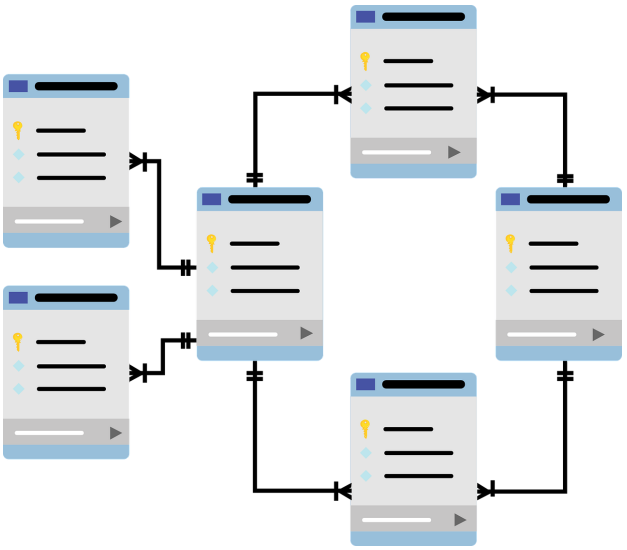
BASES DE DATOS: PRACTICA 1

CURSO 2022-2023

13 de Marzo de 2023

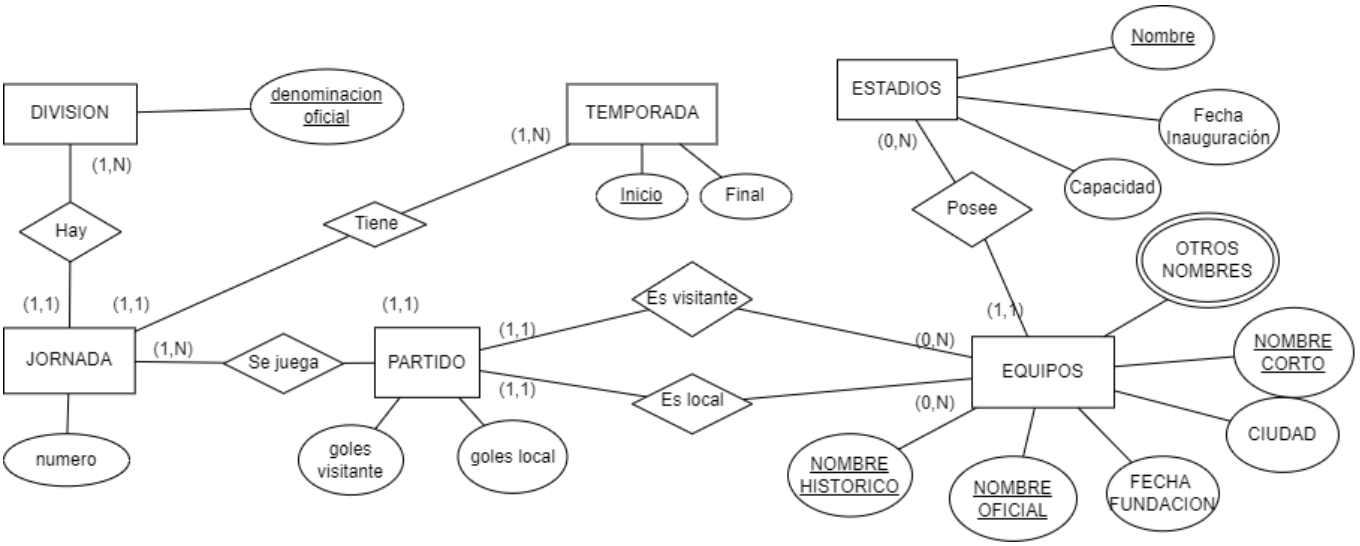


Alejandro Benedi Andrés - 843826@unizar.es

Javier Julve Yubero - 840710@unizar.es

Álvaro de Francisco Nievas - 838819@unizar.es

Parte 1: Creación de una base de datos



Entidad *Partido*:

- Se considerará ganador del partido al equipo que tenga más goles a favor, y sumará tres puntos. En caso de empate a goles, cada equipo obtendrá un punto.
- En un partido solo podrán jugar 2 equipos.
- Un partido solo podrá pertenecer a una jornada.
- El equipo local y visitante no puede ser el mismo

Entidad *Equipo*:

- Tanto nombre histórico y nombre oficial son claves candidatas.
- Un equipo tan solo tendrá un estadio, el actual.
- Dos equipos no pueden tener el mismo nombre oficial.
- Un equipo no puede jugar más de un partido en la misma jornada.
- Dos equipos no podrán tener el mismo nombre corto.

Entidad *Division*:

- No puede haber 2 divisiones con la misma denominación oficial
- Un equipo jugará únicamente en una división cada temporada.

Entidad *Estadio*:

- No puede haber 2 estadios con el mismo nombre.
- En un estadio pueden jugar varios equipos.

Soluciones alternativas:

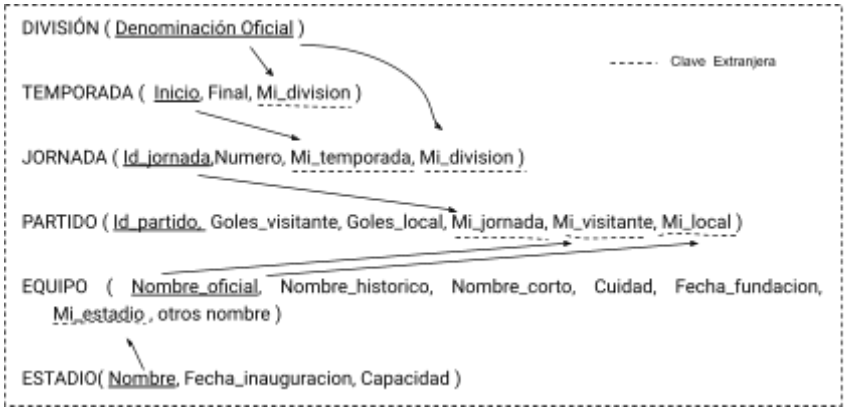
Valoramos la posibilidad de División se relacionara con Temporada, pero nos decidimos por relacionarla con Jornada para facilitar la comprensión.

Tampoco hemos puesto una relación directa entre Estadio y Partido, debido a que al poderse consultar el equipo local del partido mediante otra relación, se sabe por extensión el estadio.

Si un equipo cambia, su estadio también lo hará, pero dado a que no se pide en el enunciado y en el .csv no se prevé esta situación, hemos decidido dejarlo así para evitar redundancia.

Modelo relacional

Tras una traducción literal del entidad relacion tenemos el siguiente modelo relacional:



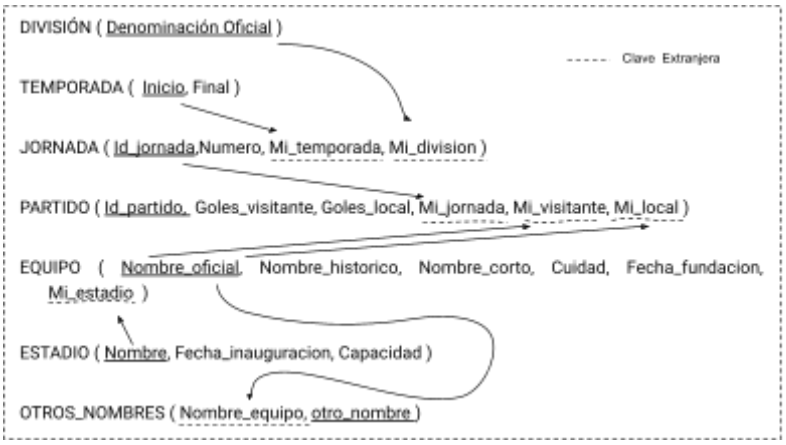
Normalización

El atributo *otros_nombre* es un atributo multivaluado que no tiene un número fijo de atributos. Por ello será necesario crear una relación independiente que permita guardar todos los otros nombres que pueda tener de equipo.

Se va a comprobar si el modelo tiene algún otro tipo de dependencia:

- En la relación DIVISIÓN tan solo hay un atributo y al ser único es clave primaria.
- En la relación TEMPORADA, Inicio será la clave primaria ya que es única.
- En JORNADA hemos añadido una clave primaria artificial, y esta relación tendrá 2 claves extranjeras: Mi_temporada y Mi_division.Es imposible que haya algún tipo de dependencia entre el resto de atributos.
- En la relación EQUIPO tenemos varias claves candidatas: *Nombre_oficial*, *Nombre_historico* y *Nombre_corto*. Hemos decidido que la clave primaria sea *Nombre_corto* y entre *Ciudad*, *Fecha_fundación* y *Mi_estadio* (clave extranjera) no haya ningún tipo de dependencia.
- En la relación ESTADIO, la clave primaria es Nombre, es imposible que haya una dependencia entre la *Fecha_inauguracion* y *capacidad*.

Todas las dependencias se basan en la clave primaria, por lo tanto el siguiente modelo relacional cumple con la forma forma normal de Boyce-Codd (FNBC):



El esquema ha sido modificado para cumplir con la primera forma normal (1FN). Sin embargo, como resultado de esta modificación, también cumple con la segunda forma normal (2FN) porque no hay dependencias funcionales que involucren ninguna parte de la clave. Además, el hecho de que no haya dependencias funcionales transitivas significa que cumple con la tercera forma normal (3FN).

Creación de las tablas

En SQL el orden de creación de las tablas es importante, por ello para poder crear nuestra base de datos, y posteriormente poblarla, es necesario que se ejecuten en orden los siguientes comandos:

```
CREATE TABLE Divisiones (
    den_oficial     VARCHAR(7)      CONSTRAINT PK_Div PRIMARY KEY
);
CREATE TABLE Temporadas (
    fecha_inicio    NUMBER(4)       CONSTRAINT PK_Temp PRIMARY KEY,
    fecha_fin       NUMBER(4),
    CONSTRAINT CK_Temp CHECK (fecha_inicio = fecha_fin-1)
);
CREATE TABLE Jornadas (
    id_jor          VARCHAR(8)   CONSTRAINT PK_Jor    PRIMARY KEY,
    num_jor         NUMBER(2),  -- hasta 99 jornadas (42 max)
    mi_temp         NUMBER(4),
    mi_div          VARCHAR(7),
    CONSTRAINT FK_Temp_Div FOREIGN KEY (mi_div) REFERENCES Divisiones(den_oficial),
    CONSTRAINT FK_Jor_Temp FOREIGN KEY (mi_temp) REFERENCES Temporadas(fecha_inicio)
);
CREATE TABLE Estadios (
    nom_est         VARCHAR(50)  CONSTRAINT PK_Est  PRIMARY KEY,
    capacidad       NUMBER(6),
    fecha_inag      NUMBER(4),
    CONSTRAINT CK_Est CHECK (capacidad > 0)
);
CREATE TABLE Equipos (
    nom_corto       VARCHAR(50) CONSTRAINT PK_Equ   PRIMARY KEY,
    nom_oficial     VARCHAR(50),
    nom_his         VARCHAR(50),
    ciudad          VARCHAR(50),
    fecha_fund      NUMBER(4),
    mi_estadio      VARCHAR(50),
    CONSTRAINT FK_Equ_Est FOREIGN KEY (mi_estadio) REFERENCES Estadios(nom_est)
);
CREATE TABLE Otros_nombres (
    otro_nombre         VARCHAR(50) CONSTRAINT PK_Otros_Nom PRIMARY KEY,
    nom_corto           VARCHAR(50),
    CONSTRAINT FK_Otros_Nom_Equ FOREIGN KEY (nom_corto) REFERENCES Equipos(nom_corto)
);
CREATE TABLE Partidos (
    id_par          VARCHAR(5)   CONSTRAINT PK_Par    PRIMARY KEY,
    goles_local     NUMBER(2)    CONSTRAINT NN_goles_local NOT NULL,
    goles_visit     NUMBER(2)    CONSTRAINT NN_goles_visit NOT NULL,
    mi_jor          VARCHAR(4),
    equ_local       VARCHAR(50),
    equ_visit       VARCHAR(50),
    CONSTRAINT FK_Par_Jor FOREIGN KEY (mi_jor) REFERENCES Jornadas(id_jor),
    CONSTRAINT FK_Par_Equ_Local FOREIGN KEY (equ_local) REFERENCES Equipos(nom_corto),
    CONSTRAINT FK_Par_Equ_Visit FOREIGN KEY (equ_visit) REFERENCES Equipos(nom_corto)
   CONSTRAINT CK_Par CHECK ((equ_local <> equ_visit) AND (goles_local >= 0) AND (goles_visit >= 0))
);
```

Población de la base de datos

En el .csv de la práctica se encuentre toda la información necesaria para poblar, sin embargo toda ella está desordenada y se nos hizo muy difícil poblar la base de datos con ese .csv.Por lo tanto la solución fue hacer 7 .csv diferentes (una por cada tabla) en los cuales cada columna del .csv será un tipo de datos de la tabla SQL.

Una vez realizados los 7 ficheros .csv se ha utilizado la aplicación SQL *Loader de ORacle (sqlldr2) para poblar toda la base de datos a partir de estos 7 csv. Para ello es necesario implementar unos ficheros .ctl (ficheros de control) en los cuales se especifican el nombre del fichero .csv donde se van a extraer los datos, el nombre de la tabla destino, el carácter separador (generalmente una coma) y los datos que se van a introducir (columnas de la tabla) junto con el tipo de datos que son (char, number, etc).

Este proceso fue bastante largo y nos costó varios días de trabajo, no por tener que realizar 7 ficheros .csv sino porque una serie de problemas que se van a comentar a continuación:

1. La aplicación SQL *Loader de ORacle (sqlldr2) leía los ficheros .csv en tipo char.Esto provocó que al insertar los datos en las diferentes tablas, se produjeran errores en los tipos de datos que eran diferentes. Este problema se agravaba más si las claves primarias y/o clave extranjeras no eran de tipo “char”. Para solucionar esto se usó TRAILING NULLCOL seguido del tipo de dato: decimal external , char, etc.

2. Al realizar el .ctl (fichero de control) y ejecutarlo muchas veces en la salida mostraba que se había insertado 0 líneas. Tras investigar un buen rato el porqué de este error nos dimos cuenta que el problema era el carácter de salto de línea.Los ficheros .csv se realizaron en windows por ello al exportarlos, el carácter de salto de línea era el correspondiente a Windows y no a Linux. Este error se soluciono fácilmente cambiando cada uno de los ficheros .csv a formato Linux.

3. Al igual que para crear las tablas, para poblar el orden también es muy importante.Al principio del proceso intentamos poblar la tabla PARTIDOS, que tiene varias claves extranjeras, sin poblar ninguna otra tabla antes de este. Esto provocaba un error debido a que no se encontraban las claves primarias en la tabla a la que se hacía referencia.Por ello para solucionar el error era necesario seguir un orden determinado muy parecido al usado para crear las tablas y asegurarse de que antes de poblar una tabla con claves extranjeras, se pueblen las tablas a la que hacen referencia.

4. El fichero .csv que referente a las Divisiones tan solo consta de 4 filas de una columna únicamente. Al ser una tabla de solo una columna las filas no tienen comas y tan solo terminan en un salto de linea.Al no poder separar por comas, la aplicación SQL *Loader únicamente insertaba en la tabla el primer carácter de cada fila.Para solucionar esto tuvimos que indicar en el archivo .ctl qué tipo de dato a leer sea CHAR(7)

Consultas

1. Equipo que más ligas de primera división ha ganado

```
 -- Calcula el equipo que mas ligas ha ganado usando la consulta anterior
SELECT C1.nom_eq AS Equipo, COUNT(*) AS LIGAS_GANADAS
FROM Clasificacion C1, (
    SELECT C2.temp, MAX(C2.puntos) AS puntos_primero
    FROM Clasificacion C2
    GROUP BY C2.temp) C3
WHERE C1.temp = C3.temp AND
      C1.puntos = C3.puntos_primero
GROUP BY C1.nom_eq
ORDER BY COUNT(*) DESC
FETCH FIRST 1 ROWS ONLY;
```

Para esta consulta usaremos una vista Clasificación, que sumará los puntos conseguidos en cada temporada de la primera división de un equipo.

Para calcular el equipo que más ligas ha ganado , es necesario saber primero el el vencedor de cada temporada,Para ello calculamos el maximo de puntos de cada temporada y luego, averiguamos a qué equipo le corresponden esos puntos.

Una vez hecho esto se cuenta el número de veces que un equipo ha ganado cada temporada.La salida se trunca a la primera fila

Salida:

EQUIPO	LIGAS_GANADAS
Real Madrid	19

2. **Listar estadios en los que el local ha ganado o empatado más del 85% de las veces**

```
-- Calcula que equipos han ganado mas del 85% de los partido y muetra el
   estadio
SELECT DISTINCT EQU_LOCAL, MI_ESTADIO
FROM PARTIDOS, EQUIPOS, par_gan, par_jug
WHERE EQU_LOCAL = NOM_CORTO AND num_gan > (0.85 * num_jug) AND
   par_jug.nom_eq = EQU_LOCAL AND par_gan.nom_eq = EQU_LOCAL
    AND par_jug.nom_eq = par_gan.nom_eq AND NOM_CORTO = par_jug.nom_eq AND
   NOM_CORTO = par_gan.nom_eq
ORDER BY EQU_LOCAL ASC;
```

Usamos dos vistas, par_jug y par_gan. La primera calcula todos los partidos que ha jugado cada equipo, independientemente de la división en la que juegan. Para ello contamos el todos los partidos en el cual en los cuales el local ha jugado. La vista resultante llamada par_jug tiene en la primera columna el nombre de todos los equipos locales y en la segunda tiene el número.

La segunda vista, par_gan, calcula el número de partidos que cada equipo ha ganado.

Una vez calculadas las 2 vistas anteriores, vamos a listar el número de estadios que hayan ganado más del 85% por ciento de las veces.Para ello tenemos que hacer “JOIN” de las 2 vistas creadas anteriormente y de las tablas PARTIDOS y EQUIPOS. Una vez hecho esto ya podemos comprobar fácilmente si la tasa de victorias de cada equipo es mayor del 85%. Tras ello la consulta mostrará los equipos con sus respectivos estadios que cumplan que la tasa de victorias es mayor del 85%.

Salida:

EQU_LOCAL	MI_ESTADIO
Almer??a (A.D.)	
Barakaldo	Nuevo Lasesarre
Barcelona	Camp Nou
Orihuela	
Real Madrid	Santiago Bernab??u
Sant Andreu	Narc??s Sala

3. Número de goles marcados por el Real Zaragoza en cada temporada de liga en la que haya ganado al menos a 4 equipos en ambos partidos de la temporada (ida y vuelta)

Para esta consulta hemos hecho 6 vistas distintas, con las que hacemos la consulta final.

Las dos primeras vistas muestran los equipos a los que el Real Zaragoza ha ganado como visitante y como local, respectivamente; y la temporada del partido ganado. Para ello seleccionamos de la tabla Partidos y Jornadas los partidos en

```
CREATE VIEW equ_gan_local (temp, nom_eq) AS
    SELECT MI_TEMP, EQU_LOCAL
    FROM PARTIDOS, JORNADAS
    WHERE MI_JOR = ID_JOR AND EQU_VISIT = 'Zaragoza' AND GOLES_LOCAL <
GOLES_VISIT
    ORDER BY MI_TEMP ASC;

CREATE VIEW equ_gan_visit (temp, nom_eq) AS
    SELECT MI_TEMP, EQU_VISIT
    FROM PARTIDOS, JORNADAS
    WHERE MI_JOR = ID_JOR AND EQU_LOCAL = 'Zaragoza' AND GOLES_LOCAL >
GOLES_VISIT
    ORDER BY MI_TEMP ASC;
```

La siguiente vista utiliza las dos anteriores y muestra los equipos que aparecen en ambas tablas en una misma temporada.

```
CREATE VIEW equ_gan_2_veces (temp, nom_eq) AS
    SELECT L.temp, L.nom_eq
    FROM equ_gan_visit V, equ_gan_local L
    WHERE V.nom_eq = L.nom_eq AND V.temp = L.temp
    ORDER BY temp ASC;a
```

La cuarta vista cuenta sobre la vista anterior el nº de equipos a los que ha ganado dos veces, y muestra en la tabla la temporada y el nº calculado a continuación.

```
CREATE VIEW tot_eq_gan(temp, num_eq) AS
    SELECT temp, COUNT(*)
    FROM equ_gan_2_veces
    GROUP BY temp
    ORDER BY temp ASC;
```

Las dos siguientes vistas calculan el número de goles que ha marcado el Zaragoza como local y como visitante en cada temporada de las que ha ganado a 4 o más equipos como local y como visitante.

```
CREATE VIEW goles_local(temp, num_goles_local) AS
    SELECT MI_TEMP, SUM(GOLES_LOCAL)
    FROM PARTIDOS, JORNADAS, tot_eq_gan
    WHERE MI_JOR = ID_JOR AND MI_TEMP = temp AND EQU_LOCAL = 'Zaragoza'
        AND num_eq >= 4
    GROUP BY MI_TEMP
    ORDER BY MI_TEMP ASC;

CREATE VIEW goles_visit(temp, num_goles_visit) AS
    SELECT MI_TEMP, SUM(GOLES_VISIT)
    FROM PARTIDOS, JORNADAS, tot_eq_gan
    WHERE MI_JOR = ID_JOR AND MI_TEMP = temp AND EQU_VISIT = 'Zaragoza'
        AND num_eq >= 4
    GROUP BY MI_TEMP
    ORDER BY MI_TEMP ASC;
```

Por último juntamos las temporadas y sumamos los goles de ambas tablas para cada temporada.

```
SELECT L.temp, SUM(num_goles_local + num_goles_visit) GOLES_TOTALES
FROM goles_local L, goles_visit V
WHERE L.temp = V.temp
GROUP BY L.temp
ORDER BY temp ASC;
```

Salida:

TEMP	GOLES_TOTALES
1982	59
1985	51
1998	57
2002	54
2008	79

Triggers

Con la base de datos creada es necesario la creación de ciertos triggers para impedir al usuario introducir nuevos datos erróneos, ya que sino se podría comprometer el uso de la base de datos y todo lo creado anteriormente.

De la forma en la que la tenemos montada solo necesitábamos impedir un par de casos, por lo que hemos creado un trigger para cada uno.

Los casos que hemos valorado son:

1. Dos equipos no pueden poseer el mismo nombre oficial.
2. Un equipo no puede jugar dos veces o más en la misma jornada.
3. Un equipo no puede jugar contra sí mismo (ser local y visitante a la vez).

El primer trigger impide que haya dos equipos con el mismo nombre oficial, ya que en la realidad no se puede y además no debemos preocuparnos por el nombre corto, ya que es una clave primaria y Oracle se encargaría de eso. Además con este trigger las consultas hacia un equipo concreto podrían ir sin fallos ya que de lo contrario no podríamos buscar con exactitud el que buscamos, por la mezcla de ellos.

```
CREATE OR REPLACE TRIGGER Duplicado_nombre
BEFORE INSERT ON Equipos
FOR EACH ROW
DECLARE
filas NUMBER;
BEGIN

SELECT count(*) INTO filas FROM Equipos WHERE nom_oficial=:new.nom_oficial;

if (filas > 0)
then raise_application_error(-20002,
'Dos equipos no pueden tener el mismo nombre oficial');
end if;
END Duplicado_nombre;
```

El segundo trigger está destinado a que en una misma jornada un equipo no juegue dos veces, ya que al solo poder jugar una vez, podría causar problemas con la consulta 1, pues está pensada bajo esas condiciones y las clasificaciones de puntos se verían alteradas.

```
CREATE OR REPLACE TRIGGER Duplicado_partido
BEFORE INSERT ON Partidos
FOR EACH ROW
DECLARE
```

```
filas NUMBER;
BEGIN

SELECT count(*) INTO filas FROM Partidos WHERE mi_jor=:new.mi_jor
AND (equ_local = :new.equ_local OR equ_visit= :new.equ_visit OR
equ_visit = :new.equ_local OR equ_local = :new.equ_visit);

if (filas > 0)
then raise_application_error(-20002,
'No puedes jugar dos veces una misma jornada');
end if;
END Duplicado_partido;
/
```

El tercer y último trigger que hemos desarrollado evita que un mismo equipo juegue como local y visitante, o lo que es lo mismo que juegue contra sí mismo, ya que pese a que en la realidad no es físicamente posible, en la base de datos se nos presenta otro problema adicional además de las consultas y es que al no haber penalización por perder en el sentido de quitar puntos (solo no los ganas), podría ser que el equipo gane puntos de forma fraudulenta mediante una mala inserción de victoria o mediante empate, lo cual crearía resultados falsos y afectaría a la clasificación y su respectiva consulta, ya sea una que hagamos o una solicitada por el enunciado.

```
CREATE OR REPLACE TRIGGER Mismos_equipos
BEFORE INSERT ON Partidos
FOR EACH ROW
DECLARE
filas NUMBER;
BEGIN

SELECT count(*) INTO filas FROM Partidos WHERE :new.equ_local =
:new.equ_visit;

if (filas > 0)
then raise_application_error(-20002,
'El equipo visitante y local no pueden ser el mismo');
end if;
END Mismos_equipos;
/
```

Problemas de rendimiento:

Segunda consulta

```
-----------------------------------------------------------------------------------------------
| Id  | Operation             | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |          |     1 |   161 |   213   (3)| 00:00:01 |
|   1 |  SORT ORDER BY        |          |     1 |   161 |   213   (3)| 00:00:01 |
|   2 |   HASH UNIQUE         |          |     1 |   161 |   212   (3)| 00:00:01 |
|*  3 |    HASH JOIN SEMI     |          |     1 |   161 |   211   (2)| 00:00:01 |
|*  4 |     HASH JOIN         |          |    11 |  1474 |   143   (3)| 00:00:01 |
|   5 |      TABLE ACCESS FULL| EQUIPOS  |   105 |  5670 |     3   (0)| 00:00:01 |
|*  6 |      HASH JOIN        |          |  1196 | 95680 |   140   (3)| 00:00:01 |
|   7 |       VIEW            | PAR_GAN  | 23916 |   934K|    70   (3)| 00:00:01 |
|   8 |        HASH GROUP BY  |          | 23916 |  1237K|    70   (3)| 00:00:01 |
|*  9 |         TABLE ACCESS FULL| PARTIDOS | 23916 |  1237K|    68   (0)| 00:00:01 |
|  10 |       VIEW            | PAR_JUG  | 30965 |  1209K|    70   (3)| 00:00:01 |
|  11 |        HASH GROUP BY  |          | 30965 |   816K|    70   (3)| 00:00:01 |
|  12 |         TABLE ACCESS FULL| PARTIDOS | 30965 |   816K|    68   (0)| 00:00:01 |
|  13 |      TABLE ACCESS FULL| PARTIDOS | 30965 |   816K|    68   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

Tercera consulta

```
-----------------------------------------------------------------------------------------------
| Id  | Operation          | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |          |   139 | 14595 |    74   (2)| 00:00:01 |
|   1 |  SORT ORDER BY     |          |   139 | 14595 |    74   (2)| 00:00:01 |
|*  2 |   HASH JOIN        |          |   139 | 14595 |    73   (0)| 00:00:01 |
|*  3 |    TABLE ACCESS FULL| PARTIDOS |   139 | 11954 |    68   (0)| 00:00:01 |
|   4 |    TABLE ACCESS FULL| JORNADAS | 3398 | 64562 |     5   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

```
-----------------------------------------------------------------------------------------------
| Id  | Operation              | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |            |   838 | 43576 |   445   (2)| 00:00:01 |
|   1 |  MERGE JOIN            |            |   838 | 43576 |   445   (2)| 00:00:01 |
|   2 |   SORT JOIN            |            |   194 |  5044 |   222   (2)| 00:00:01 |
|   3 |    VIEW                | GOLES_VISIT|   194 |  5044 |   222   (2)| 00:00:01 |
|   4 |     HASH GROUP BY      |            |   194 | 15132 |   222   (2)| 00:00:01 |
|*  5 |      HASH JOIN         |            |   194 | 15132 |   221   (1)| 00:00:01 |
|   6 |       VIEW             | TOT_EQ_GAN |    12 |   156 |   148   (2)| 00:00:01 |
|*  7 |        HASH GROUP BY   |            |    12 |  2520 |   148   (2)| 00:00:01 |
|*  8 |         HASH JOIN      |            |    12 |  2520 |   147   (1)| 00:00:01 |
|   9 |          TABLE ACCESS FULL | JORNADAS   |  3398 | 64562 |     5   (0)| 00:00:01 |
|* 10 |          HASH JOIN     |            |   528 |   98K|   142   (1)| 00:00:01 |
|* 11 |           HASH JOIN    |            |   139 | 14595 |    73   (0)| 00:00:01 |
|* 12 |            TABLE ACCESS FULL| PARTIDOS   |   139 | 11954 |    68   (0)| 00:00:01 |
|  13 |            TABLE ACCESS FULL| JORNADAS   |  3398 | 64562 |     5   (0)| 00:00:01 |
|* 14 |           TABLE ACCESS FULL | PARTIDOS   |   352 | 30272 |    68   (0)| 00:00:01 |
|* 15 |       HASH JOIN        |            |   713 | 46345 |    73   (0)| 00:00:01 |
|* 16 |        TABLE ACCESS FULL   | PARTIDOS   |   713 | 32798 |    68   (0)| 00:00:01 |
|  17 |        TABLE ACCESS FULL   | JORNADAS   |  3398 | 64562 |     5   (0)| 00:00:01 |
|* 18 |   SORT JOIN            |            |   190 |  4940 |   223   (2)| 00:00:01 |
|  19 |    VIEW                | GOLES_LOCAL|   190 |  4940 |   222   (2)| 00:00:01 |
|  20 |     HASH GROUP BY      |            |   190 | 14820 |   222   (2)| 00:00:01 |
|* 21 |      HASH JOIN         |            |   190 | 14820 |   221   (1)| 00:00:01 |
|  22 |       VIEW             | TOT_EQ_GAN |    12 |   156 |   148   (2)| 00:00:01 |
|* 23 |        HASH GROUP BY   |            |    12 |  2520 |   148   (2)| 00:00:01 |
```

```
|* 24 |       HASH JOIN          |           |   12 |  2520 |   147   (1)| 00:00:01 |
|  25 |        TABLE ACCESS FULL  | JORNADAS  | 3398 | 64562 |     5   (0)| 00:00:01 |
|* 26 |        HASH JOIN          |           |  528 |    98K|   142   (1)| 00:00:01 |
|* 27 |         HASH JOIN         |           |  139 | 14595 |    73   (0)| 00:00:01 |
|* 28 |          TABLE ACCESS FULL| PARTIDOS  |  139 | 11954 |    68   (0)| 00:00:01 |
|  29 |          TABLE ACCESS FULL| JORNADAS  | 3398 | 64562 |     5   (0)| 00:00:01 |
|* 30 |         TABLE ACCESS FULL | PARTIDOS  |  352 | 30272 |    68   (0)| 00:00:01 |
|* 31 |       HASH JOIN           |           |  697 | 45305 |    73   (0)| 00:00:01 |
|* 32 |        TABLE ACCESS FULL  | PARTIDOS  |  697 | 32062 |    68   (0)| 00:00:01 |
|  33 |        TABLE ACCESS FULL  | JORNADAS  | 3398 | 64562 |     5   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------------------------------
```

Tal y como se aprecia en las tablas y por el modelo realizado en las consultas, se puede apreciar que como cabe esperar el tiempo de ejecución y los recursos empleados para ello son minúsculos, sin embargo hay ciertas tablas (las creadas con el view) y los hashes redirigidos hacia estas mismas tablas también ocupa mucho, esto se produce debido a que con los views realmente lo que se hace es crear una tabla adicional donde guardar ciertos datos para luego ser empleados y pese a que son tablas temporales, el espacio de memoria gastado está ahí al menos hasta que se borren dichas tablas, además hay ciertos casos como en la primera consulta que son tan grandes que no se han podido poner aquí, además de que el tiempo de ejecución de dicha consulta también era bastante elevado, sin embargo si bien con la memoria se podría solucionar un poco el problema, con el tiempo es imposible pues debe recorrer y seleccionar los datos establecidos para guardarlos lo cual hace que su optimización sea posible pero en la práctica bastante más complicado y tampoco cambiaría en gran medida el tiempo de ejecución.

Conclusiones

Horas trabajadas:

TABLA DE HORAS DE TRABAJO	ALEJANDRO BENEDÍ	ÁLVARO DE FRANCISCO	JAVIER JULVE
PARTE I	4H	4H	4H
PARTE II	7H	5H	6H
PARTE III	8H	8H	8H

División del trabajo:

Al dividirnos el trabajo, acordamos dividir equitativamente las tareas para

Dificultades:

Nos encontramos con problemas de desconocimiento al principio de la práctica, ya que no sabíamos cómo afrontar la creación de las tablas y de la base de datos en general.

Para realizar la primera consulta, tuvimos problemas con la tabla Clasificacion, que enfocamos de una forma equivocada.