, 6-3, 6-2
2	SF	N	Dominic Thiem	Marco Cecchinato	7-5, 7-6(10), 6-1
3	SF	N	Rafael Nadal	Juan Martin del Potro	6-4, 6-1, 6-2

Fig. 4. Modelo relacional Postgresql, consulta 3.

.4. Muestra la lista de jugadores españoles (ES) que ganaron algún torneo de nivel Gran Slam (G). Para cada jugador muestra los siguientes datos resumen de todos sus partidos: número de partidos jugados, porcentaje de victorias, porcentaje de aces, porcentaje de dobles faltas, porcentaje de servicios ganados, porcentaje de restos ganados, porcentaje de break points salvados (de los sufridos en contra), porcentaje de break points ganados (de los provocados a favor)

Esta consulta la dividiremos en dos partes: primero, buscaremos los jugadores españoles que ganaron algún torneo de nivel Grand Slam (G), y, a continuación, calcularemos los datos resumen que se piden de todos sus partidos. Para la primera parte, necesitamos información de los jugadores, los partidos y las ediciones de los torneos; comenzamos en el `from` con el producto cartesiano de las tablas `jugador`, `partido` y `edicion_torneo` (especificando un alias para cada una de ellas). A continuación, especificamos las condiciones de *join* en el `where`, así como otras condiciones de selección:

- Condiciones de *join* que, en global, nos permitirán seleccionar las tuplas de un torneo específico:
  - El predicado `j.id = p.ganador` sirve para unir la tabla `jugador` con la tabla `partido` mediante el atributo `id` de la tabla `jugador` y el atributo `ganador` de la tabla `partido`, haciendo una selección únicamente de las tuplas que cumplan esta condición. Esto nos permite, para cada jugador, seleccionar solo los partidos en los que ha ganado.

  - El predicado `p.torneo = et.torneo` sirve para unir la tabla `partido` con la tabla `edicion_torneo` mediante el atributo `torneo` de la tabla `partido` y el atributo `torneo` de la tabla `edicion_torneo`, haciendo una selección únicamente de las tuplas que cumplan esta condición. Esto nos permite seleccionar solo los partidos que se han jugado en una edición concreta de un torneo.
  - El predicado `p.fecha = et.fecha` sirve para unir la tabla `partido` con la tabla `edicion_torneo` mediante el atributo `fecha` de la tabla `partido` y el atributo `fecha` de la tabla `edicion_torneo`, haciendo una selección únicamente de las tuplas que cumplan esta condición. Esto nos permite seleccionar solo los partidos que se han jugado en una fecha concreta.

- Como de todo lo anterior solo nos interesan los partidos finales (F) de torneos de nivel Grand Slam (G) y ganados por jugadores españoles (ES), las condiciones de selección son:
  - El predicado `p.ronda = 'F'` selecciona solo los partidos que han sido finales.
  - El predicado `et.nivel = 'G'` selecciona solo los torneos de nivel Grand Slam (G).
  - El predicado `j.pais = 'ES'` selecciona solo los jugadores españoles.

Con todo esto, obtenemos una tabla con los jugadores españoles que han ganado algún torneo de nivel Grand Slam (G), pero hay repeticion ya que estamos considerando cada edición de cada Grand Slam. Para quedarnos solo con el subconjunto que nos interesa (sin repetición), usamos un `distinct` en la proyección. Concretamente, obtenemos la proyección del `id` y el nombre completo del jugador, que concatenamos con el operador `||`, y aplicamos el `distinct` sobre la proyección. Esta tabla resultado la guardamos como `jugadores_espanoles_ganadores`, y la usaremos en la segunda parte de la consulta.

Para obtener las estadísticas resumen de cada jugador español ganador de un Grand Slam, necesitamos solo información sobre quiénes son esos jugadores y sobre los partidos que han jugado. Por tanto, en el `from` usamos la tabla `jugadores_espanoles_ganadores` que acabamos de crear, y la tabla `partido` (especificando un alias para cada una de ellas). A continuación, especificamos las condiciones de *join* en el `where`, así como otras condiciones de selección:

- Condiciones de *join* que, en global, nos permitirán seleccionar las tuplas de partidos de un jugador específico:
  - El predicado `jeg.id`$_j ugador = p.ganadororjeg.id_j ugador = p.perdedornospermiteseleccionarlospartidos, tantogana$

Con esto, obtenemos una fila por cada partido de cada jugador, por lo que agrupamos por el jugador antes de agregar valores. En la proyección, obtenemos varios atributos, que redondearemos a un decimal usando `round(..., 1)`:

- El nombre completo del jugador, que concatenamos previamente con el operador `||`.
- El número de partidos jugados como el conteo de los partidos en los que ha participado (`count(p.num_partido)`).
- El porcentaje de victorias, que calculamos como el número de partidos ganados entre el número total de partidos jugados, multiplicado por 100. Para obtener el número de partidos ganados, usamos un `case` que comprueba si el jugador es el ganador del partido o no. Esto genera una lista que contiene 1 si el jugador es el ganador y 0 si no lo es, y con `sum()` sumamos los valores de esta lista para obtener el número de partidos ganados.

```
with jugadores_espanoles_ganadores as (
    select distinct j.id as id_jugador, j.nombre || ' ' || j.apellido as jugador
    from partido p, jugador j, edicion_torneo et
```

```
    where p.ganador = j.id
        and p.torneo = et.torneo
        and p.fecha = et.fecha
        and j.pais = 'ES'
        and p.ronda = 'F'
        and et.nivel = 'G'
)

select jeg.jugador, count(p.num_partido) as partidos,
    round(100.0 * sum(case when jeg.id_jugador = p.ganador then 1 else 0 end) / count(p.num_partido), 1) as
    round(100.0 * sum(case when jeg.id_jugador = p.ganador then p.num_aces_ganador else p.num_aces_perdedor
        nullif(sum(case when jeg.id_jugador = p.ganador then p.num_ptos_servidos_ganador  else p.num_ptos_se
    round(100.0 * sum(case when jeg.id_jugador = p.ganador then p.num_dob_faltas_ganador  else p.num_dob_fal
        nullif(sum(case when jeg.id_jugador = p.ganador then p.num_ptos_servidos_ganador  else p.num_ptos_se
    round(100.0 * sum(case when jeg.id_jugador = p.ganador then p.num_primeros_servicios_ganados_ganador + p
            else p.num_primeros_servicios_ganados_perdedor + p.num_segundos_servicios_ganados_perdedor end)
        nullif(sum(case when jeg.id_jugador = p.ganador then p.num_ptos_servidos_ganador  else p.num_ptos_se
    round(100.0 * sum(case when jeg.id_jugador = p.ganador
            then p.num_ptos_servidos_perdedor - p.num_primeros_servicios_ganados_perdedor - p.num_segundos_s
            else p.num_ptos_servidos_ganador - p.num_primeros_servicios_ganados_ganador - p.num_segundos_ser
        nullif(sum(case when jeg.id_jugador = p.ganador then p.num_ptos_servidos_perdedor else p.num_ptos_se
    round(100.0 * sum(case when jeg.id_jugador = p.ganador then p.num_break_salvados_ganador else p.num_brea
        nullif(sum(case when jeg.id_jugador = p.ganador then p.num_break_afrontados_ganador else p.num_breal
    round(100.0 * sum(case when jeg.id_jugador = p.ganador then p.num_break_afrontados_perdedor - p.num_brea
            else p.num_break_afrontados_ganador - p.num_break_salvados_ganador end) /
        nullif(sum(case when jeg.id_jugador = p.ganador then p.num_break_afrontados_perdedor else p.num_brea
from jugadores_espanoles_ganadores jeg, partido p
where jeg.id_jugador = p.ganador
        or jeg.id_jugador = p.perdedor
group by jeg.jugador
```

.5. Lista los jugadores que fueron derrotados (en algún partido del 2018) por el rival de Rafael Nadal de la primera ronda (R128) de Roland Garros de 2018

```
with rival_nadal as (
        select case when jg.nombre = 'Rafael' then jp.id else jg.id end as id_jugador,
                case when jg.nombre = 'Rafael' then jp.nombre || ' ' || jp.apellido else jg.nombre || ' ' |
        from partido p, jugador jg, jugador jp, edicion_torneo et, torneo t
        where p.ganador = jg.id
                and p.perdedor = jp.id
                and p.torneo = et.torneo
                and p.fecha = et.fecha
                and et.torneo = t.id
                and t.nombre = 'Roland Garros'
                and p.ronda = 'R128'
                and extract(year from p.fecha) = '2018'
                and (jg.nombre = 'Rafael' and jg.apellido = 'Nadal' or jp.nombre = 'Rafael' and jp.apellido
)

select j.nombre || ' ' || j.apellido as jugador, j.pais as pais
from rival_nadal rn, partido p, jugador j
where rn.id_jugador = p.ganador
        and p.perdedor = j.id
        and extract(year from p.fecha) = '2018'
```

2. DATOS AGREGADOS EN SQL (POSTGRESQL)

(un array es homogeneo, una tupla no !!)

A. Tipos compuestos

```
create type pais_info as (
        codigo_iso2 char(2),
        codigo_iso3 char(3),
        codigo_ioc char(3),
        nombre varchar(100)
)

create type torneo_info as (
        id integer,
        nombre varchar(100),
        pais pais_info
)

create type edicion_torneo_info as (
        torneo torneo_info,
        fecha date,
        superficie varchar(20),
        tamano integer,
        nivel char(1)
)

create type set_info as (
        torneo torneo_info, --quitar
        fecha date, --quitar
        num_partido integer, --quitar
    num_set integer,
   juegos_ganador integer,
   juegos_perdedor integer,
   puntos_tiebreak_perdedor integer
)

create type jugador_stats as (
        num_aces integer,
        num_dob_faltas integer,
        num_ptos_servidos integer,
        num_primeros_servicios integer,
        num_primeros_servicios_ganados integer,
        num_segundos_servicios_ganados integer,
        num_juegos_servidos integer,
        num_break_salvados integer,
        num_break_afrontados integer
)

create type jugador_info as (
   id integer,
   nombre varchar(100),
   apellido varchar(100),
   diesto boolean,
   fecha_nacimiento date,
   pais pais_info,
   altura integer
```

```
)
```

```
create table partidos (
  torneo edicion_torneo_info,
  fecha date,
  num_partido integer,
  num_sets integer,
  info_sets set_info array, -- array de elementos de tipo set_info
  ronda varchar(5),
        desenlace char(1),
        ganador jugador_info,
        perdedor jugador_info,
        ganador_stats jugador_stats,
        perdedor_stats jugador_stats
)
```

```
insert into partidos(
torneo, fecha, num_partido, num_sets, info_sets, ronda, desenlace, ganador, perdedor, ganador_stats, perdedo
select
   -- campo 'torneo' (tipo 'torneo_info')
   case
       when t.id is null then null
       else cast((cast((t.id, t.nombre, cast((pa.codigo_iso2, pa.codigo_iso3, pa.codigo_ioc, pa.nombre) as p
       et.fecha, et.superficie, et.tamano, et.nivel) as edicion_torneo_info)
   end as torneo,

   p.fecha as fecha,
   p.num_partido as num_partido,
   p.num_sets as num_sets,

   -- campo 'info_set' (array de 'set_info')
   array(select cast((cast((t.id, t.nombre, cast((pa.codigo_iso2, pa.codigo_iso3, pa.codigo_ioc, pa.nombre)
                       sp.fecha, sp.num_partido, sp.num_set, sp.juegos_ganador, sp.juegos_perdedor, sp.puntos
         from public.sets_partido sp
         where sp.torneo = p.torneo
                 and sp.fecha = p.fecha
                 and sp.num_partido = p.num_partido
         order by sp.num_set) as info_sets,

   p.ronda as ronda,
   p.desenlace as desenlace,

   -- campo 'ganador' (tipo 'jugador_info')
   case
       when jg.id is null then null
       else cast((jg.id, jg.nombre, jg.apellido, jg.diestro, jg.fecha_nacimiento,
                           cast((pg.codigo_iso2, pg.codigo_iso3, pg.codigo_ioc, pg.nombre) as pais_info),
                           jg.altura) as jugador_info)
   end as ganador,

   -- campo 'perdedor' (tipo 'jugador_info')
   case
       when jp.id is null then null
       else cast((jp.id, jp.nombre, jp.apellido, jp.diestro, jp.fecha_nacimiento,
                           cast((pp.codigo_iso2, pp.codigo_iso3, pp.codigo_ioc, pp.nombre) as pais_info),
                           jp.altura) as jugador_info)
```

```
    end as perdedor,

    -- campo 'ganador_stats' (tipo 'jugador_stats')
    case
        when jg.id is null and jp.id is null then null
        else cast((p.num_aces_ganador, p.num_dob_faltas_ganador,
                   p.num_ptos_servidos_ganador, p.num_primeros_servicios_ganador,
                   p.num_primeros_servicios_ganados_ganador, p.num_segundos_servicios_ganados_ganador,
                   p.num_juegos_servidos_ganador, p.num_break_salvados_ganador,
                   p.num_break_afrontados_ganador) as jugador_stats)
    end as ganador_stats,

    -- campo 'perdedor_stats' (tipo 'jugador_stats')
    case
        when jg.id is null and jp.id is null then null
        else cast((p.num_aces_perdedor, p.num_dob_faltas_perdedor,
                   p.num_ptos_servidos_perdedor, p.num_primeros_servicios_perdedor,
                   p.num_primeros_servicios_ganados_perdedor, p.num_segundos_servicios_ganados_perdedor,
                   p.num_juegos_servidos_perdedor, p.num_break_salvados_perdedor,
                   p.num_break_afrontados_perdedor) as jugador_stats)
    end as perdedor_stats
from public.partido p join public.edicion_torneo et on et.fecha = p.fecha and et.torneo = p.torneo
        left join public.torneo t on t.id = et.torneo
        left join public.pais pa on t.pais = pa.codigo_iso2
        left join public.jugador jg on p.ganador = jg.id
        left join public.pais pg on jg.pais = pg.codigo_iso2
        left join public.jugador jp on p.perdedor = jp.id
        left join public.pais pp on jp.pais = pp.codigo_iso2
```

A.1. Muestra todos los ganadores del torneo "Wimbledon" (Nombre apellidos y año). Ordena el resultado por año.

```
select (ganador).nombre, (ganador).apellido, extract(year from fecha) as ano
from partidos
where (torneo).torneo.nombre = 'Wimbledon'
        and ronda = 'F'
order by ano
```

A.2. Muestra los años en los que Roger Federer ganó algún torneo de nivel Gran Slam (G) o Master 1000 (M). Para cada año, muestra el número de torneos y lista sus nombres (ordenados por la fecha de celebración). Ordena el resultado por el año

```
select extract(year from (torneo).fecha) as ano, count(distinct (torneo).torneo.id) as numero_torneos,
        string_agg((torneo).torneo.nombre, ', ' order by (torneo).fecha) as torneos
from partidos
where (torneo).nivel in ('G', 'M')
        and (ganador).nombre = 'Roger'
        and (ganador).apellido = 'Federer'
        and ronda = 'F'
group by ano
```

A.3. Muestra los partidos de semifinales (ronda='SF') y final (ronda = 'F') del torneo de "Roland Garros" del 2018. Para cada partido muestra la ronda, el tipo de desenlace, el nombre y apellidos del ganador y el nombre y apellidos del perdedor y el resultado con el número de juegos del ganador y del perdedor en cada set, y opcionalmente en paréntesis el número de juegos del perdedor en el tie break

```
select ronda, desenlace,
        (ganador).nombre || ' ' || (ganador).apellido as ganador,
```

```
        (perdedor).nombre || ' ' || (perdedor).apellido as perdedor,
        (select string_agg(iset.juegos_ganador || '-' || iset.juegos_perdedor ||
        case
                when iset.puntos_tiebreak_perdedor is not null then
                        '(' || iset.puntos_tiebreak_perdedor || ')'
                else ''
        end, ', ' order by iset.num_set)
        from unnest(info_sets) as iset) as resultado
from partidos
where ronda in ('SF', 'F')
        and (torneo).torneo.nombre = 'Roland Garros'
        and extract(year from fecha) = '2018'
```

A.4. Muestra la lista de jugadores españoles (ES) que ganaron algún torneo de nivel Gran Slam (G). Para cada jugador muestra los siguientes datos resumen de todos sus partidos: número de partidos jugados, porcentaje de victorias, porcentaje de aces, porcentaje de dobles faltas, porcentaje de servicios ganados, porcentaje de restos ganados, porcentaje de break points salvados (de los sufridos en contra), porcentaje de break points ganados (de los provocados a favor)

```
with jugadores_espanoles_ganadores as (
    select distinct (ganador).id as id_jugador, (ganador).nombre || ' ' || (ganador).apellido AS jugador
    from partidos
    where (ganador).pais.codigo_iso2 = 'ES'
        and ronda = 'F'
        and (torneo).nivel = 'G'
)

select
    jeg.jugador,
    count(p.num_partido) as partidos,
    round(100.0 * sum(case when jeg.id_jugador = (p.ganador).id then 1 else 0 end) / count(p.num_partido), 1
    round(100.0 * sum(case when jeg.id_jugador = (p.ganador).id then (p.ganador_stats).num_aces else (p.perd
        nullif(sum(case when jeg.id_jugador = (p.ganador).id then (p.ganador_stats).num_ptos_servidos else (
    round(100.0 * sum(case when jeg.id_jugador = (p.ganador).id then (p.ganador_stats).num_dob_faltas else (
        nullif(sum(case when jeg.id_jugador = (p.ganador).id then (p.ganador_stats).num_ptos_servidos else (
    round(100.0 * sum(case when jeg.id_jugador = (p.ganador).id then (p.ganador_stats).num_primeros_servicio
            else (p.perdedor_stats).num_primeros_servicios_ganados + (p.perdedor_stats).num_segundos_servici
        nullif(sum(case when jeg.id_jugador = (p.ganador).id then (p.ganador_stats).num_ptos_servidos else (
    round(100.0 * sum(case when jeg.id_jugador = (p.ganador).id
            then (p.perdedor_stats).num_ptos_servidos - (p.perdedor_stats).num_primeros_servicios_ganados -
            else (p.ganador_stats).num_ptos_servidos - (p.ganador_stats).num_primeros_servicios_ganados - (p
        nullif(sum(case when jeg.id_jugador = (p.ganador).id then (p.perdedor_stats).num_ptos_servidos else
    round(100.0 * sum(case when jeg.id_jugador = (p.ganador).id then (p.ganador_stats).num_break_salvados el
        nullif(sum(case when jeg.id_jugador = (p.ganador).id then (p.ganador_stats).num_break_afrontados els
    round(100.0 * sum(case when jeg.id_jugador = (p.ganador).id then (p.perdedor_stats).num_break_afrontados
            else (p.ganador_stats).num_break_afrontados - (p.ganador_stats).num_break_salvados end) /
        nullif(sum(case when jeg.id_jugador = (p.ganador).id then (p.perdedor_stats).num_break_afrontados el
from jugadores_espanoles_ganadores jeg, partidos p
where jeg.id_jugador = (p.ganador).id
    or jeg.id_jugador = (p.perdedor).id
group by jeg.jugador
```

A.5. Lista los jugadores que fueron derrotados (en algún partido del 2018) por el rival de Rafael Nadal de la primera ronda (R128) de Roland Garros de 2018

```
with rival_nadal as (
        select case when (ganador).nombre = 'Rafael' then (perdedor).nombre || ' ' || (perdedor).apellido
```

```
                                                else (ganador).nombre || ' ' || (ganador).apellido end as j
        from partidos
        where (((ganador).nombre = 'Rafael' and (ganador).apellido = 'Nadal') or
                  ((perdedor).nombre = 'Rafael' and(perdedor).apellido = 'Nadal'))
                and ronda = 'R128'
                and (torneo).torneo.nombre = 'Roland Garros'
                and extract(year from fecha) = '2018'
)

select (p.perdedor).nombre || ' ' || (p.perdedor).apellido as jugador, (p.perdedor).pais.codigo_iso2 as pais
from partidos p, rival_nadal rn
where (p.ganador).nombre || ' ' || (p.ganador).apellido = rn.jugador
        and extract(year from p.fecha) = '2018'
order by jugador
```

B. JSON

```
create table tenisjson as
select
        -- columna 'jugador' como objeto json
        jsonb_build_object(
                'id', j.id,
                'nombre', j.nombre,
                'apellido', j.apellido,
                'diestro', j.diestro,
                'fecha_nacimiento', j.fecha_nacimiento,
                'altura', j.altura
        ) as jugador,
  -- columna 'pais' como objeto json
        jsonb_build_object(
                'codigo_iso2', p.codigo_iso2,
                'codigo_iso3', p.codigo_iso3,
                'codigo_ioc', p.codigo_ioc,
                'nombre', p.nombre
        ) as pais,
  -- columna 'partidos_ganados' como agregado json
        (select jsonb_agg(jsonb_build_object(
                'torneo', jsonb_build_object(
                        'nombre', t.nombre,
                        'pais', jsonb_build_object(
                                'codigo_iso2', pa.codigo_iso2,
                                'codigo_iso3', pa.codigo_iso3,
                                'codigo_ioc', pa.codigo_ioc,
                                'nombre', pa.nombre),
                        'fecha', et.fecha,
                        'superficie', et.superficie,
                        'tamano', et.tamano,
                        'nivel', et.nivel),
                'fecha', pg.fecha,
                'ronda', pg.ronda,
                'desenlace', pg.desenlace,
                'num_partido', pg.num_partido,
                'rival', pg.perdedor,
                'sets', (select jsonb_agg(
                        jsonb_build_object(
                                'num_set', sp.num_set,
                                'juegos_ganador', sp.juegos_ganador,
```

```
                        'juegos_perdedor', sp.juegos_perdedor,
                        'puntos_tiebreak_perdedor', sp.puntos_tiebreak_perdedor
                    )
                )
                from sets_partido sp
                where sp.torneo = pg.torneo
                    and sp.fecha = pg.fecha
                    and sp.num_partido = pg.num_partido),
                'stats', jsonb_build_object(
                    'num_aces', pg.num_aces_ganador,
                    'num_dob_faltas', pg.num_dob_faltas_ganador,
                    'num_puntos_servidos', pg.num_ptos_servidos_ganador,
                    'num_primeros_servicios', pg.num_primeros_servicios_ganador,
                    'num_primeros_servicios_ganados', pg.num_primeros_servicios_ganados_ganador,
                    'num_segundos_servicios_ganados', pg.num_segundos_servicios_ganados_ganador,
                    'num_juegos_servidos', pg.num_juegos_servidos_ganador,
                    'num_break_salvados', pg.num_break_salvados_ganador,
                    'num_break_afrontados', pg.num_break_afrontados_ganador
                ),
                'stats_rival', jsonb_build_object(
                    'num_aces', pg.num_aces_perdedor,
                    'num_dob_faltas', pg.num_dob_faltas_perdedor,
                    'num_puntos_servidos', pg.num_ptos_servidos_perdedor,
                    'num_primeros_servicios', pg.num_primeros_servicios_perdedor,
                    'num_primeros_servicios_ganados', pg.num_primeros_servicios_ganados_perdedor,
                    'num_segundos_servicios_ganados', pg.num_segundos_servicios_ganados_perdedor,
                    'num_juegos_servidos', pg.num_juegos_servidos_perdedor,
                    'num_break_salvados', pg.num_break_salvados_perdedor,
                    'num_break_afrontados', pg.num_break_afrontados_perdedor
                )
            ))
            from public.partido pg
                left join public.edicion_torneo et on pg.torneo = et.torneo and pg.fecha = et.fecha
                left join public.torneo t on et.torneo = t.id
                left join public.pais pa on t.pais = pa.codigo_iso2
            where pg.ganador = j.id) as partidos_ganados,
        -- columna 'partidos_perdidos' como json
            (select jsonb_agg(jsonb_build_object(
                'torneo', jsonb_build_object(
                    'nombre', t.nombre,
                    'pais', jsonb_build_object(
                        'codigo_iso2', pa.codigo_iso2,
                        'codigo_iso3', pa.codigo_iso3,
                        'codigo_ioc', pa.codigo_ioc,
                        'nombre', pa.nombre
                    ),
                    'fecha', et.fecha,
                    'superficie', et.superficie,
                    'tamano', et.tamano,
                    'nivel', et.nivel
                ),
                'fecha', pp.fecha,
                'ronda', pp.ronda,
                'desenlace', pp.desenlace,
                'num_partido', pp.num_partido,
                'rival', pp.ganador,
                'sets', (select jsonb_agg(
```

```
                        jsonb_build_object(
                                'num_set', sp.num_set,
                                'juegos_ganador', sp.juegos_ganador,
                                'juegos_perdedor', sp.juegos_perdedor,
                                'puntos_tiebreak_perdedor', sp.puntos_tiebreak_perdedor
                        )
                )
                from sets_partido sp
                where sp.torneo = pp.torneo
                        and sp.fecha = pp.fecha
                        and sp.num_partido = pp.num_partido),
                'stats', jsonb_build_object(
                        'num_aces', pp.num_aces_perdedor,
                        'num_dob_faltas', pp.num_dob_faltas_perdedor,
                        'num_puntos_servidos', pp.num_ptos_servidos_perdedor,
                        'num_primeros_servicios', pp.num_primeros_servicios_perdedor,
                        'num_primeros_servicios_ganados', pp.num_primeros_servicios_ganados_perdedor,
                        'num_segundos_servicios_ganados', pp.num_segundos_servicios_ganados_perdedor,
                        'num_juegos_servidos', pp.num_juegos_servidos_perdedor,
                        'num_break_salvados', pp.num_break_salvados_perdedor,
                        'num_break_afrontados', pp.num_break_afrontados_perdedor
                ),
                'stats_rival', jsonb_build_object(
                        'num_aces', pp.num_aces_ganador,
                        'num_dob_faltas', pp.num_dob_faltas_ganador,
                        'num_puntos_servidos', pp.num_ptos_servidos_ganador,
                        'num_primeros_servicios', pp.num_primeros_servicios_ganador,
                        'num_primeros_servicios_ganados', pp.num_primeros_servicios_ganados_ganador,
                        'num_segundos_servicios_ganados', pp.num_segundos_servicios_ganados_ganador,
                        'num_juegos_servidos', pp.num_juegos_servidos_ganador,
                        'num_break_salvados', pp.num_break_salvados_ganador,
                        'num_break_afrontados', pp.num_break_afrontados_ganador
                )
        ))
        from public.partido pp
                left join public.edicion_torneo et on pp.torneo = et.torneo and pp.fecha = et.fecha
                left join public.torneo t on et.torneo = t.id
                left join public.pais pa on t.pais = pa.codigo_iso2
        where pp.perdedor = j.id) as partidos_perdidos
from public.jugador j
        left join public.pais p on j.pais = p.codigo_iso2
where j.id in (select perdedor from public.partido)
        or j.id in (select ganador from public.partido)
```

B.1. Muestra todos los ganadores del torneo "Wimbledon" (Nombre apellidos y año). Ordena el resultado por año.

```
select tj.jugador ->> 'nombre' as nombre,
    tj.jugador ->> 'apellido' as apellido,
    extract(year from (pg ->> 'fecha')::date) as ano
from tenisjson tj, jsonb_array_elements(partidos_ganados) as partidos(pg)
where pg ->> 'ronda' = 'F'
    and pg -> 'torneo' ->> 'nombre' = 'Wimbledon'
order by ano
```

B.2. Muestra los años en los que Roger Federer ganó algún torneo de nivel Gran Slam (G) o Master 1000 (M). Para cada año, muestra el número de torneos y lista sus nombres (ordenados por la fecha de celebración). Ordena el resultado por el año

```
select extract(year from (pg -> 'torneo' ->> 'fecha')::date) as ano,
    count(distinct pg -> 'torneo'->>'nombre') as numero_torneos,
    string_agg(pg -> 'torneo'->>'nombre', ', ' order by pg -> 'torneo' ->> 'fecha') as torneos
from tenisjson tj, jsonb_array_elements(partidos_ganados) as partidos(pg)
where tj.jugador ->> 'nombre' = 'Roger'
    and tj.jugador ->> 'apellido' = 'Federer'
    and pg ->> 'ronda' = 'F'
    and pg -> 'torneo'->>'nivel' in ('G', 'M')
group by ano
order by ano
```

B.3. Muestra los partidos de semifinales (ronda='SF') y final (ronda = 'F') del torneo de "Roland Garros" del 2018. Para cada partido muestra la ronda, el tipo de desenlace, el nombre y apellidos del ganador y el nombre y apellidos del perdedor y el resultado con el número de juegos del ganador y del perdedor en cada set, y opcionalmente en paréntesis el número de juegos del perdedor en el tie break

```
select pg ->> 'ronda' as ronda, pg ->> 'desenlace' as desenlace,
        (tj.jugador ->> 'nombre')::text || ' ' || (tj.jugador ->> 'apellido')::text as ganador,
        (tjr.jugador ->> 'nombre')::text || ' ' || (tjr.jugador ->> 'apellido')::text as perdedor,
        string_agg((s ->> 'juegos_ganador')::text || '-' || (s ->> 'juegos_perdedor')::text ||
                case when s ->> 'puntos_tiebreak_perdedor' is not null
                        then '(' || (s ->> 'puntos_tiebreak_perdedor')::text || ')'
                        else '' end, ', ' order by s ->> 'num_set') as resultado
from tenisjson tj, jsonb_array_elements(tj.partidos_ganados) as partidos(pg),
        jsonb_array_elements(pg -> 'sets') as setss(s), tenisjson tjr
where pg ->'torneo' ->> 'nombre' = 'Roland Garros'
    and extract(year from (pg ->> 'fecha')::date) = 2018
    and pg ->> 'ronda' in ('SF', 'F')
    and tjr.jugador->>'id' = pg ->> 'rival'
group by pg ->> 'ronda', pg ->> 'desenlace',
        tj.jugador ->> 'nombre', tj.jugador ->> 'apellido',
        tjr.jugador ->> 'nombre', tjr.jugador ->> 'apellido', pg ->> 'fecha'
```

B.4. Muestra la lista de jugadores españoles (ES) que ganaron algún torneo de nivel Gran Slam (G). Para cada jugador muestra los siguientes datos resumen de todos sus partidos: número de partidos jugados, porcentaje de victorias, porcentaje de aces, porcentaje de dobles faltas, porcentaje de servicios ganados, porcentaje de restos ganados, porcentaje de break points salvados (de los sufridos en contra), porcentaje de break points ganados (de los provocados a favor)

```
with jugadores_espanoles_ganadores as (
    select distinct (tj.jugador ->> 'id')::integer as id_jugador,
            (tj.jugador ->> 'nombre')::text || ' ' || (tj.jugador ->> 'apellido')::text as jugador
    from tenisjson tj, jsonb_array_elements(partidos_ganados) as partidos(pg)
    where tj.pais ->> 'codigo_iso2' = 'ES'
        and pg ->> 'ronda' = 'F'
        and pg -> 'torneo' ->> 'nivel' = 'G'
)

select jeg.jugador,
    count(*) as partidos,
    round(100.0 * count(case when pg ->> 'rival' = jeg.id_jugador::text then null else 1 end)::numeric / cou
    round(100.0 * sum(case when pg ->> 'rival' = jeg.id_jugador::text
            then (pg -> 'stats_rival' ->> 'num_aces')::numeric
            else (pg -> 'stats' ->> 'num_aces')::numeric end) /
```

```
            nullif(sum(case when pg ->> 'rival' = jeg.id_jugador::text
                    then (pg -> 'stats_rival' ->> 'num_puntos_servidos')::numeric
                    else (pg->'stats'->>'num_puntos_servidos')::numeric end), 0), 1) as pcje_aces,
    round(100.0 * sum(case when pg ->> 'rival' = jeg.id_jugador::text
            then (pg -> 'stats_rival' ->> 'num_dob_faltas')::numeric
            else (pg -> 'stats' ->> 'num_dob_faltas')::numeric end) /
            nullif(sum(case when pg ->> 'rival' = jeg.id_jugador::text
                    then (pg -> 'stats_rival' ->> 'num_puntos_servidos')::numeric
                    else (pg -> 'stats' ->> 'num_puntos_servidos')::numeric end), 0), 1) as pcje_dobles_falt
    round(100.0 * sum(case when pg ->> 'rival' = jeg.id_jugador::text
            then (pg -> 'stats_rival' ->> 'num_primeros_servicios_ganados')::numeric + (pg -> 'stats_rival'
            else (pg -> 'stats' ->> 'num_primeros_servicios_ganados')::numeric + (pg -> 'stats' ->> 'num_seg
            nullif(sum(case when pg ->> 'rival' = jeg.id_jugador::text
                    then (pg -> 'stats_rival' ->> 'num_puntos_servidos')::numeric
                    else (pg -> 'stats' ->> 'num_puntos_servidos')::numeric end), 0), 1) as pcje_servicios_g
    round(100.0 * sum(case when pg ->> 'rival' = jeg.id_jugador::text
            then (pg -> 'stats' ->> 'num_puntos_servidos')::numeric - (pg -> 'stats' ->> 'num_primeros_servi
                 (pg -> 'stats' ->> 'num_segundos_servicios_ganados')::numeric
            else (pg -> 'stats_rival' ->> 'num_puntos_servidos')::numeric - (pg -> 'stats_rival' ->> 'num_pi
                 (pg -> 'stats_rival' ->> 'num_segundos_servicios_ganados')::numeric end) /
            nullif(sum(case when pg ->> 'rival' = jeg.id_jugador::text
                    then (pg -> 'stats' ->> 'num_puntos_servidos')::numeric
                    else (pg -> 'stats_rival' ->> 'num_puntos_servidos')::numeric end), 0), 1) as pcje_resto
    round(100.0 * sum(case when pg ->> 'rival' = jeg.id_jugador::text
            then (pg->'stats_rival'->>'num_break_salvados')::numeric
            else (pg -> 'stats' ->> 'num_break_salvados')::numeric end) /
            nullif(sum(case when pg ->> 'rival' = jeg.id_jugador::text
                    then (pg -> 'stats_rival' ->> 'num_break_afrontados')::numeric
                    else (pg -> 'stats' ->> 'num_break_afrontados')::numeric end), 0), 1) as pcje_breaks_sal
    round(100.0 * sum(case when pg ->> 'rival' = jeg.id_jugador::text
            then (pg -> 'stats' ->> 'num_break_afrontados')::numeric - (pg -> 'stats' ->> 'num_break_salvado
            else (pg -> 'stats_rival' ->> 'num_break_afrontados')::numeric - (pg -> 'stats_rival' ->> 'num_b
            nullif(sum(case when pg ->> 'rival' = jeg.id_jugador::text
                    then (pg -> 'stats' ->> 'num_break_afrontados')::numeric
                    else (pg -> 'stats_rival' ->> 'num_break_afrontados')::numeric end), 0), 1) as pcje_brea
from jugadores_espanoles_ganadores jeg, tenisjson tj,
        jsonb_array_elements(tj.partidos_ganados) as partidos(pg)
where (tj.jugador ->> 'id')::integer = jeg.id_jugador
        or pg ->> 'rival' = jeg.id_jugador::text
group by jeg.jugador
```

B.5. Lista los jugadores que fueron derrotados (en algún partido del 2018) por el rival de Rafael Nadal de la primera ronda (R128) de Roland Garros de 2018

```
with rival_nadal as (
        select case when tj.jugador ->> 'nombre' = 'Rafael'
                        then (pg ->> 'rival')::integer
                        else (tj.jugador ->> 'id')::integer end as id_jugador,
                case when tj.jugador ->> 'nombre' = 'Rafael'
                        then (tjr.jugador ->> 'nombre')::text || ' ' || (tjr.jugador ->> 'apellido')::text
                        else (tj.jugador ->> 'nombre')::text || ' ' || (tj.jugador ->> 'apellido')::text end
        from tenisjson tj, jsonb_array_elements(tj.partidos_ganados) as partidos(pg), tenisjson tjr
    where pg ->> 'torneo' ->> 'nombre' = 'Roland Garros'
            and pg ->> 'ronda' = 'R128'
                and extract(year from (pg ->> 'fecha')::date) = 2018
        and tjr.jugador->>'id' = pg ->> 'rival'
        and ((tj.jugador ->> 'nombre' = 'Rafael' and tj.jugador ->> 'apellido' = 'Nadal')
```

```
                or (tjr.jugador ->> 'nombre' = 'Rafael' and tjr.jugador ->> 'apellido' = 'Nadal'))
)

select tj.jugador->>'nombre' || ' ' || (tj.jugador->>'apellido')::text as jugador,
        tj.pais->>'codigo_iso2' as pais
from rival_nadal rn, tenisjson tj, jsonb_array_elements(tj.partidos_perdidos) as partidos(pg)
where rn.id_jugador = (pg->>'rival')::integer
        and extract(year from (pg->>'fecha')::date) = 2018
```

3. BASES DE DATOS DISTRIBUIDAS CON SQL (CITUS DATA)

.1. Muestra todos los ganadores del torneo "Wimbledon" (Nombre apellidos y año). Ordena el resultado por año.

```
-- Q1
```

.2. Muestra los años en los que Roger Federer ganó algún torneo de nivel Gran Slam (G) o Master 1000 (M). Para cada año, muestra el número de torneos y lista sus nombres (ordenados por la fecha de celebración). Ordena el resultado por el año

```
-- Q2
```

.3. Muestra los partidos de semifinales (ronda='SF') y final (ronda = 'F') del torneo de "Roland Garros" del 2018. Para cada partido muestra la ronda, el tipo de desenlace, el nombre y apellidos del ganador y el nombre y apellidos del perdedor y el resultado con el número de juegos del ganador y del perdedor en cada set, y opcionalmente en paréntesis el número de juegos del perdedor en el tie break

```
-- Q3
```

.4. Muestra la lista de jugadores españoles (ES) que ganaron algún torneo de nivel Gran Slam (G). Para cada jugador muestra los siguientes datos resumen de todos sus partidos: número de partidos jugados, porcentaje de victorias, porcentaje de aces, porcentaje de dobles faltas, porcentaje de servicios ganados, porcentaje de restos ganados, porcentaje de break points salvados (de los sufridos en contra), porcentaje de break points ganados (de los provocados a favor)

```
-- Q4
```

.5. Lista los jugadores que fueron derrotados (en algún partido del 2018) por el rival de Rafael Nadal de la primera ronda (R128) de Roland Garros de 2018

```
-- Q5
```

4. BASES DE DATOS NOSQL: DOCUMENTALES (MONGODB)

.1. Muestra todos los ganadores del torneo "Wimbledon" (Nombre apellidos y año). Ordena el resultado por año.

```
-- Q1
```

.2. Muestra los años en los que Roger Federer ganó algún torneo de nivel Gran Slam (G) o Master 1000 (M). Para cada año, muestra el número de torneos y lista sus nombres (ordenados por la fecha de celebración). Ordena el resultado por el año

```
-- Q2
```

.3. Muestra los partidos de semifinales (ronda='SF') y final (ronda = 'F') del torneo de "Roland Garros" del 2018. Para cada partido muestra la ronda, el tipo de desenlace, el nombre y apellidos del ganador y el nombre y apellidos del perdedor y el resultado con el número de juegos del ganador y del perdedor en cada set, y opcionalmente en paréntesis el número de juegos del perdedor en el tie break

```
-- Q3
```

.4. Muestra la lista de jugadores españoles (ES) que ganaron algún torneo de nivel Gran Slam (G). Para cada jugador muestra los siguientes datos resumen de todos sus partidos: número de partidos jugados, porcentaje de victorias, porcentaje de aces, porcentaje de dobles faltas, porcentaje de servicios ganados, porcentaje de restos ganados, porcentaje de break points salvados (de los sufridos en contra), porcentaje de break points ganados (de los provocados a favor)

```
-- Q4
```

.5. Lista los jugadores que fueron derrotados (en algún partido del 2018) por el rival de Rafael Nadal de la primera ronda (R128) de Roland Garros de 2018

```
-- Q5
```

5. BASES DE DATOS NOSQL: GRAFOS (NEO4J)

A. Creación de índices

```
CREATE INDEX IF NOT EXISTS FOR (j:Jugador) ON (j.id);
CREATE INDEX IF NOT EXISTS FOR (et:EdicionTorneo) ON (et.torneo, et.fecha);
CREATE INDEX IF NOT EXISTS FOR (p:Partido) ON (p.num_partido, p.fecha);
CREATE INDEX IF NOT EXISTS FOR (p:Pais) ON (p.codigo_iso2);
CREATE INDEX IF NOT EXISTS FOR (t:Torneo) ON (t.id);
CREATE INDEX partido_fecha_num IF NOT EXISTS FOR (p:Partido) ON (p.num_partido, p.fecha);
```

B. Carga de datos con JBDC

```
// cargamos los paises
WITH "jdbc:postgresql://localhost:5432/tenis?user=alumnogreibd&password=greibd2021" as url
CALL apoc.load.jdbc(url, "SELECT codigo_iso2, codigo_iso3, codigo_ioc, nombre FROM pais") YIELD
    row
CREATE (p:Pais {
codigo_iso2: row.codigo_iso2,
codigo_iso3: row.codigo_iso3,
codigo_ioc: row.codigo_ioc,
nombre: row.nombre
});

// cargamos los jugadores
WITH "jdbc:postgresql://localhost:5432/tenis?user=alumnogreibd&password=greibd2021" AS url
CALL apoc.load.jdbc(url,
"SELECT id, nombre, apellido, diestro, fecha_nacimiento, pais, altura FROM jugador") YIELD row
MATCH (pa:Pais {codigo_iso2: row.pais})
CREATE (j:Jugador {
    id: row.id,
    nombre: row.nombre,
    apellido: row.apellido,
    diestro: row.diestro,
    fecha_nacimiento: row.fecha_nacimiento,
    altura: row.altura
```

```
})
CREATE (j)-[:REPRESENTA_A]->(pa);

// cargamos los torneos
WITH "jdbc:postgresql://localhost:5432/tenis?user=alumnogreibd&password=greibd2021" AS url
CALL apoc.load.jdbc(url,
"SELECT id, nombre, pais FROM torneo") YIELD row
CREATE (t:Torneo {
    id: row.id,
    nombre: row.nombre
})
WITH t, row
OPTIONAL MATCH (pa:Pais {codigo_iso2: row.pais})
WITH t, pa
WHERE pa IS NOT NULL
CREATE (t)-[:SE_CELEBRA_EN]->(pa);

// cargamos las ediciones de los torneos
WITH "jdbc:postgresql://localhost:5432/tenis?user=alumnogreibd&password=greibd2021" AS url
CALL apoc.load.jdbc(url,
"SELECT torneo, fecha, superficie, tamano, nivel FROM edicion_torneo") YIELD row
MATCH (t:Torneo {id: row.torneo})
CREATE (et:EdicionTorneo {
    fecha: row.fecha,
    superficie: row.superficie,
    tamano: row.tamano,
    nivel: row.nivel,
    torneo: row.torneo
})
CREATE (et)-[:EDICION_DE]->(t);

// cargamos los partidos
WITH "jdbc:postgresql://localhost:5432/tenis?user=alumnogreibd&password=greibd2021" AS url
CALL apoc.load.jdbc(url,
"SELECT p.torneo, p.fecha, p.num_partido, p.num_sets, p.ronda, p.desenlace,
p.ganador, p.perdedor,
p.num_aces_ganador, p.num_dob_faltas_ganador, p.num_ptos_servidos_ganador,
p.num_primeros_servicios_ganador, p.num_primeros_servicios_ganados_ganador,
p.num_segundos_servicios_ganados_ganador, p.num_juegos_servidos_ganador,
p.num_break_salvados_ganador, p.num_break_afrontados_ganador,
p.num_aces_perdedor, p.num_dob_faltas_perdedor, p.num_ptos_servidos_perdedor,
p.num_primeros_servicios_perdedor, p.num_primeros_servicios_ganados_perdedor,
p.num_segundos_servicios_ganados_perdedor, p.num_juegos_servidos_perdedor,
p.num_break_salvados_perdedor, p.num_break_afrontados_perdedor
FROM partido p
WHERE p.fecha IS NOT NULL AND p.torneo IS NOT NULL
ORDER BY p.fecha, p.torneo, p.num_partido
LIMIT 10000 OFFSET 0") YIELD row

// creamos el partido independientemente de las referencias
CREATE (p:Partido {
num_partido: row.num_partido,
fecha: row.fecha,
num_sets: row.num_sets,
ronda: row.ronda,
desenlace: row.desenlace,
torneo_id: row.torneo // Guardamos la referencia al torneo aunque no exista el nodo
```

```
})

// vinculamos con el torneo si existe
WITH p, row
OPTIONAL MATCH (t:Torneo {id: row.torneo})
WITH p, row, t
WHERE t IS NOT NULL
CREATE (p)-[:SE_JUEGA_EN]->(t)

// vinculamos con el ganador si existe
WITH p, row
OPTIONAL MATCH (ganador:Jugador {id: toInteger(row.ganador)})
WITH p, row, ganador
WHERE ganador IS NOT NULL
CREATE (p)-[:GANADO_POR {
num_aces: row.num_aces_ganador,
num_dob_faltas: row.num_dob_faltas_ganador,
num_ptos_servidos: row.num_ptos_servidos_ganador,
num_primeros_servicios: row.num_primeros_servicios_ganador,
num_primeros_servicios_ganados: row.num_primeros_servicios_ganados_ganador,
num_segundos_servicios_ganados: row.num_segundos_servicios_ganados_ganador,
num_juegos_servidos: row.num_juegos_servidos_ganador,
num_break_salvados: row.num_break_salvados_ganador,
num_break_afrontados: row.num_break_afrontados_ganador
}]->(ganador)

// vinculamos con el perdedor si existe
WITH p, row
OPTIONAL MATCH (perdedor:Jugador {id: toInteger(row.perdedor)})
WITH p, row, perdedor
WHERE perdedor IS NOT NULL
CREATE (p)-[:PERDIDO_POR {
num_aces: row.num_aces_perdedor,
num_dob_faltas: row.num_dob_faltas_perdedor,
num_ptos_servidos: row.num_ptos_servidos_perdedor,
num_primeros_servicios: row.num_primeros_servicios_perdedor,
num_primeros_servicios_ganados: row.num_primeros_servicios_ganados_perdedor,
num_segundos_servicios_ganados: row.num_segundos_servicios_ganados_perdedor,
num_juegos_servidos: row.num_juegos_servidos_perdedor,
num_break_salvados: row.num_break_salvados_perdedor,
num_break_afrontados: row.num_break_afrontados_perdedor
}]->(perdedor);

// cargamos los sets de los partidos
WITH "jdbc:postgresql://localhost:5432/tenis?user=alumnogreibd&password=greibd2021" AS url
CALL apoc.load.jdbc(url,
"SELECT * FROM sets_partido LIMIT 10000 OFFSET 0") YIELD row
OPTIONAL MATCH (p:Partido {num_partido: row.num_partido, fecha: row.fecha})
WITH row, p
WHERE p IS NOT NULL
CREATE (sp:SetPartido {
num_set: row.num_set,
juegos_ganador: row.juegos_ganador,
juegos_perdedor: row.juegos_perdedor,
puntos_tiebreak_perdedor: row.puntos_tiebreak_perdedor
})
CREATE (sp)-[:PERTENECE_A]->(p);
```

B.1. Muestra todos los ganadores del torneo "Wimbledon" (Nombre apellidos y año). Ordena el resultado por año.

```
MATCH (j:Jugador)-[gp:GANADO_POR]-(p:Partido)-[:SE_JUEGA_EN]->(t:Torneo)
WHERE t.nombre = 'Wimbledon'
AND p.ronda = 'F'
RETURN j.nombre AS nombre,
j.apellido AS apellido,
substring(toString(p.fecha), 0, 4) AS ano
ORDER BY ano;
```

B.2. Muestra los años en los que Roger Federer ganó algún torneo de nivel Gran Slam (G) o Master 1000 (M). Para cada año, muestra el número de torneos y lista sus nombres (ordenados por la fecha de celebración). Ordena el resultado por el año

```
MATCH (j:Jugador)-[:GANADO_POR]-(p:Partido)-[:SE_JUEGA_EN]->(t:Torneo)
MATCH (et:EdicionTorneo)-[:EDICION_DE]->(t)
WHERE j.nombre = 'Roger'
AND j.apellido = 'Federer'
AND p.ronda = 'F'
AND et.nivel IN ['G', 'M']
AND et.fecha = p.fecha
AND et.torneo = t.id
WITH substring(toString(p.fecha), 0, 4) AS ano,
collect(DISTINCT t.nombre) AS torneos_nombres,
count(DISTINCT t) AS num_torneos
RETURN ano,
num_torneos,
reduce(s = head(torneos_nombres), x IN tail(torneos_nombres) | s + ', '+ x) AS torneos
ORDER BY ano;
```

B.3. Muestra los partidos de semifinales (ronda='SF') y final (ronda = 'F') del torneo de "Roland Garros" del 2018. Para cada partido muestra la ronda, el tipo de desenlace, el nombre y apellidos del ganador y el nombre y apellidos del perdedor y el resultado con el número de juegos del ganador y del perdedor en cada set, y opcionalmente en paréntesis el número de juegos del perdedor en el tie break

```
MATCH (jg:Jugador)-[:GANADO_POR]-(p:Partido)-[:PERDIDO_POR]-(jp:Jugador),
(p)-[:SE_JUEGA_EN]->(t:Torneo),
(sp:SetPartido)-[:PERTENECE_A]->(p)
WHERE t.nombre = 'Roland Garros'
AND p.ronda IN ['SF', 'F']
AND substring(toString(p.fecha), 0, 4) = '2018'
WITH p, jg, jp, sp
ORDER BY sp.num_set
WITH p, jg, jp,
collect(sp.juegos_ganador + '-' + sp.juegos_perdedor +
CASE sp.puntos_tiebreak_perdedor
WHEN null THEN ''
ELSE '(' + toString(sp.puntos_tiebreak_perdedor) + ')'
END) as sets
RETURN p.ronda as ronda,
p.desenlace as desenlace,
jg.nombre + ''+ jg.apellido as ganador,
jp.nombre + ''+ jp.apellido as perdedor,
reduce(s = head(sets), x IN tail(sets) | s + ', '+ x) as resultado
ORDER BY p.fecha;
```

B.4. Muestra la lista de jugadores españoles (ES) que ganaron algún torneo de nivel Gran Slam (G). Para cada jugador muestra los siguientes datos resumen de todos sus partidos: número de partidos jugados, porcentaje de victorias, porcentaje de aces, porcentaje de dobles faltas, porcentaje de servicios ganados, porcentaje de restos ganados, porcentaje de break points salvados (de los sufridos en contra), porcentaje de break points ganados (de los provocados a favor)

```
// Primero identificamos los jugadores espanoles que han ganado finales de Grand Slam
MATCH (j:Jugador)-[:REPRESENTA_A]->(p:Pais {codigo_iso2: 'ES'})
MATCH (j)-[:GANADO_POR]-(partido:Partido)-[:SE_JUEGA_EN]->(t:Torneo)
MATCH (et:EdicionTorneo)-[:EDICION_DE]->(t)
WHERE partido.ronda = 'F'
AND et.nivel = 'G'
AND et.fecha = partido.fecha
AND et.torneo = t.id
WITH DISTINCT j.id as id_jugador, j.nombre + ''+ j.apellido as jugador


// Ahora calculamos todas las estadisticas para estos jugadores
MATCH (j:Jugador)
WHERE j.id = id_jugador
MATCH (p:Partido)
MATCH (j)-[r:GANADO_POR|PERDIDO_POR]-(p)
WITH jugador, p, j, r, type(r) as tipo,
CASE type(r) WHEN 'GANADO_POR' THEN 1 ELSE 0 END as es_ganador,
CASE type(r)
WHEN 'GANADO_POR' THEN r.num_aces
ELSE r.num_aces END as aces,
CASE type(r)
WHEN 'GANADO_POR' THEN r.num_ptos_servidos
ELSE r.num_ptos_servidos END as ptos_servidos,
CASE type(r)
WHEN 'GANADO_POR' THEN r.num_dob_faltas
ELSE r.num_dob_faltas END as dobles_faltas,
CASE type(r)
WHEN 'GANADO_POR' THEN r.num_primeros_servicios_ganados + r.num_segundos_servicios_ganados
ELSE r.num_primeros_servicios_ganados + r.num_segundos_servicios_ganados END as
    servicios_ganados,
// Para restos ganados necesitamos el rival
CASE type(r)
WHEN 'GANADO_POR' THEN [(p)-[rp:PERDIDO_POR]-() | rp.num_ptos_servidos -
    rp.num_primeros_servicios_ganados - rp.num_segundos_servicios_ganados][0]
ELSE [(p)-[rg:GANADO_POR]-() | rg.num_ptos_servidos - rg.num_primeros_servicios_ganados -
    rg.num_segundos_servicios_ganados][0] END as restos_ganados,
CASE type(r)
WHEN 'GANADO_POR' THEN [(p)-[rp:PERDIDO_POR]-() | rp.num_ptos_servidos][0]
ELSE [(p)-[rg:GANADO_POR]-() | rg.num_ptos_servidos][0] END as ptos_servidos_rival,
CASE type(r)
WHEN 'GANADO_POR' THEN r.num_break_salvados
ELSE r.num_break_salvados END as breaks_salvados,
CASE type(r)
WHEN 'GANADO_POR' THEN r.num_break_afrontados
ELSE r.num_break_afrontados END as breaks_afrontados,
CASE type(r)
WHEN 'GANADO_POR' THEN [(p)-[rp:PERDIDO_POR]-() | rp.num_break_afrontados -
    rp.num_break_salvados][0]
ELSE [(p)-[rg:GANADO_POR]-() | rg.num_break_afrontados - rg.num_break_salvados][0] END as
    breaks_ganados,
```

```
CASE type(r)
WHEN 'GANADO_POR' THEN [(p)-[rp:PERDIDO_POR]-() | rp.num_break_afrontados][0]
ELSE [(p)-[rg:GANADO_POR]-() | rg.num_break_afrontados][0] END as breaks_rival
WITH jugador,
count(p) as partidos,
100.0 * sum(es_ganador) / count(p) as pcje_victorias,
CASE WHEN sum(ptos_servidos) = 0 THEN 0
ELSE 100.0 * sum(aces) / sum(ptos_servidos) END as pcje_aces,
CASE WHEN sum(ptos_servidos) = 0 THEN 0
ELSE 100.0 * sum(dobles_faltas) / sum(ptos_servidos) END as pcje_dobles_faltas,
CASE WHEN sum(ptos_servidos) = 0 THEN 0
ELSE 100.0 * sum(servicios_ganados) / sum(ptos_servidos) END as pcje_servicios_ganados,
CASE WHEN sum(ptos_servidos_rival) = 0 THEN 0
ELSE 100.0 * sum(restos_ganados) / sum(ptos_servidos_rival) END as pcje_restos_ganados,
CASE WHEN sum(breaks_afrontados) = 0 THEN 0
ELSE 100.0 * sum(breaks_salvados) / sum(breaks_afrontados) END as pcje_breaks_salvados,
CASE WHEN sum(breaks_rival) = 0 THEN 0
ELSE 100.0 * sum(breaks_ganados) / sum(breaks_rival) END as pcje_breaks_ganados
RETURN
jugador,
partidos,
round(pcje_victorias, 1) as pcje_victorias,
round(pcje_aces, 1) as pcje_aces,
round(pcje_dobles_faltas, 1) as pcje_dobles_faltas,
round(pcje_servicios_ganados, 1) as pcje_servicios_ganados,
round(pcje_restos_ganados, 1) as pcje_restos_ganados,
round(pcje_breaks_salvados, 1) as pcje_breaks_salvados,
round(pcje_breaks_ganados, 1) as pcje_breaks_ganados
ORDER BY jugador;
```

B.5. Lista los jugadores que fueron derrotados (en algún partido del 2018) por el rival de Rafael Nadal de la primera ronda (R128) de Roland Garros de 2018

```
// Primero encontramos al rival de Nadal en Roland Garros 2018 R128
MATCH (nadal:Jugador {nombre: 'Rafael', apellido: 'Nadal'})
MATCH (rival:Jugador)
MATCH (p:Partido)-[:SE_JUEGA_EN]->(t:Torneo {nombre: 'Roland Garros'})
WHERE p.ronda = 'R128'
AND substring(toString(p.fecha), 0, 4) = '2018'
AND ((nadal)-[:GANADO_POR]-(p)-[:PERDIDO_POR]-(rival) OR
(nadal)-[:PERDIDO_POR]-(p)-[:GANADO_POR]-(rival))


// Ahora buscamos los partidos donde este rival perdio en 2018
WITH rival
MATCH (rival)-[:GANADO_POR]-(derrotas:Partido)-[:PERDIDO_POR]-(perdedor:Jugador)
WHERE substring(toString(derrotas.fecha), 0, 4) = '2018'
MATCH (perdedor)-[:REPRESENTA_A]->(pais:Pais)
RETURN DISTINCT
perdedor.nombre + ''+ perdedor.apellido as jugador,
pais.codigo_iso2 as pais
ORDER BY jugador;
```