Proyecto de Desarrollo de una Base de Datos para un concesionario

Etienne Boshoff de Jong
Enginyeria en Informàtica

Juan Martinez Bolaños

14 enero 2013

1. Resumen

Este documento corresponde a la Memoria del Proyecto Fin de Carrera (PFC) de los estudios de Ingeniería de Informática presentado por el alumno Etienne Boshoff de Jong, autor de este documento. El área escogido para el proyecto es “Base de Datos Relaciones”. Este trabajo corresponde al semestre Octubre 2012 – Enero 2013.

El proyecto ha consistido en desarrollar una *Base de Datos* para un cliente ficticio cuyas necesidades están descritas en el enunciado de la práctica presentado por el profesorado de la asignatura.

Esta *Base de Datos* debe proporcionar persistencia a una aplicación cuyo desarrollo no forma parte de este proyecto. La *Base de Datos* debe ofrecer los procedimientos y funciones almacenados necesarios para que la aplicación que interactúe con la *Base de Datos* no tenga que acceder directamente a las tablas.

Esta Memoria describe las diferentes fases de desarrollo del proyecto: Planificación, Análisis y Diseño. En la Memoria también se presenta una estimación económica del proyecto y se describe los mecanismos que la *Base de Datos* proporciona para permitir la integración con un *Data Warehouse* ajeno al presente proyecto. Por último, la Memoria incluye los guiones de testeo que permiten comprobar el correcto funcionamiento de la *Base de Datos*.

Como anexo a la Memoria, se entrega una presentación (en PowerPoint) que resume los detalles del proyecto. También se entrega como anexo, el código fuente desarrollado (en forma de sentencias SQL) para la creación de las tablas, triggers y los procedimientos y funciones almacenados junto con las instrucciones de la ejecución de estas sentencias.

2. Índice de contenido y de Figuras

Índice de contenido

Índice de Figuras

3. Introducción

3.1. Justificación del PFC: punto de partida y aportación del PFC

El cliente es un gran concesionario de vehículos de primera y segunda mano. A parte de dedicarse a la venta de vehículos, el cliente también se dedica a realizar reparaciones y revisiones de vehículos.

El cliente ya dispone de cierto grado de informatización. Las aplicaciones más importantes que el cliente utiliza son para gestionar la contabilidad y la facturación.

El Comité de Dirección de la organización cliente ha decidido recientemente que la empresa debe mejorar en el registro de la actividad de la empresa con el fin de poder analizar la actividad en más detalle. Se espera que el análisis más exhaustivo de la actividad permita facilitar la toma de decisiones estratégicas futuras.

Al Comité de Dirección de la empresa quiere que se registre la actividad acerca de las ventas de vehículos, las reparaciones de los vehículos y de sus revisiones. A partir del registro de esta información, interesa poder obtener estadísticas de por ejemplo, qué comerciales realizan más ventas, en qué épocas se realizan más ventas, qué vehículos son los más vendidos, qué mecánicos realizan el mayor número de revisiones y reparaciones y detalles del tipo de reparaciones que se realizan con más frecuencia.

Se ha hecho un estudio de aplicaciones informáticas existentes en el mercado actualmente que permitiría el registro de esta información. Aunque existan aplicaciones que cumplen los requerimientos iniciales, el Comité de Dirección cree que con el tiempo se va a querer ampliar las funcionalidades con respecto a lo indicado inicialmente.

Por tanto, se contrata un *software* a medida para la recogida de información sobre las ventas, revisiones y reparaciones. También, como proyecto aparte, se adquirirá un *Data Warehouse* para analizar la información registrada en la aplicación que debe ser desarrollada a medida.

El proyecto correspondiente al desarrollo del *software* a medida será dividida en dos proyectos que podrán desarrollarse de forma independiente. Por un lado se desarrollará la aplicación que los usuarios de la organización cliente utilicen para registrar la actividad y por otro, la base de datos que le proporcione persistencia a dicha aplicación. Esta separación permite el desarrollo del proyecto por fases y distribuir la inversión a lo largo de un periodo más largo.

El proyecto considerado en este documento corresponde al desarrollo de la *Base de Datos* mencionado. La aplicación al que la *Base de Datos* proporcione persistencia accederá a la *Base de Datos* a través de los procedimientos y funciones almacenados de la *Base de Datos* y nunca directamente a sus tablas. La *Base de Datos* también deberá proporcionar los mecanismos necesarios para la integración con el *Data WareHouse*.

3.2. Objetivos del PFC

Los objetivos del proyecto son los siguientes:

- **Proporcionar persistencia al *software* a medida que debe desarrollarse para el cliente**.
  La *Base de Datos* de este proyecto debe ser el único mecanismo de persistencia para el *software* encargado por cliente.

- **Permitir al *software* a medida ignorar la estructura interna de la *Base de Datos* ofreciendo las funcionalidades de acceso a través de un conjunto de procedimientos y funciones almacenados.**
  La *Base de Datos* deberá proporcionar los procedimientos y funciones almacenados necesarios para para evitar que el *software* a medida tenga que acceder directamente a las tablas de la *Base de Datos*.

- **Proporcionar documentación que describe la estructura de la *Base de Datos* y sus procedimientos y funciones almacenados.**
  El proyecto de desarrollo de la *Base de Datos* deberá incluir documentación que describe la estructura de la Base Datos y los procedimientos y funciones ofrecidas.

- **La Base de Datos deberá cumplir con los requerimientos identificados en la definición del Alcance Inicial.**
  Los requerimientos del Alcance Inicial son los siguientes:

  - **R1: Gestión de Ventas.**
    Descripción: Registro y modificación de ventas. Sobre una venta, se ha de poder identificar el coche vendido, el modelo y extras del coche vendido, el cliente que ha adquirido el coche, la fecha en la que se ha vendido el coche y el comercial responsable de la venta.

  - **R2: Gestión de reparaciones y revisiones**.
    Descripción: Registro y modificación de reparaciones y revisiones y la planificación de revisiones futuras. Para cada reparación o revisión se debe poder gestionar la información identificativa del cliente, el motivo de la reparación, el tiempo que se ha tardado en realizar la reparación o revisión y el mecánico responsable. De las revisiones se ha de poder saber la fecha de planificación y si ya se ha realizado o no.

  - **R3: Gestión de coches y sus características.**
    Descripción: Registro y modificación de datos sobre los coches: modelo, matrícula, extras/características y propietario.

  - **R4: Escalabilidad**.

Descripción: La Base de Datos ha de ser escalable permitiendo la incorporación progresiva de nuevas funcionalidades. Concretamente, la *Base de Datos* simulará la posibilidad de agregar atributos a las tablas principales de forma dinámica. También se ha de poder definir dinámicamente entidades adicionales para gestionar la información de referencia.

- **R5: Integración con un *Data Warehouse.***
  Descripción: *La Base de Datos* debe disponer de mecanismos que permitan el traspaso de datos desde las tablas de la *Base de Datos* hacia un *Data Warehouse*.

- **R6: Gestión del registro de acciones.**
  Descripción: La *Base de Datos* deberá mantener un registro de las acciones llevados a cabo por los usuarios.

- **R7: Mecanismos para el testeo de la *Base de Datos*.**
  Descripción: El proyecto debe incluir mecanismos para poder ser probado. Concretamente, la documentación que describe la *Base de Datos* deberá incluir guiones de testeo para poder testear el funcionamiento de la *Base de Datos.*

Se ha definido los objetivos y requerimientos siguiendo los principios SMART [1]:

- *Specifics* (específicos, sin ambigüedades)
- Measurable (cuantificables)
- Agreed (acordados con los interesados)
- Realistic (asumibles dentro de las limitaciones técnicas, alcance, tiempo y calendario)

3.3. Enfoque y método seguido

Se ha considerado las fases del proyecto según la definición PMBOCK [1]:

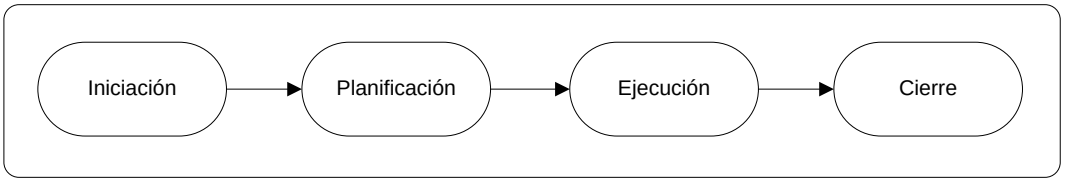


Figura 1. Fases PMBOCK.

Dentro de cada fase, se definen las acciones siguientes:

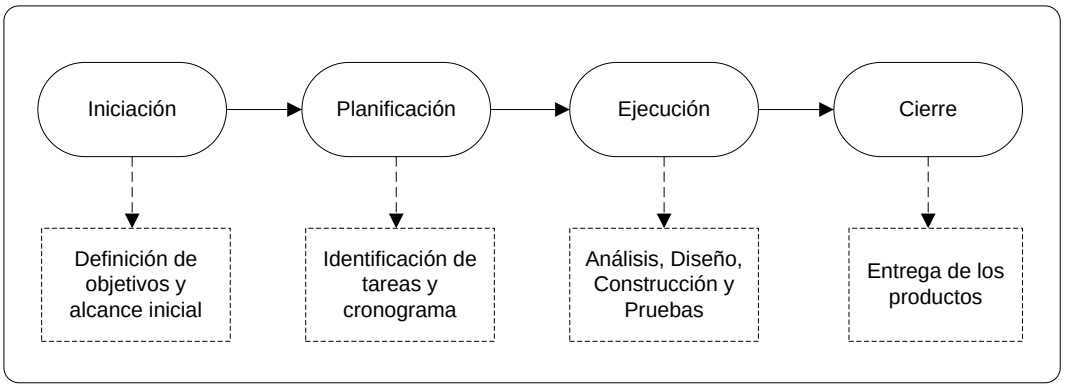


Figura 2. Acciones en cada fase

A continuación se describe las diferentes fases y acciones:

- **Iniciación**: Definición de los objetivos y el alcance inicial (por favor, vea el apartado *Objetivos del PFC*).
- **Planificación**: Identificación de tareas, estimación de esfuerzo para cada tarea y cronograma. Identificación de riesgos y plan de contingencia.
- **Análisis**: Definición del Alcance Detallado, Diagrama de Casos de Uso y Modelo Conceptual de la *Base de Datos*.
- **Diseño**: Definición del *Modelo Lógico* de la *Base de Datos* e identificación de las estructuras necesarias (procedimientos, funciones, t*riggers*...).
- **Construcción**: Desarrollo de las sentencias SQL que permiten la construcción de las tablas y demás estructuras de la *Base de Datos*.
- **Pruebas**: Redacción y ejecución de los guiones de pruebas. Gestión de incidencias.
- **Cierre**: Entrega de productos finales.

Para el desarrollo de las fases se ha seguido una mezcla entre la metodología ***Clásica (en Cascada)*** y la de ***ciclos de vida iterativos e incrementales*** [2].

Se siguió la metodología Clásica para el desarrollo de las fases del proyecto. Dentro de la fase Ejecución, se ha seguido la metodología iterativa para las acciones de construcción de los procedimientos, funciones y triggers y la realización de pruebas. Para ello, se dividió el conjunto de procedimientos, funciones y triggers en dos grupos o bloques. Durante la primera iteración, se construyó los procedimientos, funciones y *triggers* del primer grupo, se redactó los guiones correspondientes, se ejecutó los guiones y se solucionó las incidencias correspondientes. A continuación se completó la segunda iteración siguiendo los mismos pasos que la iteración anterior, pero sobre los procedimientos, funciones y triggers del segundo grupo o bloque.

El diagrama siguiente describe el método seguido es el siguiente:

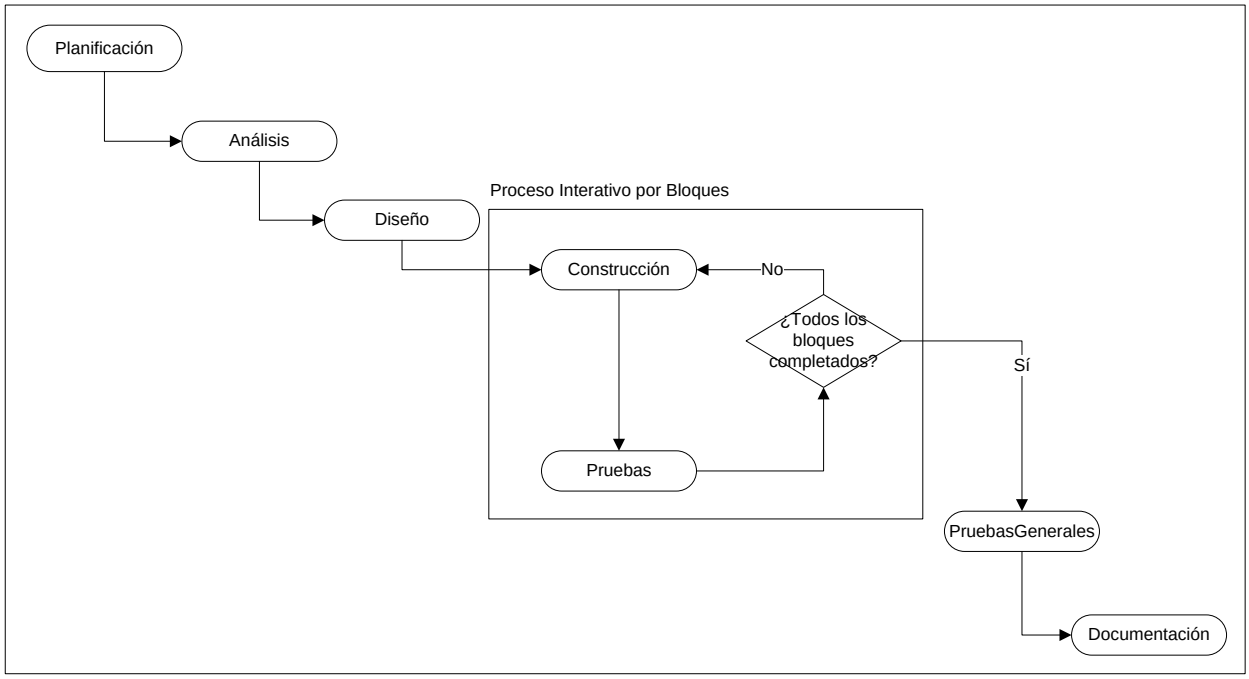


Figura 3. Descripción método seguido

3.4. Planificación del proyecto

En la fase de Iniciación del proyecto, entre otros, se identificó los objetivos del proyecto y el alcance inicial (por favor, ver el apartado *Objetivos del PFC*).

Para la fase de Planificación, se identificaron las tareas concretas que deben ser realizadas durante el proyecto, su duración aproximada y el intervalo de fechas en que deban realizarse. Se ha definido un diagrama de Gantt para mostrar la planificación temporal de las tareas. También se ha identificado los riesgos para el proyecto y el plan de contingencia correspondiente.

A través de los apartados siguientes, se describen los diferentes componentes de la planificación.

3.4.1. Identificación de tareas concretas

A continuación se muestra la lista de tareas identificadas para la realización del proyecto [1]. Las tareas están clasificadas en los grupos generales siguientes: Preparación del proyecto, Análisis, Diseño, Construcción de tablas, Construcción de estructuras del grupo 1, Construcción de estructuras del grupo 2 y Pruebas generales (grupos 1 y 2). Se incluye una descripción de cada tarea y su duración aproximada.

Preparación del Proyecto

- **Tarea: Instalación de Oracle en estación de trabajo. Duración aproximada: 5 horas.**
  Instalación del SGBD Oracle para poder empezar el desarrollo y pruebas. Se instalará el *Oracle Database 10g Express Edition*.

- **Tarea: Creación de tabla y procedimientos de pruebas. Duración aproximada: 5 horas.**
  Creación de una tabla y algunos procedimientos con el fin de probar que la instalación y configuración inicial sea correcta.

Análisis

- **Tarea: Refinar requerimientos. Duración aproximada: 2 horas.**
  Se parte del *alcance inicial* definido en la fase Inicial del proyecto y se mantiene reuniones con los interesados del proyecto para definir el *alcance detallado*. El *alcance detallado* consiste en el conjunto de requerimientos concretados y refinados a partir de los que se realizará el diseño y posteriormente, el desarrollo de la *Base de Datos*.

- **Tarea: Diagrama de Casos de Uso. Duración aproximada: 4 horas.**

Se diseña el *diagrama de Casos de Uso* con el fin de identificar las funcionalidades principales que el sistema tendrá desde el punto de vista de los usuarios finales.

- **Tarea: Modelo Conceptual de la Base de Datos. Duración aproximada: 8 horas.**
  Una vez que se tenga detallado las características que el sistema deba tener, se realiza el diagrama Entidad-Relación UML y la descripción de las entidades y relaciones.

Diseño

- **Tarea: Decisiones de Diseño. Duración aproximada: 4 horas.**
  Documentación de las decisiones de diseño que se han tomado.

- **Tarea: Modelos Lógico de la Base de Datos. Duración aproximada: 4 horas.**
  Diagrama Entidad-Relación para describir el modelo lógico de la *Base de Datos*.

- **Tarea: Identificación de Procedimientos, Funciones y Triggers a implementar. Duración aproximada: 10 horas.**
  Identificación de los Procedimientos, Funciones y Triggers que deberán ser implementados.

- **Tarea: División de los Procedimientos, Funciones y Triggers identificados en dos grupos. Duración aproximada: 2 horas.**
  Se dividen los Procedimientos, Funciones y Triggers identificados en 2 grupos con el fin de desarrollarlos por separado.

Construcción tablas

- **Tarea: Creación de tablas en la Base de Datos. Duración aproximada: 6 horas.**
  Componer un script las sentencias SQL necesarias para crear las tablas del Modelo Lógico.

Construcción (estructuras del grupo 1)

- **Tarea: Creación de procedimientos, funciones y triggers. Duración aproximada: 10 horas.**
  Componer y ejecutar el script con las sentencias SQL necesarias para crear los procedimientos, funciones y triggers del grupo 1.

- **Tarea: Desarrollo de los guiones de pruebas. Duración aproximada: 4 horas.**
  Se desarrollan los guiones de prueba para probar las estructuras construidas del grupo 1.

- **Tarea: Ejecución de los guiones de pruebas. Duración aproximada: 2 horas.**
  Se ejecutan los guiones de prueba para comprobar que los guiones ya la construcción sea correcta.

- **Tarea: Gestión de incidencias. Duración aproximada: 6 horas.**
  Se arreglan las incidencias detectadas durante la ejecución de los guiones de prueba.

Construcción (estructuras del grupo 2)

- **Tarea: Creación de procedimientos, funciones y triggers. Duración aproximada: 10 horas.**
  Componer y ejecutar el script con las sentencias SQL necesarias para crear los procedimientos, funciones y triggers del grupo 1.

- **Tarea: Desarrollo de los guiones de pruebas. Duración aproximada: 4 horas.**
  Se desarrollan los guiones de prueba para probar las estructuras construidas del grupo 1.

- **Tarea: Ejecución de los guiones de pruebas. Duración aproximada: 2 horas.**
  Se ejecutan los guiones de prueba para comprobar que los guiones ya la construcción sea correcta.

- **Tarea: Gestión de incidencias. Duración aproximada: 6 horas.**
  Se arreglan las incidencias detectadas durante la ejecución de los guiones de prueba.

Pruebas generales (grupos 1 y 2) u documentación

- **Tarea: Preparación de la Base de Datos para las pruebas. Duración aproximada: 2 horas.**
  Se elimina el contenido de las tablas con el fin de poder ejecutar los guiones de pruebas de nuevo.

- **Tarea: Ejecución de los guiones de pruebas. Duración aproximada: 2 horas.**
  Se ejecutan los guiones de prueba para comprobar que los guiones ya la construcción sea correcta.

- **Tarea: Gestión de incidencias. Duración aproximada: 2 horas.**
  Se arreglan las incidencias detectadas durante la ejecución de los guiones de prueba.

Documentación

- **Tarea: Documentación. Duración aproximada: 16 horas.**
  Se redacta la documentación final del proyecto.

3.4.2. Calendario de tareas y sus duraciones: Diagrama de *Gantt*

A continuación de muestra el cronograma para la realización de las tareas del apartado anterior:

ID	Task Name	Start	Finish	Duration	% Complete
1	Instalación de Oracle en estación de trabajo	10/22/2012	10/26/2012	5d	0%
2	Creación de tabla y procedimientos de pruebas	10/22/2012	10/26/2012	5d	0%
3	Refinar Requerimientos	10/29/2012	10/29/2012	1d	0%
4	Diagrama de casos de uso	11/1/2012	11/2/2012	2d	0%
5	Modelo Conceptual de la BBDD	11/5/2012	11/8/2012	4d	0%
6	Decisiones de Diseño	11/13/2012	11/14/2012	2d	0%
7	Modelo Lógico de la BBDD	11/15/2012	11/16/2012	2d	0%
8	Identificación de Procedimientos, Funciones y Triggers a implementar	11/19/2012	11/23/2012	5d	0%
9	Dividir Procedimientos, Funciones y Triggers identificados en 2 grupos	11/23/2012	11/23/2012	1d	0%
10	Creación de las tablas en la BBDD	11/26/2012	11/28/2012	3d	0%
11	Creación de los Proc, Funciones y Triggers (grupo 1) en BBDD	11/26/2012	11/30/2012	5d	0%
12	Desarrollo de guiones de prueba (grupo 1)	12/3/2012	12/4/2012	2d	0%
13	Ejecución de los guiones de pruebas (grupo 1)	12/7/2012	12/7/2012	1d	0%
14	Gestión de Incidencias	12/10/2012	12/12/2012	3d	0%
15	Creación de los Proc, Funciones y Triggers (grupo 2) en BBDD	12/17/2012	12/21/2012	5d	0%
16	Desarrollo de guiones de prueba (grupo 2)	12/26/2012	12/27/2012	2d	0%
17	Ejecución de los guiones de pruebas (grupo 2)	12/28/2012	12/28/2012	1d	0%
18	Gestión de incidencias	12/31/2012	12/31/2012	1d	0%
19	Preparación de las tablas para repetir las pruebas (Borrado del contenido)	1/2/2013	1/2/2013	1d	0%
20	Ejecución de guiones de prueba (grupos 1 y 2)	1/3/2013	1/3/2013	1d	0%
21	Gestión de incidencias	1/4/2013	1/4/2013	1d	0%
22	Documentación Final	1/7/2013	1/16/2013	8d	0%

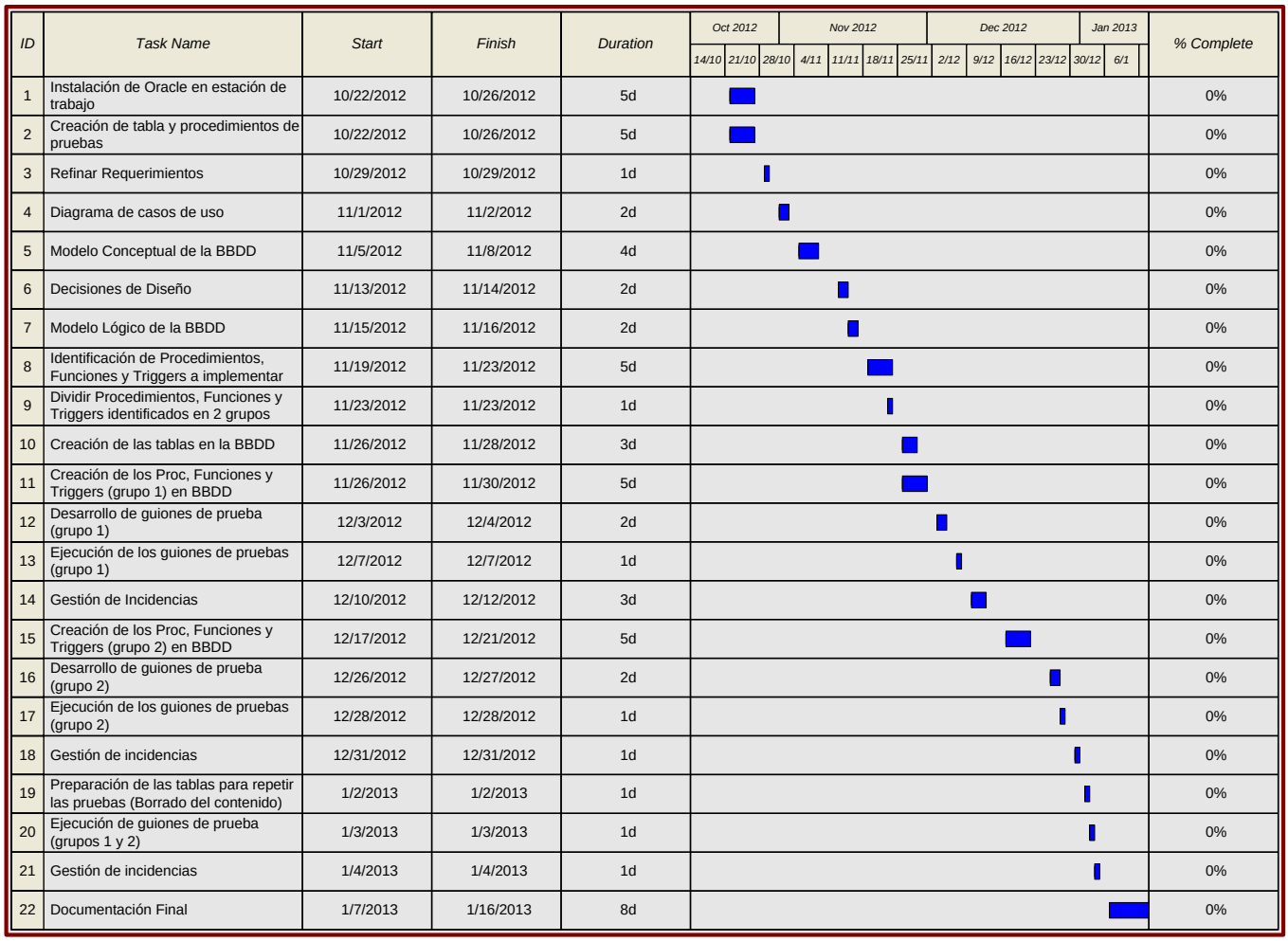


Figura 4. Cronograma

Para cada tarea, se muestra una descripción corta, la fecha en que debe comenzar, la fecha en que debe haberse finalizado, la duración en días entre la fecha inicio y la fecha fin, una visualización gráfica de la extensión de cada tarea a lo largo del calendario y el porcentaje de finalización en el momento de realizar la planificación.

3.4.3. Hitos principales

Los hitos principales del proyecto son los siguientes:

Entrega de la PAC 1: 8 de octubre
La PAC 1 consiste en la entrega de la planificación del proyecto. Esta planificación debe incluir los objetivos del proyecto, las tareas identificadas para el desarrollo del proyecto, una estimación aproximada de la duración de cada tarea, la planificación temporal de las tareas (diagrama de Gant) y un plan de riesgos.

Entrega de la PAC 2: 12 de noviembre
En la PAC 2 se entrega el diseño de la aplicación a implementar. Este diseño incluye el refinamiento de los requerimientos identificado en este plan de trabajo, el diagrama Entidad-Relación correspondiente al Modelo Lógico de la Base de Datos y la identificación de los Procedimientos, Triggers y Funciones que se desarrollarán.

Entrega de la PAC 3: 13 de diciembre
Se entrega la construcción de las tablas y procedimientos, funciones y triggers del primer grupo. Se incluye la fase de Pruebas y Resolución de Incidencias correspondientes.

Entrega Final: 14 de enero
Entrega de la Memoria final del proyecto, la presentación y el producto desarrollado.

3.4.4. Identificación de riesgos y plan de contingencia

El principal riesgo observado para este proyecto es el siguiente:

Riesgo R001: Conflicto que pueda surgir con otros proyectos. Hay otros dos proyectos planificados para pasar a producción durante los meses de Octubre y Noviembre. Estas puestas en producción podrían requerir una dedicación mayor de lo deseado y así poner en peligro el desarrollo del PFC.

Plan de contingencia para R001: Se re-planifica las vacaciones del personal para después de la entrega de la documentación final. Ya se han establecidos los acuerdos convenientes con el personal que pueda ser afectado.

3.5. Productos obtenidos

Como resultado del desarrollo de este proyecto se obtiene los productos siguientes:

- **Memoria del proyecto**: documento descriptivo de la estructura del proyecto y del trabajo realizado. Incluye los guiones de pruebas que permiten la comprobación del funcionamiento de la *Base de Datos* desarrollada.
- **Sentencias SQL para la creación de tablas y demás estructuras de la *Base de Datos*.** Se trata de un fichero de texto con las sentencias. El fichero incluye las instrucciones de ejecución de estas sentencias.
- **Presentación en PowerPoint.** La presentación resume el contenido del documento Memoria del proyecto.

3.6. Breve descripción del resto de capítulos

Los siguientes capítulos de este apartado corresponden al Análisis, Diseño, Guiones de Pruebas, Integración con el *Data Warehouse*, la Valoración económica y las conclusiones del desarrollo del proyecto.

Capítulo Análisis

El capítulo correspondiente al Análisis incluye lo siguiente:

- El alcance detallado, es decir, el refinamiento de los requerimientos del alcance inicial.
- El Modelo de Casos de uso junto con una descripción de los actores identificados y de cada caso de uso.
- El Modelo Conceptual de la *Base de Datos* que incluye el diagrama Entidad-Relación y una descripción de las entidades y de las relaciones entre ellas.
- Una descripción de las decisiones de diseño más relevantes.

Capítulo Diseño

El capítulo acerca de la fase del Diseño del proyecto describe el Modelo Lógico de la Base de Datos a través de un Diagrama Entidad-Relación detallado. Se incluye la descripción de las tablas, índices, claves primarias y extranjeras, secuencias y restricciones sobre el contenido de los datos de los atributos de las tablas.

Capítulo Guiones de Prueba

Este capítulo presenta los guiones de prueba que permite probar que funcionamiento correcto de la *Base de Datos*. Cada guión consta de una secuencia de pasos a ejecutar. Para cada paso, los guiones especifican:

- instrucciones sobre cómo ejecutar el paso
- la funcionalidad concreta que el paso permite probar
- el resultado esperado

Los guiones dejan un espacio al lado de cada paso para especificar si el resultado obtenido como consecuencia de la ejecución del paso es el esperado. También, para cada paso hay un espacio para realizar anotaciones sobre el resultado devuelto.

Capítulo Integración con la *Data Warehouse*

En este capítulo se describe la manera en que la Base de Datos integra con un *Data Warehouse*. La integración consiste en la exportación del contenido de las tablas a ficheros CSV que deben ser importados al *Data Warehouse*.

Capítulo Valoración económica

Se realiza una estimación sobre el coste que ha requerido el desarrollo del proyecto y se compara con la estimación teórica obtenido a partir de las metodologías *Líneas de código* y *Puntos de Función*.

3.7. Análisis

El Análisis incluye los apartados siguientes:

- Definición del alcance detallado
- Modelo de Casos de uso
- Modelo conceptual de la Base de datos
- Decisiones de diseño relevantes

3.7.1. Definición del alcance detallado

Cada requerimiento del *alcance inicial* ha sido refinado en requerimientos más concretos y detallados [1]. El conjunto de requerimientos concretos y detallados definen el *alcance detallado*. Se partirá de los requerimientos definidos en el *alcance detallado* para realizar el diseño y la implementación.

A continuación se muestra para cada requerimiento inicial, los requerimientos refinados correspondientes:

Requerimiento inicial R1: Gestión de Ventas.

Descripción: Registro y modificación de ventas. Sobre una venta, se ha de poder identificar el coche vendido, el modelo y extras del coche vendido, el cliente que ha adquirido el coche, la fecha en la que se ha vendido el coche y el comercial responsable de la venta.

Requerimientos derivados:

- **R1-001**: Para cada venta se permite registrar y modificar la fecha/hora, el comercial responsable, la identificación del coche vendido, el modelo del coche junto con sus extras o características y el cliente que ha comprado el coche.
- **R1-002**: Un cliente ha de poder comprar más de un coche.
- **R1-003**: Para un cliente comprador, se ha de poder registrar su nombre, teléfono de contacto, NIF, dirección, sexo y fecha de nacimiento.
- **R1-004**: Se ha de poder identificar los comerciales en el sistema y han de ser usuarios permitiéndoles registrar sus propias ventas.

Requerimiento inicial R2: Gestión de reparaciones y revisiones.

Descripción: Registro y modificación de reparaciones y revisiones y la planificación de revisiones futuras. Para cada reparación o revisión se debe poder gestionar la información identificativa del cliente, el

motivo de la reparación, el tiempo que se ha tardado en realizar la reparación o revisión y el mecánico responsable. De las revisiones se ha de poder saber la fecha de planificación y si ya se ha realizado o no.

Requerimientos derivados:

- **R2-002**: Para cada reparación/revisión se permite registrar y modificar la descripción del servicio prestado, el coche afectado, la fecha y hora, la duración de la reparación/servicio y el técnico responsable.
- **R2-003**: se ha de poder programar reparaciones/revisiones futuras.
- **R2-003**: debe de poderse definir un catálogo con los diferentes tipos de reparaciones y revisiones.
- **R2-004**: el cliente ha de poder mantener el catálogo de reparaciones y revisiones (R2-003).

Requerimiento inicial R3: Gestión de coches y sus características

Descripción: Registro y modificación de datos sobre los coches: modelo, matrícula, extras/características y propietario.

Requerimientos derivados:

- **R3-001**: Para cada coche que se registra, se ha de poder especificar su modelo.
- **R3-002**: Para cada modelo registrado, el cliente ha de poder definir los extras.
- **R3-003**: Para coche se debe registrar el propietario actual (sin perder el histórico sobre ventas anteriores).

Requerimiento inicial R4: Escalabilidad

Descripción: La Base de Datos ha de ser escalable permitiendo la incorporación progresiva de nuevas funcionalidades. Concretamente, la *Base de Datos* simulará la posibilidad de agregar atributos a las tablas principales de forma dinámica. También se ha de poder definir dinámicamente entidades adicionales para gestionar la información de referencia.

Requerimientos derivados:

- **R4-001**: la *Base de Datos* ha de permitir simular la ampliación de los atributos de las tablas de actividad del sistema de forma dinámica, sin requerir la modificación de la estructura de las tablas.
- **R4-002**: la *Base de Datos* ha de permitir la definición de entidades adicionales para gestionar las categorías de la información de referencia.
- **R4-003**: la *Base de Datos* debe ofrecer mecanismos que garantizan la integridad sobre la ampliación de atributos.

Requerimiento inicial R5: Integración con un *Data Warehouse*

Descripción: *La Base de Datos* debe disponer de mecanismos que permitan el traspaso de datos desde las tablas de la *Base de Datos* hacia un *Data Warehouse*.

Requerimientos derivados:

- **R5-001**: La *Base de Datos* dispondrá de los procedimientos necesarios para facilitar la exportación del contenido de las tablas de actividad del sistema (entre ellas, la tabla de Ventas y la de reparaciones/revisiones).
- **R5-002**: La *Base de Datos* permitirá la definición de códigos para los elementos de los diferentes catálogos con el fin de facilitar la explotación de la información en el *Data Warehouse*.

Requerimiento inicial R6: Gestión del registro de acciones

Descripción: La *Base de Datos* deberá mantener un registro de las acciones llevados a cabo por los usuarios.

Requerimientos derivados:

- **R6-001**: El sistema dispondrá de procedimientos para registrar y consultar el histórico de las acciones realizadas sobre los registros de las tablas.
- **R6-002**: Por acciones, se entiende Agregaciones y Modificaciones de registros.
- **R6-003**: No se requiere mantener el histórico a nivel de los valores de los campos. Es decir, no se requiere conocer valores anteriores de los atributos.

Requerimiento inicial R7: Mecanismos de testeo de la *Base de Datos*

Descripción: El proyecto debe incluir mecanismos para poder ser probado. Concretamente, la documentación que describe la *Base de Datos* deberá incluir guiones de testeo para poder testear el funcionamiento de la *Base de Datos.*

Requerimientos derivados:

- **R7-001**: El proyecto incluirá guiones de pruebas con pasos concretos y criterios específicos para poder comprobar el funcionamiento de la *Base de Datos* de forma objetiva.

3.7.2. Modelo Casos de Uso

A continuación se describen los actores y casos de uso identificados [3].

El modelo consta de 3 actores diferenciados: Técnico del Sistema, Comercial, Mecánico y Administrativo.

El Técnico del Sistema identifica los usuarios informáticos que configuran el sistema. Los Administrativos deben poder realizar todas las acciones no propias de los Técnicos del Sistema. Los Administrativos representan el mayor grupo de usuarios del sistema.

Aunque no sea frecuente que los Mecánicos y Comerciales utilicen el sistema, tendrán acceso al sistema y deberán poder registrar información en el sistema sin depender necesariamente de una persona administrativa. Por eso comparten casos de uso con los administrativos.

El Técnico del Sistema es el actor principal de los casos de uso "Gestionar Entidades Maestras", "Gestionar Usuarios" y "Exportar Datos". El caso de uso "Gestionar Atributos Virtuales" permite especificar un valor para un atributo virtual de un registro de una tabla de información de actividad. Un atributo virtual es un mecanismo del sistema para poder simular el añadir un atributo a una tabla sin modificar la estructura original de la tabla.

El Administrativo puede ejecutar los casos de usos: "Gestionar Atributos Virtuales", "Gestionar Personas", "Gestionar Ventas", "Gestionar Coches" y "Gestionar Servicios".

El Comercial puede ejecutar los casos de usos: "Gestionar Atributos Virtuales", "Gestionar Personas", "Gestionar Ventas" y "Gestionar Coches".

El Mecánico puede ejecutar los casos de usos: "Gestionar Atributos Virtuales", "Gestionar Personas", y "Gestionar Servicios" y "Gestionar Coches".

Los casos de uso "Gestionar Usuarios" y "Gestionar Coches" requieren que previamente se haya creado un registro en la tabla Persona. Por eso mantienen una relación de *include* con el caso de uso "Gestionar Personas". Todo usuario es también persona. Para el caso de uso "Gestionar Coches", la persona corresponde al propietario del coche.

Los casos de uso "Gestionar Ventas" y "Gestionar Servicios" requieren que previamente se haya creado un registro en la tabla Usuarios y además, en la tabla Coches. Por tanto, mantienen una relación de *include* con el caso de uso "Gestionar Usuarios" y con el caso de uso "Gestionar Coches". Para el caso de uso "Gestionar Ventas" el usuario identifica el comercial responsable de la venta. Para el caso de uso "Gestionar Servicios", el usuario identifica el mecánico responsable del servicio (reparación/revisión).

El diagrama de casos de uso que describe este análisis es el siguiente:

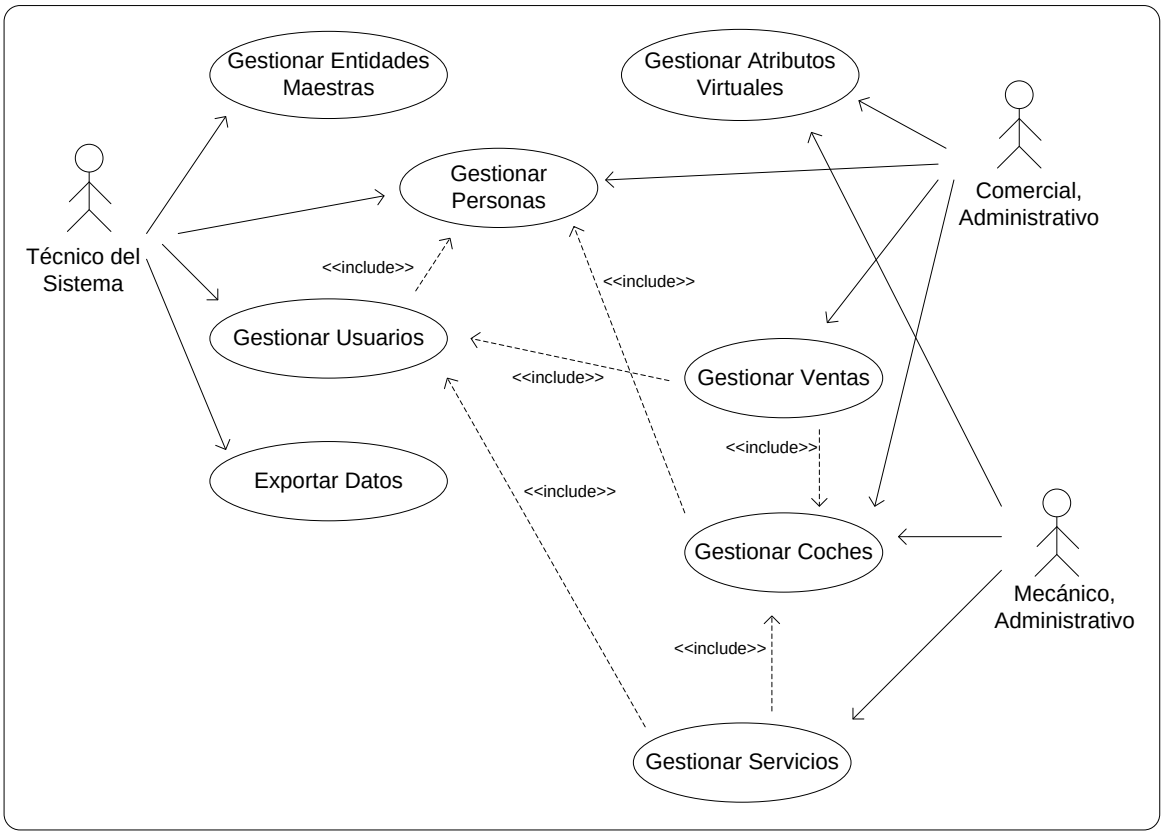


Figura 5. Diagrama Casos de uso

A continuación se describen los casos de uso de forma textual:

Caso de Uso 1: Gestionar Entidades Maestras

Nombre: Gestionar Entidades Maestras

Actor: Técnico del Sistema

Descripción: Permite mantener la información de referencia en el sistema. La información de referencia consiste en las entidades (conjuntos de valores) municipios, provincias, modelos de coches, extras, perfiles de usuario entre otros.

Flujo de eventos principal:

1. El usuario indica la opción de consultar las entidades de información de referencia.
2. El sistema muestra las entidades de información de referencia.
3. El usuario indica la opción de agregar una nueva entidad de información de referencia.
4. El sistema añade la nueva entidad de información de referencia.
5. El usuario especifica los valores (o registros) que el sistema deba añadir para la entidad de información de referencia.
6. El sistema añade los valores (o registros) especificado por el usuario.

Flujo de eventos alternativos:

5.1. El usuario especifica una modificación para un valor (o registro).

5.2. El sistema actualiza el valor (o registro) según la modificación especificada por el usuario.

Caso de Uso 2: Gestionar Atributos Virtuales

Nombre: Gestionar Atributos Virtuales.

Actor: Administrativo, Comercial, Mecánico.

Descripción: Permite especificar un valor para un atributo virtual de un registro de una tabla de información de actividad. Un atributo virtual es un mecanismo del sistema para poder simular el añadir un atributo a una tabla sin modificar la estructura original de la tabla.

Flujo de eventos principal:

1. El usuario indica la opción de especificar un valor para un Atributo Virtual y especifica los detalles necesarios (tipo de atributo, registro siendo extendido y valor del atributo).
2. El sistema añade el valor para el atributo y registro.

Flujo de eventos alternativo:

1. El usuario indica la opción de modificar el valor de un Atributo Virtual existente y especifica los detalles necesarios (tipo de atributo, registro siendo extendido y valor del atributo) para su actualización.
2. El sistema actualiza el valor para el atributo y registro.

Excepciones: Se anula la transacción en los casos siguientes:

- el atributo no fue definido correctamente
- el atributo especificado no existe
- el registro referenciado no existe en la tabla identificada por la definición del atributo
- el valor especificado para el atributo no es adecuado según la definición del atributo
- al agregar: si ya existe un valor para el atributo y registro
- al modificar: si no existe un valor para el atributo y registro especificados

Caso de Uso 3: Gestionar Usuarios

Nombre: Gestionar Usuarios

Actor: Técnico del Sistema

Descripción: Permite crear o modificar usuarios en el sistema. Al crear un usuario, se debe especificar el nombre, los apellidos, el nombre de usuario y el perfil de seguridad. La contraseña no es gestionada por la Base de Datos.

Flujo de eventos principal:

1. El usuario indica la opción para agregar un usuario nuevo y especifica los detalles.
2. El sistema crea el usuario nuevo en el sistema.

Flujo de eventos alternativo:

1. El usuario indica la opción para modificar un usuario existente y especifica los detalles.
2. El sistema crea el usuario nuevo en la Base de Datos.

Excepciones: Se anula la transacción en los casos siguientes:

- el valor especificado para el perfil del usuario no es correcto
- el registro referenciado en la tabla Personas no existe
- al agregar: si ya existe un usuario para la persona referenciada

Caso de Uso 4: Exportar Datos

Nombre: Exportar Datos

Actor: Técnico del Sistema, Aplicación

Descripción: Este caso de uso permite la exportación de datos que han sido modificados después de la fecha que el usuario especifica como parámetro. La exportación se realiza por tabla y deja los datos en ficheros CSV.

Flujo de eventos principal:

1. El usuario indica la opción de exportar datos, la tabla de la que desea exportar y la fecha deseada.
2. El sistema genera un fichero CSV con los datos de la tabla correspondiente que hayan sido modificados después de la fecha especificado como parámetro.

Excepciones:

- La fecha especificada no es válida

Caso de Uso 5: Gestionar Ventas

Nombre: Gestionar Ventas

Actor: Comercial, Administrativo

Descripción: Permite registrar una venta o modificar los detalles de una venta existente. Estas acciones tanto las podrá realizar un Administrativo como un Comercial.

Flujo de eventos principal:

1. El usuario indica la opción de registrar una nueva venta y especifica los detalles de la venta.
2. El sistema registra la nueva venta en la Base de Datos.

Flujo de eventos alternativo:

1. El usuario indica la opción de modificar una venta existente y especifica los detalles de la venta.
2. El sistema actualiza los detalles de la venta en la Base de Datos.

Excepciones: Se anula la transacción en los casos siguientes:

- el coche referenciado no existe
- el comercial referenciado no existe

Caso de Uso 6: Gestionar Servicios

Actor: Administrativo, Mecánico

Descripción: Permite registrar o modificar un servicio. Se entiende como servicio una reparación o una revisión tanto si está realizada como si está planificada.

Flujo de eventos principal:

1. El usuario indica la opción de registrar un servicio nuevo y especifica los detalles del servicio.
2. El sistema registra el nuevo servicio en la Base de Datos.

Flujo de eventos alternativo:

1. El usuario indica la opción de modificar un servicio existente y especifica los detalles del servicio.
2. El sistema actualiza los detalles del servicio en la Base de Datos.

Excepciones: Se anula la transacción en los casos siguientes:

- el valor para el Tipo de Actividad (cambio de aceite, arreglo limpiaparabrisas,...) especificado no es adecuado.
- el coche referenciado no existe
- el técnico referenciado no existe
- el valor para el Tipo de Servicio (reparación, revisión) especificado no es adecuado
- no se ha especificado una fecha de realización ni una fecha de planificación
- si se ha especificado una fecha de realización pero no se ha especificado el mecánico o no se ha especificado un valor para la duración
- si se ha especificado una fecha de planificación y los campos duración o técnico contienen algún valor.

Caso de Uso 7: Gestionar Personas

Nombre: Gestionar Personas

Actor: Técnico del Sistema, Administrativo, Mecánico y Comercial

Descripción: Permite crear o modificar personas en el sistema. Se debe crear un registro en la tabla PERSONAS para cada usuario y cada cliente.

Flujo de eventos principal:

3. El usuario indica la opción para agregar una persona nueva y especifica los detalles.
4. El sistema crea la persona nueva en la Base de Datos.

Flujo de eventos alternativo:

3. El usuario indica la opción para modificar una persona existente y especifica los detalles.
4. El sistema crea la persona nueva en la Base de Datos.

Excepciones: Se anula la transacción en los casos siguientes:

- el valor especificado para el sexo de la persona no es correcto
- el valor especificado para el municipio en el que la persona reside no es correcto
- ya existe una persona con el NIF especificado

Caso de Uso 8: Gestionar Coches

Nombre: Gestionar Coches

Actor: Administrativo, Mecánico y Comercial

Descripción: Permite crear o modificar coches en la Base de Datos.

Flujo de eventos principal:

5. El usuario indica la opción para agregar un coche nuevo y especifica los detalles.
6. El sistema crea el coche nuevo en la Base de Datos.

Flujo de eventos alternativo:

5. El usuario indica la opción para modificar un coche existente y especifica los detalles.
6. El sistema modifica el coche en la Base de Datos.

Excepciones: Se anula la transacción en los casos siguientes:

- ya existe un coche para la matrícula especificada
- el valor especificado para el modelo no es adecuado
- el propietario referenciado no existe

3.7.3. Modelo Conceptual de la Base de Datos

El diagrama Entidad-Relación siguiente muestra el modelo conceptual diseñado para la Base de Datos [4]:

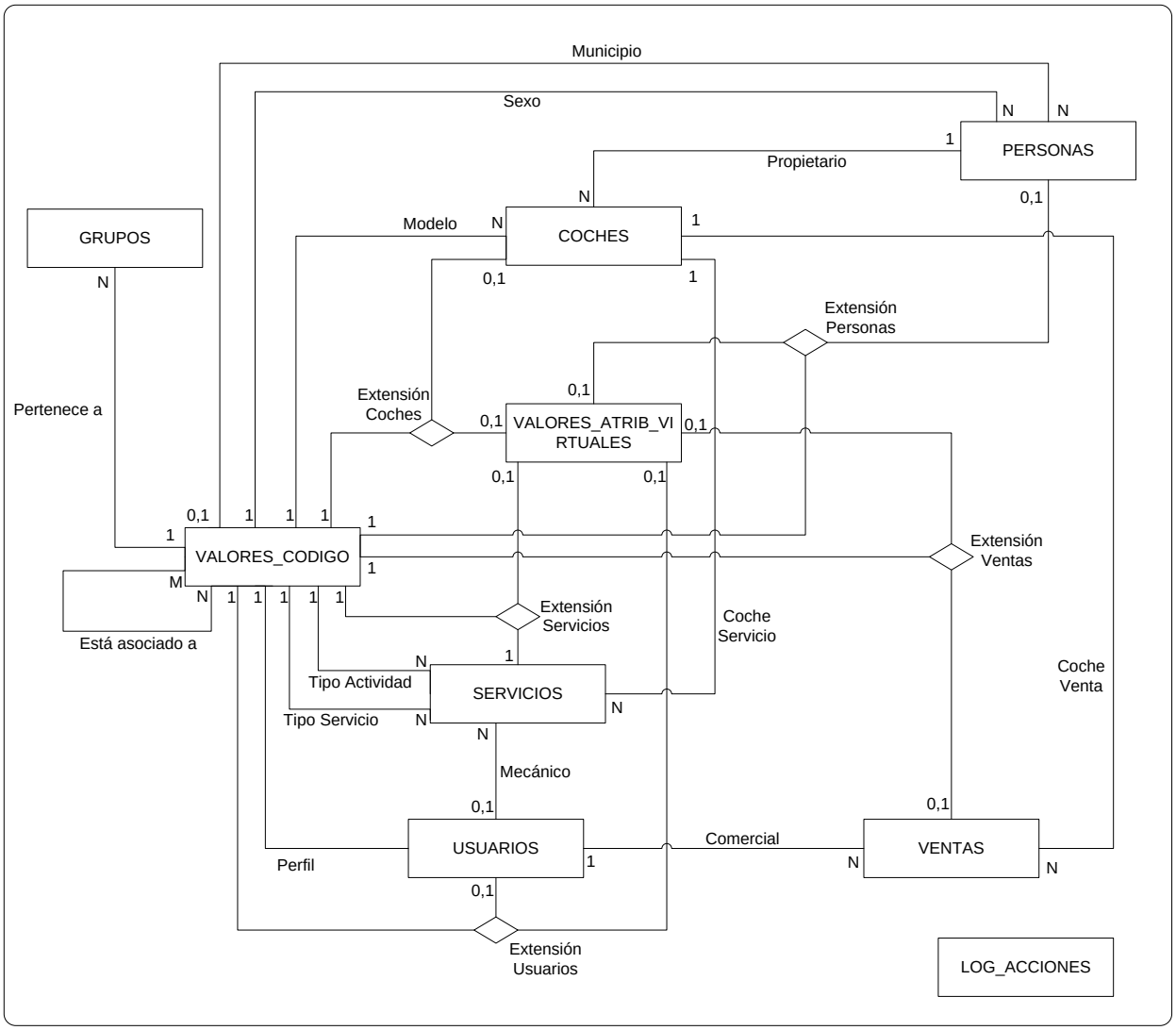


Figura 6. Diagrama Entidad-Relación para el modelo conceptual

Nota importante sobre el diagrama:

- Todas las tablas que aparecen en el diagrama están relacionadas con la tabla USUARIOS con el fin de identificar al usuario responsable de la inserción o última modificación de cada registro. No se han dibujado estas relaciones para simplificar el diagrama.

Descripción de las entidades del modelo conceptual

A continuación se describe las entidades identificadas en el modelo conceptual:

La tabla **VALORES_CODIGO** contiene el conjunto de registros que define la información de referencia de la Base de Datos. En esta tabla se definen los diferentes valores para los *modelos de coche*, los diferentes valores de *extras* que caracterizan los modelos, los diferentes valores para los *municipios*, etc.... Se ha llamado *valores de código* a los registros de esta tabla. Los valores masculino, femenino, verde, rojo, azul, Palma de Mallorca, Barcelona.... son ejemplos de registros de esta tabla, y por tanto son *valores de código*.

La tabla **GRUPOS** contiene las diferentes agrupaciones o categorías en las que se agrupan o clasifican los *valores de código* de la tabla VALORES_CODIGO en base a su naturaleza. Las entidades Municipios, Provincias y Modelos de coches, entre muchos otros, son ejemplos de *Grupos*.

La tabla **VENTAS** contiene las ventas de coches realizadas por los comerciales del concesionario.

La tabla **COCHES** contiene los coches vendidos por el concesionario, los coches sobre los que se ha hecho alguna reparación o revisión y los coches para los que se tiene planificado una revisión.

La tabla **PERSONAS** contiene los propietarios de los coches y también los empleados del concesionario que son usuarios del sistema. Los usuarios son los administrativos, comerciales, mecánicos y técnicos informáticos del concesionario.

La tabla **USUARIOS** contiene los usuarios del sistema, es decir las cuentas de usuarios de los comerciales, mecánicos y técnicos informáticos del concesionario. La tabla USUARIOS extiende la tabla PERSONAS.

La tabla **SERVICIOS** contiene las reparaciones y revisiones realizadas y/o planificadas. Un servicio puede ser tanto una revisión como una reparación. Se distingue las reparaciones de las revisiones por el atributo *Tipo de Servicio*. Si un servicio ha sido planificado si tiene un valor para el atributo *Fecha Planificación*. Si un servicio ha sido realizado si tiene un valor para el atributo *Fecha Servicio*.

La tabla **VALORES_ATRIB_VIRTUALES** contiene los valores de los *atributos virtuales*. Un *atributo virtual* es un mecanismo del sistema para poder simular el añadir un atributo a una tabla sin modificar la estructura original de la tabla. Si por ejemplo, se define el *atributo virtual* Color para tabla Coches, la tabla VALORES_ATRIB_VIRTUALES contiene los colores concretos de cada coche.

La tabla **LOG_ACCIONES** contiene los registros de las acciones de inserción y modificación sobre los registros de las demás tablas. Cada vez que se inserta o se modifica un registro, se registra la acción en la tabla LOG_ACCIONES.

Descripción de los atributos de las entidades del modelo conceptual

A continuación se describe las entidades identificadas en el modelo conceptual:

GRUPOS (valorGrupo, grupo, nombre)

VALORES_CODIGO (ID, grupo, visualización, descripción)

COCHES (ID, Matrícula, Modelo, Propietario)

PERSONAS (ID, NIF, Apellidos, Nombre, Fecha nacimiento, Sexo, Teléfono, Calle y Número, CP, Municipio)

USUARIOS (ID, Persona, Username, Perfil)

SERVICIOS (ID, Tipo Actividad, Coche, Fecha planificada, Fecha servicio, Duración, Mecánico, Tipo servicio)

VENTAS (ID, Fecha venta, Coche, Comercial)

VALORES_ATRIB_VIRTUALES (ID, Tipo atributo, Registro, Valor)

LOG_ACCIONES (ID, Acción, Tabla, Registro, Fecha / Hora)

Descripción de las relaciones del modelo conceptual

A continuación se describe las relaciones mostradas en el modelo conceptual:

La relación **Municipio** permite especificar el municipio en el que reside cada persona. Una persona tiene asociado como mucho un municipio.

La relación **Sexo** define el género de cada persona. Cada persona tiene un y sólo un género.

La relación **Propietario** especifica la persona propietaria de cada coche. No toda persona es propietario de algún coche. Todo coche tiene un y sólo un propietario.

La relación **Modelo** describe el modelo de un coche. Cada coche tiene un y sólo un modelo. Por supuesto, puede haber más de un coche del mismo modelo.

La relación **Pertenece a** clasifica los *valores de código* en *grupos*. Cada *valor de código* es de un y sólo un *grupo*. Un *grupo* puede tener cero o varios *valores de código*.

La relación **Tipo Actividad** identifica el tipo de actividad de un servicio. Cada servicio (es decir, revisión o reparación), tiene un y sólo tipo de actividad. Un tipo de actividad puede estar asociado a cero o más servicios.

La relación **Tipo Servicio** identifica el tipo de servicio de un servicio. Cada servicio es de un y sólo tipo de servicio. Un tipo de servicio puede estar asociado a cero o más servicios.

La relación **Mecánico** identifica el usuario mecánico responsable de la realización de un servicio. Un servicio puede tener ningún usuario mecánico asociado. Esto ocurre si el servicio está planificado y por tanto, pendiente de realizarse. Cada se completa el servicio, pasa a tener como mucho un usuario mecánico asociado. Un usuario mecánico puede ser responsable de cero o varios servicios.

La relación **Comercial** identifica el usuario comercial responsable de una venta. Un usuario comercial puede ser responsable de cero o varias ventas. Cada venta tiene un y sólo un usuario comercial responsable.

La relación **Coche Venta** permite identificar el coche de una venta. Cada coche puede haberse vendido cero o varias veces. Cada venta está relacionada con un y sólo un coche.

La relación **Coche Servicio** identifica el coche sobre el que se ha realizado o planificado un servicio. Cada servicio tiene un y sólo un coche asociado. Un coche puede tener cero o varios servicios asociado.

La relación **Perfil** identifica el perfil (administrativo, mecánico, comercial...) de cada usuario. Cada usuario es un y sólo un perfil. Más de un usuario puede tener el mismo perfil. Puede haber perfiles sin usuarios asociados.

La relación ternaria **Extensión Coches** identifica el valor (tabla VALORES_ATRIB_VIRTUALES) que tiene un *atributo virtual* (tabla VALORES_CODIGO) para un registro coche (tabla COCHES). Cada valor es de un y sólo un coche y de un y sólo un *atributo virtual*.

La relación ternaria **Extensión Servicios** identifica el valor (tabla VALORES_ATRIB_VIRTUALES) que tiene un *atributo virtual* (tabla VALORES_CODIGO) para un registro servicio (tabla SERVICIOS). Cada valor es de un y sólo un servicio y de un y sólo un *atributo virtual*.

La relación ternaria **Extensión Usuarios** identifica el valor (tabla VALORES_ATRIB_VIRTUALES) que tiene un *atributo virtual* (tabla VALORES_CODIGO) para un registro usuario (tabla USUARIOS). Cada valor es de un y sólo un usuario y de un y sólo un *atributo virtual*.

La relación ternaria **Extensión Personas** identifica el valor (tabla VALORES_ATRIB_VIRTUALES) que tiene un *atributo virtual* (tabla VALORES_CODIGO) para un registro persona (tabla PERSONAS). Cada valor es de una y sólo una persona y de un y sólo un *atributo virtual*.

La relación ternaria **Extensión Ventas** identifica el valor (tabla VALORES_ATRIB_VIRTUALES) que tiene un *atributo virtual* (tabla VALORES_CODIGO) para un registro venta (tabla VENTAS). Cada valor es de una y sólo una venta y de un y sólo un *atributo virtual*.

3.7.4. Decisiones de diseño más relevantes

A continuación se describen las decisiones de diseño más relevantes:

Registro de acciones

Para cada registro de todas las tablas se guardará, en el mismo registro, la fecha y hora de la última modificación (o de la inserción del registro) además de la clave extranjera el usuario que realizó la acción. Así, en el mismo registro se puede saber quién realizó la acción más reciente y en qué momento. Se definirán estos campos de la forma siguiente:

- actualiz_f_h: DATE
- actualiz_ce: NUMBER (10,0)

Las exportaciones de datos para el *Data Warehouse* se basarán en el campo *actualiz_f_h*. Conociendo la fecha de la última exportación, este campo permite a los procesos de exportación detectar los registros modificaciones después de la última exportación.

Además, de anotar en cada registro el responsable e instante de la inserción o de la última modificación, se mantiene la tabla LOG_ACCIONES. El sistema crea un registro en esta tabla para cada inserción y modificación en las demás tablas de la Base de Datos. Para cada acción registrada, se almacena, la tabla en la que se realizó la modificación o inserción, el registro afectado, el tipo de acción (inserción/modificación), el usuario responsable y la fecha y hora en que realizó la acción.

Integración con el *Data Warehouse*

La configuración de un *Data Warehouse* no está incluida en el alcance de este proyecto. Sí están incluidos los mecanismos que permiten la integración con un *Data Warehouse.* Se ha decidido proporcionar procedimientos almacenados que exporte el contenido de las tablas a ficheros CSV. Estos ficheros CSV deberán ser importados a *Data Warehouse* que deberá ser utilizado. Estos procedimientos almacenados deberán ser ejecutados con una fecha especificada como parámetro. Se exportará los registros insertados o modificados después de dicha fecha. De esta forma, se puede evitar exportar la misma información más de una vez.

Eliminación de registros

El sistema no ofrecerá mecanismos de eliminación de registros. Un técnico de sistema podrá ejecutar sentencias de eliminación de datos pero el sistema no ofrecerá procedimientos almacenados que realicen eliminaciones. Todas las tablas contarán con el atributo *activo_ind* cuyo valor predeterminado será 1. Si se desea eliminar un registro, se deberá modificar el valor de este campo a 0. A la aplicación a la que la Base de Datos aporta persistencia, deberá reconocer los registros eliminados a partir de este campo. Se considera que este proceso es más sencillo de gestionar que la eliminación real de registros dado que no se producen conflictos de integridad. Además, los registros inactivados seguirían estando disponibles para ser consultados. Una opción alternativa a esta opción es copiar los campos del registro siendo eliminado a tabla histórica.

Campos Auto-numéricos

A excepto de la tabla Grupos, la clave primaria de todas las tablas será auto-numérico. Se implementará los atributos auto-numéricos mediante la definición de secuencias y triggers que asignarán al campo auto-numérico el siguiente valor disponible de la secuencia.

Revisiones y Reparaciones

Se considera que hay poca diferencia entre una reparación y una revisión. Salvo poder distinguir si una actividad es una reparación o una revisión, no hay más diferencias importantes. Existe el requerimiento de poder registrar la planificación de revisiones. No es estrictamente necesario poder planificar reparaciones, aunque no se ha pedido evitar que ello sea posible. Para simplificar, se ha decidido crear la tabla SERVICIO para registrar tanto las reparaciones y las revisiones. Para cada registro, se deberá poder saber si se ha realizado o se únicamente está planificado. Para poder gestionar las reparaciones y revisiones y si fueron o no planificados, se definirá los campos siguientes:

- planif_serv_f_h: fecha para el que se planifica la revisión o reparación.
- servicio_f_h: fecha (y hora) en la que se ha realizado la revisión o reparación.
- tipoServicio_cd: permite saber si el servicio es una revisión o una reparación.
- tipoActividad_cd: describe la reparación o la revisión.

Gestión de los extras

Se pide poder registrar los extras seleccionado por los clientes cuando compran un coche. Dado que los extras disponibles dependen en realidad del modelo del coche y no directamente del coche, se asociarán los extras al modelo del coche en vez de asociarse al coche. Se interpreta que si un cliente compra un coche de un modelo, implica que ha seleccionado los extras del modelo.

Información de referencia: *valores de código* y *grupos*

Se considera la *información de referencia* como la información que no es de actividad. Es decir, es la información referenciada por la información de actividad (capturada durante la actividad de negocio del cliente) para categorizar aspectos de la información de actividad. La información de referencia es por ejemplo, los municipios, las provincias, los tipos de actividad, los modelos de los coches etc... Dado que toda la información de referencia es tratada de forma semejante, se ha decidido crear una única tabla para gestionar toda esta actividad de referencia. Es decir, se registran todos los municipios, provincias, modelos de coches... en la misma tabla: VALORES_CODIGO. Se distingue por ejemplo los municipios de las provincias porque pertenecen a *grupos* diferentes. Esta opción ha permitido rebajar mucho el número de tablas, procedimiento y funciones a desarrollar. Además, permite al cliente definir sus propios grupos y valores para los *grupos* que define sin tener que modificar la estructura de la Base de datos. Por ejemplo, el cliente podría definir el *grupo* Colores y definir los *valores de código* Azul, Verde, Amarillo... para el *grupo* Colores.

Asociación entre valores de código

Surge la necesidad de poder especificar, por ejemplo, los extras que los coches del mismo modelo tienen o los municipios de una provincia. Para gestionar estas asociaciones, se ha definido la tabla ASOCIACIONES_VALORES_CODIGO.

Extensión de tablas: atributos virtuales

Se pide el requerimiento de hacer que la Base de Datos sea escalable para permitir al cliente ir incorporando nuevas necesidades. Se define el concepto de atributo virtual para permitir al cliente extender las tablas existentes del sistema de forma dinámica sin tener que modificar la estructura de la Base de Datos. Los *atributos virtuales* se configuran en la tabla VALORES_CODIGO (grupo 9). En la definición de un *atributo virtual* se especifica, entre otros, la tabla al que se aplica, el tipo de dato que gestiona y el nombre descriptivo del atributo. El tipo de dato puede ser TEXTO o un grupo (conjunto de valores predefinidos) que el cliente puede crear. Los valores que se asignan a estos atributos se guardan en la tabla VALORES_ATRIB_VIRTUALES. Cada registro en la tabla VALORES_ATRIB_VIRTUALES depende de un *atributo virtual* y un registro concreto en la tabla para el que se ha creado el *atributo virtual*.

Tabla USUARIOS: una extensión de la tabla PERSONAS

Los clientes, comerciales, administrativos, mecánicos y técnicos del sistema tienen en común que son personas y como tales tienen atributos comunes como nombre, apellidos, NIF, fecha de nacimiento y dirección. La distinción más importante entre estos grupos de personas es que los clientes no son usuarios del sistema y los demás, sí lo son. Para evitar redundancias, se crea la tabla PERSONAS para todas las personas y una tabla relacionada USUARIOS para las personas usuarias del sistema. De esta forma, se puede ver la tabla USUARIOS como una extensión de la tabla PERSONAS.

3.8. Diseño

3.8.1. Modelo Lógico de la Base de Datos

El diagrama Entidad-Relación correspondiente al diseño lógico es el siguiente [4]:

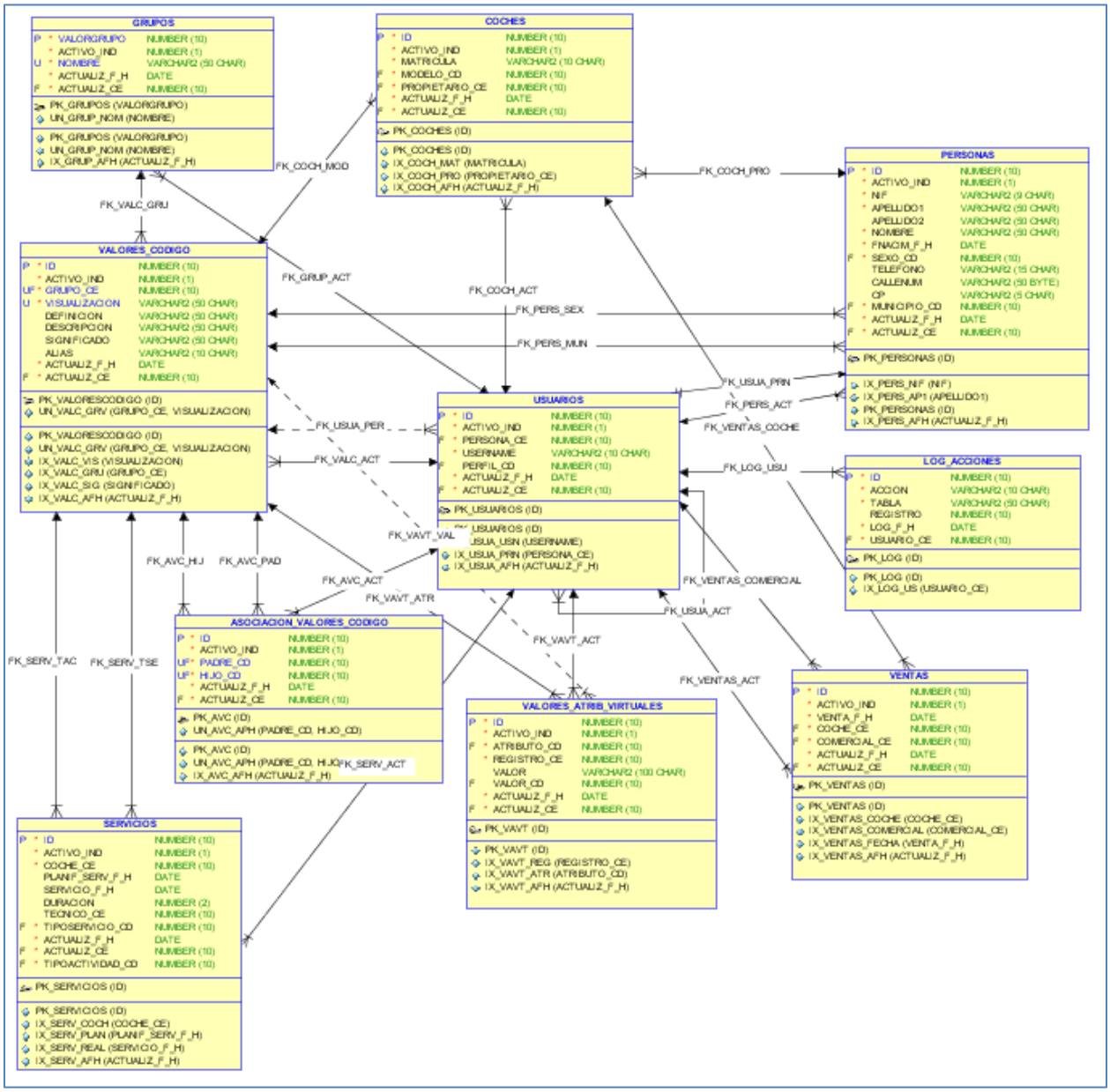


Figura 7. Diagrama ER modelo lógico

El diagrama anterior muestra las tablas, los atributos de las tablas, los tipos de datos de cada atributo, las claves principales, las claves extranjeras, y los índices.

3.8.2. Descripción Lógica de las tablas y demás estructuras

Tabla GRUPOS

La tabla Grupos contiene los Grupos definidos en el sistema. Cada Grupo representa una entidad cuyos valores forman parte del conjunto de información de referencia del sistema. Las entidades siguientes son ejemplos de grupos: Provincias, Municipios y Modelos de Coche. Cada grupo es identificado por un código numérico. El cliente puede agregar todos los grupos adicionales que desea pero como mínimo, debe existir en el sistema los grupos siguientes (se ha de respetar los códigos numéricos):

- Modelo de coche: Grupo 1.
- Sexo: Grupo 2.
- Municipio: Grupo 3.
- Provincia: Grupo 4.
- Tipo de Servicio: Grupo 5.
- Perfil de usuario: Grupo 6.
- Extras: Grupo 7.
- Tipos de Actividad: Grupo 8.
- Atributos virtuales: Grupo 9.

Atributos

valorGrupo: Identificador del Grupo. No es asignado por el sistema.
activo_ind: Debe valer 1 si el registro es activo y 0 en caso contrario.
nombre: Nombre descriptivo del Grupo.
actualiz_f_h: Fecha/hora en la que el registro fue modificado por última vez.
actualiz_ce: Usuario que provocó la última actualización en el registro.

Restricciones y Tipos de Datos

	Tipo de Dato	Nulo	Duplic.	
valorGrupo	NUMBER(10,0)	No	No	Clave Primaria. Indexado.
activo_ind	NUMBER(1,0)	No	Sí	Valor predeterminado: 1.
nombre	VARCHAR2(50 CHAR)	No	No	Indexado.
actualiz_f_h	DATE	No	Sí	Valor predeterminado: *sysdate*. Indexado.
actualiz_ce	NUMBER(10,0)	No	Sí	Clave Extranjera a Usuarios (ID)

Procedimientos

AGREGAR_GRUPO (pValorGrupo, pNombre, pUsuario_ce)
Inserta un registro nuevo en la tabla GRUPOS.

MODIFICAR_GRUPO (pValorGrupo, pActivo_ind, pNombre, pUsuario_ce)
Modifica el registro pValorGrupo en la tabla GRUPOS. Actualiza las columnas *activo_ind, nombre, actualiz_f_h* y *actualiz_ce*. No permite modificar el valor de la columna *valorGrupo*.

OBTENER_GRUPO_POR_VALORGRUPO (pValorGrupo, oRC)
Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente:

*SELECT * FROM GRUPOS WHERE valorGrupo = pValorGrupo;*

DESCARGAR_GRUPOS_POR_ACTFH (pActualiz_f_h)
Realiza la exportación a un fichero CSV los registros de la tabla GRUPOS que hayan sido insertados o modificados después de la fecha/hora especificado por el parámetro *pActualiz_f_h*.

Triggers

AGRMOD_GRUPO
Se ejecuta antes de una operación de agregación o modificación de un registro en la tabla GRUPOS. Este *trigger* realiza las acciones siguientes:

- lanza una excepción si el valor *pValorGrupo* es inferior a 10
- inserta un registro en la tabla LOG_ACCIONES según se haya insertado un registro nuevo o modificado un registro existente

Tabla VALORES_CODIGO

La tabla VALORES_CODIGO contiene los registros que identifican los diferentes *valores de código* de los Grupos. Esta tabla contiene los registros correspondientes a los diferentes Municipios, Sexos... etc.

Atributos

ID: Identificador secuencial asignado por el sistema.
activo_ind: Debe valer 1 si el registro es activo y 0 en caso contrario.
grupo_ce: Identifica el Grupo al que pertenece el *valor de código*.
visualizacion: Valor con el que se debe mostrar el *valor de código* al usuario final.
definicion: Campo descriptivo. Su uso es arbitrario y puede depender el Grupo.
descripcion: Campo descriptivo. Su uso es arbitrario y puede depender el Grupo.
significado: Campo que identifica el *valor de código* en la aplicación.
alias: Código alternativo para identificar el *valor de código.*
actualiz_f_h: Fecha/hora en la que el registro fue modificado por última vez.
actualiz_ce: Usuario que provocó la última actualización en el registro.

Restricciones y Tipos de Datos

	Tipo de Dato	Nulo	Duplic.	
ID	NUMBER(10,0)	No	No	Clave Primaria. Indexado.
activo_ind	NUMBER(1,0)	No	Sí	Valor predeterminado: 1.
grupo_ce	NUMBER(10,0)	No	Sí	Clave Extranjera a Grupos (valorGrupo). Indexado.
visualizacion	VARCHAR2(50 CHAR)	No	Sí	Indexado.
definicion	VARCHAR2(50 CHAR)	Sí	Sí	Indexado.
descripcion	VARCHAR2(50 CHAR)	Sí	Sí	
significado	VARCHAR2(50 CHAR)	Sí	Sí	Indexado.
alias	VARCHAR2(50 CHAR)	Sí	Sí	
actualiz_f_h	DATE	No	Sí	Valor predeter.: *sysdate*. Indexado.
actualiz_ce	NUMBER(10,0)	No	Sí	Clave Extranjera a Usuarios (ID)

Restricciones adicionales:
No se permiten duplicados para el conjunto (grupo_ce, visualización).

Procedimientos

AGREGAR_VALOR_CODIGO (pGrupo_ce, pVisualizacion, pDef, pDesc, pSignif, pAlias, pUsuario_ce)
Inserta un registro nuevo en la tabla VALORES_CODIGO.

MODIFICAR_VALOR_CODIGO (pID, pActivo_ind, pVisualizacion, pDef, pDesc, pSignif, pAlias, pUsuario_ce)

Modifica el registro pID en la tabla VALORES_CODIGO. Actualiza las columnas *activo_ind, visualizacion, definicion, descripcion, significado, alias, actualiz_f_h* y *actualiz_ce*. No permite modificar el valor de las columnas ID ni grupo_ce.

OBTENER_VALOR_CODIGO_POR_ID (pId, oRC)
Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente:

*SELECT * FROM VALORES_CODIGO WHERE ID = pID;*

BUSCAR_VALORES_CODIGO_POR_GR (pGrupo_ce, oRC)
Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente:

*SELECT * FROM VALORES_CODIGO WHERE grupo_ce = pGrupo_ce;*

DESCARGAR_VALCOD_POR_ACTFH (pActualiz_f_h)
Realiza la exportación a un fichero CSV los registros de la tabla VALORES_CODIGO que hayan sido insertados o modificados después de la fecha/hora especificado por el parámetro *pActualiz_f_h*.

Triggers

Trigger AGRMOD_VALOR_CODIGO
Se ejecuta antes de una operación de agregación o modificación de un registro en la tabla VALORES_CODIGO. Este *trigger* realiza las acciones siguientes:

- al insertarse un registro nuevo, asigna un valor secuencial al atributo ID.
- inserta un registro en la tabla LOG_ACCIONES según se haya insertado un registro nuevo o modificado un registro existente.

Funciones

CONSULTAR_VISUALIZACION_POR_ID (pId, pGrupo_ce) **RETURN VARCHAR**
Esta función devuelve el valor de visualización de un *valor de código* buscando por el ID del *valor de código.*

CONSULTAR_ID_POR_DEFINICION (pDef, pGrupo_ce) **RETURN NUMBER**
Esta función devuelve el ID de un *valor de código* buscando por el valor *definición* del *valor de código.*

CONSULTAR_ID_POR_SIGNIFICADO (pSig, pGrupo_ce) **RETURN NUMBER**
Esta función devuelve el ID de un *valor de código* buscando por el valor *significado* del *valor de código.*

CONSULTAR_ID_POR_VISUALIZACION (pGrupo_ce, pVis) **RETURN NUMBER**
Esta función devuelve el ID de un *valor de código* buscando por el valor de *visualización* del *valor de código.*

Tabla PERSONAS

La tabla PERSONAS contiene los registros que identifican y describen los Propietarios de los coches (es decir, los clientes) y Empleados que trabajan en el concesionario que sean usuarios del sistema. Para cada empleado usuario del sistema, también deberá existir un registro en la tabla Usuarios.

Atributos

ID: identificador secuencial asignado por el sistema.
activo_ind: TRUE si el registro es activo. FALSE en caso contrario.
nif: NIF de la persona.
apellido1: Primer Apellido de la persona.
apellido2: Segundo Apellido de la persona.
nombre: Nombre de la persona.
fechaNacim_f_h: Fecha de nacimiento de la persona.
sexo_cd: Valor de código que identifica el sexo de la persona (Grupo 2).
telefono: Teléfono de contacto de la persona.
calleNum: Calle y número en la que vive la persona.
cp: Código postal de la calle en la que vive la persona.
municipio_cd: Valor de código que identifica el municipio de la persona (Grupo 3).
actualiz_f_h: Fecha/hora en la que el registro fue modificado por última vez.
actualiz_ce: Usuario que provocó la última actualización en el registro.

Restricciones y Tipos de Datos

	Tipo de Dato	Nulo	Duplic.	
ID	NUMBER(10,0)	No	No	Clave Primaria. Indexado.
activo_ind	NUMBER(1,0)	No	Sí	Valor predeterminado: 1.
nif	VARCHAR2(9 CHAR)	No	No	Indexado.
apellido1	VARCHAR2(50 CHAR)	No	Sí	Indexado.
apellido2	VARCHAR2(50 CHAR)	Sí	Sí	
nombre	VARCHAR2(50 CHAR)	No	Sí	
fnacim_f_h	VARCHAR2(50 CHAR)	No	Sí	
sexo_cd	NUMBER(10,0)	Sí	Sí	
telefono	VARCHAR2(15 CHAR)	Sí	Sí	
calleNum	VARCHAR2(50 CHAR)	Sí	Sí	
cp	VARCHAR2(5 CHAR)	Sí	Sí	
municipio_cd	NUMBER(10,0)	Sí	Sí	
actualiz_f_h	DATE	No	Sí	Valor predeter.: *sysdate*. Indexado.
actualiz_ce	NUMBER(10,0)	No	Sí	Clave Extranjera a Usuarios (ID)

Procedimientos

AGREGAR_PERSONA (pNIF, pApellido1, pApellido2, pNombre, pFnacim, pSexo_cd, pTel, pCalleNum, pCP, pMunicipio_cd, pUsuario_ce)
Inserta un registro nuevo en la tala PERSONAS.

MODIFICAR_PERSONA (pID, pActivo_ind, pNIF, pApellido1, pApellido2, pNombre, pFechaNacim, pSexo_cd, pTel, pCalleNum, pCP, pMunicipio_cd, pUsuario_ce)
Modifica el registro pID en la tabla PERSONAS. Actualiza las columnas *activo_ind, nif, apellido1, apellido2, nombre, fnacim_f_h, sexo_cd, telefono, calleNum, cp, municipio_cd actualiz_f_h* y *actualiz_ce*. No permite modificar el valor de la columna ID.

OBTENER_PERSONA_POR_ID (pID, oRC)
Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente:
*SELECT * FROM PERSONAS WHERE ID = pID;*

OBTENER_PERSONA_POR_NIF (pNIF, oRC)
Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente:
*SELECT * FROM PERSONAS WHERE nif = pNIF;*

BUSCAR_PERSONA_POR_APELLIDO1 (pCadena, oRC)
Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente:
*SELECT * FROM PERSONAS WHERE apellido1 like '*pCadena*';*

DESCARGAR_PERSONAS_POR_ACTFH (pActualiz_f_h)
Realiza la exportación a un fichero CSV los registros de la tabla PERSONAS que hayan sido insertados o modificados después de la fecha/hora especificado por el parámetro *pActualiz_f_h*.

Triggers

AGRMOD_PERSONA
Se ejecuta antes de una operación de agregación o modificación de un registro en la tabla PERSONAS. Este *trigger* realiza las acciones siguientes:

- al insertarse un registro nuevo, asigna un valor secuencial al atributo ID.
- inserta un registro en la tabla LOG_ACCIONES según se haya insertado un registro nuevo o modificado un registro existente.
- lanza una excepción si el *valor de código* especificado para el sexo (pSexo_cd) no es del grupo 2.
- lanza una excepción si el *valor de código* especificado para el sexo (pMunicipio_cd) no es del grupo 3.

Tabla COCHES

La tabla coches contiene todos los coches vendidos por el concesionario. También contiene los coches a los que el concesionario ha realizado alguna reparación o revisión.

Atributos

ID: identificador secuencial asignado por el sistema.
activo_ind: TRUE si el registro es activo. FALSE en caso contrario.
matricula: Matrícula del coche.
modelo_cd: Valor de código que identifica el modelo del coche (Grupo 1).
propietario_ce: Identificador de la persona propietaria del coche.
actualiz_f_h: Fecha/hora en la que el registro fue modificado por última vez.
actualiz_ce: Usuario que provocó la última actualización en el registro.

Restricciones y Tipos de Datos

	Tipo de Dato	Nulo	Duplic.	
ID	NUMBER(10,0)	No	No	Clave Primaria. Indexado.
activo_ind	NUMBER(1,0)	No	Sí	Valor predeterminado: 1.
matricula	VARCHAR2(10 CHAR)	No	No	Indexado.
modelo_cd	NUMBER(10,0)	No	Sí	Clave Extranjera a VALORES_CODIGO (ID)
propietario_ce	NUMBER(10,0)	No	Sí	Clave Extranjera a PERSONAS (ID). Indexado.
actualiz_f_h	DATE	No	Sí	Valor predeter.: *sysdate*. Indexado.
actualiz_ce	NUMBER(10,0)	No	Sí	Clave Extranjera a Usuarios (ID)

Procedimientos

AGREGAR_COCHE (pMatrícula, pModelo_cd, pPropietario_ce, pUsuario_ce)
Inserta un registro nuevo en la tabla COCHES.

MODIFICAR_COCHE (pId, activo_ind, pMatricula, pModelo_cd, pPropietario_ce, pUsuario_ce)
Modifica el registro pID en la tabla COCHES. Actualiza las columnas *activo_ind, matricula, modelo_cd, pPropietario_ce, actualiz_f_h* y *actualiz_ce*. No permite modificar el valor de la columna ID.

OBTENER_COCHE_POR_ID (pId, oRC)
Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente: *SELECT * FROM COCHES WHERE ID = pID;*

OBTENER_COCHE_POR_MATRICULA (pMatricula, oRC)

Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente: *SELECT * FROM COCHES WHERE matricula = pMatricula;*

BUSCAR_COCHES_POR_PROPIETARIO (pPropietario_ce, oRC)
Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente: *SELECT * FROM COCHES WHERE propietario_ce = pPropietario_ce;*

DESCARGAR_COCHES_POR_ACTFH (pActualiz_f_h)
Realiza la exportación a un fichero CSV los registros de la tabla COCHES que hayan sido insertados o modificados después de la fecha/hora especificado por el parámetro *pActualiz_f_h*.

Triggers

AGRMOD_COCHE

Se ejecuta antes de una operación de agregación o modificación de un registro en la tabla COCHES. Este *trigger* realiza las acciones siguientes:

- al insertarse un registro nuevo, asigna un valor secuencial al atributo ID.
- inserta un registro en la tabla LOG_ACCIONES según se haya insertado un registro nuevo o modificado un registro existente.
- lanza una excepción si el *valor de código* especificado para el modelo (*pModelo_cd*) no es del grupo 1.

Tabla SERVICIOS

La tabla Servicios contiene los registros que identifican y describen las reparaciones y revisiones que el taller ha realizado sobre los coches (o que se tiene planificado).

Atributos

ID: identificador secuencial asignado por el sistema.
activo_ind: TRUE si el registro es activo. FALSE en caso contrario.
tipoActividad_cd: valor de código que identifica el tipo de actividad (Grupo 8).
coche_ce: identificador del coche (ID en tabla Coches).
planif_serv_f_h: fecha/hora para el que se planifica el servicio.
servicio_f_h: fecha y hora en que comienza el servicio.
duracion: la duración en horas que ha durado el servicio.
tecnico_ce: identificador del técnico realizador del servicio (ID en tabla Usuarios).
tipoServicio_cd: valor de código que identifica el tipo de servicios (Grupo 5).
actualiz_f_h: fecha/hora en la que el registro fue modificado por última vez.
actualiz_ce: Usuario que provocó la última actualización en el registro.

Restricciones y Tipos de Datos

	Tipo de Dato	Nulo	Duplic.	
ID	NUMBER(10,0)	No	No	Clave Primaria. Indexado.
activo_ind	NUMBER(1,0)	No	Sí	Valor predeterminado: 1.
tipoActividad_cd	NUMBER(10,0)	No	Sí	Clave Extranjera a VALORES_CODIGO (ID)
coche_ce	NUMBER(10,0)	No	Sí	Indexado.
planif_serv_f_h	DATE	Sí	Sí	Indexado.
servicio_f_h	DATE	Sí	Sí	Indexado.
duracion	NUMBER(2,0)	Sí	Sí	
tecnico_ce	NUMBER(10,0)	Sí	Sí	
tipoServicio_cd	NUMBER(10,0)	No	Sí	Clave Extranjera a VALORES_CODIGO (ID)
actualiz_f_h	DATE	No	Sí	Valor predeterminado: *sysdate*. Indexado.
actualiz_ce	NUMBER(10,0)	No	Sí	Clave Extranjera a Usuarios (ID)

Procedimientos

AGREGAR_SERVICIO (pActividad_cd, pCoche_ce, pFHPlan, pFHInicio, pDuracion, pTecnico_ce, pTipoServicio_cd, pUsuario_ce)
Inserta un registro nuevo en la tabla SERVICIOS.

MODIFICAR_SERVICIO (pID, pActividad_cd, pCoche_ce, pFHPlan, pFHInicio, pDuracion, pTecnico_ce, pTipoServicio_cd, pActivo_ind, pUsuario_ce)

Modifica el registro pID en la tabla SERVICIOS. Actualiza las columnas *activo_ind, tipoActividad_cd, coche_ce, planif_serv_f_h, servicio_f_h, duración, técnico_ce, tipoServicio_cd actualiz_f_h* y *actualiz_ce*. No permite modificar el valor de la columna ID.

OBTENER_SERVICIO_POR_ID (pID, oRC)
Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente:

*SELECT * FROM SERVICIOS WHERE ID = pID;*

BUSCAR_SERVICIOS_POR_RFPLANIF (pFH_Inicio, pFH_Fin, oRC)
Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente:

*SELECT * FROM SERVICIOS WHERE planif_serv_f_h between pFH_Inicio and pFH_Fin;*

BUSCAR_SERVICIOS_POR_RFSERV (pFH_Inicio, pFH_Fin, oRC)
Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente:

*SELECT * FROM SERVICIOS WHERE servicio_f_h between pFH_Inicio and pFH_Fin;*

BUSCAR_SERV_POR_RFSERV_Y_TEC (pFH_Inicio, pFH_Fin, pTecnico_ce, oRC)
Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente:

*SELECT * FROM SERVICIOS WHERE tecnico_ce = pTecnico_ce AND servicio_f_h between pFH_Inicio and pFH_Fin;*

BUSCAR_SERV_POR_RFSERV_Y_COCHE (pFH_Inicio, pFH_Fin, pCoche_ce, oRC)
Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente:

*SELECT * FROM SERVICIOS WHERE coche_ce = pCoche_ce AND servicio_f_h between pFH_Inicio and pFH_Fin;*

DESCARGAR_SERVICIOS_POR_ACTFH (pActualiz_f_h)
Realiza la exportación a un fichero CSV los registros de la tabla SERVICIOS que hayan sido insertados o modificados después de la fecha/hora especificado por el parámetro *pActualiz_f_h*.

Triggers

AGRMOD_SERVICIO

Se ejecuta antes de una operación de agregación o modificación de un registro en la tabla SERVICIOS. Este *trigger* realiza las acciones siguientes:

- al insertarse un registro nuevo, asigna un valor secuencial al atributo ID.
- inserta un registro en la tabla LOG_ACCIONES según se haya insertado un registro nuevo o modificado un registro existente.
- lanza una excepción si el *valor de código* especificado para el tipo de actividad (*pTipoActividad_cd*) no es del grupo 8.

- lanza una excepción si el *valor de código* especificado para el tipo de servicio (*pTipoServicio_cd*) no es del grupo 5.
- lanza una excepción si tanto *pServicio_f_h* como *pPlanif_serv_f_h* son nulos.
- lanza una excepción si *pServicio_f_h* no es nulo siendo nulo alguno de los parámetros *pDuración* y/o *pTécnico_ce.*

Tabla ASOCIACION_VALORES_CODIGO

Esta tabla contiene la asociación establecida entre *valores de código*. Se puede asociar por ejemplo, *valores de código* del *grupo* Modelos con *valores de código* del *grupo* Extras para describir qué extras tiene cada modelo de coche.

Atributos

ID: identificador secuencial asignado por el sistema.
activo_ind: TRUE si el registro es activo. FALSE en caso contrario.
padre_cd: *valor de código* principal (al que se le relaciona hijo_cd).
hijo_cd: *valor de código* relacionado.
actualiz_f_h: Fecha/hora en la que el registro fue modificado por última vez.
actualiz_ce: Usuario que provocó la última actualización en el registro.

Restricciones y Tipos de Datos

	Tipo de Dato	Nulo	Duplic.	
ID	NUMBER(10,0)	No	No	Clave Primaria. Indexado.
activo_ind	NUMBER(1,0)	No	Sí	Valor predeterminado: 1. Rango de valores permitido:{0,1}
padre_cd	NUMBER(10,0)	No	Sí	Clave Extranjera a VALORES_CODIGO (ID). Indexado
hijo_cd	NUMBER(10,0)	No	Sí	Clave Extranjera a VALORES_CODIGO (ID). Indexado
actualiz_f_h	DATE	No	Sí	Valor predeterminado: *sysdate*. Indexado.
actualiz_ce	NUMBER(10,0)	No	Sí	Clave Extranjera a Usuarios (ID)

Procedimientos

AGREGAR_ASOC_VAL_CODIGO (pPadre_cd, pHijo_cd, pUsuario_ce)
Inserta un registro nuevo en la tabla ASOCIACION_VALORES_CODIGO.

MODIFICAR_ASOC_VAL_CODIGO (pID, pPadre_cd, pHijo_cd, pActivo_ind, pUsuario_ce)
Modifica el registro pID en la tabla ASOCIACION_VALORES_CODIGO. Actualiza las columnas *activo_ind, pPadre_cd, pHijo_cd, actualiz_f_h* y *actualiz_ce*. No permite modificar el valor de la columna ID.

OBTENER_ASOC_VAL_COD_POR_ID (pID, oRC)
Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente:

*SELECT * FROM ASOCIACION_VALORES_CODIGO WHERE ID = pID;*

BUSCAR_ASOC_VAL_COD_POR_PADRE (pPadre_cd, oRC)
Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente:

*SELECT * FROM ASOCIACION_VALORES_CODIGO WHERE padre_cd = pPadre_cd;*

BUSCAR_ASOC_VAL_COD_POR_GRPADRE (pGRPadre, oRC)

Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente:

```
SELECT
  ASOCIACION_VALORES_CODIGO.ID,
  ASOCIACION_VALORES_CODIGO.activo_ind,
  ASOCIACION_VALORES_CODIGO.padre_cd,
  ASOCIACION_VALORES_CODIGO.hijo_cd,
  ASOCIACION_VALORES_CODIGO.actualiz_f_h,
  ASOCIACION_VALORES_CODIGO.actualiz_ce
FROM
  ASOCIACION_VALORES_CODIGO,
  VALORES_CODIGO
WHERE VALORES_CODIGO.ID = ASOCIACION_VALORES_CODIGO.padre_cd
AND VALORES_CODIGO.grupo_ce = pGrupo_ce;
```

DESCARGAR_SERVICIOS_POR_ACTFH (pActualiz_f_h)

Realiza la exportación a un fichero CSV los registros de la tabla ASOCIACION_VALORES_CODIGO que hayan sido insertados o modificados después de la fecha/hora especificado por el parámetro *pActualiz_f_h*.

Triggers

AGRMOD_ASOC_VALORES_CODIGO

Se ejecuta antes de una operación de agregación o modificación de un registro en la tabla ASOCIACION_VALORES_CODIGO. Este *trigger* realiza las acciones siguientes:

- al insertarse un registro nuevo, asigna un valor secuencial al atributo ID.
- inserta un registro en la tabla LOG_ACCIONES según se haya insertado un registro nuevo o modificado un registro existente.

Tabla USUARIOS

La tabla Usuarios contiene los registros que identifican y describen los usuarios del sistema. Cada registro usuario debe estar asociado a uno y solo un registro de la tabla Persona.

Atributos

ID: identificador secuencial asignado por el sistema.
activo_ind: TRUE si el registro es activo. FALSE en caso contrario.
persona_ce: identifica el registro relacionado en la tabla Persona.
userName: nombre de usuario para la autenticación del usuario en el sistema.
perfil_cd: valor de código que identifica el perfil de seguridad (Grupo 6).
actualiz_f_h: Fecha/hora en la que el registro fue modificado por última vez.
actualiz_ce: Usuario que provocó la última actualización en el registro.

Restricciones y Tipos de Datos

	Tipo de Dato	Nulo	Duplic.	
ID	NUMBER(10,0)	No	No	Clave Primaria. Indexado.
activo_ind	NUMBER(1,0)	No	Sí	Valor predeterminado: 1. Rango de valores permitido:{0,1}
persona_ce	NUMBER(10,0)	No	No	Indexado.
userName	VARCHAR2(10 CHAR)	No	No	Indexado.
perfil_cd	NUMBER(10,0)	No	Sí	
actualiz_f_h	DATE	No	Sí	Valor predeterminado: *sysdate*. Indexado.
actualiz_ce	NUMBER(10,0)	No	Sí	Clave Extranjera a Usuarios (ID)

Procedimientos

AGREGAR_USUARIO (pPersona_ce, pUsername, pPerfil_cd)
Inserta un registro nuevo en la tabla USUARIOS.

MODIFICAR_USUARIO (pID, pActivo_ind, pPersona_ce, pUsername, pPerfil_cd, pUsuario_ce)
Modifica el registro pID en la tabla USUARIOS. Actualiza las columnas *activo_ind, persona_ce, userName, perfil_cd, actualiz_f_h* y *actualiz_ce*. No permite modificar el valor de la columna ID.

OBTENER_USUARIO_POR_ID (pID, oRC)
Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente:

*SELECT * FROM USUARIOS WHERE ID = pID;*

OBTENER_USUARIO_POR_USERNAME (pUsername, oRC)
Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente:

*SELECT * FROM USUARIOS WHERE userName = pUserName;*

DESCARGAR_USUARIOS_POR_ACTFH (pActualiz_f_h)
Realiza la exportación a un fichero CSV los registros de la tabla USUARIOS que hayan sido insertados o modificados después de la fecha/hora especificado por el parámetro *pActualiz_f_h*.

Triggers

AGRMOD_USUARIO

Se ejecuta antes de una operación de agregación o modificación de un registro en la tabla SERVICIOS. Este *trigger* realiza las acciones siguientes:

- al insertarse un registro nuevo, asigna un valor secuencial al atributo ID.
- inserta un registro en la tabla LOG_ACCIONES según se haya insertado un registro nuevo o modificado un registro existente.
- lanza una excepción si el *valor de código* especificado para el perfil (*pPerfil_cd*) no es del grupo 6.

Tabla VENTAS

La tabla Ventas contiene los registros que identifican y describen las ventas realizadas por los comerciales.

Atributos

ID: identificador secuencial asignado por el sistema.
activo_ind: TRUE si el registro es activo. FALSE en caso contrario.
venta_f_h: fecha y hora en que se ha vendido el coche.
coche_ce: identificador del coche de la venta (ID en la tabla Coche).
comercial_ce: identificador del comercial (ID en la tabla Usuarios).
actualiz_f_h: Fecha/hora en la que el registro fue modificado por última vez.
actualiz_ce: Usuario que provocó la última actualización en el registro.

Restricciones y Tipos de Datos

	Tipo de Dato	Nulo	Duplic.	
ID	NUMBER(10,0)	No	No	Clave Primaria. Indexado.
activo_ind	NUMBER(1,0)	No	Sí	Valor predeterminado: 1. Rango de valores permitido:{0,1}
venta_f_h	DATE	No	Sí	Indexado.
coche_ce	NUMBER(10,0)	No	Sí	Indexado. Clave Extranjera a Coches (ID)
comercial_ce	NUMBER(10,0)	No	Sí	Indexado. Clave Extranjera a Usuarios (ID)
actualiz_f_h	DATE	No	Sí	Valor predeterminado: *sysdate*. Indexado.
actualiz_ce	NUMBER(10,0)	No	Sí	Clave Extranjera a Usuarios (ID)

Procedimientos

AGREGAR_VENTA (pVenta_f_h, pCoche_ce, pComercial_ce, pUsuario_ce)
Inserta un registro nuevo en la tabla VENTAS.

MODIFICAR_VENTA (pID, pActivo_ind, pVenta_f_h, pCoche_ce, pComercial_ce, pUsuario_ce)
Modifica el registro pID en la tabla VENTAS. Actualiza las columnas *activo_ind, venta_f_h, coche_ce, comercial_ce, actualiz_f_h* y *actualiz_ce*. No permite modificar el valor de la columna ID.

OBTENER_VENTA_POR_ID (pID, oRC)
Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente:

*SELECT * FROM VENTAS WHERE ID = pID;*

BUSCAR_VENTAS_POR_COM_Y_RFVENT (pFH_Inicio, pFH_Fin, pComercial_ce, oRC)
Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente:

*SELECT * FROM VENTAS WHERE comercial_ce = pComercial_ce AND venta_f_h BETWEEN pFH_Inicio AND pFH_Fin;*

BUSCAR_VENTAS_POR_RFVENTAS (pFH_Inicio, pFH_Fin, oRC)
Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente:

*SELECT * FROM VENTAS WHERE venta_f_h BETWEEN pFH_Inicio AND pFH_Fin;*

Triggers

AGRMOD_VENTA

Se ejecuta antes de una operación de agregación o modificación de un registro en la tabla VENTAS. Este *trigger* realiza las acciones siguientes:

- al insertarse un registro nuevo, asigna un valor secuencial al atributo ID.
- inserta un registro en la tabla LOG_ACCIONES según se haya insertado un registro nuevo o modificado un registro existente.

Tabla VALORES_ATRIB_VIRTUALES

Esta tabla contiene los valores asignados a los *atributos virtuales* agregados a las tablas principales del sistema. Si por ejemplo, se agrega el *atributo virtual* Color a la tabla COCHES, el color que se especifica para un coche concreto es guardado en esta tabla.

Atributos

ID: identificador secuencial asignado por el sistema.
activo_ind: TRUE si el registro es activo. FALSE en caso contrario.
atributo_cd: *valor de código* que identifica el atributo virtual (Grupo 9).
registro_ce: identificador del registro al que el atributo virtual complementa.
valor: valor del atributo virtual.
valor_cd: valor del atributo virtual si se trata de un *valor de código*.
actualiz_f_h: Fecha/hora en la que el registro fue modificado por última vez.
actualiz_ce: Usuario que provocó la última actualización en el registro.

Restricciones y Tipos de Datos

	Tipo de Dato	Nulo	Duplic.	
ID	NUMBER(10,0)	No	No	Clave Primaria. Indexado.
activo_ind	NUMBER(1,0)	No	Sí	Valor predeterminado: 1. Rango de valores permitido:{0,1}
atributo_cd	NUMBER(10,0)	No	Sí	Clave Extranjera a VALORES_CODIGO (ID). Indexado
registro_ce	NUMBER(10,0)	No	Sí	Indexado
valor	VARCHAR2(50 CHAR)	Sí	Sí	
valor_cd	NUMBER(10,0)	Sí	Sí	
actualiz_f_h	DATE	No	Sí	Valor predeterminado: *sysdate*. Indexado.
actualiz_ce	NUMBER(10,0)	No	Sí	Clave Extranjera a Usuarios (ID)

Procedimientos

AGREGAR_VAL_ATRIB_VIRT (pAtributo_cd, pRegistro_ce, pValor, pValor_cd, pUsuario_ce)
Inserta un registro nuevo en la tabla VALORES_ATRIB_VIRTUALES.

MODIFICAR_VAL_ATRIB_VIRT (pID, pActivo_ind, pAtributo_cd, pRegistro_ce, pValor, pValor_cd, pUsuario_ce)
Modifica el registro pID en la tabla VALORES_ATRIB_VIRTUALES. Actualiza las columnas *activo_ind, atributo_cd, registro_ce, valor, valor_cd, actualiz_f_h* y *actualiz_ce*. No permite modificar el valor de la columna ID.

OBTENER_VAL_ATRIB_VIRT_POR_ID (pID, oRC)

Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente:

*SELECT * FROM VALORES_ATRIB_VIRTUALES WHERE ID = pID;*

BUSCAR_VALATRVIRT_POR_ATR_Y_REG (pAtributo_cd, pRegistro_ce)

Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente:

*SELECT * FROM VALORES_ATRIB_VIRTUALES WHERE atributo_cd = pAtributo_cd AND registro_ce = pRegistro_ce;*

BUSCAR_VALATRVIRT_POR_ATR_Y_VAL (pAtributo_cd, pTexto)

Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente:

*SELECT * FROM VALORES_ATRIB_VIRTUALES WHERE atributo_cd = pAtributo_cd AND valor like '%' || pTexto || '%';*

BUSCAR_VALATRVIRT_POR_ATR_Y_VCD (pAtributo_cd, pValor_cd)

Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente:

*SELECT * FROM VALORES_ATRIB_VIRTUALES WHERE atributo_cd = pAtributo_cd AND valor_cd = pValor_cd;*

Triggers

AGRMOD_VALORES_ATRIB_VIRTUALES

Se ejecuta antes de una operación de agregación o modificación de un registro en la tabla VALORES_ATRIB_VIRTUALES. Este *trigger* realiza las acciones siguientes:

- al insertarse un registro nuevo, asigna un valor secuencial al atributo ID.
- inserta un registro en la tabla LOG_ACCIONES según se haya insertado un registro nuevo o modificado un registro existente.
- lanza una excepción si el *valor de código* especificado para el atributo (*pAtributo_cd*) es nulo.
- lanza una excepción si el *valor de código* especificado para el atributo (*pAtributo_cd*) no es del grupo 9.
- lanza una excepción si el valor especificado para el campo registro_ce es nulo.
- lanza una excepción si no existe el registro *registro_ce* en la tabla al que se aplica el atributo.
- lanza una excepción si campo *valor* es nulo y el atributo *pAtributo_cd* es de tipo TEXTO.
- lanza una excepción si campo *valor_cd* es nulo y el atributo *pAtributo_cd* es de tipo GRUPO.
- lanza una excepción si campo *valor_cd* no pertenece al grupo al que hace referencia el atributo *pAtributo_cd.*

Tabla LOG_ACCIONES

Esta tabla contiene los registros de las acciones de inserción y modificación sobre los registros de las demás tablas. Cada vez que se inserta o se modifica un registro, se registra la acción en la tabla LOG_ACCIONES.

Atributos

ID: identificador secuencial asignado por el sistema.
accion: tipo de acción: INSERTAR o MODIFICAR.
tabla: nombre de la tabla sobre la que se ha realizado la acción.
registro: registro afectado por la acción.
log_f_h: fecha/hora en la que se realizó la acción.
actualiz_ce: usuario que realizó la acción.

Restricciones y Tipos de Datos

	Tipo de Dato	Nulo	Duplic.	
ID	NUMBER(10,0)	No	No	Clave Primaria. Indexado.
accion	VARCHAR2(10, CHAR)	No	Sí	
tabla	VARCHAR2(50, CHAR)	No	Sí	
registro	NUMBER(10,0)	No	Sí	
log_f_h	DATE	Sí	Sí	Indexado.
usuario_ce	NUMBER(10,0)	Sí	Sí	Indexado.

Procedimientos

OBTENER_LOG_ACCION_ POR_ID (pID, oRC)
Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente:
*SELECT * FROM LOG_ACCIONES WHERE ID = pID;*

BUSCAR_LOGS_POR_TB_Y_REG (pTabla, pRegistro_ce, oRC)
Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente:
*SELECT * FROM LOG_ACCIONES WHERE tabla = pTabla AND registro = pRegistro;*

BUSCAR_LOGS_POR_TB_Y_USUARIO (pTabla, pUsuario_ce, oRC)
Carga dentro de la variable oRC (de tipo REFCURSOR) el resultado de la sentencia SELECT siguiente:
*SELECT * FROM LOG_ACCIONES WHERE tabla = pTabla AND usuario_ce =* pUsuario_ce *;*

3.9. Guiones de pruebas

El proyecto incluye unos guiones de prueba que permite probar el funcionamiento correcto de la *Base de Datos*. Cada guión consta de una secuencia de pasos a ejecutar. Para cada paso, los guiones especifican:

- instrucciones sobre cómo ejecutar el paso
- la funcionalidad concreta que el paso permite probar
- el resultado esperado

Los guiones dejan un espacio al lado de cada paso para especificar si el resultado obtenido como consecuencia de la ejecución del paso es el esperado. También, para cada paso hay un espacio para realizar anotaciones sobre el resultado devuelto.

La imagen siguiente muestra el aspecto que tiene un guión de prueba:

GUIÓN DE PRUEBAS 8: Tabla VENTAS.

Paso nº	Descripción Paso	Funcionalidad a comprobar	Resultado Esperado	Resultado (OK, Fallado)	Observaciones
8.1	execute AGREGAR_VENTA (´´, 100, 2, 1);	El procedimiento AGREGAR_VENTA no debe agregar registros sin especificar la fecha de la venta.	ORA -01400		
8.2	execute AGREGAR_VENTA (´02/01/13´, ´´, 2, 1);	El procedimiento AGREGAR_VENTA no debe agregar registros sin especificar el coche vendido.	ORA -01400		
8.3	execute AGREGAR_VENTA (´02/01/13´, 100, ´´, 1);	El procedimiento AGREGAR_VENTA no debe agregar registros sin especificar el comercial responsable de la venta.	ORA -01400		
8.4	execute AGREGAR_VENTA (´02/01/13´, 100, 2, 1); select * from VENTAS;	El procedimiento AGREGAR_VENTA inserta una venta correctamente.	Se devuelve el registro: (100, 1, ´02/01/13´, 100, 2, Fhoy, 1)		
8.5	var RS refcursor Execute OBTENER_VENTA_POR_ID (100, :RS) print RS	El procedimiento OBTENER_VENTA_POR_ID recupera una venta correctamente.	Se devuelve el registro: (100, 1, ´02/01/13´, 100, 2, Fhoy, 1)		
8.6	var RS refcursor Execute BUSCAR_VENTAS_POR_COM_Y_RFVENT (´01/01/13´, ´05/01/13´, 2, :RS) print RS	El procedimiento BUSCAR_VENTAS_POR_COM_Y_RFVENT encuentra las ventas de forma adecuada.	Se devuelve el registro: (100, 1, ´02/01/13´, 100,		

Figura 8. Aspecto de un guión de prueba

La lista de guiones de pruebas son los siguientes:

- GUIÓN DE PRUEBAS 1: Tabla GRUPOS.
- GUIÓN DE PRUEBAS 2: Tabla VALORES_CODIGO.
- GUIÓN DE PRUEBAS 3: Tabla PERSONAS.
- GUIÓN DE PRUEBAS 4: Tabla COCHES.
- GUIÓN DE PRUEBAS 5: Tabla SERVICIOS.
- GUIÓN DE PRUEBAS 6: Tabla ASOCIACION_VALORES_CODIGO.
- GUIÓN DE PRUEBAS 7: Tabla USUARIOS.
- GUIÓN DE PRUEBAS 8: Tabla VENTAS.
- GUIÓN DE PRUEBAS 9: Tabla VALORES_ATRIB_VIRTUALES.

3.10. Integración con la *Data Warehouse*

Se contempla la posibilidad de definir un *Data Warehouse* para analizar los datos capturados mediante la *Base de Datos*. El *Data Warehouse* permitiría analizar los datos para obtener indicadores sobre la actividad de la organización cliente como por ejemplo, el tiempo medio para reparar un coche, quienes son los vendedores que más venden, en qué meses se realizan más ventas...

El alcance del proyecto de desarrollo de la *Base de Datos* no incluye una *Data Warehouse* ni su configuración, ni la importación directa de datos a una *Data Warehouse* pero sí contempla la posibilidad de poder exportar datos de la *Base de Datos* a ficheros CSV. La gestión del *Data Warehouse* deberá contemplar la importación de estos ficheros CSV al *Data Warehouse*.

Para permitir esta exportación, la *Base de Datos* dispone de unos procedimientos almacenados que extraen datos de su base de datos y los almacena en ficheros CSV en una ubicación concreta.

Estos procedimientos almacenados consultan las tablas principales de la base de datos y extraen los registros que hayan sido modificados. De esta forma, se evita extraer cada vez el conjunto total de datos de las tablas principales. Se extraen únicamente los registros que hayan sido modificados después de la fecha especificada como parámetro en la invocación de los procedimientos almacenados.

Se puede ejecutar los procedimientos almacenados de exportación en cualquier momento y con la frecuencia que se considera necesario. Si por ejemplo, se ejecutan los procedimientos almacenados de exportación cada día con fecha del día, se obtendría cada día el listado de registros modificados durante el día.

Se recomiendo la ejecución de estos procedimientos por las noches cuando el uso de la base de datos sea menor.

Ejecución, parámetros y salidas generadas

Los procedimientos almacenados de exportación pueden ser ejecutados desde *sqlplus* o pueden ser ejecutados desde una aplicación externa. Como parámetro, se especifica únicamente la fecha a partir de la cual se desea extraer registros modificados o añadidos. Cada procedimiento obtiene información de una tabla concreta y genera un fichero CSV.

Si existe ya existe un fichero con el mismo nombre que el fichero siendo generado, el fichero original es sustituido.

Nomenclatura de ficheros generados

El nombre de cada fichero contiene una referencia a la tabla desde donde se ha exportado los datos y la fecha del día en que se generó el fichero. De esta forma, mirando el nombre de un fichero exportado, se sabe de qué tabla contiene información y en qué día se generó.

Las referencias que identifican las diferentes tablas que se exportan son las siguientes:

TABLA	Referencia en nomenclatura fichero
GRUPOS	GRUPOS
VALORES_CODIGO	VALCOD
PERSONAS	PERSONAS
COCHES	COCHES
SERVICIOS	SERVICIOS
ASOCIACION_VALORES_CODIGO	AVC
USUARIOS	USUARIOS
VENTAS	VENTAS
VALORES_ATRIB_VIRTUALES	VAV

Cada nombre de fichero también incluye la cadena "_POR_ACTFH_". Esta cadena recuerda que el fichero se generó a partir de una consulta sobre el campo *actualiz_f_h*. En cada tabla, para cada registro, este campo es actualizado con la fecha y hora en que se añade o se modifica el registro.

La nomenclatura se puede representar de la forma siguiente:

Ref a tabla + "_POR_ACTFH_" + Fecha + ".CSV"

Un ejemplo podría ser: ***GRUPOS_POR_ACTFH_20130103.CSV***. A partir de este nombre, se sabe que la información del fichero fue obtenido de la tabla GRUPOS y que el fichero fue generado el 3 de enero de 2013.

Estructura de ficheros generados

Los ficheros generados están en formato texto y respetan la estructura CSV (*Comma Separated Value*). Como separador de campo se, se ha usado el símbolo "|".

Los ficheros contienen encabezados con los mismos nombres que las columnas de la tabla desde donde se ha generado los datos. La única excepción es que las columnas de nombre "ID" aparecen en los ficheros CSV con el nombre "-ID". Esto es para evitar que Excel confunda estos ficheros con ficheros de formato SYLK (este formato se caracteriza por el comienzo de ficheros con la cadena "ID").

Ubicación en donde se generan los ficheros

Los procedimientos almacenados de exportación consultan la variable D_TEMP de la base de datos. Esta variable debe ser definida antes de usar estos procedimientos. El valor de esta variable define la ubicación en la que se almacenarán los ficheros. Esta ubicación debe ser creada antes de usar estos procedimientos.

Para crear esta variable y asignarle un valor, se debe ejecutar en sqlplus (con usuarios SYSTEM) la sentencia como la siguiente:

```
SQL>create or replace directory D_TEMP as 'D:\temp'
```

Se puede ejecutar la sentencia anterior con la ubicación que se desea. A continuación se debe otorgar acceso a la variable al usuario Oracle CONCESIONARI:

```
SQL>grant read, write on directory D_TEMP to CONCESIONARI;
```

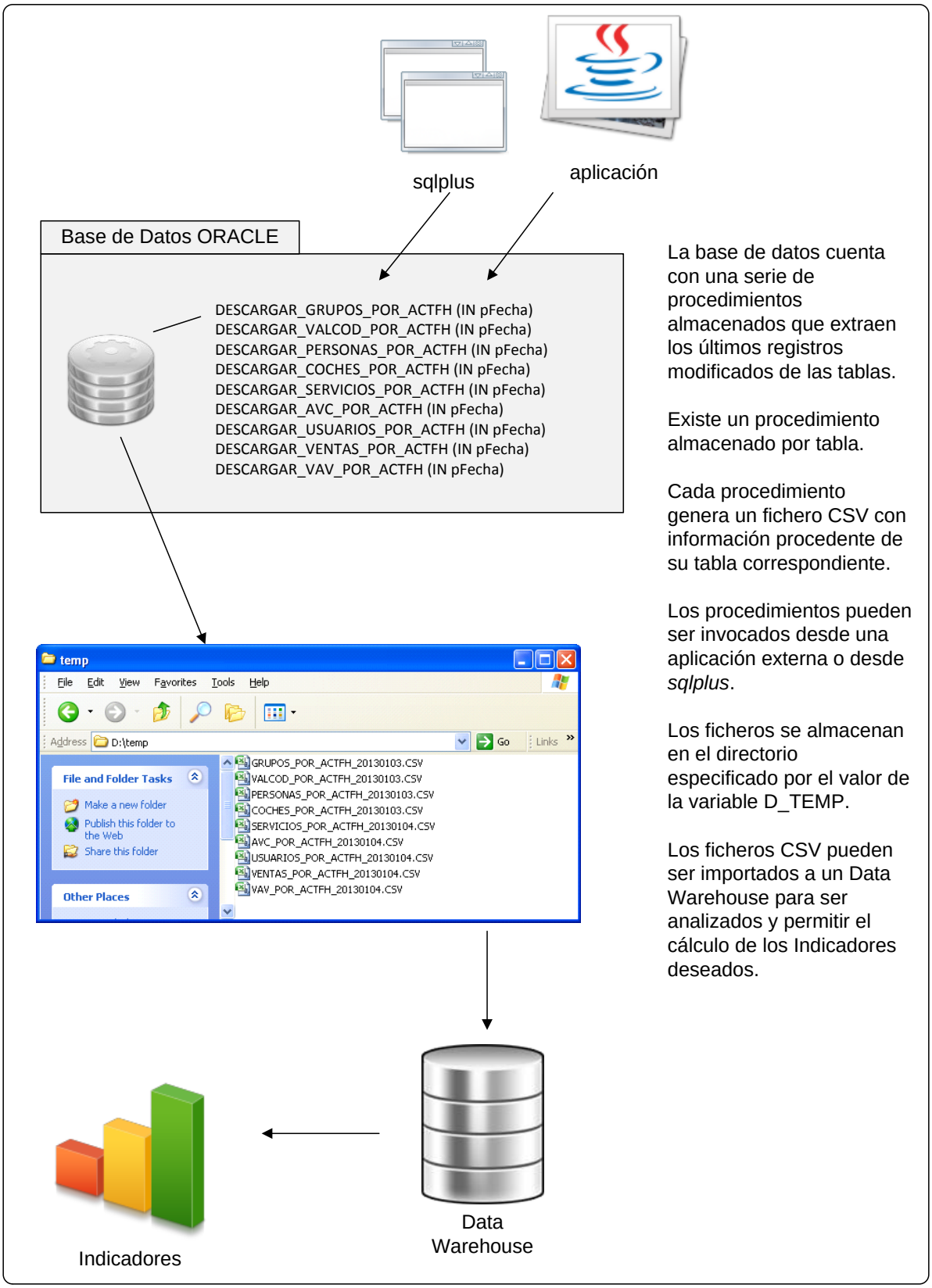
Consideración técnica especial: uso del paquete Oracle UTL_FILE

Los procedimientos almacenados de las exportaciones, requieren que el usuario Oracle que los ejecute tenga el privilegio de ejecutar sobre el paquete ***UTL_FILE***. Este paquete es proporcionado por Oracle. La versión *Oracle Database 10g Express Edition* incluye este paquete. Para que sea utilizable por el usuario CONCESIONARI, se le debe otorgar acceso mediante el usuario SYS:

```
SQL>GRANT EXECUTE ON UTL_FILE TO public;
```

Diagrama gráfica de la exportación de ficheros

A continuación se resume de forma gráfica el proceso de generación de ficheros CSV a ser importados en una *Data Warehouse*.



La base de datos cuenta con una serie de procedimientos almacenados que extraen los últimos registros modificados de las tablas.

Existe un procedimiento almacenado por tabla.

Cada procedimiento genera un fichero CSV con información procedente de su tabla correspondiente.

Los procedimientos pueden ser invocados desde una aplicación externa o desde *sqlplus*.

Los ficheros se almacenan en el directorio especificado por el valor de la variable D_TEMP.

Los ficheros CSV pueden ser importados a un Data Warehouse para ser analizados y permitir el cálculo de los Indicadores deseados.

Figura 9. Proceso integración con un Data Warehouse

3.11. Valoración económica del proyecto

A continuación se hará la estimación económica a partir de la duración estimada de las tareas planificadas y se comparará con la duración real que ha tardado el proyecto. Por último se comprará la estimación real con las métricas LOC y Puntos de Función [2].

En el coste del desarrollo de un software, predomina el coste del personal técnico (director de proyecto, analistas y programadores). Por tanto, a la hora de hacer la estimación de costes, debemos centrarnos principalmente en calcular el esfuerzo (en personas-hora) que requiere el desarrollo.

La mejor manera de estimar el coste de un software es partiendo de los costes que conllevaron proyectos semejantes en el pasado.

Para este proyecto, se conoce el tiempo presupuestado y el tiempo que real invertido.

A la hora de realizar el Plan de Trabajo, se dividió el proyecto en tareas y ase asignó una duración aproximada a cada tarea. Según esta división de tareas y asignación de duraciones, se llegó al total de 47 jornadas de dedicación. El proyecto *se* desarrolló en paralelo con otros proyectos y por tanto, se supuso jornadas de menos de 8 horas. Por experiencias pasadas, se esperó dedicar aproximadamente 2 horas por jornada. Por tanto, se presupuestó 47 x 2 = 94 horas.

La dedicación aproximada ha sido la siguiente:

- PAC1: 20 horas
- PAC 2: 16 horas
- PAC 3: 20 horas
- Restante (dedicación durante diciembre y enero): 80 horas.

La dedicación real se acerca a las 140 horas. Por tanto, vemos que se ha tenido que dedicar más tiempo de lo estimado inicialmente.

Afortunadamente para este proyecto, los roles de jefe de proyecto, analista y programador los desempeñaron la misma persona y las tareas se han podido desarrollar secuencialmente. De no haber sido así, los retrasos de algunas actividades podrían haber implicado el mal gasto de horas de los programadores y las pérdidas habrían sido mayores.

Es interesante comparar estos datos con los ratios de productividad estándares propuestos por Caper T. Jones (en su libro Programming productivity, 1986).

Para facilitar la comparación con estudios estándares, supongamos que trabajé jornadas de 8 horas para desarrollar el software. La cantidad de jornadas habrían sido: 130horas / (8horas/jornada) = **16 jornadas**. La cantidad de líneas de código generados (LOC) son aproximadamente **1600**. Por tanto, la productividad teórica aproximada ha sido:

1600 LOC / 16 jornadas = **100 LOC persona-día**.

Según los estudios de Caper T. Jones, contando el análisis de requerimientos, el diseño, la codificación (en Cobol) y pruebas, la productividad anual media es de **2.500 LOC por persona-año**. Eso equivale a 2.500 / 220 (jornadas al año) = **11 LOC persona-día**.

Otra métrica muy conocida de estimación de esfuerzo es la métrica ***Puntos de Función***. También fue diseñado por Caper T. Jones. Esta métrica intenta calificar el esfuerzo a partir de las funcionalidades en lugar de la cantidad de líneas de código.

Aplicando los pasos de la métrica, calculo los *Puntos de Función*:

Puntos de Función sin Ajustar (TUFP)									
Tipo	Descripción	Complejidad						Total	
		Simple		Medio		Complejo			
EI	External Inputs	16	3	3	4	0	6	60	(Procedimientos de insert y update)
EO	External Outputs	10	4	0	5	0	7	40	Ficheros CSV
LIF	Logical Interface File	10	7	0	10	0	15	70	(Tablas)
EIF	External Interface File	0	5	0	7	0	10	0	(Tablas, pero no mantenidas por la aplicación)
EQ	External Queries	30	3	0	4	0	6	90	Select sin datos calculados
Total Puntos de Función sin Ajustar (TUFP):								260	

Grado Total de Influencia (TDI)		
ID	Característica del Sistema	Total
C1	Comunicació de dades	0
C2	Funcions distribuïdes	0
C3	Rendiment	0
C4	Configuració molt utilitzada	0
C5	Freqüència de transaccions	2
C6	Entrada on-line de dades	2
C7	Eficiència d'usuari final	1
C8	Actualització on-line	2
C9	Processos complexos	0
C10	Reusabilitat	0
C11	Facilitat d'instal·lació	1
C12	Facilitat d'operació	1
C13	Instal·lació múltiple	0
C14	Facilitat de canvis	2.5
Total		11.5

Ajuste por la complejidad (PCA):
PCA
= 0.65 + (0,01 x TDI)
PCA
= 0.765

Puntos de Función (FP):
FP = TUFP x PCA
FP = 198.9

Jones estimó una media de 1,6 horas por punto de función. Según el total de ***Puntos de Función*** estimados, el desarrollo debería haber llevado:

198.9 PF x 1.65 horas/PF = **328.19 horas**.

De ambos resultados, vemos que el esfuerzo de desarrollo invertido es mucho menor de lo estimado por las métricas.

La diferencia entre lo estimado mediante las métricas anteriores y la realidad se debe seguramente a los factores siguientes:

- Las métricas y estándares anteriores fueron diseñados en los años 80. Desde entonces, la informática y la forma de programar han variado considerablemente y por tanto, los estándares de productividad deben ser actualizados. Hoy en día no existe una propuesta universalmente acordado.
- Las métricas ***LOC*** y ***Puntos de Función*** han sido diseñadas para aplicaciones propiamente dichas (escritas en Cobol, por ejemplo, y posiblemente con acceso a alguna base de datos pequeña). Estas métricas no asignan mucho peso al esfuerzo dedicado al desarrollo de la base de datos. Es decir, estamos usando métricas diseñados para ser aplicados al producto final (la aplicación y la base de datos) a estimar el esfuerzo de únicamente el desarrollo de la base de datos, que además, estos métodos tienden a infravalorar.
- La complejidad del código de los procedimientos y funciones almacenados es baja y además, de pocas líneas de código. Las métricas estándares propuestos fueron pensados para aplicaciones mucho más grandes y reales.

También hay que tener en cuenta que los resultados de la aplicación de las métricas son orientativos. No proporcionan valores exactos. Para estimar el esfuerzo requerido con ciertas garantías, es imprescindible poder comprar el proyecto con proyectos semejantes ya desarrollados anteriormente.

Para este proyecto, el sobre coste ha sido significativo aunque asumible. Para desarrollos futuros similares, sabremos que requerimos un esfuerzo aproximadamente de 12 horas por tabla, aunque este valor podría bajar a medida que aumente la experiencia en el uso de PL/SQL.

Para finalizar la estimación económica, habría que multiplicar el total de horas invertidas por el precio que el analista / programador cuesta por hora. Suponiendo una tarifa media de 50 € por hora, el desarrollo del proyecto habría costado el importe siguiente: 120 horas x 50 € / hora = **6000 €**.

3.12. Conclusiones

He podido seguir el plan de proyecto, aunque con un desfase importante con respecto de la planificación. La atención que exigía la puesta en producción de otros proyectos ha provocado un retraso importante en el desarrollo del PFC. A final de año he podido coger vacaciones y poner el PFC al día. Este retraso fue considerado en el plan de riesgos. El plan de contingencia para este riesgo consistía en coger vacaciones las 3 semanas antes de la entrega del PFC.

El producto desarrollado cumple con los objetivos iniciales planteados.

A continuación, comento algunas lecciones aprendidas:

- Se simplifica mucho el acceso a la Base de Datos si se hace a través de procedimientos y funciones almacenados.
- La identificación de tareas lo más concretas posible y la estimación realista de la duración de cada tarea es importantísima para que la planificación del proyecto sea acertada.
- Para una estimación económica realista, se debe poder comparar el proyecto siendo presupuestado con proyectos anteriores. La experiencia anterior ayuda a aplicar correctamente las metodologías estándares como el de *Puntos de Función* [2].
- La documentación puede llevar más tiempo que la codificación.

4. Glosario

Base de Datos: En este documento, el término *Base de Datos*, refiere al producto base de datos que se está desarrollando. Incluye las tablas, las relaciones entre las tablas, las claves, los índices, las secuencias, los *triggers*, los procedimientos y las funciones.

Alcance inicial: conjunto de requerimientos genéricos que se definen durante la definición del proyecto.

Alcance detallado: conjunto de requerimientos concretos y detallados. Se obtienen a partir del refinamiento de los requerimientos genéricos del alcance inicial.

Información de referencia: información categorizada para caracterizar la información de actividad. Los municipios, las provincias, los modelos de coches son ejemplos de información de referencia.

Información de actividad: información registrada como resultado de la realización de la actividad de la empresa. Las ventas, reparaciones y revisiones son ejemplos de información de actividad.

Valor de código: son los valores concretos de municipios, de provincias de modelos de coches, de extras o características.... Es la unidad de la información de referencia.

Grupo: categoría o entidad conceptual en la que se clasifican los *valores de código*. Por ejemplo, Palma de Mallorca es *valor de código* del grupo *municipios*.

Aplicación: En este documento, el término *aplicación*, refiere a la aplicación a la que la *Base de Datos* ofrece persistencia.

Data Warehouse: Almacén de datos.

Software a medida: en este documento, se ha denominado *software a medida* al conjunto aplicación y *Bases de Datos* que el cliente necesita para poder registrar su actividad.

5. Bibliografía

[1]: José Ramón Rodríguez, Pere Mariné Jové, “Gestió de projectes”, UOC, Barcelona, 2010.

[2]: Miquel Barceló García, Joan Antoni Pastor i Collado “Gestión de organizaciones y proyectos informáticos”, UOC, Barcelona, 2004.

[3]: Jordi Fernández González, Jordi Pradel i Miquel, “Desenvolupament del programari orientat a objectes”, UOC, Barcelona, 2005.

[4]: Abraham Silberschatz, “Fundamentos de Bases de datos”, McGraw Hill, Madrid 1998.

[5]: Jaume Sistac Planas, “Bases de dades II”, UOC, Barcelona, 2004.

6. Anexos

En su totalidad, se entregan los productos siguientes:

- Esta Memoria
- La presentación que resume la Memoria
- Fichero de texto con las sentencias SQL cuya ejecución permiten la construcción de la Base de Datos.
- Los guiones de prueba.