-Bases de Datos- 2022/2023

Memoria de la Practica 1

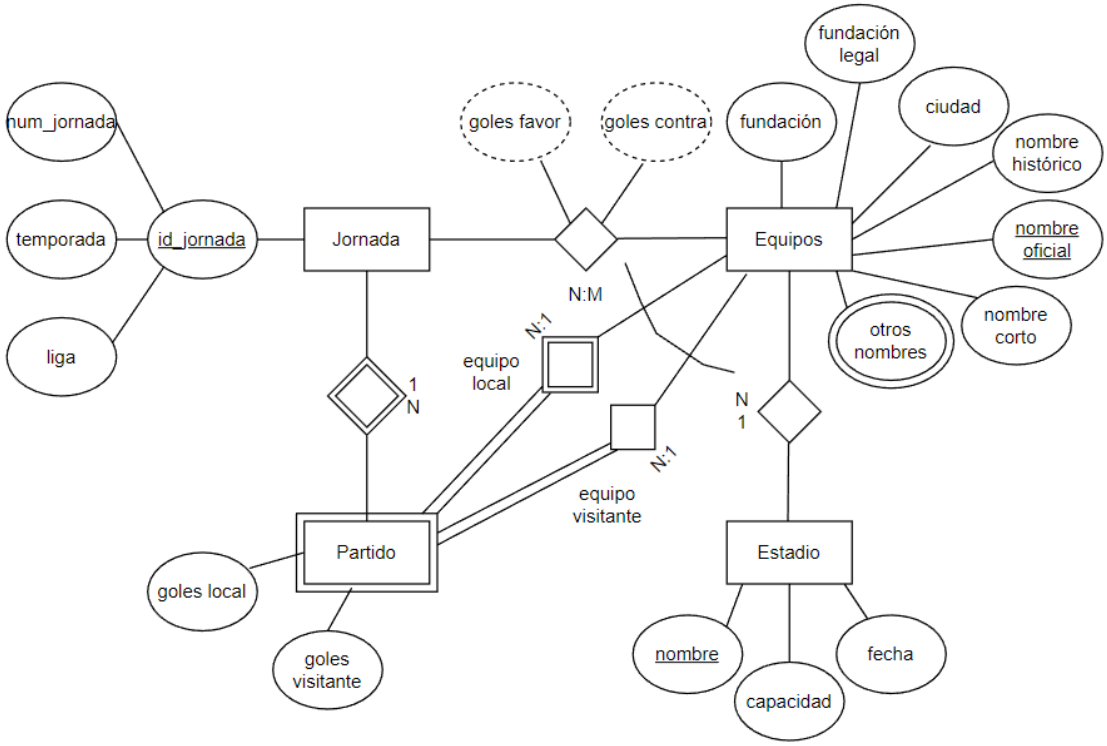
Realizado el 13/03/2023

Autor: Jorge Leris Lacort e-mail: 845647@unizar.es
Autor: Ángel Villanueva Agudo e-mail: 844759@unizar.es
Autor: Inés Román Gracia e-mail: 820731@unizar.es

ÍNDICE

PARTE 1: CREACIÓN DE LA BASE DE DATOS

1.1 Esquema E/R global



Restricciones:

- Cada equipo solo puede jugar un partido en cada jornada.
- Un partido tiene que ser jugado por dos equipos diferentes.
- Un equipo no puede jugar en más de una liga a la vez en la misma temporada.
- Cada equipo en cada temporada solo puede jugar un partido como visitante y otro como local contra un mismo equipo.

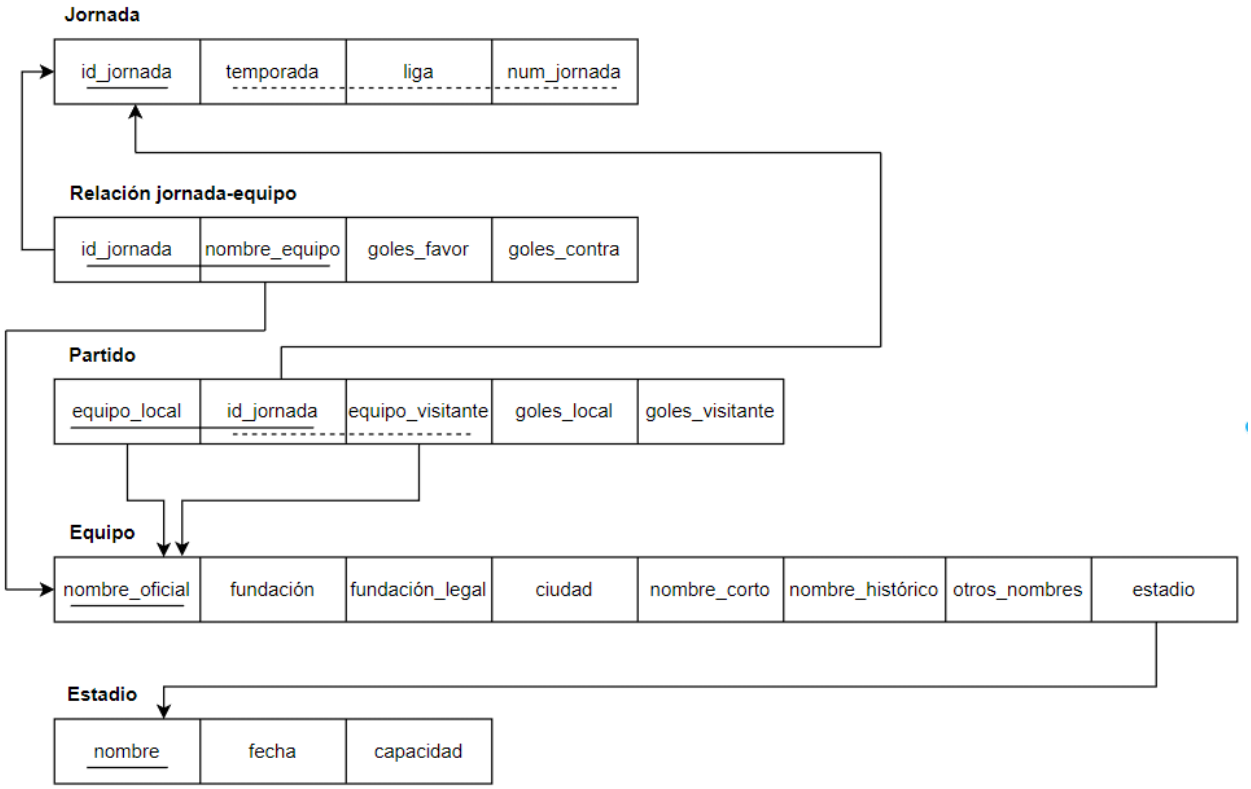
Soluciones alternativas

- Una solución alternativa a las relaciones equipo visitante y equipo local entre equipos y partidos sería crear una relación N:M entre ambas entidades, con la restricción de que en un partido solo pueden participar dos equipos diferentes; es decir, que sea 2:N.

- Otra solución alternativa a la tabla de relación entre equipo y jornada donde se guardan los puntos de cada equipo en una jornada, sería dividir la entidad jornada creando otra entidad temporada que se relacione con la entidad equipo, de forma que en la tabla de relación N:M entre temporada y equipo se guarden todos los goles a favor y en contra y los puntos de un equipo en una jornada determinada.

1.2 Esquema relacional

Inicial:



Normalización:

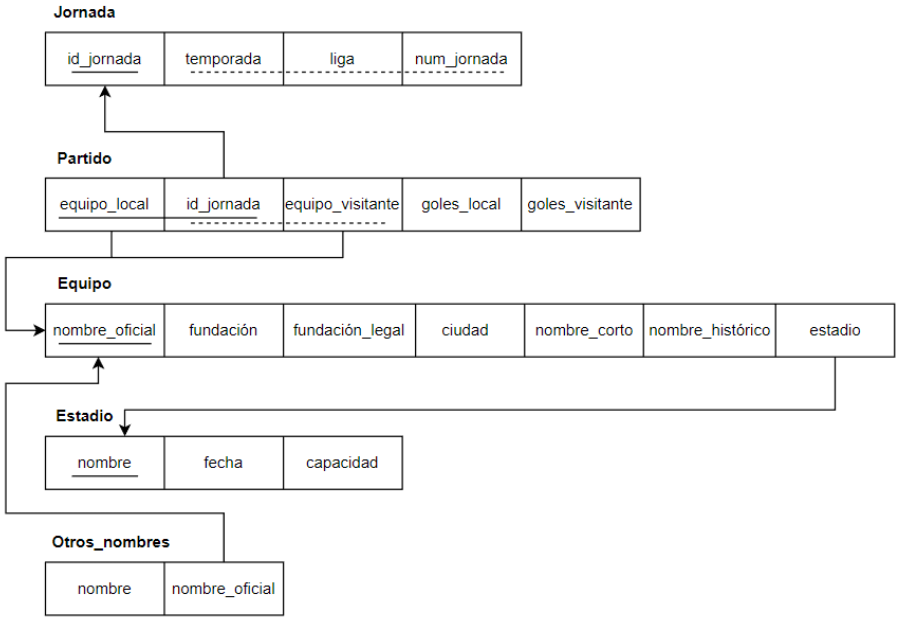
Para normalizar nuestro esquema relacional hemos analizado las siguientes formas normales:

- 1FN: Para que se cumpla tenemos que eliminar los valores multivaluados de nuestro diseño que en nuestro caso es el atributo “otros_nombres”, para ello crearemos una nueva tabla para almacenar todos estos nombres, los cuales se identificarán con la clave del nombre oficial.

- 2FN: Nuestro diseño ya está en la segunda forma normal, ya que no hay ninguna dependencia funcional con parte de la clave.

- 3FN: Para lograr esta forma debemos eliminar las dependencias funcionales transitivas, en nuestro caso esto sucede con los atributos de “goles_favor” y “goles_contra” de la tabla relación jornada-equipo y “goles_local” y “goles_visitante” de la tabla partido, de manera que dependen del valor que tienen entre ellos. Para solucionar esto guardaremos los goles solo en la tabla partido, por lo que la tabla relación desaparecerá.

Modelo normalizado:



Tras la normalización, queda un modelo simplificado sin dependencias o datos repetidos, donde el principal cambio es que se ha eliminado la tabla que relaciona equipos con jornadas. Sin embargo, en el modelo final que emplearemos en SQL volveremos a utilizar esta tabla. Además, borraremos de la tabla de partidos los resultados para no guardar la información triplicada.

El motivo de esta decisión es por la eficiencia de las consultas, al tener solamente almacenados los goles de cada partido, las consultas de puntos y goles almacenados, debían hacerse utilizando código condicional como “CASE WHEN …”. A parte, esto no lo hemos trabajado en clase, resulta que las consultas que utilizan estos controles de flujo resultan altamente ineficientes. Por esto cada equipo se relaciona con una tabla en la que se podrá consultar los puntos y los goles en una jornada determinada.

Otra solución es la división de la entidad jornada creando otra entidad temporada que podamos relacionar con equipos, donde se almacenarían los goles a favor y en contra acumulados de cada equipo para una temporada determinada. Esta solución resulta más eficiente en espacio, ya que no habría tantas filas con los resultados. La desventaja de esta solución frente a la nuestra es que cada vez que se añade un partido de una temporada habría que modificar esta tabla.

Finalmente se ha optado por la primera solución porque en un primer momento poblamos nuestra base de datos de acuerdo al modelo normalizado hasta que descubrimos los costes de tiempo que tenían nuestras consultas, la primera solución resulta más fácil de cambiar desde nuestra forma normalizada por eso optamos por esta opción. Este diseño desnormalizado en este caso no tiene un coste extra al añadir nuevos partidos, ya que solamente se crearía también otra fila en la tabla de relación equipo-jornada. Lo mismo pasaría con las actualizaciones de datos de resultados de partidos. Aunque en principio no deberían ser muy frecuentes en esta base de datos.

1.3 Sentencias SQL de creación de tablas

Para almacenar la información de nuestra base de datos hemos usado las siguientes secuencias SQL para crear las tablas necesarias:

Tabla "jornada":

Para la creación de esta tabla hemos generado una clave "id_jornada" la cual se basará en los atributos: "num_jornada", "temporada" y "liga", de manera que se creará un nuevo id_jornada en cada num_jornada con diferente liga y diferente temporada.

```
CREATE TABLE jornada
(
    id_jornada      NUMBER PRIMARY KEY,
    num_jornada     NUMBER,
    temporada       NUMBER,
    liga            VARCHAR(10),
    UNIQUE(num_jornada, temporada, liga)
);
```

Tabla "estadio":

Esta tabla contiene como clave el nombre del estadio y contendrá los atributos de "fecha" y "capacidad" para almacenar la respectiva información de los estadios.

```
CREATE TABLE estadio
(
    nombre          VARCHAR(60) PRIMARY KEY,
    fecha           NUMBER(5),
    capacidad       NUMBER(5)
);
```

Tabla "equipo"

La clave de esta tabla es el nombre oficial del equipo y contiene los atributos: "nombre_corto", "nombre_hist", "fundacion", "fundacion_legal", "ciudad" y "estadio" para contener toda la información necesaria de los equipos, donde el nombre corto tiene que ser una clave única de cada equipo y estadio es una clave foránea que hace referencia al nombre de la tabla estadio.

```
CREATE TABLE equipo
(
    nombre          VARCHAR(60) PRIMARY KEY,
    nombre_corto    VARCHAR(60),
    nombre_hist     VARCHAR(60),
    fundacion       NUMBER,
    fundacion_legal NUMBER,
    ciudad          VARCHAR(60),
    estadio         VARCHAR(60),
    UNIQUE(nombre_corto),
    FOREIGN KEY(estadio) REFERENCES estadio(nombre) ON DELETE SET NULL
);
```

Tabla "otros_nombres"

Esta tabla se identifica con el nombre oficial de un equipo y contiene todos los demás nombres de este que no estén especificados en la tabla de equipo, donde "nombre_oficial" es una clave foránea que hace referencia a la tabla que contiene los equipos

```
CREATE TABLE otros_nombres
(
    nombre          VARCHAR(60) PRIMARY KEY,
    nombre_oficial  VARCHAR(60),
    FOREIGN KEY(nombre_oficial) REFERENCES equipo ON DELETE CASCADE
);
```

Tabla “partido”

Los partidos los identificaremos solo con la jornada y el equipo local, ya que no son necesarios los dos equipos del partido para identificarlo, aunque sí que almacenaremos los dos como atributos y la jornada y el equipo visitante formarán una clave única de partido. Además, jornada será una clave foránea con referencia a “id_jornada” de la tabla jornada y el equipo local y visitante formarán otra clave foránea con referencia a la tabla de equipo.

Respecto al modelo relacional normalizado hemos quitado los goles de aquí para introducirlos en una nueva tabla como hemos comentado anteriormente para conseguir un acceso eficiente a los datos.

```
CREATE TABLE partido
(
    jornada         NUMBER,
    equipo_local    VARCHAR(60) NOT NULL,
    equipo_visit    VARCHAR(60) NOT NULL,
    PRIMARY KEY(jornada, equipo_local),
    UNIQUE(jornada, equipo_visit),
    FOREIGN KEY(jornada) REFERENCES jornada(id_jornada) ON DELETE CASCADE,
    FOREIGN KEY(equipo_local) REFERENCES equipo ON DELETE CASCADE,
    FOREIGN KEY(equipo_visit) REFERENCES equipo ON DELETE CASCADE
);
```

Tabla “puntos”

Se identificará con la jornada y el equipo y contendrá información de los puntos y goles a favor y en contra de cada equipo en los partidos. Además “jornada” será una clave foránea que hace referencia a “id_jornada” en tabla jornada y “equipo” será otra clave foránea con referencia a la tabla de equipos.

Esta tabla la hemos añadido para obtener un acceso eficiente a los datos.

```
CREATE TABLE puntos(
    jornada         NUMBER,
    equipo          VARCHAR(60),
    puntos          NUMBER NOT NULL,
    goles_favor     NUMBER,
    goles_contra    NUMBER,
    PRIMARY KEY(jornada, equipo),
    FOREIGN KEY(jornada) REFERENCES jornada(id_jornada) ON DELETE CASCADE,
    FOREIGN KEY(equipo) REFERENCES equipo ON DELETE CASCADE
);
```

Secuencias de borrado de tablas

```
DROP TABLE puntos;        DROP TABLE estadio;
DROP TABLE partido;       DROP TABLE jornada;
DROP TABLE otros_nombres;
DROP TABLE equipo;
```

PARTE 2: INTRODUCCIÓN DE DATOS Y EJECUCIÓN DE CONSULTAS

2.1 Pasos seguidos para la población de la BD:

Una vez ya creada la base de datos descrita con anterioridad nos encontramos ante diversos problemas para rellenar los datos en las diferentes tablas creadas para cada una de ellas se expondrá la dificultad más relevante para su población y otros problemas.

En la tabla de jornada, la dificultad radica en que diferentes ligas repiten la misma temporada y jornada, por ello es necesario crear una secuencia (id_jornada) para poder distinguir cada una de ellas. Para la tabla de estadio y su población simplemente se ha tenido que sacar los datos relativos al nombre, fecha de inauguración y aforo de cada uno de ellos de la tabla datosdb.ligahost.

En la tabla de equipo para poblar hay algunos equipos los cuales no tienen un nombre de club, por ello se tiene que recoger todos los equipos los cuales tienen un nombre de club dentro de la tabla de datos que no sea nulo, para luego crear una tabla auxiliar la cual será eliminada después de insertar los datos en la tabla equipos, con aquellos equipos que no disponen de nombre de club sustituyéndolo en su lugar por el nombre del equipo local que disputó el partido, ejemplo de esto es la línea 9 de datosdb.ligahost en la cual se aprecia que el campo de nombre del club está vacío, por ello se utiliza el único nombre disponible del equipo local (Málaga (C. D.)) en su lugar.

En la tabla otros_nombres a la hora de rellenar con los datos relativos al nombre de un club oficial y otros nombres, si el equipo tiene ya un nombre de club valido este puede tener o no un nombre adicional, el cual para poblar esta tabla ha de ser no nulo puesto que es la clave primaria.

En la tabla de partido se ha encontrado que en la tabla original, en las últimas filas de esta se encuentran algunos partidos que nunca fueron disputados en la liga, para poblar esta tabla se ha asociado el id_jornada el cual ha sido mencionado con anterioridad, a cada partido disputado con su equipo visitante y equipo local, relacionando con las tablas de jornada y equipo además de la base de datos original.

En la tabla de puntos se ha calculado la clasificación resultante de un equipo tras participar en un partido, para ello se tiene en cuenta cuando un equipo es visitante o local, si los goles marcados por uno u otro en el partido son goles a favor del equipo o por el contrario son goles del oponente y por ello son goles en contra, además se tiene que tener en cuenta la jornada en la que fue disputado el partido, la temporada y la liga a la que pertenece el partido para disponer de los datos correctos en la tabla.

2.2 Consultas SQL realizadas:

Código de la primera consulta en SQL:

En primer lugar, se crea una vista para realizar la primera consulta con los ganadores de cada temporada de liga de primera división:

```
CREATE VIEW ganadores(equipo, temporada, puntos) AS
    SELECT p1.equipo, j1.temporada, SUM(puntos)
    FROM puntos p1
    JOIN jornada j1 ON j1.id_jornada = p1.jornada
    WHERE j1.liga='1'
    GROUP BY p1.equipo, j1.temporada
    HAVING SUM(p1.puntos) = (
                             SELECT MAX(SUM(p2.puntos))
                             FROM puntos p2
                            JOIN jornada j2 ON j2.id_jornada = p2.jornada
                            WHERE j2.liga='1' AND j2.temporada=j1.temporada
                            GROUP BY p2.equipo, j2.temporada)
    ORDER BY j1.temporada, p1.equipo;
```

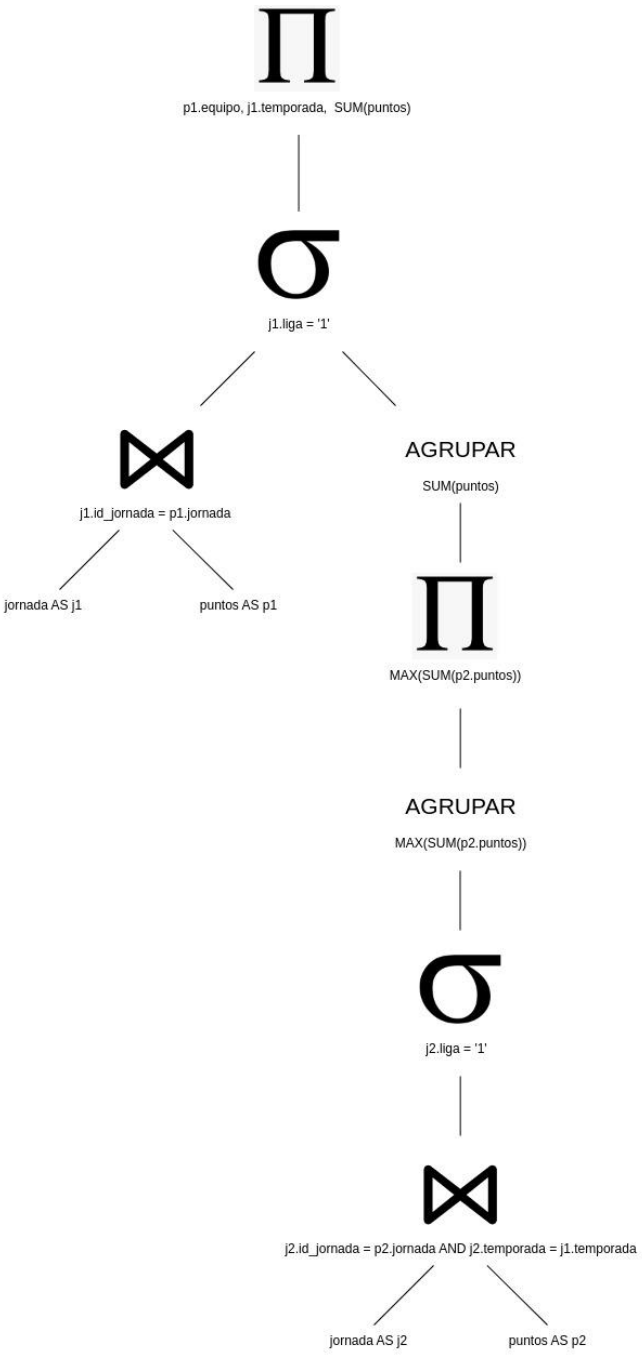
Una vez realizada la vista la utilizamos para tomar de ella el equipo el cual tiene el máximo de ligas ganadas en primera división:

```
SELECT equipo, COUNT(*) AS NUM_GANADAS
   FROM ganadores
   HAVING COUNT(*) = (
                  SELECT MAX(COUNT(*))
                  FROM ganadores
                 GROUP BY equipo)
   GROUP BY equipo;


   DROP VIEW ganadores;
```

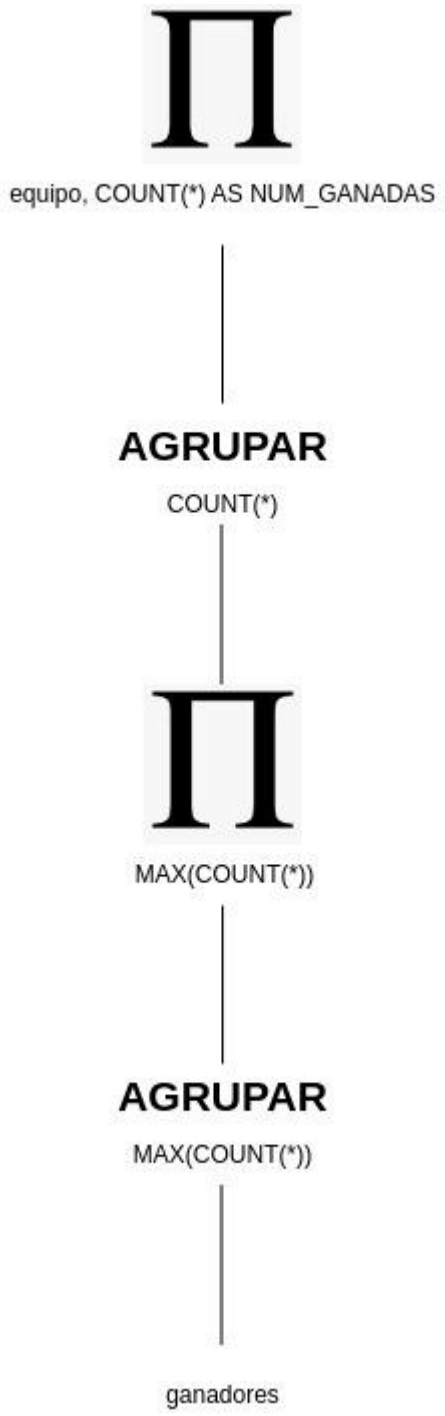
Respuestas obtenidas de la BD:

```
Vista creada.


EQUIPO                                                    NUM_GANADAS
--------------------------------------------------------- -----------
Real Madrid Club de Futbol                                         19


Vista borrada.
```

Árbol sintáctico de la vista “ganadores”:



Árbol sintáctico de la consulta:



Código de la segunda consulta en SQL:

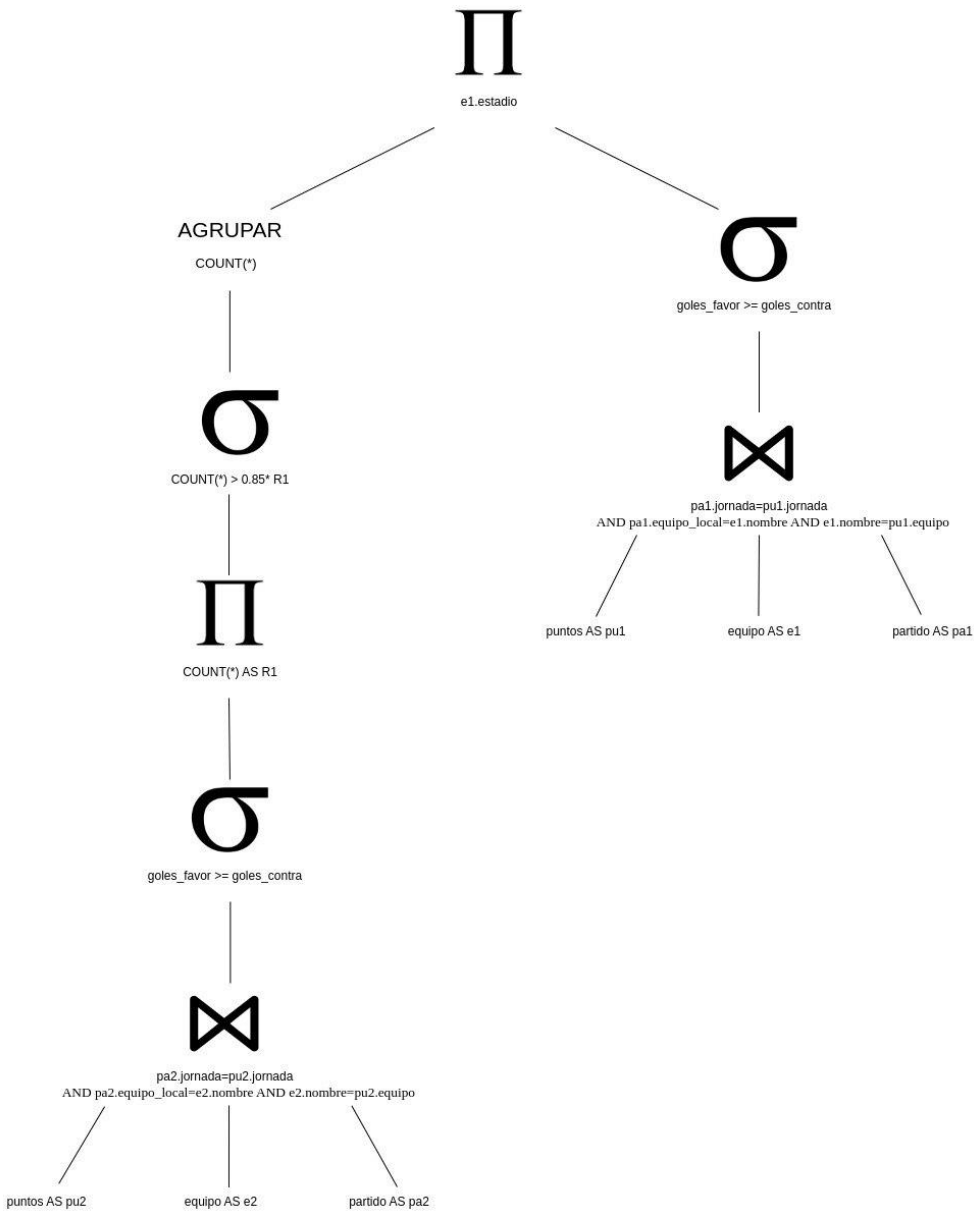
Para realizar la consulta, cogemos información de las tablas partido, equipo y puntos, donde el equipo local que juegue en dicho estadio debe de haber ganado más de un 85% de las veces en él, para ello tendremos que tener en cuenta todos los partidos que se han jugado en dicho estadio, para saber si el equipo ha ganado más de un 85% de las veces.

```
SELECT DISTINCT e1.estadio
FROM partido pa1, equipo e1, puntos pu1
WHERE goles_favor >= goles_contra AND pa1.jornada=pu1.jornada
        AND pa1.equipo_local=e1.nombre AND e1.nombre=pu1.equipo
HAVING COUNT(*) > 0.85* (
        SELECT COUNT(*)
        FROM partido pa2 , equipo e2, puntos pu2
        WHERE e2.estadio = e1.estadio AND pa2.jornada=pu2.jornada
                AND pa2.equipo_local=e2.nombre AND e2.nombre=pu2.equipo)
GROUP BY e1.estadio
ORDER BY e1.estadio;
```

Respuestas obtenidas de la BD:

ESTADIO
Camp Nou
Narcis Sala
Nuevo Lasesarre
Santiago Bernabeu

Árbol sintáctico:



Código de la tercera consulta en SQL:

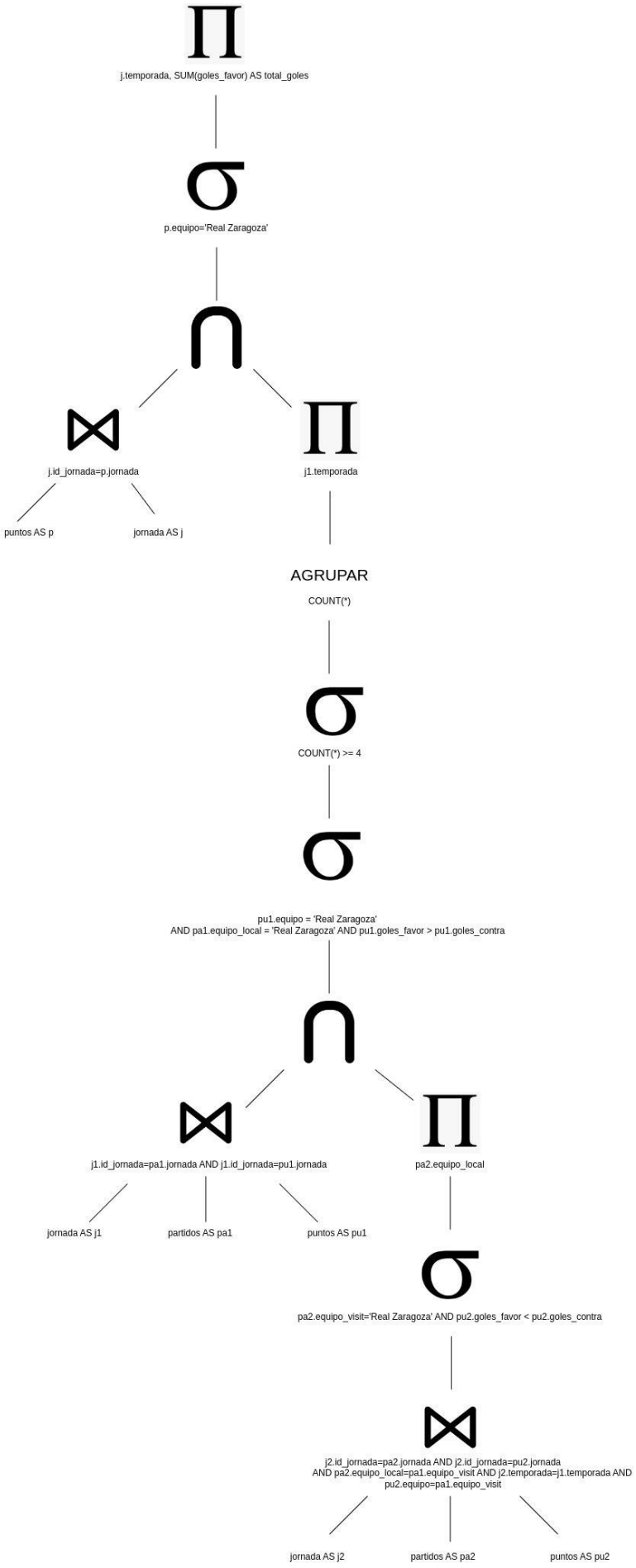
Para realizar la consulta se tiene que tener en cuenta la tabla de puntos y de jornada, por ello el equipo tiene que ser el Real Zaragoza y se ha de considerar cuando el Zaragoza es equipo local y visitante para concluir si el equipo ha ganado al oponente en ida y vuelta, y si en esa temporada ha ganado a por lo menos 4 equipos en ida y vuelta.

```
SELECT j.temporada, SUM(goles_favor) AS total_goles
FROM puntos p, jornada j
WHERE j.id_jornada=p.jornada AND p.equipo='Real Zaragoza'
        AND j.temporada IN(
        SELECT j1.temporada
        FROM puntos pu1, partido pa1, jornada j1
        WHERE j1.id_jornada=pa1.jornada AND j1.id_jornada=pu1.jornada AND
pu1.equipo = 'Real Zaragoza'
                AND pa1.equipo_local = 'Real Zaragoza' AND pu1.goles_favor >
pu1.goles_contra
                AND EXISTS (
                SELECT pa2.equipo_local
                FROM puntos pu2, partido pa2, jornada j2
                WHERE j2.id_jornada=pa2.jornada AND j2.id_jornada=pu2.jornada
                        AND pa2.equipo_local=pa1.equipo_visit AND
j2.temporada=j1.temporada
                        AND pa2.equipo_visit='Real Zaragoza' AND
pu2.equipo=pa1.equipo_visit
                        AND pu2.goles_favor < pu2.goles_contra)
        HAVING COUNT(*) >= 4
        GROUP BY j1.temporada)
GROUP BY j.temporada
ORDER BY j.temporada;
```

Respuestas obtenidas de la BD:

TEMPORADA	TOTAL_GOLES
1982	59
1985	51
1998	57
2002	54
2008	79

Árbol sintáctico:



PARTE 3: DISEÑO FÍSICO:

3.1 Primera Consulta:

Uso en bytes = 46.050.830 bytes
Uso en CPU = 2.080

En este caso, se podría considerar la materialización de la vista de ganadores, ya que es una tabla con solo una fila por temporada (tiene muy poco coste en espacio), a la que solo se le añadirá una nueva fila cada año al obtener el nuevo ganador de la temporada (tiene un coste en tiempo asumible solo una vez al año) y mejora considerablemente el resultado de la consulta:

Uso en bytes = 5.824 bytes
Uso en CPU = 22

3.2 Segunda Consulta:

Uso en bytes = 22.641.871 bytes
Uso en CPU = 1.172

Las cláusulas “WHERE” solo condicionan los valores de las claves de las tablas que se consultan. Así que no sería necesaria la creación de ningún índice adicional para mejorar el rendimiento de las consultas, ya que los índices de las claves primarias de las tablas ya son creados automáticamente por el sistema gestor de bases de datos.

3.3 Tercera Consulta:

Uso en bytes = 2.484.790 bytes
Uso en CPU = 3.956

En este caso, en la segunda cláusula “WHERE” se condiciona la búsqueda de partido a los valores de su clave primaria (jornada y equipo local) y del equipo visitante. El sistema gestor de base de datos ya crea un índice por defecto de la clave primaria. Así que podríamos añadir un nuevo índice que también incluya el valor de equipo visitante en la búsqueda de partidos:

```
CREATE INDEX jornada_local_visitante ON partido (jornada, equipo_local, equipo_visit)
```

Uso de bytes con índice = 2.484.790 bytes
Uso en CPU = 3.514

Sería necesario hacer un estudio de si realmente la gestión y el coste físico de este índice sale rentable para esta pequeña mejora en los resultados de la consulta.

ANEXO

Horas dedicadas a cada parte:

	Inés Román Gracia	Ángel Villanueva Agudo	Jorge Leris Lacort	Total
Parte 1	18h	9h	17h	44h
Parte 2	24h	27h	17h	68h
Parte 3	2h	2h	1.5 h	5.5h
Total	44h	38h	35.5h	117.5h

División del trabajo:

- Parte 1:
  - Esquema E/R global y restricciones: Inés
  - Normalización: Inés
  - Sentencias SQL de creación de tablas: Jorge, Inés
- Parte 2:
  - Pasos Seguidos para la población: Ángel, Jorge, Inés
  - Realizar las Consultas SQL: Ángel, Jorge, Inés
- Parte 3:
  - Problemas de Rendimiento en consultas: Ángel, Jorge, Inés

Problemas encontrados:

- Muchas horas dedicadas al estudio de la asignatura en la primera práctica.
- Problemas de coordinación y reparto de tareas a mejorar en futuras sesiones para liberar carga de trabajo (ya se ha hablado para solucionarlos).
- No todos los miembros han llevado la práctica al día.

Sesiones de reunión:

Todos hemos asistido a todas las reuniones.

16/02/2023 - Sesión 1 de prácticas (2 horas)
23/02/2023 - Sesión 2 de prácticas (2 horas)
28/02/2023 - Sesión de reunión (1 hora)
02/03/2023 - Sesión 3 de prácticas (2 horas)
05/03/2023 - Sesión de reunión (2 horas)
06/03/2023 - Sesión de reunión (2 horas)
09/03/2023 - Sesión de reunión (2 horas)
10/03/2023 - Sesión de reunión (2 horas)
11/03/2023 - Sesión de reunión (1 hora)
12/03/2023 - Sesión de reunión (2 horas)
13/03/2023 - Sesión de reunión (2 horas)