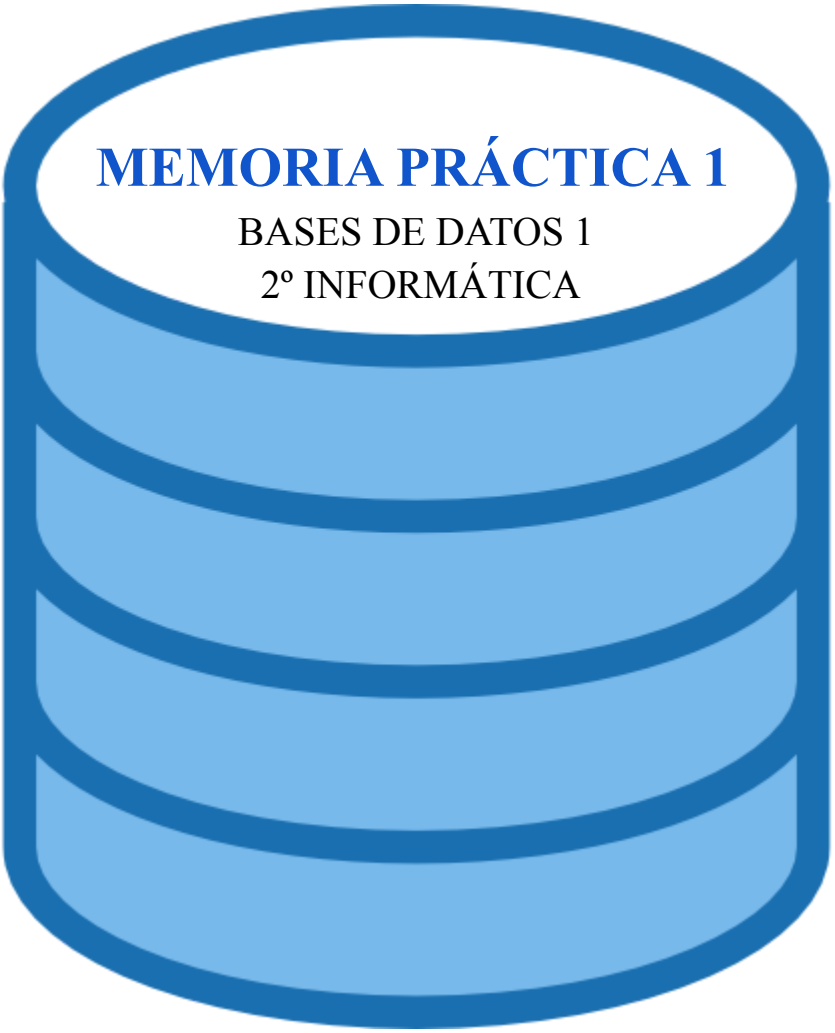
MEMORIA PRÁCTICA 1

BASES DE DATOS 1

2º INFORMÁTICA



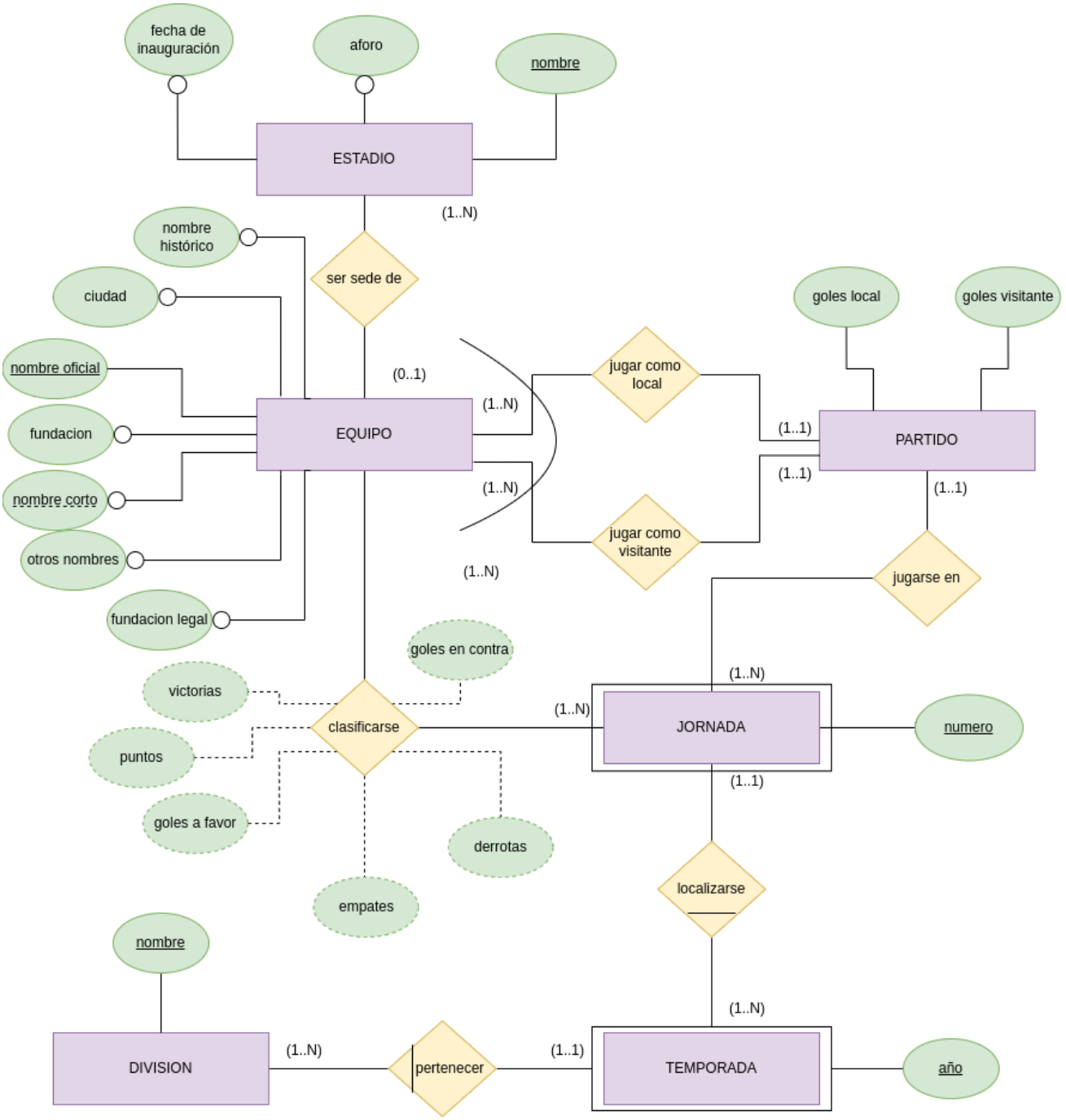
Ignacio Millán Aznar (841537@unizar.es)
Diego Roldán Urueña (841723@unizar.es)
Abel Romeo Lancina (846088@unizar.es)

13/03/2023

ÍNDICE

PARTE 1: CREACIÓN DE UNA BASE DE DATOS

Esquema E/R global



Nuestro esquema entidad-relación consta de seis entidades, como se puede ver: División, Temporada, Estadio, Equipo, Partido y Jornada.
Sobre cada uno existen varios atributos, así como se pedía en el enunciado, los cuales guardan información sobre cada entidad.

En cuanto a los atributos vemos distintos tipos, la mayoría son regulares, pero vemos ciertas excepciones como los atributos de la relación “clasificarse” que tienen el contorno discontinuo ya que se calculan a partir de otros. “Goles a favor” no puede ser cualquier valor, sino una suma de todos los goles que ha marcado cierto equipo hasta cierta jornada en una temporada concreta.

Las claves primarias de cada entidad se denotan subrayando el nombre del atributo con línea continua. Si el atributo es único, la línea del subrayado es discontinua.

La relación entre Equipo y Partido es una interrelación exclusiva, ya que en un mismo partido el mismo equipo no puede ser local y visitante a la vez.

En la relación entre Temporada, División y Jornada hay una dependencia en existencia e identificación, pues Temporada es débil con respecto a División porque para identificar la temporada, hace falta el atributo “nombre”, propio de División. A su vez Jornada es débil frente a Temporada, ya que para identificar la jornada hace falta la clave primaria de la temporada.

Además, se ha supuesto que desde el inicio las victorias sean de tres puntos.

Listado de restricciones:

Entidad “Equipo”:

- Todos los atributos son opcionales menos la clave primaria “nombre oficial”
- “nombre corto” es único
- Un equipo solo puede jugar un partido en una jornada y no puede participar como local y visitante a la vez
- Un equipo puede tener uno o ningún estadio

Entidad “Estadio”:

- El nombre del estadio es único
- Un estadio puede ser sede de varios equipos
- Un estadio tendrá que ser sede de algún equipo

Entidad “Partido”:

- Un partido solo podrá ser disputado en una jornada
- Un partido será disputado por dos equipos

Entidad “Jornada”:

- No puede haber dos jornadas con el mismo id_jornada.

Entidad “Temporada”:

- No puede haber dos temporadas con el mismo año y nombre de división.

Entidad “División”:

- No puede haber dos divisiones con el mismo nombre.

Relación “ser sede de”:

- Un estadio puede pertenecer a dos equipos simultáneamente

Relación “jugar como local” y “jugar como visitante”:

- Cada equipo podrá jugar como local o como visitante N veces, pero en cada partido solo podrá jugar como local o como visitante

Relación “clasificarse”:

- Esta relación representa información de cierto equipo en una jornada de una temporada de una división concreta. Para cada jornada se actualizarán los valores de goles, victorias, derrotas….

Relación “localizarse”:

- Una temporada tiene varias jornadas, pero una jornada solo pertenece a una temporada

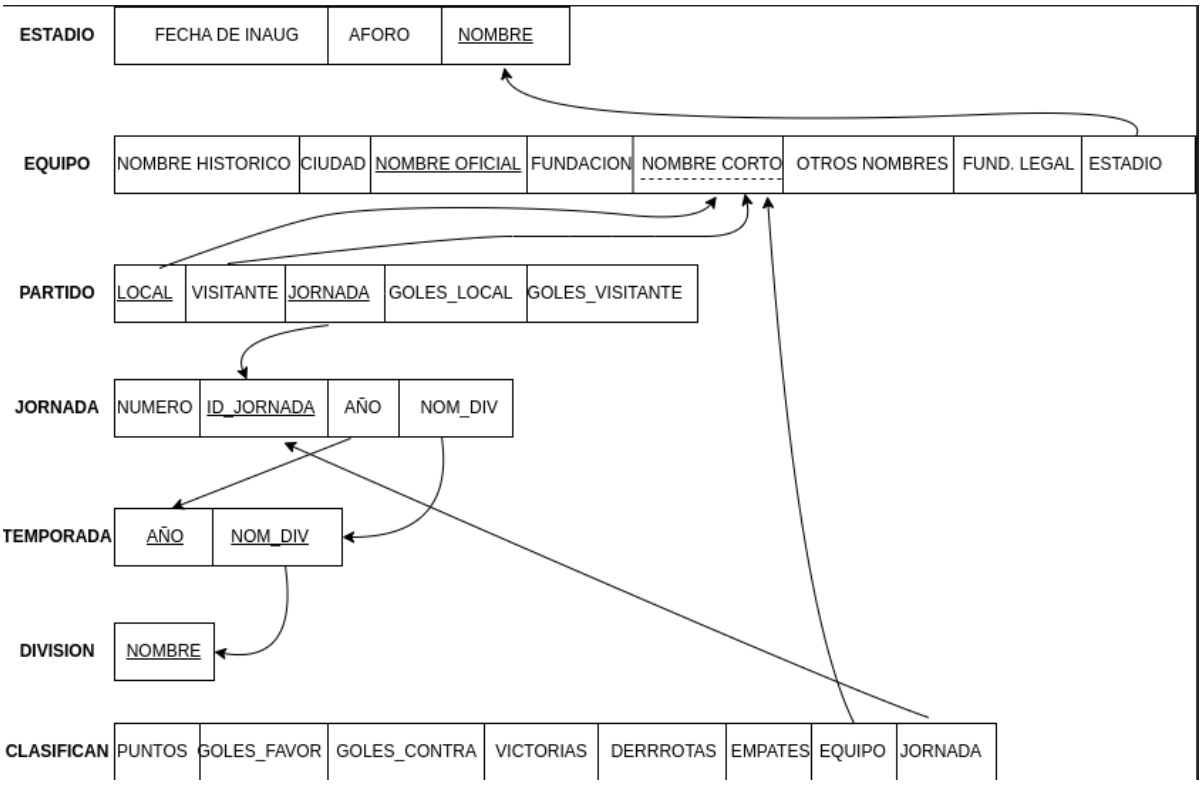
Relación “jugarse en”:

- Un partido pertenece a una jornada y una jornada tiene varios partidos

Relación “pertenecer”:

- Una división tiene varias temporadas. Cada temporada solo pertenece a una división. Temporada es débil respecto de división

Esquema relacional



Como se puede observar, se ha añadido al esquema la clave artificial de jornada “id_jornada”, la cual antes estaba compuesta por los otros tres atributos, los cuales eran claves ajenas.

De esta forma se simplifica el esquema, sustituyendo en Partido los identificadores que había anteriormente por el atributo identificativo ajeno “jornada”, al igual que en la relación Clasifican.

Normalización

El modelo E/R presentado ya está normalizado, pues para la 1FN no deben existir atributos multivaluados, caso que no se da. En una primera instancia, el atributo “otros_nombres” era multivaluado, pero no existen equipos con más de 1 valor para ese atributo, así que no se consideró esa opción.

Para la 2FN ningún atributo no clave depende funcionalmente de la clave, y esto no se da en nuestro esquema; en la 3FN no hay atributos no clave que dependan funcionalmente de otros no clave; y en la Forma Normal de Boyce-Codd ningún atributo que forme parte de la clave depende de otro que no es parte de la clave.
Todo esto se cumple en nuestro modelo, y, por lo tanto, está normalizado.

Sentencias SQL

Para la sentencias SQL de creación de tablas hay que seguir un orden específico debido a la existencia de claves ajenas. De este modo habría que comenzar añadiendo las tabla Estadio y División, que no contienen ninguna clave ajena:

```
CREATE TABLE Estadio(
    nombre VARCHAR(50) PRIMARY KEY,
    aforo NUMBER(9) CHECK (aforo > 0),
    fecha_inag NUMBER(4)
);

CREATE TABLE Division(
    nombre VARCHAR(32) PRIMARY KEY
);
```

Ahora habría que añadir Equipo y Temporada, las cuales tienen una referencia a Estadio y División respectivamente:

```
CREATE TABLE Equipo(
  nombre_oficial VARCHAR(64) PRIMARY KEY,
  nombre_historico VARCHAR(64),
  otro_nombre VARCHAR(64),
  ciudad VARCHAR (64),
  fundacion NUMBER(4),
  fund_legal NUMBER(4),
  nombre_corto VARCHAR(32) UNIQUE,
  estadio VARCHAR(64),
  FOREIGN KEY (estadio) REFERENCES Estadio(nombre)
);
```

```
CREATE TABLE Temporada(
    -- agno en el que empieza la temporada
    agno NUMBER(4),
    nom_div VARCHAR(32),
    PRIMARY KEY (agno, nom_div),
    FOREIGN KEY (nom_div) REFERENCES Division(nombre)
);
```

La siguiente tabla que se debe añadir es Jornada, que hace referencia a Temporada:

```
CREATE TABLE Jornada(
    id NUMBER GENERATED ALWAYS as IDENTITY(START with 1 INCREMENT by 1),
    numero NUMBER(3),
    agno NUMBER(4),
    nom_div VARCHAR(32),
    PRIMARY KEY (id),
    FOREIGN KEY (agno, nom_div) REFERENCES Temporada
);
```

Por último, las tablas Partido y Clasifican tienen referencias a Jornada y Equipo, por eso, aunque exista una de ellas, es indispensable la existencia de la otra para poder crearlas:

```
CREATE TABLE Partido(
    id_jornada NUMBER(5),
    nombre_corto_local VARCHAR(32),
    nombre_corto_visitante VARCHAR(32),
    goles_locales NUMBER(2),
    goles_visitantes NUMBER(2),
    PRIMARY KEY (nombre_corto_local, id_jornada),
    FOREIGN KEY (nombre_corto_local) REFERENCES Equipo(nombre_corto),
    FOREIGN KEY (nombre_corto_visitante) REFERENCES Equipo(nombre_corto),
    CHECK (nombre_corto_local != nombre_corto_visitante),
    FOREIGN KEY (id_jornada) REFERENCES Jornada(id)
);


CREATE TABLE Clasifican(
    nombre_corto_equipo VARCHAR(64),
    id_jornada NUMBER (5),
    puntos NUMBER (3),
    goles_favor NUMBER (3),
    goles_contra NUMBER (3),
    victorias NUMBER (3),
    derrotas NUMBER (3),
    empates NUMBER (3),
    PRIMARY KEY (nombre_corto_equipo, id_jornada),
    FOREIGN KEY (nombre_corto_equipo) REFERENCES Equipo(nombre_corto),
    FOREIGN KEY (id_jornada) REFERENCES Jornada(id)
);
```

PARTE 2: INTRODUCCIÓN DE DATOS Y EJECUCIÓN DE CONSULTAS

Población de la BD

Para poblar la base de datos debíamos seleccionar la columnas específicas de cada tabla en el archivo “LigasHost.csv” y eliminar los datos duplicados, para luego poblar cada tabla mediante un archivo ctl:

```
load data
 infile './csv/nombreArchivo.csv'
 into table nombreTabla
 fields terminated by ";"
 (Conjunto de los atributos de la tabla)
```

Un problema encontrado ha sido poblar la tabla Clasifican, ya que los datos no se encuentran en el csv original.
Para solucionarlo, se ha creado un código que escribe en un fichero, a medida que va leyendo la información de cada partido, una tabla por jornada con la información de los equipos que han participado en la misma.

De este modo el código escribe el nombre del equipo que ha jugado, el identificador de la jornada, los puntos acumulados hasta esa jornada a partir de la primera de ese mismo año, los goles a favor y en contra acumulados, y las victorias, derrotas y empates hasta entonces.

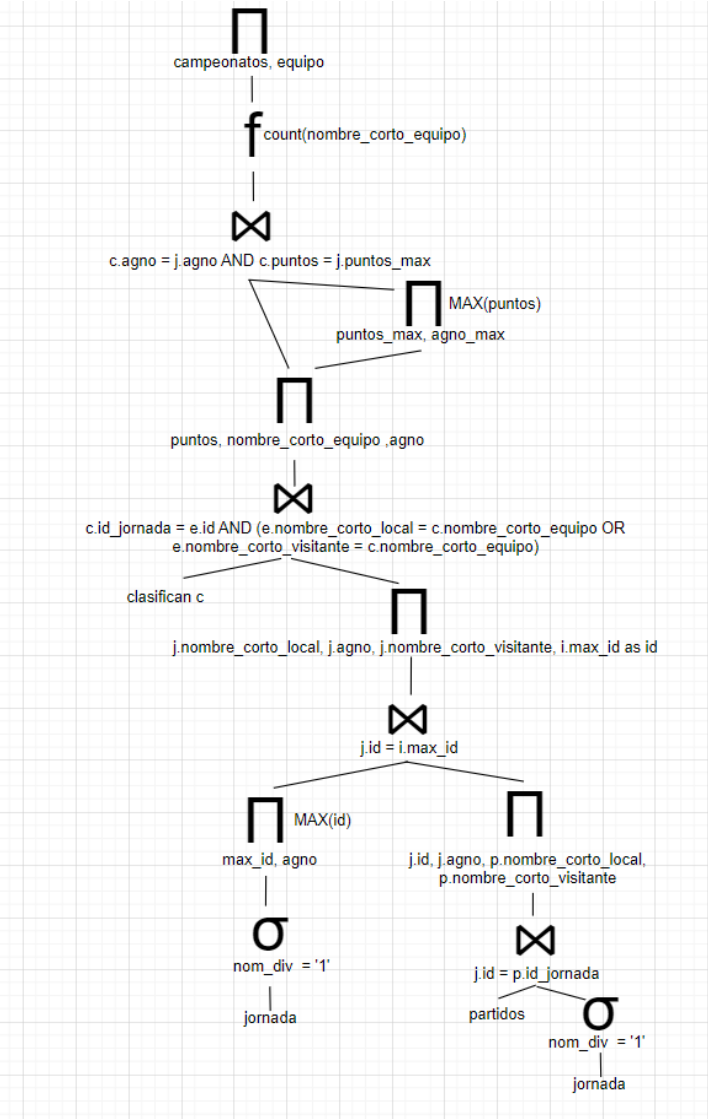
El último problema ha sido la incoherencia de los datos, pues en oracle las tildes y la ‘ñ’ no se pueden escribir, porque aparece el carácter “??”, entonces se han tenido que sustituir en todos los ficheros esos caracteres por los mismos sin la tilde, y las ‘ñ’ por ‘n’.

Consultas SQL

CONSULTA 1

Primero hemos creado una vista de todas jornadas de primera división, después hemos hecho una segunda vista con las últimas jornadas de cada año. Luego creamos una vista con todos los partidos de todas las jornadas de primera, y otra con todos los partidos de las últimas jornadas de primera. Después hacemos lo mismo que con las jornadas y partidos pero con las clasificaciones, obteniendo una vista de todas las clasificaciones en todas las jornadas y otra con las clasificaciones en las últimas jornadas. Por último, creamos una vista con los puntos que consiguió el ganador de cada año, después hacemos un join de la anterior con la clasificación de las últimas jornadas para obtener el nombre corto del equipo que ganó en cada año y finalmente una vista que obtiene las veces que aparece el nombre de cada equipo en la tabla de los ganadores de la liga. Hacemos una proyección sobre la primera línea de esa tabla en la que estará el equipo que más ligas ha obtenido, el Real Madrid con 19 ligas.

```
CREATE OR REPLACE VIEW jornadas_primera AS(
SELECT *
FROM jornada
WHERE nom_div='1');

CREATE OR REPLACE VIEW max_ids AS(
SELECT MAX(id) as max_id, agno
FROM jornada
WHERE nom_div='1'
GROUP BY agno);

CREATE OR REPLACE VIEW equipos_agno AS(
SELECT j.id, j.agno, p.nombre_corto_local,
p.nombre_corto_visitante
FROM jornadas_primera j
JOIN partido p
ON j.id = p.id_jornada);

CREATE OR REPLACE VIEW partidos_ultima_jornada
AS(
SELECT j.nombre_corto_local, j.agno,
j.nombre_corto_visitante, i.max_id as id
FROM max_ids i
JOIN equipos_agno j
ON j.id = i.max_id);

CREATE OR REPLACE VIEW clasificacion AS(
SELECT c.puntos, c.nombre_corto_equipo, e.agno
FROM partidos_ultima_jornada e
JOIN clasifican c
ON c.id_jornada = e.id AND
(e.nombre_corto_local = c.nombre_corto_equipo
OR e.nombre_corto_visitante =
c.nombre_corto_equipo));

CREATE OR REPLACE VIEW
clasificacion_ultima_jornada AS(
SELECT t.puntos, t.nombre_corto_equipo, t.agno
FROM clasificacion t);

CREATE OR REPLACE VIEW puntos_ganadores AS (
SELECT MAX(puntos) AS puntos_max, agno AS agno_max
FROM clasificacion_ultima_jornada
GROUP BY agno);

CREATE OR REPLACE VIEW ganadores AS(
SELECT *
FROM clasificacion_ultima_jornada c
JOIN puntos_ganadores j ON c.agno = j.agno_max AND c.puntos = j.puntos_max);

CREATE OR REPLACE VIEW num_ganadores AS (
SELECT COUNT(nombre_corto_equipo) AS cantidad, nombre_corto_equipo
FROM ganadores
GROUP BY nombre_corto_equipo);

SELECT cantidad AS campeonatos, nombre_corto_equipo AS equipo
FROM num_ganadores
ORDER BY cantidad DESC
```



```
FETCH FIRST ROW ONLY;
```

Esta es la salida de la consulta primera:

```
CAMPEONATOS  EQUIPO
-----------  ----------------
          19  Real Madrid
```

CONSULTA 2

```
CREATE OR REPLACE VIEW ganados_local AS(
SELECT COUNT(*) as partidos_ganados, s1.nombre AS nombre_estadio
FROM estadio s1
JOIN equipo e1 ON s1.nombre = e1.estadio
JOIN partido p1 ON e1.nombre_corto = p1.nombre_corto_local
WHERE p1.goles_locales >= p1.goles_visitantes
GROUP BY s1.nombre);
--Vista que contiene la cantidad de partidos en los que ha ganado
--el local junto con el estadio donde se han ganado

CREATE OR REPLACE VIEW jugados_local AS(
SELECT COUNT(*) as partidos_jugados, s2.nombre AS nombre_estadio
FROM estadio s2
JOIN equipo e2 ON s2.nombre = e2.estadio
JOIN partido p2 ON e2.nombre_corto = p2.nombre_corto_local
GROUP BY s2.nombre);
--Vista que contiene la cantidad partidos que se han jugado en cada estadio
```

```
SELECT s1.nombre_estadio AS
estadio ,
partidos_ganados/partidos_jugados
AS ratio
FROM ganados_local s1
JOIN jugados_local s2
ON s1.nombre_estadio =
s2.nombre_estadio
WHERE
partidos_ganados/partidos_jugados
> (85/100));
--Muestra la tabla solo con el
ratio de victorias y el estadio
```

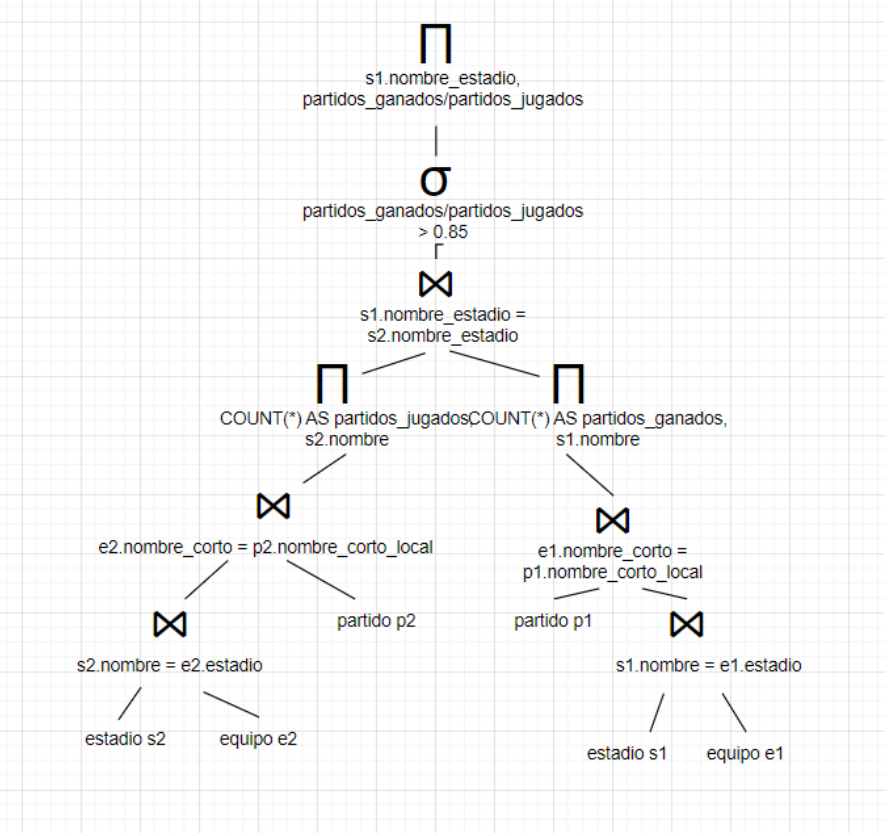


Tabla de resultados:

```
ESTADIO                    RATIO
-------------------------- ---------
Nuevo Lasesarre                 ,91228070
Santiago Bernabeu          ,90841584
Camp Nou                   ,91460396
Narcis Sala                ,87368421
```

CONSULTA 3

```
CREATE OR REPLACE VIEW rivales_perdieron_romareda AS(
SELECT p.nombre_corto_visitante, j.agno, j.numero
FROM partido p
JOIN jornada j ON p.id_jornada = j.id AND p.nombre_corto_local = 'Zaragoza' AND
(j.nom_div='1' OR j.nom_div='2')
WHERE p.goles_locales > p.goles_visitantes);
--Vista de los rivales, el año y la jornada en la que perdieron contra el Zaragoza
en la Romareda(estadio del Zaragoza)

CREATE OR REPLACE VIEW rivales_perdieron_su_casa AS(
SELECT j.agno, p.nombre_corto_local, j.numero
FROM partido p
JOIN jornada j ON p.id_jornada = j.id AND (p.nombre_corto_visitante = 'Zaragoza')
WHERE p.goles_locales < p.goles_visitantes);
--Vista de los rivales, el año y la jornada en la que perdieron contra el Zaragoza
en su propio estadio

CREATE OR REPLACE VIEW cantidad_casos_agno AS(
SELECT COUNT(*) AS cuentas, j.agno
FROM rivales_perdieron_romareda t
JOIN rivales_perdieron_su_casa j
ON (j.nombre_corto_local = t.nombre_corto_visitante) AND j.agno=t.agno
GROUP BY j.agno
HAVING COUNT(*) >=4);
--Vista de la los años y cuantos equipos ganó el zaragoza en su campo y en el del
rival


CREATE OR REPLACE VIEW id_max_2 AS (
SELECT MAX(id) AS max_id, agno
FROM jornada
WHERE nom_div='2'
GROUP BY agno);
--Vista de las últimas jornadas de las temporadas segunda división

CREATE OR REPLACE VIEW  id_max_1 AS (
SELECT MAX(id) AS max_id, agno
FROM jornada
WHERE nom_div='1'
GROUP BY agno);
--Vista de las últimas jornadas de las temporadas de primera división


CREATE OR REPLACE VIEW goles_favor_zgz AS(
```

```
SELECT goles_favor, id_jornada
FROM clasifican
WHERE nombre_corto_equipo='Zaragoza');
--Vista de los goles a favor del zaragoza en cada jornada de cada temporada (en
todas las divisiones)

SELECT a.agno, c.goles_favor, a.cuentas AS ocurrencias
FROM id_max_1 i1
JOIN id_max_2 i2
ON i1.agno = i2.agno
JOIN cantidad_casos_agno a
ON (a.agno = i1.agno) OR (a.agno = i2.agno)
JOIN goles_favor_zgz c
ON (c.id_jornada = i1.max_id) OR (c.id_jornada = i2.max_id)
ORDER BY agno;
--Selección de los años, los goles a favor y las veces en las que el zaragoza ganó
a más de cuatro equipos en su casa y fuera de casa
```

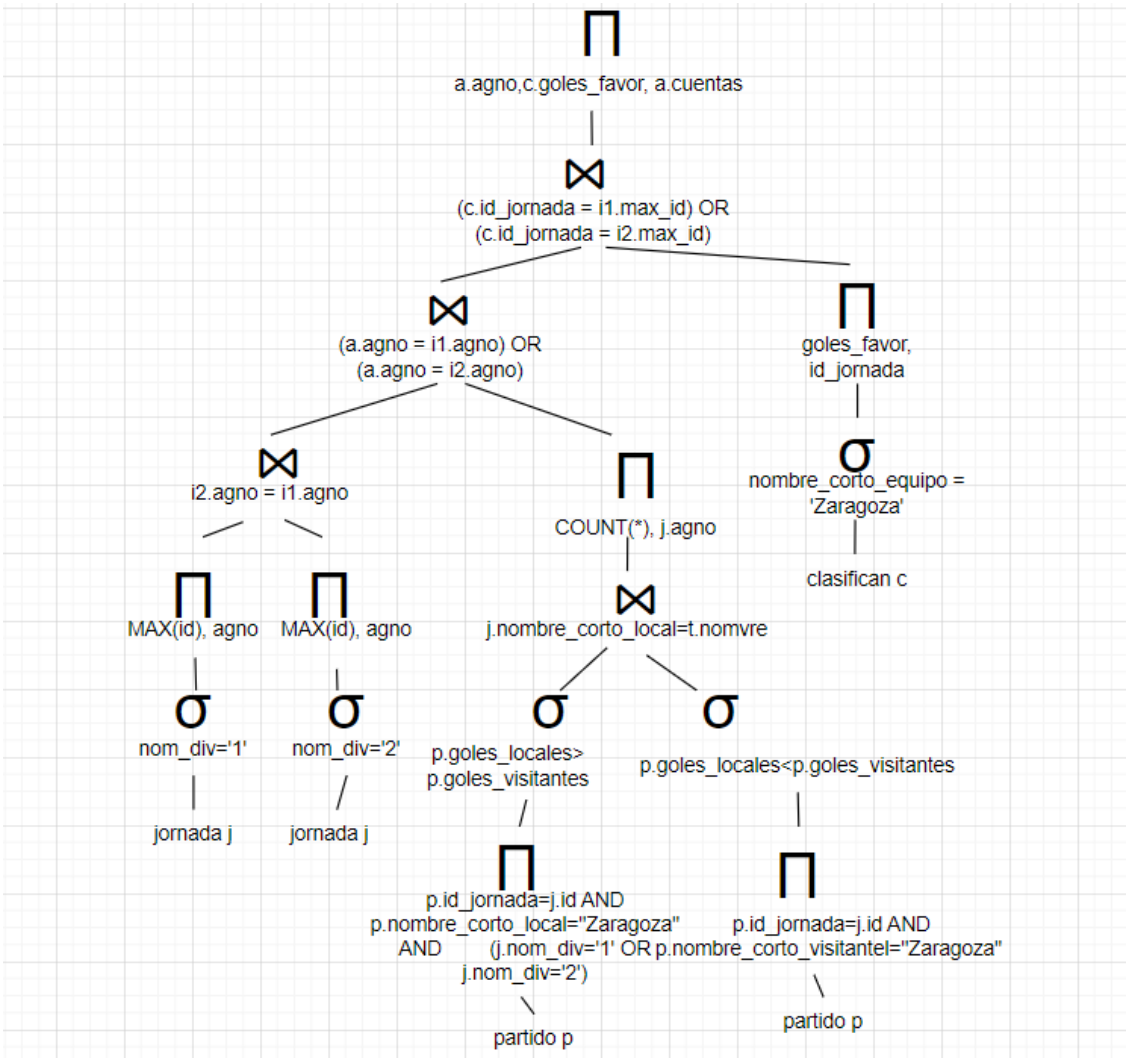


TABLA DE RESULTADOS

AGNO	GOLES_FAVOR	OCURRENCIAS
1982	59	5
1985	51	4
1998	57	4
2002	54	6
2008	79	6

PARTE 3: DISEÑO FÍSICO

OPTIMIZACIÓN DE CÓDIGO

Al ejecutar el comando "EXPLAIN PLAN FOR <consulta SQL>" aparecen como problemas de rendimiento en la consulta 1:

La operación HASH_JOIN es el de mayor coste en memoria con 4837 K, por otro lado la operación de mayor coste a nivel de procesador es el SORT ORDER BY, la cual no es imprescindible para la consulta, pero es óptima para visualizar los datos como usuario.
Otras operaciones de gran costo, son SELECT STATEMENT y otros HASH_JOIN.

Como una solución a los problemas de rendimiento hemos decidido emplear índices para disminuir el tiempo ya que acelera la búsqueda y recuperación de datos, aunque también ocupan espacio en disco.
Estos son los índices que utilizaremos para mejorar el rendimiento:

```
CREATE INDEX idx_jornada_nom_div ON jornada(nom_div);
CREATE INDEX idx_jornada_nom_div_agno ON jornada(nom_div, agno);
CREATE INDEX idx_partido_id_jornada ON partido(id_jornada);
CREATE INDEX idx_clasifican_id_jornada ON clasifican(id_jornada);
```

Aunque el uso de vistas materializadas sea más rápido a la hora de ejecutar las consultas, no hemos optado por esa opción debido a que estas se almacenan en disco, y ocupan un gran de más espacio en este. Además como suponemos que las tablas están en constante evolución (se añaden los resultados de los partidos de las ligas profesionales jugados cada semana) y disponemos de un espacio limitado, no sale a cuenta utilizarlas ya que creemos que ganaríamos tiempo a la hora de hacer las consultas pero perderíamos tiempo cuando se realicen las actualizaciones de la base de datos semanal.

Al analizar la segunda consulta, hemos considerado que estos índices podrían mejorar el rendimiento de la consulta, y así lo demuestran los resultados del realizar el EXPLAIN PLAN.

```
CREATE INDEX idx_estadio_nombre ON estadio(nombre);
CREATE INDEX idx_equipo_estadio ON equipo(estadio);
CREATE INDEX idx_partido_local ON partido(nombre_corto_local);
CREATE INDEX idx_partido_goles_local ON partido(goles_locales,
goles_visitantes);
```

Para esta segunda consulta encontramos 2 operaciones con un coste en memoria grande, 22 M, que son un HASH_JOIN y un SELECT STATEMENT, a su vez estas operaciones son las más costosas a nivel de CPU con diferencia,

Para la consulta 3 encontramos muchas más operaciones siendo ejecutadas. La mayoría no son demasiado costosos pero destaca una operación MERGE JOIN CARTESIAN con una ocupación en memoria de 1230 K y que trabaja con más de 24000 tuplas. A nivel de coste de CPU encontramos 2 operaciones que llegan hasta el 17% de utilización de la misma, que se repiten varias veces, estas operaciones son un HASH GROUP BY y una VIEW
Las optimizaciones que hemos visto oportunas utilizando vistas son las siguientes:

```
CREATE INDEX idx_partido_id_jornada ON partido(id_jornada);
CREATE INDEX idx_partido_local ON partido(nombre_corto_local);
CREATE INDEX idx_jornada_nom_div,agno ON jornada(nom_div, agno);

CREATE INDEX idx_partido_visitante ON
partido(nombre_corto_visitante);
CREATE INDEX idx_clasifican_nombre_corto ON
clasifican(nombre_corto_equipo);
CREATE INDEX idx_clasifican_id_jornada ON clasifican(id_jornada);
```

Algunos de estos índices ya han sido creados para las consultas anteriores, por lo cual no haría falta volverlas a crear si ya los tuvieses creados de antes.

COORDINACIÓN DE LA BD

Tarea	Miembros Encargados	Cantidad Trabajo	Observaciones
Modelo E/R	Todos	7 horas	No demasiado trabajo en casa, casi todo en las horas de prácticas de la asignatura
Esquema relacional	Todos	1 hora	Al tener el modelo E/R fue fácil completarlo
Normalización del ER	Todos	2 horas	Fue difícil el inicio ya que no teníamos claros los conceptos necesarios
Ficheros para crear tablas	Todos	2 horas	Nos costó adaptarnos a la sintaxis del cargador y el manejo de los ficheros csv
Sentencias SQL	Todos	4 horas	Fuimos resolviendo multitud de errores que surgieron a medida que avanzábamos
Población de tablas	Todos	1 hora	Teníamos los ficheros csv, no fue demasiado complejo
Consultas SQL	Todos	8 horas	La parte más compleja de la práctica
Árboles relacionales	Todos	2 horas	No fue demasiado complejo pero sí extenso
Optimización	Todos	3 horas	Confusión en el output de los comandos

Para la realización de esta práctica, trabajamos cada uno en nuestra casa haciendo uso de documentos compartidos para poder trabajar al mismo tiempo. Además estábamos constantemente en contacto por videollamada. Las horas de prácticas han sido de gran ayuda ya que las dudas planteadas fueron resueltas.