PRÁCTICA 1

LIGA DE FÚTBOL

BASES DE DATOS

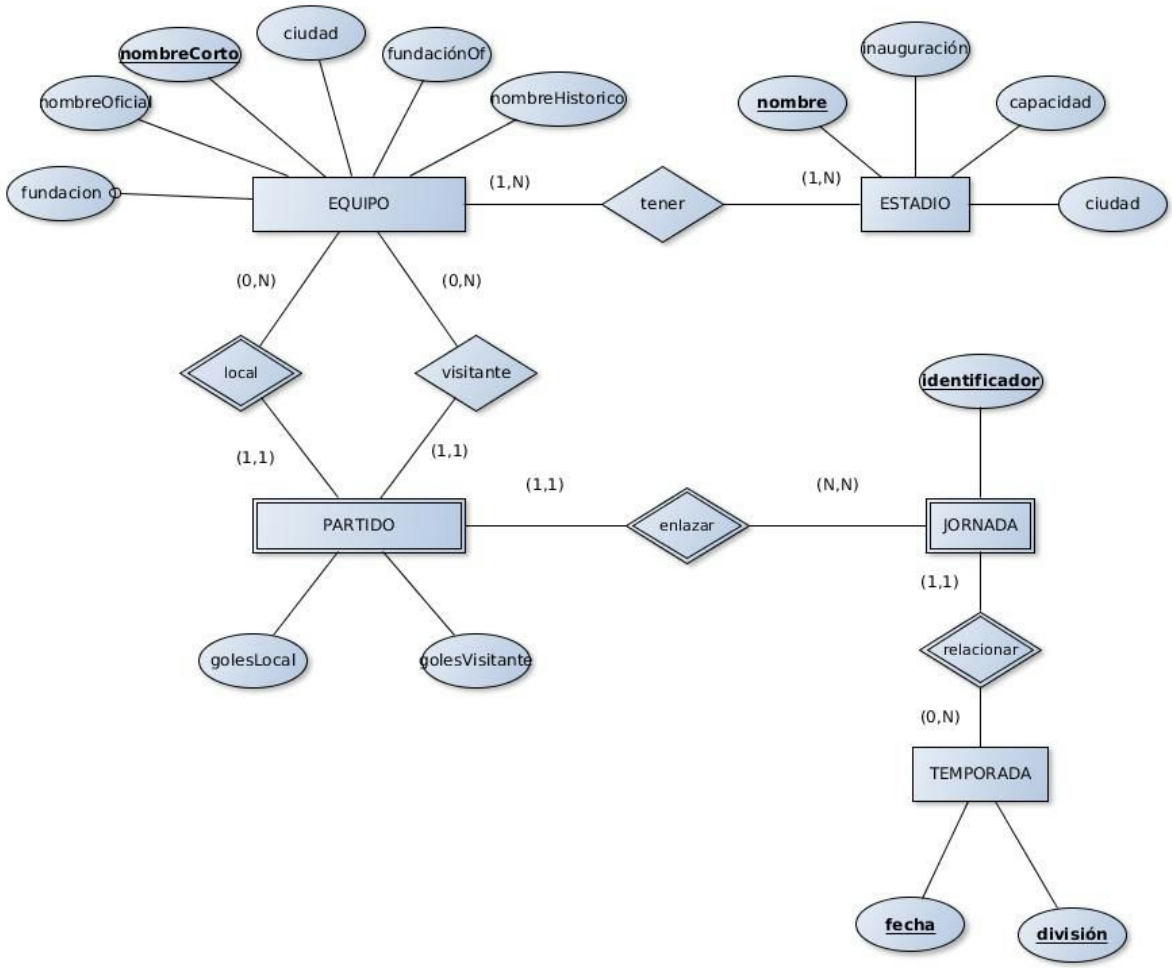
Jorge Sanz Alcaine, 680182
Álvaro Monteagudo Moreno, 681060
Daniel Rueda Macías, 559207

Total horas dedicadas: 100h

Universidad Zaragoza

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

PARTE 1
MODELO ENTIDAD-RELACIÓN



El esquema entidad-relación se compone de cinco entidades. La entidad equipo representa un equipo de fútbol de la liga de fútbol española. Está compuesto por su nombre corto el cual es la clave primaria, su nombre oficial, su nombre histórico, su ciudad, su fundación oficial y su fundación, el cual es un atributo opcional.

La entidad estadio que representa un estadio de fútbol. Sus atributos son su nombre, el cual es la clave primaria, su año de inauguración, su capacidad en número de personas y la ciudad donde se encuentra dicho estadio.

La entidad partido representa un partido de fútbol, es una entidad débil que depende de equipo y de jornada. Sus atributos son golesLocal, correspondiente al número de goles que ha marcado el equipo local y golesVisitante, correspondiente al número de goles que ha marcado el equipo visitante.

La entidad jornada representa una jornada de la liga de fútbol española, depende de temporada y de partido, ya que sin un partido y sin una temporada, la existencia de esta es imposible. Su clave primaria es un identificador el cual es un número que se corresponde con el número de jornada.

Y por último, la entidad temporada que representa una temporada de fútbol de la liga española. Está compuesta por los atributos fecha, el cual es una clave primaria y de división, el cual es la división a la que pertenece esa temporada (1ª o 2ª).

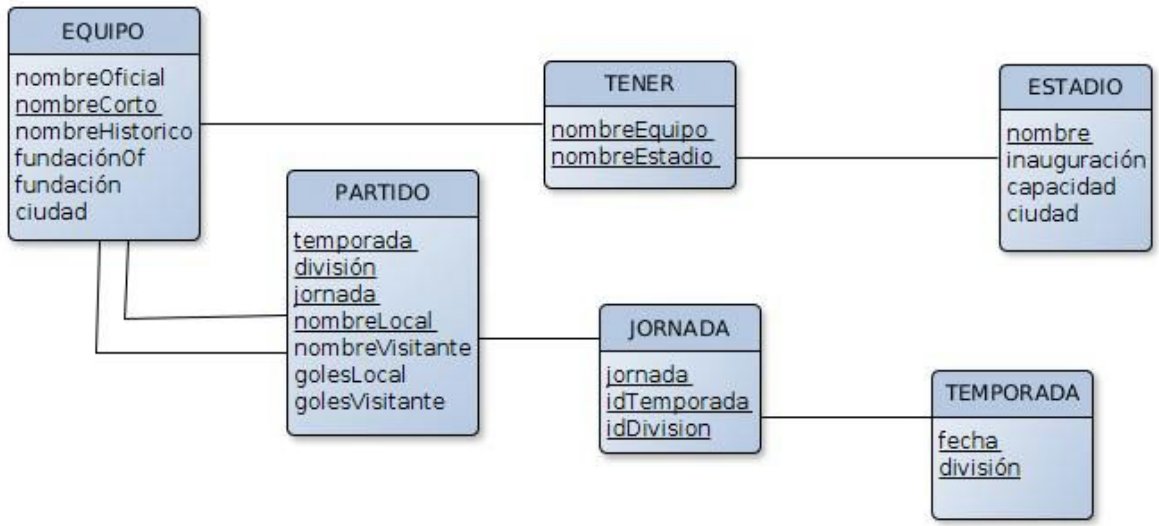
En lo que respecta a las relaciones entre las entidades, está la relación tener, que conecta equipo y estadio. Un equipo puede tener varios estadios asignados y un estadio puede albergar a varios equipos.

Un equipo puede jugar como equipo local o como equipo visitante en un partido. Siendo local una relación débil que depende de equipo y que hace que la entidad partido dependa de este. Un equipo puede jugar varios partidos como local y un partido tiene un solo equipo como local. Lo mismo sucede para la relación visitante salvo que esta no es una relación débil.

La relación enlazar enlaza un partido con una jornada, ya que un partido pertenece a una determinada jornada de liga. Un partido enlaza con una sola jornada y una jornada puede enlazar con varios partidos. Es una relación débil respecto de partido.

Por último, la relación relacionar que conecta temporada y jornada. Una jornada se relaciona con una temporada y una temporada tiene varias jornadas.

MODELO RELACIONAL



FORMAS NORMALES

1. Está en primera porque no presenta atributos multivaluados.
2. Está en segunda forma porque los atributos no primarios dependen siempre de toda la clave.
3. Está en tercera forma normal porque ningún atributo no primario depende de otro no primario.
4. Está en cuarta forma normal porque no hay dependencias multivaluadas.
5. Está en quinta forma normal porque no es posible evitar los ciclos.

CREACIÓN DE TABLAS SQL

```
/* CREACION DE LAS TABLAS SEGUN EL MODELO RELACIONAL */

CREATE TABLE EQUIPO (
 nombreOficial VARCHAR(40) NOT NULL,
 nombreCorto VARCHAR(32) PRIMARY KEY NOT NULL,
 nombreHistórico VARCHAR(40),
 ciudad VARCHAR(32) NOT NULL,
 fundacionOf NUMBER(4) NOT NULL,
 fundación NUMBER(4)
);
```

```
CREATE TABLE ESTADIO (
 nombre VARCHAR(32) PRIMARY KEY NOT NULL,
 capacidad NUMBER(6) NOT NULL,
 inauguración NUMBER(4) NOT NULL,
 ciudad VARCHAR(32) NOT NULL
);

CREATE TABLE TENER (
 nombreEquipo VARCHAR(32) NOT NULL,
 nombreEstadio VARCHAR(32) NOT NULL,
 FOREIGN KEY (nombreEquipo) REFERENCES EQUIPO(nombreCorto) ON DELETE CASCADE,
 FOREIGN KEY (nombreEstadio) REFERENCES ESTADIO(nombre) ON DELETE CASCADE,
 PRIMARY KEY (nombreEquipo,nombreEstadio)
);

CREATE TABLE TEMPORADA (
 fecha VARCHAR(32),
 division VARCHAR(2),
 PRIMARY KEY (fecha,division)
);

CREATE TABLE JORNADA (
 jornada NUMBER(2),
 idTemporada VARCHAR(32),
 idDivision VARCHAR (2),
 FOREIGN KEY (idTemporada,idDivision) REFERENCES TEMPORADA(fecha,division) ON DELETE
CASCADE,
 PRIMARY KEY (jornada,idTemporada,idDivision)
);

CREATE TABLE PARTIDOS (
 temporada VARCHAR(32),
 division VARCHAR(2),
 jornada NUMBER(2),
 nombreLocal VARCHAR(32),
 nombreVisitante VARCHAR(32),
 golesLocal NUMBER(2) NOT  NULL,
 golesVisitante NUMBER(2) NOT NULL,
 FOREIGN KEY (jornada,temporada,division) REFERENCES
JORNADA(jornada,idTemporada,idDivision) ON DELETE CASCADE,
 FOREIGN KEY (nombreLocal) REFERENCES EQUIPO(nombreCorto) ON DELETE CASCADE,
 FOREIGN KEY (nombreVisitante) REFERENCES EQUIPO(nombreCorto) ON DELETE CASCADE,
 PRIMARY KEY (nombreLocal,jornada,temporada,division)
);
```

Aspectos a comentar:

- En la tabla equipo en un principio se eligió como clave primaria, pero como al obtener el fichero con los partidos se vió que se usaba el nombre corto, se decidió cambiarla por este. El atributo fundación como se ha dicho que podía ser opcional no se le exige ser no nulo.
- En la tabla temporada la clave primaria ha de estar compuesta por los dos atributos ya que si no fuera así se darían los casos de que habría claves primarias iguales.
- El mismo caso se dió para la tabla jornada. Se puso en un principio como clave primaria el par idTemporada, idDivision, pero sucedió que había claves primarias repetidas y se cambió la composición de la clave primaria a los tres atributos.

DATOS SOBRE REUNIONES Y DIVISIÓN DE TRABAJO

Horas dedicadas: 10h.

PARTE 2

POBLACIÓN DE LA BASE DE DATOS

Antes de nada se va a mencionar las tecnologías utilizadas para el filtrado de datos y la población de estos:

- Java
- Microsoft excel o similares
- Sublime text
- Aplicación de terminal

Filtrado de datos

Primero se empezaron filtrando los equipos y los estadios. Se copiaban los datos de la página a un excel, y mediante éste se iban borrando las imágenes y se eliminaban las columnas que no se necesitaban. Se realizaba lo mismo para los estadios. Una vez obtenidas las tablas como se querían, se guardaban como fichero de extensión .csv, con los campos entrecomillados entre comillas simples y separados por coma, con la finalidad de que se fuera acercando a la sintaxis de inserción sql. Luego mediante un programa java se obtenía el fichero de inserción sql.

Para las temporadas y las jornadas se consultaron las páginas y se hicieron los ficheros de inserción a mano.

Para los partidos, se descargaron los ficheros de los partidos de toda la liga y mediante un programa java se generaba el fichero de inserción sql.

Por último el fichero de inserción de la tabla TENER se hizo a mano.

INSERCIÓN DE LOS DATOS EN LA BASE

Aquí viene uno de los mayores problemas y donde más tiempo se tuvo que invertir.

Una vez con las tablas creadas, habiendo resuelto unos problemas a la hora de su creación, se empezaron a insertar los estadios. Aparecieron los primeros errores, sql no permitía insertar caracteres con acentos. Se procedió a hacer un programa

java que eliminara los acentos. Se pasó el programa por todos los ficheros de inserción que contenían caracteres.

Se volvió a probar a insertar los estadios sin acentos. Pero volvió a surgir otro problema, al estar los campos char entre comillas simples, algunos nombres de las ciudades de algunos estadios al estar en catalán el apóstrofe estropeaba toda la sintaxis del fichero de inserción. Por tanto, los apóstrofes tenían que ser eliminados.

Una vez insertadas todas las tablas menos la tabla partidos, se procedió a insertar esta última, pero aquí es donde vino el mayor problema de todos a la hora de insertar los datos. Muchos nombres de los equipos tanto local como visitante, que hacían referencia a la clave primaria de la tabla equipos, no podían hacerlo, ya que por ejemplo si en la tabla PARTIDOS el nombre del equipo era “Real Madrid”, en la tabla de equipos el identificador al que hacía referencia era “Madrid”. Y el método que siempre se seguía era el mismo, una vez que se insertaba y se generaban los errores, se seleccionaba todo el texto de la terminal y se copiaba a un documento nuevo en el editor de texto Sublime text y, mediante expresiones regulares se quitaba todo el feedback de las filas que se habían insertado bien (1 row created), y así, solo se tenían los errores. Una vez con los errores filtrados, se miraba si el error que se producía era si el nombre del equipo no se correspondía con su identificador, en ese caso se cambiaba la clave primaria del fichero de equipos por el nombre del equipo del fichero de partidos, se vaciaba la tabla de equipos y se volvía a introducir en sql. Podía darse el caso también, de que dicho equipo no estuviera en el fichero de equipos, la solución para esto era buscar en internet y generar a mano una instrucción de inserción sql con todos los datos del equipo.

Al principio se quitaban los errores bastante rápido, ya que con un equipo que estaba mal había cientos o miles de coincidencias. Entonces de una atacada se solucionaban muchos errores, pero luego para buscar los errores más pequeños fue más difícil.

DATOS ESTADÍSTICOS DE LAS TABLAS

	Equipo	Estadio	Tener	Temporada	Jornada	Partidos
Número de tuplas	121	169	80	84	3378	32474

Tamaño en KBytes	13,5	12,0	4,5	5,6	279,6	2560

Lista de las primeras 5 tuplas de cada tabla

EQUIPO

NOMBREOFICIAL

NOMBRECORTO

NOMBREHISTÓRICO

CIUDAD

FUNDACIONOF FUNDACIÓN

Real Club Recreativo de Huelva
Rec._Huelva
Huelva Recreation Club

NOMBREOFICIAL

NOMBRECORTO

NOMBREHISTÓRICO

CIUDAD

FUNDACIONOF FUNDACIÓN

Huelva
1889

NOMBREOFICIAL

NOMBRECORTO

NOMBREHISTÓRICO

CIUDAD

FUNDACIONOF FUNDACIÓN

Athletic Club
Ath._Bilbao
Athletic Club

NOMBREOFICIAL

NOMBRECORTO

NOMBREHISTÓRICO

CIUDAD

FUNDACIONOF FUNDACIÓN
---------- ----------
Bilbao
      1898   1901

NOMBREOFICIAL

NOMBRECORTO

NOMBREHISTÓRICO

CIUDAD

FUNDACIONOF FUNDACIÓN
---------- ----------
Futbol Club Barcelona
Barcelona
Foot-Ball Club Barcelona

NOMBREOFICIAL

NOMBRECORTO

NOMBREHISTÓRICO

CIUDAD

FUNDACIONOF FUNDACIÓN
---------- ----------
Barcelona
      1899

NOMBREOFICIAL

NOMBRECORTO

NOMBREHISTÓRICO

CIUDAD

FUNDACIONOF FUNDACIÓN
---------- ----------
Real Club Deportivo Espanol
Espanyol
Sociedad Espanola de Foot-Ball

NOMBREOFICIAL

NOMBRECORTO

NOMBREHISTÓRICO

CIUDAD

FUNDACIONOF FUNDACIÓN
---------- ----------
Barcelona
1900

ESTADIO

NOMBRE

CAPACIDAD INAUGURACIÓN
---------- ----------
CIUDAD

Camp Nou
99354 1957
Barcelona

Santiago Bernabeu
81044 1947
Madrid

NOMBRE

CAPACIDAD INAUGURACIÓN
---------- ----------
CIUDAD

Olimpico de Madrid
67500 2016
Madrid

Olimpico de La Cartuja
60000 1997

NOMBRE

CAPACIDAD INAUGURACIÓN
--- ---
CIUDAD

Sevilla

TENER

NOMBRE

CAPACIDAD INAUGURACIÓN
--- ---
CIUDAD

Camp Nou
99354 1957
Barcelona

Santiago Bernabeu
81044 1947
Madrid

NOMBRE

CAPACIDAD INAUGURACIÓN
--- ---
CIUDAD

Olimpico de Madrid
67500 2016
Madrid

Olimpico de La Cartuja
60000 1997

NOMBRE

CAPACIDAD INAUGURACIÓN
--- ---
CIUDAD

Sevilla

SQL> select * from tener where rownum<5;

NOMBREEQUIPO

NOMBREESTADIO

Alaves
Mendizorroza

Albacete
Carlos Belmonte

Alcorcon
Santo Domingo 2

NOMBREEQUIPO

NOMBREESTADIO

Algeciras
Nuevo Mirador

TEMPORADA

FECHA

DIVISI

1972-1973
1

1972-1973
2

1973-1974
1

FECHA

DIVISI

1973-1974
2

JORNADA

JORNADA

IDTEMPORADA

IDDIVI

1
1972-1973
1

1
1972-1973
2

JORNADA

IDTEMPORADA

IDDIVI

1
1973-1974
1

1
1973-1974

JORNADA

IDTEMPORADA

IDDIVI

2

PARTIDOS

TEMPORADA

DIVISI JORNADA
--- ---
NOMBRELOCAL

NOMBREVISITANTE

GOLESLOCAL GOLESVISITANTE
--- ---
1972-1973
1 1
Granada

TEMPORADA

DIVISI JORNADA
------ ----------
NOMBRELOCAL

NOMBREVISITANTE

GOLESLOCAL GOLESVISITANTE
---------- --------------
Zaragoza
0 0

TEMPORADA

DIVISI JORNADA
------ ----------
NOMBRELOCAL

NOMBREVISITANTE

GOLESLOCAL GOLESVISITANTE
---------- --------------
1972-1973
1 1
Barcelona

TEMPORADA

DIVISI JORNADA
------ ----------
NOMBRELOCAL

NOMBREVISITANTE

GOLESLOCAL GOLESVISITANTE
---------- --------------
Dptivo._Coruna
3 1

TEMPORADA

DIVISI JORNADA
------ ----------
NOMBRELOCAL

NOMBREVISITANTE

GOLESLOCAL GOLESVISITANTE

---------- ----------
1972-1973
1 1
At._Madrid

TEMPORADA

DIVISI JORNADA
------ ----------
NOMBRELOCAL

NOMBREVISITANTE

GOLESLOCAL GOLESVISITANTE
---------- ----------
Valencia
1 3

TEMPORADA

DIVISI JORNADA
------ ----------
NOMBRELOCAL

NOMBREVISITANTE

GOLESLOCAL GOLESVISITANTE
---------- ----------
1972-1973
1 1
Las_Palmas

TEMPORADA

DIVISI JORNADA
------ ----------
NOMBRELOCAL

NOMBREVISITANTE

GOLESLOCAL GOLESVISITANTE
---------- ----------
Oviedo
2 1

CONSULTAS

Consulta 1

En la consulta 1 se nos pide los equipos que han ganado una liga no estando entre los 3 primeros equipos a mitad de liga.

Primero se empezó mirando el sistema de puntuación y clasificación de la liga de fútbol española. A los equipos ganadores se le asignan 3 puntos, a los perdedores 0 y si hay empate un punto a cada uno. Partiendo de esta idea, se formaron unas tablas de puntuación de equipos en cada jornada de cada liga de cada temporada seleccionando los campos temporada, división, jornada, nombre del equipo y una cláusula CASE para asignar la puntuación. Obviamente hay que asignar la puntuación tanto al equipo local como al equipo visitante, por ello, se construyeron dos tablas, tabla para los equipos locales y tabla para los equipos visitantes. A continuación, se formó una tabla nueva la cual es la unión de las dos tablas.

Una vez con esta tabla en nuestro poder, lo que se hizo es sumar los puntos de cada equipo de cada jornada de cada temporada de cada división. Lo siguiente a hacer, es sumar las puntuaciones de todas las jornadas de cada temporada de cada división. Pero claro, no todas las temporadas de cada división tienen las mismas jornadas, entonces, lo que se necesita son el número de jornadas que tuvo cada temporada de cada división. Por ello, se hizo una consulta a la tabla de temporadas para sacar el número de jornadas de cada temporada de cada división.

Con estas dos tablas, se obtuvo la suma de los puntos para cada división de cada temporada y esta misma suma pero a mitad de liga (solo hay que cambiar el número de jornadas por número de jornadas dividido para 2). Bien, se ha llegado al punto en el que se tienen los puntos a final y mitad de liga. Ahora con estas tablas resultantes se puede construir lo que se pedía en el enunciado. Los objetivos a realizar son obtener el campeón de cada temporada de cada división y los equipos que no han estado entre los 3 primeros a mitad de liga.

Para ello, se necesita una clasificación y esta clasificación la vamos a obtener gracias a la función row_number(). Lo que permite esta función es generar una columna auxiliar la cual contiene índices según el orden y la partición que se indique en la función over(), función que sigue a la anterior. En la función over se indicó que partiera los datos por temporada y división y que los ordenara en orden descendiente por temporada, división y puntos a final y mitad de liga respectivamente para cada tabla.

Se procedió a crear dos vistas. Una con los campeones de liga y otra que contenía los equipos que no estaban entre los 3 primeros a mitad de liga. Para la primera, se hizo una consulta seleccionando los nombres de equipos (sin repeticiones, ya que hay equipos que han ganado varias veces la liga) cuyo puesto es igual a 1 de la clasificación a final de liga. Y para la segunda, se seleccionó de la clasificación a mitad de liga, los nombres de equipo (sin repeticiones), cuyo puesto en la liga es mayor que 3.

Para finalizar la consulta que se pedía en el enunciado se seleccionó el nombre de los elementos comunes en nombre entre las dos vistas (INNER JOIN entre nombres).

Árbol sintáctico de álgebra relacional

CONSULTA 1

$R1 = \Pi_{temporada, division, jornada, nombreVisitante, 3}(\sigma_{golesLocal < golesVisitante}(PARTIDOS))$

$R2 = \Pi_{temporada, division, jornada, nombreLocal, 1}(\sigma_{golesLocal = golesVisitante}(PARTIDOS))$

$R3 = \Pi_{temporada, division, jornada, nombreVisitante, 1}(\sigma_{golesLocal = golesVisitante}(PARTIDOS))$

$R4 = \Pi_{temporada, division, jornada, nombreLocal, 3}(\sigma_{golesLocal > golesVisitante}(PARTIDOS))$

$R5(temporada, division, jornada, nombre, puntos) = R1 \cup R2 \cup R3 \cup R4$

$R6 = AGRUPAR_{SUMA(puntos)}(R5, temporada, division, jornada, nombre)$

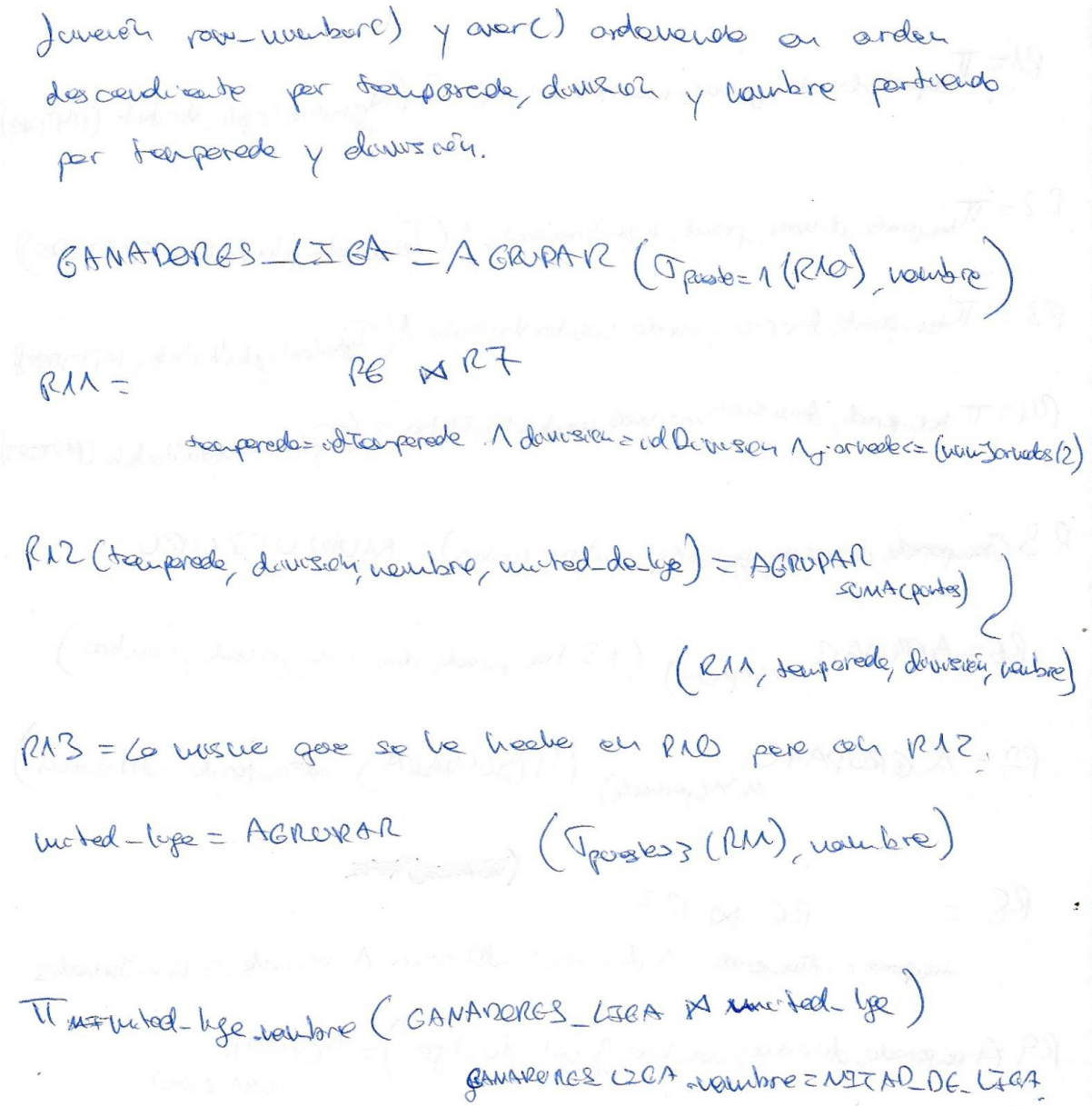
$R7 = AGRUPAR_{MAX(jornada)}(\sigma_{JORNADA}, idTemporada, idDivision)$

$R8 = R6 \bowtie R7$

$temporada = idTemporada \wedge division = idDivision \wedge jornada <= numJornadas$

$R9(temporada, division, nombre, nivel\_de\_liga) = AGRUPAR_{SUMA(puntos)}(R8, temporada, division, nombre)$

Selección de R9 y se añade el puesto de la liga mediante la función



Respuesta Obtenida

Respuesta obtenida en el fichero respuestasConsultas.

Consulta 2

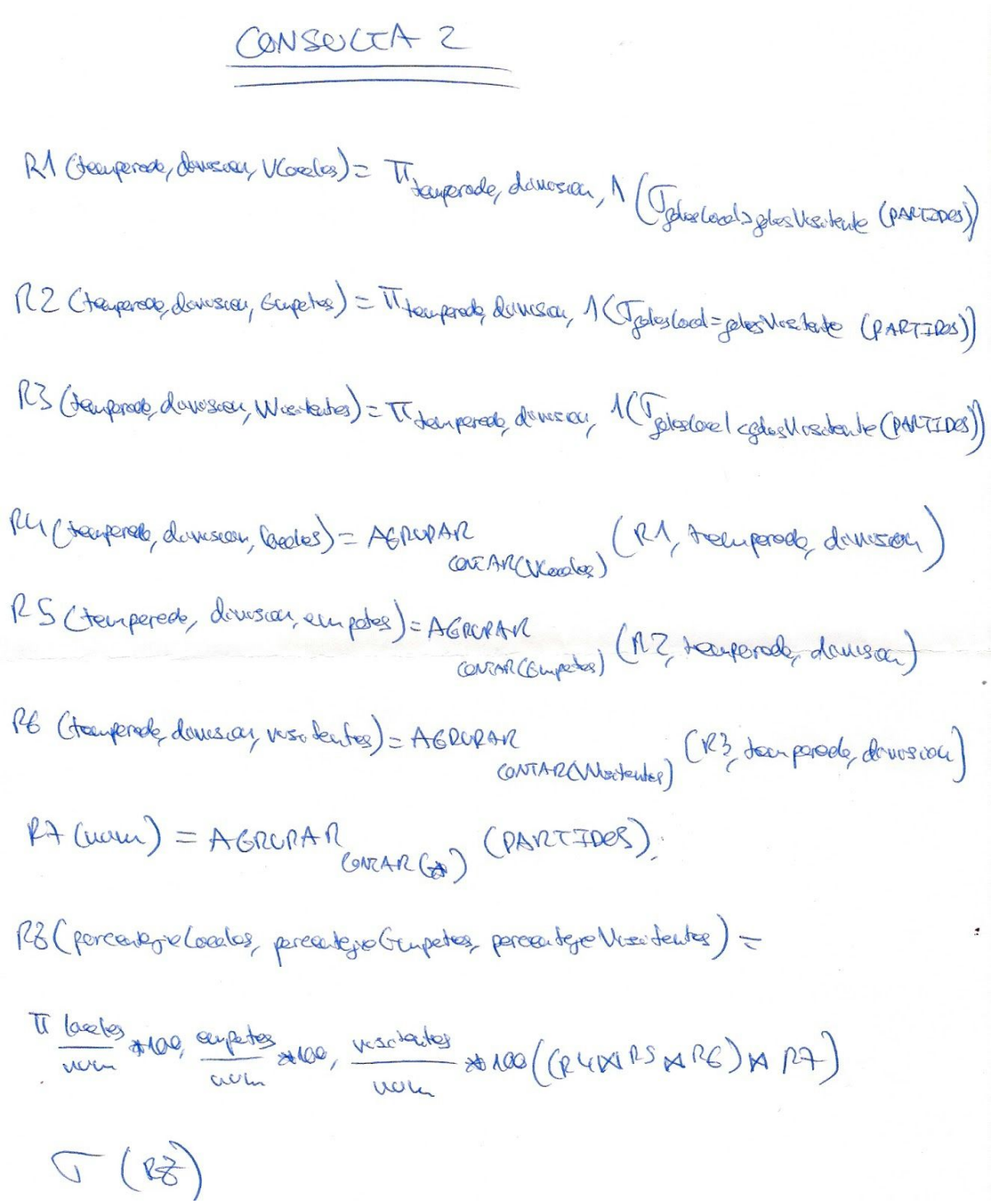
En la consulta 2 se nos pide que se obtengan el porcentaje de victorias locales, visitantes o empates para cada temporada de cada liga.

Para ello, se ha seleccionado la temporada, la división y, mediante tres cláusulas CASE, las victorias locales, las victorias visitantes y los empates respectivamente. Los resultados de las cláusulas case son simplemente un valor entero, 1 si ha

habido victoria local o visitante o empate en cada una de las cláusulas, y en cualquier otro caso, NULL. Todo esto seleccionado de la tabla PARTIDOS.

Una vez se tiene esta tabla, se selecciona de esta la temporada, la división y en las 3 columnas restantes, se hacen las correspondientes cuentas para sacar cada porcentaje, agrupando los resultados por temporada y división. Para hacer las cuentas mediante la función count() se han contado las victorias locales o visitantes o empates dependiendo de la columna en la que se esté, y se ha dividido esa cuenta entre el número total de campos de filas de la tabla (count(*)). Obviamente, este resultado será un número decimal con muchas cifras. Para optimizar la salida, se ha utilizado la función round() con el segundo parámetro puesto a 2, que lo que hace es redondear a dos decimales el resultado.

Árbol sintáctico de álgebra relacional



Respuesta Obtenida

Respuesta obtenida en el fichero respuestasConsultas.

Consulta 3

Se ha creado una vista en la cual se guardan los subcampeones de liga. Para crear esta vista se ha copiado el código de la vista de lo campeones de liga que se mencionó en la consulta 1, y en vez de seleccionar el puesto=1 en la cláusula where, se ha cambiado por puesto=2.

Se ha creado otra vista la cual es la misma que la de los campeones de liga mencionados en la consulta 1.

Ahora se obtienen los equipos a los que el campeón ha ganado haciendo un join de la vista de campeones y de la tabla de partidos con las condiciones de que las tuplas coincidan en temporada, en división y en nombre. Se selecciona la temporada, la división y el nombre del equipo visitante si ha perdido contra el equipo local.

El mismo proceso se ha seguido pero para la vista de subcampeones.

Ahora una vez con todo esto, se seleccionan los equipos a los que el campeón ha ganado pero el subcampeón no mediante la cláusula minus. Se obtienen las fechas de esta correspondiente selección y se obtienen las fechas diferentes a estas de la tabla temporada.

Árbol sintáctico de álgebra relacional

CONSULTA 3

$R1 = \Pi_{temporada, division, jornada, nombreVisitante, 3}(\sigma_{golesLocal < golesVisitante}(PARTIDOS))$

$R2 = \Pi_{temporada, division, jornada, nombreLocal, 1}(\sigma_{golesLocal = golesVisitante}(PARTIDOS))$

$R3 = \Pi_{temporada, division, jornada, nombreVisitante, 1}(\sigma_{golesLocal = golesVisitante}(PARTIDOS))$

$R4 = \Pi_{temporada, division, jornada, nombreLocal, 3}(\sigma_{golesLocal > golesVisitante}(PARTIDOS))$

$R5 = (temporada, division, jornada, nombre, puntos) = R1 \cup R2 \cup R3 \cup R4$

$R6 = AGRUPAR_{SUMA(puntos)}(R5, temporada, division, jornada, nombre)$

$R7 = AGRUPAR_{MAX(jornada)}(\sigma(JORNADA), idTemporada, idDivision)$

$R8 = R6 \bowtie R7$

$temporada = idTemporada \wedge division = idDivision \wedge jornada <= numJornadas$

$R9(temporada, division, nombre, final\_de\_liga) = AGRUPAR_{SUMA(puntos)}(R8, temporada, division, nombre)$

R10 = Se le aplicaría la función row_number() y over() para obtener una columna con los puestos de la clasificación según los puntos. Particionado por temporada y por división.

$$campeonatos = AGRUPAR\left(\sigma_{puesto=1}(RM), nombre\right)$$

$$subcampeonatos = AGRUPAR\left(\sigma_{puesto=2}(RM), nombre\right)$$

$$equipos\_camp(temporada, division, equipos) =$$

$$\pi_{E.temporada,\ campeonatos.division,\ PARTIDOS.nombreVisitante}\left(\sigma_{campeonatos}\left(campeonatos \bowtie PARTIDOS\right)\right)$$

$$campeonatos.nombre = PARTIDOS.nombreLocal \wedge campeonatos.temporada = PARTIDOS.temporada \wedge campeonatos.division = PARTIDOS.division \wedge PARTIDOS.golesLocal >= PARTIDOS.golesVisitante$$

$$equipos\_subc(temporada, division, equipo) =$$

$$\pi_{subcampeonatos.temporada,\ subcampeonatos.division,\ PARTIDOS.nombreVisitante}\left(\sigma\left(subcampeonatos \bowtie PARTIDOS\right)\right)$$

$$subcampeonatos.nombre = PARTIDOS.nombreLocal \wedge subcampeonatos.temporada = PARTIDOS.temporada \wedge subcampeonatos.division = PARTIDOS.division \wedge PARTIDOS.golesLocal <= PARTIDOS.golesVisitante$$

$R11 = \Pi_{temporada, division}(\sigma(equipos\_comp - equipos\_sob))$

$R12 = \Pi_{fecha, division}(\sigma(TEMPORADA))$

$= \sigma(R12 - R11)$

Respuesta Obtenida

Respuesta obtenida en el fichero respuestasConsultas.

Consulta no trivial 1

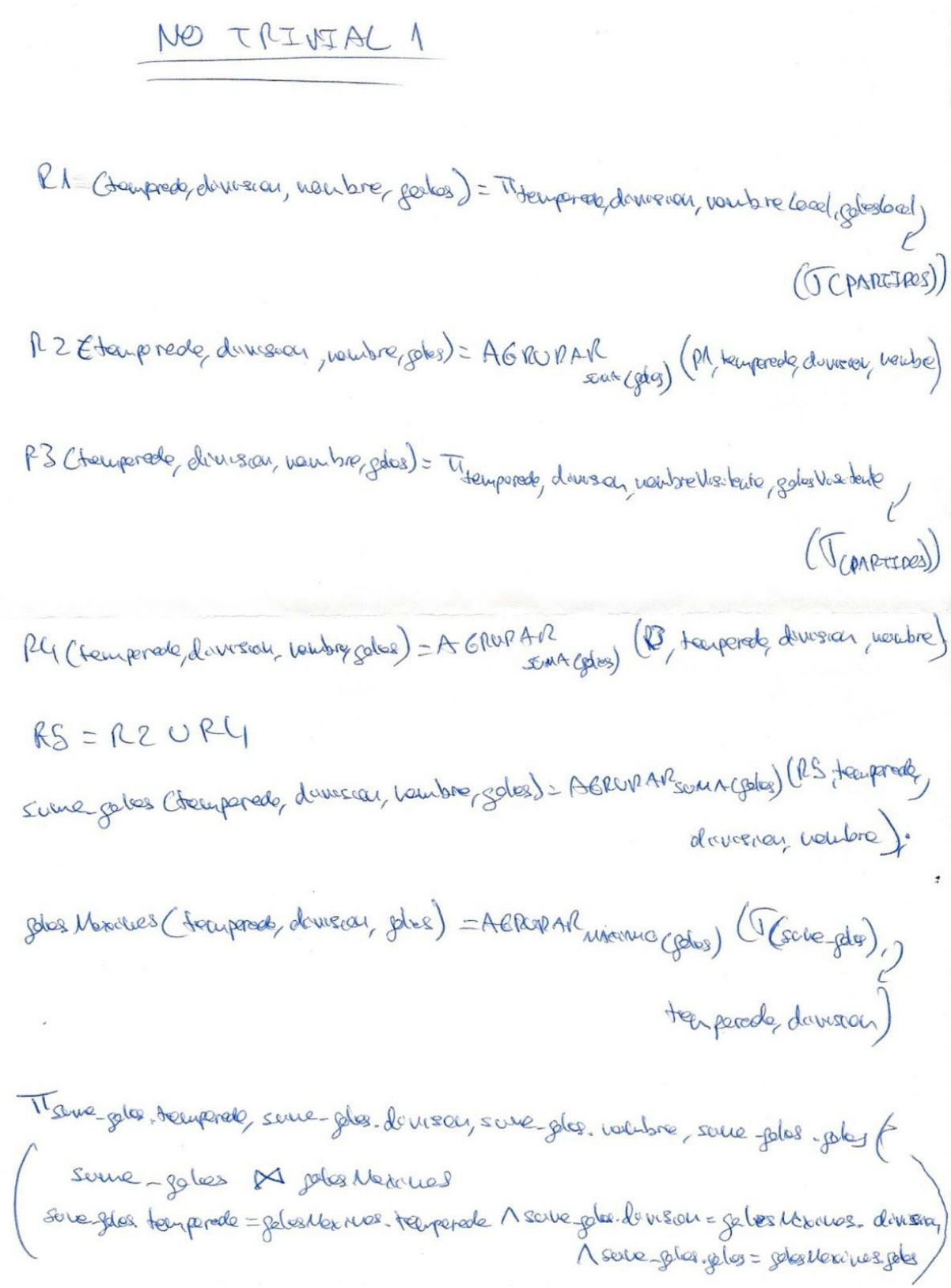
Para la primera consulta no trivial, se ha optado por obtener el máximo equipo goleador en goles totales de cada temporada de cada división.

Se ha empezado por obtener la suma de los goles para los equipos locales unido a la suma de los goles para los equipos visitantes. Una vez se tiene esto, se tiene la suma de goles para un equipo cuando ha estado de local y la suma para cuando ha estado de visitante. Como interesa tener la suma de goles totales se hizo una consulta sobre la tabla que se ha mencionado anteriormente seleccionando la temporada, division, nombre y la suma de goles agrupando por temporada, división y nombre. Se creó una vista con la tabla resultante.

Por otro lado, se creó otra vista seleccionando la temporada, la división y el número de goles máximo de las dos mencionadas anteriormente.

Para finalizar con estas dos vistas, se han seleccionado los elementos comunes en temporada, en división y en número de goles de las dos vistas creadas.

Árbol sintáctico de álgebra relacional



Respuesta Obtenida

Respuesta obtenida en el fichero respuestasConsultas.

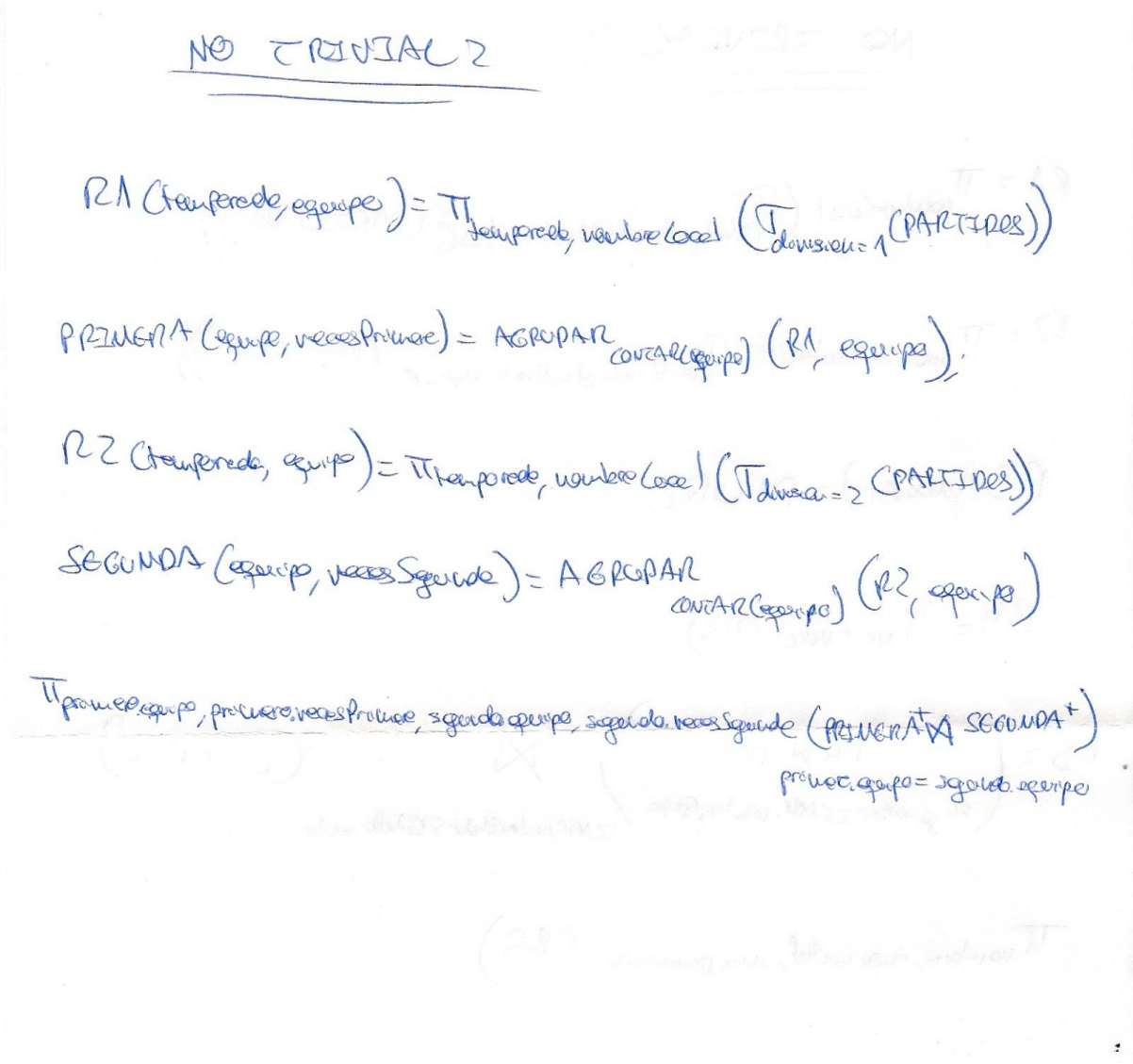
Consulta no trivial 2

Para hacer la segunda consulta no trivial, se ha optado por calcular el número de veces que un equipo ha estado en primera y en segunda división.

Se han creado dos vistas. En la primera se han seleccionado las temporadas y los nombres de los equipos que han estado en primera división. Una vez obtenida esta nueva tabla, se ha contado el número de veces que aparece cada equipo. En la segunda vista el procedimiento ha sido el mismo, salvo para los equipos que son de segunda división.

Una vez con las dos vistas creadas se ha hecho un full join para los equipos que coinciden en nombre de las dos tablas. Hay que decir que se intentó que el nombre del equipo apareciera una sola vez, pero como hay equipos que solo han estado en primera o en segunda, no se podía seleccionar solo el nombre de una vista.

Árbol sintáctico de álgebra relacional



Respuesta Obtenida

Respuesta obtenida en el fichero respuestasConsultas.

Consulta no trivial 3

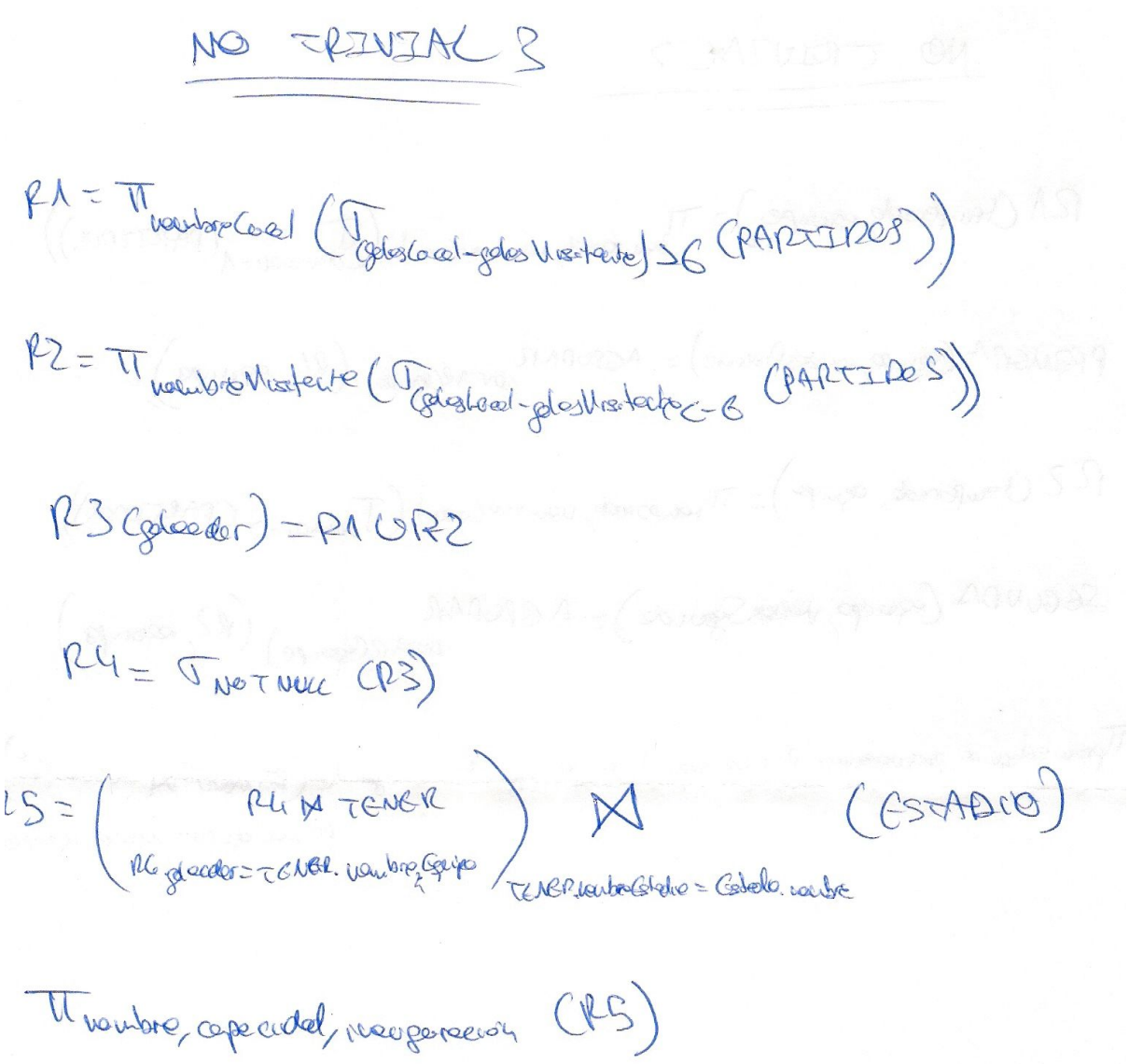
Para la última consulta no trivial, se ha optado por obtener los estadios, la capacidad y el año de inauguración de estos de los equipos que han ganado por una diferencia mayor o igual a 6 goles.

Para ello, mediante una cláusula CASE se ha obtenido el nombre del equipo local o el del visitante de la tabla PARTIDOS, dependiendo de quien haya ganado por una

diferencia mayor o igual a 6 goles. Como en la cláusula no se ha indicado que hacer para los demás casos, sql los pondrá a NULL. Por ello con esta tabla con algunos campos nulos se ha hecho una nueva consulta sobre esta seleccionando los campos no nulos. A continuación se ha creado una vista con la tabla resultante.

Ahora para sacar los estadios, se han obtenido los elementos comunes de la vista creada anteriormente y de la tabla TENER coincidentes en nombre. Acto seguido, se han seleccionado los elementos comunes en nombre de estadio de la tabla TENER y de la tabla ESTADIO. Por último, de la tabla ESTADIO se ha obtenido el nombre, la capacidad y la inauguración.

Árbol sintáctico de álgebra relacional



Respuesta Obtenida

Respuesta obtenida en el fichero respuestasConsultas.

DATOS SOBRE REUNIONES, DIVISIÓN DE TRABAJO, COORDINACIÓN DEL GRUPO

Población de la base: 26h
Consultas: 40h

PARTE 3
ESTADÍSTICAS PREDISEÑO

1)

```
Statistics
----------------------------------------------------------
          258  recursive calls
            0  db block gets
         1162  consistent gets
            0  physical reads
            0  redo size
         1087  bytes sent via SQL*Net to client
          406  bytes received via SQL*Net from client
            4  SQL*Net roundtrips to/from client
            5  sorts (memory)
            0  sorts (disk)
           32  rows processed
```

2)

```
Statistics
----------------------------------------------------------
           28  recursive calls
            0  db block gets
          270  consistent gets
            0  physical reads
            0  redo size
         3311  bytes sent via SQL*Net to client
          439  bytes received via SQL*Net from client
            7  SQL*Net roundtrips to/from client
            1  sorts (memory)
            0  sorts (disk)
           84  rows processed
```

3)

Statistics

59	recursive calls
0	db block gets
1690	consistent gets
0	physical reads
0	redo size
555	bytes sent via SQL*Net to client
384	bytes received via SQL*Net from client
2	SQL*Net roundtrips to/from client
12	sorts (memory)
0	sorts (disk)
4	rows processed

4)

Statistics

34	recursive calls
0	db block gets
1078	consistent gets
0	physical reads
0	redo size
3438	bytes sent via SQL*Net to client
439	bytes received via SQL*Net from client
7	SQL*Net roundtrips to/from client
1	sorts (memory)
0	sorts (disk)
90	rows processed

5)

```
Statistics
----------------------------------------------------------
          28  recursive calls
           0  db block gets
        1494  consistent gets
           0  physical reads
       14308  redo size
        3502  bytes sent via SQL*Net to client
         450  bytes received via SQL*Net from client
           8  SQL*Net roundtrips to/from client
           0  sorts (memory)
           0  sorts (disk)
         102  rows processed
```

6)

```
Statistics
----------------------------------------------------------
         602  recursive calls
           0  db block gets
         674  consistent gets
           0  physical reads
           0  redo size
         396  bytes sent via SQL*Net to client
         373  bytes received via SQL*Net from client
           1  SQL*Net roundtrips to/from client
          15  sorts (memory)
           0  sorts (disk)
           0  rows processed
```

DISEÑO FÍSICO

Las claves utilizadas aportan información por sí mismas en todos los casos. Hubiera sido más eficiente usar claves más cortas en equipos y estadios, pero no tenemos un lugar de donde extraer esas claves y cambiarlas a mano supone mucho trabajo.

No se han realizado particiones horizontales puesto que no hay atributos con pocos valores y que sean utilizados en las consultas.

No hemos considerado apropiado utilizar particiones verticales, puesto que ninguna entidad tiene demasiados atributos, y estos no se usan por separado.

Tampoco se ha realizado precalculo de joins.

Se ha creado una vista materializada que contiene la puntuación de cada equipo en cada temporada y su puesto.

```
create materialized view puntuaciones (temporada,division,nombre,puntos,puest)
REFRESH COMPLETE
START WITH TO_DATE('02-09-2015 19:00:00','DD-MM-YYYY HH24:MI:SS')
NEXT SYSDATE + 1
as
SELECT temporada, division, nombre, final_de_liga, row_number() over(PARTITION BY temporada, division
ORDER BY temporada, division, final_de_liga DESC) AS puesto
FROM (
        --Puntuaciones de todos los equipos a final de liga
        SELECT P.temporada, P.division, P.nombre, sum(P.puntuacion) AS final_de_liga
        FROM (
        --Puntuaciones de cada equipo para cada jornada de cada division de cada temporada
                SELECT temporada, division, jornada, nombre, sum(puntos) AS puntuacion
```

```
        FROM (
            --Tabla de asignación de puntos para los locales
            SELECT temporada, division, jornada, nombreLocal AS nombre,
                CASE
                    WHEN golesLocal>golesVisitante THEN 3
                    WHEN golesLocal<golesVisitante THEN 0
                    ELSE 1
                END AS puntos
            FROM PARTIDOS
            UNION
            --Tabla de asignación de puntos para los visitantes
            SELECT temporada, division, jornada, nombreVisitante,
                CASE
                    WHEN golesLocal>golesVisitante THEN 0
                    WHEN golesLocal<golesVisitante THEN 3
                    ELSE 1
                END AS puntos
            FROM PARTIDOS
            )
        GROUP BY temporada, division, jornada, nombre
        ORDER BY temporada, division, jornada
        ) P, (
            --Número de jornadas de cada division de cada temporada
            SELECT idTemporada, idDivision, max(jornada) AS numJornadas
            FROM JORNADA
            GROUP BY idTemporada, idDivision
            ORDER BY idTemporada, idDivision) J
    WHERE P.temporada=J.idTemporada AND P.division=J.idDivision AND P.jornada<=J.numJornadas
    GROUP BY P.temporada, P.division, P.nombre
    ORDER BY P.temporada, P.division, final_de_liga
);
```

Se han empleado índices en atributos utilizados en condiciones de selección, así como en la vista materializada.

```
CREATE INDEX capacidad_idx ON ESTADIO(capacidad);
CREATE INDEX inauguracion_idx ON ESTADIO(inauguración);
CREATE INDEX puesto_idx ON puntuaciones(puesto);
```

ESTADÍSTICAS POST DISEÑO

1
Statistics
---
34 recursive calls
0 db block gets
608 consistent gets
1 physical reads
0 redo size
1087 bytes sent via SQL*Net to client
406 bytes received via SQL*Net from client
4 SQL*Net roundtrips to/from client
2 sorts (memory)
0 sorts (disk)
32 rows processed

2)

Statistics
---
4 recursive calls
0 db block gets
268 consistent gets
0 physical reads
0 redo size
3311 bytes sent via SQL*Net to client

```
       439  bytes received via SQL*Net from client
         7  SQL*Net roundtrips to/from client
         1  sorts (memory)
         0  sorts (disk)
        84  rows processed
```

3)

```
Statistics
----------------------------------------------------------
        36  recursive calls
         0  db block gets
       622  consistent gets
         0  physical reads
         0  redo size
       555  bytes sent via SQL*Net to client
       384  bytes received via SQL*Net from client
         2  SQL*Net roundtrips to/from client
         8  sorts (memory)
         0  sorts (disk)
         4  rows processed
```

4)

```
Statistics
----------------------------------------------------------
        34  recursive calls
         0  db block gets
      1078  consistent gets
         0  physical reads
         0  redo size
      3438  bytes sent via SQL*Net to client
```

```
          439  bytes received via SQL*Net from client
            7  SQL*Net roundtrips to/from client
            1  sorts (memory)
            0  sorts (disk)
           90  rows processed
```

5)

```
Statistics
----------------------------------------------------------
           28  recursive calls
            0  db block gets
         1303  consistent gets
            0  physical reads
            0  redo size
         3502  bytes sent via SQL*Net to client
          450  bytes received via SQL*Net from client
            8  SQL*Net roundtrips to/from client
            0  sorts (memory)
            0  sorts (disk)
          102  rows processed
```

6)

```
Statistics
----------------------------------------------------------
          109  recursive calls
            0  db block gets
          578  consistent gets
            0  physical reads
            0  redo size
```

396 bytes sent via SQL*Net to client
373 bytes received via SQL*Net from client
  1 SQL*Net roundtrips to/from client
  3 sorts (memory)
  0 sorts (disk)
  0 rows processed

TRIGGERS DESARROLLADOS

TRIGGER 1

Este primer trigger consiste en que no se puedan insertar partidos con un año (temporada), anterior a su año de fundación. El trigger se dispara antes de insertar la tupla en la tabla partidos. Se dispara a nivel de fila.

Se declaran una serie de variables para guardar el año de fundación del equipo local y del visitante. Se obtienen los dos años de la temporada del partido que se quiere insertar. Se guardan en las variables que se han citado anteriormente el año de fundación de cada equipo mediante una consulta sql. Ahora si algún año de la nueva temporada del nuevo partido es menor que el año de fundación de alguno de los equipos, se produce un error por pantalla informando de dicha restricción mediante la función RAISE_APPLICATION_ERROR.

TRAZA DE FUNCIONAMIENTO SQL:

Trigger created.

```
INSERT INTO PARTIDOS VALUES('1902-1903','1',1,'Burgos','Real_Sociedad',1,0)
            *
ERROR at line 1:
```

ORA-20002: No se puede insertar un partido con el anio de la temporada anterior al anio de fundación del equipo
ORA-06512: at "RUEDINES.FUNDACION", line 7
ORA-04088: error during execution of trigger 'RUEDINES.FUNDACION'

TRIGGER 2

Este trigger consiste en que no se puede insertar un partido en el que el mismo equipo juegue como local y como visitante. Se dispara antes de una inserción en la tabla partidos a nivel de cada fila cuando el equipo local sea el mismo que el visitante. Se imprime un error por pantalla informando de dicha restricción mediante la función RAISE_APPLICATION_ERROR.

TRAZA DE FUNCIONAMIENTO SQL:

Trigger created.

INSERT INTO PARTIDOS VALUES('1992-1993','1',30,'Zaragoza','Zaragoza',1,0)
*
ERROR at line 1:
ORA-20001: El equipo local y el equipo visitante no pueden tener el mismo nombre
ORA-06512: at "RUEDINES.IGUALES", line 2
ORA-04088: error during execution of trigger 'RUEDINES.IGUALES'

TRIGGER 3

El funcionamiento de este último trigger consiste en que no se pueden insertar temporadas en la BBDD posteriores al año actual. Se dispara antes de la inserción de cada temporada a nivel de fila.

Para su realización se ha obtenido mediante la función substr el primer año de la temporada y se ha comprobado si este es mayor que el año actual. Obtenido por la función current_date y la función substr. En caso de que se cumpla la condición, se produce un mensaje de error comunicando dicha restricción mediante la función RAISE_APPLICATION_ERROR.

TRAZA DE FUNCIONAMIENTO SQL:

Trigger created.

```
INSERT INTO TEMPORADA (fecha, division) VALUES ('2029-2030','1')
            *
ERROR at line 1:
ORA-20008: No se pueden insertar temporadas con un anio posterior al actual
ORA-06512: at "RUEDINES.NOMASHOY", line 3
ORA-04088: error during execution of trigger 'RUEDINES.NOMASHOY'


INSERT INTO TEMPORADA (fecha, division) VALUES ('2029-2030','2')
            *
ERROR at line 1:
ORA-20008: No se pueden insertar temporadas con un anio posterior al actual
ORA-06512: at "RUEDINES.NOMASHOY", line 3
ORA-04088: error during execution of trigger 'RUEDINES.NOMASHOY'
```

DATOS SOBRE REUNIONES, DIVISIÓN DE TRABAJO, COORDINACIÓN DEL GRUPO

Horas dedicadas: 40 h.