Base de datos y repositorio estadístico de la Federación Internacional de Automovilismo

José Luis Rodríguez Gómez

Bases de datos relacionales

Consultor: Jordi Ferrer Duran

Data Entrega

06/2015

FICHA DEL TRABAJO FINAL

Título del trabajo:	Base de datos y repositorio estadístico para la Federación Internacional Automovilística.
Nombre del autor:	José Luis Rodríguez Gómez
Nombre del consultor:	Jordi Ferrer Duran
Fecha de entrega :	06/2015
Área del Trabajo Final:	Bases de datos
Titulación:	*Grado en Ingeniería Informática*

Resumen del Trabajo (máximo 250 palabras):

El presente TFG trata de la creación de una base de datos y un repositorio estadístico para, respectivamente, almacenar y analizar todos los datos posibles de las competiciones en las que participa. La Federación Internacional Automovilística es la entidad que encarga este proyecto y que estable sus requisitos.

La Base de datos operativa tiene que recoger los datos de todas las entidades que participan el las competiciones: pilotos, coches, clasificación, equipos, patrocinadores, fabricantes, componentes, circuitos, etc... También tiene que recoger los datos telemétricos, es decir, los datos que se obtienen a partir de sensores durante las carreras: temperatura, ...

Para abordar la creación de esta base de datos se realizó en primer lugar un diseño conceptual. Ese diseño conceptual se tradujo en un diseño lógico, que finalmente se implemento en un diseño físico.

El repositorio estadístico es una base de datos con finalidad analítica y debe dar respuesta a consultas muy concretas. Los datos de este repositorio se actualizan en tiempo real.

El diseño físico final se implementó en el sistema de gestión de bases de datos ORACLE y el lenguaje de programación PL. Para el cálculo de datos y la creación de las tablas del repositorio se han utilizado disparadores y procedimientos.

Para comprobar el funcionamiento del sistema se facilita un juego de datos suficientes y representativo para una carga inicial. También se facilita un juego

de pruebas que permiten verificar el funcionamiento de las restricciones, procedimientos y disparadores. Finalmente, se libera el producto final.

Abstract (in English, 250 words or less):

The main objective of this EOG Work is creating a database and a statistical repository for both storing and analyzing all possible data from the auto race in which the International Federation of Motoring (FIA) participates. The FIA is the entity responsible for this project and establishes your requirements.

The Operational Database is responsible for gathering all data involved the competitions: drivers, cars, sorting, teams, sponsors, manufacturers, components, circuits, etc ... It also has to collect telemetry data, i.e. the data obtained from sensors during races: temperature, speed, aceleration, etc…

In order to develop this database, it was made first a conceptual design. This conceptual design is translated into a logical design, which finally was implemented in a physical design.

Statistical repository is a database for analytical purposes and should respond to very specific queries. The data in this repository are updated in real time.

The final physical design was implemented in the Relational Database Management System ORACLE. The PL/SQL langage was the choosen programming language . For calculating data and creating the repository tables they have been used triggers and procedures.

To check the operation of the system there is provided a sufficient and representative set of data for an initial charge. Also it was provided a set of tests to verify several critical operation. Finally, the final product is released.

Palabras clave (entre 4 y 8):

Bases de datos relacionales

Almacén de datos

ORACLE

PL/SQL

Modelo relacional

Automovilismo

Índice de contenido

Tabla de contenidos

Índice de ilustraciones

1. Plan de trabajo

1.1 Contexto y justificación del Trabajo

La Federación Internacional de Automovilismo (a partir de ahora FIA), gracias a la inyección económica de un nuevo patrocinador, desea ampliar su Sistema de Información a todas las categorías de competición, dado que hasta ahora solo cubría las competiciones de Fórmula I.

El control de todas las competiciones permitirá a la FIA realizar un análisis de la evolución de los pilotos en todas las categorías y, así, poder aumentar la competitividad y profesionalidad.

El cometido del presente proyecto consiste en implementar un Sistema de Base de Datos que dé respuesta a esta necesidad y cuyo resultado final sea un Sistema de Control de todas las competiciones organizadas por la FIA.

1.2 Objetivos del Trabajo

El objetivo del trabajo es crear un Sistema de Bases de Datos que controle y permita analizar las competiciones automovilísticas de todas las categorías. En concreto, debe abarcar los tres aspectos fundamentales de las competiciones:

- Registro de las entidades participantes
- Registro de los resultados
- Registro de datos telemétricos

  El último punto es el más importante. Se entiende por telemetría la medición de los datos que los sensores incorporados a los vehículos envían en todo momento al Sistema. Esos datos permiten conocer el rendimiento del coche y por tanto del piloto, el estado de los distintos componentes, la temperatura de sus elementos mecánicos, el nivel de carburante en el depósito, etc...

1.3. Requisitos mínimos

En cuanto a requisitos mínimos el Sistema debe registrar las siguientes entidades distinguibles en este dominio, definiendo para cada una de ellas los atributos pertinentes:

- R1: Pilotos
- R2: Equipo

- R3: Fabricantes
- R4: Componentes
- R5: Coches
- R6: Patrocinadores
- R7: Carreras
- R8: Competiciones
- R9: Resultados
- R10: Datos telemétricos

Para cada una de estas entidades deben crearse los correspondientes procedimientos de alta, baja y modificación.

- Alta, baja y modificación de:
- Pilotos
- Equipos
- Fabricantes
- Componentes
- Coches
- Patrocinadores
- Carreras
- Competiciones
- Resultados
- Datos telemétricos

Otro requisito importante es la disponibilidad de un repositorio estadístico que permita ejecutar consultas y tener una visión rápida del panorama general. Los requisitos mínimos a que debe responder el repositorio estadístico son los siguientes:

- RE1: Porcentaje de componentes defectuosos de cada coche.
- RE2: Resultados de cada piloto.
- RE3: Lista de los 10 pilotos que han finalizado más carreras.

- RE4: Clasificación de cada competición.
- RE5: Para un circuito determinado, listar las 10 vueltas más rápidas.
- RE6: Temperatura más alta registrada.
- RE7: Top-5 de patrocinadores por aportación para una año dado.
- RE7: Datos telemétricos.
- RE8: Lista de los 5 fabricantes que aportan más componentes.
- RE9: Top-10 de pilotos por carreras ganadas.
- RE10: Piloto que ha tenido la mejor evolución entre dos momentos de tiempo dados.
- RE11: Coche con más consumo de carburante en un un año y competición dados.

Finalmente, con el objetivo de facilitar el mantenimiento del sistema, se requiere la implementación de un registro (l*og*) de las acciones llevadas a cabo por la Base de Datos. Esa tabla de log registrará al menos el nombre de los procedimientos, el momento en que son llamados. El procedimiento para el log deberá contar con un parámetro de salida que indique si la ejecución se ha realizado con éxito, o, en caso contrario, señale los errores. Dispondrá también de un tratamiento de excepcciones.

1.3 Enfoque y método seguido

Para el desarrollo del proyecto se ha optado por la metodología de la ingeniería de software denominada modelo en cascada. La principal característica de este tipo de desarrollo es que una determinada fase debe estar totalmente implementada y testada antes de pasar a la siguiente.

Para cada una de las fases se llevará a cabo un análisis de requisitos, el diseño, la implementación, la verificación con las pruebas oportunas, y la propuesta de mantenimiento, en particular, la implementación del log, cuando la fase concreta así lo requiere.

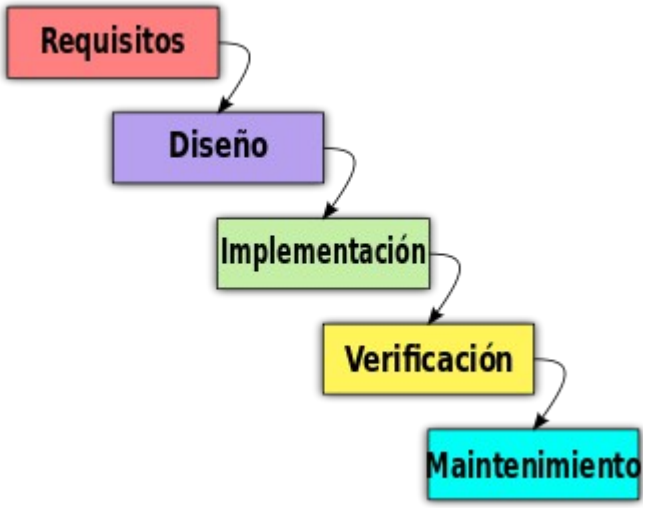


Ilustración 1: Desarrollo en cascada

Esta elección, frente a un desarrollo ágil , por ejemplo, se debe a que se trata de un TFG y los requisitos se supone que no van a ser modificados como ocurriría en un entorno real. Por otra parte, este modelo en cascada permitirá establecer un calendario más riguroso, conociendo en cada etapa el porcentaje de trabajo realizado.

1.4 Planificación del Trabajo

1.4.1. Tareas

Para la planificación se han tenido en cuenta las fechas propuestas en el calendario de la asignatura para la entrega de la PAC . Cada una de estas entregas constituye un hito en el calendario.

Título	Entrega
PAC1	09/03/15
PAC2	13/04/15
PAC3	11/05/15

Entrega final	15/06/15
Debate virtual	26/06/15

De acuerdo con la descripción de las PAC ya establecidas se han diseñado una serie de tareas para cada una de ellas con el objetivo de alcanzar una planificación viable. Se asigna, además, un número de horas aproximado a cada una de las tareas.

Tareas	Actividades	Horas
PAC1 Planificación	Contexto y justificación	2
	Objetivos	4
	Enfoque y método	2
	Planificación	4
	Tareas	6
	Incidencias y riesgos	2
	Diagrama de Gantt	5
	Redacción PAC1	5
PAC2 Diseño conceptual, lógico y físico	Análisis de los requisitos	6
	Instalación del software	25
	Definición entidades y relaciones	15
	Esquema conceptual UML	10
	Diseño lógico	15
	Diseño físico	15
	Revisión diseños	10
	Redacción PAC2	10
	Incorporación PAC2 a Memoria	4
PAC3 Procedimientos	Análisis procedimientos	6
	Procedimientos de ABM	15
	Resto de procedimientos	20
	Creación *log*	10
	Juegos de pruebas	10
	Test	10
	PAC3. Redacción y revisión	15

	Incorporación PAC3 a Memoria	4
Entrega final	Revisión final scripts	10
	Correcciones finales	30
	Redacción Memoria	40
	Autoinforme de las competencias transversales	5
	Entrega final, retoques	10
	Tiempo posibles incidencias	25
Debate virtual	Participación	5

Los cuatro principales hitos del TFG se han desarrollado, en líneas generales, de la siguiente manera:

a) PAC1 Planificación: Se entregó en la fecha prevista, 09/03/15. El consultor le concedió la calificación A.

b) PAC2 Diseño conceptual, lógico y físico. Se entregó en la fecha prevista, 13/04/15, y recibió la calificación B. Hubo un desvío importante respecto a la planificación inicial. A los apartado previstos, se añadieron los procedimiento ABA. Sin embargo no se abordó la parte de Repositorio estadístico. El consultor hizo observaciones oportunas sobre esta desviación, indicando que no llegaba al mínimo exigible en este punto.

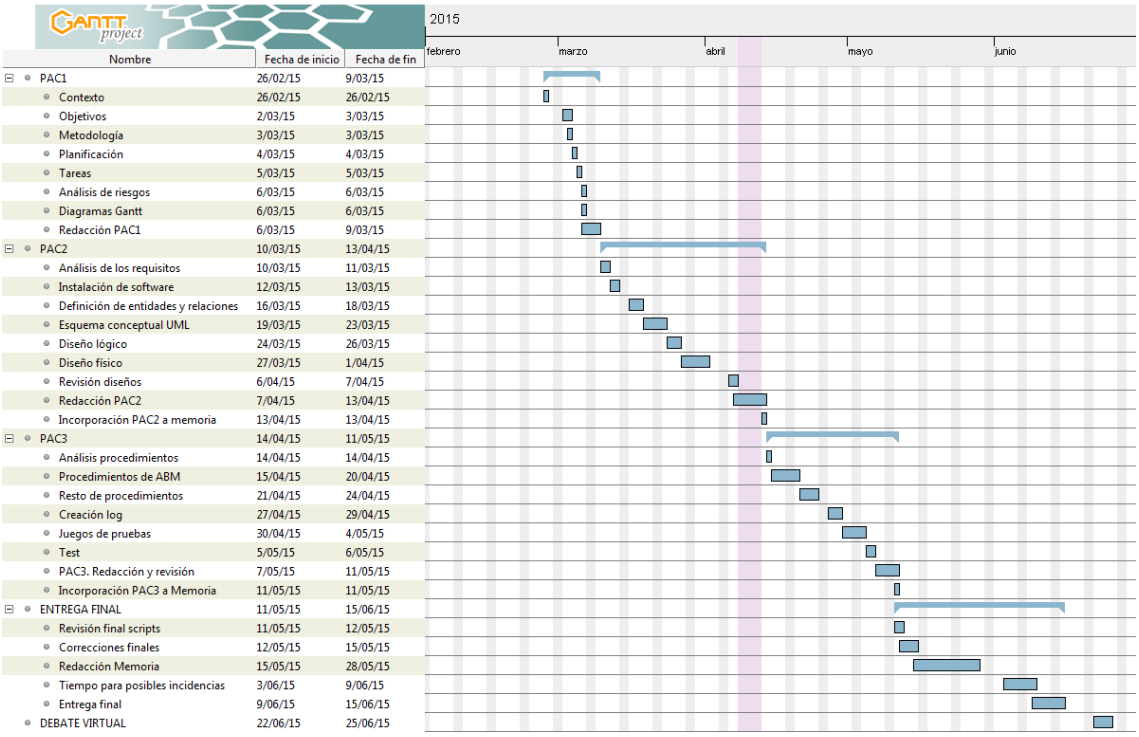
c) PAC3 Procedimientos. Se solicitó autorización al consultor para aplazar la entrega dos días, de forma que la entrega se hizo el 13/05. La evaluación fue negativa, C-. A los errores más formales, se añadían otros más relevantes, como por ejemplo la incompletud de la memoria: las explicaciones no eran suficientes para evaluar el estado del trabajo.

Para tratar de subsanar estos problemas, derivados en parte de un mal planteamiento, se realizaron contactos con en profesor para aclarar algunos puntos relativos al repositorio estadístico, quedándome claro que su actualización debía realizarse en tiempo real. A partir de ahí, he podido seguir un camino seguro y avanzar con mayor seguridad.

d) Entrega final. Con autorización del consultor se presentó el trabajó final un día después de la fecha prevista.

1.4.2. Diagramas Gantt

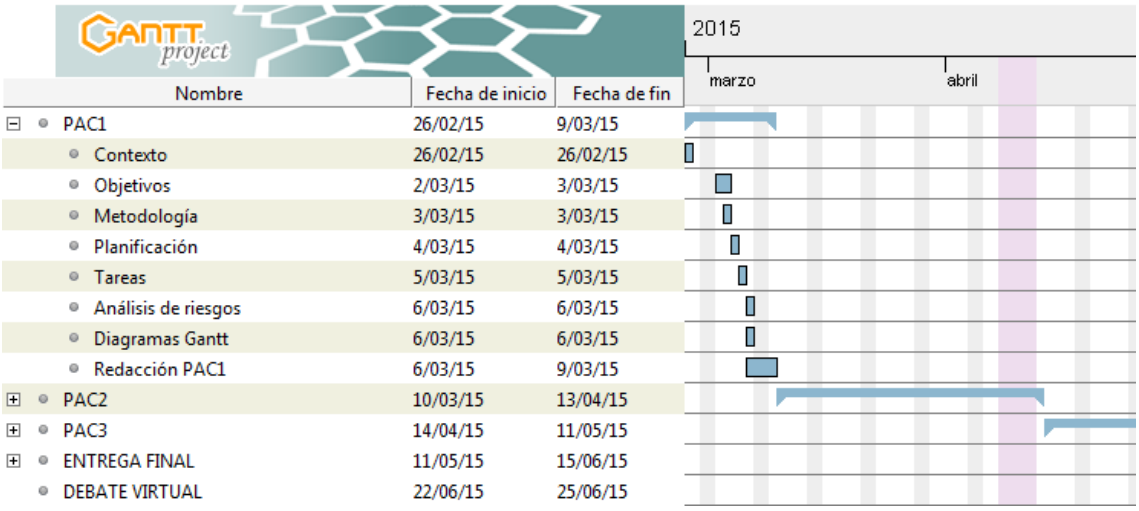
1.4.2.1. Diagramas Gantt del proyecto



Nombre	Fecha de inicio	Fecha de fin
PAC1	26/02/15	9/03/15
Contexto	26/02/15	26/02/15
Objetivos	2/03/15	3/03/15
Metodología	3/03/15	3/03/15
Planificación	4/03/15	4/03/15
Tareas	5/03/15	5/03/15
Análisis de riesgos	6/03/15	6/03/15
Diagramas Gantt	6/03/15	6/03/15
Redacción PAC1	6/03/15	9/03/15
PAC2	10/03/15	13/04/15
Análisis de los requisitos	10/03/15	11/03/15
Instalación de software	12/03/15	13/03/15
Definición de entidades y relaciones	16/03/15	18/03/15
Esquema conceptual UML	19/03/15	23/03/15
Diseño lógico	24/03/15	26/03/15
Diseño físico	27/03/15	1/04/15
Revisión diseños	6/04/15	7/04/15
Redacción PAC2	7/04/15	13/04/15
Incorporación PAC2 a memoria	13/04/15	13/04/15
PAC3	14/04/15	11/05/15
Análisis procedimientos	14/04/15	14/04/15
Procedimientos de ABM	15/04/15	20/04/15
Resto de procedimientos	21/04/15	24/04/15
Creación log	27/04/15	29/04/15
Juegos de pruebas	30/04/15	4/05/15
Test	5/05/15	6/05/15
PAC3. Redacción y revisión	7/05/15	11/05/15
Incorporación PAC3 a Memoria	11/05/15	11/05/15
ENTREGA FINAL	11/05/15	15/06/15
Revisión final scripts	11/05/15	12/05/15
Correcciones finales	12/05/15	15/05/15
Redacción Memoria	15/05/15	28/05/15
Tiempo para posibles incidencias	3/06/15	9/06/15
Entrega final	9/06/15	15/06/15
DEBATE VIRTUAL	22/06/15	25/06/15

Ilustración 2: Diagrama Gantt general

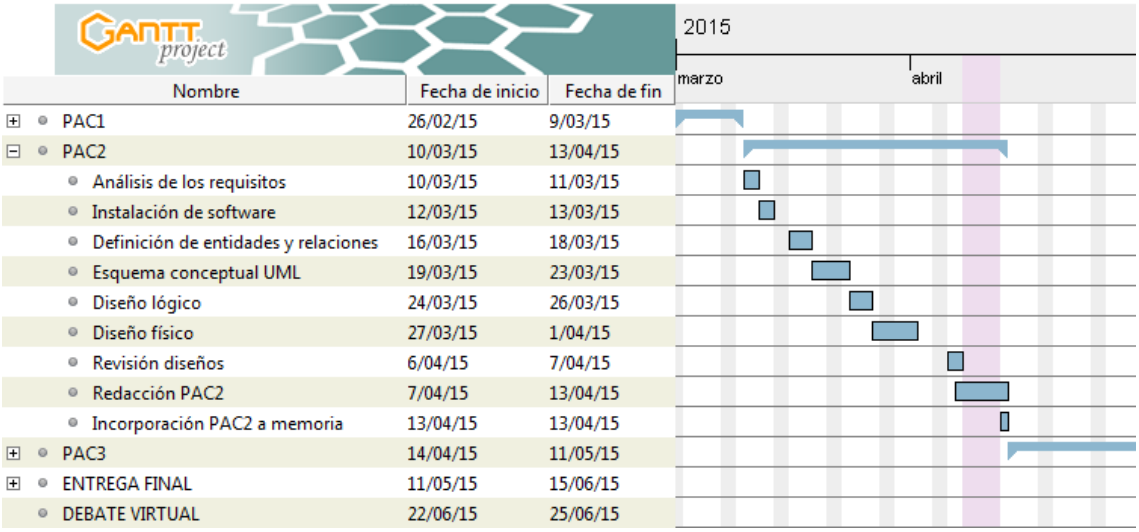
1.4.2.2. Diagramas Gantt de la PAC1



Nombre	Fecha de inicio	Fecha de fin
PAC1	26/02/15	9/03/15
Contexto	26/02/15	26/02/15
Objetivos	2/03/15	3/03/15
Metodología	3/03/15	3/03/15
Planificación	4/03/15	4/03/15
Tareas	5/03/15	5/03/15
Análisis de riesgos	6/03/15	6/03/15
Diagramas Gantt	6/03/15	6/03/15
Redacción PAC1	6/03/15	9/03/15
PAC2	10/03/15	13/04/15
PAC3	14/04/15	11/05/15
ENTREGA FINAL	11/05/15	15/06/15
DEBATE VIRTUAL	22/06/15	25/06/15

Ilustración 3: Diagrama Gantt PAC1

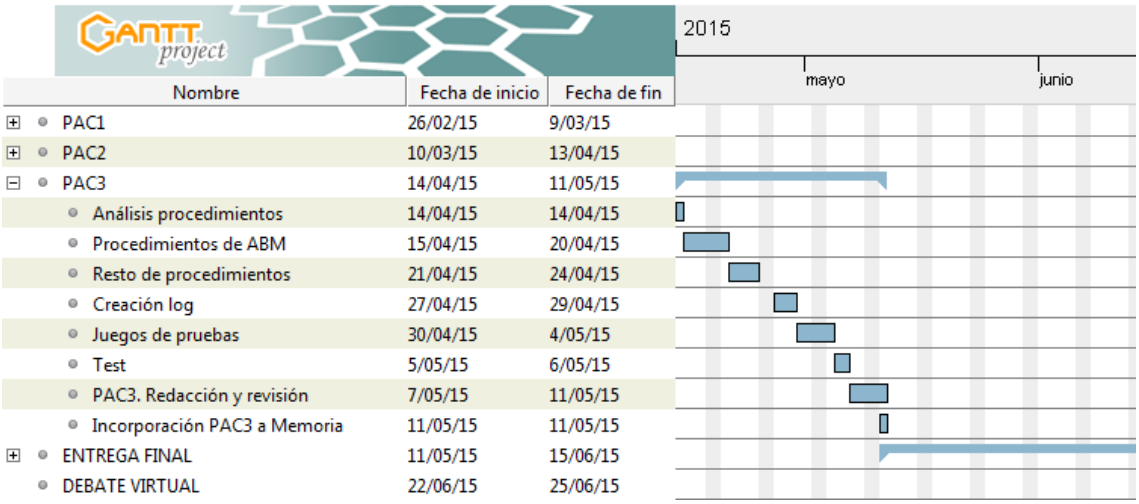
1.4.2.3. Diagramas Gantt de la PAC2



Nombre	Fecha de inicio	Fecha de fin
PAC1	26/02/15	9/03/15
PAC2	10/03/15	13/04/15
Análisis de los requisitos	10/03/15	11/03/15
Instalación de software	12/03/15	13/03/15
Definición de entidades y relaciones	16/03/15	18/03/15
Esquema conceptual UML	19/03/15	23/03/15
Diseño lógico	24/03/15	26/03/15
Diseño físico	27/03/15	1/04/15
Revisión diseños	6/04/15	7/04/15
Redacción PAC2	7/04/15	13/04/15
Incorporación PAC2 a memoria	13/04/15	13/04/15
PAC3	14/04/15	11/05/15
ENTREGA FINAL	11/05/15	15/06/15
DEBATE VIRTUAL	22/06/15	25/06/15

Ilustración 4: Diagrama Gantt PAC2

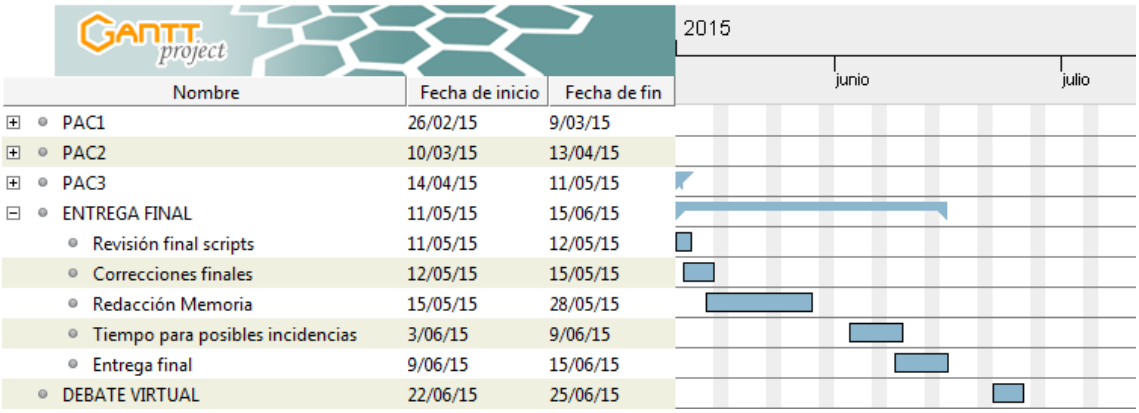
1.4.2.4. Diagramas Gantt de la PAC3



Nombre	Fecha de inicio	Fecha de fin
PAC1	26/02/15	9/03/15
PAC2	10/03/15	13/04/15
PAC3	14/04/15	11/05/15
Análisis procedimientos	14/04/15	14/04/15
Procedimientos de ABM	15/04/15	20/04/15
Resto de procedimientos	21/04/15	24/04/15
Creación log	27/04/15	29/04/15
Juegos de pruebas	30/04/15	4/05/15
Test	5/05/15	6/05/15
PAC3. Redacción y revisión	7/05/15	11/05/15
Incorporación PAC3 a Memoria	11/05/15	11/05/15
ENTREGA FINAL	11/05/15	15/06/15
DEBATE VIRTUAL	22/06/15	25/06/15

Ilustración 5: Diagrama Gantt PAC3

1.4.2.5. Diagramas Gantt de la ENTREGA FINAL



Nombre	Fecha de inicio	Fecha de fin
PAC1	26/02/15	9/03/15
PAC2	10/03/15	13/04/15
PAC3	14/04/15	11/05/15
ENTREGA FINAL	11/05/15	15/06/15
Revisión final scripts	11/05/15	12/05/15
Correcciones finales	12/05/15	15/05/15
Redacción Memoria	15/05/15	28/05/15
Tiempo para posibles incidencias	3/06/15	9/06/15
Entrega final	9/06/15	15/06/15
DEBATE VIRTUAL	22/06/15	25/06/15

Ilustración 6: Diagrama Gantt: Entrega final

1.4.3. Recursos materiales

1.4.3.1 Hardware

PC de sobremesa

HP Pavilion 500-214es
Windows 8.1 64

Procesador Intel® Core™ i7-4770
Gráficos NVIDIA GeForce GT 640M (DDR3 dedicada de 4 GB)
Memoria DDR3 de 12 GB
Disco duro SATA de 1 TB

Ordenador portátil

Ultrabook ThinkPad X1 Carbon
Windows 8.1 64
Procesador Intel Core i5-5200U
Memoria total: 4GB PC3-12800 on MB

Para guardar copias de seguridad se ha utilizado un disco duro con las siguientes características:

Puerto USB 3.0.
Capacidad: 1000GB.
Velocidad de transferencia de datos: 5000 Mbit/s.
Tasa de transferencia de datos USB: 5000 Mbit/s.

1.4.3.2. Software

MagicDraw UML

Herramienta CASE de No Magic que utilizaré para modelación y representación en UML.

Oracle Application Express (APEX)

Sistema de gestión de bases de datos que utilizaré para el proyecto.

Oracle SQL Developer

Entorno de desarrollo integrado que facilita la gestión del SIGBD de Oracle.

DBDesigner

Sistema visual de diseño y modelación de bases de datos.

GanttProject

Herramienta para la creación de diagramas Gantt.

Microsoft Office

Paquete ofimático.

Adobe Photoshop CS4

Tratamiento de imágenes.

Adobe Acrobat Professional

Tratamiento y generación de archivos pdf.

1.4.4. Análisis de riesgos

Incidencia	Riesgo	Impacto	Contingencia
Enfermedad leve	Bajo	Bajo	Compensar en fines de semana
Viajes de trabajo Alto	Bajo		Compensar con trabajo en portátil durante el viaje, estancia, huecos, etc...
Avería de equipos	Bajo	Bajo	Sustitución inmediata
Pérdida de datos	Bajo	Bajo	Restaurar copia de seguridad

1.5. Sumario de productos obtenidos

El producto final se presenta en los siguientes ficheros:

00_INSTALACION_FICHERO_UNICO .sql.	Contiene en un único fichero todos los scripts y los datos necesarios: tablas, restricciones, secuencias, procedimientos, paquetes, disparadores y datos.
01_CREATE_TABLES.sql	Contiene todas las tablas y secuencias de la base de datos.
02_DATOS.sql	Contiene los datos.
03_RESTRICCIONES.sql	Contiene las restricciones de las tablas: claves primarias, clave ajena, etc..
04_TRIGGERS.sql	Contiene todos los triggers de la base de datos.
05_PAQUETES.sql	Contiene los paquetes (especificación y bodY) de actualización de repositorio (PKG_ESTADISTICA), de Consultas al repositorio (PKG_CONSULTAS) y un paquete con el procedimiento para ejecutar la batería de pruebas

	(PKG_JUEGO_PRUEBAS)
06_PROCEDIMIENTOS_ABA.sql	Contiene los procedimientos de Alta, Baja y Modificación

1.6. Descripción de los capítulos de la memoria

En el capítulo 1 se expone el plan de trabajo, que incluye los objetivos, la metodología, los requisitos y la planificación. Las líneas fijadas en este capítulo estarán presentes a lo largo de toda la memoria. En él se determinan las tareas y se fijan los hitos más importantes, que son las entregas de PAC y la entrega final.

En el capítulo 2, partiendo del análisis de requisitos, se describen los diseños conceptual, lógico y físico de la base de datos operativa, la que se encargará de las operaciones de actualización. Se muestran también los procedimientos de Alta, Baja y Modificación (ABA) creados para la actualización de todas las tablas que lo requieran. Se dedica un apartado a los disparadores que se utilizan para realizar cálculos necesarios y llenar con ellos las columnas de algunas tablas.

Del repositorio se ocupa el capítulo 3. Se mostrarán las tablas que lo componen, que se corresponden con las consultas que constituyen los requisitos del repositorio estadístico. Se exponen los procedimientos creados para su actualización en tiempo real y también los disparadores que permiten llenar con datos pre-calculados algunas tablas. Finalmente se describen los procedimientos de Consulta.

El capítulo 4 trata de la carga inicial de la base de datos y expone los juegos de pruebas que permitirán verificar el comportamiento de todo el sistema, desde los procedimientos ABA a las consultas del repositorio estadístico.

Los capítulos 5 y 6 se ocupan, respectivamente, de la valoración económica y del glosario.

El capítulo 7 contiene el listado de referencias bibliográficas, y en el 8 se ofrecen los anexos, normalmente fragmentos de código que ejemplifican la programación de los distintos componentes del sistema.

2. Base de datos operacional

2.1. Análisis de requisitos

2.1.1. Entidades

Se trata de crear un SGBD que permite almacenar los datos relativos a las distintas entidades que participan en esa realidad compleja que constituyen los campeonatos.

En concreto, como mínimo el sistema debe dar cuenta de las siguientes entidades:

- Piloto: De cada piloto debe guardarse como mínimo el nombre, la nacionalidad y el código de licencia.
- Equipo: De cada equipo debe guardarse el código de equipo, el nombre, la sede, la fecha de debut, y el nombre del manager
- Fabricante: De cada fabricante debe guardarse el NIF, el nombre, la fecha de inicio en el mundo de la competición
- Componente: De cada componente debe guardarse el código, la descripción, utilidad.
- Coche: De cada coche debe guardarse el código, el modelo, la fecha de la primera carrera
- Patrocinador: De cada patrocinador debe guardarse el código, el nombre, el sector económico al que pertenece
- Carrera: De cada carrera debe guardarse el código, la fecha, la longitud del circuito
- Competición: De cada competición debe guardarse el código, la descripción, la fecha de inicio y fin.
- Resultado: De los resultados se guardará el código de resultado, el código del coche, la posición, y los puntos obtenidos. En una carrera el que ocupa el puesto primero obtiene 25 puntos, el segundo 15 y el tercero 5. Los restantes coches obtiene 0.

- Datos telemétricos: Se guardará el código, el instante con una precisión de nanosegundos, el ámbito de la medición, el valor y la unidad de medida.
- Al margen de estos requisitos, añadimos las siguientes entidades:
- Temporada: Simplemente indicamos el año, con cuatro dígitos.
- Circuito: De cada circuito, se guardará el identificador, el nombre y la extensión.

2.1.2 Procedimientos

Todas las modificaciones realizadas en la base de datos se llevarán a cabo mediante procedimientos. En esos procedimientos almacenados se definirán las operaciones de Alta, Baja y Modificación (ABA) para que aquellas entidades que lo requieran.

2.1.3 Registro de operaciones

Finalmente, con el objetivo de facilitar el mantenimiento del sistema, se requiere la implementación de un registro (l*ogs*) de las acciones llevadas a cabo por la Base de Datos. En ese registro constará al menos el nombre de la operación ejecutada, la fecha e instante de su realización y, en caso de fallo, los mensajes de error correspondientes.

En los procedimientos se incluirá un parámetro de salida, RSP, de tipo cadena, que devolverá, si la operación finalizó con éxito, un mensaje -Valor OK y la operación realizada-, y el error y tipo, en caso contrario.

2.2. Diseño conceptual

En el planteamiento del problema se nos ofrece la historia de un dominio de una realidad compleja que denominaremos FIA, en el que interviene un buen número de entidades y relaciones. A partir de esta historia, que está en el nivel más bajo, es decir, más cercano a la realidad, trataremos de crear un sistema.

La creación de este sistema tiene como objetivo aislar las entidades relevantes de la historia y mostrar sus relaciones. El punto de partida lo constituyen las entidades que hemos enumerado en el apartado anterior y que nos han sido dadas como requisitos mínimos. Del análisis de esos requisitos y de los del repositorio estadístico, surgen entidades adicionales que deben completar ese esquema.

Para el este diseño conceptual se han tenido en cuenta materiales de la UOC, especialmente VV.AA [8] y la aportación más general y teórica de Silberschatz, A. [4].

Se detallan a continuación todas las entidades consideradas junto con sus interrelaciones.

Se ponen *en cursiva* los atributos que obtendré a través de disparadores. Estos atributos contienen datos que serán útiles para llenar las tablas del repositorio estadístico. En el apartado correspondiente se describirán los disparadores creados.

2.2.1. Entidades

PILOTO: código de licencia, nombre, nacionalidad, *carreras ganadas*, *carreras acabadas*.

EQUIPO: código de equipo, nombre, sede, debut, manager.

FABRICANTE: NIF, nombre, fecha de inicio en el mundo de la competición.

COMPONENTE: código, descripción, utilidad.

COCHE: código, modelo, fecha de la primera carrera.

PATROCINADOR: código, nombre, sector.

CARRERA: código, fecha.

CATEGORÍA: denominación, descripción.

CIRCUITO: código, descripción, longitud, país

COMPETICIÓN: código, descripción, país organizador

PAÍS: código, país.

ABANDONO. Resultado, motivo del abandono.

Temporada: Año.

2.2.2. Entidades asociativas

CONTRATO: Clase asociativa. A la asociación Piloto – Equipo se añaden los atributos temporada, descripción y. Utilizo clave primaria compuesta de cod_piloto, temporada.

Para cumplir la restricción de que solo pueda haber dos pilotos en un equipo durante una temporada, he añadido una condición en los procedimientos que afectan a las Altas de esta tabla. El procedimiento será descrito en el apartado correspondiente.

COMPETICIÓN ANUAL: Entidad asociativa que depende de Competición y temporada. Añade a la interrelación los atributos Inicio, fin.

RESULTADO. Es una entidad asociativa entre Piloto, Carrera y Coche. Añade como atributos la posición, los puntos obtenidos, y la velocidad de la vuelta más rápida al circuito. En una carrera el que ocupa el puesto primero obtiene 25 puntos, el segundo 15 y el tercero 5. Los restantes coches obtiene 0 puntos.

TELEMETRÍA. Entidad asociativa entre Resultado y componente montado. Añade la unidad de medida y el ámbito.

2.2.2. Interrelaciones 1:N, N:1

CARRERA – CIRCUITO: 1:N. Una carrera se celebra en un solo circuito; en un circuito se pueden celebrar múltiples carreras. Al ser una relación 1:N, se resuelve añadiendo a una entidad una clave foránea, como veremos en el modelo lógico.

CARRERA – COMPETICIÓN ANUAL: N:1. Una competición anual se compone de múltiples carreras; una carrera pertenece a una sola competición anual. Al ser una relación N:1 se añadiendo a una entidad una clave foránea, como veremos en el modelo lógico.

CIRCUITO – PAÍS: N:1. En una país puede haber varios circuitos; un circuito está en un solo país. Al ser una relación N:1 se resuelve añadiendo a una entidad una clave foránea, como veremos en el modelo lógico.

COCHE – EQUIPO: 1:N. Un coche solo puede pertenecer a un Equipo. Un Equipo posee múltiples coches. Al tratarse de una relación 1:N se resuelva añadiendo a una entidad una clave foránea, como veremos en el modelo lógico.

COMPETICIÓN – PAÍS: N:1. En país puede celebrarse múltiples competiciones. Una competición es organizada por un solo país. Al tratarse de una relación 1:N se resuelva añadiendo a una entidad una clave foránea, como veremos en el modelo lógico.

COMPETICIÓN – CATEGORÍA. Una competición puede pertenecer a una sola categoría; puede haber múltiples competiciones de una sola categoría. Al tratarse de una relación 1:N se resuelva añadiendo a una entidad una clave foránea, como veremos en el modelo lógico.

COMPETICIÓN ANUAL – COMPETICIÓN. 1:N. Una competición tiene múltiples ediciones a lo largo de los años, una por año. Una competición anual pertenece a una única competición. Al tratarse de una relación 1:N se resuelva añadiendo a una entidad una clave foránea, como veremos en el modelo lógico.

COMPETICIÓN ANUAL – TEMPORADA. 1:N. Una competición anual se celebra en una sola temporada. En una temporada se celebran múltiples ediciones anuales. Al tratarse de una relación 1:N se resuelva añadiendo a una entidad una clave foránea, como veremos en el modelo lógico.

DATOS TELEMÉTRICOS: 1:N. Una telemetría puede tener infinitos datos telemétricos. Un dato telemétrico corresponde a una telemetría. Al tratarse de una relación 1:N se resuelva añadiendo a una entidad una clave foránea, como veremos en el modelo lógico.

2.2.3. Interrelaciones N:M

Estas interrelaciones múltiples dan lugar a una nueva entidad. A continuación damos el nombre de esa nueva entidad, su descripción, y, cuando sea necesario, algunas observaciones.

PATROCINIO: Recoge la interrelación entre Patrocinador y Equipo. Un patrocinador puede patrocinar a múltiples equipos, y un equipo puede ser patrocinado por múltiples patrocinadores. Hemos hecho la restricción de que un patrocinador solo pueda patrocinar a un equipo en un mismo año. Para ello, hemos añadido es esta relación Temporada como atributo (clave foránea), y hemos puesto como clave primaria Equipo y Patrocinador

COMPONENTE_MARCA: Recoge la interrelación entre Componente y Fabricante. Un componente puede ser fabricado por múltiples fabricantes, y un fabricante puede fabricar múltiples componentes.

COMPONENTE_MONTADO: Recoge la interrelación entre Componente_Marca y Coche, es decir, identifica al componente que está realmente instalado en el coche, y del que deberemos obtener los datos telemétricos.

2.2.4. Diagrama de clases

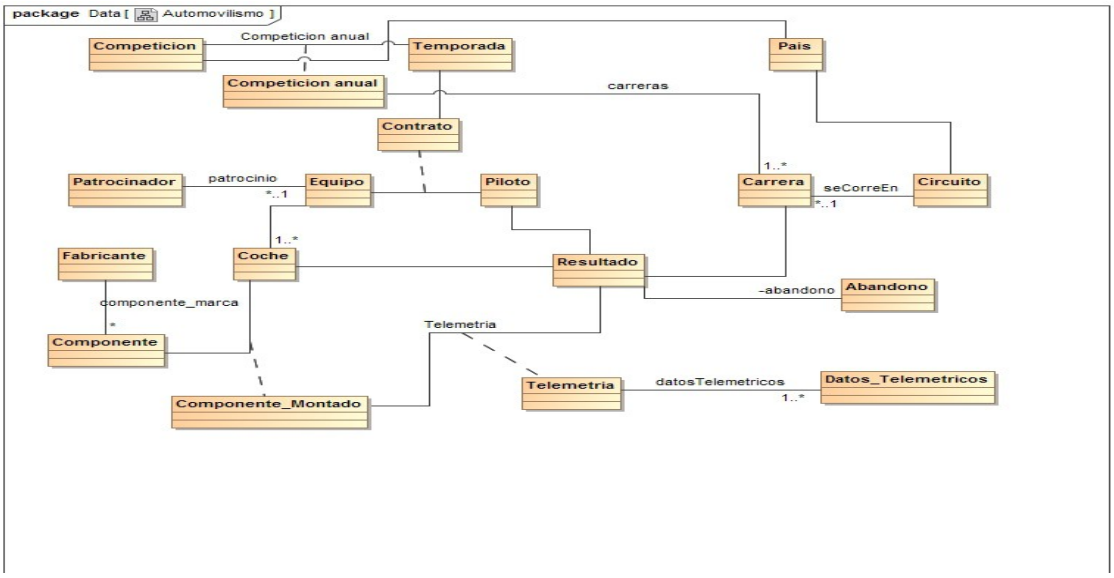


Ilustración 7: Diagrama de clases

2.3. Diseño lógico

Una vez establecido el diseño conceptual, pasamos a describir el diseño lógico. En este diseño se transforman las interrelaciones a claves foráneas, en el caso de las interrelaciones de tipo 1:N, y a nuevas entidades en el caso de las interrelaciones múltiples N:M.

ABANDONO: resultado, motivo

Resultado es clave foránea que referencia Resultado

CARRERA: cod_carrera, fecha, circuito, cod_competicion

Circuito es clave foránea que referencia Circuito

Cod_competicion referencia competicion_anual

CATEGORIA: cod_categoria, denominacion, descripcion

CIRCUITO: idCircuito, denominacion, dimensiones, cod_pais

Pais es clave foránea que referencia Pais

COCHE: Cod_coche, modelo, debut, cod_equipo

Cod_equipo es clave foránea que referencia Equipo

COMPETICIÓN: cod_competicion, descripcion, cod_pais, categoria

Cod_pais es clave foránea que referencia Pais

Categoria es clave foránea que referencia Categoria

COMPETICION_ANUAL: cod_competicion_anual, inicio, fin, cod_competicion, temporada

Cod_competicion es clave foránea que referencia Competicion

Temporada es clave foránea que referencia Temporada

COMPONENTE: cod_componente, nombre, descripcion, utilidad

COMPONENTE_MARCA: cod_componente_marca, cod_componente, cod_fabricante

Cod_componente es clave foránea que referencia Componente

Cod_fabricante es clave foránea que referencia Fabricante

COMPONENTE_MONTADO: cod_componente_montado, cod_componente_marca, cod_coche

Cod_componente_marca es clave foránea que referencia Compnente_marca

Cod_coche es clave foránea que referencia Coche

CONTRATO: cod_piloto, cod_equipo, temporada, piloto_number, descripcion

Cod_piloto es clave foránea que referencia Piloto.

Temporada es clave foránea que referencia Temporada

Piloto_number es clave foránea que referencia Piloto_number. Este campo solo puede tomar los valores PI01 y PI02.

DATOSTELEMETRICOS: cod_datostelemetricos, instante, valor, cod_telemetria

Cod_telemetria es clave foránea que referencia Telemetria

EQUIPO: cod_equipo, nombre, sede, debut, manager

FABRICANTE: NIF, nombre, fecha_inicio

PAIS: cod_pais, denominacion

PATROCINADOR: cod_patrocinador, nombre, sector

PATROCINIO: cod_equipo, cod_patrocinador, temporada

Cod_equipo es clave foránea que referencia Equipo

Cod_patrocinador es clave foránea que referencia Patrocinador

PILOTO: cod_licencia, nombre, nacionalidad

Nacionalidad es clave foránea que referencia Pais

RESULTADO: cod_resultado, posicion, puntuacion, cod_piloto, cod_carrera, cod_coche, vuelta_mas_rapida

Cod_piloto es clave foránea que referencia Piloto

Cod_carrera es clave foránea que referencia Carrera

Cod_coche es clave foránea que referencia Coche

TELEMETRIA: cod_telemetria, ambito, unidad, cod_componente_montado, cod_resultado

Cod_componente_montado es clave foránea que referencia Componente_montado

Cod_resultado es clave foránea que referencia Resultado

TEMPORADA: temporada

2.3.1. Diagrama EER

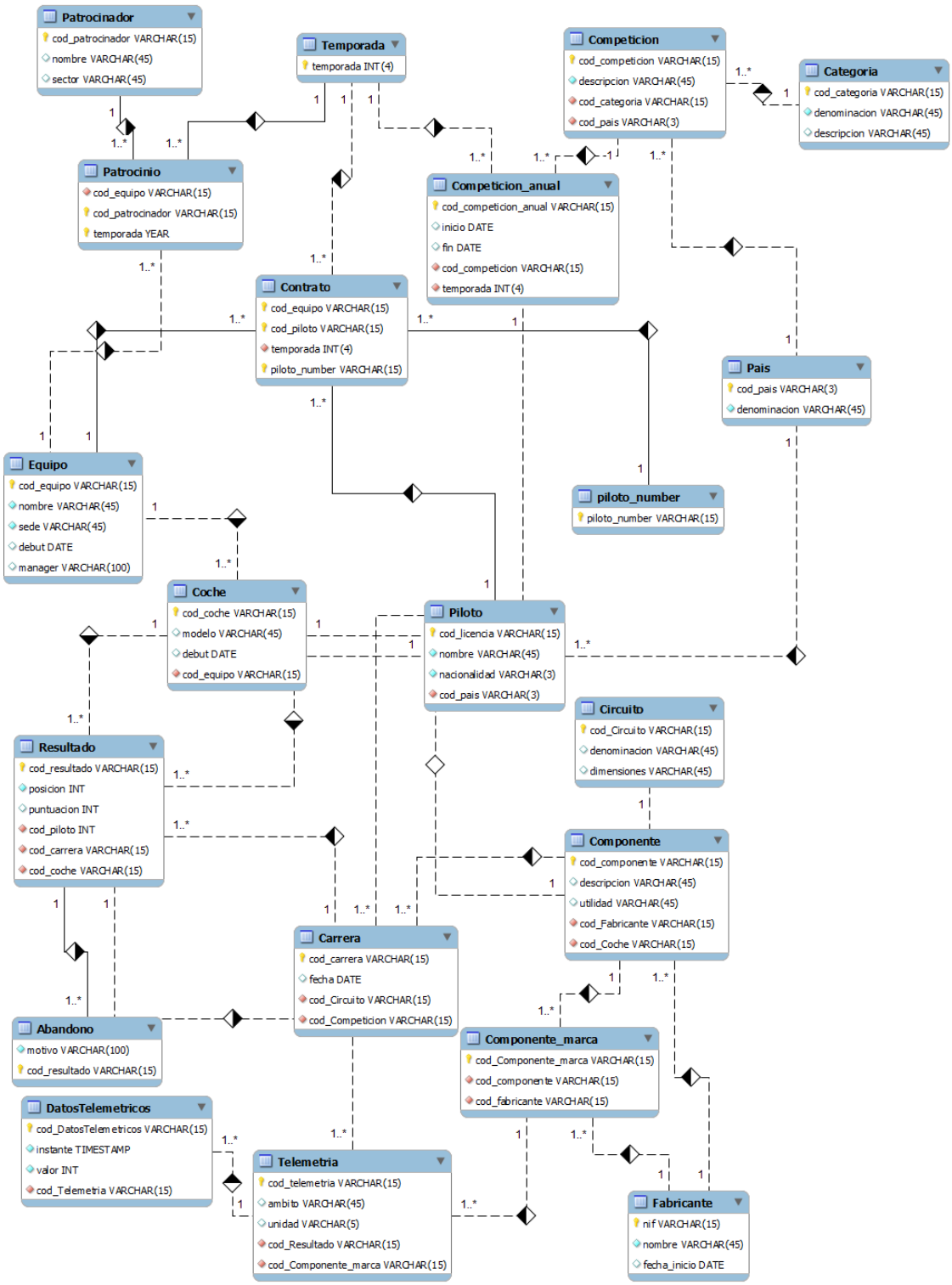
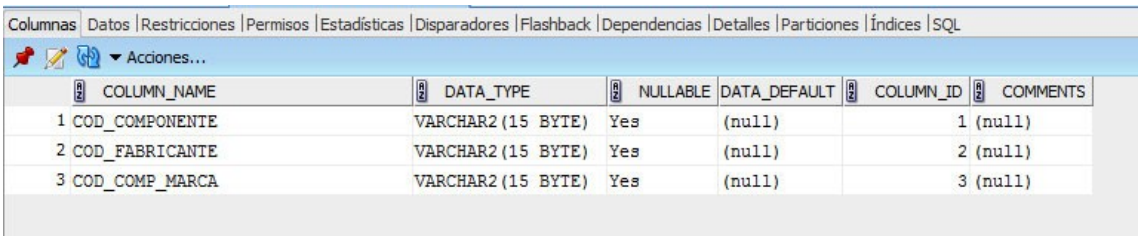


Ilustración 8: Diagrama E-R

2.4. Diseño físico

La implementación del diseño lógico anterior se ha llevado a cabo en Oracle. Y se ha utilizado la herramienta SQL Developper. En 8.1. Anexo 1 se muestra a modo de ejemplo el script de creación de algunas tablas.


Columnas | Datos | Restricciones | Permisos | Estadísticas | Disparadores | Flashback | Dependencias | Detalles | Particiones | Índices | SQL

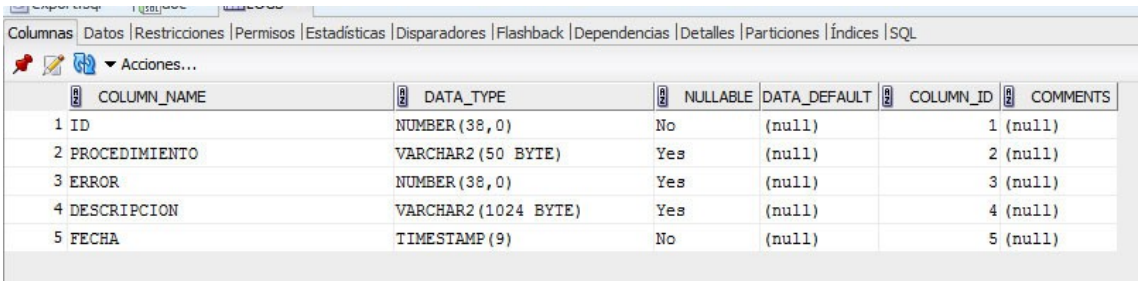
Acciones...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	COD_COMPONENTE	VARCHAR2(15 BYTE)	Yes	(null)	1	(null)
2	COD_FABRICANTE	VARCHAR2(15 BYTE)	Yes	(null)	2	(null)
3	COD_COMP_MARCA	VARCHAR2(15 BYTE)	Yes	(null)	3	(null)

Ilustración 9: Tabla en Oracle SQL Developer

2.5. Tabla de registro (BDR_Logs)

De acuerdo con los requisitos del proyecto, todas las acciones llevadas a cabo en la base de datos (Altas, Bajas y Modificaciones) deben quedar registradas. Para ello, hemos añadido a la base de datos una tabla que recoge el identificativo del registro, el procedimiento que lo ocasiona, el tipo y la descripción del error, y el instante en que se produce.


Columnas | Datos | Restricciones | Permisos | Estadísticas | Disparadores | Flashback | Dependencias | Detalles | Particiones | Índices | SQL

Acciones...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID	NUMBER(38,0)	No	(null)	1	(null)
2	PROCEDIMIENTO	VARCHAR2(50 BYTE)	Yes	(null)	2	(null)
3	ERROR	NUMBER(38,0)	Yes	(null)	3	(null)
4	DESCRIPCION	VARCHAR2(1024 BYTE)	Yes	(null)	4	(null)
5	FECHA	TIMESTAMP(9)	No	(null)	5	(null)

Ilustración 10: Registro de acciones en la BD

Para crear el identificativo de la tabla 'BDR_logs' se han creado un disparador que llama a una secuencia (Anexo 8.3).

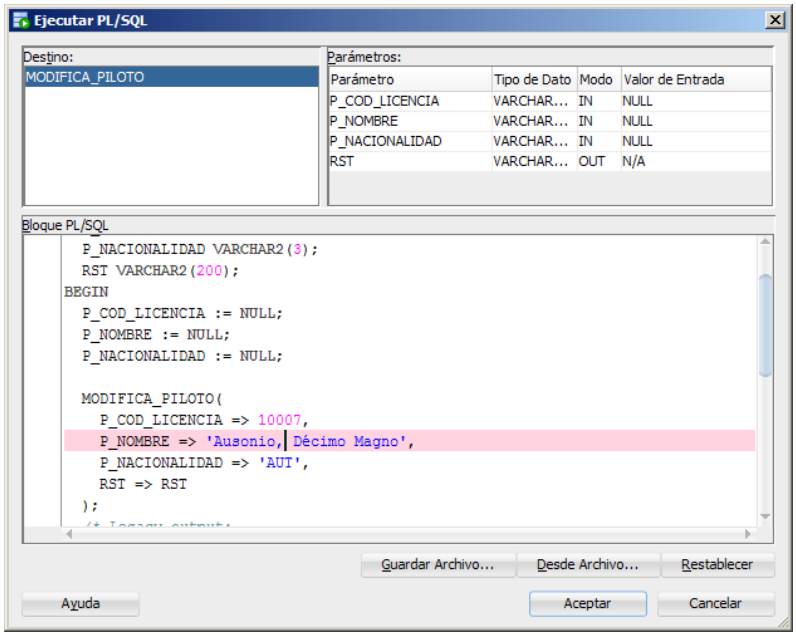
En el apartado siguiente veremos cómo se ejecuta el registro de estos datos desde los distintos procedimientos.

2.6. Procedimientos

2.6.1. Procedimientos de ABM

De acuerdo con los requisitos del enunciado, se han creado los procedimientos de altas, bajas y modificaciones para cada una de las tablas. También, siguiendo los requisitos, para cada una de esas acciones se guarda un registro en la tabla bdr_Logs. En el Anexo 8.2 puede verse, a modo de ejemplo, el código de algunos de estos procedimientos almacenados.

Para la actualización en tiempo real de las estadísticas se ha añadido una llamada a los procedimientos del paquete de actualización de estadísticas en los procedimientos ABA que actualizan datos que afectan a las tablas del repositorio estadístico.




```
Conectando a la base de datos uoc.
Modificación de Piloto con cod_licencia:100007 correcta.
El proceso ha terminado.
Desconectando de la base de datos uoc.
```

Ilustración 11: Ejecución Modifica_piloto y resultado

2.7. Disparadores para el cálculo de datos

Para responder a las necesidades del repositorio estadístico, se ha considerado conveniente añadir atributos a algunas tablas para poder contar con datos precalculados. Esos campos se van a llenar por medio de disparadores. Describo a continuación estos disparadores, de forma que podamos ver las tablas afectadas y los campos concretos que se van a llenar.

T_CARRERAS_GANADAS

Este disparador se desencadena después de insertar, modificar o borrar datos en la tabla BDR_Resultado. Su función es llenar el campo Carreras_ganadas de la tabla Piloto. Este campo se incrementará en uno cada vez que el valor del campo Posición de la Tabla Resultados sea “1”, que indica que la carrera ha sido ganada.

T_CARRERAS_ACABADAS

Funciona de forma semejante al anterior. Se desencadena despúes de insertar, modificar o borrar datos en la tabla BDR_Resultado. Cuando en esta tabla, el valor de Abandono es “0”, se incrementa en “1” el campo Carreras_acabadas de la tabla BDR_Piloto.

T_FABRICANTE_COMP

Este disparador se desencadena después de insertar, modificar o borrar registros en la tabla BDR_Componente_montado. Lo utilizamos para contabilizar el número de componentes que aporta cada fabricante, valor que pasamos al atributo num_componentes de la tabla BDR_componente_marca (Véase Anexo 8.3). Esta tabla está relacionada con el fabricante por medio de una clave ajena.

2.8. Seguimiento de la planificación e incidencias

En el apartado 1.Planificación se preveía la entrega del diseño conceptual, lógico y físico de la base de datos en cumplimiento del segundo hito del TFG, la PAC2, que se entregó el 13/04. A esa entrega pude añadir los procedimientos de ABM de la base de datos. Sin embargo, no pude incluir las tablas del repositorio estadístico ni tampoco los Triggers para el cálculo de datos descritos en el apartado anterior.

2.8. Seguimiento de la planificación e incidencias

Estas incidencias en la planificación se debieron a un error en la interpretación del enunciado, que he podido enmendar gracias a las consultas enviadas al consultor del TFG.

3. Repositorio estadístico

3.1. Introducción

Es requisito del proyecto crear un repositorio estadístico para dar respuesta rápida y actualizada a un conjunto de consultas predefinidas. Estas consultas se deben realizar en un tiempo constante 1, lo que implica una única operación SELECT sobre un registro o una tabla.

Para ello he creado el conjunto de tablas que dan respuesta inmediata a cada una de las consultas que figuran en el enunciado y que constituyen los requisitos del repositorio estadístico.

En el apartado siguiente (3.2) describo esas tablas; en 3.3. muestro los procedimientos y disparadores creados para llenar esas tablas en tiempo; finalmente en 3.4. muestro los procedimientos de consulta.

3.2 Tablas del repositorio estadístico

A continuación muestro las tablas pertenecientes al repositorio estadístico, que son las que responden exactamente a los requisitos.

DHW_TOP10_CARRERASACABADAS	
Atributos	Descripción
- nombre - carreras_acabadas	Listado de los 10 pilotos que han acabado más carreras entre todos los datos registrados.

DHW_TOP10_VUELTAS_MAS_RAPIDAS	
Atributos	Descripción
- piloto - circuito - fecha - tiempo - equipo	Dado un circuito determinado, las 10 vueltas más rápidas que se han dado en él entre todos los datos registrados. Se indica tiempo, fecha, piloto y equipo.

DHW_TOP5_PATROCINIO	
Atributos	Descripción
- nombre - temporada	Dado un año concreto, Top 5 de los patrocinadores que aportan más

- aportación	dineros a los equipos en cualquier categoría.

DHW_CARBURANTE_COCHE_AÑO	
Atributos	Descripción
- código_coche - modelo - debut - año - consumo_total	Dada una competición y un año determinado, coche que lleva consumido más carburante. Se dan los datos del coche y el total de litros consumidos.

DHW_COMP_DEFECTUOSOS	
Atributos	Descripción
- código_coche - num_componentes - comp_defectuosos - avg_defectuosos	Porcentaje de componentes defectuosos de cada coche de competición. Se considera que un componente es defectuoso cuando recibe, en cualquier carrera, un dato telemétrico erróneo.
El campo avg_defectuosos es un campo virtual, es decir, su valor se deriva de una operación entre los campos num_componentes y com_defectuosos (Véase Virtual Columns in Oracle [8])	

DHW_VOLUM_DATOS_TELE	
Atributos	Descripción
- equipo - año - volumen_datos	Volumen de datos telemétricos registrados. Habrá que indicar el equipo que, dado un año concreto, ha recogido más datos, e indicar el número.

DHW_TEMPERATURA_MAS_ALTA	
Atributos	Descripción
- coche - piloto - fecha	Temperatura más alta registrada en nuestro sistema en cualquier carrera y en cualquier coche. Se indicará fecha,

- carrera - valor	carrera, coche, piloto y valor.

DHW_TOP5_FABRICANTES	
Atributos	Descripción
- nif - nombre - num_componentes	Lista de los 5 fabricantes que aportan más componentes a la competición.

DHW_CLASIFICACION	
Atributos	Descripción
- piloto - comp_anual - puntuación	Resultado de cada piloto en cada competición disputada

DHW_CLASIFICACION_EN_CURSO	
Atributos	Descripción
- comp_anual - piloto - puntuación	Clasificación de cada competición en curso: suma de puntos de cada piloto participante

DHW_MEJOR_EVOLUCION	
Atributos	Descripción
- piloto - puntuación acumulada	Datos del piloto con mejor evolución (puntos acumulados) durante un periodo establecido.

Para el repositorio estadístico se ha creado también la tabla DWH_log, que registra todas las operaciones de actualización y consulta que afectan a sus tablas.

DHW_LOG	
Atributos	Descripción

- ID - procedimiento - error - descripción - fecha	Registra los datos de las operaciones realizadas en el repositorio estadístico.

3.3. Disparadores y Procedimientos

Para llenar las tablas del repositorio estadístico y mantenerlas actualizadas en tiempo real se han utilizado disparadores y procedimientos. A continuación se expone la función concreta de cada uno de estos disparadores y procedimientos en relación a la tablas y atributos de cuya actualización son responsables.

3.3.1. Triggers

T_COMP_MONTADOS

- Inserta el código del coche, el modelo y el número de componentes en la tabla DWH_COMP_DEFECTUOSOS.
- Se desencadena después de insertar, modificar o borrar en la tabla BDR_Telemetria

T_DEFECTUOSOS

- Inserta el código del coche, el modelo y el número de componentes defectuosos en la tabla dwh_COMP_DEFECTUOSOS.
- Se desecadena después de insertar, modificar o borrar en la tabla BDR_Datostelemetricos
- En DWH_Comp_defectuosos se ha creado el campo virtual AVG_defectuosos que se llena realizando la operación de porcentaje de los datos de los campos de número de componentes montados y número de componentes defectuosos.
- Véase Anexo 8.7.

T_CLASIFICACION

- Inserta el código del piloto, el nombre de la competición y la puntuación en la tabla DWH_Clasificación.
- Se desecadena después de insertar, modificar o borrar en la tabla

BDR_Resultado

- Es una tabla que utilizo para tener precalculada la puntuación. De esta tabla obtendré datos para las tablas del repositorio estadístico DWH_Resultado_piloto_comp. y DWH_Clasificacion_en_curso

3.3.2. Procedimientos

Los procedimientos almacenados han sido incluidos en el paquete PKG_estadistica y PKG_estadistica_body. El primero recoge la especificación de los procedimientos, es decir, el nombre y los parámetros. El segundo añade el cuerpo, es decir, las instrucciones y sentencias SQL que se ejecutarán. Para su realización se han seguido las instrucciones expuestas en "PL/SQL Packages" en *Oracle Database Online Documentation* [2]. Sobre los beneficios de esta técnica de encapsulación puede consultarse Burleson Consulting [1].

Todos estos procedimientos son llamados desde los procedimientos ABA para conseguir la actualización en tiempo real de las tablas del repositorio estadístico.

PKG_DW_ESTADISTICAS.TOP10_CARRERASACABADAS();

- Inserta el nombre del piloto y el número de carreras acabadas en dwh_TOP10_CARRERASACABADAS.
- Utiliza la función RANK() y limita los resultados al top 10 (Véase el apartado dedicado a "RANK" en [2]).
- En caso de empate entre varios pilotos en número de carreras ganadas, obtendríamos tantos registros como pilotos comparten los diez mejores resultados.

PKG_DW_ESTADISTICAS.TOP10_CARRERASGANADAS();

- Inserta el nombre del piloto y el número de carreras ganadas en dwh_TOP10_CARRERASGANADAS.
- Utiliza la función RANK() y limita los resultados al top 10.
- En caso de empate entre varios pilotos en número de carreras ganadas, obtendríamos tantos registros como pilotos comparten los diez mejores resultados.

PKG_DW_ESTADISTICAS.TOP10_VUELTASMASRAPIDAS();

- Inserta en la tabla DWH_top10_vueltasmasrapidas los datos de

cod_resultado, piloto, circuito, fecha, tiempo y equipo.
- Utiliza la función RANK realizando una partición por circuito y ordenando por el tiempo invertido en la vuelta.
- Véase anexo 8.5. Detalle del código del procedimiento.

PKG_DW_ESTADISTICAS.TOP5_PATROCINIO();
- Inserta en la tabla DWH_top5_patrocinio el nombre del patrocinador, la temporada y la cuantía de la aportación del top 5 en aportaciones.
- Utiliza la función RANK realizando una partición por temporada y ordenando por la cuantía de la aportación.

PKG_DW_ESTADISTICAS.TOP5_FABRICANTES();
- Inserta en la tabla DWH_top5_fabricantes el nif y nombre del fabricantes y el número de componentes montados.
- Utiliza la función RANK ordenando por número de componentes.

PKG_DW_ESTADISTICAS.CARBURANTE_COCHE_ANO();
- Inserta en la tabla DWH_carburante_coche_ano todos los datos del coche, el año, el nombre de la competición y el consumo total de carburante.
- Utiliza la función RANK realizando una partición por competición y año y ordenando por consumo.
- Véase Anexo 8.6. Detalle del código del procedimiento

PKG_DW_ESTADISTICAS.PILOTO_COMP_ANO();
- Inserta en la tabla DWH_resultado_piloto_comp los datos de piloto, temporada, competición y puntuación.
- La puntuación ha sido previamente precalculada en la tabla dwh_clasificacion, gracias al trigger T_Clasificacion.

PKG_DW_ESTADISTICAS.CLASIFICACION_COMP_CURSO();
- Inserta en la tabla DWH_clasificacion_en_curso los datos de piloto, temporada, competición y puntuación. Ordena por el nombre de la competición y por la puntuación.
- La puntuación ha sido previamente precalculada en la tabla

dwh_clasificacion, gracias al trigger T_Clasificacion.

PKG_DW_ESTADISTICAS.TEMPERATURA_MAS_ALTA();

- Inserta en la tabla DWH_temperatura_mas_alta los datos de coche, piloto, fecha, carrera y valor.

PKG_DW_ESTADISTICAS.VOLUM_DATOS_TELEMETRICOS();

- Inserta en la tabla DWH_volum_telemetricos los datos de equipo, año y volumen de datos.
- Utiliza la función RANK y se realiza una partición por año.
- Véase Anexo 8.8.

PKG_DW_ESTADISTICAS.PILOTO_MEJOR_EVOLUCION();

- Inserta en la tabla DWH_mejor_evolucion los datos del piloto y el número de puntos acumulado entre dos fechas establecidas.
- Utiliza la función la pseudocolumna Rownum y la técnica Top-N Reporting (Véase “Top-N Queries” [5]).
- Véase Anexo 8.8.

3.4. Procedimientos almacenados de consulta

Los requisitos del proyecto imponen que todas las operaciones con la base de datos, incluidas las consultas al repositorio estadístico, se realicen mediante procedimientos almacenados. Además, al igual que se ha hecho para los procedimientos ABA, es necesario que quede constancia de todas las operaciones en la tabla de dwh_logs.

Los procedimientos almacenados de consulta han sido incluidos en el paquete PKG_consutas y PKG_consultas_body. El primero recoge la especificación de los procedimientos, es decir, el nombre y los parámetros. El segundo añade el cuerpo, es decir, las instrucciones PL y sentencias SQL que se ejecutarán.

Cada una de los procedimientos de consulta incluye dos parámetros de salida. El primero es un cursor para los resultados, y el segundo es la cadena RSP que se registrará en la tabla dwh_logs. En Anexo 8.9 se muestra detalle del código de uno de estos procedimientos de consulta.

Se ofrece a continuación la descripción de estas consultas, indicando, en su caso, los parámetros de entrada.

PKG_CONSULTAS.C_TOP10_CARRERASACABADAS();

- Devuelve el nombre del piloto y el número de carreras acabadas ordenadas de modo descendente por número de carreras acabadas.
- En caso de empate entre varios pilotos en número de carreras ganadas, obtendríamos tantos registros como pilotos comparten los diez mejores resultados.

PKG_CONSULTAS.C_TOP10_CARRERASGANADAS();

- Devuelve el nombre del piloto y el número de carreras ganadas ordenadas en modo descendente por número de carreras ganadas.
- En caso de empate entre varios pilotos en número de carreras ganadas, obtendríamos tantos registros como pilotos comparten los diez mejores resultados.

PKG_DW_CONSULTAS.C_COMP_DEFECTUOSOS();

- Devuelve los datos del coche y el porcentaje de componentes defectuosos ordenado de modo descendente por componentes defectuosos.

PKG_CONSULTAS.C_TOP10_VUELTASMASRAPIDAS();

- Parámetro de entrada: Código del circuito.
- Devuelve los datos de cod_resultado, piloto, circuito, fecha, tiempo y equipo, para cada circuito. Está ordenada de forma descendente por tiempo de cada vuelta.

PKG_CONSULTAS.C_TOP5_PATROCINIO();

- Parámetro de entrada: Año
- Devuelve el nombre del patrocinador, la temporada y la cuantía de la aportación del top 5 en aportaciones, para cada año. Está ordenada de modo descendente por la cuantía de la aportación.

PKG_CONSULTAS.C_TOP5_FABRICANTES();

- Devuelve el nif y nombre del fabricantes y el número de componentes montados. Está ordenada de modo descendente por el número de componentes.

PKG_CONSULTAS.C_CARBURANTE_COCHE_ANO();

- Parámetros de entrada: Código de competición y año.
- Devuelve todos los datos del coche, el año, el nombre de la competición y el consumo total de carburante, para una competición y un año dados . Está ordenada de modo descendente por carburante consumido.

PKG_CONSULTAS.C_PILOTO_COMP_ANO();

- Devuelve los datos de piloto, temporada, competición y puntuación. Está ordenada por Piloto y competición.

PKG_CONSULTAS.C_CLASIFICACION_COMP_CURSO();

- Devuelve los datos de piloto, temporada, competición y puntuación. Está ordenada por el nombre de la competición y por la puntuación.

PKG_CONSULTAS.C_TEMPERATURA_MAS_ALTA();

- Devuelve los datos de coche, piloto, fecha, carrera y valor.

PKG_CONSULTAS.C_CLASIFICACION_COMP_CURSO();

- Parámetro de entrada: Año
- Devuelve los datos del equipo, año y volumen de datos telemétricos recogidos.

PKG_CONSULTAS.C_MEJOR_EVOLUCION();

- Devuelve los datos del piloto que ha tenido una mejor evolución entre dos momentos determinados, indicando los puntos ganados entre los dos momentos.

3.5. Seguimiento de la planificación e incidencias

Esta parte del proyecto me resultó especialmente laboriosa por errores de planteamiento, mencionados en el punto 2.8.

Mi primera interpretación del enunciado me condujo a realizar un almacén de datos de tipo histórico, una implementación de tipo ROLAP [7]. Para su realización diseñé una estructura lógica de estrella, con tablas de dimensiones y hechos. Su finalidad era el Proceso analítico de datos (OLAP). Estos datos, que constituirían el repositorio estadístico era extraídos periódicamente de la base de datos operacional, es decir, del sistema de transacciones (OLTP). Para ello creé un procedimiento de Extracción, Transformación y carga de datos (ETLDW).

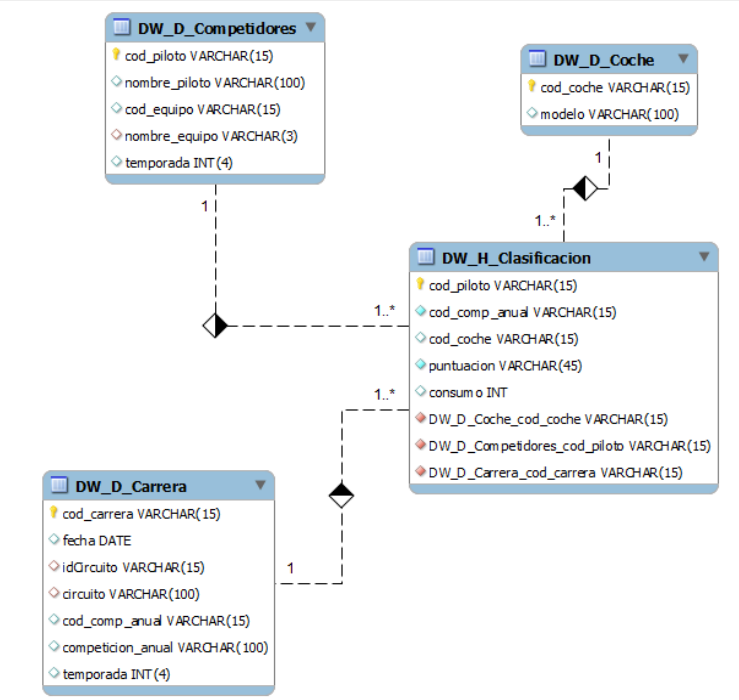


Ilustración 12: Hechos y dimensiones

Finalmente, la evaluación de la PAC3 me permitió darme cuenta del error y abordar un nuevo planteamiento. Dado que la disponibilidad de tiempo era cada ver menor opté por comenzar creando las tablas imprescindibles para las consultas del repositorio estadístico, tal como he expuesto en los epígrafes

3.5. Seguimiento de la planificación e incidencias

anteriores.

4. Pruebas de la base de datos

4.1. Inserción de valores

Se han insertado valores en las distintas tablas de la base de datos. Los valores se han creado en Excel, y se han importado a la base de datos a través de la funcionalidad 'Importar datos' de SQL Developer.

El volumen de datos con que contamos actualmente es el siguiente:

Tabla	Número de filas
Abandono	6
Carrera	13
Categoría	4
Circuito	28
Coche	26
Competición	4
Competición_anual	4
Componente	8
Componente_marca	13
Componente_montado	200
Contrato	91
Datostelemétricos	702
Equipo	13
Fabricante	12
Pais	237
Patrocinador	15
Patrocinio	166
Piloto	98
Resultado	101
Telemetría	191
Temporada	7

4.2. Juego de pruebas

Se ha creado un procedimiento “PRUEBAS_ABA” que llama a los procedimientos ABA. Los resultados de su ejecución se podrán seguir a través de los mensajes de salida. Además, en las tablas de logs, tanto en la de la base de datos operacional (bdr_logs), como en la del repositorio estadísticos (DWH_logs), quedará registro de todas las operaciones.

Se han tenido en cuenta especialmente las restricciones de clave foránea y, en su caso, el funcionamiento de los disparadores asociados a estos procedimientos. También se ha revisado el comportamiento de condiciones señaladas en algunos procedimientos. Por ejemplo en Alta_contrato, se puede verificar que se cumple la restricción impuesta en el enunciado de que un equipo solo pueda contar con dos pilotos por año.

ALTA_CONTRATO

```
Error: -20000 ORA-20000: Máximo número de pilotos para esta temporada.
ORA-06512: at "FIA_BDR.T_PILOTO_MAX_2", line 6
```

Una restricción semejante se ha impuesto en el procedimiento ALTA_COMPETICION_ANUAL, para comprobar que el año de finalización es superior al de inicio de la competición.

```
ALTA_COMPETICION_ANUAL ('CA99999', '10/01/2000', '10/02/1999',  'CO1001',
'2014', RST);
ALTA_COMPETICION_ANUAL
Error: -2290 ORA-02290: check constraint (FIA_BDR.BDR_COMPETICION_ANUAL_CHK1)
violated
```

Se ha comprobado también la actualización en tiempo real de las tablas estadísticas. En la siguiente tabla, se muestra un listado de verificaciones efectuadas.

Procedimiento	Resultados esperados	Test
Alta_piloto	Actualización estadísticas:	Ok
	PKG_DW_ESTADISTICAS.TOP10A_CARRERASACABADAS();	
	PKG_DW_ESTADISTICAS.TOP10_CARRERASGANADAS();	
	PKG_DW_ESTADISTICAS.PILOTO_COMP_ANO();	
	PKG_DW_ESTADISTICAS.CLASIFICACION_COMP_CURSO();	
	PKG_DW_ESTADISTICAS.TEMPERATURA_MAS_ALTA();	
	PKG_DW_ESTADISTICAS.PILOTO_MEJOR_EVOLUCION();	
	Comprobación restricciones clave primaria	ok

	Comprobación restricciones clave ajena	ok
	INSERT EN BDR_LOGS	ok
	INSERT EN DWH_LOGS	ok
Modifica_piloto	Actualización estadísticas:	Ok
	PKG_DW_ESTADISTICAS.TOP10A_CARRERASACABADAS();	
	PKG_DW_ESTADISTICAS.TOP10_CARRERASGANADAS();	
	PKG_DW_ESTADISTICAS.PILOTO_COMP_ANO();	
	PKG_DW_ESTADISTICAS.CLASIFICACION_COMP_CURSO();	
	PKG_DW_ESTADISTICAS.TEMPERATURA_MAS_ALTA();	
	PKG_DW_ESTADISTICAS.PILOTO_MEJOR_EVOLUCION();	
	Comprobación si existe cod_licencia	ok
	Comprobación restricciones clave ajena	ok
	INSERT EN BDR_LOGS	ok
	INSERT EN DWH_LOGS	ok
Alta_contrato	Actualización estadísticas:	Ok
	PKG_DW_ESTADISTICAS.TOP10A_CARRERASACABADAS();	
	PKG_DW_ESTADISTICAS.TOP10_CARRERASGANADAS();	
	PKG_DW_ESTADISTICAS.PILOTO_COMP_ANO();	
	PKG_DW_ESTADISTICAS.CLASIFICACION_COMP_CURSO();	
	PKG_DW_ESTADISTICAS.TEMPERATURA_MAS_ALTA();	
	PKG_DW_ESTADISTICAS.PILOTO_MEJOR_EVOLUCION();	
	Comprobación restricciones clave primaria	ok
	Comprobación de que no hay más de dos pilotos por equipo	ok
	INSERT EN BDR_LOGS	ok
	INSERT EN DWH_LOGS	ok
Modifica_contrato	Actualización estadísticas:	Ok
	PKG_DW_ESTADISTICAS.TOP10A_CARRERASACABADAS();	
	PKG_DW_ESTADISTICAS.TOP10_CARRERASGANADAS();	
	PKG_DW_ESTADISTICAS.PILOTO_COMP_ANO();	
	PKG_DW_ESTADISTICAS.CLASIFICACION_COMP_CURSO();	
	PKG_DW_ESTADISTICAS.TEMPERATURA_MAS_ALTA();	
	PKG_DW_ESTADISTICAS.PILOTO_MEJOR_EVOLUCION();	
	Comprobación restricciones clave ajena	ok
	INSERT EN BDR_LOGS	ok
	INSERT EN DWH_LOGS	ok

Alta_circuito	Actualización estadísticas:	
	PKG_DW_ESTADISTICAS.TOP10_VUELTASMASRAPIDAS();	Ok
	Comprobación restricciones clave ajena	ok
	INSERT EN BDR_LOGS	ok
	INSERT EN DWH_LOGS	ok
Alta_telemetria	Actualización estadísticas:	Ok
	PKG_DW_ESTADISTICAS.TEMPERATURA_MAS_ALTA();	
	PKG_DW_ESTADISTICAS.VOLUM_DATOSTELEMETRICOS_ANO();	ok
	Comprobación restricciones clave ajena	ok
	TRIGGER "T_COMP_MONTADOS" (Incrementa en 1 el número de componentes montados en la columna num_componentes de la tabla dwh_comp_defectuosos).	ok
	INSERT EN BDR_LOGS	ok
	INSERT EN DWH_LOGS	
Alta_resultado	Actualización estadísticas:	ok
	PKG_DW_ESTADISTICAS.TOP10_CARRERASACABADAS();	
	PKG_DW_ESTADISTICAS.TOP10_CARRERASGANADAS();	
	PKG_DW_ESTADISTICAS.PILOTO_COMP_ANO();	
	PKG_DW_ESTADISTICAS.TEMPERATURA_MAS_ALTA();	
	PKG_DW_ESTADISTICAS.VOLUM_DATOSTELEMETRICOS_ANO();	
	PKG_DW_ESTADISTICAS.PILOTO_MEJOR_EVOLUCION();	ok
	INSERT EN BDR_LOGS	ok
	INSERT EN DWH_LOGS	ok
	Comprobación restricciones clave ajena	ok
	TRIGGER "T_CARRERAS_ACABADAS"	ok
	TRIGGER "T_CARRERAS_GANADAS"	ok
	TRIGGER "T_CLASIFICACION"	ok
	INSERT EN BDR_LOGS	ok
	INSERT EN DWH_LOGS	
Alta_datostelemetricos	Actualización estadísticas	Ok
	PKG_DW_ESTADISTICAS.TEMPERATURA_MAS_ALTA();	
	PKG_DW_ESTADISTICAS.VOLUM_DATOSTELEMETRICOS_ANO();	ok

	TRIGGER "T_DEFECTUOSOS"	ok
	NSERT EN BDR_LOGS	ok
	INSERT EN DWH_LOGS	ok
	Comprobación restricciones clave ajena	
Alta_componente_marca	Actualización de estadísticas	Ok
	PKG_DW_ESTADISTICAS.TOP5_FABRICANTES();	
	NSERT EN BDR_LOGS	ok
	INSERT EN DWH_LOGS	ok
	Comprobación restricciones clave ajena	ok
Alta_fabricante	Actualización de estadísticas	Ok
	PKG_DW_ESTADISTICAS.TOP5_FABRICANTES();	
	NSERT EN BDR_LOGS	ok
	INSERT EN DWH_LOGS	ok
	Comprobación restricciones clave ajena	ok

Propuesta de mejora

Se ha creado una única base de datos para facilitar el desarrollo del proyecto y la instalación. Sin embargo en un entorno real las bases de datos operacional y el repositorio estadístico deben ser bases de datos separadas.

Dependiendo del volumen de datos y transacciones sería deseable el alojamiento en máquinas distintas, y en caso de tener tablas demasiado extensas se podría optar por un sistema distribuido.

Se ha creado un solo usuario, si bien en un entorno real sería necesario definir una política de permisos. Al menos, el sistema debería contar con un usuario del sistema, con todos los privilegios, usuarios que llevaran a cabo las actualizaciones de la base de datos, es decir, con acceso a la base de datos operacional, y usuarios con acceso al repositorio estadístico.

5. Valoración económica

De las tareas expuesta en 1.4.1 se extrae la siguiente tabla, en la que se contabilizan las horas de las distintas tareas y se distribuyen por el tipo de profesional que debería llevarlas a cabo en un entorno real: director de proyecto, analista, programador y documentalista.

Así, el director de proyecto se encargará de elaborar el plan de trabajo o asignar tareas; el analista deberá analizar los requisitos y crear el diseño de la base de datos; el programador se encargará de la instalación del software, la programación de procedimientos, disparadores, y la ejecución de pruebas. Finalmente, un documentalista deberá encargarse de la preparación de gráficos, redacción de informes, memoria final, etc...

Profesional	Horas	Precio/hora	Total
Director	30	60,00 €	1.800,00 €
Analista	60	50,00 €	3.000,00 €
Programador	140	35,00 €	4.900,00 €
Documentalista	100	25,00 €	2.500,00 €
Total			12.200,00 €
IVA 21%			2.566,00 €
Total Factura			14.766,00 €

6. Conclusiones

El diseño conceptual y su traducción final a un diseño físico es una tarea muy satisfactoria cuando se consiguen los objetivos aunque sea modestamente. Ver en funcionamiento un conjunto de entidades interrelacionadas capaces de reproducir las interacciones que se producen entre objetos de una parcela compleja de la realidad es indudablemente una labor de ingeniería.

La competencia en la creación de este tipo de sistemas se beneficia en buena medida de la experiencia a pesar de su aparente sencillez teórica. La posibilidad de crear un modelo en concreto sobre una realidad que me es completamente ajena como son las competiciones automovilísticas ha sido un reto interesante.

Como segunda conclusión, gracias al TFG me he acercado por primera vez al concepto y las técnicas del almacén de datos, el Data Ware House. A lo largo del grado no he visto nada sobre Dataware. Afrontarlo ahora al final, con la presión que implica un TFG, tal vez no sea el mejor momento de hacerlo; de hecho los principales problemas e incidencias, que he señalado en la memoria, se derivan de mi ignorancia sobre las técnicas del Data Ware. He partido de un planteamiento inicial poco acertado. Sin embargo creo que he podido reconducirlo, y esto me da cierta confianza para la ejecución de futuros proyectos.

Como tercera conclusión he podido ver en funcionamiento el potencial de PL/SQL y, en general de las tecnologías en torno a las bases de datos relacionales, y creo que podré aplicarlas con mayor eficiencia en mi práctica profesional diaria.

Pero un TFG no es solo programación PL/SQL, sino también metodología, gestión del tiempo, plan de trabajo, búsqueda de fuentes, etc... Afrontar un calendario e ir cumpliendo sus hitos es una disciplina imprescindible para garantizar el éxito de cualquier proyecto. Creo que a lo largo de este TFG he ganado algo en el aprendizaje de esa difícil tarea de gestionar el tiempo.

El momento de las conclusiones suele ser idóneo para plantear nuevos retos. Creo que, si me adentro profesionalmente en el campo de las bases de datos, deberé ganar destreza en programación PL/SQL y, si continúo con ORACLE, deberé conocer sus productos, y en concreto los orientados a Data Ware House. También trataré de conocer otros productos de libre distribución, al

margen de MySQL, con el que habitualmente trabajo, como, por ejemplo, PostgreSQL.

6. Glosario

Base de datos operacional: Base de datos que gestiona las transacciones comunes de actualización, borrado e inserción de datos

CASE. Ingeniería de software asistida por computadora. Permiten aumentar la calidad y, sobre todo, incrementar la productividad.

Data Ware House. Base de datos que almacena información con finalidad generalmente estadística. Es también el conjunto de tecnologías que permiten su construcción. Se utiliza dentro de la organización en la toma de decisiones.

Diseño conceptual. Primera fase en el desarrollo de la base de datos. Traduce el universo de discurso o historia a un esquema de entidades, atributos e interrelaciones.

Diseño físico. Implementación del diseño lógico en un Sistema de Gestión de Bases de Datos específico.

Diseño lógico. Segunda fase en el desarrollo de la base de datos. Traduce el diseño conceptual a un modelo lógico estándar: Las entidades se convierten en relaciones, y las interrelaciones, dependiendo de su cardinalidad, en relaciones o propagación de claves.

Disparador (del inglés Trigger). Disparador: Conjunto de instrucciones de la base de datos que tiene esta estructura: llamada de actuación, restricción y acción a ejecutar.

ETL (Extract, Transform and Load): Proceso que permite extraer información de la base de datos operacional, transformarla y cargarla en el la base de datos multidimensional del Data Warehouse.

OLAP (On-Line Analytical Processing): Conjunto de tecnologías y procedimientos para el proceso analítico de los datos.

OLTP (On-Line Transactional Proccessing) Procesamiento destinado a la gestión de la base de datos operacional, facilitando las tareas de actualización de registros.

Paquete (ORACLE). Sistema utilizado para guardar subprogramas y otros objetos de la base de datos. Normalmente consta de una especificación y del cuerpo. La especificación contiene la signatura o declaraciones públicas del programa, y el cuerpo desarrolla las instrucciones del programa.

PL/SQL: Lenguaje de programación procedural para el Sistema de Gestión de Bases de Datos de ORACLE. Está almacenado y compilado en la base de datos y corre dentro de ORACLE

Procedimento (del inglés *procedure*) : Conjunto de instrucciones que interactúan con la base de datos. Pueden incluir parámetros de entrada y salida.

SGBDR (del inglés RDBMS). Sistema de Gestión de Bases de datos.

SQL. Lenguaje estándar de consulta de bases de datos.

UML. Lenguaje unificado de modelación. Lenguaje estándar de modelado de sistemas de software. Es el más utilizado en la actualidad.

7. Bibliografía

[1] Burleson Consulting (2008), How to create an Oracle PL/SQL package body, May 17, [En línea]. http://www.dba-oracle.com/t_create_package_body.htm [Consultado el 9/06/2015]

[2] Oracle Database Online Documentation 10g Release 2 (10.2). [En línea] . http://docs.oracle.com/cd/B19306_01/index.htm [Consultado 11/06/2015]

[3] Feuerstein, Steven (2002), Oracle PL/SQL Programming, O'Reilly.

[4] Silberschatz, A.; Korth, H.F.; Sudarshan, S. (2006). *Fundamentos de bases de datos* (5a. ed.). Madrid: McGraw-Hill.

[5] Top-N Queries. [En línea]. https://oracle-base.com/articles/misc/top-n-queries [Consultado 10/06/2015]

[6] Urman, S. (2002). *Oracle9i: Programación PL/SQL*. Oracle Press Osborne, McGraw Hill.

[7] Utley, Craig, “Designing the Star Schema Database. Version 1.1”, CIOBriefings, [en línea], http://www.ciobriefings.com/Publications/WhitePapers [Consutado: 15/04/2014]

[8] Virtual Columns in Oracle Database 11g Release 1. [En línea] https://oracle-base.com/articles/11g/virtual-columns-11gr1.php [Consultado 2/05/2015]

[9] VV.AA. (2004), *Bases de dades II,* Barcelona, Fundación per a la UOC, 2ª edición.

8. Anexos

8.1. Tablas de la Base de Datos Operacional

```
CREATE TABLE UOC.Competicion_anual (
  cod_competicion_anual VARCHAR(15) NOT NULL,
  descripcion VARCHAR(45) NOT NULL,
  inicio DATE NULL,
  fin DATE NULL,
  cod_competicion VARCHAR(15) NOT NULL,
  temporada INT(4) NOT NULL,
  PRIMARY KEY (cod_competicion_anual),
  CONSTRAINT fk_Competicion_anual_Competicion1
    FOREIGN KEY (cod_competicion)
    REFERENCES UOC.Competicion (cod_competicion),
  CONSTRAINT fk_Competicion_anual_Temporada1
    FOREIGN KEY (temporada)
    REFERENCES UOC.Temporada (temporada));

CREATE TABLE UOC.Categoria (
  cod_categoria VARCHAR(15) NOT NULL,
  denominacion VARCHAR(45) NOT NULL,
  descripcion VARCHAR(45) NULL,
  PRIMARY KEY (cod_categoria));

CREATE TABLE UOC.Equipo (
  cod_equipo VARCHAR(15) NOT NULL,
  nombre VARCHAR(45) NOT NULL,
  sede VARCHAR(45) NOT NULL,
  debut DATE NULL,
  manager VARCHAR(100),
  PRIMARY KEY (cod_equipo));
```

8.2. Procedimientos de ABA. Ejemplos

ALTA_PILOTO:

```
create or replace PROCEDURE "ALTA_PILOTO" (
  P_cod_licencia In piloto.cod_licencia%Type,
  P_nombre In piloto.nombre%Type,
  P_nacionalidad In piloto.nacionalidad%Type,
  RST Out Nocopy Varchar)
Is
P_error Integer;

Begin
RST:='Dado de alta piloto con cod_licencia:' || P_cod_licencia || '
correcta.';
Insert Into piloto(cod_licencia, nombre, nacionalidad) Values (P_cod_licencia,
P_nombre, P_nacionalidad);
P_error:= SQLCODE;
commit;
-- Actualizamos estadísticas
PKG_DW_ESTADISTICAS.TOP10_CARRERASACABADAS();
PKG_DW_ESTADISTICAS.TOP10_CARRERASGANADAS();
PKG_DW_ESTADISTICAS.PILOTO_COMP_ANO();
KG_DW_ESTADISTICAS.CLASIFICACION_COMP_CURSO();
PKG_DW_ESTADISTICAS.TEMPERATURA_MAS_ALTA();
```

```
PKG_DW_ESTADISTICAS.PILOTO_MEJOR_EVOLUCION();
Insert into logs(procedimiento,error,descripcion,fecha) Values ($$PLSQL_UNIT,
0, RST, SYSTIMESTAMP);
commit;
Dbms_output.Put_line(RST);

Exception

  When DUP_VAL_ON_INDEX Then
    P_error:= SQLCODE;
    RST:='Error: El piloto ya existe.';
    Dbms_output.Put_line(RST);
    rollback;
    Insert into logs(procedimiento,error,descripcion,fecha) Values ($
$PLSQL_UNIT, P_error, RST, SYSTIMESTAMP);
    commit;

   When Others Then
     P_error:= SQLCODE;
     RST:= 'Error: ' || P_error ||' ' || SQLERRM;
     Dbms_output.Put_line(RST);
     rollback;
     Insert into logs(procedimiento,error,descripcion,fecha) Values ($
$PLSQL_UNIT, P_error, RST, SYSTIMESTAMP);
     commit;
End;
```

BAJA_PILOTO:

```
create or replace PROCEDURE "BAJA_PILOTO" (
  P_cod_licencia In piloto.cod_licencia%Type,
  RST Out Nocopy Varchar)
Is
P_error Integer;
P_error_codigo exception;

Begin
RST:='Eliminación del piloto con cod_licencia:' || P_cod_licencia || '
correcta.';
Delete From piloto Where cod_licencia= P_cod_licencia;
P_error := SQLCODE;
IF (SQL%ROWCOUNT=0) Then
    Raise P_error_codigo;
END IF;
commit;
Insert into logs(procedimiento,error,descripcion,fecha) Values ($$PLSQL_UNIT,
0, RST, SYSTIMESTAMP);
commit;
Dbms_output.Put_line(RST);

Exception

 When NO_DATA_FOUND Then
    P_error := SQLCODE;
    RST:='Error: No existe el piloto';
    Dbms_output.Put_line(RST);
    Rollback;
    Insert into logs(procedimiento,error,descripcion,fecha) Values ($
$PLSQL_UNIT, P_error, RST, SYSTIMESTAMP);
    commit;
```

```
  When P_error_codigo Then
    P_error := SQLCODE;
    RST:='Error: Piloto a eliminar no se encuentra.';
    Dbms_output.Put_line(RST);
    Rollback;
    Insert into logs(procedimiento,error,descripcion,fecha) Values ($
$PLSQL_UNIT, P_error, RST, SYSTIMESTAMP);
    commit;

  When Others Then
     P_error:= SQLCODE;
     RST:= 'Error: ' || P_error ||' ' || SQLERRM;
     Dbms_output.Put_line(RST);
     rollback;
     Insert into logs(procedimiento,error,descripcion,fecha) Values ($
$PLSQL_UNIT, P_error, RST, SYSTIMESTAMP);
     commit;
End;
```

MODIFICA_PILOTO

```
create or replace PROCEDURE "MODIFICA_PILOTO" (
  P_cod_licencia In piloto.cod_licencia%Type,
  P_nombre In piloto.nombre%Type,
  P_nacionalidad In piloto.nacionalidad%Type,
  RST Out Nocopy Varchar)
Is
P_error Integer;
P_error_codigo exception;

Begin
RST:='Modificación de Piloto con cod_licencia:' || P_cod_licencia || '
correcta.';
Update piloto Set nombre = P_nombre, nacionalidad = P_nacionalidad WHERE
cod_licencia = P_cod_licencia;
P_error := SQLCODE;
IF (SQL%ROWCOUNT=0) Then
   Raise P_error_codigo;
END IF;
-- Actualizamos estadísticas
PKG_DW_ESTADISTICAS.TOP10_CARRERASACABADAS();
PKG_DW_ESTADISTICAS.TOP10_CARRERASGANADAS();
PKG_DW_ESTADISTICAS.PILOTO_COMP_ANO();
PKG_DW_ESTADISTICAS.CLASIFICACION_COMP_CURSO();
PKG_DW_ESTADISTICAS.TEMPERATURA_MAS_ALTA();
PKG_DW_ESTADISTICAS.PILOTO_MEJOR_EVOLUCION();
Commit;
Insert into logs(procedimiento,error,descripcion,fecha) Values ($$PLSQL_UNIT,
0, RST, SYSTIMESTAMP);
commit;
Dbms_output.Put_line(RST);

Exception

  When P_error_codigo Then
    P_error := SQLCODE;
    RST:='Error: El piloto no existe.';
    Dbms_output.Put_line(RST);
    Rollback;
```

```
    Insert into logs(procedimiento,error,descripcion,fecha) Values ($
$PLSQL_UNIT, P_error, RST, SYSTIMESTAMP);
    commit;

  When Others Then
     P_error:= SQLCODE;
     RST:= '¡Error!: ' || P_error ||' ' || SQLERRM;
     Dbms_output.Put_line(RST);
     rollback;
     Insert into logs(procedimiento,error,descripcion,fecha) Values ($
$PLSQL_UNIT, P_error, RST, SYSTIMESTAMP);
     commit;
End;
```

8.3. Trigger: identificativos en la tabla de registro bdr_Log

```
create or replace
trigger UOC.Insertar_ID_Logs
    Before Insert On uoc.Logs
    For Each Row
Begin
   Select UOC.insertar_id_logs.Nextval Into :New.ID
   From Dual;
End Insert_ID_Logs;
```

8.4. Trigger: componentes que aporta cada fabricante

```
trigger "T_FABRICANTE_COMP"
AFTER INSERT OR UPDATE OR DELETE ON BDR_COMPONENTE_MONTADO
FOR EACH ROW
BEGIN

IF (INSERTING OR UPDATING) THEN
update BDR_componente_marca set num_componentes = num_componentes+1
where cod_comp_marca = :new.cod_comp_marca;
if (sql%rowcount = 0) then
raise_application_error(-20000, 'Error. No existe el componente.');
end if;
END IF;


IF (UPDATING OR DELETING) THEN
update BDR_componente_marca set num_componentes = num_componentes-1
where cod_comp_marca = :old.cod_comp_marca;
if (sql%rowcount = 0) then
raise_application_error(-20000, 'Error. No existe el compoente.');
end if;
END IF;
end fabricante_comp;
```

8.5. Procedimiento TOP10_VUELTASMASRAPIDAS

```
 PROCEDURE TOP10_VUELTASMASRAPIDAS IS
    nombreProcedimiento VARCHAR (50) := 'TOP10_VUELTASMASRAPIDAS';
  BEGIN

  INSERT INTO FIA_DWH.top10_vueltasmasrapidas
  SELECT COD_RESULTADO,
```

```
        PILOTO,
        CIRCUITO,
        FECHA,
        TIEMPO,
        EQUIPO
    FROM (SELECT COD_RESULTADO,
                 PILOTO,
                 CIRCUITO,
                 FECHA,
                 TIEMPO,
                 EQUIPO,
      RANK() OVER (PARTITION BY CIRCUITO ORDER BY TIEMPO DESC) RANGO
        FROM (SELECT RE.COD_RESULTADO,
                     PI.NOMBRE PILOTO,
                     CI.DENOMINACION CIRCUITO,
                     CA.FECHA,
                     RE.VUELTA_MAS_RAPIDA TIEMPO,
            EQ.NOMBRE EQUIPO FROM bdr_RESULTADO RE,
                                  bdr_PILOTO PI,
                                  bdr_CIRCUITO CI,
                                  bdr_carrera CA,
                                  bdr_CONTRATO CO,
                                  bdr_EQUIPO EQ
           WHERE re.cod_piloto = co.cod_piloto AND
           CO.COD_PILOTO=pi.cod_licencia AND
           CO.COD_EQUIPO = EQ.COD_EQUIPO AND
           RE.COD_CARRERA = CA.COD_CARRERA AND
           ca.cod_circuito = CI.COD_CIRCUITO

        ) RANKING

    ) RESULTADO WHERE RESULTADO.RANGO <=10;


END TOP10_VUELTASMASRAPIDAS;
```

8.6. Procedimiento CARBURANTE_COCHE_ANO

```
PROCEDURE CARBURANTE_COCHE_ANO IS
    nombreProcedimiento VARCHAR (50) := 'CARBURANTE_COCHE_ANO';
  BEGIN

  INSERT INTO dwh_carburante_coche_ano
    SELECT COD_COCHE,
           MODELO,
           DEBUT,
           ANO,
           CONSUMO_TOTAL,
           COMPETICION
     FROM (SELECT COD_COCHE,
                  MODELO,
                  DEBUT, ANO,
                  CONSUMO_TOTAL,
                  COMPETICION,
     RANK() OVER (PARTITION BY  COMPETICION, ANO ORDER BY CONSUMO_TOTAL DESC)
RANGO
     FROM (SELECT CO.COD_COCHE,
                  CO.MODELO,
                  CO.DEBUT,
                  EXTRACT(YEAR from CA.FECHA) AS ANO,
                  SUM(RE.CONSUMO) AS CONSUMO_TOTAL,
                  COM.DESCRIPCION AS COMPETICION
```

```
                        FROM bdr_COCHE CO,
                              bdr_RESULTADO RE,
                              bdr_carrera CA,
                              BDR_competicion_anual CO_AN,
                              BDR_competicion com
                        WHERE re.cod_carrera = ca.cod_carrera and
                        ca.cod_competicion_anual= co_an.cod_competicion_anual
AND
                        co_an.cod_competicion = com.cod_competicion and
                        re.cod_coche = co.cod_coche
                          GROUP BY co.cod_coche,
                                   CO.modelo,
                                   CO.DEBUT,
                                   EXTRACT(YEAR FROM ca.fecha),
                                   com.descripcion) RANKING

    ) RESULTADO WHERE RESULTADO.RANGO <=1;
END CARBURANTE_COCHE_ANO;
```

8.7. Trigger T_defectuosos

```
trigger "T_DEFECTUOSOS"

AFTER INSERT OR DELETE OR UPDATE ON BDR_DATOSTELEMETRICOS
FOR EACH ROW
BEGIN
IF INSERTING OR UPDATING THEN
if (:NEW.valor = -1) THEN
UPDATE dwh_comp_defectuosos
SET COMP_DEFECTUOSOS = COMP_DEFECTUOSOS + 1
where DWH_COMP_DEFECTUOSOS.cod_coche =
    (select res.cod_coche
      from BDR_resultado res
    where res.cod_resultado =
    ( select te.cod_resultado
        from bdr_telemetria te
        where te.cod_telemetria = :new.cod_telemetria));
if (sql%rowcount = 0) then
insert into DWH_COMP_DEFECTUOSOS(COD_COCHE, NUM_COMPONENTES) VALUES
((select res.cod_coche
      from BDR_resultado res
        where res.cod_resultado =
        (select te.cod_resultado
        from bdr_telemetria te
        where te.cod_telemetria = :new.cod_telemetria)), 1 );
END IF;
end if;

end if;

IF  updating or deleting then

if (:OLD.valor = -1) THEN
UPDATE dwh_comp_defectuosos
SET COMP_DEFECTUOSOS = COMP_DEFECTUOSOS -1
where DWH_COMP_DEFECTUOSOS.cod_coche =
    (select res.cod_coche
    from BDR_resultado res
    where res.cod_resultado =
    (select te.cod_resultado
    from bdr_telemetria te
    where te.cod_telemetria = :old.cod_telemetria));
```

```
if (sql%rowcount = 0) then
  raise_application_error(-20000, 'Error. No existe el identificativo de
coche.');
END IF;
end if;

end if;

END t_defectuosos;
```

8.8. Procedimiento VOLUM_DATOS TELEMETRICOS

```
   /** MAYOR VOLUMEN DE DATOS TELEMETRICOS POR EQUIPO Y AÑO
  *
  *   @AUTHOR jrodriguezgomez
  **/


 PROCEDURE VOLUM_DATOSTELEMETRICOS_ANO
 IS
    nombreProcedimiento VARCHAR (50) := 'VOLUM_DATOSTELEMETRICOS_ANO';

  BEGIN
   EXECUTE IMMEDIATE 'TRUNCATE TABLE DWH_VOLUM_DTELEMETRICOS';
   Insert into dwh_volum_dtelemetricos (equipo, ano, volum_datos)
        SELECT  NOMBRE,
            ANO,
            VOLUMEN_DATOS

     FROM (SELECT NOMBRE,
                  ANO,
                  VOLUMEN_DATOS,
      RANK() OVER (PARTITION BY  ano ORDER BY VOLUMEN_DATOS DESC) RANGO
     FROM (SELECT EQ.NOMBRE,
                  EXTRACT(YEAR from CA.FECHA) AS ANO,
                  COUNT(*) AS volumen_datos

                        FROM
                              bdr_RESULTADO RE,
                              bdr_carrera CA,

                              bdr_equipo eq,
                              bdr_telemetria te,
                              BDR_datostelemetricos da,
                              BDR_CONTRATO CON
                        WHERE da.cod_telemetria = te.cod_telemetria AND
                              te.cod_resultado = re.cod_resultado AND
                              RE.cod_contrato = con.cod_contrato AND
                              CON.COD_EQUIPO = eq.cod_equipo and

                              re.cod_carrera = ca.cod_carrera


                          GROUP BY EXTRACT(YEAR from CA.FECHA), nombre

                                )  RANKING


    ) RESULTADO WHERE RESULTADO.RANGO <=1;


  P_error:= SQLCODE;
```

```
  RST:='Insercion realizada con exito.';
Insert into DWH_logs(procedimiento, error,descripcion,fecha) Values
(nombreProcedimiento, 0, RST, SYSTIMESTAMP);
COMMIT;
Dbms_output.Put_line(RST);

Exception
 When Others Then
     P_error:= SQLCODE;
     RST:= 'error!: ' || P_error ||' ' || SQLERRM;
     Dbms_output.Put_line(RST);
     rollback;
     Insert into DWH_LOGS(procedimiento,error,descripcion,fecha) Values
(nombreProcedimiento, P_error, RST, SYSTIMESTAMP);
     commit;
END VOLUM_DATOSTELEMETRICOS_ANO;
```

8.9. Consulta al repositorio estadístico

```
-- Procedimiento para obtener la clasificacion de cada competicion en curso
/**********
Parametros:
CLASIFICACION OUT RESULTADOS: Cursor con los resultados
RSP Out Nocopy Varchar: Control del resultado. Indica si se ha realizado con
exito o hubo algun error
***********/

PROCEDURE  C_CLASIFICACION_EN_CURSO (CLASIFICACION OUT RESULTADOS, RSP Out
Nocopy Varchar)  IS
   P_PROC_NOMBRE VARCHAR(50) := 'C_CLASIFICACION_EN_CURSO';
  BEGIN
  OPEN CLASIFICACION FOR
 SELECT * FROM dwh_clasificacion_en_curso;
P_error:= SQLCODE;
RSP:='Consulta realizada con exito.';
Insert into DWH_LOGS(procedimiento, error,descripcion,fecha) Values
(P_PROC_NOMBRE, 0, RSP, SYSTIMESTAMP);
COMMIT;
Dbms_output.Put_line(RSP);

Exception
 When Others Then
     P_error:= SQLCODE;
     RSP:= 'Error!: ' || P_error ||' ' || SQLERRM;
     Dbms_output.Put_line(RSP);
     rollback;
     Insert into DWh_logs(procedimiento,error,descripcion,fecha) Values
(P_PROC_NOMBRE, P_error, RSP, SYSTIMESTAMP);
     commit;


END C_CLASIFICACION_EN_CURSO;
```