Diseño e implementación de una base de datos relacional para la gestión de apuestas de fútbol

Universitat Oberta de Catalunya

Ingeniería Informática
PFC – Bases de Datos

Alumno
David Cuenca Aznar

Consultor
Juan Martínez Bolaños

Memoria
15/06/2014

A mis padres,
porque me enseñaron el camino.

A Belén,
porque siempre está ahí para darme su mano
cuando la oscuridad me impide seguir adelante.

"La más larga caminata comienza con un paso"

Proverbio Hindú

1 Índice

1.1 Índice de contenidos

1.2 Índice de Figuras

2 Introducción

El proyecto de fin de carrera consiste en un trabajo en el que aplicar de forma conjunta las competencias adquiridas en diferentes asignaturas de la carrera de Ingeniería Informática.

Concretamente en el área de Bases de Datos se trata de aplicar estos conocimientos para diseñar e implementar una solución (en este caso una base de datos) que cumpla con los requisitos que se nos plantean.

En este documento se describen a grandes rasgos las características del proyecto a desarrollar, así como la metodología y el plan de trabajo previsto para su ejecución.

2.1 Objetivos

2.1.1 Objetivos Generales

- Poner en práctica los conocimientos adquiridos a lo largo de toda la carrera, concretamente en las siguientes asignaturas:
  - Bases de Datos II
  - Sistemas de Gestión de Bases de Datos
  - Modelos Multidimensionales y Almacenes de Datos
  - Ingeniería de Programación
  - Metodología y Gestión de Proyectos informáticos
- Demostrar que estamos capacitados para gestionar y dirigir proyectos de cierta envergadura y llevarlos a cabo con éxito.
- Aprender a utilizar la programación de Bases de Datos mediante el lenguaje PL/SQL
- Administrar y Gestionar el Sistema de Gestión de Bases de Datos de *Oracle*

2.1.2 Objetivos particulares

Se nos pide diseñar e implementar una base de datos (BD) para gestionar apuestas de fútbol que cumpla los siguientes requisitos:

- Almacenar datos deportivos: equipos y ligas de diferentes países, resultados de los partidos, nombre de los jugadores que han marcado goles en qué orden, etc. El sistema tiene que permitir gestionar más de una temporada, para poder ofrecer a los usuarios datos históricos de resultados, historial deportivo de cada jugador, goles marcados, etc.
- Gestionar apuestas: identificar el usuario, importe de la apuesta, modalidad (victoria o empate, jugador que marcará el primer gol, resultado final del partido, resultado al final de la primera parte, etc.)
- La BD deberá ser escalable para que se puedan incorporar progresivamente todas aquellas necesidades que puedan surgir durante su vida.
- Deberá definirse un almacén de datos *Datawarehouse* (DW) para extraer y consolidar la información, y obtener estadísticas como por ejemplo: la distribución geográfica de los usuarios, edad, inversiones por tramos de edad, equipos por los que se apuesta más, etc.
- Es recomendable disponer de mecanismos que permitan resolver posibles problemas de integración que puedan surgir con otras partes del sistema: un log de acciones realizadas, mecanismos para comprobar la funcionalidad, etc.

2.2 Metodologia de trabajo.

Para llevar a cabo el proyecto he decidido que la metodología que utilizaré será el ciclo de vida en cascada, iterativo e incremental, dado su sencillez.

De esta forma, se dividirá todo el proyecto en subtareas que se irán realizando de forma iterativa, una tras otra, de forma que se vaya evolucionando hasta obtener el resultado final. Para poder iniciar una tarea será necesario haber terminado la tarea anterior, aunque es posible que durante la vida del proyecto haya que añadir o modificar requisitos o el diseño, lo que obligará a revisar de nuevo el camino ya recorrido para adaptar la solución a las modificaciones introducidas:

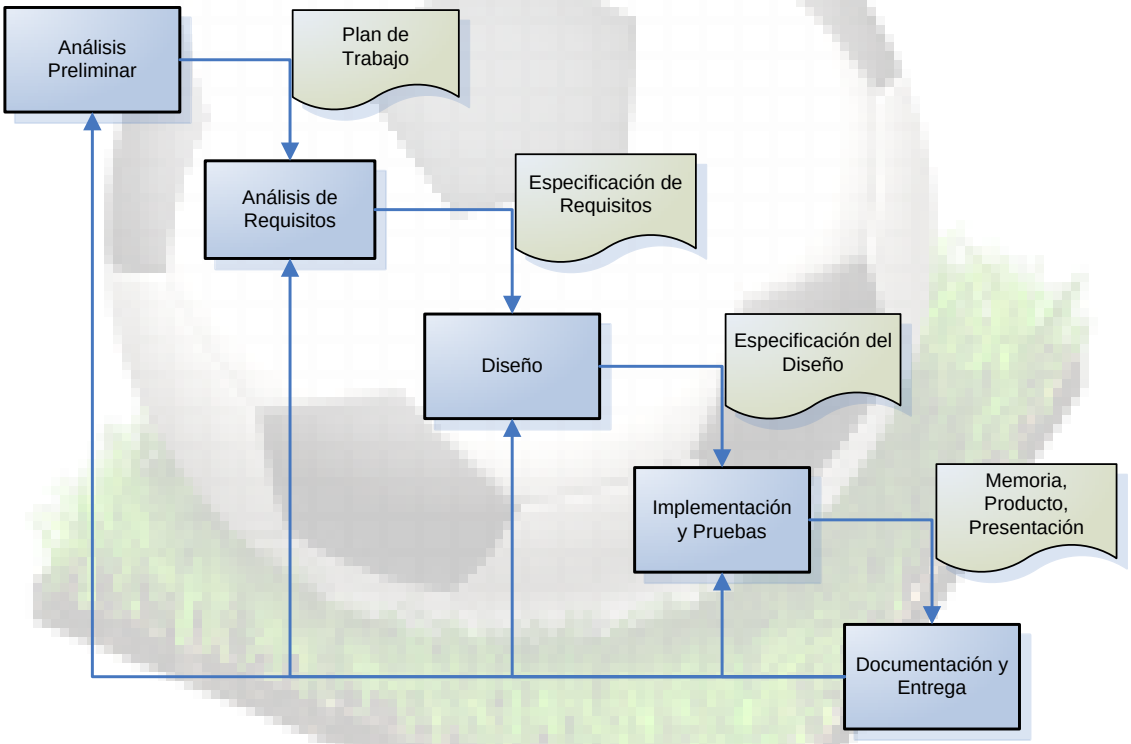


Ilustración 1: Ciclo de vida en cascada.

Después de la entrega, para completar el ciclo de vida en cascada, faltaría por implementar la etapa de mantenimiento, pero esto ya quedará fuera del alcance del proyecto actual.

2.3 Tareas a realizar. Alcance del proyecto:

Para llevar alcanzar los objetivos planteados, y siguiendo la metodología propuesta, estas son las tareas a realizar:

- **Planificación de Tareas**: Definición lo más detallada posible de cómo se desarrollará el proyecto. Esta planificación pretende ser la hoja de ruta que nos guiará hasta la finalización del proyecto.
- **Análisis de Requisitos**: Aquí haremos una recopilación de las tareas a realizar y que nos son exigidas por el cliente para cubrir sus necesidades
- **Diseño e Implementación**

  Se definirán todas las funcionalidades del producto. Hemos separado según vamos a implementar la Base de Datos Operacional y la de Obtención de Estadísticas:
  - Base de Datos Operacional
    - Diseño Conceptual (Diagrama entidad-relación)
    - Diseño Lógico
    - Diseño Físico (Scripts de Creación de los objetos de la base de datos).
    - Implementación de procedimientos almacenados para la manipulación de los datos
  - Base de Datos *Datawarehouse*
    - Diseño Conceptual (Multidimensional)
    - Diseño Lógico
    - Diseño Físico
    - Implementación de procesos de extracción, transformación y carga de datos (ETL).
- **Pruebas**

  Definiremos un plan de pruebas que permitan comprobar el correcto funcionamiento de los desarrollos realizados, en todas sus funcionalidades. Durante la fase de Implementación iremos realizando pruebas unitarias de los procedimientos que se vayan implementando.

  - Obtención de datos de prueba
  - Obtención de estadísticas a partir de los datos introducidos en las pruebas con *Pentaho BI*

- **Documentación**

  Una vez tengamos el producto terminado nos quedará recopilar toda la documentación que hayamos ido generando en el transcurso del proyecto, y prepararla para la entrega final:

  - Memoria (Recopilación de todo el trabajo realizado)
  - Presentación (Resumen del trabajo)
  - Producto entregable (Código fuente de los trabajos realizados)

2.4 Fechas clave

Las fechas clave del proyecto coinciden con las entregas parciales definidas en el enunciado del proyecto. En cada una de estas entregas se añadirá un breve documento que explique el estado en que se encuentra el proyecto.

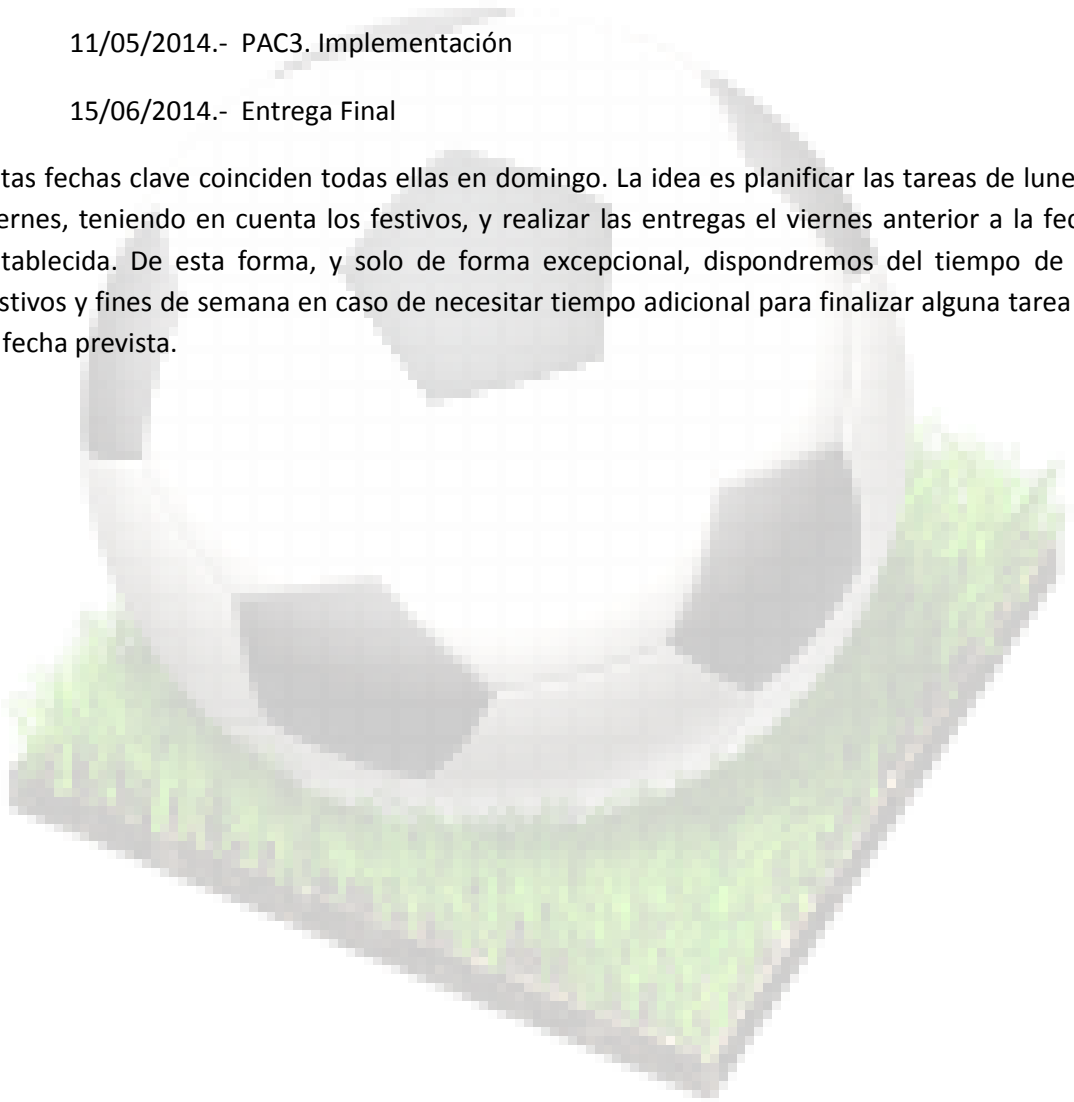
16/03/2014.- PAC1. Plan de Trabajo

13/04/2014.- PAC2. Diseño

11/05/2014.- PAC3. Implementación

15/06/2014.- Entrega Final

Estas fechas clave coinciden todas ellas en domingo. La idea es planificar las tareas de lunes a viernes, teniendo en cuenta los festivos, y realizar las entregas el viernes anterior a la fecha establecida. De esta forma, y solo de forma excepcional, dispondremos del tiempo de los festivos y fines de semana en caso de necesitar tiempo adicional para finalizar alguna tarea en la fecha prevista.



2.5 Planificación Temporal de las Tareas

Este es el desglose detallado de tareas a realizar, junto con la estimación temporal prevista para su realización:

Id	Nombre de tarea	Duración	Comienzo
1	**PFC Bases de Datos**	**71 días**	**lun 03/03/14**
2	**PAC1. Plan de Trabajo**	**10 días**	**lun 03/03/14**
3	Lectura y comprensión del enunciado de la práctica	3 días	lun 03/03/14
4	Planificación de Tareas	2 días	jue 06/03/14
5	Determinación de las herramientas y del software necesario	1 día	lun 10/03/14
6	Elaboración del Plan de Trabajo	4 días	mar 11/03/14
7	Entrega PAC1	0 días	dom 16/03/14
8	Instalación y configuración del Software	1 día	lun 17/03/14
9	**PAC2. Análisis y Diseño**	**18 días**	**mar 18/03/14**
10	Análisis de requisitos	2 días	mar 18/03/14
11	**Diseño de la Base de Datos Operacional**	**7 días**	**vie 21/03/14**
12	Diseño Conceptual	2 días	vie 21/03/14
13	Diseño Lógico	2 días	mar 25/03/14
14	**Diseño Físico**	**3 días**	**jue 27/03/14**
15	Implementación de la Base de datos Operacional	3 días	jue 27/03/14
16	Scripts de Creación de tablas, relaciones, triggers)	3 días	jue 27/03/14
17	**Diseño de la Base de Datos Datawarehouse**	**7 días**	**mar 01/04/14**
18	Diseño Conceptual	2 días	mar 01/04/14
19	Diseño Lógico	2 días	jue 03/04/14
20	**Diseño Físico**	**3 días**	**lun 07/04/14**
21	Implementación del DW	3 días	lun 07/04/14
22	Scripts de creación de los objetos de la base de datos DW"	3 días	lun 07/04/14
23	Documentación del estado del proyecto	2 días	jue 10/04/14
24	Entrega PAC2	0 días	dom 13/04/14
25	**PAC3. Desarrollo de Procedimientos Almacenados**	**17 días**	**lun 14/04/14**
26	**Base de Datos Operacional.**	**12 días**	**lun 14/04/14**
27	**Subsistema datos deportivos.**	**6 días**	**lun 14/04/14**
28	Creación de scripts	3 días	lun 14/04/14
29	Pruebas subsistema de datos deportivos	3 días	jue 17/04/14
30	**Subsistema gestión de Apuestas**	**6 días**	**jue 24/04/14**
31	Creación de Scripts	3 días	jue 24/04/14
32	Pruebas unitarias subsistema de Apuestas	3 días	mar 29/04/14
33	**Base de Datos DW**	**3 días**	**lun 05/05/14**
34	Codificación de procedimientos ETL	3 días	lun 05/05/14
35	Documentación del estado del proyecto	2 días	jue 08/05/14
36	Entrega PAC3	0 días	dom 11/05/14
37	**Pruebas**	**8 días**	**lun 12/05/14**
38	Generación y/o obtención de datos para realizar pruebas Globales	4 días	lun 12/05/14
39	Pruebas Globales	4 días	vie 16/05/14
40	Explotación del almacen de datos Datawarehouse	3 días	jue 22/05/14
41	**ENTREGA FINAL**	**14 días**	**mar 27/05/14**
42	Memoria	6 días	mar 27/05/14
43	Presentación Virtual	5 días	mié 04/06/14
44	Producto	3 días	mié 11/06/14
45	Entrega Final	0 días	dom 15/06/14

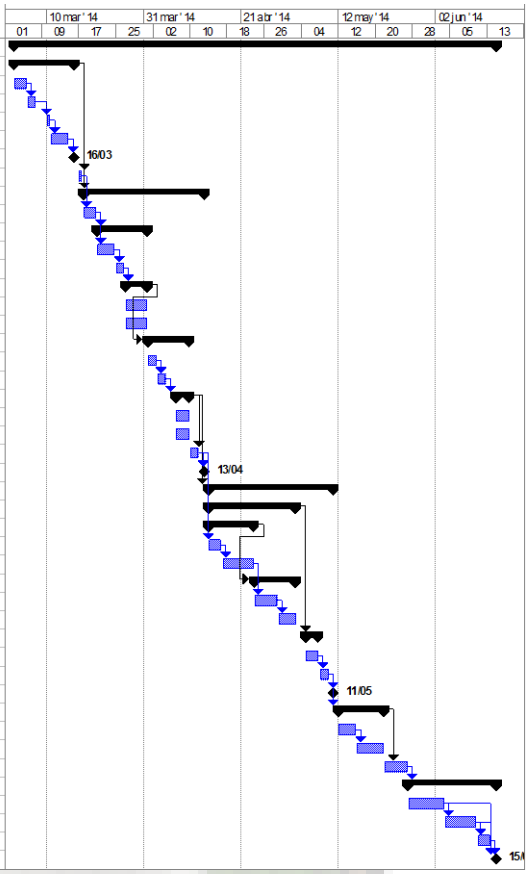


Ilustración 2: Planificación Temporal. Diagrama de Gantt

	PFC Bases de Datos	**71 días**
01	**PAC1. Plan de Trabajo**	**10 días**
01.01	Lectura y comprensión del enunciado de la práctica	3 días
01.02	Planificación de Tareas	2 días
01.03	Determinación de las herramientas y del software necesario	1 día
01.04	Elaboración del Plan de Trabajo	4 días
01.05	**Entrega PAC1**	**0 días**
02	**Instalación y configuración del Software**	**1 día**
03	**PAC2. Análisis y Diseño**	**18 días**
03.01	***Análisis de requisitos***	2 días
03.02	***Diseño de la Base de Datos Operacional***	7 días
03.02.01	Diseño Conceptual	2 días
03.02.02	Diseño Lógico	2 días
03.02.03	Diseño Físico	3 días
03.02.03.01	Implementación de la Base de datos Operacional Scripts de Creación de tablas, relaciones, *triggers*)	3 días
03.03	***Diseño de la Base de Datos Datawarehouse***	7 días
03.03.01	Diseño Conceptual	2 días
03.03.02	Diseño Lógico	2 días
03.03.03	Diseño Físico	3 días
03.03.03.01	Implementación del DW Scripts de creación de los objetos de la base de datos DW	3 días
03.04	Documentación del estado del proyecto	2 días
03.05	**Entrega PAC2**	**0 días**
04	**PAC3. Desarrollo de Procedimientos Almacenados**	**17 días**
04.01	***Base de Datos Operacional.***	12 días
04.01.01	Subsistema datos deportivos.	6 días
04.01.01.01	Creación de scripts	3 días
04.01.01.02	Pruebas subsistema de datos deportivos	3 días
04.01.02	Subsistema gestión de Apuestas	6 días
04.01.02.01	Creación de Scripts	3 días
04.01.02.02	Pruebas unitarias subsistema de Apuestas	3 días
04.02	***Base de Datos DW***	3 días
04.02.01	Codificación de procedimientos ETL	3 días
04.03	Documentación del estado del proyecto	2 días
04.04	**Entrega PAC3**	**0 días**
05	**Pruebas**	**8 días**
05.01	Generación y/o obtención de datos para realizar pruebas Globales	4 días
05.02	Pruebas Globales	4 días
06	**Explotación del almacén de datos Datawarehouse**	**3 días**
07	**ENTREGA FINAL**	**14 días**
07.01	Memoria	6 días
07.02	Presentación Virtual	5 días
07.03	Producto	3 días
07.04	**Entrega Final**	**0 días**

2.6 Evaluación del material necesario

2.6.1 Hardware:

Ordenador Personal con *Microsoft Windows XP Profesional 32 bits*,

Dispondré de otro equipo similar preparado (aunque de menores prestaciones) para minimizar el riesgo de sufrir pérdidas de información y posibles retrasos por averías en el puesto de trabajo.

2.6.2 Software:

Base de Datos

Oracle Database 11g Express Edition

Herramientas de Gestión y Modelado de Base de Datos

Toad for Oracle 12.1 Freeware

Toad Data Modeler 5.1

Explotación de Datos

Si el desarrollo del proyecto me lo permite, me gustaría poder incluir algún análisis de datos de Ejemplo usando alguna herramienta *Bussiness Intelligence*.

Pentaho BI Server 5.0.1.stable

En el peor de lo casos obtendría datos estadísticos directamente mediante consultas en Oracle

Herramienta para elaboración de Diagramas

Microsoft Visio 2003

Planificación de Proyectos

Microsoft Project 2003

Sincronización de ficheros entre equipos

Dropbox

2.7 Productos Obtenidos

El resultado de todo el trabajo realizado será el siguiente:

Plan de Trabajo: es el documento entregado en el primer plazo, y que detalla el guión seguido para alcanzar los objetivos del proyecto. En él se describen todas las tareas realizadas, junto con la estimación temporal prevista para llevarlas a cabo.

Producto: es el conjunto de ficheros que permiten crear los objetos de base de Datos:

- ***01-TABLESPACES.sql***: Se crean los ficheros en el sistema donde se guardarán las bases de datos
- ***02-USUARIOS.sql****:* Se crean dos usuarios, uno que será el propietario de los objetos de la BD Operacional, y otro de los objetos del DW
- ***03-CrearDB_OP.sql****:* Crea la Base de Datos Operacional
- ***04-CrearDB_DW.sql****:* Crea la Base de Datos Estadística
- ***05_PAQUETES.sql***: Crea los paquetes de manipulación y consulta de los Datos
- ***06-ETL.sql***: Procesos ETL
- ***07-DATOS_DE_PRUEBA.sql*** introduce Datos de ejemplo en la base de datos operacional para poder comprobar el funcionamiento de los procesos ETL, y la explotación de los datos.
- ***08-CARGA ETL.sql:*** Realiza el proceso de carga de datos del DW

Aparte de los scripts descritos anteriormente, también se incluye el fichero ESQUEMA_DW.xml, generado con Pentaho Schema Workbench, que permite realizar el análisis de la información con Saiku Analytics

Memoria: Descripción detallada de todo el trabajo realizado.

Presentación virtual: Presentación de dispositivas que resume todo el proyecto

2.8 Descripción del resto de Capítulos

Después de la introducción, el resto de este documento está compuesto por los siguientes capítulos:

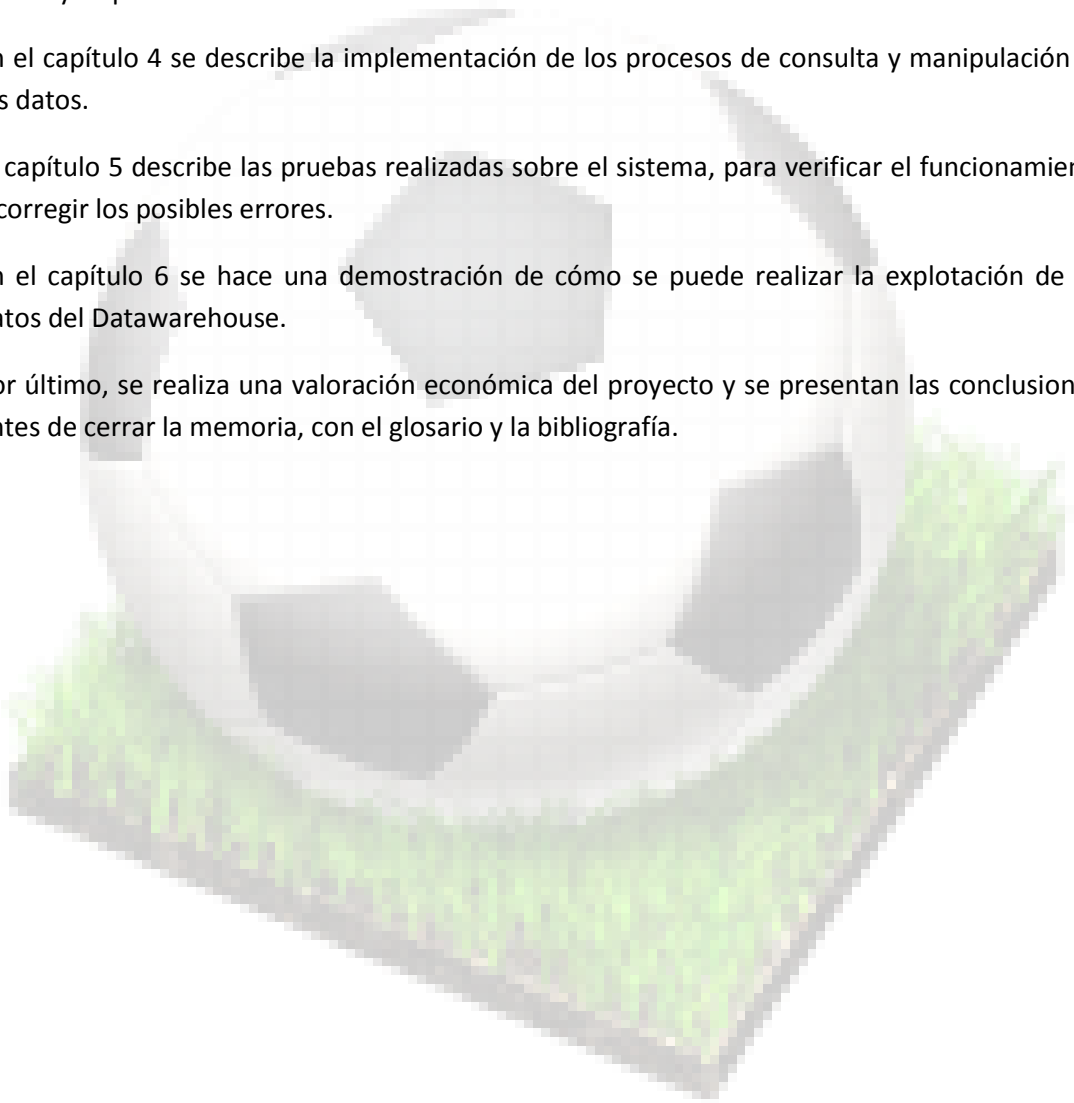
En el capítulo 3 se describe el análisis y diseño del sistema que se nos pide en el enunciado del proyecto. El resultado de esta fase son los scripts de creación de tablas de la base de datos operacional, y la base de datos estadística, junto con sus relaciones, claves primarias, claves ajenas y disparadores.

En el capítulo 4 se describe la implementación de los procesos de consulta y manipulación de los datos.

El capítulo 5 describe las pruebas realizadas sobre el sistema, para verificar el funcionamiento y corregir los posibles errores.

En el capítulo 6 se hace una demostración de cómo se puede realizar la explotación de los datos del Datawarehouse.

Por último, se realiza una valoración económica del proyecto y se presentan las conclusiones, antes de cerrar la memoria, con el glosario y la bibliografía.



3 Análisis y Diseño

Después de la planificación inicial, nos ponemos manos a la obra, y trataremos, de ahora en adelante, seguir lo máximo posible la planificación que acabamos de establecer.

En la medida que seamos capaces de cumplir la planificación será más probable que finalicemos el proyecto con éxito.

3.1 Especificación de Requisitos

La empresa en la que trabajamos nos pide desarrollar un sistema de gestión de apuestas de fútbol

Se desea que el producto obtenido cumpla los siguientes requisitos:

R1.- El sistema debe permitir almacenar datos deportivos

R1.1.- Se desean guardar datos de equipos y ligas de diferentes países, resultados de los partidos, jugadores, goles marcados /y en qué orden, etc.

R1.2.- El sistema debe gestionar más de una temporada.

R1.3- Se ofrecerán a los usuarios la posibilidad de consultar los datos históricos:

- R1.3.1.- Temporadas
  - R.1.3.1.1 - Mostrar las competiciones
- R.1.3.2 – Competiciones (Ligas)
  - R.1.3.2.1 – Mostrar los equipos que compiten en una temporada
  - R.1.3.2.2 – Mostrar los resultados de los partidos por jornadas
  - R.1.3.2.3 – Mostrar los partidos por jornadas, para que los usuarios puedan realizar apuestas
- R.1.3.3 - Equipos
  - R.1.3.3.1 – Mostrar las competiciones en las que ha participado en cada temporada
  - R.1.3.3.2 – Mostrar los resultados de los partidos en una competición y una temporada.
- R.1.3.4 - Partidos
  - R.1.3.4.1 – Mostrar información de los goles marcados
- R.1.3.5 - Jugadores
  - R.1.3.5.1 – Mostrar un histórico de equipos en los que ha jugado
  - R.1.3.5.2 – Mostrar un histórico de los goles que ha marcado

R2.- El sistema debe permitir la gestión de apuestas

R2.1.- Identificación de usuarios, importe apostado y modalidades de apuestas (victoria o empate, resultado de la primera parte y al final, jugador que marcará primero, etc.).

Según el tipo de apuesta el resultado de la misma podrá ser: un jugador, un equipo, un resultado, 1X2, etc.

R2.2.- Consulta del histórico de apuestas de un usuario

R3.- Se desea implementar un almacén de datos Datawarehouse (DW) que permita obtener datos estadísticos a partir de la información de los datos deportivos y de apuestas.

- R3.1 - Distribución geográfica de los usuarios y edad.
- R3.2 - Importe apostado por tramos de edad
- R3.3 – Equipos/jugadores por los que se apuesta más
- R3.4 - Modalidades con más apuestas
- R3.5 - Distribución geográfica de los usuarios que apuestan por liga, por equipo
- R3.6 - Competiciones con más apuestas

R4.-Toda la gestión de la información se realizará mediante procedimientos de la base de datos. Habrá que implementar un sistema que encapsule el acceso a los datos:

- R4.1 - Procedimientos de Alta, Modificación y Borrado (AMB) para poder manipular la información del sistema.
- R4.2 - Procedimientos de Consulta de datos deportivos
- R4.3 - Procedimientos de Consulta de apuestas por usuario
- R4.4 - Procedimientos de Extracción, Tratamiento y Carga de Datos (ETL) desde la Base de Datos Operacional al DW.

R5.- Se valorará la posibilidad de disponer de mecanismos de auditoría, que permitan resolver problemas de integración, y registrar todas las acciones realizadas.

Se guardará un LOG que registrará todos los procedimientos ejecutados sobre el sistema (fecha y hora, procedimiento ejecutado, parámetros y resultado).

R6.- El Sistema Gestor de Base de Datos (SGBD) sobre el que se implementará la solución será Oracle.

3.1.1 Requisitos Funcionales

3.1.1.1 Actores

A partir de los requisitos que nos ha pedido el cliente, podemos distinguir tres tipos de actores principales que interactuarán con el sistema:

Administrador de Datos Deportivos: Se encargarán del mantenimiento y la gestión de los datos deportivos: Temporadas, competiciones, equipos, jugadores, partidos, alineaciones, resultados, etc.

Administrador de Apuestas (supervisor): se encargará de revisar y supervisar la correcta gestión de las modalidades de apuesta, de revisar los resultados de las modalidades de los partidos, y el funcionamiento general de las apuestas. También se dedicará a analizar los datos recogidos por el sistema. En la vida real se correspondería con un alto mando o un cargo directivo que supervisa el correcto funcionamiento de su negocio, y revisa periódicamente los datos económicos, y las estadísticas de las apuestas que se realizan en el sistema

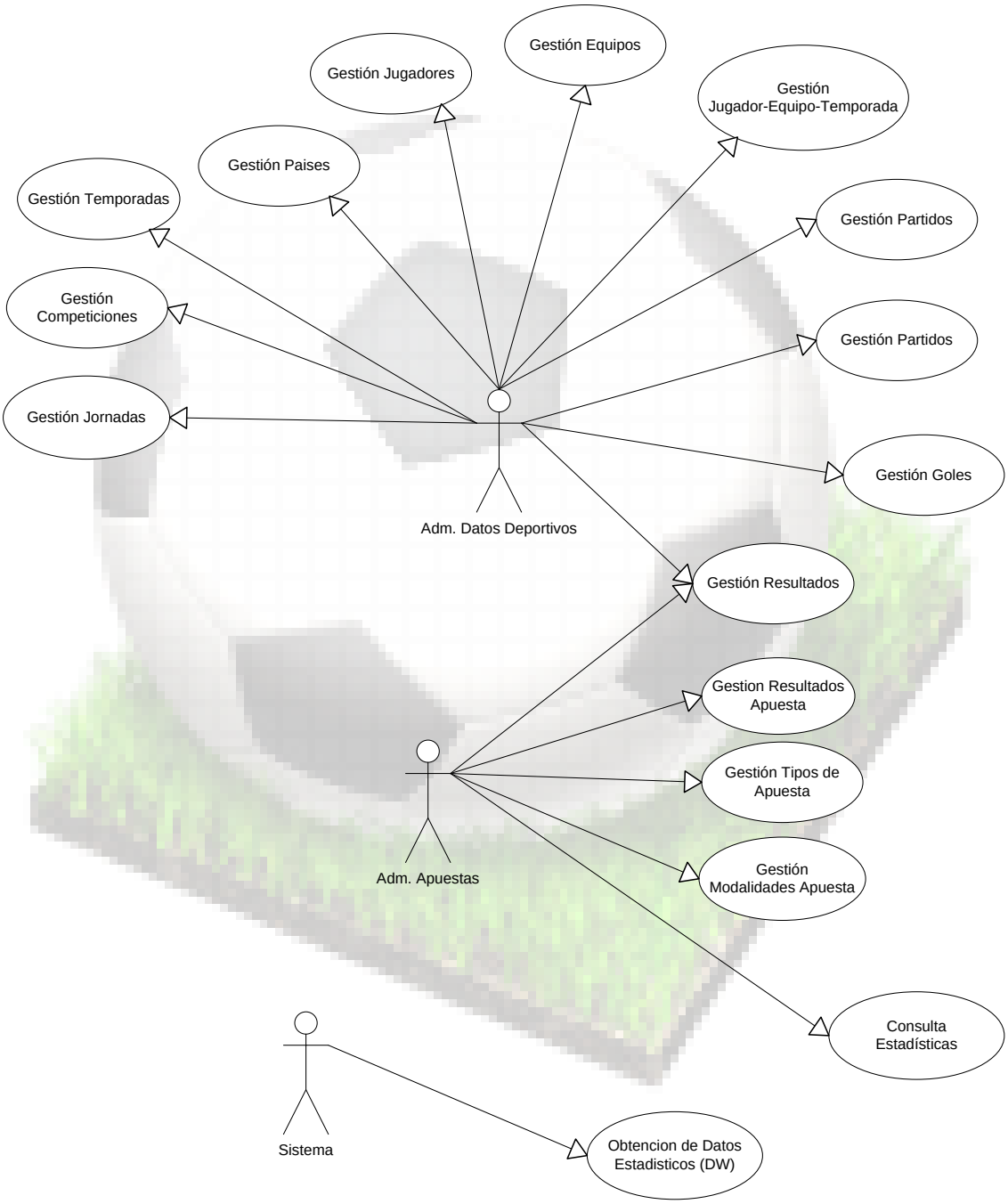
Usuario Jugador: serán los clientes del sistema, accederán a consultar la información de datos deportivos y podrán realizar apuestas sobre las distintas modalidades que se ofrezcan.

Sistema: Este rol será desempeñado por el propio sistema, que es el que se encargará de mantener los datos correctos en la base de datos, mediante la activación de los disparadores y procesos que permitan mantener la integridad de los datos. También se encargará de los procesos ETL para poder obtener los datos estadísticos. Además también gestionará el registro de modificaciones y eventos que ocurran sobre el sistema (LOG)

(*) Entendemos por gestión todos los procedimientos de alta, baja y modificación de datos en cada una de las entidades que se gestionen (que se omiten del diagrama de casos de uso para hacerlo más legible).

3.1.1.2 Casos de Uso

A continuación se destalla el diagrama de casos de uso propuesto para el problema planteado. A partir de estos casos de uso se implementarán los procesos de interacción con la base de datos:



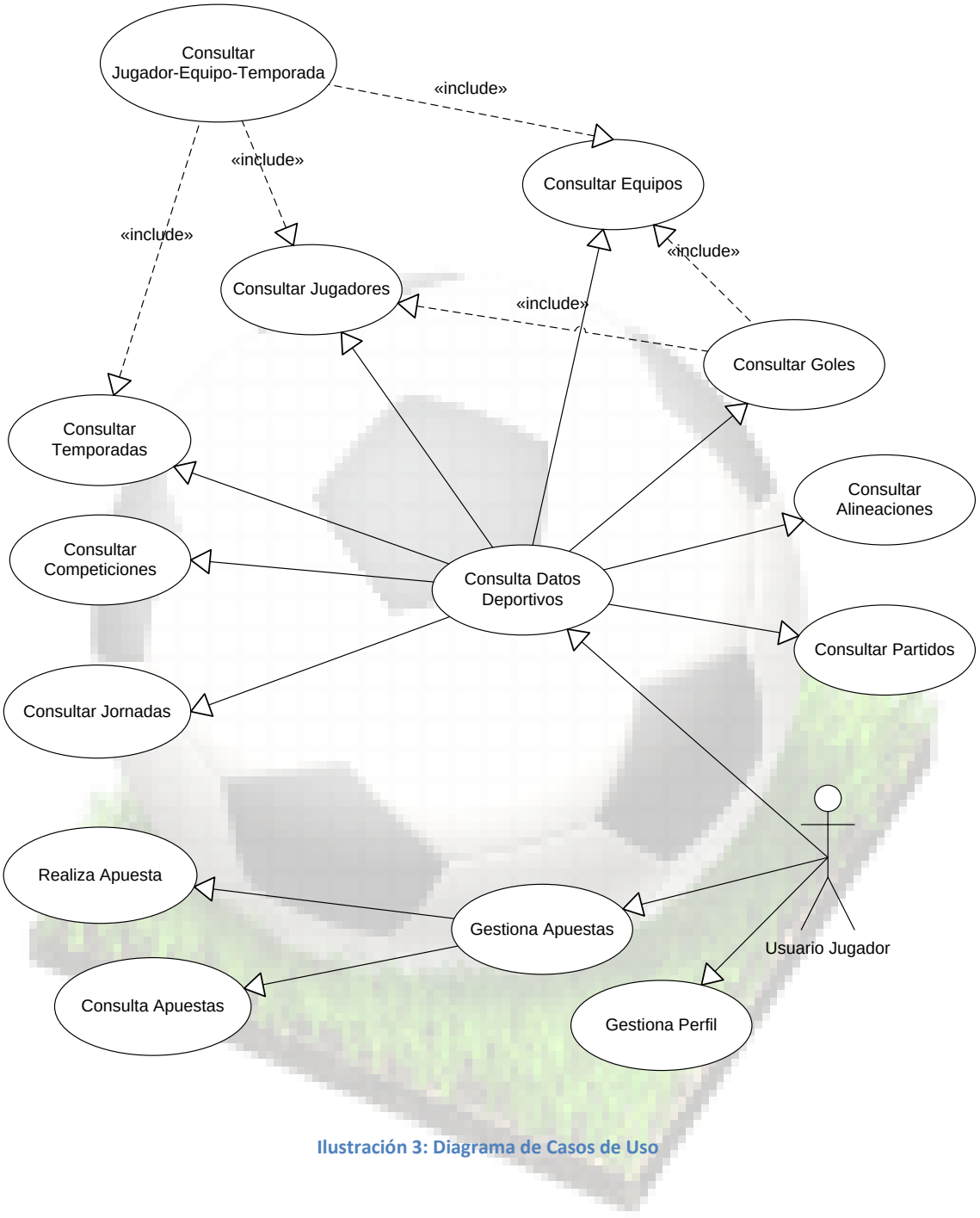


Ilustración 3: Diagrama de Casos de Uso

3.2 Diseño BD Operacional

3.2.1 Diseño Conceptual

3.2.1.1 Entidades y Atributos

El resultado de esta fase es un modelo independiente del SGBD que vamos a utilizar, que representa el sistema y que nos permite satisfacer los requisitos anteriormente enumerados.

Según los requisitos se identifican las siguientes entidades:

PAIS

Necesitamos saber a qué país pertenece un equipo, y un usuario. También podríamos asignar una competición a un país, pero en ese caso no podríamos gestionar competiciones internacionales, mundiales, ligas europeas, etc., así que en principio las competiciones no tendrán un país definido. Otra opción es definir un país como internacional, y asignar a este las competiciones internacionales.

Atributos: idPais, NombreEquipo

EQUIPO

Guardaremos datos de los equipos

Atributos: idEquipo, NombreEquipo, *idPaís*

COMPETICION

Guardaremos la información de cada competición

Atributos: idCompeticion, DescripciónCompeticion

JORNADA

Guardaremos todas las jornadas de una competición de cada temporada

Atributos: idTemporada, idJornada, DescripciónJornada

TEMPORADA

Almacenaremos en esta entidad las diferentes temporadas

Atributos: idTemporada, DescripcionTemporada

JUGADOR

Aquí se guardarán los datos de los jugadores

Atributos: idJugador, Nombre

JUEGA_EN

En esta entidad se establecerá una relación para saber en qué equipo juega un jugador en una temporada.

Sus atributos serán claves ajenas que harán referencia a JUGADOR, EQUIPO, TEMPORADA

PARTIDO

Datos de cada uno de los partidos: equipo local y visitante, fecha

Atributos: idPartido, idEquipoLocal, idEquipoVisitante, FechaPartido, Finalizado

JUEGA

Aquí definiremos las alineaciones de los partidos

Atributos: idPartido, idJugador

RESULTADO

En esta entidad guardaremos una vez finalicen los partidos los resultados de las distintas modalidades de apuesta, así se podrán evaluar las apuestas realizadas y determinar si se aciertan o no.

GOL

Guardaremos la información de los goles marcados en los partidos de futbol, quien los marca y en qué periodo.

Atributos: *idPartido,* nOrden,Tiempo, *idEquipo*, *idJugador*

USUARIO

Información de los usuarios que efectúan apuestas

Atributos: IdUsuario, Nombre, edad

APUESTA

Aquí se guardará toda la información relativa a las apuestas:

Atributos: idApuesta, *idUsuario, idPartido, idModalidad*, FechaApuesta, valorApostado, resultadoApuesta

MODALIDAD

En esta entidad definiremos las diferentes modalidades de apuesta

Atributos: idModalidad, descripcionModalidad

TIPO_MODALIDAD

Esta relación servirá para distinguir los diferentes tipos de apuesta según el valor que el usuario quiera apostar: Jugador, resultado, nº de goles , equipo, ganador-empate.

Atributos: idTipoModalidad, idDescripcionTipo, idTipoApuesta

LOG

Esta entidad nos servirá para almacenar un registro de todas las acciones que manipulen datos de nuestro sistema. Esto permitirá realizar tareas de búsqueda de errores y labores de auditoría sobre los datos (permitirá saber cuándo se modificó un dato y los parámetros con los que se realizó la modificación).

Atributos: idLog. nombreProcedimiento, parámetrosEntrada, Resultado, fecha/hora de ejecución.

3.2.1.2 Relaciones

A partir de estas entidades ya podemos construir el diagrama entidad relación, y establecer las relaciones entre ellas:

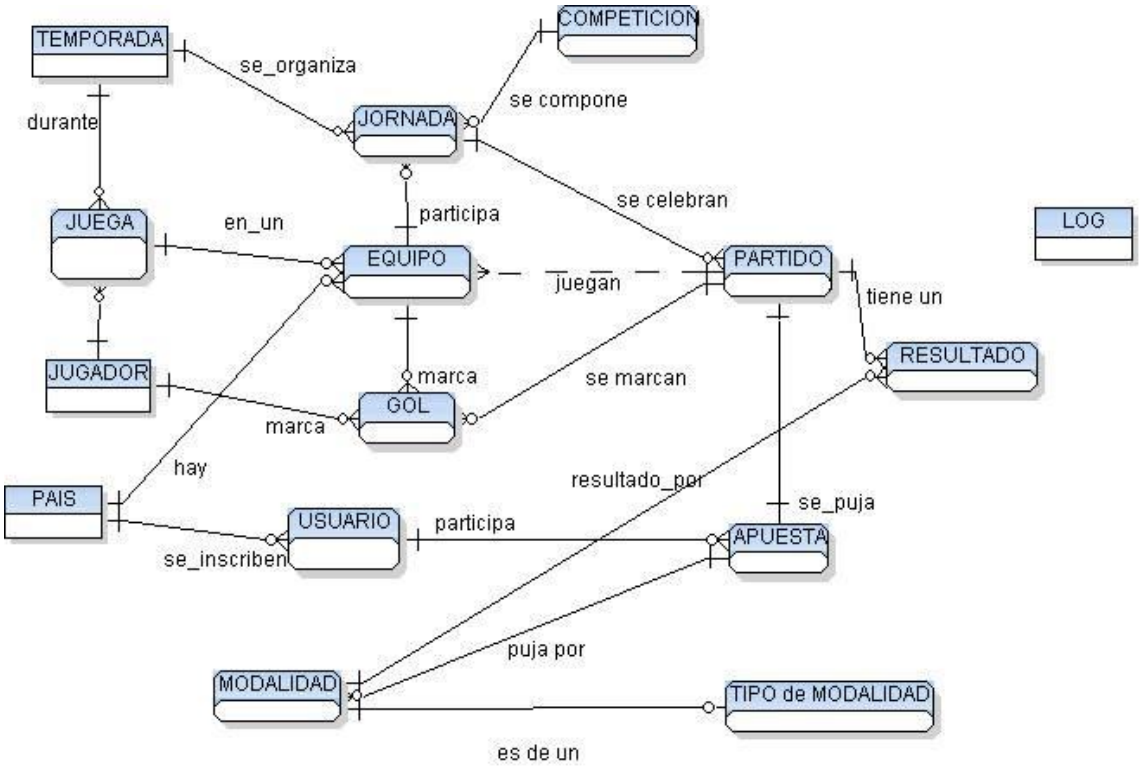


Ilustración 4. Modelo Conceptual

A partir de este diseño, estableceremos las siguientes restricciones:

- Un jugador juega en un equipo (y sólo en uno) en una temporada.
- Un jugador podrá cambiar de equipo en una temporada si todavía no ha disputado partidos en esa temporada.
- Un equipo siempre es de un país. Una competición no, porque podría darse el caso de que fuese un competición internacional.
- Todo partido será de una temporada, una competición, y entre dos equipos diferentes (uno local y otro visitante)
- Un equipo no juega más de una vez en una misma jornada
- Se podrán añadir o modificar goles de un partido siempre y cuando el partido no haya finalizado.
- El jugador que marque gol debe jugar en el equipo en la jornada que marca el gol
- Se podrá apostar por un partido solamente antes de la fecha de inicio
- Cuando se establezca el resultado de las modalidades de apuesta en un partido, se buscarán todas las apuestas realizadas sobre eventos del partido y evaluarlas para determinar si hay acierto o no.
- Se guardará un LOG de todas las acciones realizadas en la base de datos.

3.2.2 Diseño Lógico

Una vez definidas las relaciones y los atributos, el siguiente paso es, a partir del modelo lógico, realizar transformaciones y elaborar el diseño lógico de la base de datos, de forma que pueda implementarse en el SGDB elegido.

A partir del Modelo Conceptual, este es el diagrama lógico de la base de datos diseñada con la herramienta *Toad Data Modeler*:

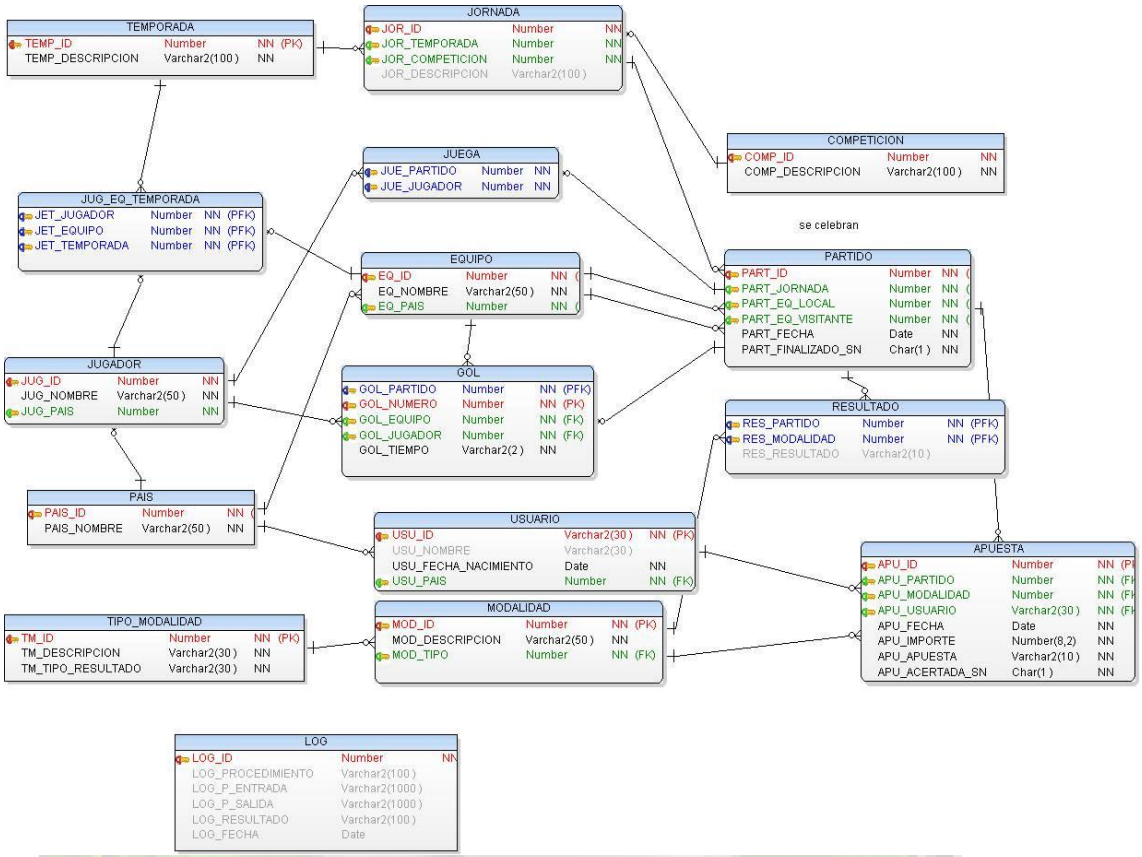


Ilustración 5: Modelo Lógico

En este diagrama ya se encuentran definidos todas las entidades, relaciones, y sus atributos, junto con sus tipos.

Por simplificar, se han definido los atributos que se consideran necesarios para resolver el problema planteado en el enunciado. En la vida real sería necesario definir más atributos, como por ejemplo datos particulares de los jugadores o de los equipos.

Esta es la relación de siglas y colores:

- **PK (Primary Key)**. Indica que el atributo es la clave primaria
- **NN (Not Null)**. Indica que el atributo tiene definida la restricción de valor no nulo
- **PFK (Primary Foreign Key)**. Indica que el atributo forma parte de la clave primaria, y que a su vez es la clave ajena con referencia a un atributo de otra relación
- **FK (Foreign Key)**. El atributo es una clave ajena que referencia la clave primaria de otra relación.

3.2.2.1 Entidades y atributos

Este es el desglose detallado de todas las entidades que se han definido, junto con el detalle de sus atributos:

PAIS
- PAIS_ID: Clave Primaria. Identificador del País. Este valor se asignará automáticamente a partir de la secuencia *SEQ_PAIS_ID*
- PAIS_NOMBRE: Nombre del País

TEMPORADA
- TEMP_ID: Clave Primaria. Identificador de la Temporada. Este valor se asignará automáticamente a partir de la secuencia *SEQ_TEMPORADA_ID*
- TEMP_DESCRIPCION: Descripción de la Temporada

COMPETICION
- COMP_ID: Clave Primaria, Identificador de la competición. Este valor se asignará automáticamente a partir de la secuencia *SEQ_COMPETICION_ID*
- COMP_DESCRIPCION: Descripción de la Competición

JORNADA (Es cada una de la etapas en que se fracciona una competición)
- JOR_ID: Clave Primaria, Identificador de la Jornada. Este valor se asignará automáticamente a partir de la secuencia *SEQ_JORNADA_ID.*
- *JOR_TEMPORADA*: Clave ajena. Identificador de la Temporada.
- *JOR_COMPETICION*: Clave ajena. Identificador de la Competición.
- JOR_DESCRIPCION: Descripción de la Jornada.

EQUIPO
- EQ_ID: Clave Primaria, Identificador del Equipo. Este valor se asignará automáticamente a partir de la secuencia *SEQ_EQUIPO_ID.*
- EQ_NOMBRE: Nombre del Equipo.
- *EQ_PAIS*: Clave ajena. Identificador del País.

PARTIDO
- PART_ID: Clave Primaria. Identificador del partido. Este valor se asignará automáticamente a partir de la secuencia *SEQ_PARTIDO_ID*
- PART_JORNADA: Clave Ajena. Identificador de la Jornada
- PART_EQ_LOCAL: Clave Ajena. Identificador del Equipo que juega como local
- PART_EQ_VISITANTE: Clave Ajena. Identificador del Equipo que juega como visitante
- PART_FECHA: Fecha del Partido
- PART_FINALIZADO_SN: Indica si el partido está finalizado o no. Una vez se actualice este campo a S, se calcularan automáticamente los resultados y ya no se podrá modificar.

JUGADOR
- JUG_ID: Identificador del Jugador
- JUG_NOMBRE: Nombre del Jugador
- JUG_PAIS: Clave ajena. Identificador del País del Jugador

JUEGA
- JUE_PARTIDO: Clave Primaria. Clave ajena. Identificador del Partido.
- JUE_JUGADOR: Clave Primaria. Clave ajena. Identificador del Jugador

GOL
- GOL_PARTIDO: Clave Primaria. Clave ajena. Identificador del Partido.
- GOL_NUMERO: Clave Primaria. Número de orden que se calculará automáticamente
- GOL_EQUIPO: Clave ajena. Identificador del Equipo que marca el gol
- GOL_JUGADOR: Clave ajena. Identificador del Jugador
- GOL_TIEMPO: Período en que se marca el gol: 1T si se marca en la 1ª parte, y 2T si se marca en la 2ª parte.

JUG_EQ_TEMPORADA (Histórico de equipos en los que juega un jugador)
- JET_JUGADOR: Clave primaria. Clave ajena. Identificador del Jugador
- JET_EQUIPO: Clave primaria. Clave ajena. Identificador del Equipo
- JET_TEMPORADA: clave primaria. Clave ajena. Identificador de la Temporada

TIPO_MODALIDAD (Definiremos los diferentes tipos de resultado por los que se puede apostar)
- TM_ID: Clave primaria. Identificador del Tipo de Modalidad. Esta valor se asignará automáticamente a partir de la secuencia *SEQ_TIPO_MODALIDAD_ID*
- TM_DESCRIPCION: Descripción del Tipo de Modalidad
- TM_TIPO_RESULTADO: Tipo de Resultado (JUGADOR, EQUIPO, Nº de GOLES, 1X2, RESULTADO,…)

MODALIDAD (Eventos de un partido sobre los que apostar)
- MOD_ID: Clave Primaria. Identificador de la Modalidad
- MOD_DESCRIPCION: Descripción de la Modalidad
- MOD_TIPO: Clave Ajena. Identificador de Tipo de Modalidad

USUARIO
- USU_ID: Clave Primaria. Identificador del Usuario.
- USU_NOMBRE: Nombre del Usuario
- USU_FECHA_NACIMIENTO: Edad del Usuario
- USU_PAIS: Clave ajena. Identificador del País

RESULTADO (Guardaremos el resultado de todas las modalidades que se definan, para cada partido)

- RES_PARTIDO: Clave Primaria. Clave ajena. Identificador del Partido.
- RES_MODALIDAD: Clave Primaria. Clave ajena. Identificador de Modalidad
- RES_RESULTADO: Resultado de la Modalidad de apuesta para ese partido. Se validará que es válido, según el tipo de resultado de la modalidad.

APUESTA

- APU_ID: Clave Primaria. Identificador de la Apuesta. Este valor se asignará automáticamente a partir de la secuencia SEQ_APUESTA_ID
- APU_PARTIDO: Clave ajena. Identificador del Partido.
- APU_MODALIDAD: Clave ajena. Identificador de la Modalidad
- APU_USUARIO: Clave Ajena. Identificador del Usuario que realiza la apuesta.
- APU_IMPORTE: Importe apostado.
- APU_APUESTA: Valor por el que puja el usuario
- APU_ACERTADA_SN: se guardará si el usuario acierta o no la apuesta: S=Acertada, N=Fallada

LOG

- LOG_ID: Clave primaria. Identificador del registro Log
- LOG_PROCEDIMIENTO: Nombre del procedimiento ejecutado
- LOG_P_ENTRADA: parámetros que se pasan al procedimiento
- LOG_P_SALIDA: parámetros que devuelve el procedimiento
- LOG_RESULTADO: Resultado devuelto por el procedimiento
- LOG_FECHA: fecha de ejecución del procedimiento

3.2.3 Diseño Físico

En esta fase se trata de implementar el modelo creado en el diseño lógico en el SGBD elegido, en este caso *Oracle.*

3.2.3.1 Pasos previos

Si suponemos que tenemos ya instalado *Oracle* en nuestro equipo de trabajo, los pasos previos a la creación de los objetos de la base de datos son los siguientes:

Creación del Tablespace

El *tablespace* es una unidad lógica de almacenamiento (un archivo) donde guardaremos todos los objetos del esquema de la base de datos.

Para albergar nuestra base de datos crearemos un *tablespace* con un tamaño inicial de 50MB, espacio más que suficiente para guardar los datos con los que vamos a trabajar. Lo crearemos como gestionado localmente, y será el administrador quien deberá aumentar el tamaño manualmente si es necesario. Esta será la sentencia de creación:

```
CREATE TABLESPACE TS_PFC_OP
DATAFILE 'C:\PFC\PFC_OP.dbf' SIZE 50M
EXTENT MANAGEMENT LOCAL AUTOALLOCATE;
```

Crear un usuario

Necesitamos crear un usuario que será el propietario de todos los objetos que creemos en la base de datos. Por tanto tendrá que tener permisos para poder crear y modificar objetos en la base de Datos.

```
--
--Creación del Usuario
--
CREATE USER USR_OP
IDENTIFIED BY operacional
DEFAULT TABLESPACE TS_PFC_OP
QUOTA UNLIMITED ON TS_PFC_OP
TEMPORARY TABLESPACE temp;
--
-- Asignación de permisos al Usuario
--
GRANT CREATE SESSION ,
      CREATE TABLE   ,
      CREATE TRIGGER  ,
      CREATE TYPE     ,
      CREATE SEQUENCE,
      CREATE PROCEDURE
TO USR_OP;
```

Una vez creado el *Tablespace* y el *usuario*, ya podremos conectarnos a Oracle como el usuario USR_OP, y proceder a crear los objetos de la base de datos.

3.2.3.2 Crear los objetos de la base de Datos

Secuencias

Tablas

Se crean todas las tablas que hemos definido en el diseño lógico, junto con sus claves primarias

Constraints (restricciones)

Definimos las restricciones de las tablas

Disparadores (Triggers)

Se han creado disparadores para asignar automáticamente los Identificadores en aquellas tablas donde el sistema debe asignarlos automáticamente.

3.3 Diseño BD Estadística (DW)

El modelo entidad relación (ER) fue concebido para reducir la cantidad de datos redundantes y evitar tener que modificar múltiples registros cuando se hace algún cambio. Esto se consigue a costa de penalizar el tiempo de respuesta de las consultas.

El modelo ER, que resulta muy conveniente en entornos en los que se producen cambios frecuentes, no es demasiado adecuado para entornos de análisis en los que prevalece la necesidad de obtener respuestas rápidas a las consultas, y el esquema debe resultar fácil de entender.

Necesitamos un sistema que nos proporcione la flexibilidad y la potencia de una hoja de cálculo, y la estructura y facilidad de consulta de una base de datos.

Para la base de datos estadística optamos por el diseño multidimensional, donde estableceremos los hechos sobre los cuales queremos conocer unos determinados datos, y las dimensiones o puntos de vista con las que queremos obtener los resultados.

3.3.1 Diseño Conceptual

A partir de las estadísticas que se desean obtener y que se han establecido en la especificación de requisitos, definiremos una estrella para cada estadística que deseamos obtener. El centro de la estrella será el hecho objeto de estudio, y las puntas de la estrella se corresponderán con las dimensiones o puntos de vista.

Los hechos que se desean analizar son los **Usuarios** (cantidad), y las **Apuestas** (cantidad e importe)

Una vez definidos los hechos el siguiente paso es definir la granularidad adecuada para tener el nivel de detalle deseado, pero que no se penalice el rendimiento del sistema por tener una gran cantidad de registros.

Estas son las dimensiones o puntos de vista requeridos para satisfacer los requisitos:

Edad de los usuarios

Esta dimensión nos permitirá realizar consultas desde el punto de vista de la edad de los usuarios. Además tendremos dos niveles de detalle:

- Tramo de Edad
- Edad

Con esta dimensión podremos conocer el importe apostado por edades o por tramos de edad.

País de los usuarios

Esta dimensión nos permitirá realizar consultas según la procedencia de los usuarios,

Equipo

Equipo por el que se apuesta. Nos permitirá conocer los equipos por los que más se apuesta, o países de los equipos por los que más se puja.

Modalidad

Esta dimensión también tiene dos niveles de detalle:

- Tipo de Modalidad
- Modalidad

Competición

Competición sobre la que se apuesta.

Jugador

Jugador por el que se apuesta. Este valor solamente se podrá rellenar en aquellos casos en los que el tipo de apuesta sea 'Jugador'

Guardaremos el nombre del jugador y el país

Temporada

Temporada en la que se realiza la apuesta

Resultado

Resultado de la apuesta (acierto o fallo)

Dependiendo del tipo de modalidad de cada apuesta, algunos valores de apuesta pueden tener valores no definidos para algunas dimensiones, por ejemplo, si el tipo de apuesta es EQUIPO, la dimensión Jugador no la podremos determinar. En la fase de diseño lógico especificaremos detalladamente estas restricciones.

3.3.2 Diseño Lógico

A partir de los hechos y las dimensiones que hemos definido en el diseño conceptual, lo que he hecho al definir el modelo lógico, es realizar el diseño lógico del DW, intentando adaptar las dimensiones a una estructura en estrella, de manera que aglutinamos algunas dimensiones como agregadas, ya que en el caso de la base de datos estadísticas lo prioritario es que las consultas sean ágiles y fáciles de implementar, más allá de que los datos se encuentren normalizados.

El diagrama lógico resultante es el siguiente:

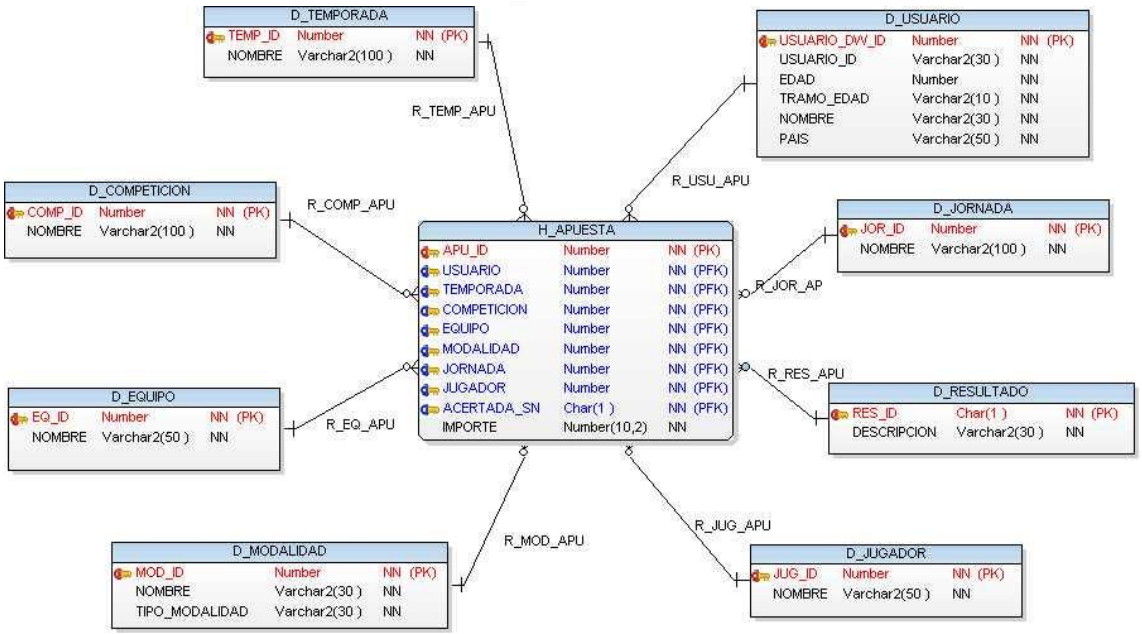


Ilustración 6: Modelo Lógico del Almacén de Datos

A continuación se detallan todas las tablas con sus campos y las restricciones y condiciones que hay que tener en cuenta:

En el enunciado del proyecto se nos indica que se necesitarán obtener estadísticas de los usuarios y de las apuestas. Por ello, centramos todo el diseño de la estrella a partir del hecho de las Apuestas, aunque en la dimensión D_USUARIO hemos añadido todos los campos que permiten realizar análisis estadísticos de los usuarios por edad y por país.

Dimensión Usuario

La clave primaria de esta tabla no coincide con la de la tabla de usuarios del modelo operacional, ya que si ha variado la edad de un usuario, entonces hay que dar de alta un nuevo registro en esta dimensión para reflejar todas las edades con las que participa en apuestas. Esta edad se calcula como la edad en años del usuario en el momento de realizar cada apuesta. El identificador del paciente se guarda, por si en un futuro se desean añadir más datos de los usuarios a esta dimensión.

El resto de campos son el nombre, el tramo de edad (la longitud de los tramos de edad se define en los procesos ETL. Inicialmente se establece a 5 años, pero se podrá modificar fácilmente si se necesita), y el país del usuario. El dato del país he decidido incluirlo en esta dimensión, y no aumentar la granularidad con el fin de que las consultas sean más rápidas.

Dimensión Temporada

Esta dimensión tomará sus valores de la tabla *Temporada* del modelo operacional, identificador y descripción.

Dimensión Competición

Esta dimensión tomará sus valores de la tabla *Competición* del modelo operacional.

Dimensión Jornada

Esta dimensión tomará sus valores de los campos JOR_ID, JOR_DESCRIPCION de la tabla *Jornada* del modelo operacional.

Dimensión Equipo

Esta dimensión tomará sus valores de la tabla *Equipo* del modelo operacional.

Puede ocurrir que según el tipo de apuesta (por ejemplo si se apuesta que el resultado de un partido será empate) no seamos capaces de identificar el equipo por el que se apuesta. Para evitar dejar estos valores a nulo, y permitir realizar intercambios en las tablas donde se realizan las consultas (*Drill-Across*) crearemos un Equipo ficticio cuyo código será *0* y su descripción *'- EQUIPO NO DEFINIDO -'*

Dimensión Modalidad

Esta dimensión tomará sus valores de la tabla *Modalidad* (MOD_ID, MOD_DESCRIPCION). Además se incluirá el valor de la descripción del tipo de modalidad.

El tipo de modalidad, como ya se ha comentado en el modelo operacional, indicará el tipo de resultado por el que se realiza una apuesta (JUGADOR, EQUIPO, Nº de GOLES, '1X2', etc.).

Dimensión Jugador

Esta dimensión tomará los valores de la tabla *Jugador* (JUG_ID, JUG_NOMBRE) del modelo operacional.

Igual que en la dimensión equipo, puede que según el tipo de apuesta no sea posible asignar un valor a esta dimensión. Por eso, igual que en el caso anterior, para evitar asignar el valor NULO a esta dimensión definiremos un Jugador con el identificador *0* y el nombre *'-JUGADOR NO DEFINIDO -'*.

Dimensión Resultado

Esta dimensión tomará los posibles valores que puede tomar una apuesta (acierto o fallo)

Hecho APUESTA

Esta tabla es la que permitirá centralizar todas las consultas estadísticas de apuestas. Contendrá aparte del campo clave que identificará cada apuesta, claves ajenas que referenciaran todas las dimensiones.

Como ya se ha comentado, ocurrirá que según el tipo de apuesta no sea posible dar valor a algunas de estas claves ajenas. Por esto en algunas dimensiones habrá que definir valores genéricos para poderlos asignar en aquellos casos en que el valor de una dimensión en una apuesta debería ser nulo.

Aparte de los campos que hacen referencia a las dimensiones, también incluimos el valor del importe de la apuesta, y el valor que indica si el jugador ha ganado la apuesta o no. Así podremos obtener estadísticas de importe apostado, o número de apuestas totales, o acertadas, etc.

3.3.3 Diseño Físico

En esta fase se trata de implementar la base de datos DW diseñada sobre el SGBD Oracle. Para ello crearemos un fichero script que creará todos los objetos (tablas, restricciones y relaciones).

El proceso de carga de datos en el DW será mediante un proceso ETL sobre la base de datos operacional. Esto minimizará la posibilidad de que se produzcan errores no esperados en la introducción de datos, ya que todos ellos se encuentran en una base de datos que hemos diseñado nosotros, y que conocemos perfectamente cómo almacena los datos.

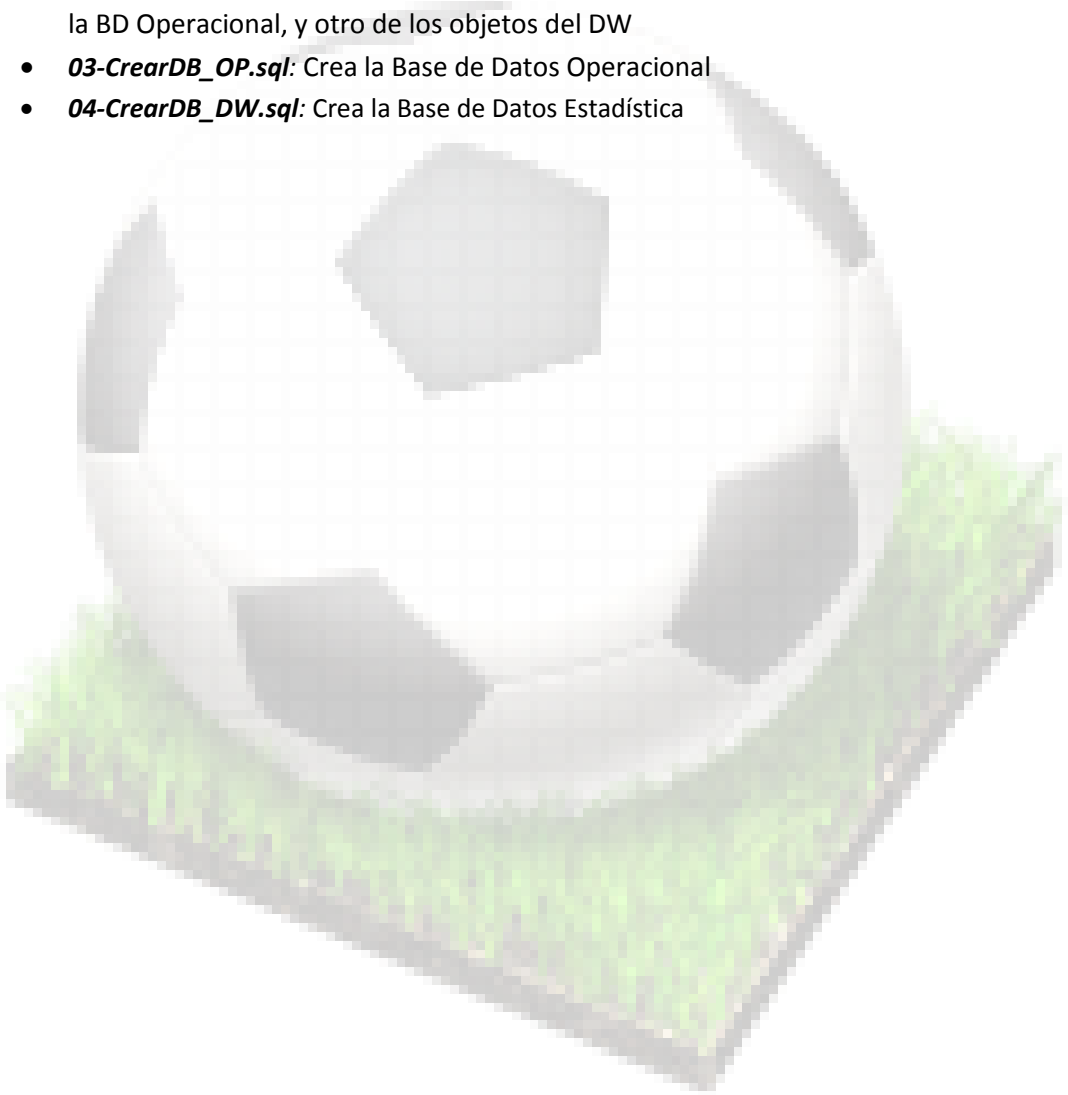
Este proceso podrá ser ejecutado por el usuario Supervisor, y realizará una carga completa de los datos a partir de la BD Operacional. Para ello vaciará todos los datos de las tablas, y las volverá a llenar con los datos existentes en el momento de su ejecución.

3.4 Resumen

Llegados a este punto del proyecto ya hemos cumplido los hitos previstos en la planificación correspondientes a la segunda entrega parcial (PAC2).

En este punto ya disponemos de los siguientes ficheros script que formarán parte del producto final:

- ***01-TABLESPACES.sql***: Se crean los ficheros en el sistema donde se guardarán las bases de datos
- ***02-USUARIOS.sql****:* Se crean dos usuarios, uno que será el propietario de los objetos de la BD Operacional, y otro de los objetos del DW
- ***03-CrearDB_OP.sql****:* Crea la Base de Datos Operacional
- ***04-CrearDB_DW.sql****:* Crea la Base de Datos Estadística



4 Implementación

4.1 Procesos de manipulación de la Base de Datos Operacional

Tal y como se especifica en el enunciado, todas las manipulaciones sobre los datos desde el exterior se realizarán mediante llamadas a procedimientos almacenados de la base de datos, que proporcionen las funcionalidades descritas en los casos de uso. Estos procedimientos serán los encargados que los datos son válidos, y además de ejecutar las operaciones correspondientes, registrarán en el Log todos las manipulaciones realizadas.

4.1.1 Gestión de Datos Deportivos

4.1.1.1 Gestión de Temporadas

Alta de temporadas

Función: Inserta un nuevo registro en la tabla de **TEMPORADA**

Parámetros de Entrada: _descripción

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Si se consigue insertar correctamente la Temporada, la respuesta es OK, y se devuelve 0 (cero).

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Baja de Temporadas

Función: Elimina un registro de la tabla **TEMPORADA**

Parámetros de Entrada: _id

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Si se consigue borrar correctamente la Temporada, la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Modificación de Temporadas

Función: Permite modificar la descripción un registro en la tabla **TEMPORADA**

Parámetros de Entrada: _id, _descripción

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Si se consigue modificar correctamente el registro, la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Consulta de temporadas

Función: Permite consultar los datos de una o todas las Temporadas

Parámetros de Entrada: _id

Parámetros de Salida: _respuesta, _rs

Funcionamiento:

Si *_id* es nulo se devuelven todas las temporadas

En caso contrario se busca la temporada cuyo TEMP_ID coincide con el parámetro *_id.*

Si no se encuentran datos se devuelve el cursor RS vacío.

Se devuelve un cursor con los datos obtenidos.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error.

Se guarda la información de la ejecución en el LOG.

4.1.1.2 Gestión de Competiciones

Alta

Función: Permite insertar un registro en la tabla **COMPETICION**

Parámetros de Entrada: descripción

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Si se consigue insertar correctamente la competición, la respuesta es OK, y se devuelve 0 (cero).

En caso contrario la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Baja

Función: Permite eliminar un registro en la tabla **COMPETICION**

Parámetros de Entrada: _id

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Si se consigue eliminar correctamente la competición, la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Modificación

Función: Permite modificar la descripción un registro en la tabla **COMPETICION**

Parámetros de Entrada: _id, _descripción

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Si se consigue modificar correctamente la competición, la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Consulta de competiciones

Función: Permite consultar los datos de una o todas las Competiciones

Parámetros de Entrada: _id

Parámetros de Salida: _respuesta, _rs

Funcionamiento:

Si *_id* es nulo se devuelven todas las competiciones.

En caso contrario se busca la competición cuyo ID coincide con el parámetro

Si no se encuentran datos se devuelve el cursor RS vacío..

Se devuelve un cursor con los datos solicitados.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error.

Se guarda la información de la ejecución en el LOG.

4.1.1.3 Gestión de Paises

Alta de Países

Función: Inserta un nuevo registro en la tabla **PAIS**

Parámetros de Entrada: _descripción

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Si se consigue insertar correctamente el País, la respuesta es OK, y se devuelve 0 (cero).

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Baja de Países

Función: Elimina un registro de la tabla **PAIS**

Parámetros de Entrada: _id

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Si se consigue borrar correctamente el País, la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Modificación de Países

Función: Permite modificar la descripción un registro en la tabla **PAIS**

Parámetros de Entrada: _id, _descripción

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Si se consigue modificar correctamente el registro, la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Consulta de Países

Función: Permite consultar los datos de uno o todos los Países

Parámetros de Entrada: _id

Parámetros de Salida: _respuesta, _rs

Funcionamiento:

Si *_id* es nulo se devuelven todas las temporadas

En caso contrario se busca la temporada cuyo PAIS_ID coincide con el parámetro *_id.*

Si no se encuentran datos se devuelve el cursor RS vacío.

Se devuelve un cursor con los datos obtenidos.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error.

Se guarda la información de la ejecución en el LOG.

4.1.1.4 Gestión de Jornadas

Alta

Función: Permite insertar un registro en la tabla **JORNADA**

Parámetros de Entrada: _temporada, _competición, _descripción

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Debe existir tanto la *_temporada* como la *_competición*.

Si se consigue insertar correctamente la jornada, la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Baja

Función: Permite eliminar un registro en la tabla **JORNADA**

Parámetros de Entrada: _id

Parámetros de Salida: _respuesta

Funcionamiento:

Si se consigue eliminar correctamente la jornada, la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Modificación

Función: Permite modificar un registro en la tabla **JORNADA**

Parámetros de Entrada: _id, _temporada, _competición, _descripción

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Debe existir tanto la *_temporada* como la *_competición*.

Si se consigue insertar correctamente la jornada, la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Consulta de jornadas

Función: Permite consultar las jornadas de una competición en una temporada

Parámetros de Entrada: _temporada, _competición

Parámetros de Salida: _respuesta, _rs

Funcionamiento:

Si alguno de los parámetros de entrada es nulo se devuelve un error

Si no se encuentran datos se devuelve el cursor RS vacío.

Devuelve un cursor con las jornadas encontradas.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error.

Se guarda la información de la ejecución en el LOG.

4.1.1.5 Gestión de Equipos

Alta

Función: Permite insertar un registro en la tabla **EQUIPO**

Parámetros de Entrada: _nombre, _pais

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Debe existir un registro en la tabla país cuyo PAIS_ID sea igual a *_pais*.

Si se consigue insertar correctamente el equipo, la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Baja

Función: Permite eliminar un registro en la tabla **EQUIPO**

Parámetros de Entrada: _id

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Si se consigue borrar correctamente el equipo, la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Modificación

Función: Permite modificar un registro en la tabla **EQUIPO**

Parámetros de Entrada: _id, _nombre, _pais

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Si no existe el equipo identificado por *_id* se genera un error.

Si no existe el país identificado por *_pais* se genera un error

Si se consigue borrar correctamente el equipo, la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Consulta de Equipos

Función: Permite consultar los equipos de un país o todos los equipos

Parámetros de Entrada: _pais

Parámetros de Salida: _respuesta, _rs

Funcionamiento:

Si se indica algún valor en _id, se busca el país correspondiente al id.

Si *_pais* es nulo se devuelven todos los equipos

En otro caso se devuelven los equipos del país indicado

Si no se encuentran datos se devuelve el cursor RS vacío.

Devuelve un cursor con los equipos encontrados.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error.

Se guarda la información de la ejecución en el LOG.

4.1.1.6 Gestión de Partidos

Alta

Función: Permite insertar un registro en la tabla **PARTIDO**

Parámetros de Entrada: _jornada, _equipoLocal, _equipoVisitante, _fecha

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

El *_equipoLocal* y el *_equipoVisitante* deben existir en la tabla EQUIPO, y no pueden ser el mismo.

Se comprueba que ninguno de los dos equipos juega en otro partido en la misma jornada.

Si la *_fecha* es nula se devuelve un error.

Inicialmente el partido se considera no finalizado (PART_FINALIZADO_SN='N')

Si se consigue insertar correctamente el partido, la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Baja

Función: Permite eliminar un registro en la tabla **PARTIDO**

Parámetros de Entrada: _id

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Si se consigue borrar correctamente el partido, la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Modificación

Función: Permite modificar un registro de la tabla **PARTIDO**

Parámetros de Entrada: _id, _fecha, _finalizado_sn

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Si se consigue modificar correctamente el partido, la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Consulta de Partidos

Función: Permite consultar los partidos de una jornada, o de un equipo en una competición.

Parámetros de Entrada: _ID, _jornada, _equipo

Parámetros de Salida: _respuesta, _rs

Funcionamiento:

Los parámetros de entrada no pueden ser nulos todos a la vez.

Si *_id* no es nulo entonces se busca el partido por id.

Se buscaran los partidos que cumplan los requisitos de los parámetros de entrada

Se devolverá un cursor con los registros solicitados

Si no se encuentran registros se devuelve el cursor RS vacío.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

4.1.1.7 Gestión de Jugadores

Alta

Función: Permite insertar un registro en la tabla **JUGADOR**

Parámetros de Entrada: _nombre, _pais

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

El *_pais* debe existir en la tabla PAIS.

Si se consigue insertar correctamente el jugador, la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Baja

Función: Permite eliminar un registro en la tabla **JUGADOR**

Parámetros de Entrada: _id

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Si se consigue borrar correctamente el registro devuelve OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Modificación

Función: Permite modificar un registro en la tabla **JUGADOR**

Parámetros de Entrada: _id, _nombre, _pais

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

El *_pais* debe existir en la tabla PAIS.

Si se consigue modificar correctamente el jugador, la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Consulta

Función: Permite consultar registro en la tabla JUGADOR, o todos los jugadores de un equipo en una temporada.

Parámetros de Entrada: id, _equipo, _temporada

Parámetros de Salida: _respuesta, _rs

Funcionamiento:

Si *_temporada* es nulo se mostrarán los datos de la temporada con *id* mayor.

Si los parámetros *_equipo* y *_id* son nulos se devuelve un error.

Si no se encuentran registros se devuelve un error.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error.

Se guarda la información de la ejecución en el LOG.

4.1.1.8 Gestión de Alineaciones (Tabla JUEGA)

Alta

Función: Permite insertar un registro en la tabla **JUEGA**.

Parámetros de Entrada: _jugador, _partido

Parámetros de Salida: _respuesta

Funcionamiento:

Se comprueba que el jugador pertenece a alguno de los dos equipos que disputan el partido en la temporada.

Por supuesto, el *_jugador* y el *_partido* deben existir.

Si se consigue insertar correctamente el registro, la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Baja

Función: Permite eliminar un registro en la tabla **JUEGA**

Parámetros de Entrada: _jugador, _partido

Parámetros de Salida: _respuesta

Funcionamiento:

Si se consigue borrar correctamente el registro, la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Modificación

Este procedimiento no se implementará. Solamente se permitirá añadir o eliminar registros

Consulta

Función: Permite consultar los jugadores que juegan en un partido, de un equipo, o de los dos equipos.

Parámetros de Entrada: , _partido, _equipo

Parámetros de Salida: _respuesta

Funcionamiento:

Si no se indica el *_equipo* se devuelven todos los jugadores que disputan el partido

Si no se indica el *_partido* se devuelve un error

Si la consulta no devuelve datos se devuelve un error

Si se consigue borrar correctamente el registro la respuesta es OK, y se devuelve un cursor con los datos solicitados

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error

Se guarda la información de la ejecución en el LOG.

4.1.1.9 Gestión de Goles

Alta

Función: Permite registrar los goles que se marquen en los partidos (tabla **GOL**)

Parámetros de Entrada: _partido, _equipo, _jugador, _tiempo

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Todos los parámetros son obligatorios.

El parámetro *_tiempo* solamente puede tomar los valores '1T' o '2T'.

Para calcular el valor de GOL_NUMERO, se buscará el valor máximo de GOL_NUMERO para ese PARTIDO, y se incrementará en 1.

Se comprobará si el *_jugador* pertenece al equipo *_equipo* en la temporada correspondiente a la jornada en que se disputa el partido, y que juega en la jornada en que se disputa el partido.

Si se consigue introducir correctamente el registro, la respuesta es OK, y se devuelve el valor de GOL_NUMERO.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Baja

Función: Permite eliminar un gol (tabla **GOL**).

Parámetros de Entrada: _partido, _numero

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Todos los parámetros son obligatorios.

Solamente se podrán eliminar registros si el partido no ha finalizado todavía (PART_FINALIZADO_SN='N')

Si se consigue eliminar correctamente el registro, la respuesta es OK, y se devuelve el valor de *_numero*.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Modificación

Función: Permite modificar los datos de un gol que se marquen en los partidos (tabla **GOL**)

Parámetros de Entrada: _partido, _numero, _jugador, _tiempo

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Solamente se permite modificar el jugador y el tiempo en que se marca el gol.

El parámetro *_tiempo* solamente puede tomar los valores '1T' o '2T'.

Se comprueba que el jugador juega en el equipo que marca el gol en la temporada en que se disputa el partido

Si se consigue modificar correctamente el registro se devuelve OK.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG

Consulta

Función: Permite consultar los goles por partido, jugador o equipo (tabla **GOL**)

Parámetros de Entrada: _partido, _jugador, _equipo

Parámetros de Salida: _respuesta, _rs

Funcionamiento:

Se deberá indicar al menos uno de los parámetros.

Devuelve un cursor con los datos obtenidos.

Si no se encuentran datos se devuelve el cursor RS vacío.

Se guarda la información de la ejecución en el LOG.

4.1.1.10 Gestión de Jugadores-Equipo-Temporada

Alta

Función: Permite registrar en qué equipo juega un jugador en una temporada

Parámetros de Entrada: _jugador, _equipo, _temporada

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Todos los parámetros son obligatorios.

Si no existe alguno de los parámetros se devuelve un error.

Se comprueba si el *_jugador* juega en otro equipo en la misma temporada. Si se encuentra se genera un error.

Si se consigue introducir correctamente el registro, la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Baja

Función: Permite eliminar un jugador de un equipo en una temporada

Parámetros de Entrada: _jugador, _equipo, _temporada

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Todos los parámetros son obligatorios.

Se comprueba si el *_jugador* ha jugado algún partido con el equipo en la temporada. En ese caso se genera un error.

Si se consigue eliminar correctamente el registro, la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Modificación

Función: Permite modificar el equipo en que juega un jugador en una temporada

Parámetros de Entrada: _jugador, _equipo, _temporada

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Todos los parámetros son obligatorios.

Si no existe alguno de los parámetros se devuelve un error.

Se comprueba si el *_jugador* ha jugado algún partido en el equipo en esa temporada. En ese caso no se permite y se genera un error.

Si se consigue actualizar correctamente el registro, la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Consulta

Función: Permite consultar los jugadores de un equipo en una temporada, o los equipos en los que ha jugado un jugador en cada temporada.

Parámetros de Entrada: _jugador, _equipo, _temporada

Parámetros de Salida: _respuesta, _rs

Funcionamiento:

Se deberá indicar el equipo y la temporada, o el jugador

Realiza la búsqueda de los registros según los criterios indicados en los parámetros de entrada.

Devuelve un cursor con los datos obtenidos.

Si no se encuentran datos se devuelve el cursor RS vacío.

Se guarda la información de la ejecución en el LOG.

4.1.2 Gestión de Apuestas

4.1.2.1 Gestión de Tipos de Modalidades

Alta

Función: Permite registrar los Tipos de Resultado por los que se puede apostar.

Parámetros de Entrada: _descripcion, _tipoResultado

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Todos los parámetros son obligatorios.

_tipoResultado deberá tomar uno de los siguientes valores: 'EQUIPO', 'JUGADOR', 'RESULTADO', '1X2', 'N_GOLES'

Si se consigue introducir correctamente el tipo de modalidad, la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Baja

Función: Permite eliminar los Tipos de Resultado por los que se puede apostar.

Parámetros de Entrada: _id

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Si se consigue eliminar correctamente el Tipo de Modalidad, la respuesta es OK, y se devuelve 0.

Si se produce algún error, se devuelve ERROR, y el detalle del error.

Se guarda la información de la ejecución en el LOG.

Modificación

Función: Permite modificar un registro de Tipos de Resultado por los que se puede apostar.

Parámetros de Entrada: _id, _descripcion, _tipoResultado

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Si *_tipoResultado* no es nulo, y hay alguna apuesta cuya modalidad tenga este tipo de resultado, no se podrá modificar.

Si se consigue actualizar correctamente el registro la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1.

Se guarda la información de la ejecución en el LOG.

Consulta

Función: Permite consultar un Tipo de Resultado o todos

Parámetros de Entrada: _id

Parámetros de Salida: _respuesta, _rs

Funcionamiento:

Si *_id* es nulo, se devuelven todos los registros

Devuelve un cursor con los datos obtenidos.

Si no se encuentran datos se devuelve el cursor RS vacío..

Se guarda la información de la ejecución en el LOG.

4.1.2.2 Gestión de Modalidades

Alta

Función: Permite registrar los Modalidades de Apuesta

Parámetros de Entrada: _descripcion, _tipo

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Todos los parámetros son obligatorios.

Si se consigue introducir correctamente la modalidad, la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1

Se guarda la información de la ejecución en el LOG.

Baja

Función: Permite eliminar una modalidad de apuesta del sistema

Parámetros de Entrada: _id

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Si se consigue eliminar correctamente el registro se devuelve OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error

Se guarda la información de la ejecución en el LOG.

Modificación

Función: Permite modificar los datos de una modalidad de apuesta

Parámetros de Entrada: _id, _descripcion, _tipo

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

_tipo deberá tomar uno de los valores existentes en TIPO_MODALIDAD, y no se podrá modificar si existen apuestas para esa modalidad en concreto.

Si se consigue actualizar correctamente el registro se devuelve OK.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1

Se guarda la información de la ejecución en el LOG.

Consulta

Función: Permite consultar una modalidad de apuesta, o todas.

Parámetros de Entrada: _id,

Parámetros de Salida: _respuesta, _rs

Funcionamiento:

Si *_id* es nulo, entonces se buscan todas las modalidades

Devuelve un cursor con los datos obtenidos.

Si no se encuentran datos se devuelve el cursor RS vacío..

Se guarda la información de la ejecución en el LOG.

4.1.2.3 Gestión de Usuarios

Alta

Función: Permite registrar los Usuarios en el sistema

Parámetros de Entrada: _id, _nombre, _fechaNacimiento, _pais

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Todos los parámetros son obligatorios.

El _país debe existir

Si ya existe un usuario con el mismo *_id* la respuesta es ERROR.

Si se consigue introducir correctamente el usuario, la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1

Se guarda la información de la ejecución en el LOG.

Baja

Función: Permite eliminar los Usuarios en del sistema

Parámetros de Entrada: _id,

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Si se consigue eliminar correctamente el registro se devuelve OK.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1

Se guarda la información de la ejecución en el LOG.

Modificación

Función: Permite modificar los datos de un usuario.

Parámetros de Entrada: _id, _nombre, _fechaNacimiento, _pais

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

Si *_id* es nulo, entonces se genera un error

Se comprobará que el *_pais* (si no es nulo) existe

Si se consigue actualizar correctamente el registro se devuelve OK.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1

Se guarda la información de la ejecución en el LOG.

Consulta

Función: Permite consultar los datos de los usuarios

Parámetros de Entrada: _id, _país

Parámetros de Salida: _respuesta, rs

Funcionamiento:

Cualquiera de los dos parámetros puede ser nulo.

Devuelve un cursor con los datos obtenidos.

Si no se encuentran datos se devuelve el cursor RS vacío..

Se guarda la información de la ejecución en el LOG

4.1.2.4 Gestión de Resultados

Alta

Función: Permite registrar los resultados de todas las modalidades de apuestas que se producen en cada partido.

Parámetros de Entrada: _partido, _modalidad, re_resultado

Parámetros de Salida: _respuesta

Funcionamiento:

El *_partido* y la *_modalidad* deben existir.

El *_resultado* puede ser nulo.

Si el *_resultado* no es nulo, entonces hay que comprobar si es un valor correcto llamando a **ComprobarResultado**

Una vez introducido el registro se revisan todas las apuestas para ese partido y esa modalidad, y para aquellas apuestas cuyo resultado coincida con el parámetro *_resultado* hay que actualizar el valor del campo APU_ACERTADA_SN a 'S'.

Si se consigue introducir correctamente el registro la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error.

Se guarda la información de la ejecución en el LOG

Baja

Función: Permite eliminar un resultado de una modalidad de un partido.

Parámetros de Entrada: _partido, _modalidad

Parámetros de Salida: _respuesta, _resultat

Funcionamiento:

Si se consigue eliminar correctamente el registro, hay que actualizar todas las apuestas asociadas a ese partido y esa modalidad, y poner el campo APU_ACERTADA_SN='N'

Si todo va bien se la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1

Se guarda la información de la ejecución en el LOG.

Modificación

Función: Permite modificar un resultado de una modalidad de un partido.

Parámetros de Entrada: _partido, _modalidad, _re_resultado

Parámetros de Salida: _respuesta, _resultat

Funcionamiento:

El *partido* y la *modalidad* deben existir.

El _resultado puede ser nulo.

Si el *_resultado* no es nulo, entonces hay que comprobar si es un valor correcto llamando a **ComprobarResultado**

Una vez actualizado el registro se revisan todas las apuestas para ese partido y esa modalidad, y para aquellas apuestas cuyo resultado coincida con el parámetro *_resultado* hay que actualizar el valor del campo APU_ACERTADA_SN a 'S'.

Para aquellas apuestas cuyo resultado no coincide con el nuevo valor de *_resultado*, o si *_resultado* es nulo, hay que modificar el campo APU_ACERTADA_SN a 'N'.

Si se consigue actualizar correctamente el registro la respuesta es OK.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error.

Se guarda la información de la ejecución en el LOG

Consulta

Función: Permite consultar los resultados de todas las modalidades de apuesta de un partido, o los resultados de una modalidad de todos los partidos, o el resultado de una modalidad de un partido.

Parámetros de Entrada: _partido, _modalidad

Parámetros de Salida: _respuesta, _rs

Funcionamiento:

El *_partido* y la *_modalidad* pueden ser nulos. Si ambos son nulos se devolverán todos los resultados.

Devuelve un cursor con los datos obtenidos, que cumplen los criterios establecidos en los parámetros de entrada.

Si no se encuentran datos se devuelve el cursor RS vacío..

Se guarda la información de la ejecución en el LOG

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error.

Se guarda la información de la ejecución en el LOG

ComprobarResultado

Función: Comprobar si un valor de resultado de una modalidad es válido para una apuesta, en función del tipo de modalidad a la que se refiere

Parámetros de Entrada: _partido, _modalidad, re _resultado

Parámetros de Salida: _respuesta, _resultat

Funcionamiento:

El *_partido* y la *_modalidad* deben existir.

El *_resultado* puede ser nulo.

Si el *_resultado* no es nulo, entonces hay que comprobar si es un valor correcto, dependiendo del Tipo de Modalidad asociado a la modalidad de apuesta, según el siguiente criterio:

- Si el Tipo de Modalidad es 'EQUIPO', se comprueba que el valor de *_resultado* se corresponde con un id de equipo, y que ese equipo es uno de los dos que disputan el partido.
- Si el Tipo de Modalidad es 'JUGADOR', se comprueba que el valor de *_resultado* se corresponde con un id de jugador.
- Si el Tipo de Modalidad es 'RESULTADO', el valor del parámetro *_resultado* debe ser un valor con el formato 'NUMERO-NUMERO' (por ejemplo, '0-0', '1-0', etc.).
- Si el Tipo de Modalidad es '1X2', _resultado debe tomar uno de los siguientes valores: '1' para referirse al equipo que juega como local; '2' para hacer referencia al equipo que juega como visitante, 'X' para indicar que el resultado es un empate.
- Si el Tipo de Modalidad es 'N_GOLES', _resultado debe tomar un valor numérico.

Si no se cumple alguna de las condiciones anteriores, se devuelve un error, y la salida es -1.

Si todo es correcto la salida es 0.

4.1.2.5 Gestión de Apuestas

Alta

Función: Permite registrar una apuesta en el sistema.

Parámetros de Entrada: _partido, _modalidad, _usuario, _fecha, _importe, _apuesta

Parámetros de Salida: _respuesta

Funcionamiento:

El *_partido*, la *_modalidad* y el *_usuario* deben existir.

Si alguno de ellos no existe o es nulo, la respuesta es ERROR.

Se comprueba que la fecha de la apuesta es anterior a la fecha del partido

Se comprueba que el *_importe* es una cantidad numérica mayor que cero. No puede ser nulo.

Se comprueba que *_apuesta* toma un valor válido dependiendo de la modalidad por la que se apuesta, llamando a la función **ComprobarResultado** definida en la gestión de Resultados.

Inicialmente APU_ACERTADA_SN tomará el valor 'N'

Se inserta un nuevo registro en APUESTA

Si se consigue introducir correctamente el usuario, la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1

Se guarda la información de la ejecución en el LOG.

Baja

Función: Permite eliminar una apuesta

Parámetros de Entrada: _id

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

La *apuesta* debe existir.

Solamente se permitirá eliminar una apuesta si la fecha actual es anterior a la fecha del partido por el que se realiza la apuesta

Si todo va bien la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1

Se guarda la información de la ejecución en el LOG.

Modificación

Función: Permite realizar modificaciones en una apuesta

Parámetros de Entrada: _id, _importe, _apuesta

Parámetros de Salida: _respuesta, _resultado

Funcionamiento:

La *apuesta* debe existir.

Solamente se permite modificar una apuesta si la modificación se realiza antes de la fecha del partido.

Solamente se permite modificar el *_importe*, y la *_apuesta*.

Si se consigue actualizar correctamente la apuesta, la respuesta es OK, y se devuelve 0.

Si se produce algún error, la respuesta es ERROR, junto con el detalle del error, y se devuelve el valor -1

Se guarda la información de la ejecución en el LOG

Consulta

Función: Permite consultar apuestas, por partido, por modalidad y por usuario, o una apuesta en particular

Parámetros de Entrada: _id, _partido, _modalidad, _usuario

Parámetros de Salida: _respuesta, _rs

Funcionamiento:

Los cuatro parámetros no pueden ser nulos a la vez.

Devuelve un cursor con los datos obtenidos, que cumplen los criterios establecidos en los parámetros de entrada.

Si no se encuentran datos se devuelve el cursor RS vacío..

Se guarda la información de la ejecución en el LOG

4.1.3 Gestión del Registro de Actividad

4.1.3.1 Gestión de Log

Alta

Función: Permite registrar los eventos que se producen en la base de datos.

Parámetros de Entrada: _procedimiento, _pEntrada, _pSalida, _resultado, _fecha

Parámetros de Salida: _respuesta

Funcionamiento:

Se insertará un registro en la tabla LOG con los valores indicados en los parámetros.

La función devuelve 0.

Lógicamente, en este caso no se inserta un registro de la ejecución en el LOG.

Consulta

Función: Permite consultar los eventos ocurridos en la base de datos, por procedimiento ejecutado, por fecha, y por id.

Parámetros de Entrada: _id, _procedimiento, _fecha

Parámetros de Salida: _respuesta

Funcionamiento:

Todos los parámetros pueden ser nulos.

Si _id no es nulo, se buscará por id.

En otro caso se buscará por procedimiento y/o por fecha, si se pasa algún valor por el que filtrar.

Se realizará la consulta filtrando por aquellos parámetros que no sean nulos.

Devuelve un cursor con los datos obtenidos.

Si no se encuentran datos la respuesta será error.

Se guarda la información de la ejecución en el LOG

4.2 Procesos ETL de datos estadísticos

Para la extracción, transformación y carga de datos se han subdividido la tarea en varios pasos.

Se crea un procedimiento para la carga de cada una de las dimensiones, y posteriormente se realiza la carga de la tabla de hechos H_APUESTA.

El resultado de esta etapa es un fichero que encapsula la creación del paquete ETL en el que se implementan todas las funcionalidades.

Se define un procedimiento para la carga de cada una de las dimensiones y los hechos.

También se ha creado un procedimiento que elimina todos los datos existentes, y otro que realiza una carga completa de todo, es decir, primero borra todo y después ejecuta todos los procesos de carga de datos uno a uno.

Únicamente comentar que el proceso de carga de la dimensión de usuarios tiene un parámetro de entrada que indica la longitud de los tramos de edad que queremos para analizar los datos. Actualmente tiene el valor 5, pero puede modificarse por el administrador si se quiere que los tramos de edad sean distintos a 5 años.

4.3 Resumen

En este punto del proyecto ya disponemos de los scripts de implementación de los paquetes con los procedimientos que permiten la manipulación y consulta de los datos. También se ha implementado el paquete ETL que permite realizar los procesos de carga de datos desde el modelo operacional al Datawarehouse. Todo ello ha sido probado mediante las correspondientes pruebas unitarias.

En este punto ya disponemos de los siguientes ficheros script que formarán parte del producto final:

- ***01-TABLESPACES.sql***: Se crean los ficheros en el sistema donde se guardarán las bases de datos
- ***02-USUARIOS.sql****:* Se crean dos usuarios, uno que será el propietario de los objetos de la BD Operacional, y otro de los objetos del DW
- ***03-CrearDB_OP.sql****:* Crea la Base de Datos Operacional
- ***04-CrearDB_DW.sql****:* Crea la Base de Datos Estadística
- ***05_PAQUETES.sql***: Crea los paquetes de manipulación y consulta de los Datos
- ***06-ETL.sql***: Procesos ETL

El siguiente paso es introducir una serie de datos de prueba en el sistema, de forma que permitan realizar una prueba global del sistema implementado.

Posteriormente se realizará un análisis de la base de datos Datawarehouse para comprobar que el funcionamiento global del sistema es el esperado.

4.3.1 Incidencias

Por lo que respecta al cumplimiento del calendario previsto, quiero comentar que en este punto se ha producido un imprevisto de carácter personal (un problema de salud familiar), que me ha quitado una gran cantidad de tiempo disponible para la realización del proyecto durante las últimas tres semanas. No obstante, en el momento actual y a pesar de todo estoy llegando a tempo con los plazos, y creo que si no se complica más la cosa el proyecto podrá ser concluido en el plazo previsto.

5 Pruebas

En este punto se describen todas las pruebas que se han realizado sobre los paquetes implementados, para comprobar que el código está libre de errores, y que se cumplen las funcionalidades establecidas en los requisitos.

5.1 Subsistema de datos Deportivos

5.1.1 Gestión del Registro de LOG

- Insertar un nuevo registro de log (G_LOG.ALTA)
- Consultar el registro de log (G_LOG.CONSULTA)
- Buscar un identificador de log inexistente (G_LOG.CONSULTA)
- Comprobar que todos los procedimientos almacenados reportan correctamente el resultado de su ejecución

5.1.2 Gestión de Temporadas

- Insertar una nueva temporada
- Insertar una nueva temporada con la misma descripción -> error
- Borrar una temporada inexistente
- Borrar una temporada
- Borrar una temporada con registros relacionados en JUG_EQ_TEMPORADA o JORNADA
- Actualizar la descripción de una temporada
- Consultar temporadas

5.1.3 Gestión de Competiciones

- Insertar una nueva competición
- Insertar una nueva competición con la misma descripción (error)
- Borrar una competición inexistente (error)
- Borrar una competición
- Borrar una competición con registros relacionados en JORNADA (error)
- Actualizar la descripción de una competición
- Consultar competiciones

5.1.4 Gestión de Paises

- Insertar un nuevo país
- Insertar un nuevo país con el mismo nombre (error)
- Borrar un país inexistente (error)
- Borrar un país
- Borrar un país con registros relacionados en EQUIPO, USUARIO o JUGADOR (error)
- Actualizar el nombre de un país
- Consultar países

5.1.5 Gestión de Equipos

- Insertar un nuevo equipo
- Insertar un nuevo equipo con el mismo nombre (error)
- Borrar un equipo inexistente (error)
- Borrar un equipo
- Borrar un equipo con registros relacionados en PARTIDO, JUG_EQ_TEMPORADA, GOL (error)
- Actualizar el nombre de un equipo
- Consultar equipos

5.1.6 Gestión de Jugadores

- Insertar un nuevo jugador
- Insertar un nuevo jugador con el mismo nombre (error)
- Borrar un jugador inexistente (error)
- Borrar un jugador
- Borrar un jugador con registros relacionados en JUEGA, GOL (error)
- Actualizar el nombre de un jugador
- Consultar jugadores

5.1.7 Gestión de Jornadas

- Insertar una nueva jornada
- Borrar una jornada inexistente (error)
- Borrar una jornada
- Borrar a jornada con registros relacionados en PARTIDO (error)
- Actualizar la descripción de una jornada
- Consultar jornadas

5.1.8 Gestión de Partidos

- Insertar un nuevo partido
- Insertar un partido con los dos equipos iguales (error)
- Insertar un partido con un equipo que ya juegue en la misma jornada (error)
- Borrar un partido inexistente (error)
- Borrar un partido
- Borrar un partido con registros relacionados en APUESTA, RESULTADO, GOL (error)
- Actualizar la descripción de una jornada
- Consultar partidos

5.1.9 Gestión de Plantillas (JUG_EQ_TEMPORADA)

- Insertar un jugador en un equipo en una temporada
- Insertar el jugador anterior jugador en otro equipo en la misma temporada
- Borrar un jugador de un equipo en una temporada.
- Borrar un jugador que ya haya jugado partidos de un equipo en una temporada (error)
- Modificar el equipo en que juega un jugador
- Modificar el equipo en que juega un jugador que ya ha jugado en el equipo
- Consultar plantillas de equipos

5.1.10 Gestión de Alineaciones (JUEGA)

- Insertar un jugador en un partido
- Insertar un jugador de un equipo que no juega el partido (error)
- Borrar un jugador de un partido
- Borrar un jugador de un partido en el que no juega.
- Consultar alineaciones de partidos (de los dos equipos o de uno solo)

5.1.11 Gestión de Goles

- Insertar gol
- Insertar gol en un partido, un equipo que no existen (error)
- Insertar un gol por un jugador que no juega el partido (error)
- Borrar un gol de un partido no finalizado
- Borrar un gol de un partido finalizado (error)
- Borrar un gol que no existe (error)
- Modificar el jugador que marca un gol
- Modificar el jugador que marca un gol por otro que no juega el partido (error)
- Consultar goles

5.2 Subsistema de Apuestas

5.2.1 Gestión de Tipos de Modalidad

- Insertar un nuevo tipo de modalidad
- Insertar un nuevo tipo de modalidad con el mismo nombre (error)
- Borrar un tipo de modalidad inexistente (error)
- Borrar un tipo de modalidad
- Borrar un tipo de modalidad con registros relacionados en MODALIDAD (error)
- Actualizar el nombre de un TIPO DE MODALIDAD
- Consultar tipos de modalidad

5.2.2 Gestión de Modalidades

- Insertar una nueva modalidad
- Insertar una nueva modalidad con el mismo nombre (error)
- Borrar una modalidad inexistente (error)
- Borrar una modalidad
- Borrar una modalidad con registros relacionados en APUESTA (error)
- Actualizar el nombre de una MODALIDAD
- Consultar modalidades

5.2.3 Gestión de Usuarios

- Insertar un nuevo usuario
- Insertar un nuevo usuario con el mismo nombre (error)
- Borrar un usuario inexistente (error)
- Borrar un usuario
- Borrar un usuario con registros relacionados en APUESTA (error)
- Actualizar el nombre de un usuario
- Consultar usuarios

5.2.4 Gestión de Resultados

- Insertar un nuevo resultado
- Insertar un nuevo para el mismo partido y la misma modalidad (error)
- Borrar un resultado inexistente (error)
- Borrar un resultado
- Actualizar un resultado.
- Consultar resultados.
- Comprobar que las apuestas se actualizan correctamente en todos los casos

5.2.5 Gestión de Apuestas

- Insertar una nueva apuesta
- Insertar un nueva apuesta en un partido en una fecha posterior a la fecha del partido
- Insertar una apuesta con un resultado incorrecto para la modalidad
- Borrar una apuesta inexistente (error)
- Borrar una apuesta cuando el partido ha finalizado (error)
- Borrar una apuesta
- Actualizar una apuesta
- Actualizar una apuesta cuando el partido ha finalizado
- Consultar apuestas

5.3 Procesos ETL

- Comprobar que todos los procedimientos se ejecutan sin errores y los datos se cargan correctamente en la base de datos estadística

6 Explotación de la Base de Datos estadística con Pentaho.

En este capítulo mostramos como se puede realizar la explotación y análisis de la información almacenada en el Datawarehouse mediante la herramienta de Bussiness Intelligence Pentaho BI Server.

Para poder probar esta funcionalidad, hemos introducido una serie de datos de prueba sobre una base de datos vacía, utilizando el fichero ***07-DATOS_DE_PRUEBA.sql***.

A continuación hemos ejecutado el script ***08-CARGA_ETL.sql***, que realiza la carga de datos en el Datawarehouse.

Para la explotación de los datos utilizaremos una herramienta OLAP. Este tipo de herramientas presentan al usuario una visión multidimensional de los datos, de tal forma que éste puede realizar todo tipo de consultas sin necesidad de conocer la estructura interna (diseño físico) de la Base de Datos.

Para el diseño de los esquemas OLAP utilizamos la herramienta Schema Workbench. Aquí definiremos los cubos sobre los que queremos analizar los datos, con las dimensiones, y las medidas correspondientes.

Después mediante la herramienta Saiku Analytics realizaremos la explotación y análisis de los datos, de forma fácil y sencilla para el usuario final.

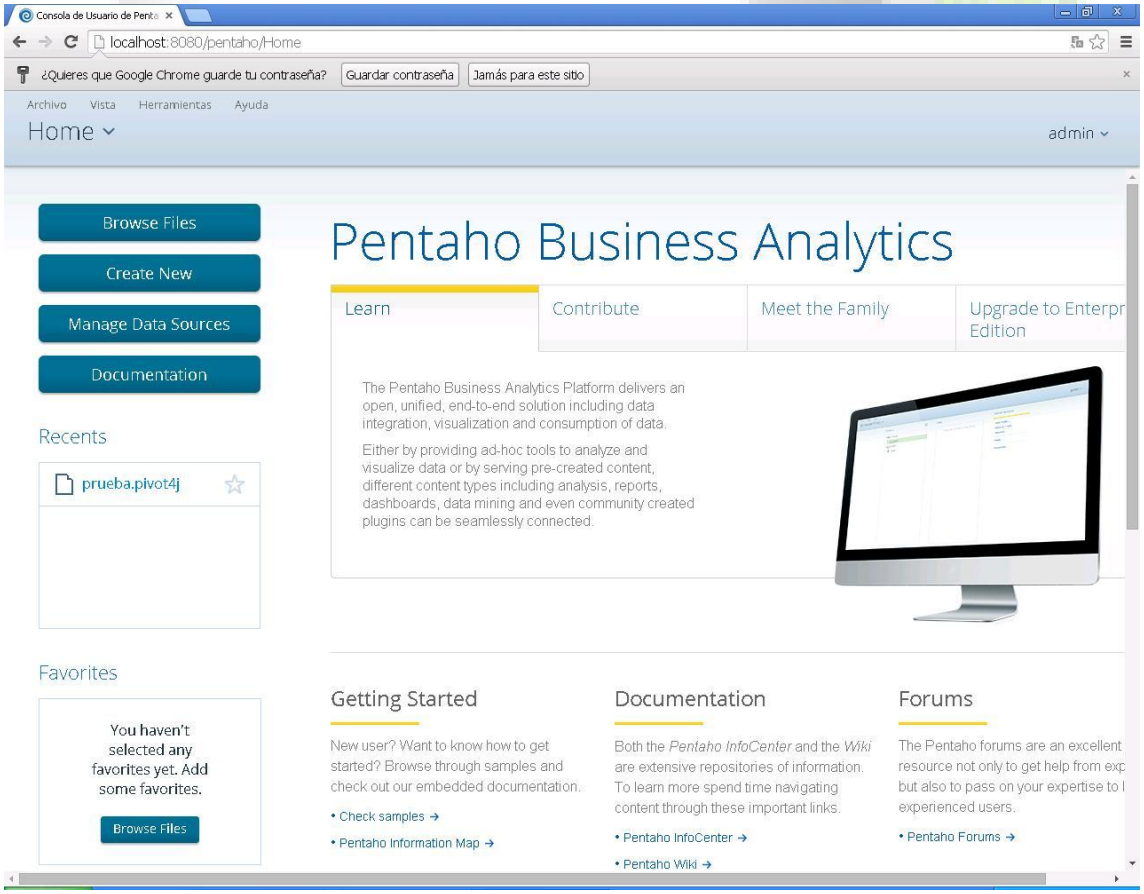


Ilustración 7: pantalla principal de Pentaho Business Analytics

6.1 Diseño del esquema con Pentaho Schema Workbench

Creamos un esquema que incluye dos cubos, a partir de los cuales queremos analizar la información.

Uno de ellos estará formado por la tabla H_APUESTA, que constituye el hecho a analizar. Alrededor de ella añadiremos todas las dimensiones por las que el usuario va a poder analizar los datos: Usuarios, temporadas, equipos, modalidades, jornadas, jugadores, competiciones y resultados. En el caso de la dimensión de Usuarios, definimos varios niveles, que nos permitirán analizar los datos por el país del usuario, el tramo de edad o la edad.

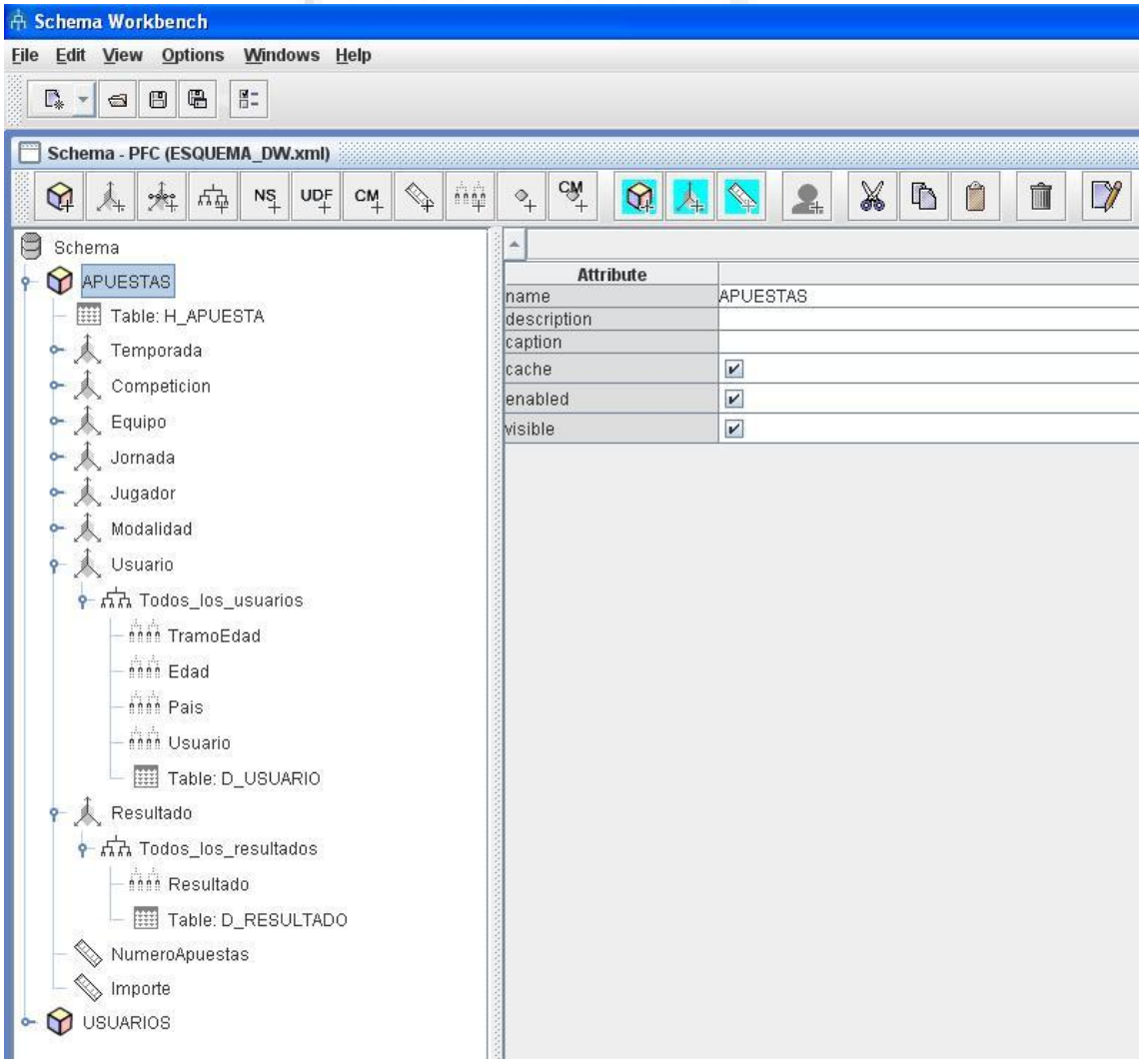


Ilustración 8: Definición del cubo de Apuestas con Pentaho Schema Workbench

El otro cubo tomará como tabla principal D_USUARIO, y permitirá realizar consultas sobre los usuarios tales como país de procedencia, o edad (cuyas dimensiones definiremos).

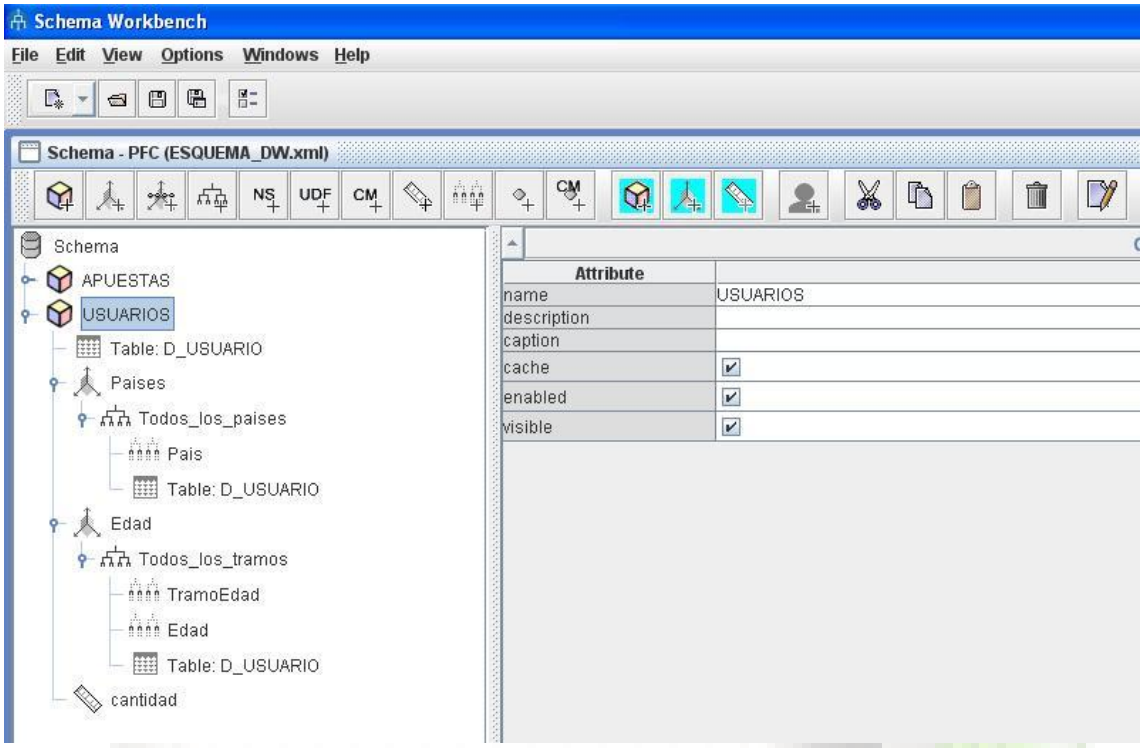


Ilustración 9: Definición del cubo de Usuarios con Pentaho Schema Workbench

Una vez definidos los cubos, ya podemos pasar a analizar los datos recogidos en el sistema

6.2 Análisis de los datos

Para esta tarea utilizaremos Saiku Analytics, que nos permitirá realizar todo tipo de consultas sobre los datos de forma sencilla y rápida.

6.2.1 Distribución geográfica de los usuarios y edad

Con Saiku Analytics es muy fácil obtener estadísticas a partir de un cubo creado. Simplemente tenemos que elegir las dimensiones y las medidas que queremos analizar y arrastrarlas a la casilla filas o columnas.

Primero de todo hemos seleccionado el nivel País de la dimensión de usuarios definida en el cubo USUARIOS.

Así obtenemos el número de usuarios por país:

Ilustración 10: Cantidad de Usuarios por País

También podemos obtener el resultado con un gráfico:

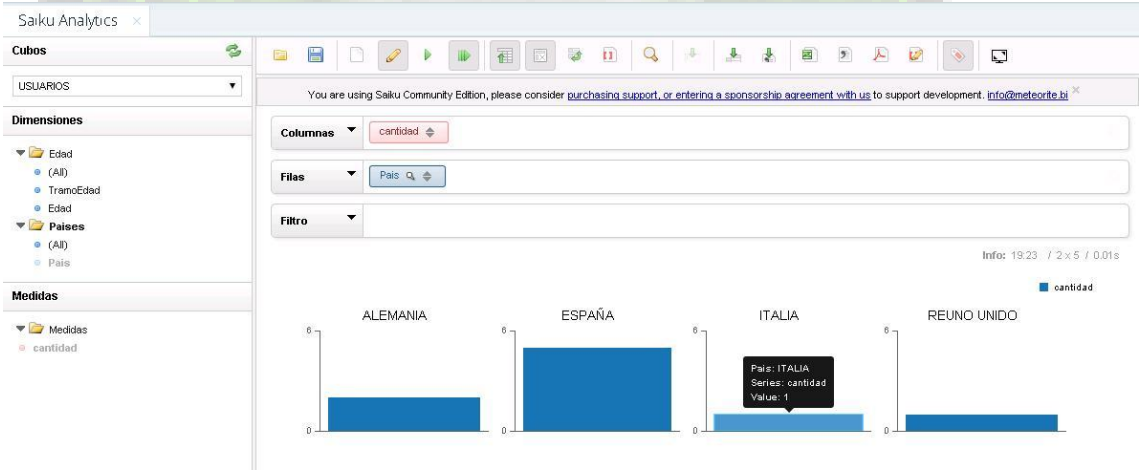


Ilustración 11: Cantidad de Usuarios por País (Gráfico)

Si añadimos el nivel Tramo Edad a la casilla de filas, tenemos la cantidad de usuarios por país y por tramos de edad:

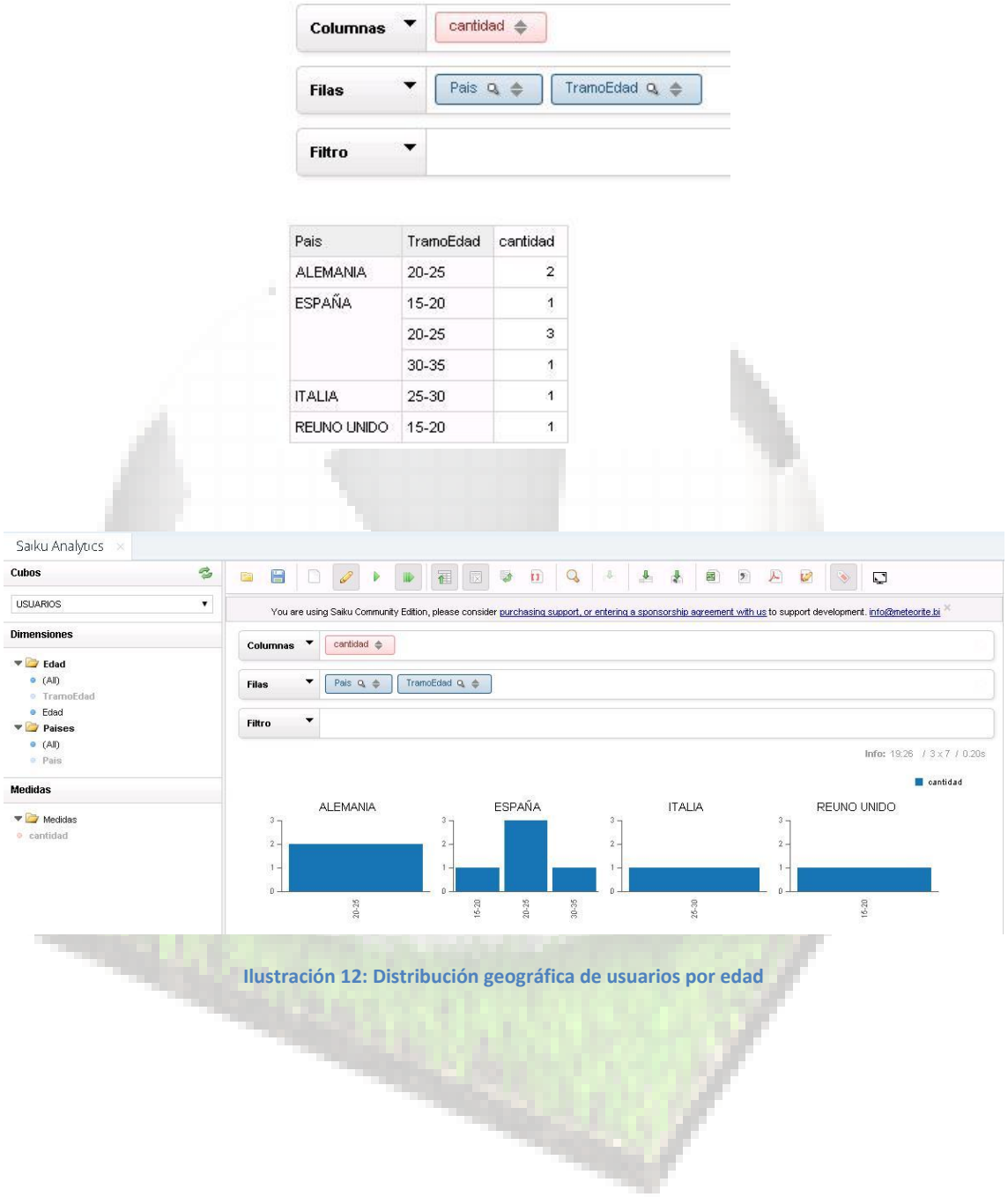


Ilustración 12: Distribución geográfica de usuarios por edad

6.2.2 Importe apostado por tramos de edad

Para este análisis basta con añadir la dimensión TramoEdad a la casilla de Filas, y el importe a la casilla de Columnas. A partir de aquí, podemos obtener los datos en una tabla, o un gráfico de barras donde ver el importe total de cada tramos de edad, o un gráfico circular que muestre el resultado por porcentajes.

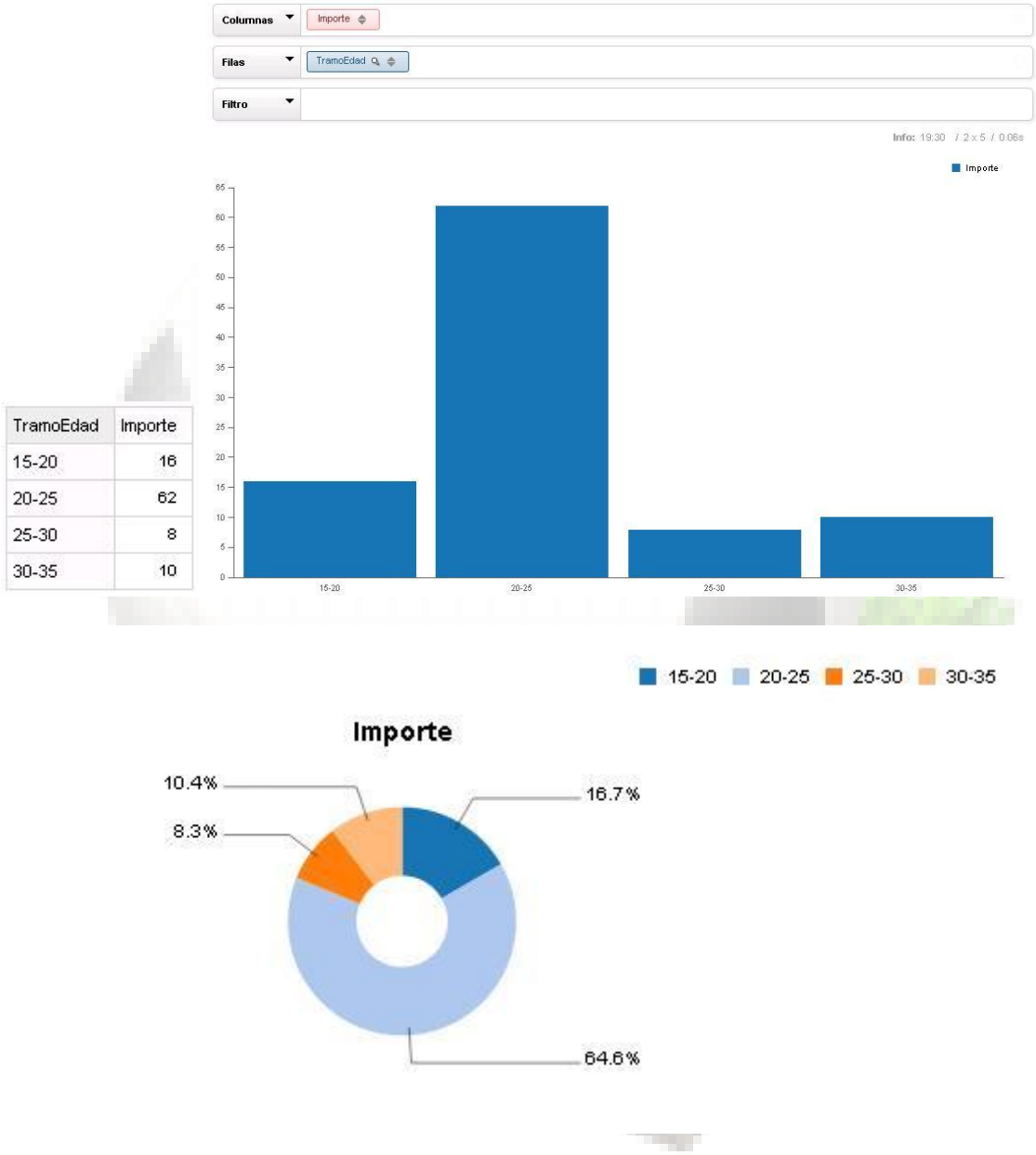
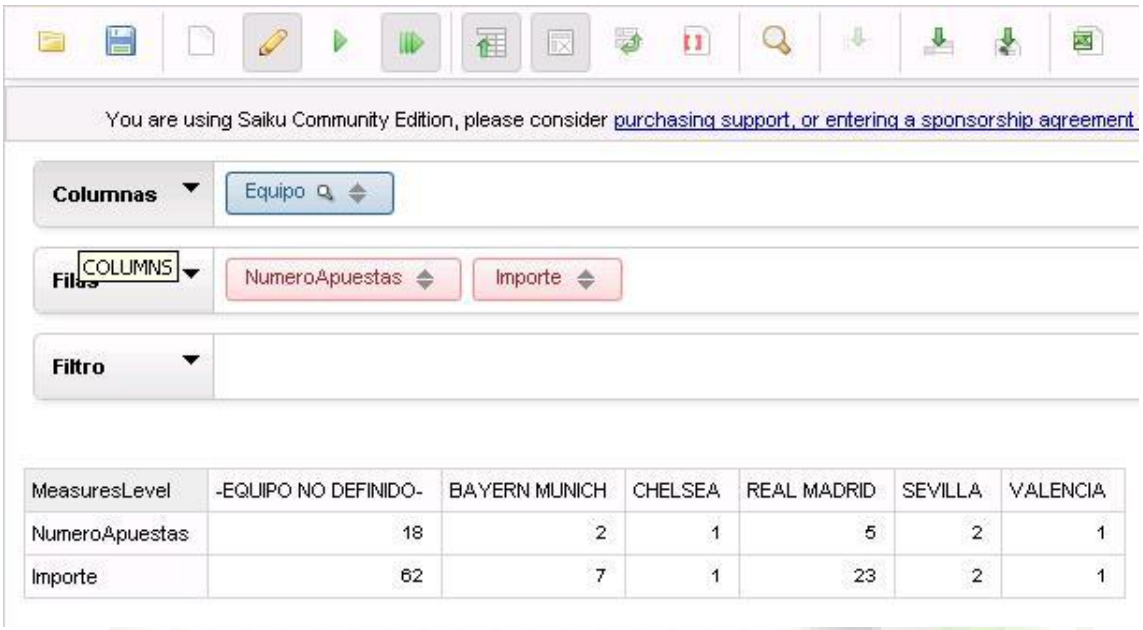


Ilustración 13: Importe apostado por tramos de edad

6.2.3 Equipos y jugadores por los que se apuesta más

Empezaremos el análisis por los equipos. Movemos las medidas Importe y número de apuestas a uno de los ejes, y la dimension jugador al otro, y obtenemos el siguiente resultado:



MeasuresLevel	-EQUIPO NO DEFINIDO-	BAYERN MUNICH	CHELSEA	REAL MADRID	SEVILLA	VALENCIA
NumeroApuestas	18	2	1	5	2	1
Importe	62	7	1	23	2	1

Ilustración 14: Equipos por los que se apuesta más. Tabla de datos

Aparecen todos los equipos junto con el equipo no definido que creamos para aquellas apuestas en las que no podiamos determinar el equipo por el que se apuesta.

Si obtenemos el gráfico, podemos suprimir los datos correspondientes al equipo genérico, que en este caso no nos interesan:

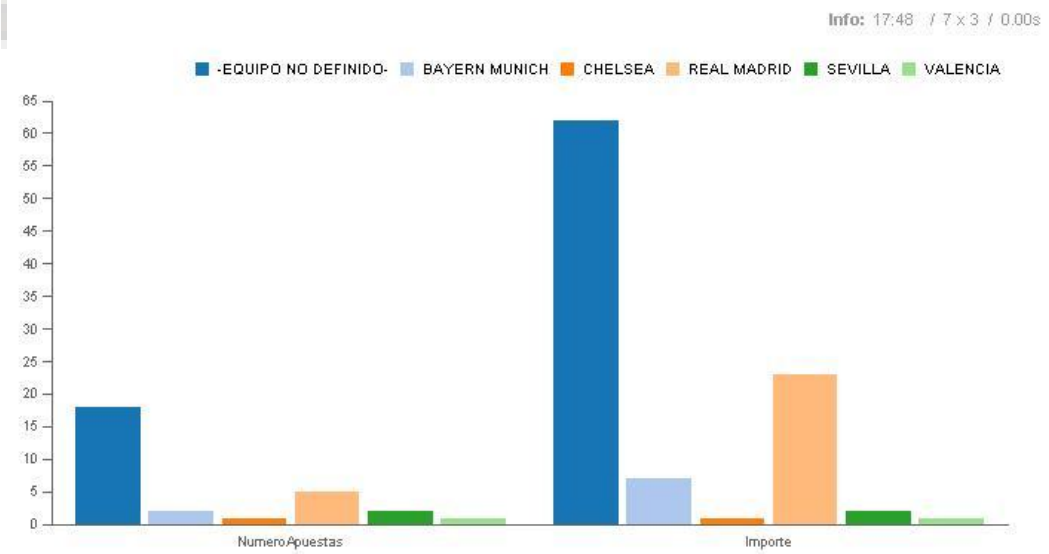


Ilustración 15: Equipos por los que se apuesta más. Gráfico I

También podemos obtener los datos en un gráfico de tarta, con los valores porcentuales para cada medida:

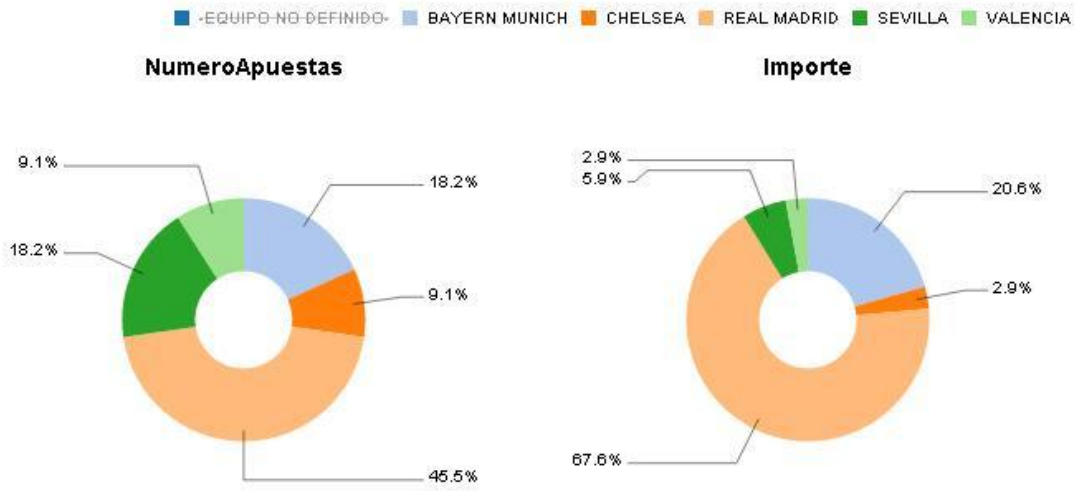
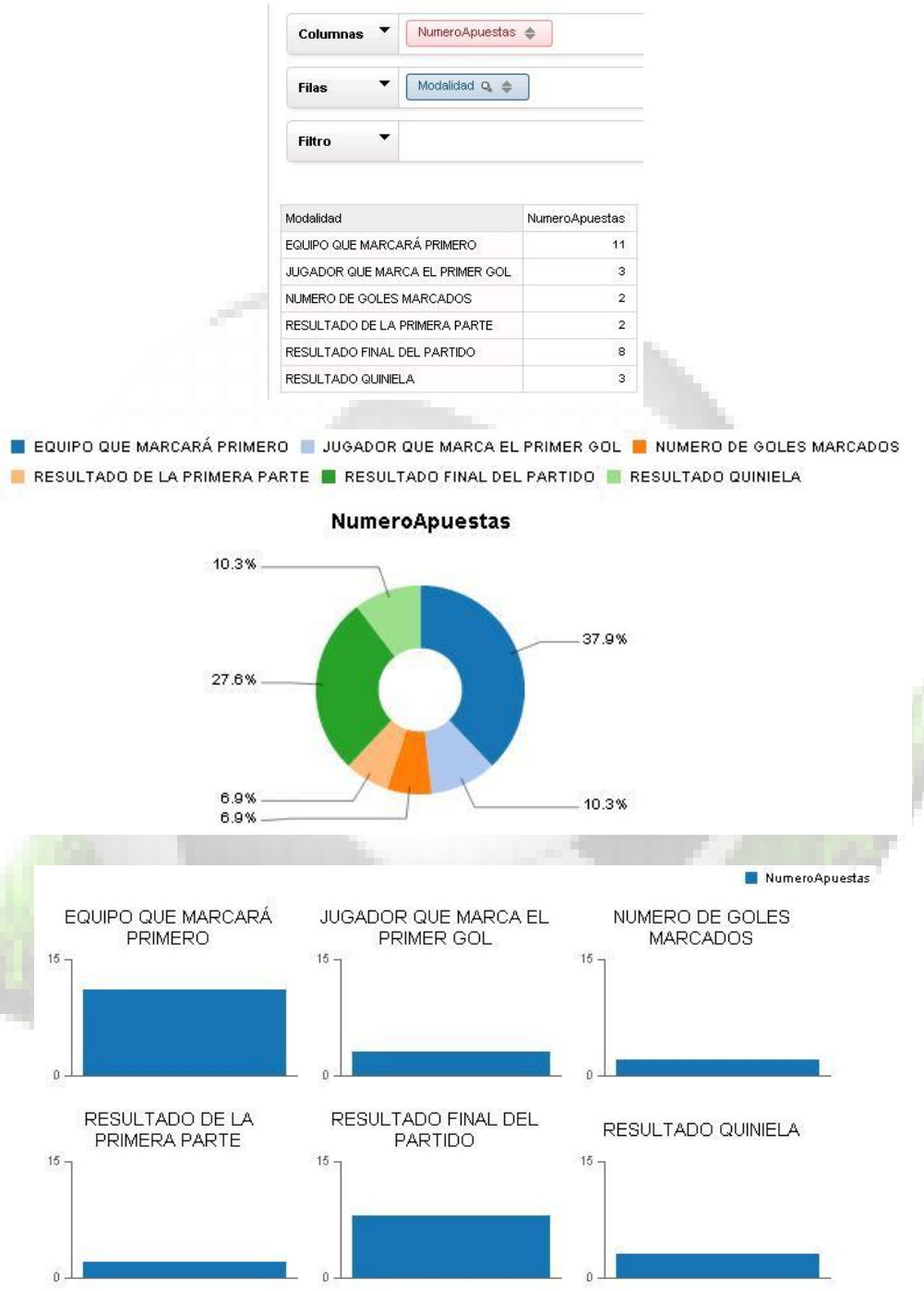


Ilustración 16: Equipos por los que se apuesta más. Gráfico II

Para sacar la estadistica de los jugadores procedemos igual:

Ilustración 17: Jugadores por los que se apuesta más

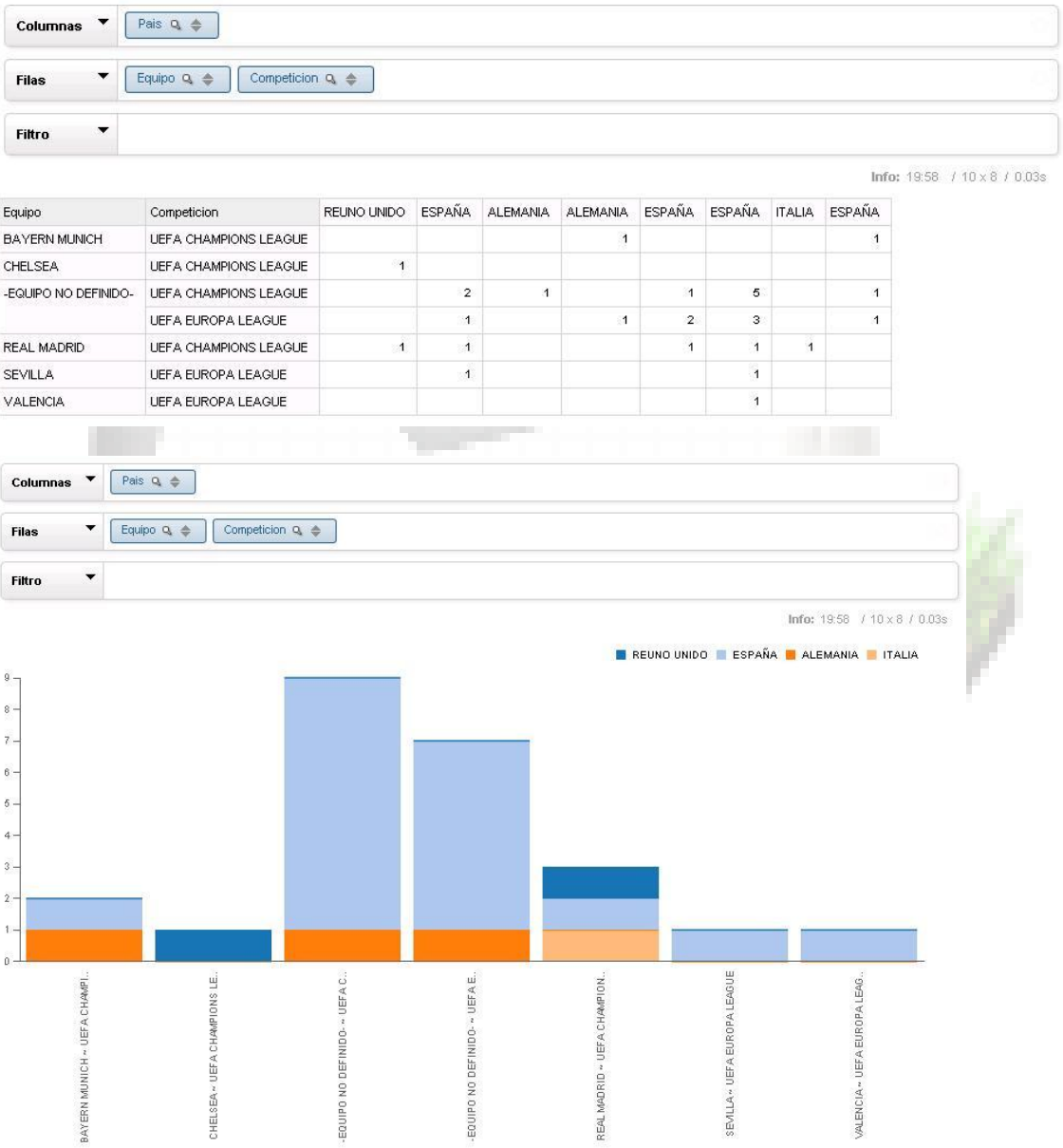
6.2.4 Modalidades con más apuestas



6.2.5 Distribución geográfica de los usuarios que apuestan por competición, por equipo.

Para obtener esta estadística, cruzamos los datos de varias dimensiones. En la casilla de filas ponemos el equipo y la competición, y en las columnas el país.

En este caso la gráfica apilada permite ver el resultado de una forma más legible que la tabla de datos obtenida.



Equipo	Competicion	REUNO UNIDO	ESPAÑA	ALEMANIA	ALEMANIA	ESPAÑA	ESPAÑA	ITALIA	ESPAÑA
BAYERN MUNICH	UEFA CHAMPIONS LEAGUE				1				1
CHELSEA	UEFA CHAMPIONS LEAGUE	1							
-EQUIPO NO DEFINIDO-	UEFA CHAMPIONS LEAGUE		2	1		1	5		1
	UEFA EUROPA LEAGUE		1		1	2	3		1
REAL MADRID	UEFA CHAMPIONS LEAGUE	1	1			1	1	1	
SEVILLA	UEFA EUROPA LEAGUE		1				1		
VALENCIA	UEFA EUROPA LEAGUE						1		

Competiciones con más apuestas

Para esta información primero obtenemos el número de apuestas por competición y modalidad:

	EQUIPO QUE MARCARÁ PRIMERO	JUGADOR QUE MARCA EL PRIMER GOL	NUMERO DE GOLES MARCADOS	RESULTADO DE LA PRIMERA PARTE	RESULTADO FINAL DEL PARTIDO	RESULTADO QUINIELA
Competicion	NumeroApuestas	NumeroApuestas	NumeroApuestas	NumeroApuestas	NumeroApuestas	NumeroApuestas
UEFA CHAMPIONS LEAGUE	8	2		2	4	2
UEFA EUROPA LEAGUE	3	1	2		4	1

Ilustración 18: Competiciones con más apuestas

Si añadimos la dimensión resultado podemos obtener el número de apuestas por modalidad, y la cantidad de aciertos y fallos:

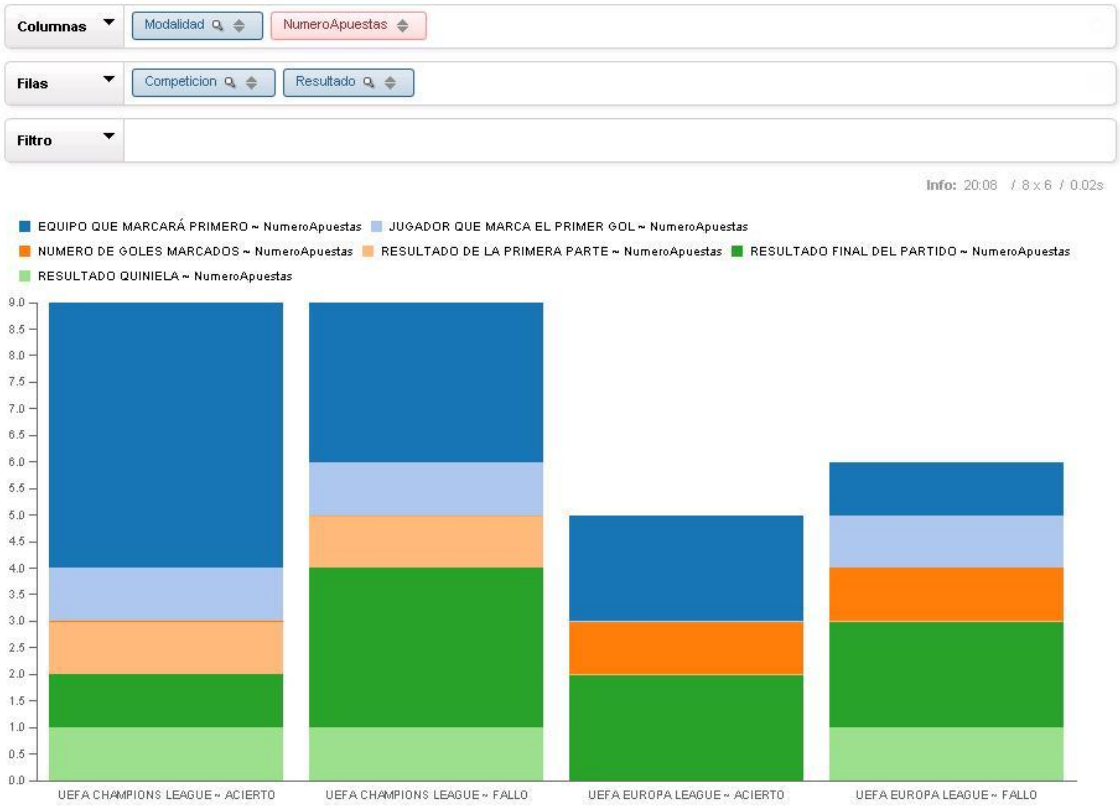


Ilustración 19: Competiciones con más apuestas y resultado

7 Valoración Económica del proyecto

Para calcular la valoración económica del proyecto, solamente tendremos en cuenta el coste en horas que hemos tardado en su desarrollo. No hacemos ninguna valoración sobre la inversión necesaria en infraestructuras, equipos, licencias de software, etc., dado que todos estos factores son variables. Lo que no cambia es el número de horas necesarias para llevarlo a cabo, ni el trabajo que hay que realizar.

Para este cálculo cogemos el detalle de tareas a realizar y separamos las tareas según el tipo de cualificación que debe tener la persona o el grupo de personas encargado de llevarlas a cabo.

Con estas premisas tenemos tres roles diferentes dentro del equipo de desarrollo: Jefe de Proyecto, Analista y Programador.

Si tenemos en cuenta que el proyecto se ha ajustado a la planificación marcada en el plan de proyecto, y que las jornadas de trabajo han tenido una duración de 3 horas cada una, el coste total del proyecto es el siguiente:

Jefe de Proyecto:	27 días de trabajo		
	3 horas cada día	→	81 horas
	60 euros/hora	→	TOTAL: 4.860 euros
Analista:	15 días de trabajo		
	3 horas cada día	→	45 horas
	50 euros/hora	→	TOTAL: 2.250 euros
Programador:	29 días de trabajo		
	3 horas cada día	→	87 horas
	40 euros/hora	→	TOTAL: 3.480 euros

Según estos cálculos el coste total del proyecto sería de **10.590** euros

8 Conclusiones

Mediante la realización de este trabajo hemos demostrado que los estudios de Ingeniería Informática nos han proporcionado las habilidades necesarias para planificar, gestionar y dirigir proyectos de cierta complejidad con resultados satisfactorios.

También hemos puesto en práctica los conocimientos adquiridos en las asignaturas de la rama de bases de datos cursadas a lo largo de toda la carrera, para poder desarrollar una solución que satisface unos requisitos planteados por el cliente, aplicando la metodología de desarrollo en cascada, y cumpliendo estrictamente la planificación inicial planteada.

Primero de todo realizamos una estimación inicial del trabajo a realizar y se detallaron todas las tareas en la planificación del Proyecto.

Seguidamente comenzamos el trabajo propiamente dicho con el análisis el diseño de los modelos de datos operacional y estadístico.

Una vez creados los objetos de la base de datos hemos implementado los procedimientos que permiten manipular y modificar los datos del modelo operacional, y los procedimientos de extracción, carga y transformación de datos en la base de datos estadística.

Finalmente, después de las pruebas de funcionamiento de las funcionalidades, hemos introducido un juego de datos para realizar las pruebas de explotación del modelo estadístico, y hemos mostrado como realizar la explotación de los datos.

Llegados a este punto podemos concluir que el proyecto se ha finalizado con éxito.

Este trabajo pone fin a cuatro años de dedicación, de tardes y fines de semana robados a mi familia, con la sensación gratificante de haber aprendido muchas cosas, de haber aprovechado el tiempo y de haber llegado por fin a la meta.

Quiero terminar expresando mi agradecimiento a todos los consultores que he tenido durante mi estancia en la Universidad. De todos ellos he tenido un trato correcto, amable, y siempre me han ayudado cuando los he necesitado.

Y también a todos aquellos compañeros con los que he intercambiado mensajes en los foros, y que muchas veces me han dado luz para poder superar las dificultades.

9 Glosario

Almacén de Datos (Datawarehouse): colección de datos orientada a un determinado ámbito (empresa, organización, etc.), integrado, no volátil y variable en el tiempo, que ayuda a la toma de decisiones en la entidad en la que se utiliza.

Base de Datos Operacional: es un contenedor de datos activos, es decir operacionales que ayudan al soporte de decisiones y a la operación. Su función es integrar los datos al igual que en el Datawarehouse pero con una ventana de actualización muy pequeña (del orden de minutos) y con mucho menos detalle

Base de datos: conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso

Bussiness Intelligence (Inteligencia empresarial): conjunto de estrategias y aspectos relevantes enfocadas a la administración y creación de conocimiento sobre el medio, a través del análisis de los datos existentes en una organización o empresa

Clave ajena: identifica una columna o grupo de columnas en una tabla (tabla hija o referendo) que se refiere a una columna o grupo de columnas en otra tabla (tabla maestra o referenciada). Las columnas en la tabla referendo deben ser la clave primaria u otra clave candidata en la tabla referenciada.

Clave primaria: campo o a una combinación de campos que identifica de forma única a cada fila de una tabla. Una clave primaria comprende de esta manera una columna o conjunto de columnas. No puede haber dos filas en una tabla que tengan la misma clave primaria.

Cubo OLAP (*OnLine Analytical Pressing* o procesamiento Analítico en Línea): es una base de datos multidimensional, en la cual el almacenamiento físico de los datos se realiza en un vector multidimensional. Los cubos OLAP se pueden considerar como una ampliación de las dos dimensiones de una hoja de cálculo

Dimensiones: Las tablas de dimensiones son elementos que contienen atributos (o campos) que se utilizan para restringir y agrupar los datos almacenados en una tabla de hechos cuando se realizan consultas sobre dicho datos en un entorno de almacén de datos

Diseño Conceptual: Es la primera fase del desarrollo de Bases de Datos. En esta etapa se obtiene una estructura de la información de la futura BD independiente de la tecnología que hay que emplear

Diseño Lógico: en esta etapa se parte del resultado del diseño conceptual, que se transforma de forma que se adapte a la tecnología que se debe emplear. Más concretamente, es preciso que se ajuste al modelo del SGBD con el que se desea implementar la base de datos.

Diseño Físico: se transforma la estructura obtenida en la etapa del diseño lógico, con el objetivo de conseguir una mayor eficiencia; además, se completa con aspectos de implementación física que dependerán del SGBD

Disparador (trigger): es un procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación. Dependiendo de la operación realizada en la base de datos, los triggers pueden ser de inserción (INSERT), actualización (UPDATE) o borrado (DELETE)

Integración: Las pruebas de integración (algunas veces llamadas integración y testeo I&t) es la fase de la prueba de software en la cual módulos individuales de software son combinados y probados como un grupo. Son las pruebas posteriores a las pruebas unitarias y preceden a las pruebas del sistema

Paquete: es un conjunto de funciones y procedimientos de software que suelen estar relacionados con la consecución de un objetivo común, o en la gestión conjunta de un grupo de datos.

PL/SQL: Es el lenguaje de programación que usa Oracle

Script: es un programa normalmente sencillo, o un conjunto de instrucciones que normalmente son almacenadas en un fichero de texto plano. En concreto en las bases de datos Oracle se utilizan este tipo de ficheros como interfaz fundamental la programación y desarrollo de bases de datos.

Sistema de Gestión de Bases de Datos: conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos, además de proporcionar herramientas para añadir, borrar, modificar y analizar los datos

10 Bibliografía

10.1 Bibliografia consultada

Josep Ramon Rodríguez, Pere Mariné Jové. *Gestió de Projectes*. UOC

Jaume Sistac i Planas. *Sistemes de gestió de base de dades*. UOC.

Àngels Rius Gavídia, Montse Serra Vizern, Alberto Abelló Gamazo, José Samos Jiménez, Josep Vidal Portolés, Josep Curto Díaz. *Datawarehouse. Magatzems de dades i models multidimensionals*. UOC

Rafael Camps Paré. *Sistemas de gestión de Bases de Datos*. UOC.

Dolors Costal Costa. *Introducció al disseny de bases de dades*. UOC

Pentaho Corporation. Pentaho open source bussiness intelligence. *Introducing the Pentaho BI Suite Community Edition*.

10.2 Direcciones Web

- Oracle: www.oracle.com
- Tutorial sobre Pentaho: http://pentaho.almacen-datos.com/tutorial.html
- Instalación de Pentaho: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=InstalacionPENTAHOBISuite
- Pasos para crear cubos con Schema Workbench: http://api.ning.com/files/L5YSaUwgABX4OaeGHjIc*sylZHnUzt3A8smqDv7mlAwVFmxyEZSnvj8CpjHgp-O-cefRiYMM*KB8HY26h6lXlKyfK6JvOVju/PasosparacrearCubosconSchemaWorkbench.pdf
- Wikipedia: es.wikipedia.org