MEMORIA

PRÁCTICA 1 - BASE DE DATOS DE FÚTBOL

Asignatura: Bases de datos

Grupo J2.04, curso 2023-2024

Fecha de entrega : 06/03/2024

PARTICIPANTES:

ATHANASIOS USERO SAMARÁS, EMAIL : 839543@unizar.es
CURRO VALERO CASAJÚS , EMAIL : 842545@unizar.es
DANIEL RAMÓN MONTERRUBIO, EMAIL: 872428@unizar.es

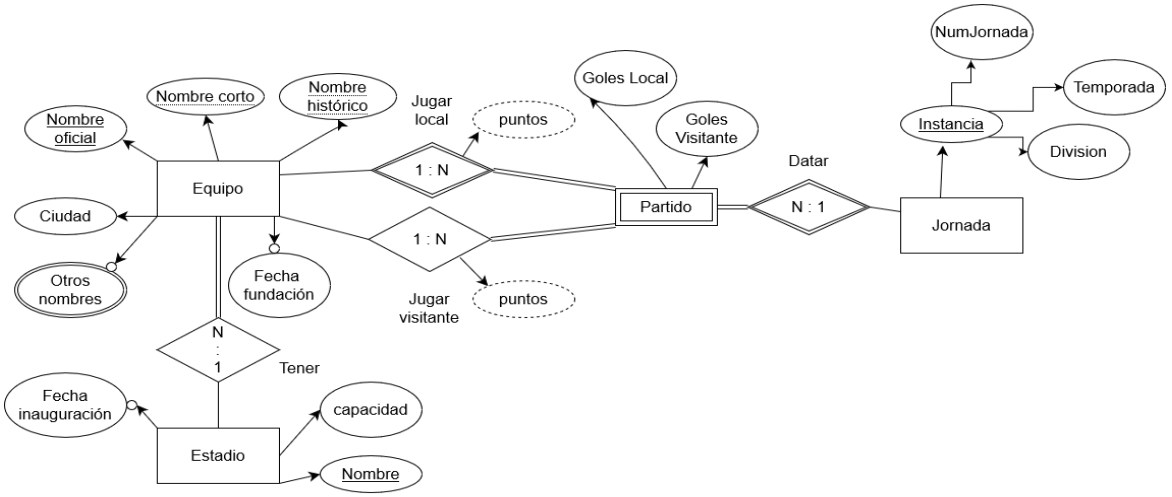
Índice

Índice 2
PARTE 1: Creación de una base de datos 3
1.1 Modelo E/R: 3
1.2 Esquema relacional 4
1. 3 Creación de tablas 5
PARTE 2: Introducción de datos y ejecución de consultas 7
2.1 Población de la BD 7
2.2 Consultas SQL 9
2.2.1 Consulta 1 9
2.2.2 Consulta 2 11
2.2.3 Consulta 3 12
PARTE 3: DISEÑO FÍSICO 14
3.1 Mejoras de Rendimiento 14
3.2 Triggers 15
Anexo 1. Evaluación individual 19
Anexo 2. Consulta 1 sin puntos derivados 20

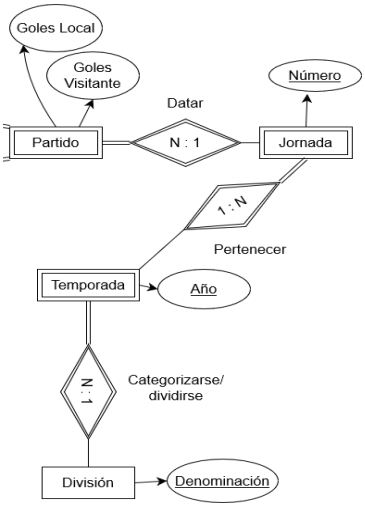
PARTE 1: Creación de una base de datos

1.1 Modelo E/R:

El esquema E/R global diseñado es el siguiente:



- **Restricciones:** Un partido no puede tener jugando como local y visitante al mismo equipo, y tampoco puede darse que dos equipos jueguen como local y visitante entre ellos en más de una jornada de una misma temporada y división. Un equipo no puede disputar ni como local ni como visitante más de un partido dada la jornada de una temporada y división, y tampoco podrá disputar partidos datados en jornadas de primera y segunda o promociones una misma temporada. El valor de los goles de un partido tiene que ser mayor o igual a 0, el número de jornada mayor que 0, el de la temporada mayor que 1971 y el de las demás fechas mayor que 1888. Los puntos asociados a la participación de un equipo de un partido se calculan como: 0 si ha marcado menos goles que su contrincante, 1 si ambos han marcado los mismos goles y 3 si ha marcado más goles.
- **Premisas:** Se ha supuesto que todo equipo tendrá un estadio (si no, la participación de Equipo en su relación con Estadio sería de 0 y no 1), y que podrá identificarse igualmente por su nombre oficial, corto, o histórico. También se ha supuesto que para todo estadio se puede conocer su capacidad, pero no así con su fecha de inauguración ni la de fundación de equipos. Además, dada la trayectoria histórica de la liga española, se ha considerado que las promociones siempre están asociadas con la segunda división, así como que dos temporadas diferentes inician en años distintos, y por tanto pueden quedar identificadas totalmente por su comienzo.



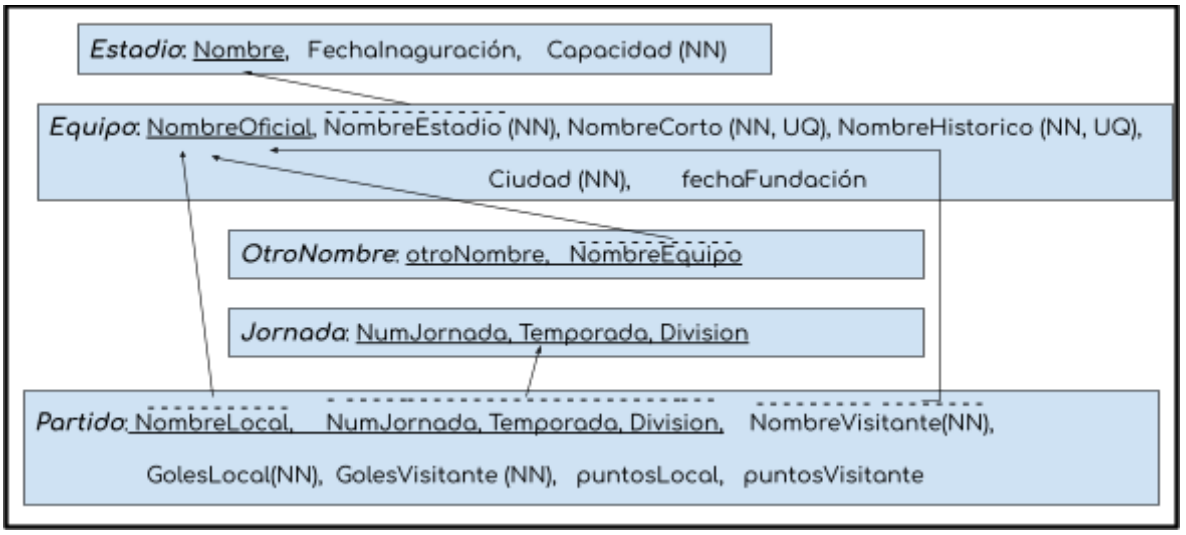
Las cuestiones más debatibles giraban en torno a la identificación de la entidad débil Partido y la representación de instancias (jornada, temporada, división). Cabe incluir dos soluciones alternativas:

- Considerar *Partido* como una entidad fuerte cuya clave primaria fuese artificial. El esquema queda modificado quedando sin relaciones identificativas y con un nuevo atributo identificador primario *idPartido*.
- Desglosar la entidad *Jornada* en tres entidades *Jornada, Temporada y División,* débiles respecto a la siguiente. Tal propuesta parece apropiada para gestiones de control individual de cada entidad (por ejemplo, dada una

división ofrecer un número de temporadas válidas), pero se descartó por la ausencia de dichos procesos en el proyecto. Las modificaciones resultantes aparecen en la figura al margen derecho.

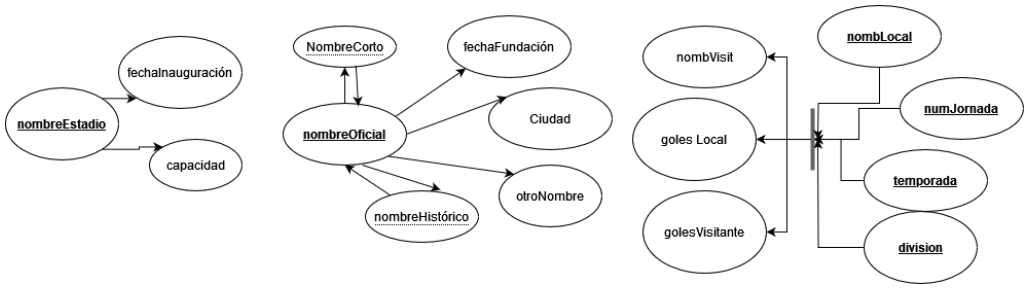
1.2 Esquema relacional

El mapeado directo del modelo E/R a relacional resulta en el siguiente esquema:



Este hereda todas las restricciones y premisas del modelo E-R, y para la relación *Equipo* se ha decidido mantener *NombreOficial* como clave primaria. Cabe indicar que existe una nueva relación *OtroNombre*, que aparecía como atributo multivaluado de *Equipo* en el modelo relacional, pero que al transformar se ha convertido en una nueva relación ya que el modelo relacional no admite atributos multivaluados.

Por esta última razón, el esquema está en 1FN. Además, construyendo un grafo de dependencias funcionales (del que se excluyen los puntos del local y visitante por ser atributos derivados),



se puede constatar que todo atributo no clave depende funcionalmente de la clave primaria completa, por lo que también se cumple la 2FN. Además, también se puede probar que el esquema está en 3FN, puesto que los goles marcados en un partido solo dependen de la instancia en la que se disputaron y del equipo local (o visitante), al solo poder jugar una vez el equipo por jornada. Por último, también es inherente al esquema la FNBC, puesto que por una parte se ha considerado que cualquier otro nombre dado a un equipo no puede determinar al mismo, y que tampoco puede determinar el equipo visitante a ningún atributo que forme parte de la clave. En todo caso, {nombVisit, numJornada, temporada, división} → {nombLocal}, pero el primer conjunto forma una clave candidata, y además

nombVisit no actúa por sí solo, sino que forma parte del mismo conjunto. Por tanto, el esquema presentado inicialmente está en Forma Normal de Boyce-Codd.

1. 3 Creación de tablas

Las sentencias SQL de creación de tablas se muestran a continuación:

```
Unset
CREATE TABLE Estadio (
       nombEstadio VARCHAR(60) CONSTRAINT nombEstadio_PK PRIMARY KEY,
       capacidad NUMBER(8),
       fechInaug NUMBER(4),
       CONSTRAINT fechaInaug_VAL CHECK (fechInaug > 1888),
       CONSTRAINT capacidad_POS  CHECK (capacidad > 0)
);
CREATE TABLE Equipo (
       nombOf VARCHAR(60) CONSTRAINT nombOf_UN UNIQUE,
       nombCorto VARCHAR(60) CONSTRAINT nombCorto_PK PRIMARY KEY,
       nombHist  VARCHAR(60),
       ciudad VARCHAR(60),
       fechFund     NUMBER(4),
       nombEstadioEq VARCHAR(60),
        CONSTRAINT  nombEstadioEq_FK FOREIGN KEY (nombEstadioEq) REFERENCES
                                                Estadio(nombEstadio),
       CONSTRAINT fechFund_VAL CHECK (fechFund > 1888)
);
CREATE TABLE otrosNombres (
       otroNomb VARCHAR(60),
       nombEq  VARCHAR(60)  CONSTRAINT nombOfEq_NN  NOT NULL,
       CONSTRAINT nombEq_FK FOREIGN KEY (nombEq) REFERENCES Equipo(nombCorto)
                                                ON DELETE CASCADE,
       CONSTRAINT otrosNombres_PK PRIMARY KEY (otroNomb, nombEq)
);
CREATE TABLE Jornada (
       numJornada  NUMBER(2),
       temporada    NUMBER(4),
       division     VARCHAR(60),
       CONSTRAINT instancia_PK   PRIMARY KEY (numJornada, temporada,
division),
       CONSTRAINT numJornada_POS CHECK (numJornada > 0),
       CONSTRAINT temporada_VAL  CHECK (temporada >= 1972)
);
CREATE TABLE Partido (
       golesLocal NUMBER(2) CONSTRAINT golesLocal_NN NOT NULL,
       golesVisit NUMBER(2) CONSTRAINT golesVisit_NN NOT NULL,
       puntosLocal NUMBER(1),
```

```
        puntosVisit NUMBER(1),
        nombLocal VARCHAR(60),
        nombVisit VARCHAR(60)  CONSTRAINT nombVisit_NN NOT NULL,
        jorPart  NUMBER(2),
        tempPart      NUMBER(4),
        divPart       VARCHAR(60),
        CONSTRAINT nombLocal_PK FOREIGN KEY (nombLocal) REFERENCES
Equipo(nombCorto)
                            ON DELETE CASCADE,
        CONSTRAINT nombVisit_PK FOREIGN KEY (nombVisit) REFERENCES
Equipo(nombCorto)
                            ON DELETE CASCADE,
        CONSTRAINT partInst_FK FOREIGN KEY (JorPart, tempPart, divPart)
                                        REFERENCES Jornada(numJornada,
temporada, division)
                                        ON DELETE CASCADE,
        CONSTRAINT partido_PK PRIMARY KEY (nombLocal, jorPart, tempPart,
divPart),
        CONSTRAINT unicidadJor_VIS UNIQUE (nombVisit, jorPart, tempPart,
divPart)
);
```

Debido a ciertos aspectos en el lote de datos a introducir, fue necesario modificar algunas características a la hora de crear las tablas. Sin embargo, dado que estos cambios no suponen un cambio sustancial en los modelos planteados anteriormente, y con el fin de no solapar el diseño conceptual y lógico desarrollado a partir del enunciado con el quizás algo más práctico posterior, se decidió incluir en la memoria los modelos E/R y relacional inalterados ante estos cambios.

Con el fin de determinar las particularidades de la tabla de datos proporcionada, se efectuaron diversas consultas. En primer lugar, se comprobó que existen 21 equipos distintos que han participado en partidos pero que en ninguna ocurrencia presentan más información que la de su nombre corto. Esto llevó a establecer el nombre corto como clave primaria, mientras que el nombre oficial, histórico, ciudad y el estadio se establecen como no obligatorios. Por otra parte, otra consulta despojó que, pese a que la unicidad de los nombres históricos se cumple en general, hay una anomalía que obligó a establecerlo como no único, (la Agrupacion Deportiva Alcorcon comparte nombre histórico con su filial). Por último, mencionar que la restricción relativa a la unicidad de encuentros entre dos equipos como local y visitante no se pudo implementar dado que durante la temporada 1986/87 hubo más de un encuentro repetido en la misma división donde un equipo jugaba como local y el otro como visitante. Su implementación requería de otros recursos.

Sin embargo, sí se incluyeron otras restricciones, como aquellas asociadas a dominios de datos, o que un equipo no pueda jugar como visitante en una jornada más de una vez. Con fines didácticos, se pospuso la implementación de las restricciones del valor válido de goles e imposibilidad de que un equipo se enfrente a sí mismo para su gestión con disparadores.

PARTE 2: Introducción de datos y ejecución de consultas

2.1 Población de la BD

Para poblar la base de datos, se optó por insertar en las tablas el resultado de consultas sobre la tabla *datosdb.ligahost:*

```
Unset
INSERT INTO ESTADIO SELECT DISTINCT ESTADIO, AFORO, FECHA_INAG
FROM datosdb.ligahost
WHERE ESTADIO IS NOT NULL;

INSERT INTO JORNADA SELECT DISTINCT JORNADA, INICIO_TEMPORADA, DIVISION
FROM datosdb.ligahost
WHERE JORNADA IS NOT NULL AND INICIO_TEMPORADA IS NOT NULL
      AND DIVISION IS NOT NULL;

INSERT INTO EQUIPO SELECT DISTINCT CLUB, EQUIPO_LOCAL, NOMBRE, CIUDAD,
                                   FUNDACION, ESTADIO
FROM datosdb.ligahost
WHERE EQUIPO_LOCAL IS NOT NULL;

INSERT INTO OTROSNOMBRES SELECT DISTINCT EQUIPO, EQUIPO_LOCAL
FROM datosdb.ligahost
WHERE EQUIPO IS NOT NULL AND EQUIPO_LOCAL IS NOT NULL;

INSERT INTO PARTIDO SELECT DISTINCT GOLES_LOCAL, GOLES_VISITANTE,
                           TO_NUMBER(NULL), TO_NUMBER(NULL), EQUIPO_LOCAL,
                           EQUIPO_VISITANTE, JORNADA,INICIO_TEMPORADA,
                           DIVISION
FROM datosdb.ligahost
WHERE GOLES_LOCAL IS NOT NULL AND GOLES_VISITANTE IS NOT NULL AND
      EQUIPO_LOCAL IS NOT NULL AND EQUIPO_VISITANTE IS NOT NULL AND JORNADA
      IS NOT NULL AND INICIO_TEMPORADA IS NOT NULL AND DIVISION IS NOT NULL;
```

Alguno de los problemas, como la no obligatoriedad de algunos atributos, se han detallado anteriormente por ser inherentes a la definición de tablas.Y es que el proceso de comprobaciones previas a la creación de tablas evitó el enfrentamiento posterior con algunas particularidades en el formato de los datos proporcionados.

Dicho esto, el orden de inserción se basó en la restricción de integridad referencial. De este modo, se poblaron primero las tablas sin claves ajenas (*Jornada* y *Equipo*), y posteriormente el resto de tablas, de manera que toda clave ajena hiciese referencia a una tabla ya creada.

Para evitar problemas de integridad de entidades y obligatoriedad, se seleccionaron tuplas de la tabla con los atributos requisados no nulos. Como consecuencia, se omitieron hasta 62 equipos especificados con nombres oficiales distintos que en toda ocurrencia aparecían disociados de cualquier partido y cuyo nombre corto era desconocido. Pero tampoco se mostraba lógico tratar de introducirlos, puesto que la base de datos gestiona fundamentalmente datos de partidos de ligas, no de equipos.

Por último, mencionar que la selección de columnas se realizó de manera conjuntista (*Distinct*), para evitar que intentase almacenar más de una vez la misma tupla (por ejemplo, un equipo puede aparecer en miles de partidos). Asimismo, indicar que se tomó la licencia de emplear la columna del nombre local para introducir los nombres cortos, puesto que todo equipo que apareció en algún partido como local lo hizo igualmente como visitante en al menos otro partido.

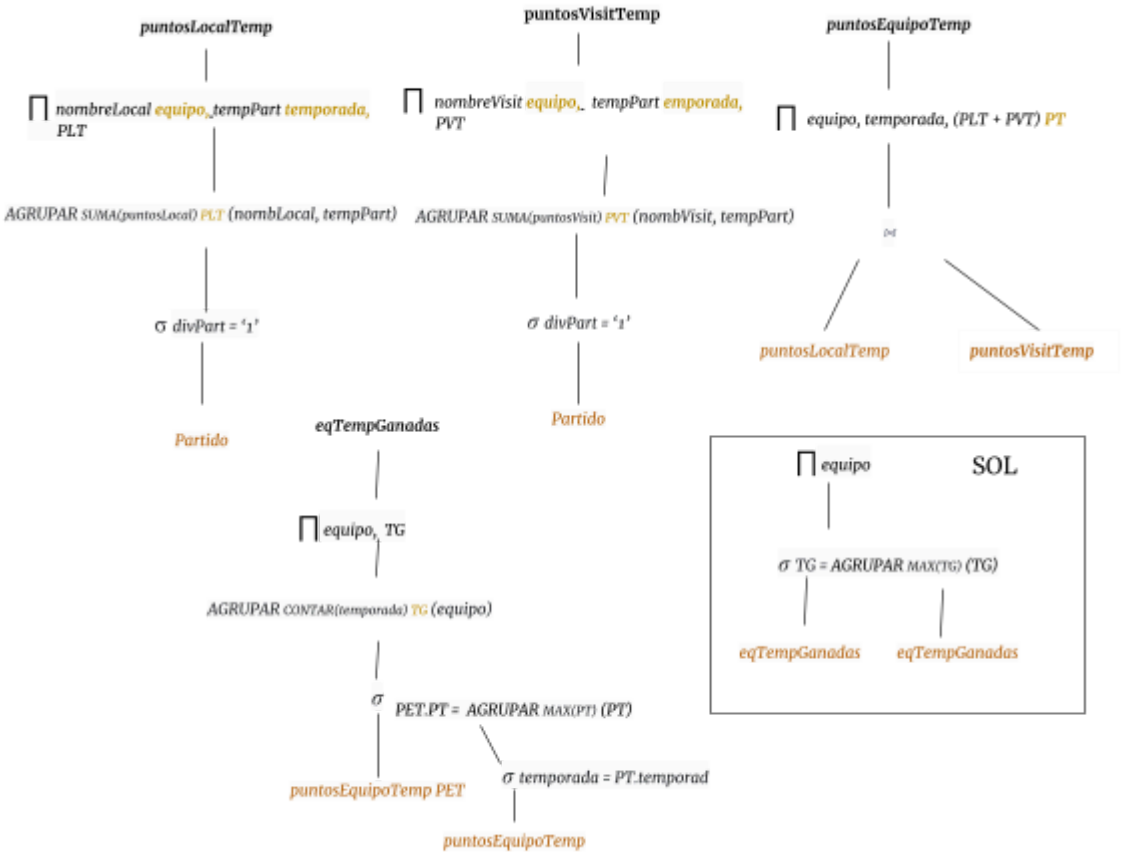
2.2 Consultas SQL

2.2.1 Consulta 1

Equipo que más ligas de primera división ha ganado.

Para la primera consulta, se tomó una asunción importante. En la especificación de la base de datos se estableció que el puesto del equipo se decide de acuerdo a su puntuación en una temporada, pero no se indicó un segundo criterio en caso de empate. Por tanto, se aplicará tan solo el criterio de puntos, permitiendo que dos equipos compartan el mismo puesto en la clasificación de una división.

A continuación se muestra el árbol sintáctico resultante:



Traduciendo el álgebra a lenguaje sql, se obtiene:

```
Unset
WITH
   -- Puntos ganados como local por equipo por temporada de primera
   puntosLocalTemp AS
       (SELECT nombLocal AS equipo, tempPart AS temporada, SUM(puntosLocal)
AS puntosLocalTot
       FROM PARTIDO
```

```
        WHERE divPart = '1'
        GROUP BY nombLocal, tempPart),
    -- Puntos ganados como visitante por equipo por temporada de primera
    puntosVisitTemp AS
        (SELECT nombVisit AS equipo, tempPart AS temporada, SUM(puntosVisit)
AS puntosVisitTot
        FROM PARTIDO
        WHERE divPart = '1'
        GROUP BY nombVisit, tempPart),
    -- Puntos totales ganados por equipo por temporada de primera
    puntosEquipoTemp AS
        (SELECT equipo, temporada, (puntosLocalTot + puntosVisitTot) AS
puntosTemp
        FROM puntosLocalTemp NATURAL JOIN puntosVisitTemp),
    -- Ligas de primera división ganadas por equipo
    eqTempGanadas AS
        (SELECT equipo, COUNT(temporada) tempGanadas
        FROM PuntosEquipoTemp PT
        WHERE PT.puntosTemp = (SELECT MAX(puntosTemp) FROM PuntosEquipoTemp
                               WHERE temporada = PT.temporada)
        GROUP BY equipo
        ORDER BY tempGanadas DESC)
-- Equipo con más ligas de primera ganadas
SELECT equipo "Equipo/s con mas ligas de primera ganadas"
FROM eqTempGanadas
WHERE tempGanadas = (SELECT MAX(tempGanadas) FROM eqTempGanadas);
```

La respuesta obtenida por la base de datos ha sido:

```
Equipo/s con más ligas de primera ganadas
- ----------------------------
Real Madrid
```

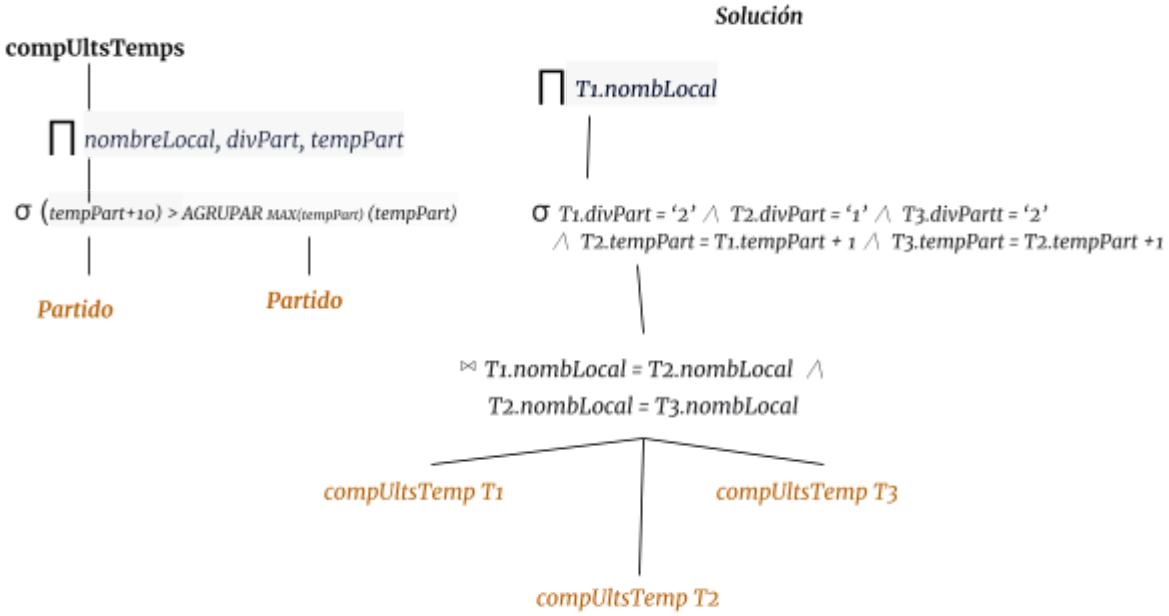
Adicionalmente, como quizás no estaba previsto el empleo de un atributo derivado para los puntos por parte de la dirección, se decidió abordar otra consulta sin hacer uso de ellos. Esta consulta se recoge en el Anexo 2., y devolvía el mismo resultado que la principal.

En la consulta se calculan los puntos ganados por los equipos como local y como visitante y suman los mismos para generar una clasificación de cada temporada. Finalmente se selecciona el equipo que más puntuación ha conseguido en cada temporada y se consigue el equipo que más veces ha conseguido la mayor puntuación en más ocasiones.

2.2.2 Consulta 2

Equipos de segunda división que han ascendido a primera y al año siguiente han vuelto a descender en las últimas diez temporadas.

A continuación se muestra el árbol sintáctico resultante:



En cuanto a la consulta:

```
Unset
WITH
       -- Composición de cada división para las últimas 10 temporadas
       compUltsTemps AS
       (SELECT DISTINCT nombLocal, divPart, tempPart
       FROM Partido
       WHERE tempPart + 10 > (SELECT DISTINCT MAX(tempPart) FROM Partido))
SELECT DISTINCT T1.nombLocal "Equipo/s Ascendentes-Descendentes"
FROM  compUltsTemps T1
       JOIN compUltsTemps T2 ON T1.nombLocal = T2.nombLocal
       JOIN compUltsTemps T3 ON T2.nombLocal = T3.nombLocal
WHERE T1.divPart = '2' AND T2.divPart = '1' AND T3.divPart = '2'
       AND T2.tempPart = T1.tempPart + 1 AND T3.tempPart = T2.tempPart + 1;
```

En la consulta se han considerado combinaciones de tres ocurrencias de la composición de las últimas ligas donde el equipo es el mismo. De este modo, cada tripleta de datos puede aparecer con cualquier permutación, por lo que si el equipo cumple la condición impuesta en el enunciado habrá una combinación para ese equipo con valores de temporada ascendentes. Por otra parte se ha empleado un JOIN y no más condiciones para facilitar la legibilidad del código.

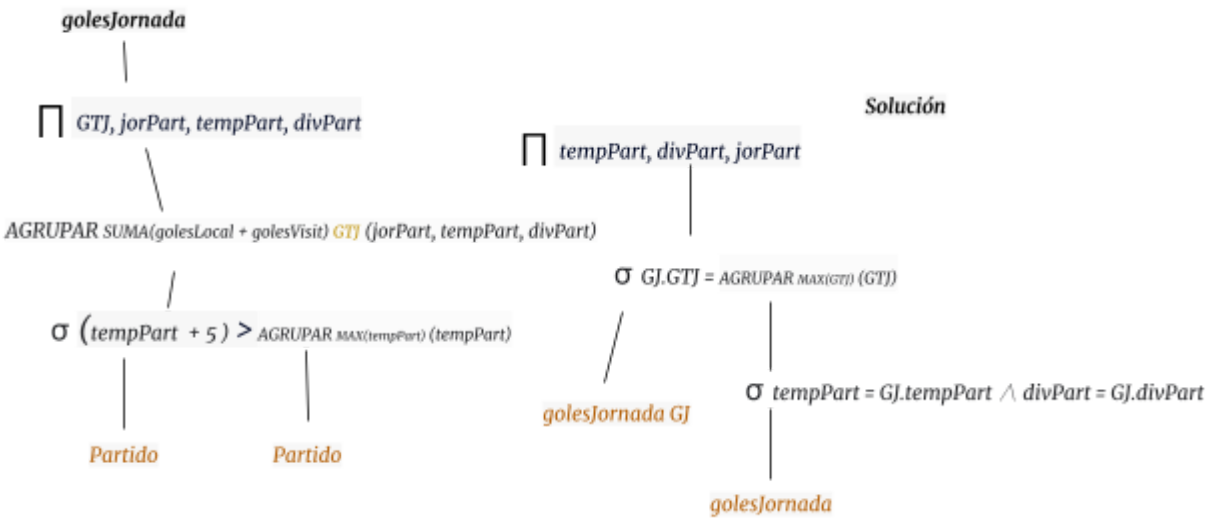
A continuación se muestra una tabla con los resultados obtenidos:

Equipo/s ascendente-descendentes
- -
Tenerife
Hercules
Deportivo. Coru?a
Numancia
Cordoba
Murcia
Xerez

2.2.3 Consulta 3

Jornadas de las últimas cinco temporadas donde se han marcado más goles.

A continuación se muestra el árbol sintáctico resultante:



La pregunta en sql es directa a partir de su álgebra relacional:

```
Unset
WITH
	--- Goles totales por jornada de una temporada y división
	golesJornada AS
	(SELECT SUM(golesLocal + golesVisit) AS golesJor, jorPart, tempPart,
		divPart
	FROM Partido
	WHERE (tempPart + 5) > (SELECT DISTINCT MAX(tempPart) FROM Partido)
	GROUP BY jorPart, tempPart, divPart)
SELECT tempPart "Temporada", divPart "Division", jorPart "Jornada Maxima"
FROM golesJornada GJ
```

```
WHERE golesJor = (SELECT MAX(golesJor)
                  FROM golesJornada
                  WHERE tempPart = GJ.tempPart AND divPart = GJ.divPart)
ORDER BY tempPart Desc, divPart DESC;
```

De esta consulta se debe mencionar una decisión importante a la hora de interpretar el enunciado. Al mencionar la jornada donde se han marcado más goles de las últimas temporadas, se entiende que esas jornadas se diferencian por temporada. Pero además, y en relación con la manera en la que se ha propuesto la entidad *Jornada* en el modelo E/R, la misma jornada está ligada a la división si no se afirma lo contrario. Así, se ha diferenciado también por división.

A continuación se muestra una tabla con los resultados obtenidos:

Temporada	División	Jornada Máxima
2015	2	1
2015	2	20
2014	2	18
2014	1	38
2014	1	4
2014	Promocion	1
2014	Promocion	2
2013	2	14
2013	1	14
2013	Promocion	2
2012	2	5
2012	1	30
2012	1	11
2012	1	38
2012	Promocion	2
2012	2	29
2011	1	24
2011	Promoción	2

PARTE 3: DISEÑO FÍSICO

3.1 Mejoras de Rendimiento

El análisis del rendimiento de consultas será menos riguroso que el resto de las tareas dadas las limitaciones temporales. En todo caso:

- En la consulta 1, se constató cómo las operaciones que más bytes comportan fueron el *HASH JOIN* correspondiente a seleccionar los partidos de la primera división y las funciones de agrupación sobre estas tablas. También se observó que el coste en CPU para realizar la agrupación era bastante elevado.
- En la consulta 2 y 3, el mayor coste en bytes se da al indexar el nombre en la consulta JOIN.

Tanto en la primera consulta como en la segunda se aplican restricciones sobre el atributo de la división. Además, como la división toma y se prevé que tome pocos valores diferentes, se propuso crear un índice bitmap para agilizar el acceso a este.

Se pensó asimismo en materializar la vista de puntos totales debido a que se usará frecuentemente (aunque esta medida no se aplicó), además de crear índices compuestos para temporada y división para agilizar la búsqueda de los valores en la consulta 2.

Finalmente en la tercera consulta se propuso materializar la vista de los goles totales por jornada de una temporada y división (medida tampoco implementada).

Código de los BitMaps e Índices compuestos:

```
Unset
CREATE BITMAP INDEX bitmap_divPart_idx ON Partido(divPart);
```

```
Unset
CREATE INDEX IDX_TEMP_DIV ON Partido(tempPart, divPart);
```

```
Unset
CREATE INDEX IDX_NOMB_TEMP_DIV ON Partido(tempPart, divPart, nombLocal);
```

3.2 Triggers

Trigger 1:

El primer trigger propuesto servirá para calcular los atributos derivados de puntos obtenidos por los equipos (*puntosLocal, puntosVisit*) de cada partido:

```
Unset
CREATE OR REPLACE TRIGGER Puntos
BEFORE INSERT OR UPDATE ON Partido
FOR EACH ROW
BEGIN
    IF  :NEW.golesLocal > :NEW.golesVisit THEN
       :NEW.puntosLocal := 3; -- El local ha ganado
       :NEW.puntosVisit := 0;
    ELSIF :NEW.golesLocal < :NEW.golesVisit THEN
       :NEW.puntosLocal := 0;
       :NEW.puntosVisit := 3; -- El visitante ha ganado
    ELSE                      -- Empate entre ambos
       :NEW.puntosLocal := 1;
       :NEW.puntosVisit := 1;
    END IF;
END;
/
```

El trigger se disparará antes de cada inserción o modificación de datos a nivel de tupla en la tabla *Partido,* pero no se contemplan errores de tabla mutante ya que lo único que el único elemento con el que se trabaja es el pseudo-registro *NEW.* Por otra parte, se podría discutir si la implementación de los puntos como atributos derivados es adecuada, pues el hecho de calcularlos con disparadores podría dar lugar a posibles inconsistencias a largo plazo si se descuida y cambia el sistema de puntaje.

El funcionamiento del trigger se ha comprobado en la primera consulta al comparar los resultados obtenidos con los atributos derivados con aquellos a partir del procedimiento detallado en el anexo.

Trigger 2:

El segundo trigger propuesto mantiene la restricción de que un equipo no puede jugar contra sí mismo:

```
Unset

CREATE OR REPLACE TRIGGER reflexPartido
BEFORE INSERT OR UPDATE ON Partido
FOR EACH ROW
WHEN (NEW.nombLocal = NEW.nombVisit)
BEGIN
  raise_application_error(-20200, 'El equipo '|| :NEW.nombLocal||' no puede
                          jugar contra sí mismo');
END;
/
```

Su funcionamiento se puede observar insertando esta tupla:

```
Unset
INSERT INTO PARTIDO(GOLESLOCAL, GOLESVISIT, PUNTOSLOCAL, PUNTOSVISIT,
NOMBLOCAL, NOMBVISIT, JORPART, TEMPPART, DIVPART) VALUES (2, 0, NULL, NULL,
'Burgos', 'Burgos', 7, 2012, 1);
```

El trigger mostrará el siguiente resultado:

```
ERROR en linea 1:
ORA-20200: El equipo Burgos no puede jugar contra s?? mismo
ORA-06512: en "A842545.REFLEXPARTIDO", linea 2
ORA-04088: error durante la ejecucion del disparador 'A842545.REFLEXPARTIDO'
```

Trigger 3:

El tercer trigger preserva la restricción de que ningún en ningún partido se puede marcar un número de goles negativos:

```
Unset
CREATE OR REPLACE TRIGGER golesNoNeg
BEFORE INSERT OR UPDATE ON Partido
FOR EACH ROW
BEGIN
IF :NEW.golesLocal < 0 AND :NEW.golesVisitante < 0 THEN
     raise_application_error(-20210, 'Los goles del equipo local o del
equipo visitante no pueden ser negativos');
ELSIF :NEW.golesLocal < 0 THEN
    raise_application_error(-20211, 'Los goles del equipo local no
                            pueden ser negativos');
ELSIF :NEW.golesVisit < 0 THEN
    raise_application_error(-20212, 'Los goles del equipo visitante
                            no pueden ser negativos');
END IF;
END;
```

Se ha diseñado para mostrar con mayor precisión el error, distinguiendo entre goles marcados por el local y por el visitante, pero se podría haber generalizado igualmente a goles.

Se puede observar su funcionamiento insertando las siguientes tuplas:

```
Unset
INSERT INTO Partido(golesLocal, golesVisit, puntosLocal, puntosVisit,
nombLocal, nombVisit, jorPart, tempPart, divPart) VALUES (-1, -2, NULL,
NULL, 'Burgos', 'Real Sociedad', 5, 2011, 1);
```

El trigger muestra este error:

```
ERROR en linea 1:
ORA-20003:  El equipo local Burgos y el equipo visitante Real Sociedad no
pueden tener un n??mero negativo de goles
ORA-06512: en "A842545.GOLESNONEG", linea 3
ORA-04088: error durante la ejecucion del disparador 'A842545.GOLESNONEG'
```

```
Unset
INSERT INTO Partido(golesLocal, golesVisit, puntosLocal, puntosVisit,
nombLocal, nombVisit, jorPart, tempPart, divPart) VALUES (-1, 3, NULL, NULL,
'Burgos', 'Real Sociedad', 5, 2011, 1);
```

El trigger muestra este error:

```
ERROR en linea 1:
ORA-20001:  El equipo local Burgos no puede tener un n??mero negativo de goles
ORA-06512: en "A842545.GOLESNONEG", linea 7
ORA-04088: error durante la ejecucion del disparador 'A842545.GOLESNONEG'
```

```
Unset
INSERT INTO Partido(golesLocal, golesVisit, puntosLocal, puntosVisit,
nombLocal, nombVisit, jorPart, tempPart, divPart) VALUES (3, -1, NULL, NULL,
'Burgos', 'Real Sociedad', 5, 2011, 1);
```

El trigger muestra este error:

```
ERROR en linea 1:
ORA-20002: El equipo visitante Real Sociedad no puede tener un n??mero negativo
de goles
ORA-06512: en "A842545.GOLESNONEG", linea 5
ORA-04088: error durante la ejecucion del disparador 'A842545.GOLESNONEG''
```

Anexo 1. Evaluación individual

A continuación se muestra un desglose de las horas que ha empleado cada integrante del equipo a cada apartado:

	ATHANASIOS USERO	**CURRO VALERO**	**DANIEL RAMÓN**
Trabajo de clase	6 h	6 h	6 h
Modelo E/R	3h	1.5h	1.5 h
Esquema Relacional	2h	2h	1 h
Creación Tablas	3h	3h	3 h
Población	4h	4h	3 h
Consultas SQL	8h	4h	3 h
Rendimiento	2h	1.5h	1 h
Triggers	3h	3h	2 h

Anexo 2. Consulta 1 sin puntos derivados

```
Unset
WITH
  -- Tabla con ocurrencia de equipo-temporada por partido ganado y puntos
      asociados a victoria
      partGanados AS
       (SELECT nombCorto AS equipo, tempPart, jorPart, 3 AS puntos  FROM
        Partido, Equipo WHERE
       ((nombCorto = nombLocal AND golesLocal > golesVisit) OR (nombCorto =
nombVisit AND golesVisit > golesLocal)) AND divPart = '1'),
 -- Tabla con ocurrencia de equipo-temporada por partido empatado y puntos
asociados a empate
       partEmp AS (SELECT nombCorto AS equipo, tempPart, jorPart, 1 AS puntos
FROM Partido, Equipo WHERE
       ((nombCorto = nombLocal AND golesLocal = golesVisit) OR (nombCorto =
nombVisit AND golesVisit = golesLocal)) AND divPart = '1'),
 -- Tabla con ocurrencia de equipo-temporada por partido puntuado y puntos
correspondientes
       puntosPart AS (SELECT * FROM partGanados UNION SELECT * FROM partEmp),
 -- Tabla con puntos totales de equipo por temporada
       puntosTemp AS (SELECT equipo, tempPart AS temporada, SUM(puntos)
puntosTot FROM puntosPart
       GROUP BY equipo, tempPart),
 -- Tabla con temporadas ganadas por equipo
       ganadorTemp AS (SELECT equipo, count(temporada) AS tempGanadas FROM
puntosTemp PT WHERE
       PT.puntosTot = (SELECT MAX(puntosTot) FROM puntosTemp
           WHERE PT.temporada = temporada)
       GROUP BY equipo)
-- Equipo con más primeras ganadas
SELECT equipo FROM ganadorTemp WHERE tempGanadas = (SELECT MAX(tempGanadas)
FROM ganadorTemp);
```