Memoria Práctica 1
Bases de Datos

Curso 24/25

Carlos Alejos Fumanal - 872342@unizar.es
Mario Caudevilla Ruiz - 870421@unizar.es
Rodrigo Dávila Duarte - 872715@unizar.es

Grupo T15

Ingeniería Informática

Escuela de Ingeniería y Arquitectura

Universidad de Zaragoza

14 de Marzo, 2025

Contenidos

1 | Creación de la base de datos

1.1 Esquema E/R

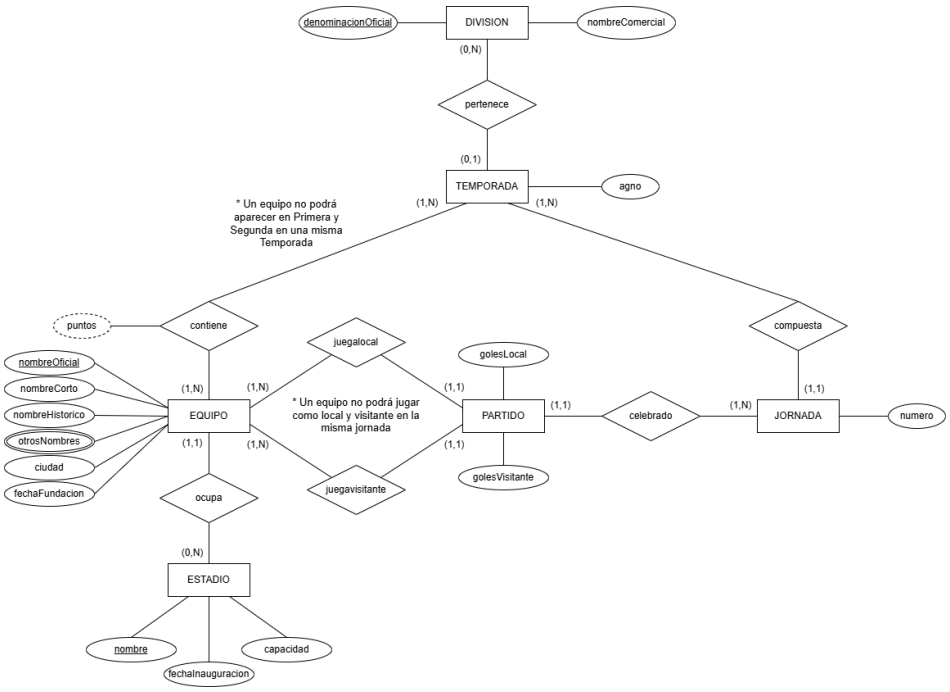

Figura 1: Esquema Entidad-Relación

Se han aplicado las siguientes restricciones para evitar conflictos que se podrían dar con el esquema E/R creado:

- Con el fin de evitar que un equipo participe en Primera y Segunda en una misma temporada (Se especifica de esta forma ya que, por ejemplo: un equipo puede aparecer en Segunda y en Promoci en la misma Temporada).
  Asumimos que un equipo no podrá aparecer en Primera y Segunda en una misma temporada.
- Con el fin de evitar que un equipo aparezca dos veces en el mismo partido (por ejemplo: un "Real Zaragoza vs Real Zaragoza").
  Asumimos que un equipo no podrá jugar como local y visitante en la misma jornada.
- Se asume que los goles de un equipo local y visitante son mayores o iguales que 0.

El esquema permite que se den ciertas circunstancias que hemos asumido como posibles:

- Una división puede no tener ninguna temporada asociada.
- Una temporada puede no tener ninguna división asociada.
- Una temporada puede tener como máximo una división asociada (debido a que temporada participa en la relación con una multiplicidad (0,1)). Con esta decisión vamos a tener cada temporada replicada tantas veces como divisiones haya en esa temporada.
- Una temporada tiene que tener como mínimo una jornada asociada.
- Una jornada tiene que tener como mínimo un partido asociado.
- Un estadio puede no estar asociado a ningún equipo.

1.2 Esquema Relacional

1.2.1 Esquema relacional no normalizado

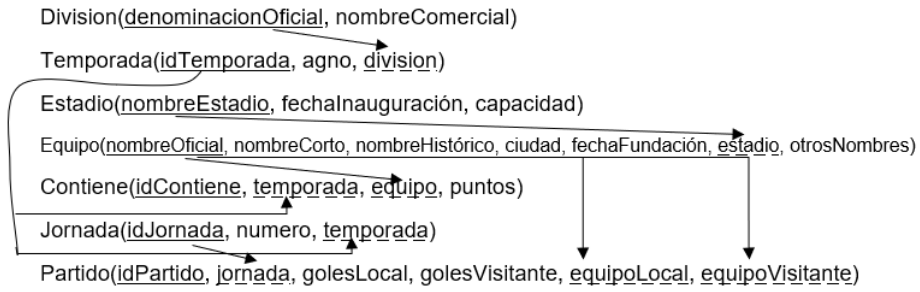

Figura 2: Esquema Relacional sin normalizar

1.2.2 Esquema relacional normalizado

Primero de todo, se ha comprobado si estaba en primera forma normal (1FN) y para ello se ha verificado que todos los atributos tuviesen sólo un valor posible. Como se puede ver, la tupla Equipo contiene un atributo multivaluado llamado otrosNombres por lo que se ha eliminado el atributo y se ha creado otra tupla llamada OtrosNombres que lo contiene.
Respecto a la segunda forma normal (2FN), se cumple que todas las tuplas están en 1FN y no tienen dependencias parciales ya que no hay claves compuestas y este tipo de dependencias sólo tiene lugar cuando la clave es compuesta (es decir, si hay atributos que dependen solo de una parte de la clave y no de toda).
En cuanto a la tercera forma normal (3FN), se cumple que todas las tuplas están en 2FN y no tienen dependencias transitivas (es decir, si un atributo no clave depende de otro atributo no clave) ya que en cada tupla todos los atributos dependen directamente de la clave primaria única.
Por último, se ha comprobado que el esquema esté en FNBC ya que no existen dependencias funcionales que no partan de la clave, tal y como se ha comprobado anteriormente en la 3FN.

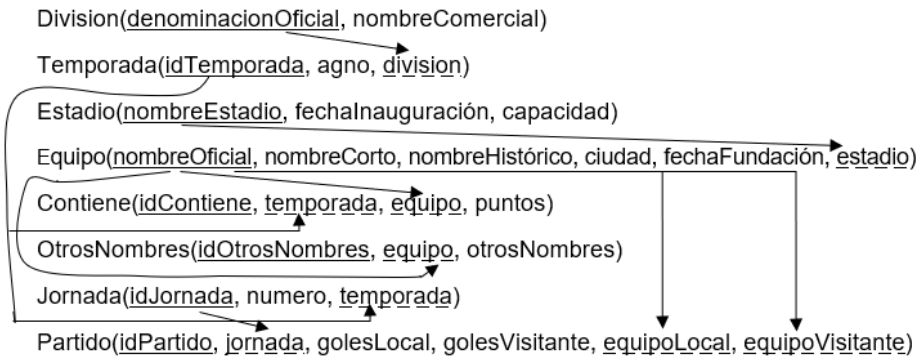

Figura 3: Esquema Relacional normalizado

1.3 Sentencias SQL para la creación de tablas

```sql
-- Tabla para las divisiones
CREATE TABLE DIVISION (
    denominacionOficial  VARCHAR(20) PRIMARY KEY,
    nombreComercial      VARCHAR(20)
);

-- Tabla para las temporadas
CREATE TABLE TEMPORADA (
    idTemporada NUMBER(20) PRIMARY KEY,
    agno        NUMBER(5) NOT NULL,
    division    VARCHAR(20) NULL,
    FOREIGN KEY (division) REFERENCES DIVISION(denominacionOficial)
);
-- Secuencia para la tabla TEMPORADA
CREATE SEQUENCE secTemporada
    START WITH 1
    INCREMENT BY 1;
-- Trigger para la secuencia en la clave artificial idTemporada de la tabla TEMPORADA
CREATE OR REPLACE TRIGGER trg_temporada_id
BEFORE INSERT ON TEMPORADA
FOR EACH ROW
BEGIN
    :NEW.idTemporada := secTemporada.NEXTVAL;
END;
/

-- Tabla para los estadios
CREATE TABLE ESTADIO (
    nombreEstadio     VARCHAR(60) PRIMARY KEY,
    fechaInauguracion NUMBER(5),
    capacidad         NUMBER(9) NULL
);

-- Tabla para los equipos
CREATE TABLE EQUIPO (
    nombreOficial   VARCHAR(60) PRIMARY KEY,
    nombreCorto     VARCHAR(60) NULL,
    nombreHistorico VARCHAR(70),
    ciudad          VARCHAR(60),
    fechaFundacion  NUMBER(5),
    estadio         VARCHAR(60),
    FOREIGN KEY (estadio) REFERENCES ESTADIO(nombreEstadio)
);

-- Tabla para la relacion entre equipos y temporadas
CREATE TABLE contiene (
    idContiene NUMBER(20) PRIMARY KEY,
    temporada  NUMBER(20) NOT NULL,
    equipo     VARCHAR(60) NOT NULL,
    puntos     NUMBER(4) NOT NULL, -- Al poblar la base puntos seran 0, se calculan con una consulta
    FOREIGN KEY (temporada) REFERENCES TEMPORADA(idTemporada),
    FOREIGN KEY (equipo)    REFERENCES EQUIPO(nombreOficial)
    -- La restriccion de que un equipo no puede aparecer en 1 y 2 en el mismo agno se hace en un trigger
);
-- Secuencia para la tabla contiene
CREATE SEQUENCE secContiene
    START WITH 1
    INCREMENT BY 1;
-- Trigger para la secuencia en la clave artificial idContiene de la tabla contiene
CREATE OR REPLACE TRIGGER trg_contiene_id
BEFORE INSERT ON contiene
FOR EACH ROW
BEGIN
    :NEW.idContiene := secContiene.NEXTVAL;
END;
/
```

```
-- Tabla para los otros nombres de los equipos
CREATE TABLE otrosNombres (
    idOtrosNombres NUMBER(20) PRIMARY KEY,
    equipo         VARCHAR(60) NOT NULL,
    otrosNombres   VARCHAR(60),
    FOREIGN KEY (equipo) REFERENCES EQUIPO(nombreOficial)
);
-- Secuencia para la tabla otrosNombres
CREATE SEQUENCE secOtrosNombres
    START WITH 1
    INCREMENT BY 1;
-- Trigger para la secuencia en la clave artificial idOtrosNombres de la tabla otrosNombres
CREATE OR REPLACE TRIGGER trg_otrosNombres_id
BEFORE INSERT ON otrosNombres
FOR EACH ROW
BEGIN
    :NEW.idOtrosNombres := secOtrosNombres.NEXTVAL;
END;
/

-- Tabla para las jornadas
CREATE TABLE JORNADA (
    idJornada NUMBER(20) PRIMARY KEY,
    numero    NUMBER(4) NOT NULL,
    temporada NUMBER(20) NOT NULL,
    FOREIGN KEY (temporada) REFERENCES TEMPORADA(idTemporada)
);
-- Secuencia para la tabla JORNADA
CREATE SEQUENCE secJornada
    START WITH 1
    INCREMENT BY 1;
-- Trigger para la secuencia en la clave artificial idJornada de la tabla JORNADA
CREATE OR REPLACE TRIGGER trg_jornada_id
BEFORE INSERT ON JORNADA
FOR EACH ROW
BEGIN
    :NEW.idJornada := secJornada.NEXTVAL;
END;
/

-- Tabla para los partidos
CREATE TABLE PARTIDO (
    idPartido       NUMBER(20) PRIMARY KEY,
    jornada         NUMBER(20) NOT NULL,
    equipoLocal     VARCHAR(60) NOT NULL,
    equipoVisitante VARCHAR(60) NOT NULL,
    golesLocal      NUMBER(3) NOT NULL,
    golesVisitante  NUMBER(3) NOT NULL,
    FOREIGN KEY (jornada)        REFERENCES JORNADA(idjornada),
    FOREIGN KEY (equipoLocal)    REFERENCES EQUIPO(nombreOficial),
    FOREIGN KEY (equipoVisitante) REFERENCES EQUIPO(nombreOficial),
    CONSTRAINT chk_teams_not_equal CHECK (equipoLocal <> equipoVisitante),
    CONSTRAINT chk_goles_no_negativos CHECK (golesLocal >= 0 AND golesVisitante >= 0)
);
-- Secuencia para la tabla PARTIDOS
CREATE SEQUENCE secPartidos
    START WITH 1
    INCREMENT BY 1;
-- Trigger para la secuencia en la clave artificial idPartido de la tabla PARTIDOS
CREATE OR REPLACE TRIGGER trg_partidos_id
BEFORE INSERT ON PARTIDO
FOR EACH ROW
BEGIN
    :NEW.idPartido := secPartidos.NEXTVAL;
END;
/
```

Código 1: Creación de tablas en SQL

2 | Introducción de datos y ejecución de consultas

2.1 Pasos para poblar la base de datos

Para poblar la base de datos, se ha optado por usar archivos .csv, que luego han sido añadidos a la base de datos a través de archivos .ctl específicos para cada uno de ellos. Lo primero que se hizo fue quitar las tildes y cambiar las "ñ" de LigaHost.csv. Acto seguido se procedió con la extracción de la información correspondiente a cada tabla en su respectivo .csv. Es decir, cada .csv cuenta con una columna para cada atributo que tiene en la tabla, exceptuando los índices propios de cada tabla ya que se generan solos mediante secuencias. Los atributos están separados por ";".
Después de extraer todos los datos del archivo inicial LigaHost.csv y distribuirlos en los correspondientes archivos .csv, el siguiente paso consistió en crear los archivos .ctl (Control Files). Estos archivos son procesados por SQL*Loader, un software de Oracle que facilita la carga en masa de datos en la base de datos. Cada archivo .ctl establece la estructura del archivo .csv correspondiente y especifica la manera en que los datos deben ser introducidos en la base de datos. A continuación, se presenta un archivo de control (Temporada.ctl) utilizado para cargar la información de Temporada.csv:

```
load data
infile './Temporada.csv'
into table TEMPORADA
fields terminated by ';'
(agno, division)
```

Finalmente, tras la creación de los archivos .ctl, se puso en marcha SQL*Loader para introducir los datos en la base de datos Oracle mediante el comando: sqlldr aNIP@barret.danae04.unizar control=datosEquipo.ctl
Este comando se ejecutó para cada tabla de la BD, siguiendo un orden concreto para no crear tablas que guardan claves extranjeras antes que las tablas a las que hacen referencia.
Por último se creó una consulta para asignar a cada fila de la tabla CONTIENE los puntos que obtuvo cada equipo en cada temporada.

```
UPDATE CONTIENE c
SET c.puntos = (
    SELECT SUM(puntos)
    FROM (
        SELECT p.equipoLocal AS equipo, t.idTemporada AS temporada, 3 AS puntos
        FROM PARTIDO p
        JOIN JORNADA j ON p.jornada = j.idJornada
        JOIN TEMPORADA t ON j.temporada = t.idTemporada
        WHERE p.golesLocal > p.golesVisitante
        UNION ALL
        SELECT p.equipoLocal AS equipo, t.idTemporada AS temporada, 1 AS puntos
        FROM PARTIDO p
        JOIN JORNADA j ON p.jornada = j.idJornada
        JOIN TEMPORADA t ON j.temporada = t.idTemporada
        WHERE p.golesLocal = p.golesVisitante
        UNION ALL
        SELECT p.equipoVisitante AS equipo, t.idTemporada AS temporada, 1 AS puntos
        FROM PARTIDO p
        JOIN JORNADA j ON p.jornada = j.idJornada
        JOIN TEMPORADA t ON j.temporada = t.idTemporada
        WHERE p.golesVisitante = p.golesLocal
        UNION ALL
        SELECT p.equipoVisitante AS equipo, t.idTemporada AS temporada, 3 AS puntos
        FROM PARTIDO p
        JOIN JORNADA j ON p.jornada = j.idJornada
        JOIN TEMPORADA t ON j.temporada = t.idTemporada
        WHERE p.golesVisitante > p.golesLocal
        UNION ALL
        SELECT p.equipoLocal AS equipo, t.idTemporada AS temporada, 0 AS puntos
```

```
        FROM PARTIDO p
        JOIN JORNADA j ON p.jornada = j.idJornada
        JOIN TEMPORADA t ON j.temporada = t.idTemporada
        WHERE p.golesLocal < p.golesVisitante
        UNION ALL
        SELECT p.equipoVisitante AS equipo, t.idTemporada AS temporada, 0 AS puntos
        FROM PARTIDO p
        JOIN JORNADA j ON p.jornada = j.idJornada
        JOIN TEMPORADA t ON j.temporada = t.idTemporada
        WHERE p.golesVisitante < p.golesLocal
    ) puntos_totales
    WHERE puntos_totales.equipo = c.equipo
    AND puntos_totales.temporada = c.temporada
)
WHERE EXISTS (
    SELECT 1
    FROM PARTIDO p
    JOIN JORNADA j ON p.jornada = j.idJornada
    JOIN TEMPORADA t ON j.temporada = t.idTemporada
    WHERE (p.equipoLocal = c.equipo OR p.equipoVisitante = c.equipo)
    AND t.idTemporada = c.temporada
);
```

Código 2: Población del atributo puntos de CONTIENE

Para aprovechar mejor el espacio se pone aquí el árbol sintáctico de la consulta 2, en ella aparecerá un enlace para que al pulsar se venga aquí.

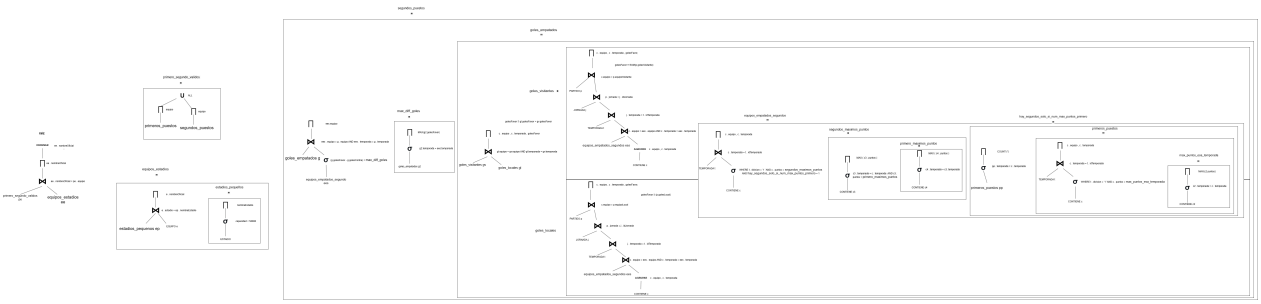

Figura 4: Árbol sintáctico de la consulta 2

2.2 Consultas SQL

2.2.1 Consulta 1

```sql
WITH victorias AS (
    SELECT j.temporada, p.equipoLocal AS zaragoza, p.equipoVisitante AS rival
    FROM PARTIDO p
    JOIN JORNADA j ON p.jornada = j.idJornada
    WHERE p.equipoLocal = 'Real Zaragoza' AND p.golesLocal > p.golesVisitante
    UNION ALL
    SELECT j.temporada, p.equipoVisitante AS zaragoza, p.equipoLocal AS rival
    FROM PARTIDO p
    JOIN JORNADA j ON p.jornada = j.idJornada
    WHERE p.equipoVisitante = 'Real Zaragoza' AND p.golesVisitante > p.golesLocal
),
conteo_victorias AS (
    SELECT v.temporada, v.rival, COUNT(*) AS num_victorias
    FROM victorias v
    GROUP BY v.temporada, v.rival
    HAVING COUNT(*) = 2
),
temporadas_validas AS (
    SELECT c.temporada
    FROM conteo_victorias c
    GROUP BY c.temporada
    HAVING COUNT(*) >= 4
),
goles_locales AS (
    SELECT t.division, t.agno, SUM(p.golesLocal) AS goles_zaragoza
    FROM TEMPORADA t
    JOIN JORNADA j ON t.idTemporada = j.temporada
    JOIN PARTIDO p ON j.idJornada = p.jornada
    WHERE p.equipoLocal = 'Real Zaragoza'
      AND t.idTemporada IN (SELECT temporada FROM temporadas_validas)
    GROUP BY t.division, t.agno
),
goles_visitantes AS (
    SELECT t.division, t.agno, SUM(p.golesVisitante) AS goles_zaragoza
    FROM TEMPORADA t
    JOIN JORNADA j ON t.idTemporada = j.temporada
    JOIN PARTIDO p ON j.idJornada = p.jornada
    WHERE p.equipoVisitante = 'Real Zaragoza'
      AND t.idTemporada IN (SELECT temporada FROM temporadas_validas)
    GROUP BY t.division, t.agno
)
SELECT gl.division, gl.agno, (gl.goles_zaragoza + gv.goles_zaragoza) AS goles_zaragoza
FROM goles_locales gl
JOIN goles_visitantes gv ON gl.division = gv.division AND gl.agno = gv.agno
ORDER BY gl.agno;
```

Código 3: Código de la Consulta 1

```
DIVISION                  AGNO GOLES_ZARAGOZA
-------------------- ---------- --------------
1                          1982             59
1                          1985             51
1                          1998             57
2                          2002             54
2                          2008             79
```

Código 4: Resultados de la Consulta 1

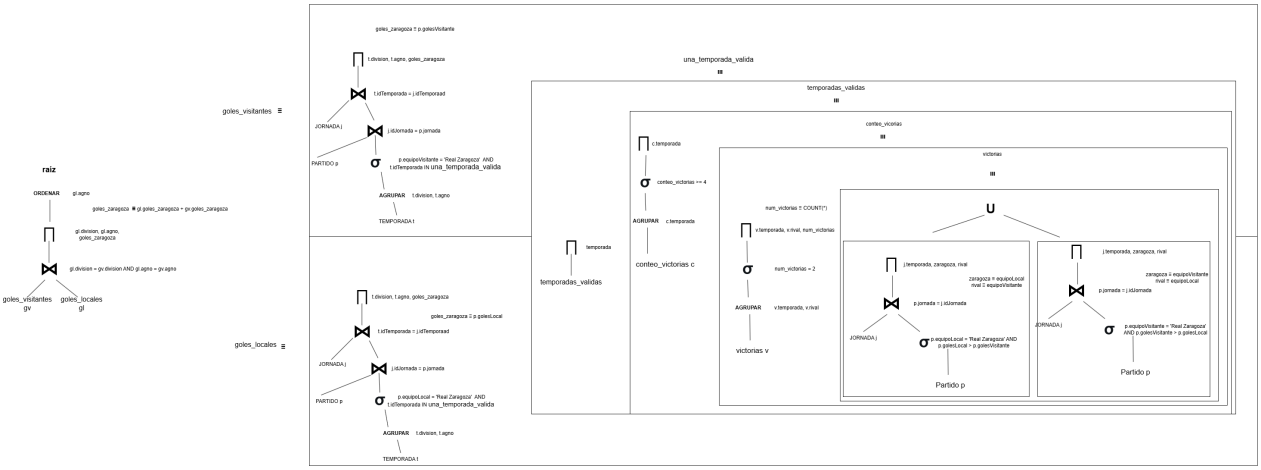

Figura 5: Árbol sintáctico de la consulta 1

2.2.2 Consulta 2

En esta consulta hemos identificado un problema principal, ¿qué pasaría si varios equipos estuviesen empatados a puntos en la segunda posición?. Este caso se da en la temporada 1999/2000, donde el Barça y el Valencia empataron a puntos por la segunda plaza. Para resolver este conflicto hemos pensado en resolver los empates como normalmente se hace en el fútbol, por diferencia de goles.

```sql
WITH estadios_pequenos AS (
    -- Equipos que juegan en estadios con capacidad menor a 50,000
    SELECT nombreEstadio
    FROM ESTADIO
    WHERE capacidad < 50000
),
equipos_estadios AS (
    -- Equipos que juegan en estadios pequegnos
    SELECT e.nombreOficial
    FROM EQUIPO e
    JOIN estadios_pequenos ep ON e.estadio = ep.nombreEstadio
),
primeros_puestos AS (
    -- Equipos que quedaron en primer lugar (incluyendo empates)
    SELECT c.equipo, c.temporada
    FROM CONTIENE c
    JOIN TEMPORADA t ON c.temporada = t.idTemporada
    WHERE t.division = '1'
    AND c.puntos = (
        SELECT MAX(c2.puntos)
        FROM CONTIENE c2
        WHERE c2.temporada = c.temporada
    )
),
equipos_empatados_segundo AS (
    -- Equipos que estan empatados en el segundo puesto (sin calcular aun los goles)
    SELECT c.equipo, c.temporada
    FROM CONTIENE c
    JOIN TEMPORADA t ON c.temporada = t.idTemporada
    WHERE t.division = '1'
    AND c.puntos = (
        -- Segundo maximo puntaje en la temporada, excluyendo el primero
        SELECT MAX(c3.puntos)
        FROM CONTIENE c3
        WHERE c3.temporada = c.temporada
        AND c3.puntos < (
            SELECT MAX(c4.puntos)
```

```
            FROM CONTIENE c4
            WHERE c4.temporada = c3.temporada
        )
    )
    -- Solo incluir segundos puestos si hay exactamente 1 equipo en primer lugar en esa temporada
    AND (SELECT COUNT(*) FROM primeros_puestos pp WHERE pp.temporada = c.temporada) = 1
),
goles_locales AS (
    -- Calcular goles SOLO de los equipos que estan en el segundo puesto empatados en la temporada en
        cuestion
    SELECT c.equipo, c.temporada, SUM(p.golesLocal) AS golesFavor
    FROM CONTIENE c
    JOIN PARTIDO p ON c.equipo = p.equipoLocal
    JOIN JORNADA j ON p.jornada = j.idJornada
    JOIN TEMPORADA t ON j.temporada = t.idTemporada
    JOIN equipos_empatados_segundo ees ON c.equipo = ees.equipo AND c.temporada = ees.temporada
    GROUP BY c.equipo, c.temporada
),
goles_visitantes AS (
    SELECT c.equipo, c.temporada, SUM(p.golesVisitante) AS golesFavor
    FROM CONTIENE c
    JOIN PARTIDO p ON c.equipo = p.equipoVisitante
    JOIN JORNADA j ON p.jornada = j.idJornada
    JOIN TEMPORADA t ON j.temporada = t.idTemporada
    JOIN equipos_empatados_segundo ees ON c.equipo = ees.equipo AND c.temporada = ees.temporada
    GROUP BY c.equipo, c.temporada
),
goles_empatados AS (
    SELECT gl.equipo, gl.temporada, (gl.golesFavor + gv.golesFavor) AS golesFavor
    FROM goles_locales gl
    JOIN goles_visitantes gv ON gl.equipo = gv.equipo AND gl.temporada = gv.temporada
),
segundos_puestos AS (
    -- Seleccionar el equipo con la mejor diferencia de goles en caso de empate en el segundo puesto
    SELECT ees.equipo
    FROM equipos_empatados_segundo ees
    JOIN goles_empatados g ON ees.equipo = g.equipo AND ees.temporada = g.temporada
    WHERE g.golesFavor = (
        -- Seleccionamos el equipo con la mejor diferencia de goles
        SELECT MAX(g2.golesFavor)
        FROM goles_empatados g2
        WHERE g2.temporada = ees.temporada
    )
)
SELECT DISTINCT ee.nombreOficial
FROM equipos_estadios ee
JOIN (
    SELECT equipo FROM primeros_puestos
    UNION ALL
    SELECT equipo FROM segundos_puestos
) pe ON ee.nombreOficial = pe.equipo
ORDER BY ee.nombreOficial;
```

Código 5: Código de la Consulta 2

```
NOMBREOFICIAL
------------------------------------------------------------
Real Club Deportivo de La Corugna
Real Sociedad de Futbol
Real Zaragoza
Valencia Club de Futbol
```

Código 6: Resultados de la Consulta 2

Pulsa aquí para ver el árbol sintáctico de la Consulta 2.

2.2.3 Consulta 3

```sql
WITH ultima_temporada_segunda AS (
    SELECT idTemporada, agno
    FROM TEMPORADA t1
    WHERE division = '2'
    AND agno = (SELECT MAX(t2.agno) FROM TEMPORADA t2 WHERE t2.division = '2')
),
goles_por_jornada AS (
    SELECT j.idJornada, j.numero AS numero_jornada, j.temporada,
           SUM(p.golesLocal + p.golesVisitante) AS total_goles
    FROM PARTIDO p
    JOIN JORNADA j ON p.jornada = j.idJornada
    WHERE j.temporada IN (SELECT idTemporada FROM ultima_temporada_segunda)
    GROUP BY j.idJornada, j.numero, j.temporada
),
jornada_max_goles AS (
    SELECT temporada, numero_jornada, total_goles
    FROM goles_por_jornada
    WHERE total_goles = (SELECT MAX(g.total_goles) FROM goles_por_jornada g)
)
SELECT t.agno, j.numero_jornada, j.total_goles
FROM jornada_max_goles j
JOIN TEMPORADA t ON j.temporada = t.idTemporada
ORDER BY j.numero_jornada;
```

Código 7: Código de la Consulta 3

```
      AGNO NUMERO_JORNADA TOTAL_GOLES
---------- -------------- -----------
      2015              1          37
```

Código 8: Resultados de la Consulta 3

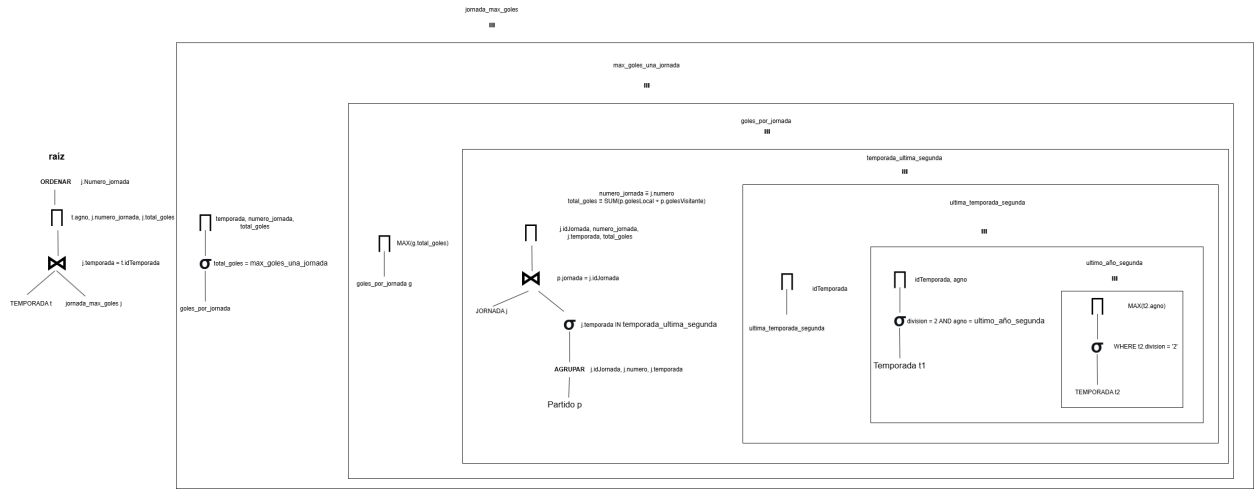

Figura 6: Árbol sintáctico de la consulta 3

3 | Diseño Físico

3.1 Problemas, acciones y mejoras de rendimiento

3.1.1 Consulta 1

(*Este es el Explain Plan de la Consulta 1 antes de ser optimizada.*)

```
| Id  | Operation            | Cost (%CPU) | Time     |
|-----|----------------------|-------------|----------|
|   0 | SELECT STATEMENT     |    440 (2)  | 00:00:01 |
```

En la primera consulta, mediante el comando proporcionado por los docentes para visualizar el plan de ejecución, se ha observado que se realiza una exploración completa (*TABLE ACCESS FULL*) sobre la tabla PARTIDO, JORNADA y TEMPORADA. Por ello, se ha decidido tomar la decisión de utilizar índices para reducir el número de escaneos en las tablas y optimizar las búsquedas.

```
CREATE INDEX idx_partido_equipo_goles ON PARTIDO(equipoLocal, equipoVisitante, golesLocal, golesVisitante
    , jornada);
CREATE INDEX idx_partido_jornada ON PARTIDO(jornada);
CREATE INDEX idx_jornada_temporada ON JORNADA(temporada);
CREATE INDEX idx_temporada_division_agno ON TEMPORADA(division, agno);
```

Código 9: Índices de la Consulta 1

- El primer índice optimiza las búsquedas dado el equipo local o visitante que en este caso es el *Real Zaragoza*. Además, acelera las comparaciones entre goles locales y visitantes y facilita la búsqueda por jornada.
- El segundo se ha creado para optimizar la unión (JOIN) entre PARTIDO y JORNADA. El tercero es parecido al anterior, mejora la relación de JORNADA con TEMPORADA y agiliza las búsquedas por temporada.
- El último mejora el filtrado de temporadas por división y año.

Al añadir estos índices, el coste total ha reducido considerablemente y en vez de observar TABLE ACCESS FULL, aparece en su lugar INDEX RANGE SCAN o INDEX FAST FULL SCAN, lo que significa que los índices están funcionando.

(*Este es el Explain Plan de la Consulta 1 optimizada.*)

```
| Id  | Operation            | Cost (%CPU) | Time     |
|-----|----------------------|-------------|----------|
|   0 | SELECT STATEMENT     |   249 (3)   | 00:00:01 |
```

3.1.2 Consulta 2

(*Este es el Explain Plan de la consulta 2 antes de ser optimizada.*)

```
| Id  | Operation            | Cost (%CPU) | Time     |
|-----|----------------------|-------------|----------|
|   0 | SELECT STATEMENT     |   309 (3)   | 00:00:01 |
```

Respecto a la segunda consulta, se ha contemplado que se realizaban exploraciones completas sobre las tablas PARTIDO, JORNADA, CONTIENE y TEMPORADA. Debido a esto, se ha tomado la decisión de crear índices para reducir el número de escaneos completos de tabla (*TABLE ACCESS FULL*) y optimizar las búsquedas y uniones (*JOIN*), mejorando así la eficiencia de la consulta.

```
CREATE INDEX idx_contiene_temporada_puntos ON CONTIENE (temporada, puntos DESC);
CREATE INDEX idx_contiene_equipo ON CONTIENE (equipo);
CREATE INDEX idx_partido_local_visitante ON PARTIDO (equipoLocal, equipoVisitante);
CREATE INDEX idx_partido_jornada ON PARTIDO (jornada);
CREATE INDEX idx_jornada_temporada ON JORNADA (temporada);
CREATE INDEX idx_temporada_division ON TEMPORADA (division);
CREATE INDEX idx_estadio_capacidad ON ESTADIO (capacidad);
CREATE INDEX idx_equipo_estadio ON EQUIPO (estadio);
```

Código 10: Índices de la Consulta 2

- El primer índice acelera la consulta al buscar los equipos con mayor puntaje en una temporada específica y optimiza las subconsultas que buscan el primer y segundo puesto en cada temporada.
- El segundo mejora la eficiencia en las consultas que filtran por equipo, especialmente en la selección de equipos en primer y segundo puesto.
- Permite una búsqueda más eficiente de los partidos donde un equipo jugó como local o visitante y optimiza el cálculo de goles a favor y en contra en la consulta.
- Respecto al cuarto índice, acelera la consulta al unir PARTIDO con JORNADA, facilitando la asociación entre los partidos y la temporada correspondiente.
- Facilita la búsqueda de jornadas dentro de una temporada y mejora la unión entre JORNADA y TEMPORADA en los cálculos de partidos jugados.
- El sexto optimiza la consulta que filtra temporadas de la primera división (WHERE division = '1').
- El índice de la tabla ESTADIO permite filtrar rápidamente los estadios con capacidad menor a 50,000.
- El último índice acelera la consulta que une EQUIPO con ESTADIO para determinar qué equipos juegan en estadios pequeños.

Tras añadir estos índices, el coste total de la consulta apenas se ha reducido. Lo bueno es que en lugar de TABLE ACCESS FULL, ahora el plan de ejecución muestra INDEX RANGE SCAN o INDEX FAST FULL SCAN, lo que indica que la base de datos está utilizando los índices correctamente y ha mejorado la eficiencia en la recuperación de datos. Se han probado otros índices con los cuales el porcentaje CPU disminuye pero el coste aumenta.

(*Este es el Explain Plan de la Consulta 2 optimizada.*)

```
| Id  | Operation            | Cost (%CPU) | Time     |
|-----|----------------------|-------------|----------|
|   0 | SELECT STATEMENT     |   297 (3)   | 00:00:01 |
```

3.1.3 Consulta 3

(*Este es el Explain Plan de la Consulta 3 antes de ser optimizada.*)

```
| Id  | Operation            | Cost (%CPU) | Time     |
|-----|----------------------|-------------|----------|
|   0 | SELECT STATEMENT     |   124 (2)   | 00:00:01 |
```

Al igual que en las anteriores, se observa que se realiza una exploración completa en la tabla PARTIDO, TEMPORADA y JORNADA. Primero de todo, se ha optado por una partición horizontal de la tabla PARTIDO por el id de la jornada ya que este contiene tanto la jornada como la temporada en la que se jugó dicha jornada. Con esto se quiere conseguir que para consultar los partidos de la última temporada solo tenga que acceder a estos y no tenga que recorrer toda la tabla.

```
PARTITION BY RANGE (jornada) (
    PARTITION PARTIDO_HISTORICO VALUES LESS THAN (3375),
    PARTITION PARTIDO_ULTIMA_TEMPORADA VALUES LESS THAN (3399),
    PARTITION PARTIDO_FUTURO VALUES LESS THAN (MAXVALUE)
);
```

Código 11: Partición de la Consulta 3

Sin embargo, los resultados no han sido los esperados y la consulta no ha mejorado eficientemente por lo que se ha decidido añadir índices.

```
CREATE INDEX idx_partido_jornada ON PARTIDO (jornada);
CREATE INDEX idx_jornada_temporada ON JORNADA (idJornada, temporada);
CREATE INDEX idx_temporada_division ON TEMPORADA (division, agno, idTemporada);
```

Código 12: Índices de la Consulta 3

- Con el primer índice, Oracle encuentra los partidos de una jornada en específico de forma más rápida.
- Respecto al segundo, es el mismo que se utiliza en la consulta 1.
- El tercero es utilizado para encontrar la última temporada sin escanear toda la tabla.

Con estos índices el rendimiento de la consulta ha mejorado considerablemente reduciendo en gran medida el coste total.

(*Este es el Explain Plan de la Consulta 3 optimizada.*)

```
| Id  | Operation            | Cost (%CPU) | Time     |
|-----|----------------------|-------------|----------|
|   0 | SELECT STATEMENT     |    51 (4)   | 00:00:01 |
```

3.2 Triggers

Para realizar los triggers, se ha revisado el script de creación de tablas, identificando restricciones no gestionables con CHECK. Tras este análisis, se han seleccionado los siguientes triggers.

3.2.1 Trigger 1

Este trigger se utiliza para actualizar los puntos de los equipos en las distintas temporadas que participan cuando hay cambios en la tabla PARTIDO.

```
CREATE OR REPLACE TRIGGER trg_actualizar_puntos
AFTER INSERT OR UPDATE OR DELETE ON PARTIDO
BEGIN
    UPDATE CONTIENE c
    SET c.puntos = (
        SELECT SUM(puntos)
        FROM (
            SELECT p.equipoLocal AS equipo, t.idTemporada AS temporada, 3 AS puntos
            FROM PARTIDO p
            JOIN JORNADA j ON p.jornada = j.idJornada
            JOIN TEMPORADA t ON j.temporada = t.idTemporada
            WHERE p.golesLocal > p.golesVisitante
            UNION ALL
            SELECT p.equipoLocal AS equipo, t.idTemporada AS temporada, 1 AS puntos
            FROM PARTIDO p
            JOIN JORNADA j ON p.jornada = j.idJornada
            JOIN TEMPORADA t ON j.temporada = t.idTemporada
            WHERE p.golesLocal = p.golesVisitante
            UNION ALL
            SELECT p.equipoVisitante AS equipo, t.idTemporada AS temporada, 1 AS puntos
            FROM PARTIDO p
            JOIN JORNADA j ON p.jornada = j.idJornada
            JOIN TEMPORADA t ON j.temporada = t.idTemporada
            WHERE p.golesVisitante = p.golesLocal
            UNION ALL
            SELECT p.equipoVisitante AS equipo, t.idTemporada AS temporada, 3 AS puntos
            FROM PARTIDO p
            JOIN JORNADA j ON p.jornada = j.idJornada
            JOIN TEMPORADA t ON j.temporada = t.idTemporada
            WHERE p.golesVisitante > p.golesLocal
            UNION ALL
            SELECT p.equipoLocal AS equipo, t.idTemporada AS temporada, 0 AS puntos
            FROM PARTIDO p
            JOIN JORNADA j ON p.jornada = j.idJornada
            JOIN TEMPORADA t ON j.temporada = t.idTemporada
            WHERE p.golesLocal < p.golesVisitante
            UNION ALL
            SELECT p.equipoVisitante AS equipo, t.idTemporada AS temporada, 0 AS puntos
            FROM PARTIDO p
            JOIN JORNADA j ON p.jornada = j.idJornada
            JOIN TEMPORADA t ON j.temporada = t.idTemporada
            WHERE p.golesVisitante < p.golesLocal
        ) puntos_totales
        WHERE puntos_totales.equipo = c.equipo AND puntos_totales.temporada = c.temporada
    )
    WHERE EXISTS (
        SELECT 1 FROM PARTIDO p
        JOIN JORNADA j ON p.jornada = j.idJornada
        JOIN TEMPORADA t ON j.temporada = t.idTemporada
        WHERE (p.equipoLocal = c.equipo OR p.equipoVisitante = c.equipo) AND t.idTemporada = c.temporada
    );
    UPDATE CONTIENE SET puntos = 0 WHERE puntos IS NULL;
END;
/
```

Código 13: Trigger 1

3.2.2 Trigger 2

El siguiente trigger es utilizado para comprobar que un equipo no juegue el mismo año en primera y segunda división ya que esto es una inconsistencia debido a que es imposible que este suceso ocurra en la vida real.

```
CREATE OR REPLACE TRIGGER trg_evitar_primera_segunda
BEFORE INSERT OR UPDATE ON contiene
FOR EACH ROW
DECLARE
    v_count NUMBER;
    v_division VARCHAR2(20);
BEGIN
    -- Obtener la division de la nueva temporada
    SELECT division INTO v_division
    FROM TEMPORADA
    WHERE idTemporada = :NEW.temporada;

    -- Contar cuantas veces el equipo ha jugado en Primera o Segunda en el
        mismo agno
    SELECT COUNT(DISTINCT t.division)
    INTO v_count
    FROM contiene c
    JOIN TEMPORADA t ON c.temporada = t.idTemporada
    WHERE c.equipo = :NEW.equipo
      AND t.agno = (SELECT agno FROM TEMPORADA WHERE idTemporada = :NEW.
          temporada)
      AND t.division IN ('1', '2')  -- Solo nos interesa Primera y Segunda
      AND c.temporada <> :NEW.temporada;  -- Excluir la misma temporada

    -- Si el equipo ya ha jugado en Primera o Segunda en el mismo agno y
       ahora se intenta agregar en una de esas divisiones, impedir la
       insercion
    IF v_count > 0 AND v_division IN ('1', '2') THEN
        RAISE_APPLICATION_ERROR(-20010, 'Un equipo no puede jugar en
            Primera y Segunda en la misma temporada.');
    END IF;
END;
/
```

Código 14: Trigger 2

3.2.3 Trigger 3

Por último, el tercer trigger valida que en un partido de una determinada temporada no se inserte un equipo cuya fecha de fundación registrada es posterior al año de la respectiva temporada.

```
CREATE OR REPLACE TRIGGER trg_validar_fundacion_equipo
BEFORE INSERT OR UPDATE ON PARTIDO
FOR EACH ROW
DECLARE
    v_fundacion_local NUMBER;
    v_fundacion_visitante NUMBER;
    v_anio_temporada NUMBER;
BEGIN
    -- Obtener el agno de fundacion del equipo local
    SELECT fechaFundacion INTO v_fundacion_local
    FROM EQUIPO
    WHERE nombreOficial = :NEW.equipoLocal;

    -- Obtener el agno de fundacion del equipo visitante
    SELECT fechaFundacion INTO v_fundacion_visitante
    FROM EQUIPO
    WHERE nombreOficial = :NEW.equipoVisitante;

    -- Obtener el agno de la temporada del partido
    SELECT agno INTO v_anio_temporada
    FROM TEMPORADA t
    JOIN JORNADA j ON t.idTemporada = j.temporada
    WHERE j.idJornada = :NEW.jornada;

    -- Si el equipo local tiene fecha de fundacion, verificar que no sea posterior
        a la temporada
    IF v_fundacion_local IS NOT NULL AND v_fundacion_local > v_anio_temporada THEN
        RAISE_APPLICATION_ERROR(-20060, 'Error: El equipo local no puede jugar en
            una temporada anterior a su fundacion.');
    END IF;

    -- Si el equipo visitante tiene fecha de fundacion, verificar que no sea
        posterior a la temporada
    IF v_fundacion_visitante IS NOT NULL AND v_fundacion_visitante >
        v_anio_temporada THEN
        RAISE_APPLICATION_ERROR(-20061, 'Error: El equipo visitante no puede jugar
            en una temporada anterior a su fundacion.');
    END IF;
END;
/
```

Código 15: Trigger 3

4 | Anexo: Coordinación del grupo

A la hora de coordinarnos para hacer la BD no se ha tenido ningún problema significativo, esto puede deberse a la experiencia previa del equipo trabajando juntos y por lo tanto se conoce que cosas se le dan mejor a cada uno, consiguiendo así una distribución más rápida y eficiente de las tareas.
Cuando ha sido necesario reunirse para discutir sobre decisiones a tomar, o para tareas que necesitasen de los 3 integrantes del grupo, el equipo se ha reunido tanto de forma presencial como de forma online mediante Discord.
Para asegurarse de que todos los integrantes del equipo se han enterado de todo, incluso de las partes en las que menos se ha trabajado, se han explicado mutuamente los avances realizados, asegurando así una comprensión completa del desarrollo de la práctica.

Tareas	**Carlos**	**Rodrigo**	**Mario**
Esquema E/R	4	6	4
Esquema Relacional	0.5	1.5	2
Creación de tablas	3	2	3
Población BD	7	3	5.5
Consulta 1	2.5	2.5	1.5
Consulta 2	6	5	3.5
Consulta 3	1.5	2	0.5
Rendimiento consultas	5	4	6
Trigger 1	2	1	2.5
Trigger 2	2	1	2
Trigger 3	0.5	1	1.5
Total	**34**	**29**	**32**

Table 0.1: Tabla de tareas asignadas a Carlos, Rodrigo, y Mario